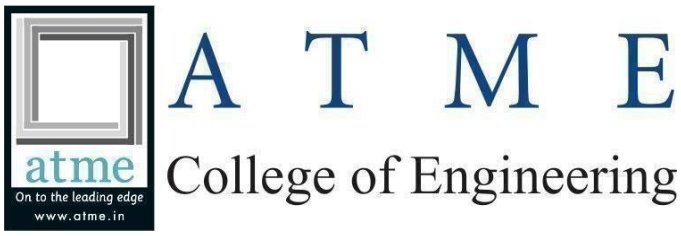
ATME COLLEGE OF ENGINEERING

13thKM Stone, Bannur Road, Mysore – 570028



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(ACADEMIC YEAR 2023-24)

LABORATORY MANUAL

SUBJECT: R PROGRAMMING LABORATORY

SUBJECT CODE: BCS358B

Composed by	Verified by	Approved by
Mrs. Sowmya K N & Ms.Pragathi B PROGRAMMER	Dr. Lakshmi Durga & Mrs. Keerthana M M FACULTY CO-ORDINATOR	Dr. Puttegowda D HOD, DEPT OF CSE

INSTITUTIONAL MISSION AND VISION

Objectives

- To provide quality education and groom top-notch professionals, entrepreneurs and leaders for different fields of engineering, technology and management.
- To open a Training-R & D-Design-Consultancy cell in each department, gradually introduce doctoral and postdoctoral programs, encourage basic & applied research in areas of social relevance, and develop the institute as a center of excellence.
- To develop academic, professional and financial alliances with the industry as well as the academia at national and transnational levels.
- To cultivate strong community relationships and involve the students and the staff in local community service.
- To constantly enhance the value of the educational inputs with the participation of students, faculty, parents and industry.

Vision

- Development of academically excellent, culturally vibrant, socially responsible and globally competent human resources.

Mission

- To keep pace with advancements in knowledge and make the students competitive and capable at the global level.
- To create an environment for the students to acquire the right physical, intellectual, emotional and moral foundations and shine as torch bearers of tomorrow's society.
- To strive to attain ever-higher benchmarks of educational excellence.

Department of Computer Science & Engineering

Vision of the Department

- To develop highly talented individuals in Computer Science and Engineering to deal with real world challenges in industry, education, research and society.

Mission of the Department

- To inculcate professional behavior, strong ethical values, innovative research capabilities and leadership abilities in the young minds & to provide a teaching environment that emphasizes depth, originality and critical thinking.
- Motivate students to put their thoughts and ideas adoptable by industry or to pursue higher studies leading to research.

Program outcomes (POs)

Engineering Graduates will be able to:

- **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- **PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- **PO3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **PO5. 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.

- **PO6. 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.
- **PO7. 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.
- **PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **PO9.** Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **PO10**. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **PO12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage

Program Educational Objectives (PEO'S):

1. Empower students with a strong basis in the mathematical, scientific and engineering fundamentals to solve computational problems and to prepare them for employment, higher learning and R&D.
2. Gain technical knowledge, skills and awareness of current technologies of computer science engineering and to develop an ability to design and provide novel engineering solutions for software/hardware problems through entrepreneurial skills.
3. Exposure to emerging technologies and work in teams on interdisciplinary projects with effective communication skills and leadership qualities.
4. Ability to function ethically and responsibly in a rapidly changing environment by applying innovative ideas in the latest technology, to become effective professionals in Computer Science to bear a life-long career in related areas.

Program Specific Outcomes (PSOs)

1. **PSO1:** Ability to apply skills in the field of algorithms, database design, web design, cloud computing and data analytics.
2. **PSO2:** Apply knowledge in the field of computer networks for building network and internet based applications.

R Programming		Semester	3
Course Code	BCS358 B	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	0:0:2:0	SEE Marks	50
Credits	01	Exam Hours	02
Examination type (SEE)	Practical		

Course objectives:

- To explore and understand how R and R Studio interactive environment.
- To understand the different data Structures, data types in R.
- To learn and practice programming techniques using R programming.
- To import data into R from various data sources and generate visualizations.
- To draw insights from datasets using data analytics techniques.

Sl.NO	Experiments
1	Demonstrate the steps for installation of R and R Studio. Perform the following: a) Assign different type of values to variables and display the type of variable. Assign different types such as Double, Integer, Logical, Complex and Character and understand the difference between each data type. b) Demonstrate Arithmetic and Logical Operations with simple examples. c) Demonstrate generation of sequences and creation of vectors. d) Demonstrate Creation of Matrices e) Demonstrate the Creation of Matrices from Vectors using Binding Function. f) Demonstrate element extraction from vectors, matrices and arrays **Suggested Reading** – Text Book 1 – Chapter 1 (What is R, Installing R, Choosing an IDE – RStudio, How to Get Help in R, Installing Extra Related Software), Chapter 2 (Mathematical Operations and Vectors, Assigning Variables, Special Numbers, Logical Vectors), Chapter 3 (Classes, Different Types of Numbers, Other Common Classes, Checking and Changing Classes, Examining Variables)
2	Assess the Financial Statement of an Organization being supplied with 2 vectors of data: Monthly Revenue and Monthly Expenses for the Financial Year. You can create your own sample data vector for this experiment) Calculate the following financial metrics: a. Profit for each month. b. Profit after tax for each month (Tax Rate is 30%). c. Profit margin for each month equals to profit after tax divided by revenue. d. Good Months – where the profit after tax was greater than the mean for the year. e. Bad Months – where the profit after tax was less than the mean for the year. f. The best month – where the profit after tax was max for the year. g. The worst month – where the profit after tax was min for the year. **Note:** a. All Results need to be presented as vectors b. Results for Dollar values need to be calculated with $0.01 precision, but need to be presented inUnits of $1000 (i.e 1k) with no decimal points c. Results for the profit margin ratio need to be presented in units of % with no decimal point. d. It is okay for tax to be negative for any given month (deferred tax asset) e. Generate CSV file for the data. **Suggested Reading** – Text Book 1 – Chapter 4 (Vectors, Combining Matrices)
3	Develop a program to create two 3 X 3 matrices A and B and perform the following operations a) Transpose of the matrix b) addition c) subtraction d) multiplication **Suggested Reading** – Text Book 1 – Chapter 4 (Matrices and Arrays – Array Arithmetic)
4	Develop a program to find the factorial of given number using recursive function calls. **Suggested Reading** – Reference Book 1 – Chapter 5 (5.5 – Recursive Programming) Text Book 1 – Chapter 8 (Flow Control and Loops – If and Else, Vectorized If, while loops, for loops),Chapter 6 (Creating and Calling Functions, Passing Functions to and from other functions)

<table>
<tr><td>5</td><td>Develop an R Program using functions to find all the prime numbers up to a specified number by themethod of Sieve of Eratosthenes.
Suggested Reading – Reference Book
1 - Chapter 5 (5.5 – Recursive Programming)
Text Book 1 – Chapter 8 (Flow Control and Loops – If and Else, Vectorized If, while loops, for loops),Chapter 6 (Creating and Calling Functions, Passing Functions to and from other functions)</td></tr>
<tr><td>6</td><td>The built-in data set mammals contain data on body weight versus brain weight. Develop R commands to:
a) Find the Pearson and Spearman correlation coefficients. Are they similar?
b) Plot the data using the plot command.
c) Plot the logarithm (log) of each variable and see if that makes a difference.
Suggested Reading – Text Book 1 –Chapter 12 – (Built-in Datasets) Chapter 14 – (Scatterplots)
Reference Book 2 – 13.2.5 (Covariance and Correlation)</td></tr>
<tr><td>7</td><td>Develop R program to create a Data Frame with following details and do the following operations.
<table>
<tr><th>itemCode</th><th>itemCategory</th><th>itemPrice</th></tr>
<tr><td>1001</td><td>Electronics</td><td>700</td></tr>
<tr><td>1002</td><td>Desktop Supplies</td><td>300</td></tr>
<tr><td>1003</td><td>Office Supplies</td><td>350</td></tr>
<tr><td>1004</td><td>USB</td><td>400</td></tr>
<tr><td>1005</td><td>CD Drive</td><td>800</td></tr>
</table>
a) Subset the Data frame and display the details of only those items whose price is greater than or equal to 350.
b) Subset the Data frame and display only the items where the category is either “Office Supplies” or “Desktop Supplies”
c) Create another Data Frame called “item-details” with three different fields itemCode, ItemQtyonHandand ItemReorderLvl and merge the two frames

Suggested Reading –Textbook 1: Chapter 5 (Lists and Data Frames)</td></tr>
<tr><td>8</td><td>Let us use the built-in dataset air quality which has Daily air quality measurements in New York, May to September 1973. Develop R program to generate histogram by using appropriate arguments for the following statements.
a) Assigning names, using the air quality data set.
b) Change colors of the Histogram
c) Remove Axis and Add labels to Histogram
d) Change Axis limits of a Histogram
e) Add Density curve to the histogram
Suggested Reading –Reference Book 2 – Chapter 7 (7.4 – The ggplot2 Package), Chapter 24 (Smoothingand Shading)</td></tr>
<tr><td>9</td><td>Design a data frame in R for storing about 20 employee details. Create a CSV file named “input.csv” that defines all the required information about the employee such as id, name, salary, start_date, dept. Import into R and do the following analysis.
a) Find the total number rows & columns
b) Find the maximum salary
c) Retrieve the details of the employee with maximum salary
d) Retrieve all the employees working in the IT Department.
e) Retrieve the employees in the IT Department whose salary is greater than 20000 and write these</td></tr>
</table>

	details into another file "output.csv" **Suggested Reading** – Text Book 1 – Chapter 12(CSV and Tab Delimited Files)
10	Using the built in dataset mtcars which is a popular dataset consisting of the design and fuel consumption patterns of 32 different automobiles. The data was extracted from the 1974 Motor Trend US magazine, andcomprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models). Format A data frame with 32 observations on 11 variables : [1] mpg Miles/(US) gallon, [2] cyl Number of cylinders [3] disp Displacement (cu.in.), [4] hp Gross horsepower [5] drat Rear axle ratio,[6] wt Weight (lb/1000) [7] qsec 1/4 mile time, [8] vs V/S, [9] am Transmission (0 = automatic, 1 = manual), [10] gear Number of forward gears, [11] carb Number of carburetors Develop R program, to solve the following: a) What is the total number of observations and variables in the dataset? b) Find the car with the largest hp and the least hp using suitable functions c) Plot histogram / density for each variable and determine whether continuous variables arenormally distributed or not. If not, what is their skewness? d) What is the average difference of gross horse power(hp) between automobiles with 3 and 4number of cylinders(cyl)? Also determine the difference in their standard deviations. e) Which pair of variables has the highest Pearson correlation? **References (Web links):** 1. https://cran.r-project.org/web/packages/explore/vignettes/explore_mtcars.html 2. https://www.w3schools.com/r/r_stat_data_set.asp 3. https://rpubs.com/BillB/217355
11	Demonstrate the progression of salary with years of experience using a suitable data set (You can create your own dataset). Plot the graph visualizing the best fit line on the plot of the given data points. Plot a curve of Actual Values vs. Predicted values to show their correlation and performance of the model. Interpret the meaning of the slope and y-intercept of the line with respect to the given data. Implement using lm function. Save the graphs and coefficients in files. Attach the predicted values of salaries as a newcolumn to the original data set and save the data as a new CSV file. **Suggested Reading** – Reference Book 2 – Chapter 20 (General Concepts, Statistical Inference, Prediction)

Course outcomes (Course Skill Set):

At the end of the course the student will be able to:

- Explain the fundamental syntax of R data types, expressions and the usage of the R-Studio IDE
- Develop a program in R with programming constructs: conditionals, looping and functions.
- Apply the list and data frame structure of the R programming language.
- Use visualization packages and file handlers for data analysis..

R Programming Language

Introduction

R is an open-source programming language that is widely used as a statistical software and data analysis tool. R generally comes with the Command-line interface. R is available across widely used platforms like Windows, Linux, and macOS. Also, the R programming language is the latest cutting-edge tool.

It was designed by **Ross Ihaka and Robert Gentleman** at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team. R programming language is an implementation of the S programming language. It also combines with lexical scoping semantics inspired by Scheme. Moreover, the project conceives in 1992, with an initial version released in 1995 and a stable beta version in 2000.

Why R Programming Language?

- R programming is used as a leading tool for machine learning, statistics, and data analysis. Objects, functions, and packages can easily be created by R.
- It's a platform-independent language. This means it can be applied to all operating system.
- It's an open-source free language. That means anyone can install it in any organization without purchasing a license.
- R programming language is not only a statistic package but also allows us to integrate with other languages (C, C++). Thus, you can easily interact with many data sources and statistical packages.
- The R programming language has a vast community of users and it's growing day by day.
- R is currently one of the most requested programming languages in the Data Science job market that makes it the hottest trend nowadays.

Features of R Programming Language

Statistical Features of R:

- **Basic Statistics:** The most common basic statistics terms are the mean, mode, and median. These are all known as "Measures of Central Tendency." So using the R language we can measure central tendency very easily.
- **Static graphics:** R is rich with facilities for creating and developing interesting static graphics. R contains functionality for many plot types including graphic maps, mosaic plots, biplots, and the list goes on.

- **Probability distributions:** Probability distributions play a vital role in statistics and by using R we can easily handle various types of probability distribution such as Binomial Distribution, Normal Distribution, Chi-squared Distribution and many more.
- **Data analysis:** It provides a large, coherent and integrated collection of tools for data analysis.

Programming Features of R:

- **R Packages:** One of the major features of R is it has a wide availability of libraries. R has CRAN(Comprehensive R Archive Network), which is a repository holding more than 10, 0000 packages.
- **Distributed Computing:** Distributed computing is a model in which components of a software system are shared among multiple computers to improve efficiency and performance. Two new packages **ddR and multidplyr** used for distributed programming in R were released in November 2015.

Programming in R:

Since R is much similar to other widely used languages syntactically, it is easier to code and learn in R. Programs can be written in R in any of the widely used IDE like **R Studio, Rattle, Tinn-R**, etc. After writing the program save the file with the extension **.r**. To run the program use the following command on the command line:
R file_name.r

Example:

```
# R program to print Welcome to GFG!SSSS
# Below line will print "Welcome to GFG!"
cat("Welcome to GFG!")
```

Output:
Welcome to GFG!

What are R Data types?

R Data types are used in computer programming to specify the kind of data that can be stored in a variable. For effective memory consumption and precise computation, the right data type must be selected. Each R data type has its own set of regulations and restrictions.

Data Types in R Programming Language

Each variable in R has an associated data type. Each R-Data Type requires different amounts of memory and has some specific operations which can be performed over it. R Programming language has the following basic R-data types and the following table shows the data type and the values that each data type can take.

Basic Data Types	Values	Examples
Numeric	Set of all real numbers	"numeric_value <- 3.14"
Integer	Set of all integers, Z	"integer_value <- 42L"
Logical	TRUE and FALSE	"logical_value <- TRUE"
Complex	Set of complex numbers	"complex_value <- 1 + 2i"
Character	“a”, “b”, “c”, …, “@”, “#”, “$”, …., “1”, “2”, …etc	"character_value <- "Hello Geeks"
raw	as.raw()	"single_raw <- as.raw(255)"

Variables in R:

R Programming Language is a dynamically typed language, i.e. the R Language Variables are not declared with a data type rather they take the data type of the R-object assigned to them. This feature is also shown in languages like Python and PHP.

Declaring and Initializing Variables in R Language

R supports three ways of variable assignment:

- Using equal operator- operators use an arrow or an equal sign to assign values to variables.
- Using the leftward operator- data is copied from right to left.
- Using the rightward operator- data is copied from left to right.

R Variables Syntax

- Types of Variable Creation in R:
- Using equal to operators
  variable_name = value

- using leftward operator
  variable_name <- value

- using rightward operator
  value -> variable_name

Taking Input from User in R Programming

Developers often have a need to interact with users, either to get data or to provide some sort of result. Most programs today use a dialog box as a way of asking the user to provide some type of input. Like other programming languages in R it's also possible to take input from the user. For doing so, there are two methods in R.

- Using **readline()** method
- Using **scan() method**

Using readline() method

In R language **readline()** method takes input in string format. If one inputs an integer then it is inputted as a string, lets say, one wants to input **255**, then it will input as **"255"**, like a string. So one needs to convert that inputted value to the format that he needs. In this case, string **"255"** is converted to integer 255. To convert the inputted value to the desired data type, there are some functions in R,

Data Structures in R Programming

A data structure is a particular way of organizing data in a computer so that it can be used effectively. The idea is to reduce the space and time complexities of different tasks. Data structures in R programming are tools for holding multiple values.

R's base data structures are often organized by their dimensionality (1D, 2D, or nD) and whether they're homogeneous (all elements must be of the identical type) or heterogeneous (the elements are often of various types). This gives rise to the six data types which are most frequently utilized in data analysis.

The most essential data structures used in R include:

- **Vectors**
- **Lists**
- **Dataframes**
- **Matrices**
- **Arrays**
- **Factors**

Vectors

A vector is an ordered collection of basic data types of a given length. The only key thing here is all the elements of a vector must be of the identical data type e.g homogeneous data structures. Vectors are one-dimensional data structures.

Example:

```
# R program to illustrate Vector
 # Vectors(ordered collection of same data type)
X = c(1, 3, 5, 7, 8)
 # Printing those elements in console
print(X)
```

Output:
[1] 1 3 5 7 8

R Strings

Strings are a bunch of character variables. It is a one-dimensional array of characters. One or more characters enclosed in a pair of matching single or double quotes can be considered a string in R. Strings in R Programming represent textual content and can contain numbers, spaces, and special characters. An empty string is represented by using ". R Strings are always stored as double-quoted values. A double-quoted string can contain single quotes within it. Single-quoted strings can't contain single quotes. Similarly, double quotes can't be surrounded by double quotes.

Creation of String in R
R Strings can be created by assigning character values to a variable. These strings can be further concatenated by using various functions and methods to form a big string.

Example

```
# R program for String Creation
 # creating a string with double quotes
str1 <- "OK1"
cat ("String 1 is : ", str1)
 # creating a string with single quotes
str2 <- 'OK2'
cat ("String 2 is : ", str2)
str3 <- "This is 'acceptable and 'allowed' in R"
cat ("String 3 is : ", str3)
str4 <- 'Hi, Wondering "if this "works"'
cat ("String 4 is : ", str4)
str5 <- 'hi, ' this is not allowed'
cat ("String 5 is : ", str5)
```

Output

String 1 is: OK1

String 2 is: OK2

String 3 is: This is 'acceptable and 'allowed' in R

String 4 is: Hi, Wondering "if this "works"

Error: unexpected symbol in " str5 <- 'hi, ' this"

Execution halted

R Vectors

R vectors are the same as the arrays in C language which are used to hold multiple data values of the same type. One major key point is that in R the indexing of the vector will start from '1' and not from '0'. We can create numeric vectors and character vectors as well.

Types of R vectors

Vectors are of different types which are used in R. Following are some of the types of vectors:

Numeric vectors: Numeric vectors are those which contain numeric values such as integer, float, etc.

```
# R program to create numeric Vectors
 # creation of vectors using c() function.
v1<- c(4, 5, 6, 7)
 # display type of vector
typeof(v1)
 # by using 'L' we can specify that we want integer values.
v2<- c(1L, 4L, 2L, 5L)
 # display type of vector
typeof(v2)
```

Output:

[1] "double"

[1] "integer"

Character vectors: Character vectors in R contain alphanumeric values and special characters.

```
# R program to create Character Vectors
# by default numeric values
# are converted into characters
v1<- c('geeks', '2', 'hello', 57)
# Displaying type of vector
typeof(v1)
```

Output:

[1] "character"

Logical vectors: Logical vectors in R contain Boolean values such as TRUE, FALSE and NA for Null values.

R

```
# R program to create Logical Vectors
# Creating logical vector
# using c() function
v1<- c(TRUE, FALSE, TRUE, NA)
# Displaying type of vector
typeof(v1)
```

Output:

[1] "logical"

Creating a vector

There are different ways of creating R vectors. Generally, we use ‘c’ to combine different elements together.

R

```
# R program to create Vectors
 # we can use the c function
# to combine the values as a vector.
# By default the type will be double
X<- c(61, 4, 21, 67, 89, 2)
cat('using c function', X, '\n')
# seq() function for creating
# a sequence of continuous values.
# length.out defines the length of vector.
Y<- seq(1, 10, length.out = 5)
cat('using seq() function', Y, '\n')
 # use':' to create a vector
# of continuous values.
Z<- 2:7
cat('using colon', Z)
```

Output:

using c function 61 4 21 67 89 2

using seq() function 1 3.25 5.5 7.75 10

using colon 2 3 4 5 6 7

Length of R vector

R

```
# Create a numeric vector
x <- c(1, 2, 3, 4, 5)
# Find the length of the vector
length(x)
# Create a character vector
y <- c("apple", "banana", "cherry")
 # Find the length of the vector
length(y)
# Create a logical vector
z <- c(TRUE, FALSE, TRUE, TRUE)
```

```
 # Find the length of the vector
length(z)
```

Output:

> length(x)

[1] 5

> length(y)

[1] 3

> length(z)

[1] 4

Accessing R vector elements

R – Lists

A list in R is a generic object consisting of an **ordered** collection of objects. Lists are one-dimensional, heterogeneous data structures. The list can be a list of vectors, a list of matrices, a list of characters and a list of functions, and so on.

A list is a vector but with heterogeneous data elements. A list in R is created with the use of **list()** function. R allows accessing elements of an R list with the use of the index value. In R, the indexing of a list starts with 1 instead of 0 like in other programming languages.

Creating a List

To create a List in R you need to use the function called "list()". In other words, a list is a generic vector containing other objects. To illustrate how a list looks, we take an example here. We want to build a list of employees with the details. So for this, we want attributes such as ID, employee name, and the number of employees.

Example:

```
# R program to create a List
# The first attributes is a numeric vector
# containing the employee IDs which is created
# using the command here

empId = c(1, 2, 3, 4)
```

```
# The second attribute is the employee name

# which is created using this line of code here

# which is the character vector

empName = c("Debi", "Sandeep", "Subham", "Shiba")

  # The third attribute is the number of employees

# which is a single numeric variable.

numberOfEmp = 4

  # We can combine all these three different

# data types into a list

# containing the details of employees

# which can be done using a list command

empList = list(empId, empName, numberOfEmp)

 print(empList)
```

Output:

```
[[1]]

[1] 1 2 3 4


[[2]]

[1] "Debi"    "Sandeep" "Subham"  "Shiba"


[[3]]

[1] 4
```

R – Array

Arrays are essential data storage structures defined by a fixed number of dimensions. Arrays are used for the allocation of space at contiguous memory locations. Uni-dimensional arrays are called vectors with the length being their only dimension. Two-dimensional arrays are called matrices, consisting of fixed numbers of rows and columns. Arrays consist of all elements of the same data type. Vectors are supplied as input to the function and then create an array based on the number of dimensions.

Creating an Array

An array in R can be created with the use of **array()** function. List of elements is passed to the array() functions along with the dimensions as required.

Syntax:

array(data, dim = (nrow, ncol, nmat), dimnames=names)

where,

nrow : Number of rows
ncol : Number of columns
nmat : Number of matrices of dimensions nrow * ncol
dimnames : Default value = NULL.

Otherwise, a list has to be specified which has a name for each component of the dimension. Each component is either a null or a vector of length equal to the dim value of that corresponding dimension.

Uni-Dimensional Array

A vector is a uni-dimensional array, which is specified by a single dimension, length. A Vector can be created using ‘c()‘ function. A list of values is passed to the c() function to create a vector.

Example:

```
vec1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)

print (vec1)

# cat is used to concatenate # strings and print it.

cat ("Length of vector : ", length(vec1))
```

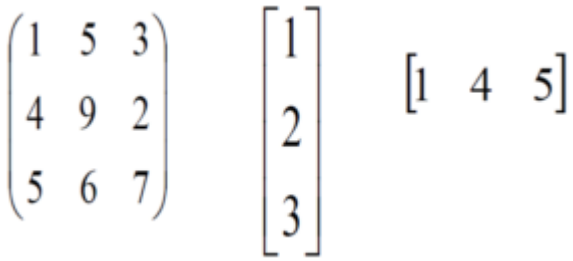
Output:

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

Length of vector : 9

R – Matrices

Matrix is a rectangular arrangement of numbers in rows and columns. In a matrix, as we know rows are the ones that run horizontally and columns are the ones that run vertically. In R programming, matrices are two-dimensional, homogeneous data structures. These are some examples of matrices:



R – Matrices

Creating a Matrix

To create a matrix in R you need to use the function called **matrix()**. The arguments to this **matrix()** are the set of elements in the vector. You have to pass how many numbers of rows and how many numbers of columns you want to have in your matrix.

Note: By default, matrices are in column-wise order.

R

```
# R program to create a matrix

A = matrix(

# Taking sequence of elements

c(1, 2, 3, 4, 5, 6, 7, 8, 9),

 # No of rows

 nrow = 3,
```

```
# No of columns

ncol = 3,

# By default matrices are in column-wise order

# So this parameter decides how to arrange the matrix

byrow = TRUE

)

# Naming rows

rownames(A) = c("a", "b", "c")

# Naming columns

colnames(A) = c("c", "d", "e")

cat("The 3x3 matrix:\n")

print(A)
```

Output:

The 3x3 matrix:

```
  c d e
a 1 2 3
b 4 5 6
c 7 8 9
```

R Factors

Factors in R Programming Language are data structures that are implemented to categorize the data or represent categorical data and store it on multiple levels.
They can be stored as integers with a corresponding label to every unique integer. The R factors may look similar to character vectors, they are integers and care must be taken while

using them as strings. The R factor accepts only a restricted number of distinct values. For example, a data field such as gender may contain values only from female, male, or transgender.
In the above example, all the possible cases are known beforehand and are predefined. These distinct values are known as levels. After a factor is created it only consists of levels that are by default sorted alphabetically.

Attributes of Factors in R Language

- **x:** It is the vector that needs to be converted into a factor.
- **Levels:** It is a set of distinct values which are given to the input vector x.
- **Labels:** It is a character vector corresponding to the number of labels.
- **Exclude:** This will mention all the values you want to exclude.
- **Ordered:** This logical attribute decides whether the levels are ordered.
- **nmax:** It will decide the upper limit for the maximum number of levels.

Creating a Factor in R Programming Language

The command used to create or modify a factor in R language is – **factor()** with a vector as input.
The two steps to creating an R factor :

- Creating a vector
- Converting the vector created into a factor using function factor()
-

Examples: Let us create a factor gender with levels female, male and transgender.

R

```
# Creating a vector

x <-c("female", "male", "male", "female")

print(x)

# Converting the vector x into a factor

# named gender

gender <-factor(x)

print(gender)
```

Output

[1] "female" "male" "male" "female"

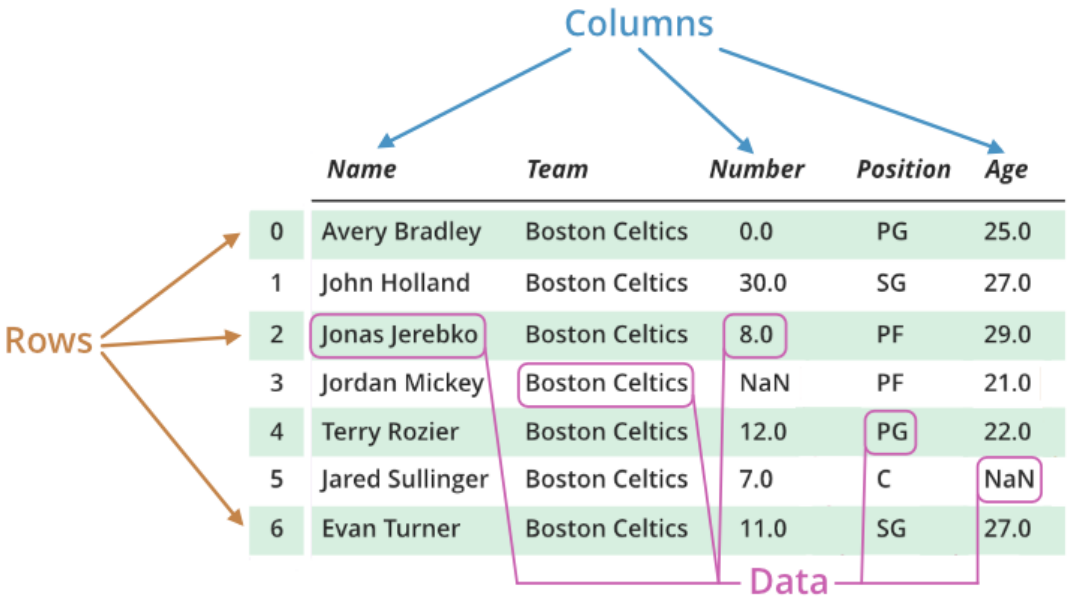
[1] female male male female

Levels: female male

R – Data Frames

R Programming Language is an open-source programming language that is widely used as a statistical software and data analysis tool. **Data Frames in R Language** are generic data objects of R that are used to store tabular data. Data frames can also be interpreted as matrices where each column of a matrix can be of different data types. R DataFrame is made up of three principal components, the data, rows, and columns.

R – Data Frames



R – Data Frames

Create Dataframe in R Programming Language

To create an R data frame use **data.frame()** command and then pass each of the vectors you have created as arguments to the function.

Example:

R

```
# R program to create dataframe

 # creating a data frame

friend.data <- data.frame(

friend_id = c(1:5),

friend_name = c("Sachin", "Sourav",

                "Dravid", "Sehwag",

                "Dhoni"),

   stringsAsFactors = FALSE

)

# print the data frame

print(friend.data)
```

Output:

```
  friend_id friend_name
1         1      Sachin
2         2      Sourav
3         3      Dravid
4         4      Sehwag
5         5       Dhoni
```

R dataset

A dataset is a data collection presented in a table.

The R programming language has tons of built-in datasets that can generally be used as a demo data to illustrate how the R functions work.

Most Used built-in Datasets in R

In R, there are tons of datasets we can try but the mostly used built-in datasets are:

airquality - New York Air Quality Measurements

AirPassengers - Monthly Airline Passenger Numbers 1949-1960

mtcars - Motor Trend Car Road Tests

iris - Edgar Anderson's Iris Data

.

PROGRAMS

Program 1:

Demonstrate the steps for installation of R and R Studio. Perform the following:

a.Assign different type of values to variables and display the type of variable. Assign different types such as Double, Integer, Logical, Complex and Character and understand the difference between each data type.

b. Demonstrate Arithmetic and Logical Operations with simple examples.

c. Demonstrate generation of sequences and creation of vectors.

d. Demonstrate Creation of Matrices

e. Demonstrate the Creation of Matrices from Vectors using Binding Function.

f. Demonstrate element extraction from vectors, matrices and arrays

Install R and RStudio

1. To install R, go to cran.r-project.org. ...
2. Click Download R for Windows.
3. Install R Click on install R for the first time.
4. Click Download R for Windows. ...
5. Select the language you would like to use during the installation. ...
6. Click Next.
7. Select where you would like R to be installed.

a. Assign different type of values to variables and display the type of variable. Assign different types such as Double, Integer, Logical, Complex and Character and understand the difference between each data type.

```
# Assigning to Variables

c <- "Hello, R!"

# Character d=3.14159

# Double i <- 42

# Integer

l <- TRUE
```

```
# Logical
cmp <- 3 + 2i # Complex

# Display variable types
cat("d is of type:",
typeof(d))
cat("i is of type:",
typeof(i), "\n")
cat("l is of type:",
typeof(l), "\n")
cat("cmp is of type:",
typeof(cmp), "\n")
cat("c is of type:", typeof(c),
"\n")

# Arithmetic Operations
cat("d + i:", d + i, "\n")
cat("d * i:", d * i, "\n")
cat("d / i", d / i, "\n")

# Logical Operations cat
("Logical AND l && T:", l && T,"\n")
cat("Logical OR l || F:", l || F, "\n")
cat("Logical NOT !l:", !l, "\n")

# Generate a sequence of integers from 1 to 10
sv <- 1:10
# Create a vector
v <- c(5, 10, 15, 20, 25)
cat("Sequence of Integers:", sv, "\n")
```

```
cat("My Vector:", v, "\n")
# Create a matrix
m <- matrix(1:12,
nrow <- 3, ncol <- 4)
cat("My Matrix:\n")
cat(m)
# Create vectors
v1 <- c(1, 2, 3)
v2 <- c(4, 5, 6)
v3 <- c(7, 8, 9)
# Bind vectors into a matrix
mv <- rbind(v1, v2,v3)
cat("Matrix from Vectors:\n")
cat(mv)

# Extract elements from vectors
e1 <- mv[2] e2 <- sv[5]
# Extract elements from matrices
e3 <- m[2, 3] e4 <- mv[1, 2]
cat("Element from My Vector:", e1, "\n")
cat("Element from Sequence Vector:", e2, "\n")
cat("Element from My Matrix:", e3, "\n")
cat("Element from Matrix from Vectors:", e4, "\n")
```

OUTPUT:

```
> source("C:/Users/atme/Desktop/4AD22CS093/program1.R")
d is of type: doublei is of type: double
l is of type: logical
cmp is of type: complex
c is of type: character
d+i: 45.1459
d*i: 132.1278
d/i 0.07490238
logical AND|&&T: TRUE
logocal OR|llF: TRUE
logical NOT!l: FALSE
sequence of integers: 1 2 3 4 5 6 7 8 9 10
my vector: 5 10 15 20 25
my matrix:
1 2 3 4 5 6 7 8 9 10 11 12matrix from vectors:
   [,1] [,2] [,3]
v1    1    2    3
v2    4    5    6
v3    7    8    9
```

Program 2:

Assess the Financial Statement of an Organization being supplied with 2 vectors of data: Monthly Revenue and Monthly Expenses for the Financial Year. You can create your own sample data vector for this experiment) Calculate the following financial metrics:

Profit for each month.

Profit after tax for each month (Tax Rate is 30%).

Profit margin for each month equals to profit after tax divided by revenue.

Good Months – where the profit after tax was greater than the mean for the year.

Bad Months – where the profit after tax was less than the mean for the year.

The best month – where the profit after tax was max for the year.

The worst month – where the profit after tax was min for the year.

```
# Sample data for monthly revenue and expenses (in $1000 units)
monthly_revenue <- c(50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 155, 165)
monthly_expenses <- c(30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85)

# Calculate profit for each month
 profit <- monthly_revenue - monthly_expenses

# Calculate profit after tax for each month (Tax Rate is 30%)
 tax_rate <- 0.30
profit_after_tax <- profit * (1 - tax_rate)

# Calculate profit margin for each month as a percentage
 profit_margin <- (profit_after_tax / monthly_revenue) * 100

# Calculate the mean profit after tax for the year
mean_profit_after_tax <- mean(profit_after_tax)

# Determine good months, bad months, best month, and worst month
```

```
good_months <- profit_after_tax > mean_profit_after_tax
 bad_months <- profit_after_tax
 < mean_profit_after_tax
best_month <- which.max(profit_after_tax)
worst_month <- which.min(profit_after_tax)

# Format results as vectors with appropriate units and precision
profit <- round(profit * 1000, 2)  # Convert to $1000 units
profit_after_tax <- round(profit_after_tax * 1000, 2)  # Convert to $1000 units
profit_margin <- round(profit_margin, 0)  # Remove decimal points for percentage

# Create a data frame to store the results results
<- data.frame(
Month = 1:12,
Profit = profit,
ProfitAfterTax = profit_after_tax,
ProfitMargin = profit_margin,
GoodMonth = good_months,
BadMonth = bad_months
)
# Print the results
cat("Profit for each month (in $1000 units):\n")
cat(results$Profit, "\n\n")
cat("Profit after tax for each month (in $1000 units):\n")
cat(results$ProfitAfterTax, "\n\n")
cat("Profit margin for each month (in %):\n")
cat(results$ProfitMargin, "\n\n")
cat("Good Months (Profit after tax greater than mean):\n")
cat(results$Month[results$GoodMonth], "\n\n")
cat("Bad Months (Profit after tax less than mean):\n")
```

```
cat(results$Month[results$BadMonth], "\n\n")
cat("Best Month (Max Profit after tax):\n")
cat(results$Month[best_month], "\n\n")
cat("Worst Month (Min Profit after tax):\n")
cat(results$Month[worst_month], "\n\n")

# Export the results to a CSV file
 write.csv(results, "financial_metrics.csv", row.names = FALSE)
```

OUTPUT:

```
>
> source("~/PGRM2.J/fin program 2.R")
Profit for each month (in $1000 units):
99000 25000 30000 125000 40000 45000 55000 55000 60000 65000 41000 80000

Profit after tax for each month (in $1000 units):
69300 17500 21000 87500 28000 31500 38500 38500 42000 45500 28700 56000

Profit margin for each month (in %):
54 29 30 51 31 31 33 32 32 32 24 34

Good Months (Profit after tax greater than mean):
1 4 10 12

Bad Months (Profit after tax less than mean):
2 3 5 6 7 8 11

Best Month (Max Profit after tax):
4

Worst Month (Min Profit after tax):
2

>
```

Program 3:

Develop a program to create two 3 X 3 matrices A and B and perform the following operations

a) Transpose of the matrix

b) addition

c) subtraction

d) multiplication

```
#  Create matrices A and B
A = matrix(1:9, nrow = 3, ncol = 3)
B = matrix(9:1, nrow = 3, ncol = 3)

# a) Transpose of the matrix
A_t  =  t(A)
B_t  =  t(B)

# b) Addition
sum  =  A + B

# c) Subtraction
diff  =  A - B

# d) Multiplication
prod  =  A %*% B

# Print the results
cat("Matrix A:\n")
print(A)
cat("Matrix B:\n")
```

```
print(B)
cat("Transpose of A:\n")
print(A_t)
cat("Transpose of B:\n")
print(B_t)
cat("Addition of A and B:\n")
print(sum)
cat("Subtraction of A and B:\n")
print(diff)
cat("Multiplication of A and B:\n")
print(prod)
```

OUTPUT:

```
> source("~/CS060/praju-075.R")
Matrix A:
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
Matrix B:
     [,1] [,2] [,3]
[1,]    9    6    3
[2,]    8    5    2
[3,]    7    4    1
Transpose of A:
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
Transpose of B:
     [,1] [,2] [,3]
[1,]    9    8    7
[2,]    6    5    4
[3,]    3    2    1
Addition of A and B:
     [,1] [,2] [,3]
[1,]   10   10   10
[2,]   10   10   10
[3,]   10   10   10
```

```
Multiplication of A and B:
     [,1] [,2] [,3]
[1,]   90   54   18
[2,]  114   69   24
[3,]  138   84   30
> print (prod)
     [,1] [,2] [,3]
[1,]   90   54   18
[2,]  114   69   24
[3,]  138   84   30
>
>
```

Program 4:

Develop a program to find the factorial of given number using recursive function calls.

```
# Recursive function to calculate the factorial
fact = function(n) {
if (n == 0) {
    return(1)
} else {
return(n * fact(n - 1))
 }
}
# Input a number from the user
n = as.integer(readline("Enter a non-negative integer: "))
if (n < 0) {
cat("Factorial is not defined for negative numbers.\n")
} else {
result = fact(n)
cat("The factorial of", n, "is", result, "\n")
}
```

OUTPUT:

```
> source("C:/Users/atme/Desktop/4AD22CS093/program4.R")
Enter a non-negative integer:4
the factorial of 4 is 24
```

Program 5:

Develop an R Program using functions to find all the prime numbers up to a specified number by the method of Sieve of Eratosthenes

```
# Function to find all prime numbers up to a specified number using the Sieve of Eratosthenes
sieve_of_eratosthenes <- function(n) {
 if (n < 2) {
 cat("No prime numbers in the specified range.\n")
 return()
 }
 is_prime <- rep(TRUE, n)
 is_prime[1] <- FALSE  # 1 is not prime
 p <- 2
 while (p^2 <= n) {
 if (is_prime[p]) {
 for (i in seq(p^2, n + 1, by = p)){
   is_prime[i] <- FALSE
}
  }
 p <- p + 1
}
 primes <- which(is_prime)
cat("Prime numbers up to", n, "are:\n", primes, "\n")
}

# Input a number from the user
n <- as.integer(readline("Enter a positive integer: "))
sieve_of_eratosthenes(n)
```

OUTPUT:

```
> source("C:/Users/atme/Desktop/4AD22CS093/program5.R")
Enter a positive integer:0
no prime number in the specified range.
> source("C:/Users/atme/Desktop/4AD22CS093/program5.R")
Enter a positive integer:1
no prime number in the specified range.
```

```
> source("C:/Users/atme/Desktop/4AD22CS093/program5.R")
Enter a positive integer:5
prime numbers up to 5 are:
 2 3 5
```

Program 6:

The built-in data set mammals contain data on body weight versus brain weight. Develop R commands to:

a.Find the Pearson and Spearman correlation coefficients. Are they similar?

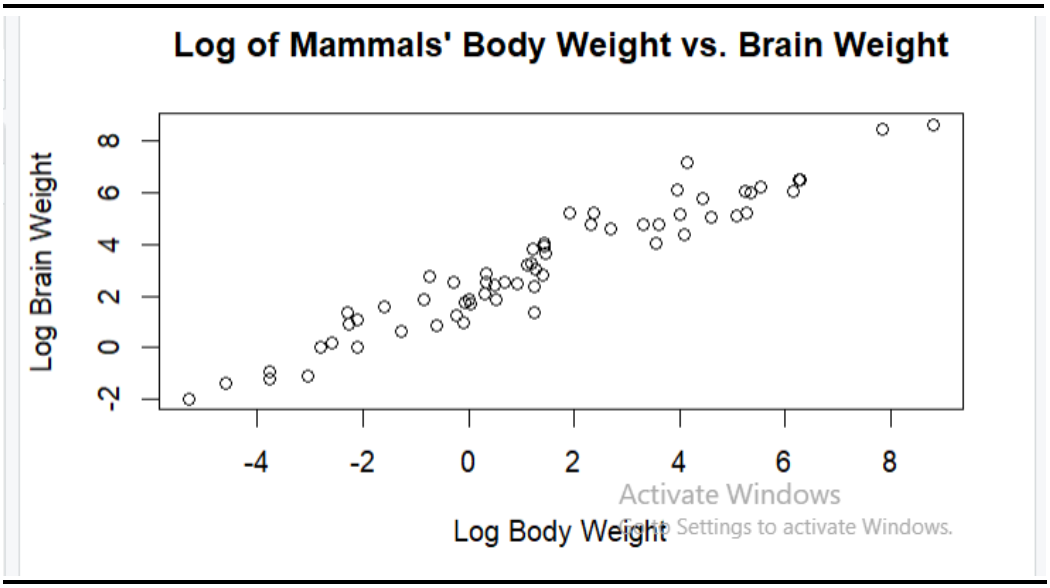
b.Plot the data using the plot command.

c.Plot the logarithm (log) of each variable and see if that makes a difference.

```
data("mammals", package = "MASS")
# a) Finding the Pearson and Spearman correlation coefficients
pcorr <- cor(mammals$body, mammals$brain, method = "pearson")
scorr <- cor(mammals$body, mammals$brain, method = "spearman")

# Displaying the correlation coefficients
cat("Pearson correlation coefficient: ", pcorr, "\n")
cat("Spearman correlation coefficient: ", scorr, "\n")

# b) Plotting the data using the plot command
plot(mammals$body, mammals$brain, xlab = "Body Weight", ylab = "Brain Weight",
main = "Mammals' Body Weight vs. Brain Weight")

# c) Plotting the logarithm (log) of each variable and checking the difference
log_body <- log(mammals$body)
log_brain <- log(mammals$brain)
 plot(log_body, log_brain, xlab = "Log Body Weight", ylab = "Log Brain Weight",
main = "Log of Mammals' Body Weight vs. Brain Weight")
```

OUTPUT:

```
>
> source("//csserver2/CS_Documents_2/pragathi/3rd sem CSE 2023-2024/R programs/
r")
Pearson correlation coefficient:  0.9341638
Spearman correlation coefficient:  0.9534986
>
```



Program 7:

Develop R program to create a Data Frame with following details and do the following operations.

itemCode	itemCategory
1001	Electronics
1002	Desktop Supplies
1003	Office Supplies
1004	USB
1005	CD Drive

Subset the Data frame and display the details of only those items whose price is greater than or equal to 350.

Subset the Data frame and display only the items where the category is either "Office Supplies" or

"Desktop Supplies"

Create another Data Frame called "item-details" with three different fields itemCode, ItemQtyonHand and ItemReorderLvl and merge the two frames

```
# Creating the data frame
itemCode <- c(1001:1005)#, 1002, 1003, 1004, 1005)
itemCategory <- c("Electronics", "Desktop Supplies", "Office Supplies", "USB", "CD Drive")
itemPrice <- c(700, 300, 350, 400, 800)

# Creating the data frame using the above vectors
items_df <- data.frame(itemCode, itemCategory, itemPrice)
```

```
# Displaying the created data frame
print("Data Frame with Items Information:")
print(items_df)
 # Summary statistics of itemPrice
 print(summary(items_df$itemPrice))


# Filtering items with price greater than 350
high_priced_items <- subset(items_df, itemPrice > 350)
print("Items with Price greater than 400:")
print(high_priced_items)


# Subset the data frame for items with category as "Office Supplies" or "Desktop Supplies"
filtered_items <- subset(items_df, itemCategory %in% c("Office Supplies", "Desktop
Supplies"))


# Display the subsetted data frame
print("Items with 'Office Supplies' or 'Desktop Supplies' category:") print(filtered_items)
 # Creating the 'item-details' data frame
itemCode <- c(1001, 1002, 1003, 1004, 1005)
ItemQtyonHand <- c(20, 15, 30, 10, 25)
ItemReorderLvl <- c(5, 10, 8, 3, 7)


# Creating the data frame using the above vectors
item_details <- data.frame(itemCode, ItemQtyonHand, ItemReorderLvl)


# Displaying the created data frame
 print("Data Frame 'item-details':")
```

```
print(item_details)

 # Merging the two data frames based on 'itemCode'

merged_data <- merge(items_df, item_details, by = "itemCode")


# Displaying the merged data frame

print("Merged Data Frame:")

print(merged_data)
```

OUTPUT:

```
>
> source("~/.active-rstudio-document")
[1] "Data Frame with Items Information:"
  itemCode     itemCategory itemPrice
1     1001      Electronics       700
2     1002 Desktop Supplies       300
3     1003  Office Supplies       350
4     1004              USB       400
5     1005         CD Drive       800
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    300     350     400     510     700     800
[1] "Items with Price greater than 400:"
  itemCode itemCategory itemPrice
1     1001  Electronics       700
4     1004          USB       400
5     1005     CD Drive       800
[1] "Items with 'Office Supplies' or 'Desktop Supplies' c
ategory:"
  itemCode     itemCategory itemPrice
2     1002 Desktop Supplies       300
3     1003  Office Supplies       350
```

```
[1] "Data Frame 'item-details':"
  itemCode ItemQtyonHand ItemReorderLvl
1     1001            20              5
2     1002            15             10
3     1003            30              8
4     1004            10              3
5     1005            25              7
[1] "Merged Data Frame:"
  itemCode     itemCategory itemPrice ItemQtyonHand ItemR
eorderLvl
1     1001      Electronics       700            20
5
2     1002 Desktop Supplies       300            15
10
3     1003  Office Supplies       350            30
8
4     1004              USB       400            10
3
5     1005         CD Drive       800            25
7
>
```

Program 8:

Let us use the built-in dataset air quality which has Daily air quality measurements in New York, May to September 1973. Develop R program to generate histogram by using appropriate arguments for the following statements.

a) Assigning names, using the air quality data set.

b) Change colors of the Histogram

c) Remove Axis and Add labels to Histogram

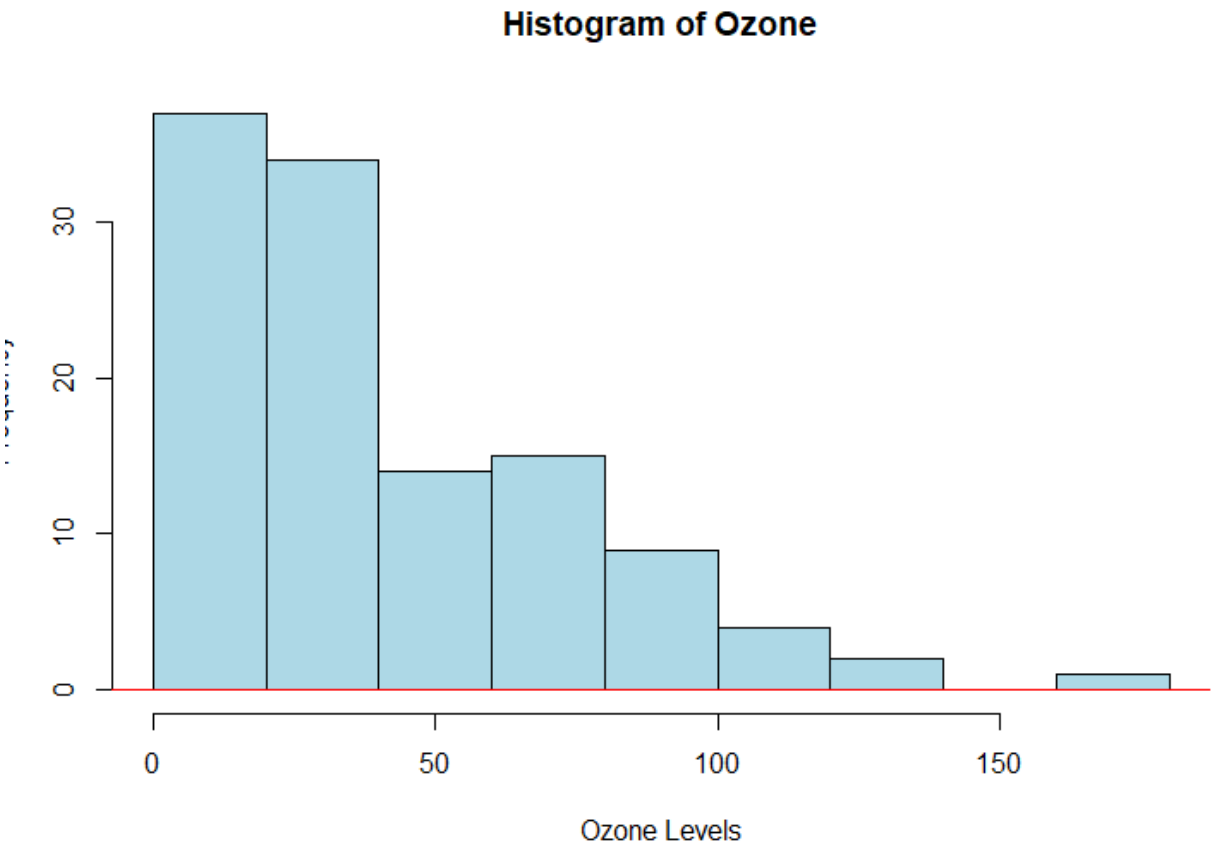
d) Change Axis limits of a Histogram

e) Add Density curve to the histogram.

```
# Load the air quality dataset

data("airquality", package = "datasets")


# a) Assigning names names

(airquality) <- c("Ozone", "Solar.R", "Wind", "Temp", "Month", "Day")


# b) Change colors of the Histogram

hist(airquality$Ozone, col = "skyblue", main = "Histogram of Ozone", xlab = "Ozone
Levels", ylab = "Frequency")


# c) Remove Axis and Add labels to Histogram

hist(airquality$Ozone, col = "lightgreen", main = "", xlab = "", ylab = "", axes = FALSE)

title(xlab = "Ozone Levels", ylab = "Frequency", main = "Histogram of Ozone")


# d) Change Axis limits of a Histogram

hist(airquality$Ozone, col = "lightpink", main = "Histogram of Ozone", xlab = "Ozone
Levels", ylab = "Frequency", xlim = c(0, 150), ylim = c(0, 40))
```

```
# e) Add Density curve to the histogram

# Remove missing values in 'Ozone' column

cleaned_data <- na.omit(airquality$Ozone)


# Create a histogram of 'Ozone' column

hist(cleaned_data, col = "lightblue", main = "Histogram of Ozone", xlab = "Ozone

Levels", ylab = "Frequency")


# Add a density curve to the histogram

lines(density(cleaned_data), col = "red")
```

OUTPUT:



Program 9:

Design a data frame in R for storing about 20 employee details. Create a CSV file named "input.csv" that defines all the required information about the employee such as id, name, salary, start_date, dept. Import into R and do the following analysis.

a) Find the total number rows & columns

b) Find the maximum salary

c) Retrieve the details of the employee with maximum salary

d) Retrieve all the employees working in the IT Department.

e) Retrieve the employees in the IT Department whose salary is greater than 20000 and write these details into another file "output.csv"

```
# Importing the CSV file into R

emp_data <- read.csv("empdata.csv")


# a) Find the total number rows & columns

num_rows <- nrow(emp_data)

num_cols <- ncol(emp_data)

cat("Total number of rows:", num_rows, "\n")

cat("Total number of columns:", num_cols, "\n")


# b) Find the maximum salary

max_salary <- max(emp_data$salary)

cat("Maximum salary:", max_salary, "\n")


# c) Retrieve the details of the employee with maximum salary employee_max_salary <-
emp_data[emp_data$salary == max_salary, ] cat("Employee with maximum salary:\n")

print(employee_max_salary)
```

d) Retrieve all the employees working in the IT Department

emp_IT <- subset(emp_data, dept == "IT")

cat("Employees working in the IT Department:\n")

print(emp_IT)

e) Retrieve the employees in the IT Department whose salary is greater than 20000

employees_IT_high_salary <- subset(emp_IT, salary > 20000) cat("Employees in the IT Department with salary > 20000:\n") print(employees_IT_high_salary)

Write these employees to a new CSV file write.csv(employees_IT_high_salary, "output.csv", row.names = FALSE)

OUTPUT:

```
> source("D:/23-24/R Programming/r programs/9.R")
Total number of rows: 20
Total number of columns: 5
Maximum salary: 77131
Employee with maximum salary:
   id        name salary start_date dept
11 11 Employee 11  77131 2017-12-16   IT
Employees working in the IT Department:
   id        name salary start_date dept
1   1  Employee 1  76662 2020-12-08   IT
3   3  Employee 3  54924 2010-08-05   IT
9   9  Employee 9  63937 2020-10-11   IT
11 11 Employee 11  77131 2017-12-16   IT
12 12 Employee 12  56312 2021-08-19   IT
15 15 Employee 15  52167 2010-02-10   IT
Employees in the IT Department with salary > 20000:
   id        name salary start_date dept
1   1  Employee 1  76662 2020-12-08   IT
3   3  Employee 3  54924 2010-08-05   IT
9   9  Employee 9  63937 2020-10-11   IT
11 11 Employee 11  77131 2017-12-16   IT
12 12 Employee 12  56312 2021-08-19   IT
15 15 Employee 15  52167 2010-02-10   IT
>
```

Program 10 :

Using the built in dataset mtcars which is a popular dataset consisting of the design and fuel consumption patterns of 32 different automobiles. The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models). Format A data frame with 32 observations on 11 variables : [1] mpg Miles/(US) gallon, [2] cyl Number of cylinders [3] disp Displacement (cu.in.), [4] hp Gross horsepower [5] drat Rear axle ratio,[6] wt Weight (lb/1000) [7] qsec 1/4 mile time, [8] vs V/S, [9] am Transmission (0 = automatic, 1 = manual), [10] gear Number of forward gears, [11] carb Number of carburetors. Develop R program, to solve the following:

a) What is the total number of observations and variables in the dataset?

b) Find the car with the largest hp and the least hp using suitable functions

c) Plot histogram / density for each variable and determine whether continuous variables are normally distributed or not. If not, what is their skewness?

d) What is the average difference of gross horse power(hp) between automobiles with 3 and 4 number of cylinders(cyl)? Also determine the difference in their standard deviations.

e) Which pair of variables has the highest Pearson correlation?

```
# Load the mtcars dataset data(mtcars)

# a) Total number of observations and variables observations <- nrow(mtcars) variables <-
ncol(mtcars) cat("Total number of observations:", observations, "\n") cat("Total number of
variables:", variables, "\n")


 # b) Car with the largest and least hp car_largest_hp <- mtcars[which.max(mtcars$hp), ]
car_least_hp <- mtcars[which.min(mtcars$hp), ] cat("Car with the largest hp:\n")
print(car_largest_hp) cat("Car with the least hp:\n") print(car_least_hp)


 # c) Histogram / density plot and skewness par(mfrow = c(4, 3), mar = c(3, 3, 1, 1))  #
Adjusting margin size for (i in 1:ncol(mtcars)) {   hist(mtcars[, i], main = names(mtcars)[i],
xlab = "", col = "skyblue")   lines(density(mtcars[, i]), col = "red")  # Adding density curve }
```

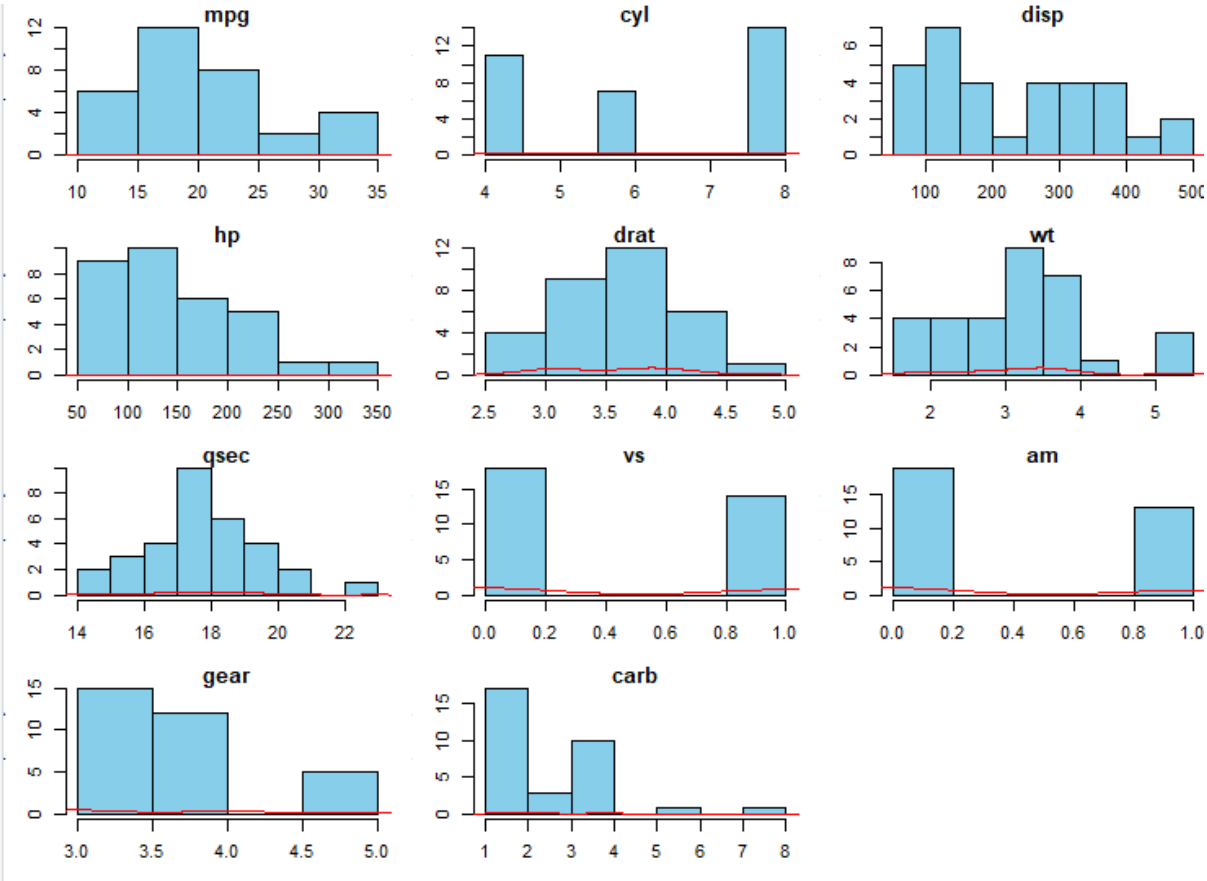
Calculate skewness library(e1071) skew <- sapply(mtcars, skewness) cat("Skewness of variables:\n") print(skew)

# d) Average difference and standard deviation between hp for 3 and 4 cylinders
hp_diff_mean <- mean(mtcars$hp[mtcars$cyl == 3]) - mean(mtcars$hp[mtcars$cyl == 4])
hp_diff_sd <- sd(mtcars$hp[mtcars$cyl == 3]) - sd(mtcars$hp[mtcars$cyl == 4])
cat("Average difference in hp between 3 and 4 cylinders:", hp_diff_mean, "\n")
cat("Difference in standard deviations of hp between 3 and 4 cylinders:", hp_diff_sd, "\n")

e) Pair of variables with the highest Pearson correlation cor_matrix <- cor(mtcars) diag(cor_matrix) <- 0 # Exclude diagonal values max_corr <- which(cor_matrix == max(cor_matrix), arr.ind = TRUE)

cat("Pair of variables with the highest Pearson correlation:", rownames(cor_matrix)[max_corr[1,1]], "and", colnames(cor_matrix)[max_corr[1,2]], "\n")

OUTPUT:



Program 11:

Demonstrate the progression of salary with years of experience using a suitable data set (You can create your own dataset). Plot the graph visualizing the best fit line on the plot of the given data points. Plot a curve of Actual Values vs. Predicted values to show their correlation and performance of the model.

Interpret the meaning of the slope and y-intercept of the line with respect to the given data. Implement using lm function. Save the graphs and coefficients in files. Attach the predicted values of salaries as a new column to the original data set and save the data as a new CSV file.

```
# Generating sample data for demonstration

set.seed(123)

Years_of_Experience <- 1:40

Salary <- 30000 + 1500 * Years_of_Experience + rnorm(40, mean = 0, sd = 2000)  #

Generating salaries

# Create a data  frame with the generated data

data <- data.frame(Years_of_Experience, round(Salary, digits = -1 ))

# Perform linear regression using lm() function

model <- lm(Salary ~ Years_of_Experience, data = data)

# Plot the data points and the best-fit line

plot(Salary ~ Years_of_Experience, data, col = "blue", main = "Salary vs. Years of
Experience")

abline(model, col = "red")

# Save the plot as an image file (e.g., PNG)

png("Salary_Experience_Plot03.png")

plot(Salary ~ Years_of_Experience, data, col = "blue", main = "Salary vs. Years of
Experience")

abline(model, col = "red")

dev.off()
```

```
# Generate predicted values from the model

predicted_values <- round( predict(model), digits=0)

# Plotting Actual vs. Predicted values

plot(Salary, predicted_values, main = "Actual vs. Predicted Salaries", xlab = "Actual Salary",
ylab = "Predicted Salary", col = "green")

# Save the plot as an image file (e.g., PNG)

jpeg("Actual_vs_Predicted_Salary.jpg")

plot(Salary, predicted_values, main = "Actual vs. Predicted Salaries", xlab = "Actual Salary",
ylab = "Predicted Salary", col = "green")

dev.off()

# Attach predicted values as a new column to the original dataset

data$Predicted_Salary <- predicted_values

# Save the dataset as a new CSV file

write.csv(data, "Salary_Experience_Predictions.csv", row.names = FALSE)
```

OUTPUT:

VIVA QUATION AND ANSWERS

1. Explain what is R?

R is data analysis software which is used by analysts, quants, statisticians, data scientists and others.

2. What is R programming and what are main feature of R?

R Programming Language is an open-source programming language that is widely used as a statistical software and data analysis tool. R generally comes with the Command-line interface. R is available across widely used platforms like Windows, Linux, and macOS. Also, the R programming language is the latest cutting-edge tool.
R programming is used as a leading tool for machine learning, statistics, and data analysis. Objects, functions, and packages can easily be created by R.

The main features of R:
Open source
Extensible
Statistical computing
Reproducibility
Graphics

3. List out some of the function that R provides?

The function that R provides are

- Mean
- Median
- Distribution
- Covariance
- Regression
- Non-linear
- Mixed Effects
- GLM
- GAM. etc.

4. Explain how you can start the R commander GUI?

Typing the command, ("Rcmdr") into the R console starts the R commander GUI.

5. In R how you can import Data?

You use R commander to import Data in R, and there are three ways through which you can enter data into it

You can enter data directly via Data New Data Set
Import data from a plain text (ASCII) or other files (SPSS, Minitab, etc.)
Read a data set either by typing the name of the data set or selecting the data set in the dialog box

6. Mention what does not 'R' language do?

Though R programming can easily connects to DBMS is not a database
R does not consist of any graphical user interface
Though it connects to Excel/Microsoft Office easily, R language does not provide any spreadsheet view of data.

7. Explain how R commands are written?

In R, anywhere in the program you have to preface the line of code with a #sign, for example

subtraction
division
note order of operations exists

8. How can you save your data in R?

To save data in R, there are many ways, but the easiest way of doing this is
Go to Data > Active Data Set > Export Active Data Set and a dialogue box will appear, when you click ok the dialogue box let you save your data in the usual way.

9. Mention how you can produce co-relations and covariances?

You can produce co-relations by the cor () function to produce co-relations and cov () function to produce covariances.

10. Explain what is t-tests in R?

In R, the t.test () function produces a variety of t-tests. T-test is the most common test in statistics and used to determine whether the means of two groups are equal to each other.

11. What are the data structures in R that is used to perform statistical analyses and create graphs?

R has data structures like
Vectors
Matrices
Arrays
Data frames

12. What is a data frame?

A data frame is made up of rows and columns, where each row denotes an observation or record and each column a variable or attribute. A data frame's columns can include a variety of data kinds, including logical, character, factor, and numeric ones, enabling the storing and management of the data.

13. How to create a data frame in R?

Use the data.frame() function in R to build a data frame. A two-dimensional tabular data structure called a data frame divides data into rows and columns. A data frame's columns can each contain a different data type, such as a logical, character, factor, or numeric one.

14. What is the function used for adding datasets in R?

rbind function can be used to join two data frames (datasets). The two data frames must have the same variables, but they do not have to be in the same order.

15**. Definitions**

- A vector is simply a list of items that are of the same type.
- A list in R can contain many different data types inside it. A list is a collection of data which is ordered and changeable.
  To create a list, use the list() function
- A matrix is a two dimensional data set with columns and rows.
  A column is a vertical representation of data, while a row is a horizontal representation of data.
  A matrix can be created with the matrix() function. Specify the nrow and ncol parameters to get the amount of rows and columns
- Compared to matrices, arrays can have more than two dimensions.
  We can use the array() function to create an array, and the dim parameter to specify the dimensions
- Data Frames are data displayed in a format as a table.
- Data Frames can have different types of data inside it. While the first column can be character, the second and third can be numeric or logical. However, each column should have the same type of data.
  Use the data.frame() function to create a data frame: