UNIT I

2 Marks:

1. What are the two ways of assignment operators in R? Give an example.

Ans: In R, there are two ways of assignment operators: **<-** and =

For example:

 Using <-: x <- 5 assigns the value 5 to the variable x.

 Using =: y = 10 assigns the value 10 to the variable y.

2. What is vector? Give an example to create a vector?

Ans: A vector in R is a one-dimensional array that can contain elements of the **same data type.** To create a vector, you can use the **c()** function.

For example:

 Numeric vector: my_vector <- c(1, 2, 3, 4, 5)

 Character vector: fruits <- c("apple", "banana", "cherry")

3. How do you find length of a vector? Give an example.

Ans: To find the length of a vector in R, you can use the **length()** function.

For example:

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

 length(vec) will return 5.

4. How do you sort a vector in descending order? Give an example

Ans: To sort a vector in descending order in R, you can use the **sort()** function with the **decreasing parameter set to TRUE**.

For example:

1. vec <- c(5, 2, 8, 1, 3)

2. sorted_vec <- sort(vec, decreasing = TRUE) will give you a vector sorted in descending order.

5. Create and store a sequence of values from 5 to −11 that progresses in steps of 0.3

Ans: To create and store a sequence of values in R, you can use the **seq()** function.

Here's how you can create a sequence from 5 to -11 in steps of 0.3:

 my_sequence <- seq(5, -11, by = -0.3)

6. Let vector, myvect with elements 5, -3,4,4,4,8,10,40221, -8,1. Write code to

delete last element from it.

Ans: To delete the last element from a vector in R, you can use the **indexing with a negative value**. For example:

 my_vect <- c(5, -3, 4, 4, 4, 8, 10, 40221, -8, 1)

 my_vect <- my_vect[-length(my_vect)] removes the last element.

7. Write the purpose of negative indexing in vectors? Give an example.

Negative indexing in vectors is used to **delete(exclude)** elements from the vector.

For example:

 my_vector <- c(1, 2, 3, 4, 5)

 my_vector[-3] will return a vector without the third element, which is 3.

8. If baz <- c(1,-1,0.5,-0.5) and qux <- 3, find the value of baz+quax.

Ans: To find the value of baz + qux you can do:

 baz <- c(1, -1, 0.5, -0.5)

 qux <- 3

 result <- baz + qux will add 3 to each element of the baz vector.

9. What is the use of cbind and rbind functions in Matrix? Give an example

Ans: cbind and rbind are functions in R used to combine vectors or matrices into a matrix.

 cbind combines objects **by columns**,

rbind combines objects **by rows**. For example:

 x <- c(1, 2, 3)

 y <- c(4, 5, 6)

 combined <- cbind(x, y) will create a 3x2 matrix with x and y as columns.

combined <- rbind(x, y) will create a 3x2 matrix with x and y as rows.

10.How do you find the dimension of the matrix? Give an example

Ans: To find the dimension of a matrix in R, you can use the **dim()** function. For example:

 mat <- matrix(1:6, nrow = 2, ncol = 3)

 dim(mat) #will return the dimensions as 2 3, indicating it's a 2x3 matrix.

11.Construct a 4 × 2 matrix that is filled row-wise with the values 4.3, 3.1, 8.2, 8.2, 3.2, 0.9, 1.6, and 6.5, in that order using R command.

Ans: To construct a 4x2 matrix filled row-wise with the specified values, you can use the **matrix()** function in R. Here's how you can create it:

 mat <- matrix(c(4.3, 3.1, 8.2, 8.2, 3.2, 0.9, 1.6, 6.5), nrow = 4, ncol = 2, **byrow = TRUE**)

12.What is the use of diag command in R? Give an example.

Ans: The diag() function in R is used to create a diagonal matrix or extract the diagonal elements from a matrix. For

example:

 To create a diagonal matrix: diag_matrix <- diag(1:4)

 To extract the diagonal elements from a matrix:

mat<-matrix(1:9,nrow=3)

diagonal_elements <- diag(mat)

13.Write proper code to replace the third column of matrix B with the values in the third row of B.

Ans: To replace the third column of matrix B with the values from the third row of B, you can use indexing as follows:

 B[, 3] <- B[3,]

14.Write an example to find transpose and inverse of a matrix using R command?

Ans: To find the transpose and inverse of a matrix in R:

mat<-matrix(1:9,nrow=3)

 Transpose: t(mat) #where mat is your matrix.

 Inverse: solve(mat) #where mat is a square, invertible matrix.

15.What is the difference between & and && in R? Give an example.

Ans: In R, & is a bitwise AND operator, and && is a logical AND operator. Here's an example:

 a <- TRUE

 b <- FALSE

 a & b will return FALSE, and a && b will also return FALSE.

16.Write R command to store the vector c(8,8,4,4,5,1,5,6,6,8) as bar. Identify the elements less than or equal to 6 AND not equal to 4.

Ans: bar <- c(8, 8, 4, 4, 5, 1, 5, 6, 6, 8)

result <- bar <= 6 & bar != 4

17.How do you count the number of individual characters in a string? Give an example.

Ans: To count the number of individual characters in a string in R, you can use the nchar() function. For example:

 text <- "Hello, World!"

 char_count <- nchar(text) will return 13.

18.What is levels function in R? Give an example

Ans: The levels() function in R is used with factors to get the levels distinct categories of a factor variable. For example:

 gender <- factor(c("Male", "Female", "Male", "Male", "Female"))

 gender_levels <- levels(gender) #will return a vector of levels: "Female" "Male".

19.What is list in R? Give an example.

list in R is a data structure that can hold a collection of objects of different types. For

example:

 my_list <- list(name = "John", age = 30, scores = c(85, 90, 78))

20.What is list slicing in lists? Give an example.

Ans: List slicing can be done using indexing and subsetting. For example:-

```
my_list <- list("apple", "orange", "banana", "grape", "melon")

subset <- my_list[2:4]   # Extracting a subset of elements from index 2 to
4 (inclusive)
print(subset)
```

21.How do you name list contents? Give an example.

Ans: You can name list contents in R by using the **names()** function or directly when creating the list. For example:

 Using names(): my_list <- list(1, 2, 3); names(my_list) <- c("A", "B", "C")

 Directly when creating: my_list <- list(A = 1, B = 2, C = 3)

22.What is the purpose of attributes and class functions? Give an example.

Ans: The attributes() function is used to attach metadata or additional information to R objects. The class() function specifically deals with the class attribute, determining the type or category of an object in R. For example:

x <- c(1, 2, 3, 4, 5)

To Check the class attribute of the vector

class(x) # Output will be "numeric"

To Check all attributes associated with the vector

attributes(x) # Output will show details about the vector's attributes

23.What is the difference between ggplot2 and base R graphics create plots? Give an example.

Ans: ggplot2 is a popular data visualization package in R that provides a more flexible and layered approach to creating plots compared to base R graphics.

ggplot2 allows you to build complex, customized plots. Base R graphics are more simplistic and are created using functions like plot(). Here's a simple example:

```
library(ggplot2)

ggplot(data = df, aes(x = category, y = values)) + geom_bar(stat = "identity")

df <- data.frame(category = c("A", "B", "C"), values = c(10, 15, 8))

barplot(df$values, names.arg = df$category)
```

4 And 6 marks

1. Explain seq, rep and length functions on vectors with example.

Ans:

seq: The seq() function in R is used to create sequences of numbers. It can be used to generate a sequence from a starting value to an ending value, with a specified increment.
For example:

```
sequence <- seq(1, 10, by = 2)                # Creates a sequence: 1, 3, 5, 7, 9
```

rep: The rep() function is used to replicate elements in a vector.
It takes two main arguments: the vector to be replicated and the number of times to repeat it.
For example:

```
vector <- c(1, 2, 3)
repeated_vector <- rep(vector, times = 3)
# Repeats the vector three times: 1, 2, 3, 1, 2, 3, 1, 2, 3
```

length: The length() function is used to determine the length of a vector, which is the number of elements it contains. For example:

```
my_vector <- c(4, 7, 2, 9, 5)
vector_length <- length(my_vector)                # Returns the length of the vector, which is 5
```

2. Repeat the vector c (-1,3, -5,7, -9) twice, with each element repeated 10 times,

and store the result. Display the result sorted from largest to smallest.

Ans:

```
original_vector <- c(-1, 3, -5, 7, -9)

repeated_vector <- rep(original_vector, times = 2, each = 10)
```

The above code will Repeat the vector twice, each element repeated 10 times

```
sorted_vector <- sort(repeated_vector, decreasing = TRUE)     # Sort the repeated vector from largest to smallest
print(sorted_vector)        #It will Display the sorted vector
```

3. How do you extract elements from vectors? Explain it using individual and vector

of indexes with example?

Ans: **Individual Indexing:** You can extract elements from a vector by specifying the index of the element you want.

For example:

```
my_vector <- c(10, 20, 30, 40, 50)
```

element1 <- my_vector[1] # Extracts the first element (10)

element3 <- my_vector[3] # Extracts the third element (30)

Vector of Indexes: You can extract multiple elements by providing a vector of indices. For example:

indices <- c(2, 4)

selected_elements <- my_vector[indices] # Extracts the 2nd and 4th elements (20, 40)

4. How do you create matrix in R? Explain with Its necessary attributes? Give an example.

Ans: To create a matrix in R, you can use the **matrix()** function. The necessary attributes are the data elements, the number of rows, and the number of columns.

Example:

mat <- matrix(data=c(1:12), nrow = 3, ncol = 4, byrow = TRUE)

 data: The vector containing the data elements.

 nrow: The number of rows in the matrix (3 in this example).

 ncol: The number of columns in the matrix (4 in this example).

 byrow: If byrow = TRUE, data is filled row-wise; if byrow = FALSE (default), data is filled column-wise.

5. Do the following operations on a square matrix

a. Retrieve third and first rows of A, in that order, and from those rows, second and third column elements.

b. Retrieve diagonal elements

c. Delete second column of the matrix.

Ans:

a. Retrieve third and first rows of A, in that order, and from those rows, second and third column elements:

A <- matrix(c(1:9), nrow = 3)

rows <- c(3, 1)

cols <- c(2, 3)

result <- A[rows, cols]

b. diagonal_elements <- diag(A)

c. A <- A[, -2]

6. Explain row, column, and diagonal extractions of matrix elements with example.

Ans:
Row extraction: You can extract specific rows from a matrix by specifying the row indices. For example:
mat <- matrix(1:12, nrow = 3)
row2 <- mat[2,] # Extracts the second row
Column extraction: You can extract specific columns from a matrix by specifying the column indices. For example:
mat <- matrix(1:12, nrow = 3)
col3 <- mat[, 3] # Extracts the third column
Diagonal extraction: To extract the diagonal elements of a matrix, you can use the **diag()** function. For example:
mat <- matrix(1:9, nrow = 3)
diagonal_elements <- diag(mat) # Extracts the diagonal elements

7. How do you omit and overwrite an element/s from a matrix? Explain with example.

Ans:

To omit and overwrite elements in a matrix, you can simply assign new values to the elements you want to modify. For example, to omit the element in the second row and third column and overwrite an element:
mat <- matrix(1:9, nrow = 3)
mat[2, 3] <- NA # Omit the element
mat[1, 1] <- 99 # Overwrite an element

8. Explain any 3 matrix operations using R commands with example.

Ans: Matrix Addition:
A <- matrix(1:6, nrow = 2)
B <- matrix(7:12, nrow = 2)
result <- A + B # Matrix addition
Matrix Multiplication:
A <- matrix(1:6, nrow = 2)
B <- matrix(7:12, nrow = 2)
result <- A %*% B # Matrix multplication
Matrix Transpe:
A <- matrix(1:6, nrow = 2)
transposed_A <- t(A) # Transpose matrix A

9. How do you create arrays in R? Explain with suitable example

Ans:
You can create arrays in R using the array() function. An array is a multi-dimensional structure that can hold data of the same type. Here's an example of creating a 3x3x3 array:
data <- 1:27 # Vector with 27 elements
my_array <- array(data, dim = c(3, 3, 3))

10.Explain any 3 relational and logical operators used in R, with an example.

ANS:

a. Relational operators:

 ==: Checks if two values are equal. Example: x == y returns TRUE if x is equal to y.
 !=: Checks if two values are not equal. Example: x != y returns TRUE if x is not equal to y.
 >: Checks if the leŌ value is greater than the right value. Example: x > y returns TRUE if x is greater
than y.

b. Logical operators:

 &: Performs element-wise logical AND. Example: x & y returns a vector of logical values with TRUE
where both x and y are TRUE.
 |: Performs element-wise logical OR. Example: x | y returns a vector of logical values with TRUE
where either x or y is TRUE.
 !: Negates the logical values in a vector. Example: !x returns a vector with the opposite logical values
of x.

11.Explain any, all and which functions with example, on logical vector.

Ans: **any():** The any() function returns TRUE if at least one element in a logical vector is TRUE. For example:

```
logical_vector <- c(TRUE, FALSE, FALSE)
result <- any(logical_vector) # Returns TRUE
```

all(): The all() function returns TRUE if all elements in a logical vector are TRUE. For example

```
logical_vector <- c(TRUE, TRUE, TRUE)
result <- all(logical_vector) # Returns TRUE
```

which(): The which() function returns the indices of elements in a logical vector that are TRUE. For example:

```
logical_vector <- c(TRUE, FALSE, TRUE, TRUE, FALSE)
indices <- which(logical_vector) # Returns indices 1, 3, 4
```

12.Explain cat and paste functions with necessary arguments in R. Give an example.

Ans:

cat: The cat() function in R is used to concatenate and print values together. It prints the values without any separators by default.

```
cat("Hello", "World")
# Output: Hello World
```

paste: The paste() function is used to concatenate strings or other objects with a specified separator.

```
result <- paste("Hello", "World", sep = ", ")
# Result: "Hello, World"
```

13.Write a note on escape sequences with example.

Ans:
Escape sequences are used in R to represent special characters within strings. They begin with a backslash \. For example:
 \n: Represents a newline character.
 \t: Represents a tab character.
 \": Represents a double quotation mark.

```
message <- "This is a line.\nThis is a new line."
print(message)
```

14.Explain substr, sub and gsub functions on strings with an example.

Ans:
substr(): The substr() function extracts a substring from a character vector. For example:

```
text <- "Hello, World"
substring <- substr(text, start = 1, stop = 5)
# Result: "Hello"
```

sub(): The sub() function is used for pattern replacement in strings. It replaces the first occurrence of a pattern with a replacement.

```
text <- "apple, banana, cherry"
replaced_text <- sub("banana", "grape", text)
# Result: "apple, grape, cherry"
```

gsub(): The gsub() function is similar to sub(), but it replaces all occurrences of a pattern

```
text <- "apple, banana, banana, cherry"
replaced_text <- gsub("banana", "grape", text)
# Result: "apple, grape, grape, cherry"
```

15.What is factor? How do you define and order levels in a factor?

Ans:

A factor is a data structure in R used to represent categorical variables. Factors are used to store data that can take on a limited, fixed number of values, known as levels. You can define and order levels in a factor as follows:

```
# Define a factor with levels in a specific order
gender <- factor(c("Male", "Female", "Male", "Male", "Female"), levels = c("Female", "Male"))
# To reorder levels
gender <- factor(gender, levels = c("Male", "Female"))
```

16.Explain cut function on factors with an example.
Ans:

The cut() function in R is used to categorize and split a continuous variable into intervals or bins. It is used with factors to create categorical variables. For example:

```
ages <- c(25, 40, 35, 60, 30, 45, 50, 22, 32, 55)
age_groups <- cut(ages, breaks = c(0, 25, 35, 45, 60), labels = c("Young", "Mid", "Mid-Old", "Old"))
```

In this example, cut() divides the ages vector into four age groups based on the specified breaks and assigns

labels to each group.

17.What do you mean by member reference in lists? Explain with an example.

Ans:

Member reference in lists refers to accessing or retrieving elements or values within a list by specifying the names or indexes of the elements. Lists in R are collections of elements, and each element can be referenced using the $ operator or the double square brackets [[]]. For example:

```
my_list <- list(name = "John", age = 30, scores = c(85, 90, 78))
# Using $ operator to access elements
name_value <- my_list$name
scores_value <- my_list$scores
# Using double square brackets to access elements
age_value <- my_list[["age"]]
```

In this example, my_list$name or my_list[["age"]] is used to access specific elements within the list.

18. What is data frame? Create a data frame as shown in the given table and write R commands
a) To extract the third, fourth, and fifth elements of the third column
b) To extract the elements of age column using dollar operator.

person	age	sex
Peter	42	M
Lois	40	F
Meg	17	F
Chris	14	M
Stewie	1	M

In R, a data frame is a two-dimensional structure that stores data in rows and columns, similar to a table in a database or a spreadsheet. Each column in a data frame can have a different data type, and it can store various types of data such as numeric, character, logical, etc.

```
# Creating the data frame
df <- data.frame(
  Person = c("Peter", "Lois", "Meg", "Chris", "Stewie"),
  Age = c(42, 40, 17, 14, 1),
  Sex = c("M", "F", "F", "M", "M")
)
# Viewing the created data frame
print(df)

# Extracting the third, fourth, and fifth elements of the third column
third_to_fifth_third_column <- df[3, 3:5]
print(third_to_fifth_third_column)
# Extracting the elements of the age column using the dollar operator
age_column <- df$Age
print(age_column)
```

19.How do you add data columns and combine data frames? Explain with example.

Ans:

You can add data columns to a data frame using the $ operator or by using functions like **cbind().** To combine data frames, you can use functions like **rbind() or merge**(). Here's an example:

```
# Create two data frames
df1 <- data.frame(Name = c("Alice", "Bob", "Charlie"), Age = c(25, 30, 22))
df2 <- data.frame(Score = c(85, 90, 78))
# Add a new column to df1
df1$Score <- df2$Score
# Combine data frames using rbind
combined_df <- rbind(df1, df2)
```

20.Write a note on special values used in R with an example for each.

Ans:

 NA: Stands for **"Not Available"** and is used to represent missing or undefined values. Example: x <-c(1, 2, NA, 4).

 NaN: Stands for **"Not a Number"** and is used to represent undefined or unrepresentable real numbers, often arising from calculations. Example: sqrt(-1).

 Inf and -Inf: Represent positive and negative infinity, respectively. Example: 1/0 and -1/0.

21.Explain Is-Dot Object-Checking Functions and As-Dot Coercion Functions with an example.

Ans:

 Is-Dot Object-Checking Functions: These functions check whether an object belongs to a specific class or type. For example, is.numeric(), is.character(), is.data.frame().

```
x <- "42"
is_numeric <- is.numeric(x) # Check if x is numeric
```

 As-Dot Coercion Functions: These functions are used to coerce or convert an object to a specific class or type. For example, as.numeric(), as.character().

```
as_numeric <- as.numeric(x) # Convert x to numeric
```

22.List and explain graphical parameters used in plot function in R with example (any 4)

Ans:

The plot() function in R allows you to specify various graphical parameters to customize the appearance of
the plot. Here are four graphical parameters:

 main: Sets the title of the plot.

 xlab: Sets the label for the x-axis.

 ylab: Sets the label for the y-axis.

 col: Sets the color of the plotted points or lines.

Example:

```
x <- 1:10
y <- x^2
```

```
plot(x, y, main = "QuadraticFunction", xlab = "X-axis", ylab = "Y-axis", col = "blue")
```

23.How do you add Points, Lines, and Text to an Existing Plot? Explain with example.

Ans:
You can add points, lines, and text to an existing plot using functions like points(), lines(), and text(). Here's
an example of adding points and text to an existing scatter plot:

```
x <- 1:5
y <- c(3, 6, 4, 9, 7)
plot(x, y, main = "Scatter Plot")
# Add points
points(x, y, col = "red", pch = 2)
# Add text
text(x, y, labels = y, pos = 3)
```

24.How do you set appearance constants and aesthetic mapping with geoms? Explain with example.

Ans: In the context of data visualization using ggplot2, appearance constants and aesthetic mappings can be set
using various geom_ functions. Appearance constants (such as color, size, and shape) define how the data elements
will look, and aesthetic mappings define how variables in your data map to visual elements. Here's an example using
geom_point():

```
library(ggplot2)
# Create a data frame
df <- data.frame(x = c(1, 2, 3, 4, 5), y = c(3, 6, 4, 9, 7), group = c("A", "B", "A", "B", "A"))
# Create a scatter plot with different colors for each group
ggplot(df, aes(x = x, y = y, color = group)) + geom_point(size = 4, shape = 21)
```