***SAVEETHA SCHOOL OF ENGINEERING***

***SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES***

***ITA 04521- STATISTICS WITH R PROGRAMMING FOR DEEP LEARNING***

DAY 2 – LAB EXERCISES

**Reg No : 192124158**

**Name : K.B.Pavan**

IMPLEMENTATION OF VECTOR RECYCLING, APPLY FAMILY &amp; RECURSION

1. Demonstrate Vector Recycling in R.

CODE :

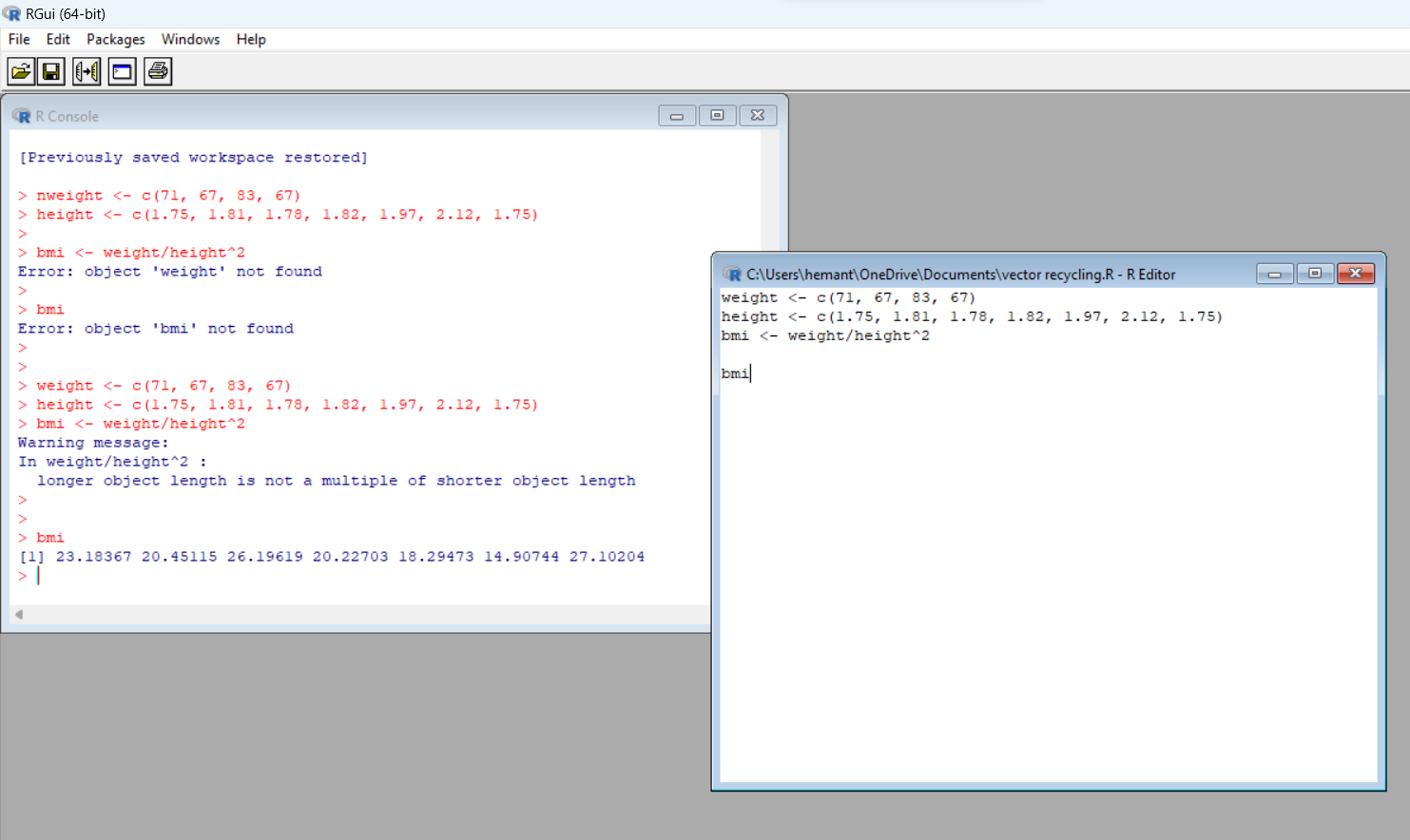
weight <- c(71, 67, 83, 67)

height <- c(1.75, 1.81, 1.78, 1.82, 1.97, 2.12, 1.75)

bmi <- weight/height\*2

bmi

OUTPUT:



1. Demonstrate the usage of apply function in R

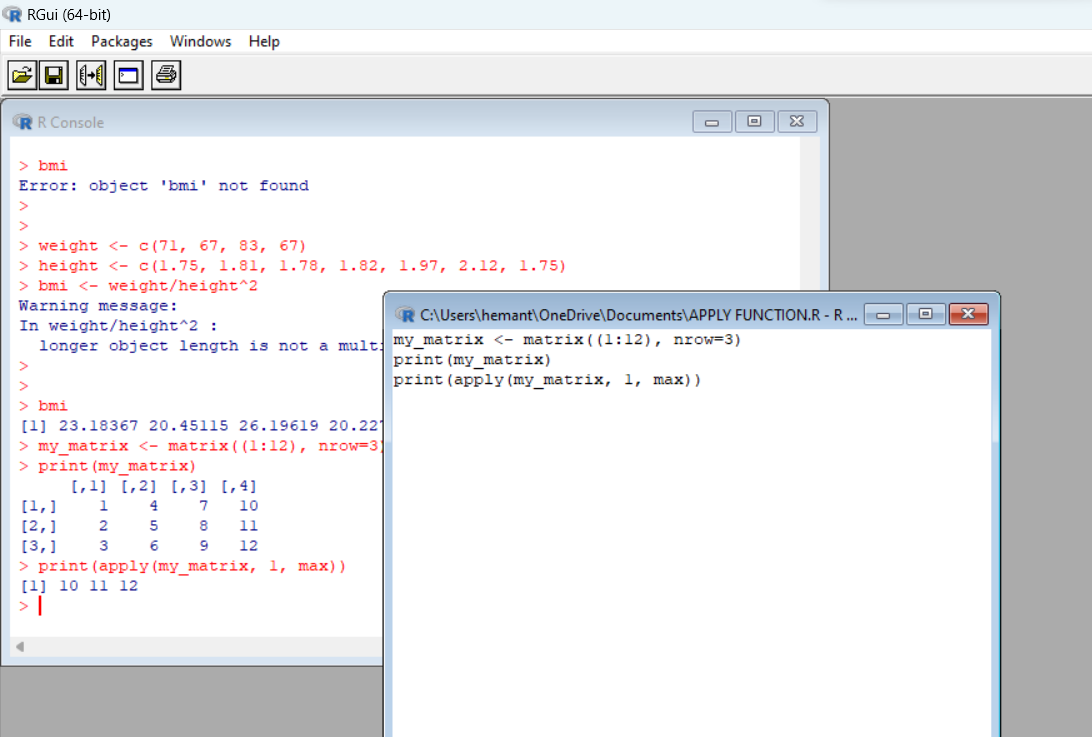
CODE:

my\_matrix <- matrix((1:12), nrow=3)

print(my\_matrix)

print(apply(my\_matrix, 1, max))

OUTPUT:



1. 4. Demonstrate the usage of sapply function in R

CODE:

# Define the list of numbers

my\_list <- list(2, 4, 6, 8)

# Define the function to square a number

square <- function(x) {

return(x^2)

}

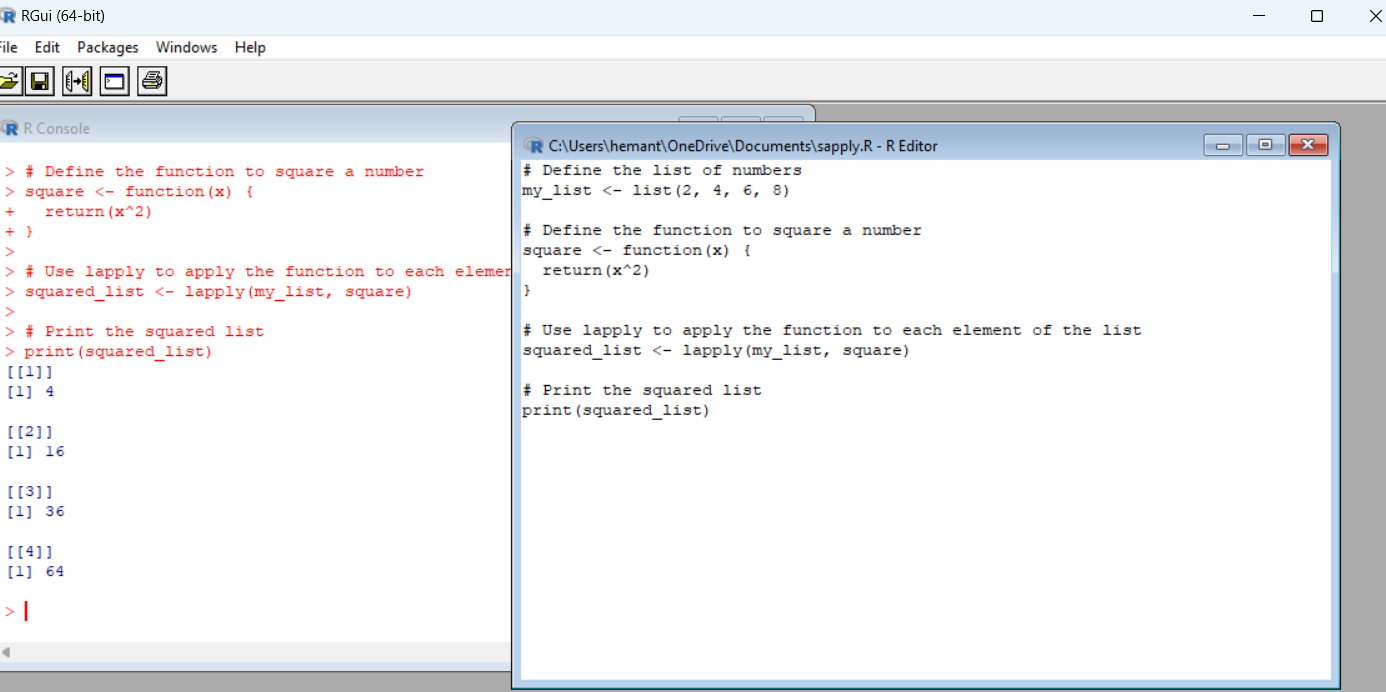
# Use lapply to apply the function to each element of the list

squared\_list <- lapply(my\_list, square)

# Print the squared list

print(squared\_list)

OUTPUT:



1. Demonstrate the usage of tapply function in R

CODE :

# Define the data frame

sales\_df <- data.frame(

sales = c(1000, 2000, 1500, 3000, 1200, 2500),

region = c("East", "West", "East", "West", "East", "West")

)

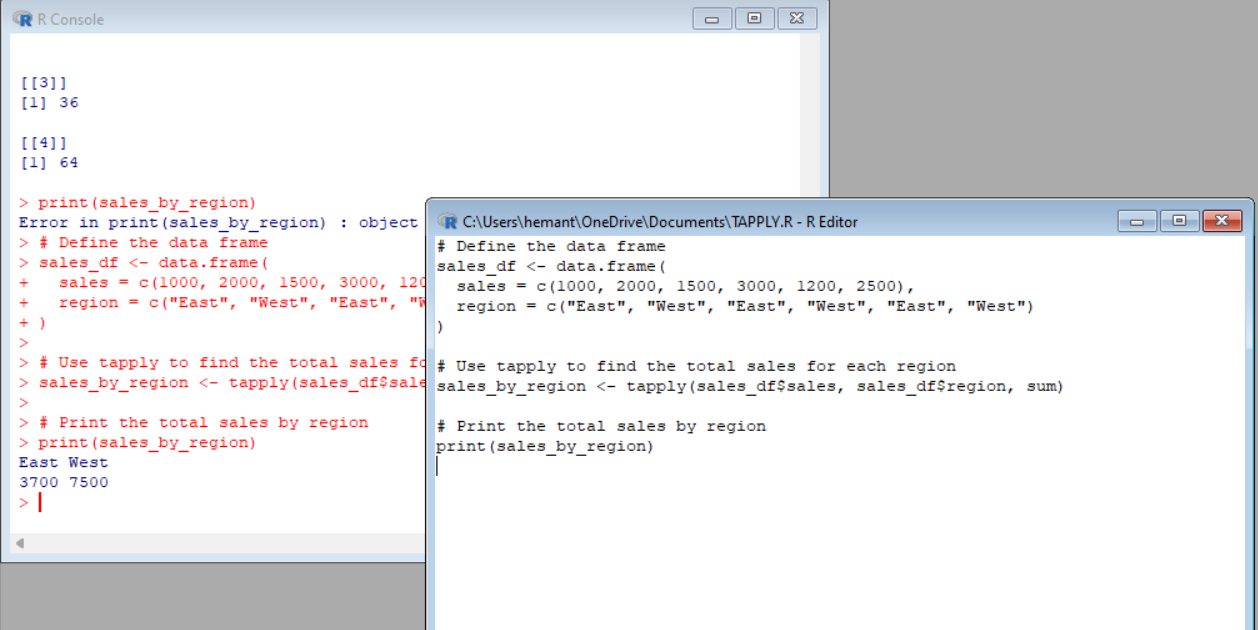
# Use tapply to find the total sales for each region

sales\_by\_region <- tapply(sales\_df$sales, sales\_df$region, sum)

# Print the total sales by region

print(sales\_by\_region)

OUTPUT:



1. Demonstrate the usage of lapply function in R

CODE:

# Define the list of numbers

my\_list <- list(2, 4, 6, 8)

# Define the function to square a number

square <- function(x) {

return(x^2)

}

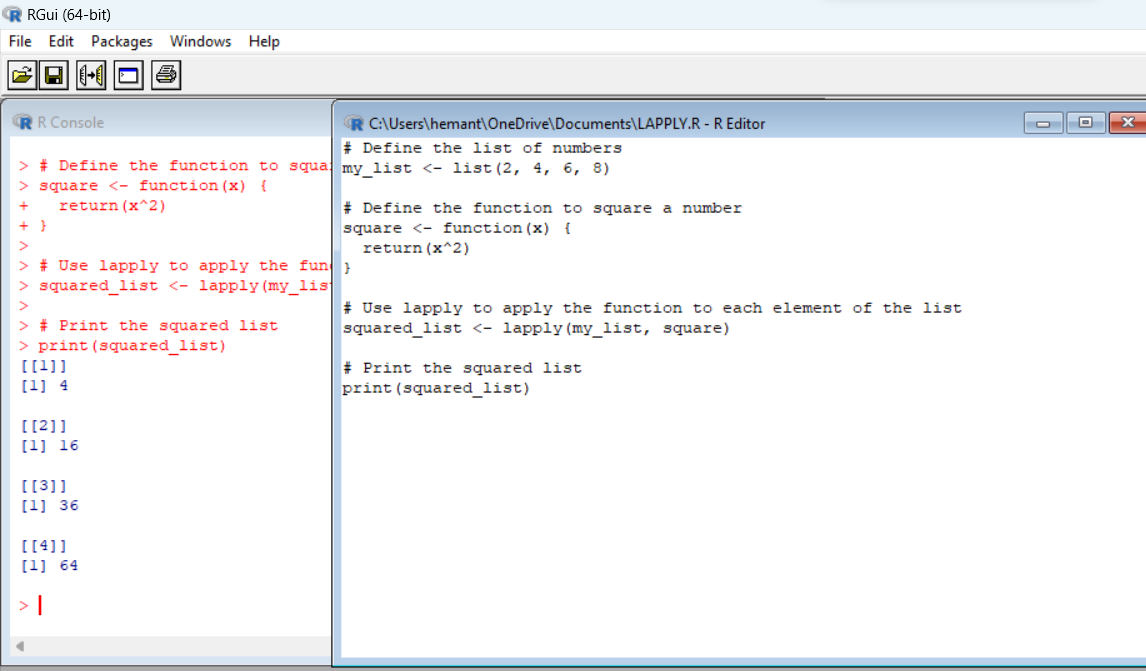
# Use lapply to apply the function to each element of the list

squared\_list <- lapply(my\_list, square)

# Print the squared list

print(squared\_list)

OUTPUT:



1. Demonstrate the usage of mapply function in R

CODE:

# Define the two vectors

a <- c(1, 2, 3)

b <- c(4, 5, 6)

# Define the function to find the product of two numbers

multiply <- function(x, y) {

return(x \* y)

}

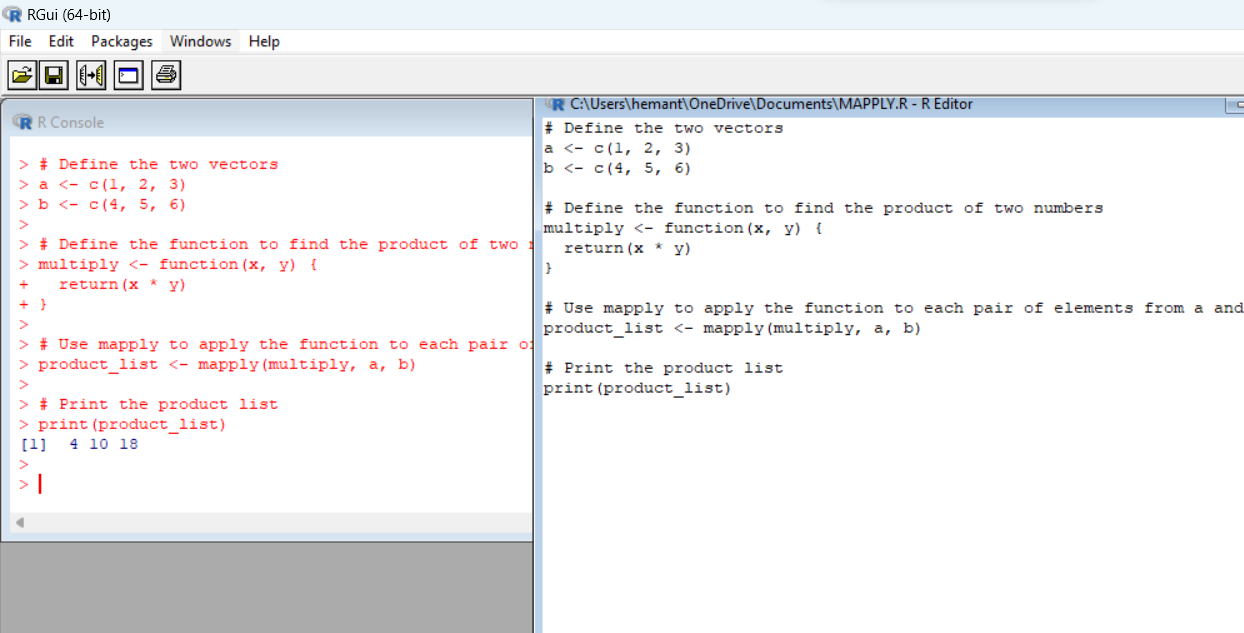
# Use mapply to apply the function to each pair of elements from a and b

product\_list <- mapply(multiply, a, b)

# Print the product list

print(product\_list)

OUTPUT:



1. Sum of Natural Numbers using Recursion

CODE:

# Define a function to find the sum of natural numbers using recursion

sum\_natural <- function(n) {

if (n <= 1) {

return(n)

} else {

return(n + sum\_natural(n - 1))

}

}

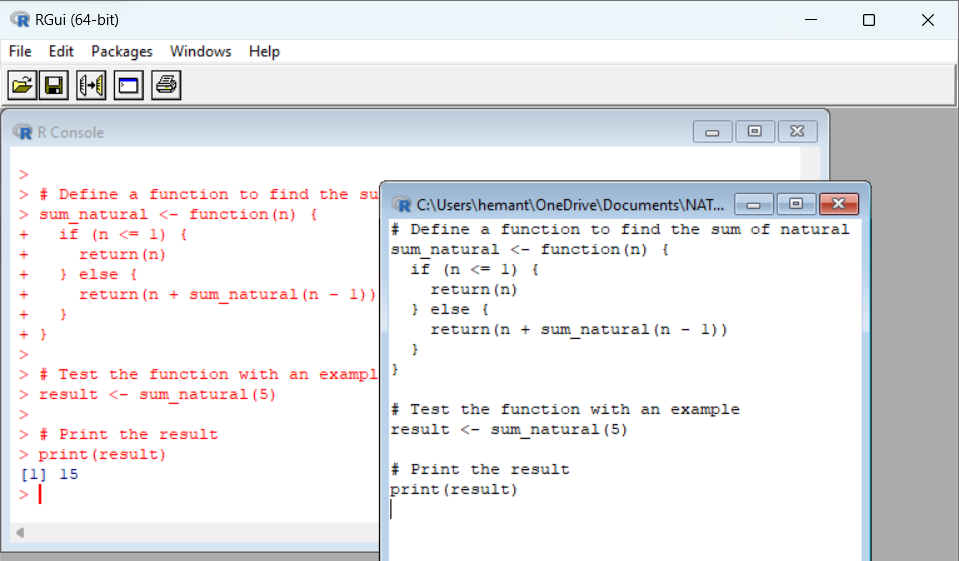
# Test the function with an example

result <- sum\_natural(5)

# Print the result

print(result)

OUTPUT:



1. Write a program to generate Fibonacci sequence using Recursion in R

CODE:

# Define a function to generate the Fibonacci sequence using recursion

fibonacci <- function(n) {

if (n <= 1) {

return(n)

} else {

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

}

}

# Test the function with an example

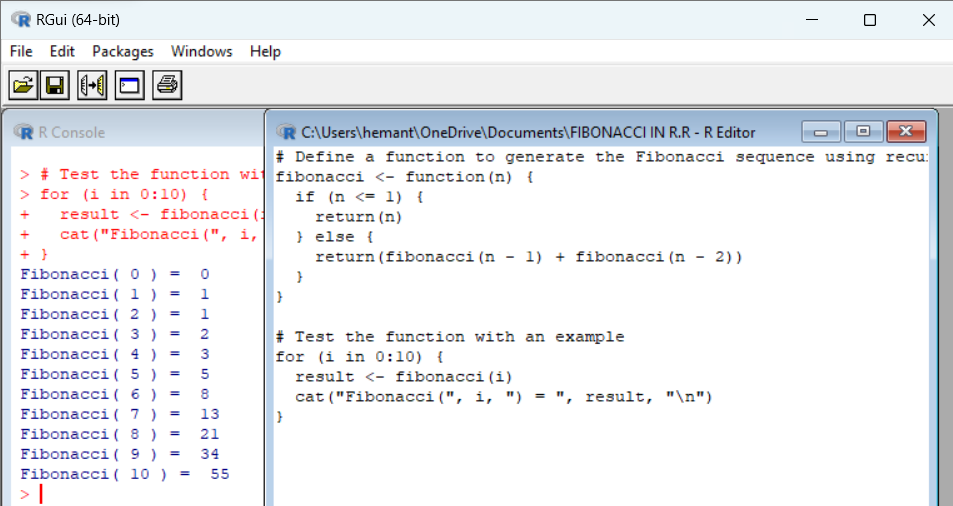
for (i in 0:10) {

result <- fibonacci(i)

cat("Fibonacci(", i, ") = ", result, "\n")

}

OUTPUT:



1. Write a program to find factorial of a number in R using recursion.

CODE:

# Define a function to find the factorial of a number using recursion

factorial <- function(n) {

if (n <= 1) {

return(1)

} else {

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

}

}

# Test the function with an example

result <- factorial(5)

# Print the result

print(result)

OUTPUT:

