ITA0448 STATISTICS WITH R PROGRAMMING  **somesh**

**Assignment-II**

SET-1:

3. Write R function to find nth highest value of a vector in the R program

**PROGRAM:**

find\_nth\_highest <- function(x, n) { if (length(x) == 0) { stop("Input vector is empty.") } else if (n > length(x)) { stop("n is larger than the length of the input vector.")

} else {

sorted\_x <- sort(unique(x), decreasing = TRUE) nth\_highest <- sorted\_x[n] return(nth\_highest)

}

}

**OUTPUT:**

> find\_nth\_highest <- function(x, n) {

+ if (length(x) == 0) {

+ stop("Input vector is empty.")

+ } else if (n > length(x)) {

+ stop("n is larger than the length of the input vector.")

+ } else {

+ sorted\_x <- sort(unique(x), decreasing = TRUE)

+ nth\_highest <- sorted\_x[n]

+ return(nth\_highest)

+ }

+ }

>

>

5. Write R Program to find maximum and minimum value of a given vector using control statement.

**PROGRAM:**

find\_max\_min <- function(x) { if (length(x) == 0) { stop("Input vector is empty.")

} else { max\_val <- x[1] min\_val <- x[1] for (i in 2:length(x)) { if (x[i] > max\_val) { max\_val <- x[i]

}

if (x[i] < min\_val) { min\_val <- x[i]

}

}

return(list("max" = max\_val, "min" = min\_val))

}

}

**OUTPUT:**

> find\_max\_min <- function(x) {

+ if (length(x) == 0) {

+ stop("Input vector is empty.")

+ } else {

+ max\_val <- x[1]

+ min\_val <- x[1]

+ for (i in 2:length(x)) {

+ if (x[i] > max\_val) {

+ max\_val <- x[i] + }

+ if (x[i] < min\_val) {

+ min\_val <- x[i]

+ }

+ }

+ return(list("max" = max\_val, "min" = min\_val)) + }

+ }

>

>

5. Write R Program to find maximum and minimum value of a given vector using control statement.

**PROGRAM:**

# Define a vector of numbers

my\_vector <- c(3, 5, 2, 8, 4, 9, 1)

# Set the initial values of the maximum and minimum to be the first element of the vector max\_value <- my\_vector[1] min\_value <- my\_vector[1]

# Loop through the vector using a for loop for (i in 2:length(my\_vector)) {

# If the current value is greater than the current maximum, update the maximum if (my\_vector[i] > max\_value) { max\_value <- my\_vector[i]

}

# If the current value is less than the current minimum, update the minimum if (my\_vector[i] < min\_value) { min\_value <- my\_vector[i]

}

}

# Print the maximum and minimum values cat("Maximum value:", max\_value, "\n") cat("Minimum value:", min\_value)

**OUTPUT:**

> # Define a vector of numbers

> my\_vector <- c(3, 5, 2, 8, 4, 9, 1)

>

> # Set the initial values of the maximum and minimum to be the first eleme nt of the vector

> max\_value <- my\_vector[1]

> min\_value <- my\_vector[1]

>

> # Loop through the vector using a for loop

> for (i in 2:length(my\_vector)) {

+

+ # If the current value is greater than the current maximum, update th e maximum

+ if (my\_vector[i] > max\_value) {

+ max\_value <- my\_vector[i]

+ }

+

+ # If the current value is less than the current minimum, update the m inimum

+ if (my\_vector[i] < min\_value) {

+ min\_value <- my\_vector[i]

+ }

+ }

>

> # Print the maximum and minimum values

> cat("Maximum value:", max\_value, "\n")

Maximum value: 9

> cat("Minimum value:", min\_value)

Minimum value: 1>

>

**SET 2 :**

1. Create the following matrices (i) Square Matrix (ii) Identity Matrix (iii)diagonal matrix

**PROGRAM:**

(i) Square Matrix:

# Create a square matrix of size 3x3

square\_matrix <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3)

# Print the matrix square\_matrix

**OUTPUT:**

> # Create a square matrix of size 3x3

> square\_matrix <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3) >

> # Print the matrix

> square\_matrix

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

[1,] 1 4 7

[2,] 2 5 8

[3,] 3 6 9

>

>

>

(ii) Identity Matrix:

# Create an identity matrix of size 3x3 identity\_matrix <- diag(3)

# Print the matrix identity\_matrix

**OUTPUT:**

> # Create an identity matrix of size 3x3

> identity\_matrix <- diag(3)

>

> # Print the matrix

> identity\_matrix

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

[1,] 1 0 0

[2,] 0 1 0

[3,] 0 0 1

>

>

(iii) Diagonal Matrix:

# Create a diagonal matrix of size 3x3 diagonal\_matrix <- diag(c(1, 2, 3))

# Print the matrix diagonal\_matrix

**OUTPUT:**

> # Create a diagonal matrix of size 3x3

> diagonal\_matrix <- diag(c(1, 2, 3))

>

> # Print the matrix

> diagonal\_matrix

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

[1,] 1 0 0

[2,] 0 2 0

[3,] 0 0 3

>

>

2. Using sapply, check that all elements of the list are vectors of the same length. Also calculate the sum of each element.

**PROGRAM:**

# Example list my\_list <- list(c(1, 2, 3), c(4, 5, 6), c(7, 8, 9))

# Check if all elements of the list are vectors of the same length if (length(unique(sapply(my\_list, length))) == 1) { print("All elements of the list are vectors of the same length")

} else { print("Elements of the list are not vectors of the same length")

}

# Calculate the sum of each element using sapply sums <- sapply(my\_list, sum)

# Print the sums

Sums

**OUTPUT:**

> # Example list

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

>

> # Check if all elements of the list are vectors of the same length

> if (length(unique(sapply(my\_list, length))) == 1) {

+ print("All elements of the list are vectors of the same length")

+ } else {

+ print("Elements of the list are not vectors of the same length")

+ }

[1] "All elements of the list are vectors of the same length"

>

> # Calculate the sum of each element using sapply

> sums <- sapply(my\_list, sum)

>

> # Print the sums

> sums

[1] 6 15 24

>

1. We found out that the blood pressure instrument is under-recording each measure and all measurement incorrect by 0.1. How would you add 0.1 to all values in the blood vector?

**PROGRAM:**

# Example vector blood\_pressure <- c(120, 130, 140, 150, 160)

# Add 0.1 to all values in the vector blood\_pressure <- blood\_pressure + 0.1

# Print the updated vector blood\_pressure

1. We found out that the first patient is 33 years old. How would you change the first element of the vector age to 33 years?

**PROGRAM:**

# Example vector

age <- c(25, 30, 35, 40, 45)

# Change the first element of the vector to 33 years age[1] <- 33

# Print the updated vector

Age

**OUTPUT**:

> # Example vector

> age <- c(25, 30, 35, 40, 45)

>

> # Change the first element of the vector to 33 years

> age[1] <- 33

>

> # Print the updated vector

> age

[1] 33 30 35 40 45

>

>

5. Suppose A = [ 1 1 3 5 2 6 −2 −1 −3 ] (a) Check that A 3 = 0 where 0 is a 3 × 3 matrix with every entry equal to 0. (b) Replace the third column of A by the sum of the second and third columns

**PROGRAM:**

A)

# Define the matrix A

A <- c(1, 1, 3, 5, 2, 6, -2, -1, -3)

# Create a 3x3 submatrix from the first nine elements of A

A\_sub <- matrix(A[1:9], nrow = 3)

# Check if A\_sub is a zero matrix all(A\_sub == 0)

**OUTPUT:**

> # Define the matrix A

> A <- c(1, 1, 3, 5, 2, 6, -2, -1, -3)

>

> # Create a 3x3 submatrix from the first nine elements of A

> A\_sub <- matrix(A[1:9], nrow = 3)

>

> # Check if A\_sub is a zero matrix

> all(A\_sub == 0)

[1] FALSE

>

>

**B)**

# Define the matrix A

A <- c(1, 1, 3, 5, 2, 6, -2, -1, -3)

# Replace the third column of A by the sum of the second and third columns A[,3] <- A[,2] + A[,3]

# Print the updated matrix A

A

**OUTPUT:**

> # Define the matrix A

> A <- c(1, 1, 3, 5, 2, 6, -2, -1, -3)

>

> # Replace the third column of A by the sum of the second and third colum ns

> A[,3] <- A[,2] + A[,3]