Reg.No.: 22MIC0019

Name: Rufina M

Slot: L3 + L4

Data Visualization Techniques

R PROGRAMMING SET – 1

AIM: To understand and practice the basics of R programming and it's data structures.

Q1. Create Data frames which contain details of 10 employees and display summary of the data

CODE:

22MIC0019_A1_Q1.R

Source on Save

```r
# Create employee data
emp_id <- 101:110
emp_name <- c("Rufina", "Alice", "Charlie", "David", "Eva",
              "Frank", "Grace", "Hannah", "Ian", "Judy")
emp_age <- c(28, 34, 25, 45, 30, 29, 41, 38, 27, 31)
emp_dept <- c("HR", "Finance", "IT", "IT", "HR",
              "Marketing", "Finance", "IT", "Marketing", "HR")
emp_salary <- c(45000, 55000, 60000, 75000, 48000,
                50000, 62000, 71000, 47000, 49000)

# Create data frame
employee_df <- data.frame(
  ID = emp_id,
  Name = emp_name,
  Age = emp_age,
  Department = emp_dept,
  Salary = emp_salary
)

# Print the data frame
print(employee_df)

# Display summary of the data
cat("\n--- Summary of Employee Data ---\n")
summary(employee_df)
```

OUTPUT:

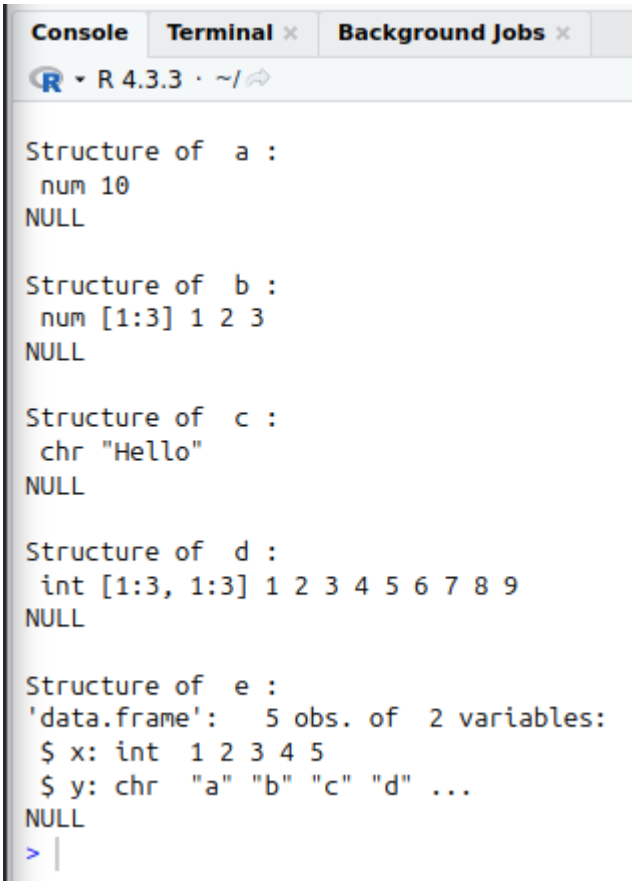
```
Console  Terminal  Background Jobs
R 4.5.1 · ~/
> emp_id <- 101:110
> emp_name <- c("Rufina", "Alice", "Charlie", "David", "Eva",
+               "Frank", "Grace", "Hannah", "Ian", "Judy")
> emp_age <- c(28, 34, 25, 45, 30, 29, 41, 38, 27, 31)
> emp_dept <- c("HR", "Finance", "IT", "IT", "HR",
+               "Marketing", "Finance", "IT", "Marketing", "HR")
> emp_salary <- c(45000, 55000, 60000, 75000, 48000,
+                 50000, 62000, 71000, 47000, 49000)
> # Create data frame
> employee_df <- data.frame(
+   ID = emp_id,
+   Name = emp_name,
+   Age = emp_age,
+   Department = emp_dept,
+   Salary = emp_salary
+ )
> # Print the data frame
> print(employee_df)
    ID    Name Age Department Salary
1  101  Rufina  28         HR  45000
2  102   Alice  34    Finance  55000
3  103 Charlie  25         IT  60000
4  104   David  45         IT  75000
5  105     Eva  30         HR  48000
6  106   Frank  29  Marketing  50000
7  107   Grace  41    Finance  62000
8  108  Hannah  38         IT  71000
9  109     Ian  27  Marketing  47000
10 110    Judy  31         HR  49000
> # Display summary of the data
> cat("\n--- Summary of Employee Data ---\n")

--- Summary of Employee Data ---
> summary(employee_df)
       ID              Name                Age         Department           Salary
 Min.   :101.0   Length:10          Min.   :25.00   Length:10          Min.   :45000
 1st Qu.:103.2   Class :character   1st Qu.:28.25   Class :character   1st Qu.:48250
 Median :105.5   Mode  :character   Median :30.50   Mode  :character   Median :52500
 Mean   :105.5                      Mean   :32.80                      Mean   :56200
 3rd Qu.:107.8                      3rd Qu.:37.00                      3rd Qu.:61500
 Max.   :110.0                      Max.   :45.00                      Max.   :75000
```

Q2. Write a R program to get the details of any 5 objects in memory
CODE:

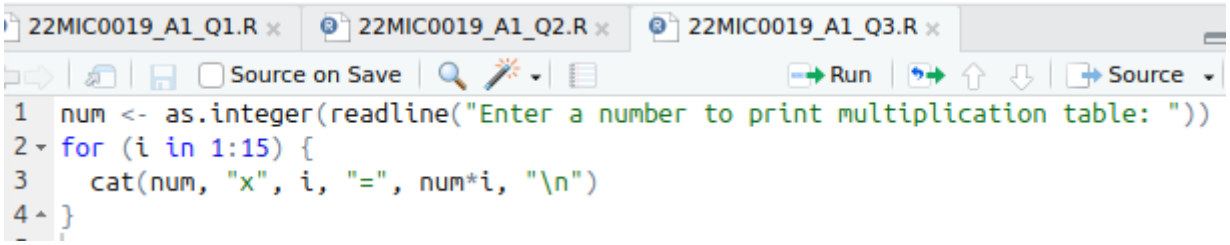
```
22MIC0019_A1_Q1.R  22MIC0019_A1_Q2.R
Source on Save
1  a <- 10
2  b <- c(1, 2, 3)
3  c <- "Hello"
4  d <- matrix(1:9, 3, 3)
5  e <- data.frame(x = 1:5, y = letters[1:5])
6
7  objects <- ls()
8  head(objects, 5)
9
10 for (obj in head(objects, 5)) {
11   cat("\nStructure of ", obj, ":\n")
12   print(str(get(obj)))
13 }
```

OUTPUT:



Q3. Write a R program to print the multiplication table of a number from 1 to 15.

CODE

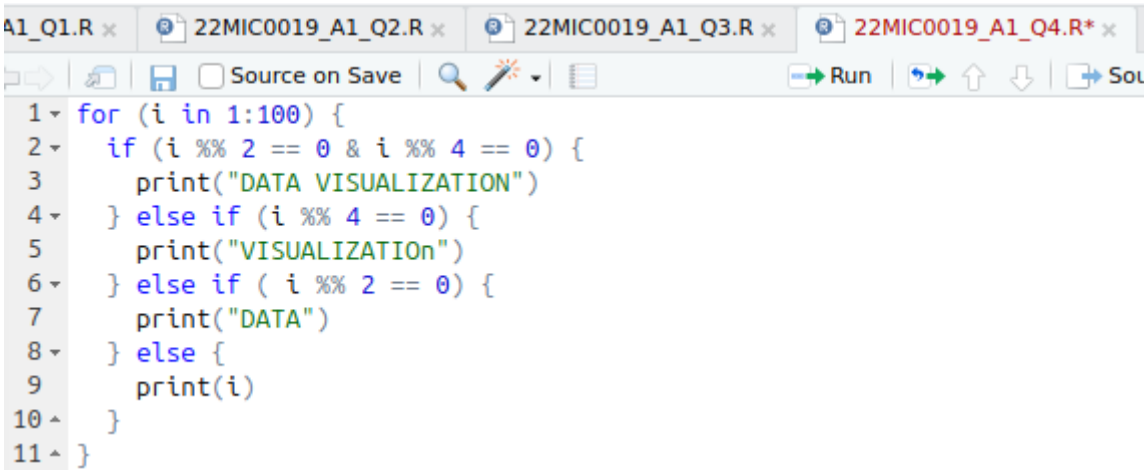


OUTPUT:

```
> num <- as.integer(readline("Enter a number to print multiplication table: "))
Enter a number to print multiplication table: 12
> for (i in 1:15) {
+   cat(num, "x", i, "=", num*i, "\n")
+ }
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120
12 x 11 = 132
12 x 12 = 144
12 x 13 = 156
12 x 14 = 168
12 x 15 = 180
```

Q4. Write a R program to print the numbers from 1 to 100 and print "DATA" for multiples of 2, print "VISUALIZATION" for multiples of 4, and print "DATA VISUALIZATION" for multiples of both.

CODE:



OUTPUT:

Console | Terminal | Background Jobs

R 4.3.3 · ~/

```
[1] 1
[1] "DATA"
[1] 3
[1] "DATA VISUALIZATION"
[1] 5
[1] "DATA"
[1] 7
[1] "DATA VISUALIZATION"
[1] 9
[1] "DATA"
[1] 11
[1] "DATA VISUALIZATION"
[1] 13
[1] "DATA"
[1] 15
[1] "DATA VISUALIZATION"
[1] 17
[1] "DATA"
[1] 19
[1] "DATA VISUALIZATION"
[1] 21
[1] "DATA"
[1] 23
[1] "DATA VISUALIZATION"
[1] 25
[1] "DATA"
[1] 27
[1] "DATA VISUALIZATION"
[1] 29
[1] "DATA"
[1] 31
[1] "DATA VISUALIZATION"
[1] 33
[1] "DATA"
[1] 35
[1] "DATA VISUALIZATION"
[1] 37
[1] "DATA"
[1] 39
[1] "DATA VISUALIZATION"
[1] 41
```

Console | Terminal | Background Jobs

R 4.3.3 · ~/

```
[1] 41
[1] "DATA"
[1] 43
[1] "DATA VISUALIZATION"
[1] 45
[1] "DATA"
[1] 47
[1] "DATA VISUALIZATION"
[1] 49
[1] "DATA"
[1] 51
[1] "DATA VISUALIZATION"
[1] 53
[1] "DATA"
[1] 55
[1] "DATA VISUALIZATION"
[1] 57
[1] "DATA"
[1] 59
[1] "DATA VISUALIZATION"
[1] 61
[1] "DATA"
[1] 63
[1] "DATA VISUALIZATION"
[1] 65
[1] "DATA"
[1] 67
[1] "DATA VISUALIZATION"
[1] 69
[1] "DATA"
[1] 71
[1] "DATA VISUALIZATION"
[1] 73
[1] "DATA"
[1] 75
[1] "DATA VISUALIZATION"
[1] 77
[1] "DATA"
[1] 79
[1] "DATA VISUALIZATION"
[1] 81
```

```
[1] 81
[1] "DATA"
[1] 83
[1] "DATA VISUALIZATION"
[1] 85
[1] "DATA"
[1] 87
[1] "DATA VISUALIZATION"
[1] 89
[1] "DATA"
[1] 91
[1] "DATA VISUALIZATION"
[1] 93
[1] "DATA"
[1] 95
[1] "DATA VISUALIZATION"
[1] 97
[1] "DATA"
[1] 99
[1] "DATA VISUALIZATION"
>
```

Q5 Create an empty factor vector, append values in it and find the sum, mean, product of vector elements using R. Also extract every nth element of the vector.

CODE:

```r
# 1. Creation of an empty factor vector
f <- factor()

# 2. Appending values (as characters initially, since factors are categorical)
f <- factor(c(f, "2", "4", "6", "8", "10"))

# 3. Convertion of the factor to numeric (correctly)
numeric_values <- as.numeric(as.character(f))

# 4. Performing sum, mean, product
sum_val <- sum(numeric_values)
mean_val <- mean(numeric_values)
prod_val <- prod(numeric_values)

# Printing the results
cat("Sum:", sum_val, "\n")
cat("Mean:", mean_val, "\n")
cat("Product:", prod_val, "\n")

# 5. Extracting every nth element (e.g., every 2nd element)
n <- 2
every_nth <- numeric_values[seq(n, length(numeric_values), by = n)]

cat("Every", n, "th element:", every_nth, "\n")
```

OUTPUT:

```
Output

Sum: 30
Mean: 6
Product: 3840
Every 2 th element: 4 8

=== Code Execution Successful ===
```

Q6. Use a nested for loop (a for loop inside a for loop) that produces the following matrix, pre-allocate the matrix with NA values.

0 1 2 3 4
1 0 1 2 3
2 1 0 1 2
3 2 1 0 1
4 3 2 1 0

CODE:

```r
#Define the size of the matrix
n <- 5

#Pre-allocate the matrix with NA values
mat <- matrix(NA, nrow = n, ncol = n)

#Fill the matrix using nested for loops
for (i in 1:n) {
  for (j in 1:n) {
    mat[i, j] <- abs(i - j)
  }
}

#Print the matrix
print(mat)
```

OUTPUT:

```
Console  Terminal  Background Jobs
R 4.5.1 · ~/
> print(mat)
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    1    2    3    4
[2,]    1    0    1    2    3
[3,]    2    1    0    1    2
[4,]    3    2    1    0    1
[5,]    4    3    2    1    0
>
```

Q7. Implement a multiplication game. A while loop that gives the user two random numbers from 2 to 12 and asks the user to multiply them. Only exit the loop after five correct answers.

CODE:

```r
# Multiplication Game

correct_answers <- 0

cat("Welcome to the Multiplication Game!\n")
cat("Answer 5 questions correctly to win.\n\n")

while (correct_answers < 5) {
  num1 <- sample(2:12, 1)
  num2 <- sample(2:12, 1)

  # Prompt the user
  user_input <- as.integer(readline(prompt = paste("What is", num1, "*", num2, "? ")))

  # Validate and check the answer
  if (!is.na(user_input) && user_input == num1 * num2) {
    correct_answers <- correct_answers + 1
    cat("Correct! Total correct answers: ", correct_answers, "\n\n")
  } else {
    cat("Incorrect. Try again!\n\n")
  }
}

cat("Congratulations! You got 5 correct answers.\n")
```

OUTPUT:

```
Console  Terminal  Background Jobs
R 4.5.1 · ~/
> correct_answers <- 0
> cat("Welcome to the Multiplication Game!\n")
Welcome to the Multiplication Game!
> cat("Answer 5 questions correctly to win.\n\n")
Answer 5 questions correctly to win.

> while (correct_answers < 5) {
+   num1 <- sample(2:12, 1)
+   num2 <- sample(2:12, 1)
+
+   # Prompt the user
+   user_input <- as.integer(readline(prompt = paste("What is", num1, "*", num2, "? ")))
+
+   # Validate and check the answer
+   if (!is.na(user_input) && user_input == num1 * num2) {
+     correct_answers <- correct_answers + 1
+     cat("Correct! Total correct answers: ", correct_answers, "\n\n")
+   } else {
+     cat("Incorrect. Try again!\n\n")
+   }
+ }
What is 11 * 7 ? 77
Correct! Total correct answers:  1

What is 11 * 7 ? 56
Incorrect. Try again!

What is 9 * 2 ? 18
Correct! Total correct answers:  2

What is 4 * 10 ? 40
Correct! Total correct answers:  3

What is 4 * 11 ? 45
Incorrect. Try again!

What is 9 * 5 ? 45
Correct! Total correct answers:  4

What is 4 * 12 ? 48
Correct! Total correct answers:  5
```

Q8. Using for loop simulate the flip a coin twenty times, keeping track of the individual outcomes (1 = heads, 0 = tails) in a vector that you pre-allocate.

CODE:

22MIC0019_A1_Q8.R

Source on Save

```r
# Set seed for reproducibility
set.seed(42)

# 1. Pre-allocate a vector to store results of 20 coin flips
coin_flips <- numeric(20)  # pre-allocate with zeros

# 2. Simulate 20 coin flips using a for loop
for (i in 1:20) {
  # Flip the coin: sample 0 or 1
  coin_flips[i] <- sample(c(0, 1), size = 1)
}

# 3. Print the outcomes
cat("Coin flip outcomes (1 = heads, 0 = tails):\n")
print(coin_flips)

# 4. Summary
cat("\nNumber of Heads:", sum(coin_flips), "\n")
cat("Number of Tails:", length(coin_flips) - sum(coin_flips), "\n")
```

OUTPUT:

Console | Terminal | Background Jobs

R 4.5.1 · ~/

```
> set.seed(42)
> # 1. Pre-allocate a vector to store results of 20 coin flips
> coin_flips <- numeric(20)  # pre-allocate with zeros
> # 2. Simulate 20 coin flips using a for loop
> for (i in 1:20) {
+   # Flip the coin: sample 0 or 1
+   coin_flips[i] <- sample(c(0, 1), size = 1)
+ }
> # 3. Print the outcomes
> cat("Coin flip outcomes (1 = heads, 0 = tails):\n")
Coin flip outcomes (1 = heads, 0 = tails):
> print(coin_flips)
 [1] 0 0 0 0 1 1 1 1 0 1 0 1 0 1 0 0 1 1 1 1
> # 4. Summary
> cat("\nNumber of Heads:", sum(coin_flips), "\n")

Number of Heads: 11
> cat("Number of Tails:", length(coin_flips) - sum(coin_flips), "\n")
Number of Tails: 9
>
```

Q9. Write a R program to know the first positive integer whose square exceeds 4000.

CODE:

22MIC0019_A1_Q9.R

Source on Save

```r
# Initialize the number
i <- 1

# while loop to find the first integer whose square exceeds 4000
while (i^2 <= 4000) {
  i <- i + 1
}

# Print
cat("The first positive integer whose square exceeds 4000 is:", i, "\n")
cat("Because", i, "^2 =", i^2, "\n")
```

OUTPUT:

12:1 (Top Level)

Console Terminal Background Jobs

R 4.5.1 · ~/

```
> i <- 1
> # while loop to find the first integer whose square exceeds 4000
> while (i^2 <= 4000) {
+   i <- i + 1
+ }
> # Print
> cat("The first positive integer whose square exceeds 4000 is:", i, "\n")
The first positive integer whose square exceeds 4000 is: 64
> cat("Because", i, "^2 =", i^2, "\n")
Because 64 ^2 = 4096
>
```

Q10. Write a R program to create a vector of a specified type and length. Create vector of numeric, complex, logical and character types of length 6.

CODE:

22MIC0019_A1_Q10.R

Source on Save

```r
# 1. Create a numeric vector of length 6 (initialized to 0)
numeric_vec <- numeric(6)

# 2. Create a complex vector of length 6 (initialized to 0+0i)
complex_vec <- complex(length = 6)

# 3. Create a logical vector of length 6 (initialized to FALSE)
logical_vec <- logical(6)

# 4. Create a character vector of length 6 (initialized to "")
character_vec <- character(6)

# Print all vectors
cat("Numeric Vector:\n")
print(numeric_vec)

cat("\nComplex Vector:\n")
print(complex_vec)

cat("\nLogical Vector:\n")
print(logical_vec)

cat("\nCharacter Vector:\n")
print(character_vec)
```

OUTPUT:

25:1 (Top Level)

Console | Terminal | Background Jobs

R 4.5.1 · ~/

```
> cat("Numeric Vector:\n")
Numeric Vector:
> print(numeric_vec)
[1] 0 0 0 0 0 0
> cat("\nComplex Vector:\n")

Complex Vector:
> print(complex_vec)
[1] 0+0i 0+0i 0+0i 0+0i 0+0i 0+0i
> cat("\nLogical Vector:\n")

Logical Vector:
> print(logical_vec)
[1] FALSE FALSE FALSE FALSE FALSE FALSE
> cat("\nCharacter Vector:\n")

Character Vector:
> print(character_vec)
[1] "" "" "" "" "" ""
```

NOTE:

In this assessment, the programming was done using the **R language** within the **RStudio** environment.
The tools utilized include the **Console** and **Script Editor** for writing, executing, and debugging code interactively.
The data structures used throughout the tasks include **vectors**, **matrices**, **data frames**, **factors**, and **logical constructs.**