10 YEARS OF UNIVERSITY RECOGNITION
20 YEARS OF ACADEMIC EXCELLENCE

REVA UNIVERSITY
Bengaluru, India

School of Electronics and Communication Engineering

Programme
B.Tech. in ECM

Laboratory Manual
B20EP0701

Data Analytics Using R

Semester-VII
2020-24

RUKMINI EDUCATIONAL
Charitable Trust

www.reva.edu.in

Vision of the University

''REVA University aspires to become an innovative university by developing excellent human resources with leadership qualities, ethical and moral values, research culture and innovative skills through higher education of global standards"

Mission of the University

- To create excellent infrastructure facilities and state-of-the-art laboratories and incubation centres
- To provide student-centric learning environment through innovative pedagogy and education reforms
- To encourage research and entrepreneurship through collaborations and extension activities
- To promote industry-institute partnerships and share knowledge for innovation and development
- To organize society development programs for knowledge enhancement in thrust areas
- To enhance leadership qualities among the youth and enrich personality traits, promote patriotism and moral values.

Vision of the School

The School of Electronics and Communication Engineering is envisioned to be a leading centre of higher learning with academic excellence in the field of electronics and communication engineering blended by research and innovation in tune with changing technological and cultural challenges supported with leadership qualities, ethical and moral values.

Mission of the School

- Establish a unique learning environment to enable the students to face the challenges in the field of Electronics and Communication Engineering and explore multidisciplinary which serve the societal requirements.
- Create state-of-the-art laboratories, resources, and exposure to the current industrial trends to enable students to develop skills for solving complex technological problems of current times and provide a framework for promoting collaborative and multidisciplinary activities.
- Promote the establishment of Centres of Excellence in niche technology areas to nurture the spirit of innovation and creativity among faculty and students.
- Offer ethical and moral value-based education by promoting activities which inculcate the leadership qualities, patriotism and set high benchmarks to serve the society

Program Educational Objectives (PEOs)

The Program Educational Objectives of B. Tech in Electronics and Computer Engineering are as follows:

PEO-1: Have successful professional career in industry, government, and software organization as innovative engineers

PEO-2: Successfully solve engineering problems related to Electronics and Computer Engineering by communicating effectively either as a team or as a team member and lead the team

PEO-3: Pursue higher studies and have an attitude of lifelong learning through cultural, technical and outreach activities

PEO-4: Serve the society regionally, globally and will take up entrepreneurship for the growth of the economy and generate employment

Program Outcomes (POs)

On successful completion of the program, the graduates of B. Tech. (Electronics and Computer Engineering) program will be able to

- **PO-1: Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals for the solution of complex problems in Electronics and computer Engineering.
- **PO-2: Problem analysis:** Identify, formulate, research literature, and analyze engineering problems to arrive at substantiated conclusions using first principles of mathematics, natural, and engineering sciences.
- **PO-3: Design/development of solutions:** Design solutions for complex engineering problems and design system components, processes to meet the specifications with consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- **PO-4: Conduct investigations of complex problems:** Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **PO-5: Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- **PO-6: The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- **PO-7: Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- **PO-8: Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice
- **PO-9: Individual and teamwork:** Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
- **PO-10: Communication:** Communicate effectively with the engineering community and with society at large. Be able to comprehend and write effective reports documentation. Make effective presentations and give and receive clear instructions.
- **PO-11: Project management and finance:** Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments.
- **PO-12: Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Programme Specific Outcomes (PSOs)

On successful completion of the program, the graduates of B. Tech. (Electronics and Computer Engineering) program will be able to

- **PSO-1:** Isolate and solve complex problems in the domains of Electronics and Computer Engineering using latest hardware and software tools and technologies, along with analytical and managerial skills to arrive at cost effective and optimum solutions either independently or as a team.

- **PSO-2:** Implant the capacity to apply the concepts of electronics, data analytics, computer networks, cloud computing, artificial intelligence, and machine learning, etc., in the design, development of hardware and software for the application engineering lifecycle systems.

- **PSO-3:** Design, develop and build electronics and software systems to solve real life industry problems using modern tools and techniques.

<table>
<tr><td>Course Title</td><td colspan="4">Data Analytics using R</td><td colspan="2">Course Type</td><td colspan="2">HC</td></tr>
<tr><td>Course Code</td><td>B20EP0701</td><td>Credits</td><td colspan="2">3</td><td colspan="2">Class</td><td colspan="2">VII Semester</td></tr>
<tr><td rowspan="5">Course Structure</td><td>LTP</td><td>Credits</td><td>Contac tHours</td><td>Work Load</td><td colspan="2" rowspan="2">Total Number ofClasses Per Semester</td><td colspan="2" rowspan="2">Assessmen t Weightage</td></tr>
<tr><td>Lecture</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>Tutorial</td><td>1</td><td>2</td><td>2</td><td rowspan="2">Theory</td><td rowspan="2">Practical</td><td rowspan="2">C I E</td><td rowspan="2">SEE</td></tr>
<tr><td>Practical</td><td>1</td><td>2</td><td>2</td></tr>
<tr><td>Total</td><td>3</td><td>5</td><td>5</td><td>39</td><td>26</td><td>50%</td><td>50 %</td></tr>
</table>

COURSE OVERVIEW:

The course provides students with some knowledge on the basic principles of machine learning, which is the study of computer algorithms that can improve automatically through experience and using data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

COURSE OBJECTIVES:

The objectives of this course are:

1. To introduce different functions in R, how to read data into R, accessing R

Packages,

2. Writing R functions, debugging, and organizing data using R functions.

3. Cover the Basics of statistical data analysis with examples.

4. To introduce the concepts involving collecting, compiling and visualize

data using statistical functions.

5. To give insight into exploratory data analysis using R

6. To introduce regression and perspective analysis

COURSE OUTCOMES (COs):

On successful completion of this course; the student shall be able to

CO#	Course Outcomes	POs	PSOs
CO1	Solve the analytical problems using R	1,2,3,4,5	1,2,3
CO2	Develop competency in the R programming language and several datarelated R libraries	1,2,3,4,5,10	1,2,3
CO3	Analyze real time data effectively using visualizations in R	1,2,3,4,5,10	1,2,3
CO4	Import, export and manipulate data and produce statistical summaries ofcontinuous and categorical data in R	1,2,3,4,5,10	1,2,3
CO5	Perform exploratory data analysis using R	1,2,3,4,5	1,2,3
CO6	Develop data visualizations with the ggplot package.	1,2,3,4,5	1,2,3

BLOOM LEVEL OF THE COURSE OUTCOMES:

CO#	Bloom's Level					
	Remember (L1)	**Understand(L2)**	**Apply (L3)**	**Analyze (L4)**	**Evaluate (L5)**	**Create (L6)**
CO1	✓	✓	✓	✓		
CO2	✓	✓	✓	✓		
CO3	✓	✓	✓	✓		
CO4	✓	✓	✓	✓		
CO5	✓	✓	✓	✓		
CO6	✓	✓	✓	✓		

COURSE ARTICULATION MATRIX:

Mapping of Course Outcomes with Program Outcomes

POS / COs	**PO1**	**P2**	**PO3**	**PO4**	**PO5**	**PO6**	**P7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
CO1	3	3	2	1	2								3	2	1
CO2	3	3	3	1	2								3	2	1
CO3	3	3	2	1	2								3	2	1
CO4	3	3	3	1	2								2	2	1
CO5	3	3	2	1	3								3	2	1
CO6	3	3	3	1	3								3	2	1

Note: 1-Low, 2-Medium, 3-High

Sl. No.	Name of the Practice Session	Tools and Techniques	Expected Skill /Ability
1	**R AS CALCULATOR APPLICATION** a. Using with and without R objects on console b. Using mathematical functions on console c. Write an R script, to create R objects for calculator application and save in a specified location in disk.	R version 4.3.0	Understand basics of Rscripting
2	**DESCRIPTIVE STATISTICS IN R** a. Write an R script to find basic descriptive statistics using summary, str, quartile function on mtcars & carsdatasets. b. Write an R script to find subset of dataset by using subset (), aggregate () functions on iris dataset.	R version 4.3.0	Understand basics of Rscripting
3	**READING AND WRITING DIFFERENT TYPESOF DATASETS** a. Reading different types of data sets (.txt, .csv) from web and disk and writing in file in specific disklocation. b. Reading Excel data sheet in R. Reading XML dataset in R.	R version 4.3.0	Understand basics of Rscripting
4	**VISUALIZATIONS** a. Find the data distributions using box and scatter plot. b. Find the outliers using plot. c. Plot the histogram, bar chart and pie chart on sampledata.	R version 4.3.0	Understand data visualization using R
5	**CORRELATION AND COVARIANCE** a. Find the correlation matrix. b. Plot the correlation plot on dataset and visualize givingan overview of relationships among data on iris data. c. Analysis of covariance: variance (ANOVA) if datahave categorical variables on iris data.	R version 4.3.0	Understand plotcorrelation
6	**REGRESSION MODEL** Import data from web storage. Name the dataset and nowdo Logistic Regression to find out relation between variables that are affecting the admission of a student in an institute based on his or her GRE score, GPA obtained and rank of the student. Also check if the model is fit or Not. Require (foreign), require (MASS).	R version 4.3.0	Understand regressiontechniques
7	**MULTIPLE REGRESSION MODEL** Apply multiple regressions if data have a continuousindependent variable. Apply on above dataset.	R version 4.3.0	Understand multiple regression

8	**REGRESSION MODEL FOR PREDICTION** Apply regression Model techniques to predict the data onabove dataset.	R version 4.3.0	Understand Proof ofwork algorithm
9	**CLASSIFICATION MODEL** a. Install relevant package for classification. b. Choose classifier for classification problem. c. Evaluate the performance of classifier.	R version 4.3.0	Understand bit coinmining
10	**CLUSTERING MODEL** a. Clustering algorithms for unsupervised classification. b. Plot the cluster data using R visualizations.	R version 4.3.0	Understand Consensusalgorithm

Contents

Sl.No	Name of Program	Page No
1	R AS CALCULATOR APPLICATION a. Using with and without R objects on console. b. Using mathematical functions on console. c. Write an R script, to create R objects for calculator application and save in a specified location in disk.	19-21
2	DESCRIPTIVE STATISTICS IN R a.Write an R script to find basic descriptive statistics using summary, str, quartile function on mtcars & cars datasets. b. Write an R script to find subset of dataset by using subset (), aggregate () functions on iris dataset	22-24
3	READING AND WRITING DIFFERENT TYPES OF DATASETS a. Reading different types of data sets (.txt, .csv) from web and disk and writing in file in specific disk location. b. Reading Excel data sheet in R. Reading XML dataset in R.	25-26
4	VISUALIZATIONS a. Find the data distributions using box and scatter plot. b. Find the outliers using plot. c. Plot the histogram, bar chart and pie chart on sample data	27-28
5	5. CORRELATION AND COVARIANCE d. Find the correlation matrix. e. Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris data. f. Analysis of covariance: variance (ANOVA) if data have categorical variables on iris data.	29-30

6	REGRESSION MODEL Import data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables that are affecting the admission of a student in an institute based on his or her GRE score, GPA obtained and rank of the student. Also check if the model is fit or Not. Require (foreign), require (MASS).	31-32
7	MULTIPLE REGRESSION MODEL Apply multiple regressions if data have a continuous independent variable. Apply on above dataset	33-34
8	REGRESSION MODEL FOR PREDICTION Apply regression Model techniques to predict the data on above dataset.	35-36
9	CLASSIFICATION MODEL a. Install relevant package for classification. b. Choose classifier for classification problem. c. Evaluate the performance of classifier.	37-38
10	CLUSTERING MODEL a. Clustering algorithms for unsupervised classification. b. Plot the cluster data using R visualizations	39-40

RStudio is an integrated development environment (IDE)

RStudio is an integrated development environment (IDE) designed specifically for R, a programming language and environment primarily used for statistical computing and data analysis. RStudio provides a user-friendly interface and a set of tools that make it easier to write, run, and manage R code efficiently. Let's delve into the features and components of RStudio:

1. **Script Editor:** The central area of the RStudio interface is the script editor. Here, you can create, edit, and execute R scripts. It offers features such as syntax highlighting, code completion, and automatic indentation to enhance your coding experience.

2. **Console:** The console is where R code is executed interactively. You can type commands directly into the console and see the immediate results. It's a handy tool for testing code snippets or experimenting with data.

3. **Environment and History:** RStudio provides panes for tracking your environment (i.e., thevariables and objects currently in memory) and your command history. This helps you keep track of objects, datasets, and functions you've used in your session.

4. **File Browser:** RStudio includes a file browser to navigate your file system, open scripts, andorganize your project files. It also allows you to set your working directory, which is crucial formanaging data and file paths in your R projects.

5. **Plots and Viewer:** You can generate plots and visualizations directly within RStudio. The Plots pane displays plots, and the Viewer pane renders web-based content such as HTML and Shiny apps. These features are essential for data visualization and interactive reporting.

6. **Packages and Help:** RStudio has dedicated panes for managing R packages. You can install, update, and load packages with ease. The Help pane provides access to R documentation andpackages' documentation, making it convenient to look up functions and their usage.

7. **Git and Version Control:** RStudio integrates with version control systems like Git, allowingyou to manage code versions and collaborate with others on projects. You can commit changes, view diffs, and access Git features directly from the IDE.

8. RMarkdown and Notebooks: RStudio supports RMarkdown, a format that combines R codewith text and generates dynamic documents. You can create reports, presentations, and dashboards using RMarkdown. Additionally, RStudio supports Jupyter Notebooks, which are popular for interactive data analysis and sharing.

9. Customization: RStudio is highly customizable. You can configure the appearance, layout, and behavior of the IDE to suit your preferences. You can also define code snippets, keyboardshortcuts, and templates to streamline your workflow.

10. Project Management: RStudio encourages good project organization by allowing you to create and manage R projects. Projects have their own working directories, making it easy tokeep data, scripts, and documents organized and separate from each other.

11. Integrated Terminal: RStudio includes a built-in terminal, which can be handy for runningsystem commands or using R's command-line interface.

12. Debugging Tools: The IDE offers debugging capabilities, such as setting breakpoints, stepping through code, and inspecting variable values, which are essential for troubleshooting and understanding code behavior.

13. Shiny Apps Development: If you're interested in creating interactive web applications in R, RStudio provides a platform for developing Shiny apps with ease.

RStudio is a versatile and powerful IDE that streamlines the R coding and data analysis workflow. It's a preferred choice for R users due to its extensive features, user-friendly interface, and strong community support. Whether you're a beginner or an experienced data analyst, RStudio can significantly enhance your productivity when working with R.

Introduction to R

1.1 What is R?

R is a powerful, open-source programming language and environment specifically designed for statistical analysis and data visualization. It was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, in the early 1990s. R has gained immense popularity among data scientists, statisticians, and researchers due to its flexibility, extensive statistical libraries, and active user community.

1.2 Why Use R?

Statistical Analysis: R provides a wide range of statistical and graphical techniques for data analysis, making it a preferred choice for statistical odelling.

Data Visualization: It offers advanced data visualization libraries like ggplot2, allowing you tocreate informative and visually appealing plots and charts.

Open Source: R is open-source software, which means it's free to use and has a vast community of users and developers continually contributing to its growth

Extensibility: You can extend R's functionality by installing and using packages, which are collections of functions and datasets created by the R community.

1.3 Installing R

To install R, follow these steps:

Go to the R Project website and select a CRAN (Comprehensive R Archive Network) mirror sitenear you.

Download and install R for your operating system (Windows, macOS, or Linux).

1.4 RStudio IDE

RStudio is a popular integrated development environment (IDE) for R. It provides a user-friendly interface for writing R code, managing files, and visualizing data. You can download RStudio from the RStudio website.

2. Getting Started

2.1 Basic Arithmetic

In R, you can perform basic arithmetic operations like addition, subtraction, multiplication, and division using operators:

```
# Addition3+4

#Subtraction 10 - 5

#Multiplication2 * 6

#Division on20 /4
```

2.2 Variables

Variables are used to store data values. In R, you can assign values to variables using the <- or = operator:

```
# assigning values to variables

x <- 5

y <- 10
# using variables in calculations
Result <- x + y
```

2.3 Data Types

R supports several data types, including integers, numerics (decimal numbers), character strings, logical (TRUE/FALSE), and more. You can check the data type of a variable using the class() function:

```
# Data types
num_var <-
5.5int_var <-
10L
char_var <- "Hello,
R!"logical_var <-
TRUE

# Checking data
types
class(num_var)
class(int_var)
class(char_var)
class(logical_var)
```

2.4 Comments

Comments in R are preceded by a # symbol. They are used to add explanatory notes to your code and are not executed:

```
# This is a comment
# Comments help explain code
```

3. Data Structures

3.1 Vectors

A vector is a one-dimensional data structure in R that can hold elements of the same data type. You can create vectors using the c () function:

```
# Creating vectors
numeric_vector <- c(1, 2, 3, 4, 5)
char_vector <- c("apple", "banana", "cherry")
```

3.2 Matrices

A matrix is a two-dimensional data structure in R with rows and columns. You can creatematrices using the matrix () function:

```
# creating matrices
Mat <- matrix (data = 1:12, nrow = 3, ncol = 4)
```

3.3 Data Frames

A data frame is a two-dimensional data structure that can hold different data types. It is similar to a spreadsheet or a table in a database. You can create data frames using the data.frame() function:

```
# Creating data frames
df <- data.frame(
  Name = c("Alice", "Bob",
  "Charlie"),Age = c(25, 30, 22),
  Grade = c("A", "B", "C")
)
```

3.4 Lists

A list is a versatile data structure in R that can hold elements of different data types. You can create lists using the list() function:

```
# Creating lists
my_list <- list(1, "apple", TRUE, c(2, 4, 6))
```

4. Working with Data

4.1 Importing Data

One of the fundamental tasks in data analysis is importing data into R. You can import data from various sources, such as CSV files, Excel spreadsheets, databases, or web APIs. Common functions for importing data include `read.csv()`, `read.table()`, and specialized functions for different file formats. For example:

```
# Importing data from a CSV file
my_data <- read.csv("data.csv")
```

4.2 Data Inspection

Once you have imported data into R, it's essential to inspect it to understand its structure and contents. Various functions help you with data inspection:

head (): View the first few rows of a data frame.

tail (): View the last few rows of a data frame.

Str (): Display the structure of the data, including data types.

summary(): Generate summary statistics for numeric variables.

For example:

```
# Viewing the first few rows of a data frame
head(my_data)

# Checking the structure of the data
str(my_data)

# generating summary
statisticssummary (my_data)
```

Data inspection helps you gain insights into your dataset and decide how to preprocess and analyze it effectively.

4.3 Data Manipulation

Data manipulation is a crucial step in data analysis. In R, you can manipulate data using variousfunctions and techniques. Some common operations include:

Sub setting data: Selecting specific rows or columns from a dataset.
Filtering: Extracting rows that meet certain conditions.

Sorting: Arranging data in a specific order.
Transforming variables: Creating new variables or modifying existing ones.
Aggregating data: Calculating summary statistics for groups of data.

4.4 Data Visualization

Data visualization is a powerful way to communicate insights from your data effectively. R offers various packages for data visualization, with **ggplot2** being one of the most popular. With ggplot2, you can create a wide range of visualizations, including scatter plots, bar charts, histograms, and more.

For example:

```
# Creating a scatter plot with ggplot2
library(ggplot2)

ggplot(data = my_data, aes(x = Age, y = Salary)) +
  geom_point()
```

This section will introduce you to the basics of data visualization in R and how to create informative and visually appealing plots.

5. Control Structures

5.1 Conditional Statements (if-else)

Control structures allow you to control the flow of your R programs based on conditions. In R, you can use conditional statements like `if`, `else if`, and `else` to make decisions. For example:

```
# Simple if-else statementx <- 10

if (x > 5) {
  print("x is greater than 5")
} else {
  print("x is not greater than 5")
}
```

Conditional statements help you write programs that can adapt to different situations and make decisions based on specific criteria.

5.2 Loops (for, while)

Loops are used in programming to repeat a set of instructions multiple times. R supports two primary types of loops:

For loops: These are used when you know in advance how many times you want to repeat a block of code. For example, iterating over a sequence of numbers.

```
# Example of a for loopfor (i in 1:5) {
  print(paste("Iteration", i))
}
```

While loops: These loops continue executing as long as a specified condition is met.# Example of a while loop

```
x <- 1
while (x <= 5) {
  print(paste("Iteration",
  x))x <- x + 1
}
```

Loops are essential for automating repetitive tasks and processing data iteratively.

These sections provide you with essential skills for working with data in R, from importing andinspecting data to manipulating and visualizing it. Control structures, like conditional statements and loops, allow you to create more dynamic and responsive programs, which are critical in data analysis and manipulation.

6. Packages

In R programming, packages are collections of functions, data sets, and documentation bundled together for specific purposes. They extend the core functionality of R by adding new capabilities and features. Packages are a fundamental part of the R ecosystem and are essential for performing a wide range of tasks, from statistical analysis to data visualization and more

Here are key points to understand about packages in R:

1. Installing Packages:

To use a package, you first need to install it. This is typically done once using the **`install.packages("package_name")`** function, where **`"package_name"`** is the name of the package you want to install. For example, to install the popular package `ggplot2`, you would run **`install.packages("ggplot2")`.**

2. Loading Packages:

Once a package is installed, you can load it into your R session using the `library**("package_name")**` or `require**("package_name")**` function. Loading a package makesits functions and datasets available for use. For example, to load the `ggplot2` package, you would run `library(ggplot2)`.

3. Listing Installed Packages:

You can list all the packages currently installed on your system with the

`installed. Packages () ` function. This function provides information about each package, including its version and dependencies.

4. Using Package Functions:

- After loading a package, you can use its functions by calling them directly. For example, ifyou've loaded the `ggplot2` package, you can use functions like `ggplot()`, `aes()`, and `geom_point()` for creating plots.

5. Searching for Packages:

- The CRAN (Comprehensive R Archive Network) website is a central repository for R packages. You can search for packages on the CRAN website to discover new packages that may be useful for your work. There are also package management tools and websites, like `CRAN Task Views` and `CRAN berries`, that can help you find packages.

6. Updating Packages:

- Packages are actively maintained, and updates may include bug fixes, new features, or compatibility improvements. You can update installed packages with the **`update.packages()`** function.

7. Creating Your Own Packages:

- If you have functions or code that you want to reuse across projects, you can create your own R packages. This allows you to package your code, documentation, and data together for distribution and sharing with others.

8. Dependencies:

- Many packages depend on other packages to function correctly. R automatically installs and loads these dependencies when you install and load a package. Managing dependencies is crucial to ensure that your R environment is stable and functions as expected.

9. Documentation:

- Packages typically come with documentation that includes help files for functions and descriptions of data sets. You can access this documentation using the `?` or `help()` functions.

10. Contributing to Packages:

- R is an open-source language, and many packages are open source as well. You can contribute to existing packages or create your own packages to share your work with the R community.

packages in R are essential for extending the functionality of the language and allow you to tap into a vast ecosystem of tools and resources. They make it possible to leverage the work of other R users and developers, saving you time and effort when working on data analysis, statistical modeling, data visualization, and many other tasks.

1. R AS CALCULATOR APPLICATION

d. Using with and without R objects on console.
e. Using mathematical functions on console.
f. Write an R script, to create R objects for calculator application and save in a specified location in disk.

b) Using mathematical functions on console.

```
# Define a function to perform
additionadd <- function (x, y) {
 return (x + y)
}

# Define a function to perform
subtractionsubtract <- function (x, y)
{
 return (x - y)
}

# Define a function to perform multiplication
multiply <- function (x, y) {
 return (x * y)
}

# Define a function to perform
divisiondivide <- function (x, y) {
 if (y! = 0) {
  return (x / y)
 } else {
  return ("Cannot divide by zero!")
 }
}

# Main
program
while
(TRUE) {
```

```
cat ("Select an
operation:\n")cat ("1.
Add\n")
cat ("2. Subtract\n")
cat ("3. Multiply\n")
cat ("4. Divide\n")
cat ("5. Exit\n")

choice <- as.integer(readline("Enter your

choice: "))if (choice == 5) {
 cat("Exiting the
 calculator.\n")break
}

num1 <- as.numeric (readline ("Enter the first number:
")) num2 <- as.numeric (readline ("Enter the second
number: "))


result <- switch(choice,
          "1" = add(num1, num2),
          "2" = subtract(num1,
          num2), "3" =
          multiply(num1, num2),
          "4" = divide(num1,
          num2))

 cat("Result: ", result, "\n")
}
```

C) Write an R script, to create R objects for calculator application and save in a specified location in disk

```
# Define calculator functions
add <-
 function(x, y) {
 return(x + y)
}
subtract <-
 function(x, y) {
 return(x - y)
}
```

```
Multiply <- function(x, y) {
return(x * y) }
divide <-
 function(x, y) { if
 (y != 0) {
  return(x / y)
 } else {
  return("Cannot divide by zero!")
 }
}
# Specify the file path to save and load the R objects
save_file <- "calculator_functions.rds"
# Save the functions as R objects

saveRDS(list(add = add, subtract = subtract, multiply = multiply, divide = divide), file =

save_file) cat("Calculator functions saved to:", save_file, "\n")

# Load the saved R objects

loaded_functions <- readRDS(save_file)

# Now you can use the loaded functions

result_add <- loaded_functions$add (5, 3)
result_subtract <- loaded_functions$subtract(10, 3)


cat ("Result of addition:", result_add, "\n")
cat ("Result of subtraction:", result_subtract, "\n")
```

Result:

```
> source("d:\\Desktop\\REVA 1st SEM\\R Lab Programs\\Exp1.r", encoding = "UTF-$
[1] "Select operation."
[1] "1.Add"
[1] "2.Subtract"
[1] "3.Multiply"
[1] "4.Divide"
Enter choice[1/2/3/4]: 3
Enter first number: 5
Enter second number: 4
[1] "5 * 4 = 20"
```

2. DESCRIPTIVE STATISTICS IN R

a. Write an R script to find basic descriptive statistics using summary, str, quartile function on mtcars & cars datasets.

b. Write an R script to find subset of dataset by using subset (), aggregate () functions on iris dataset

a) Write an R script to find basic descriptive statistics using summary, str, quartile function on mtcars & cars datasets.

```
# Load necessary datasets
data(mtc
ars)
data(cars
)

# Summary statistics for mtcars dataset
summary(mtcars)

# Structure of mtcars dataset
str(mtcars)

# Quartiles for mpg column in mtcars dataset
quantile(mtcars$mpg
# Summary statistics for cars dataset
summary(cars)

# Structure of cars dataset
str(cars)

# Quartiles for speed column in cars dataset
quantile(cars$speed)
```

b) Write an R script to find subset of dataset by using subset (), aggregate () functions on iris dataset

```
# Load iris dataset
data(iris)

# Subset rows where Sepal.Length is greater than 5.0
subset_iris <- subset(iris, Sepal.Length > 5.0)

# Display the first few rows of the subsetted dataset
head(subset_iris)

# Aggregate the mean Sepal.Length for each Species in the iris dataset
agg_result <- aggregate(Sepal.Length ~ Species, data = iris, FUN = mean)

# Display the aggregated results
agg_result
```

Result:

```
Summary of cars dataset:

Structure of mtcars dataset:
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...

Structure of cars dataset:
'data.frame':   50 obs. of  2 variables:
 $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
 $ dist : num  2 10 4 22 16 10 18 26 34 17 ...

Quartiles of mtcars$mpg:
   25%    50%    75%
15.425 19.200 22.800

Quartiles of cars$speed:
25% 50% 75%
 12  15  19

Subset of iris dataset - Setosa:
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1           5.1         3.5          1.4         0.2  setosa
2           4.9         3.0          1.4         0.2  setosa
3           4.7         3.2          1.3         0.2  setosa
4           4.6         3.1          1.5         0.2  setosa
5           5.0         3.6          1.4         0.2  setosa
6           5.4         3.9          1.7         0.4  setosa
7           4.6         3.4          1.4         0.3  setosa
8           5.0         3.4          1.5         0.2  setosa
9           4.4         2.9          1.4         0.2  setosa
10          4.9         3.1          1.5         0.1  setosa
11          5.4         3.7          1.5         0.2  setosa
12          4.8         3.4          1.6         0.2  setosa
13          4.8         3.0          1.4         0.1  setosa
14          4.3         3.0          1.1         0.1  setosa
15          5.8         4.0          1.2         0.2  setosa
16          5.7         4.4          1.5         0.4  setosa
17          5.4         3.9          1.3         0.4  setosa
18          5.1         3.5          1.4         0.3  setosa
```

3. READING AND WRITING DIFFERENT TYPES OF DATASETS

a. Reading different types of data sets (.txt, .csv) from web and disk and writing in file in specific disk location.

b. Reading Excel data sheet in R. Reading XML dataset in R**.**

Part A

```
# Load necessary libraries
library(readr)

# Read data from a CSV file on disk
csv_data <- read_csv("path/to/your/file.csv")

# Read data from a TXT file on disk (assuming tab-delimited)
txt_data <- read_delim("path/to/your/file.txt", delim = "\t")

# Read data from a CSV file from a URL
url <- "https://example.com/data.csv"
web_csv_data <- read_csv(url)

# Read data from a TXT file from a URL (assuming tab-delimited)
web_txt_data <- read_delim(url, delim = "\t")

# Write data to a CSV file in a specific location
write_csv(csv_data, "path/to/output/csv_output.csv")

# Write data to a TXT file in a specific location
write_delim(txt_data, "path/to/output/txt_output.txt", delim = "\t")
```

Part B

```
# Load necessary libraries
library(readxl)
library(XML)

# Read an Excel data sheet
excel_data <- read_excel("path/to/your/excel_file.xlsx", sheet = "Sheet1")

# Read an XML dataset
xml_data <- xmlParse("path/to/your/xml_file.xml")
# Access XML data
root <- xmlRoot(xml_data)
# Depending on your XML structure, you can navigate and
# extract data from the 'root' element

# Example of writing to an XML file (creating a simple example)
new_xml <- newXMLNode("root")
newXMLNode("data", "Some text data", parent = new_xml)
saveXML(new_xml, "path/to/output/xml_output.xml")
```

Result: Note Output depends on the file to inputted

4. VISUALIZATIONS

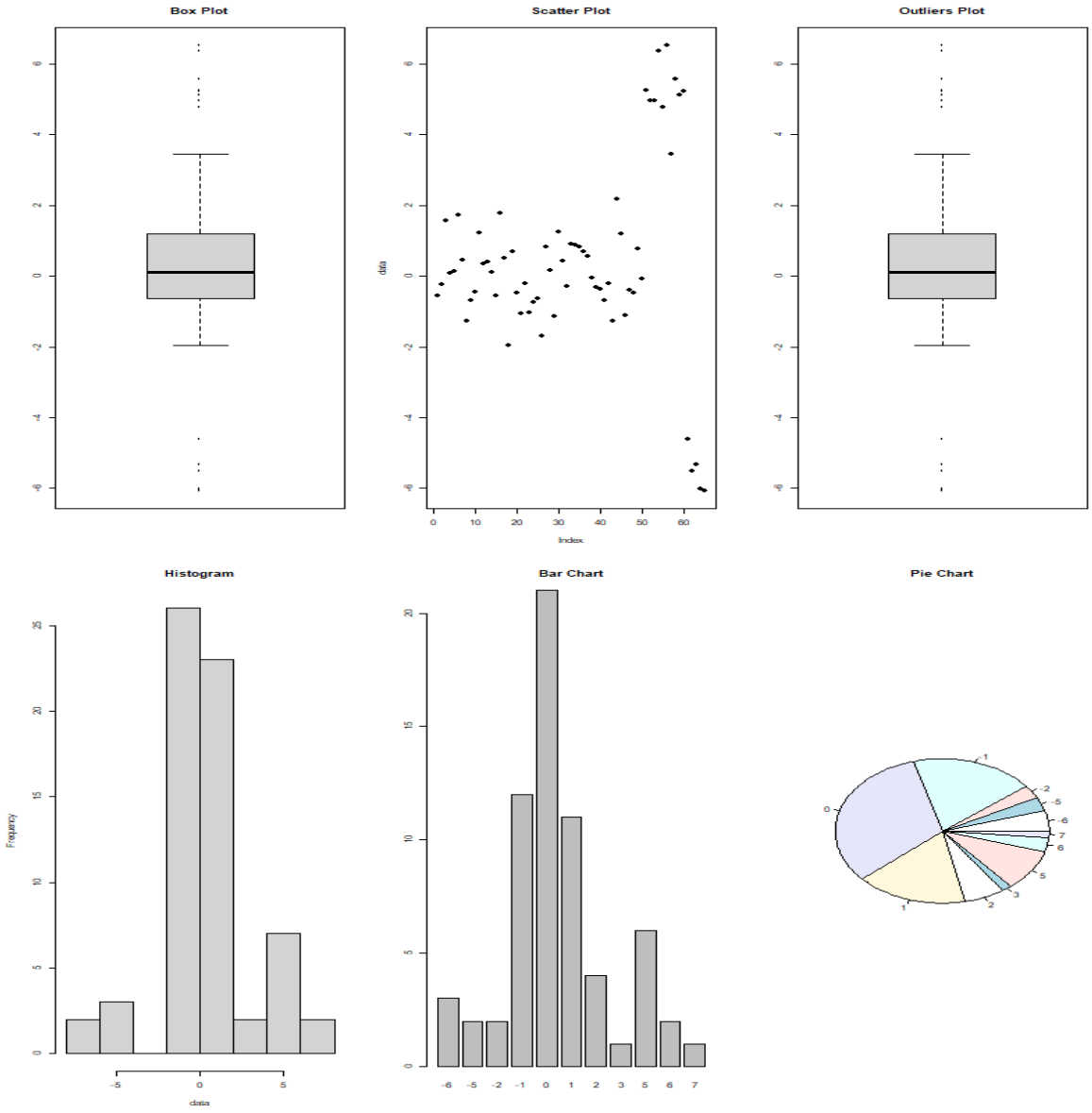
A .Find the data distributions using box and scatter plot.

b. Find the outliers using plot.

C .Plot the histogram, bar chart and pie chart on sample data**.**

```
# Graph Plotting
# Generate sample data
set.seed(123)
data <- c(rnorm(50), rnorm(10, mean = 5), rnorm(5, mean = -5))

# Plotting
par(mfrow = c(2, 3))  # Set up a 2x3 grid for plots
# Box and Scatter Plot
boxplot(data, main = "Box Plot")
plot(data, pch = 16, main = "Scatter Plot")
# Find Outliers using Plot
boxplot(data, outline = TRUE, main = "Outliers Plot")
# Histogram
hist(data, main = "Histogram")
# Bar Chart
bar_data <- table(round(data))
barplot(bar_data, main = "Bar Chart")

# Pie Chart
pie(bar_data, main = "Pie Chart")
```

Result:


Box Plot
Scatter Plot
data
Index
Outliers Plot
Histogram
Frequency
data
Bar Chart
Pie Chart

5. CORRELATION AND COVARIANCE

d. Find the correlation matrix.

e. Plot the correlation plot on dataset and visualize givingan overview of relationships among data on iris data.

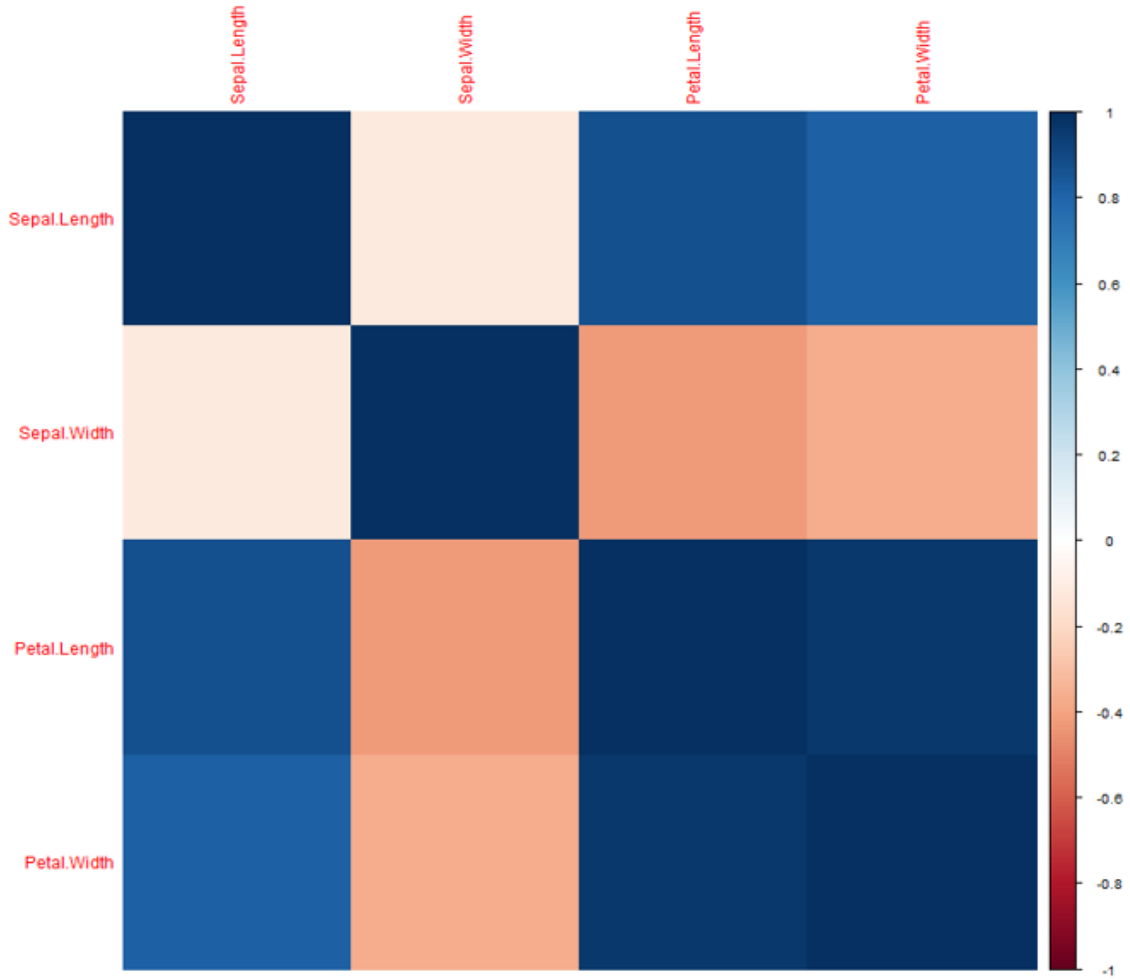
f. Analysis of covariance: variance (ANOVA) if datahave categorical variables on iris data.

```
# Should install.packages("ggplot2")
library(ggplot2)
# Should install.packages("corrplot")
library(corrplot)
# Should install.packages("stats")
library(stats)

# Load the iris dataset
data(iris)

# a. Find the correlation matrix
cor_matrix <- cor(iris[, -5])  # Exclude the species column

# Print the correlation matrix
print(cor_matrix)
# b. Plot the correlation plot
corrplot(cor_matrix, method = "color")
# c. Analysis of Variance (ANOVA)
# Perform ANOVA for Sepal.Length based on Species
anova_result <- aov(Sepal.Length ~ Species, data = iris)
```

Print ANOVA summary

print (summary(anova_result))

Result:



6. REGRESSION MODEL

Import data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables that are affecting the admission of a student in an institute based on his or her GRE score, GPA obtained and rank of the student. Also check if the model is fit or

not. require (foreign), require (MASS).

 # Regression Program

```
# Load required libraries
library(foreign)
library(MASS)

# Generate random data for the example
set.seed(123)  # Setting seed for reproducibility
n <- 200  # Number of observations
data <- data.frame(
  GRE = round(runif(n, min = 200, max = 800)),
  GPA = round(runif(n, min = 1, max = 4) * 4, 2),
  Rank = sample(1:4, n, replace = TRUE),
  Admission = factor(sample(0:1, n, replace = TRUE))
)

# Display the first few rows of the dataset
print(head(data))

# Perform logistic regression
logit_model <- glm(Admission ~ GRE + GPA + Rank, data = data, family = binomial)
```

```
# Summary of the model
summary (logit_model)

# Predicting on the same dataset for simplicity
data$Predicted <- predict(logit_model, type = "response")

# Checking the model fit using a confusion matrix
conf_matrix <- table(data$Admission, data$Predicted > 0.5)
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
print(paste("Accuracy:", accuracy))

# Checking the model's coefficients
print(coef(logit_model))
```

Result:

```
> source("d:\\Desktop\\REVA 1st SEM\\R Lab Programs\\Exp6.r", encoding = "UTF-$
  GRE   GPA Rank Admission
1 373  6.86    3         1
2 673 15.55    3         1
3 445 11.22    3         1
4 730 10.18    1         0
5 764  8.83    1         0
6 227 14.56    3         0
[1] "Accuracy: 0.53"
(Intercept)         GRE         GPA        Rank
-0.72612993  0.00035911  0.04300954  0.06182197
```

7. MULTIPLE REGRESSION MODEL

Apply multiple regressions if data have a continuous independent variable. Apply on above dataset.

```
# Multiple Regression Program
# Load required libraries
library(foreign)

# Generate random data for the example
set.seed(123)  # Setting seed for reproducibility
n <- 200  # Number of observations
data <- data.frame(
  GRE = round(runif(n, min = 200, max = 800)),
  GPA = round(runif(n, min = 1, max = 4) * 4, 2),
  Rank = sample(1:4, n, replace = TRUE),
  ContinuousVar = runif(n, min = 10, max = 50),
  Admission = factor(sample(0:1, n, replace = TRUE))
)


# Display the first few rows of the dataset
print(head(data))

# Perform logistic regression
logit_model <- glm(Admission ~ GRE + GPA + Rank + ContinuousVar, data = data, family
= binomial)
# Summary of the model
summary(logit_model)
# Checking the model's coefficients
print(coef(logit_model))
.
```

Result:

```
> source("d:\\Desktop\\REVA 1st SEM\\R Lab Programs\\Exp7.r", encoding = "UTF-$
  GRE   GPA Rank ContinuousVar Admission
1 373  6.86    3      19.48919         0
2 673 15.55    3      37.45961         0
3 445 11.22    3      19.03274         1
4 730 10.18    1      22.73978         1
5 764  8.83    1      16.95935         0
6 227 14.56    3      42.05718         0
  (Intercept)           GRE           GPA          Rank ContinuousVar
 -0.971478221   0.001014831   0.023180888   0.118496082  -0.005331046
>
```

8. REGRESSION MODEL FOR PREDICTION

Apply regression Model techniques to predict the data on above dataset.

```
# Regression Model for Prediction
# Generate random data for the example
set.seed(123)  # Setting seed for reproducibility
n <- 200  # Number of observations
data <- data.frame(
  GRE = round(runif(n, min = 200, max = 800)),
  GPA = round(runif(n, min = 1, max = 4) * 4, 2),
  Rank = sample(1:4, n, replace = TRUE),
  ContinuousVar = runif(n, min = 10, max = 50),
  Admission = factor(sample(0:1, n, replace = TRUE))
)

# Display the first few rows of the dataset
print(head(data))

# Split the data into training and testing sets
set.seed(123)  # Setting seed for reproducibility
trainIndex <- sample(n, n * 0.7)  # 70% of data for training
trainData <- data[trainIndex, ]
testData <- data[-trainIndex, ]
 # Perform logistic regression on the training data
logit_model <- glm(Admission ~ GRE + GPA + Rank + ContinuousVar, data = trainData,
family = binomial)
   # Predict admission on the testing data
predictions <- predict(logit_model, newdata = testData, type = "response")
```

```
# Convert probabilities to binary predictions (0 or 1)
  predicted_classes <- ifelse(predictions > 0.5, 1, 0)

  # Confusion matrix to evaluate model performance
  conf_matrix <- table(testData$Admission, predicted_classes)
  accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
  print(paste("Accuracy:", accuracy))
```

Result:

```
> source("d:\\Desktop\\REVA 1st SEM\\R Lab Programs\\Exp8.r", encoding = "UTF-$
  GRE   GPA Rank ContinuousVar Admission
1 373  6.86    3      19.48919         0
2 673 15.55    3      37.45961         0
3 445 11.22    3      19.03274         1
4 730 10.18    1      22.73978         1
5 764  8.83    1      16.95935         0
6 227 14.56    3      42.05718         0
[1] "Accuracy: 0.483333333333333"
```

9. CLASSIFICATION MODEL

a. Install relevant package for classification.

b. Choose classifier for classification problem.

c. Evaluate the performance of classifier.

 # Classification Model

```
# Load required libraries
library(caret)
library(e1071)  # For the SVM classifier (you can choose other classifiers as well)

# Load the iris dataset
data(iris)

# Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(iris$Species, p = 0.7, list = FALSE)
trainData <- iris[trainIndex, ]
testData <- iris[-trainIndex, ]

# Choose a classifier (Support Vector Machine in this case)
classifier <- svm(Species ~ ., data = trainData)

# Predict species on the testing data
predictions <- predict(classifier, newdata = testData)

# Confusion matrix to evaluate classifier performance
conf_matrix <- table(predictions, testData$Species)
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
print(paste("Accuracy:", accuracy))
```

Result:

```
> source("d:\\Desktop\\REVA 1st SEM\\R Lab Programs\\Exp9.r", encoding = "UTF-$
Loading required package: ggplot2
Loading required package: lattice
[1] "Accuracy: 0.933333333333333"
```

1**0.** CLUSTERING MODEL

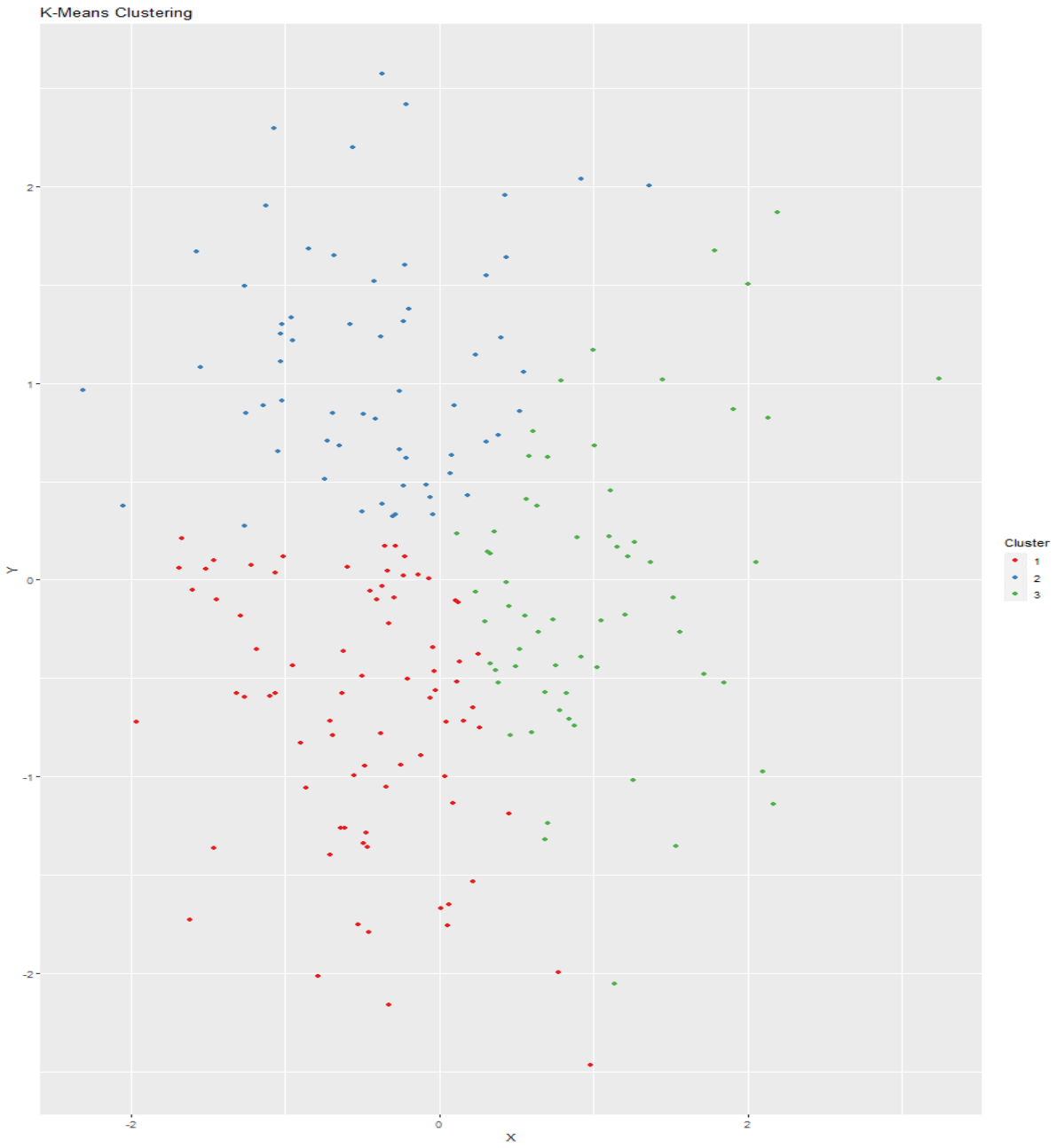
a. Clustering algorithms for unsupervised classification.

b. Plot the cluster data using R visualizations.

 # Clustering Model

```
# Load required libraries
library(ggplot2)  # For data visualization

# Generate random data for the example
set.seed(123)  # Setting seed for reproducibility
n <- 200
data <- data.frame(
  X = rnorm(n),
  Y = rnorm(n)
)

# Perform k-means clustering
num_clusters <- 3
kmeans_result <- kmeans(data, centers = num_clusters)

# Add cluster labels to the data
data$Cluster <- as.factor(kmeans_result$cluster)
# Plot the clustered data using ggplot2
print(ggplot(data, aes(x = X, y = Y, color = Cluster)) +
  geom_point() +
  scale_color_brewer(palette = "Set1") +
  labs(title = "K-Means Clustering"))
```

Result:



REVA
UNIVERSITY
Bengaluru, India

Rukmini Knowledge Park, Kattigenahalli
Yelahanka, Bengaluru - 560 064
Karnataka, India.

Ph: +91- 90211 90211, +91 80 4696 6966
E-mail: admissions@reva.edu.in

www.reva.edu.in

Follow us on

/REVA University