Experiment No :6

Aim: Write a program to implement a function, performing matrix-vector multiplication. It should take a matrix A and a vector b as arguments, and return the vector AB. Use two loops to do this, rather than %*% or any vectorization.

Objective: 1. Students will be able to implement a function, performing matrix-vector multiplication.

2. Students will be able to specify matrix-vector multiplication in R.

Software Requirements: R studio 4.2.2

Theory:

R is a programming language that is widely used for statistical analysis, data visualization, and machine learning. One of the fundamental operations in R is subsetting, which allows users to extract specific elements from an object such as a vector or a matrix. Subsetting is an essential operation for data manipulation and analysis. In R, there are four methods for subsetting, including the [ ] operator, the [[ ]] operator, the $ operator, and the subset() function. These methods provide flexibility in accessing and extracting data based on specific conditions.

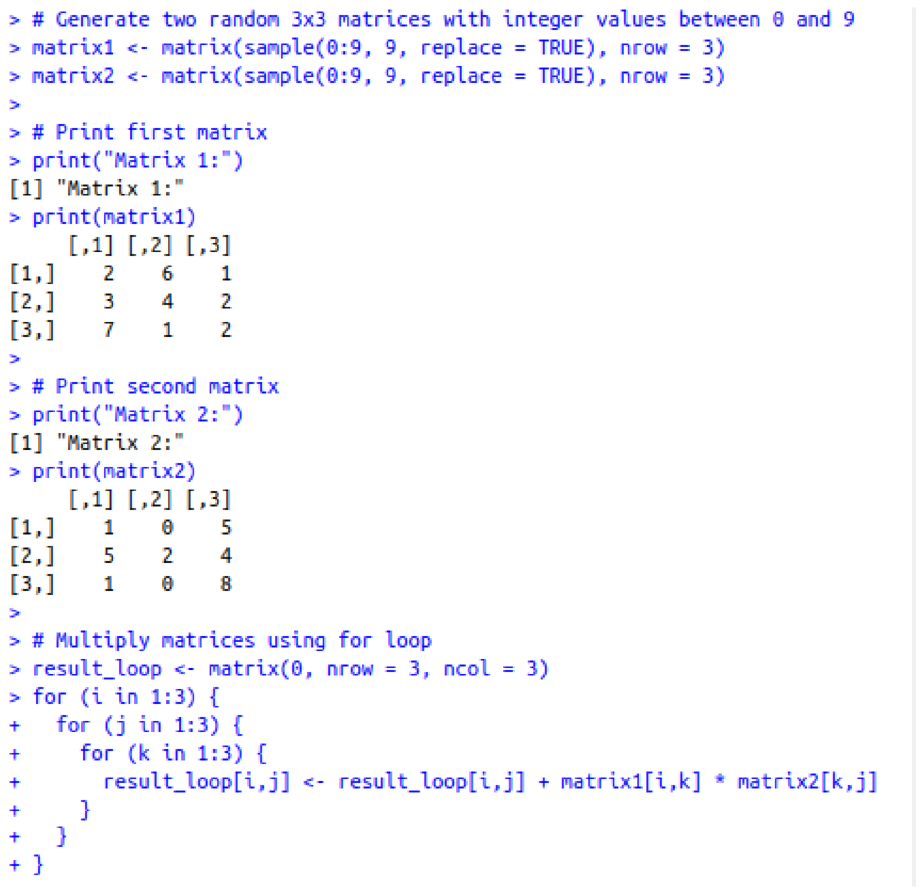
Another important concept in R programming is the handling of missing data, which is represented by NA values. Missing data can arise due to various reasons such as measurement error or data corruption. R provides several functions for identifying and handling missing data, such as is.na() and na.omit().

Control structures are also essential in R programming, allowing users to control the flow of the program based on specific conditions. These structures include if, while, for, and repeat loops, which can be used to make decisions based on variable assessment. For example, an if statement can be used to check whether a condition is true or false and execute a block of code accordingly.

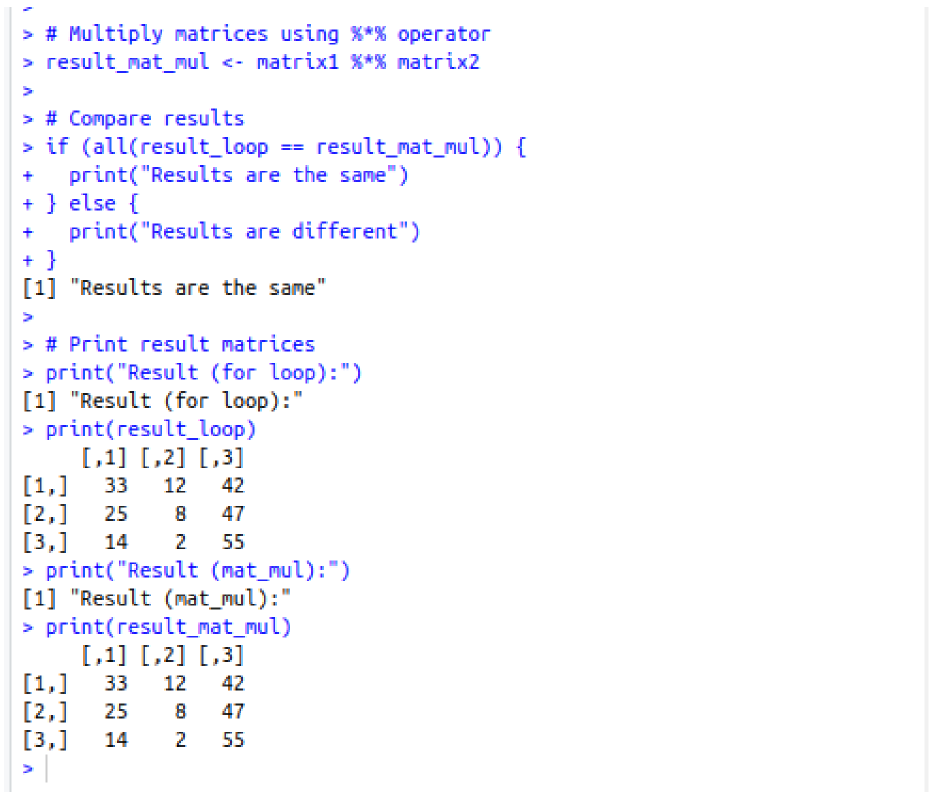
Matrix multiplication is a common operation in linear algebra, and it can be performed in R using a user-defined function called mat_mup(). This function takes two matrices as arguments and checks whether the number of columns in the first matrix matches the number of rows in the second matrix. If not, it returns "can't multiply". If the matrices can be multiplied, the function uses three nested for loops to perform the multiplication and returns the resulting matrix. Alternatively, the built-in %*% operator can also be used to multiply matrices in R. Matrix multiplication is useful in various applications such as image processing, machine learning, and statistical modeling.

Conclusion: We learned about subsetting, missing data handling, control structures, and matrix multiplication in R programming, which are essential concepts for data analysis and manipulation.

Code with Output:



```
> # Generate two random 3x3 matrices with integer values between 0 and 9
> matrix1 <- matrix(sample(0:9, 9, replace = TRUE), nrow = 3)
> matrix2 <- matrix(sample(0:9, 9, replace = TRUE), nrow = 3)
>
> # Print first matrix
> print("Matrix 1:")
[1] "Matrix 1:"
> print(matrix1)
     [,1] [,2] [,3]
[1,]    2    6    1
[2,]    3    4    2
[3,]    7    1    2
>
> # Print second matrix
> print("Matrix 2:")
[1] "Matrix 2:"
> print(matrix2)
     [,1] [,2] [,3]
[1,]    1    0    5
[2,]    5    2    4
[3,]    1    0    8
>
> # Multiply matrices using for loop
> result_loop <- matrix(0, nrow = 3, ncol = 3)
> for (i in 1:3) {
+   for (j in 1:3) {
+     for (k in 1:3) {
+       result_loop[i,j] <- result_loop[i,j] + matrix1[i,k] * matrix2[k,j]
+     }
+   }
+ }
```



```
>
> # Multiply matrices using %*% operator
> result_mat_mul <- matrix1 %*% matrix2
>
> # Compare results
> if (all(result_loop == result_mat_mul)) {
+   print("Results are the same")
+ } else {
+   print("Results are different")
+ }
[1] "Results are the same"
>
> # Print result matrices
> print("Result (for loop):")
[1] "Result (for loop):"
> print(result_loop)
     [,1] [,2] [,3]
[1,]   33   12   42
[2,]   25    8   47
[3,]   14    2   55
> print("Result (mat_mul):")
[1] "Result (mat_mul):"
> print(result_mat_mul)
     [,1] [,2] [,3]
[1,]   33   12   42
[2,]   25    8   47
[3,]   14    2   55
>
```