SAVEETHA SCHOOL OF ENGINEERING

ITA0443-STATISTICS WITH R-PROGRAMMING

Name: B.Jaswanth Reddy

Reg no: 192110623

EXERCISE-1

1.Write The Commands To Perform Basic Arithmetic In R.

A) # addition

x <- 2 + 5

print(x)

OUTPUT = 7

# subtraction

y <- 8 – 2

print(y)

OUTPUT=6

# multiplication

z <- 2 \* 10

print(z)

OUTPUT=20

# division

a <- 12 / 2

print(a)

OUTPUT=6

2) Display a String on R Console

A)

message <- "saveetha school of engineering"

cat(message)

3) Declare Variables In R And Also Write The Commands For Retrieving The Value Of The Stored Variables In R Console?

A) x <- 5

y <- 10

z <- "saveetha"

To retrieve the value of the stored variables, simply type the variable name in the R console and press enter:

Output:

> x

[1] 5

> y

[1] 10

> z

[1] "saveetha"

4) Write R script to calculate the area of Rectangle?

A) Function to calculate the area of a rectangle

calculate\_area\_of\_rectangle <- function(length, width) {

area <- length \* width

return(area)

}

# example usage

length <- 8

width <- 14

area <- calculate\_area\_of\_rectangle(length, width)

print(paste("The area of the rectangle is", area))

output:

“The area of the rectangle is 112”

5) Write Commands In R Console To Determine The Type Of Variable

A) INPUT:

class(x)

OUTPUT:

"numeric"

This will return the class of **x** as a character string, such as **numeric**, **character**, **factor**, **data.frame**, etc

6) 6.Enumerate The Process To Check Whether A Given Input Is Numeric , Integer ,Double, Complex in R ?

A) INPUT:

x <- 10

is.numeric(x)

is.integer(x)

is.double(x)

is.complex(x)

OUTPUT:

[1] TRUE

[1] TRUE

[1] FALSE

[1] FALSE

7) Illustration of Vector Arithmetic?

A) INPUT:

a <- c(1, 2, 3)

b <- c(4, 5, 6)

c <- a + b

c

d <- a - b

d

e <- a \* b

e

f <- a / b

f

OUTPUT:

[1] 5 7 9

[1] -3 -3 -3

[1] 4 10 18

[1] 0.25 0.4 0.5

8) Write an R Program to Take Input From User.

Input name as “Jack” and age as 17.

The program should display the output as

“Hai , Jack next year you will be 18 years old”

1. INPUT:

# Taking input from user

name <- readline(prompt="Enter your name: ")

age <- as.integer(readline(prompt="Enter your age: "))

# Displaying the output

next\_age <- 17 + 1

cat("Hai, ", name, " next year you will be ", next\_age, " years old.", sep="")

OUTPUT:

Next year you will be 18 years old

EXERCISE-2

1. Perform Matrix Addition &amp; Subtraction in R?

INPUT:

# Define two matrices

A <- matrix(c(1,2,3,4), nrow = 2, ncol = 2)

B <- matrix(c(5,6,7,8), nrow = 2, ncol = 2)

# Matrix addition

C <- A + B

print(C)

# Matrix subtraction

D <- A - B

print(D)

OUTPUT:

> print(C)

[,1] [,2]

[1,] 6 10

[2,] 8 12

> # Matrix subtraction

> D <- A - B

> print(D)

[,1] [,2]

[1,] -4 -4

[2,] -4 -4

1. Perform Scalar multiplication and matrix multiplication in R?

INPUT:

scalar\_value <- 5

vector\_value <- c(1, 2, 3)

scalar\_multiplication\_result <- scalar\_value \* vector\_value

scalar\_multiplication\_result

[1] 5 10 15

matrix\_a <- matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2)

matrix\_b <- matrix(c(5, 6, 7, 8), nrow = 2, ncol = 2)

matrix\_multiplication\_result <- matrix\_a %\*% matrix\_b

matrix\_multiplication\_result

[,1] [,2]

[1,] 19 22

[2,] 43 50

1. Find Transpose of matrix in R?

INPUT:

matrix\_value <- matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2)

matrix\_transpose <- t(matrix\_value)

matrix\_transpose

[,1] [,2]

[1,] 1 3

[2,] 2 4

4) Perform the operation of combining matrices in R using cbind() and rbind()

Functions?

INPUT:

mat1 <- matrix(1:6, ncol = 2)

mat2 <- matrix(7:12, ncol = 2)

result\_matrix <- cbind(mat1, mat2)

print(result\_matrix)

mat1 <- matrix(1:6, ncol = 2)

mat2 <- matrix(7:12, ncol = 2)

result\_matrix <- rbind(mat1, mat2)

print(result\_matrix)

1. Deconstruct a matrix in R?

INPUT:

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

element <- mat[2,3]

print(element)

# Extracting a row

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

row <- mat[2, ]

print(row)

# Extracting a column

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

col <- mat[ , 2]

print(col)

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

submat <- mat[1:2, 1:2]

print(submat)

6)Perform array manipulation in R?

INPUT:

# Creating arrays

a = c(1, 2, 3)

b = matrix(1:6, nrow = 2, ncol = 3)

# Accessing elements

print(a[1]) # 1

print(b[1,2]) # 2

# Slicing

print(a[1:3]) # 1 2 3

print(b[,2]) # 2 5

# Modification

a[a > 2] = 0

b[b < 4] = 0

print(a) # 1 2 0

print(b) # 0 0 4 0 0 6

# Concatenation

c = c(a, b)

d = rbind(a, b)

e = cbind(a, b)

print(c) # 1 2 0 0 0 4 0 0 6

print(d) # 1 2 0

# 0 0 4

# 0 0 6

print(e) # 1 2 0 0

# 2 0 4 0

# 3 0 6 0

Output:

print(b) # 0 0 4 0 0 6

[,1] [,2] [,3]

[1,] 0 0 5

[2,] 0 4 6

> print(d) # 1 2 0

[,1] [,2] [,3]

a 1 2 0

0 0 5

1. 4 6

7)Perform calculations across array elements in an array using the apply() function?

INPUT:

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

square <- function(x) {

return (x^2)

}

mat\_squared <- apply(mat, MARGIN = 1, FUN = square)

print(mat\_squared)

OUTPUT:

# [,1] [,2] [,3]

# [1,] 1 4 9

# [2,] 36 25 16

9) Create a data frame and print the structure of the data frame in R?

INPUT:

# Create a data frame

df <- data.frame(

name = c("John", "Jane", "Jim", "Joan"),

age = c(35, 28, 42, 31),

gender = c("Male", "Female", "Male", "Female")

)

OUTPUT:

# Print the structure of the data frame

str(df)

# 'data.frame': 4 obs. of 3 variables:

# $ name : Factor w/ 4 levels "Jane","Jim","Joan",..: 2 1 3 4

# $ age : num 35 28 42 31

# $ gender: Factor w/ 2 levels "Female","Male": 2 1 2 1

10) Demonstrate the creation of S3 class in R?

INPUT:

Circle <- function(radius) {

structure(radius, class = "Circle")

}

circle1 <- Circle(5)

class(circle1)

circumference <- function(circle) {

2 \* pi \* circle$radius

}

circumference(circle1)

OUTPUT:

# 31.42

11) Demonstrate the creation of S4 class in R?

INPUT:

# Define the class and its representation

setClass("Person",

representation(

name = "character",

age = "numeric"

)

)

# Create a constructor function for the class

setMethod("initialize",

signature("Person"),

function(.Object, name, age) {

.Object at name <- name

.Object at age <- age

.Object

}

)

# Create an object of the class

person <- new("Person", name = "John Doe", age = 30)

# Access the object's properties

person at name

person at age

12) Demonstrate the creation of Reference class in R by defining a class called students

with fields – Name, Age , GPA. Also illustrate how the fields of the object can be

accessed using the $ operator. Modify the Name field by reassigning the name to Paul?

INPUT:

students <- setRefClass("students",

fields = list(

Name = "character",

Age = "numeric",

GPA = "numeric"

)

)

student <- students$new(Name = "John Doe", Age = 25, GPA = 3.5)

student$Name

student$Age

student$GPA

student$Name <- "Paul"

student$Name

EXERCISE-3

1. 1.Write a program to check whether an integer (entered by the user) is a prime number or not using control statements?

INPUT:

is\_prime <- function(n) {

if (n <= 1) {

return(FALSE)

}

for (i in 2:(n-1)) {

if (n %% i == 0) {

return(FALSE)

}

}

return(TRUE)

}

number <- as.integer(readline(prompt="Enter a number: "))

if (is\_prime(number)) {

cat("The number is prime.")

} else {

cat("The number is not prime.")

}

2) 2.Write a program to check whether a number entered by the user is positive number or a negative number or zero?

INPUT:

number <- as.integer(readline(prompt="Enter a number: "))

if (number > 0) {

cat("The number is positive.")

} else if (number < 0) {

cat("The number is negative.")

} else {

cat("The number is zero.")

}

3.Write a program to check whether a number is an Armstrong number or not using a while loop?

INPUT:

is\_armstrong <- function(n) {

num\_digits <- nchar(as.character(n))

temp <- n

sum <- 0

while (temp > 0) {

digit <- temp %% 10

sum <- sum + digit^num\_digits

temp <- floor(temp/10)

}

return (sum == n)

}

number <- as.integer(readline(prompt="Enter a number: "))

if (is\_armstrong(number)) {

cat("The number is an Armstrong number.")

} else {

cat("The number is not an Armstrong number.")

}

4)Write a program to demonstrate Repeat Loop in R?

INPUT:

counter <- 0

repeat {

counter <- counter + 1

print(counter)

if (counter >= 5) {

break

}

}

OUTPUT:

[1] 1

[1] 2

[1] 3

[1] 4

[1] 5

5)Using functions develop a simple calculator in R?

calculate <- function(num1, num2, operator) {

if (operator == "+") {

return(num1 + num2)

} else if (operator == "-") {

return(num1 - num2)

} else if (operator == "\*") {

return(num1 \* num2)

} else if (operator == "/") {

return(num1 / num2)

} else {

return("Invalid operator")

}

}

result <- calculate(5, 3, "+")

print(result)

result <- calculate(5, 3, "-")

print(result)

result <- calculate(5, 3, "\*")

print(result)

result <- calculate(5, 3, "/")

print(result)

result <- calculate(5, 3, "^")

print(result)

**OUTPUT:**

**[1] 8**

**[1] 2**

**[1] 15**

**[1] 1.666667**

**[1] "Invalid operator"**

**6)** **Demonstrate the creation of a complex number in R?**

z1 <- complex(real = 1, imaginary = 2)

print(z1)

z2 <- 3 + 4i

print(z2)

**OUTPUT:**

**[1] 1+2i**

**[1] 3+4i**

**7)Write a program to multiply two numbers using a function with a default value Assume default value as NULL?**

multiply <- function(x, y = NULL) {

if (is.null(y)) {

y <- 1

}

return (x \* y)

}

result <- multiply(5)

print(result)

result <- multiply(5, 3)

print(result)

**OUTPUT:**

**[1] 5**

**[1] 15**

**8)** **Find sum, mean and product of vector elements using built-in functions?**

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

sum\_of\_elements <- sum(vec)

print(sum\_of\_elements)

mean\_of\_elements <- mean(vec)

print(mean\_of\_elements)

product\_of\_elements <- prod(vec)

print(product\_of\_elements)

**OUTPUT:**

**[1] 15**

**[1] 3**

**[1] 120**

**9)** **Sort a vector in R using sort() function. Also find the index of the sorted vector?**

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

sorted\_vec <- sort(vec)

print(sorted\_vec)

index\_sorted\_vec <- order(vec)

print(index\_sorted\_vec)

**OUTPUT:**

**[1] 1 2 3 4 5**

**[1] 5 4 3 2 1**

**10)** **Find the L.C.M of two numbers entered by the user by creating a user-defined**

**Function?**

find\_lcm <- function(x, y) {

return (x \* y / gcd(x, y))

}

x <- as.integer(readline(prompt = "Enter the first number: "))

y <- as.integer(readline(prompt = "Enter the second number: "))

lcm <- find\_lcm(x, y)

print(paste("The LCM of", x, "and", y, "is", lcm))

**OUTPUT:**

**[1]Enter the first number: 12**

**[1]Enter the second number:4**

**[1]The LCM of 12 and 4 is 12**

EXERCISE-4

1. Demonstrate Vector Recycling in R?

vec1 <- c(1, 2, 3)

vec2 <- c(4, 5)

sum\_of\_vectors <- vec1 + vec2

print(sum\_of\_vectors)

OUTPUT:

[1] 5 7 7

1. Demonstrate the usage of apply function in R?

INPUT:

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

row\_sums <- apply(mat, 1, sum)

print(row\_sums)

OUTPUT:

[1] 3 7 11

1. Demonstrate the usage of lapply function in R?

INPUT:

list\_example <- list(c(1, 2, 3), c(4, 5, 6), c(7, 8, 9))

sum\_of\_squares <- function(x) {

sum(x^2)

}

result <- lapply(list\_example, sum\_of\_squares)

result

OUTPUT:

[[1]]

[1] 14

[[2]]

[1] 77

[[3]]

[1] 194

1. Demonstrate the usage of sapply function in R?

INPUT:

list\_example <- list(c(1, 2, 3), c(4, 5, 6), c(7, 8, 9))

sum\_of\_squares <- function(x) {

sum(x^2)

}

result <- sapply(list\_example, sum\_of\_squares)

result

OUTPUT:

[1] 14 77 194

1. Demonstrate the usage of tapply function in R?

INPUT:

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

grouping <- c("A", "B", "A", "B", "A", "B", "A", "B", "A")

result <- tapply(values, grouping, mean)

result

OUTPUT:

A B

5.0 6.0

1. Demonstrate the usage of mapply function in R?

INPUT:

a <- c(1, 2, 3)

b <- c(4, 5, 6)

multiply\_values <- function(x, y) {

x \* y

}

result <- mapply(multiply\_values, a, b)

result

OUTPUT:

[1] 4 10 18

7) Sum of Natural Numbers using Recursion?

INPUT:

sum\_of\_numbers <- function(n) {

if (n == 1) {

return(1)

} else {

return(n + sum\_of\_numbers(n - 1))

}

}

result <- sum\_of\_numbers(10)

result

OUTPUT:

[1] 55

8)Write a program to generate Fibonacci sequence using Recursion in R?

INPUT:

fibonacci <- function(n) {

if (n == 1 || n == 2) {

return(1)

} else {

return(fibonacci(n - 1) + fibonacci(n - 2))

}

}

result <- sapply(1:10, fibonacci)

result

OUTPUT:

[1] 1 1 2 3 5 8 13 21 34 55

9) Write a program to find factorial of a number in R using recursion?

INPUT:

factorial <- function(n) {

if (n == 0) {

return(1)

} else {

return(n \* factorial(n-1))

}

}

factorial(5)

OUTPUT:

[1] 120

# exponentiation

b <- 2 ^ 3

print(b)