**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**ITA 0451 - STATISTICS WITH R PROGRAMMING FOR REAL TIME PROBLEM**

**DAY 2 – LAB EXERCISES**

**NAME:** Bala Manohar. J

**REG NO:**192124133

1. **Demonstrate Vector Recycling in R.**

**Source Code:**

# create two vectors of di

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

b <- c(2, 4)

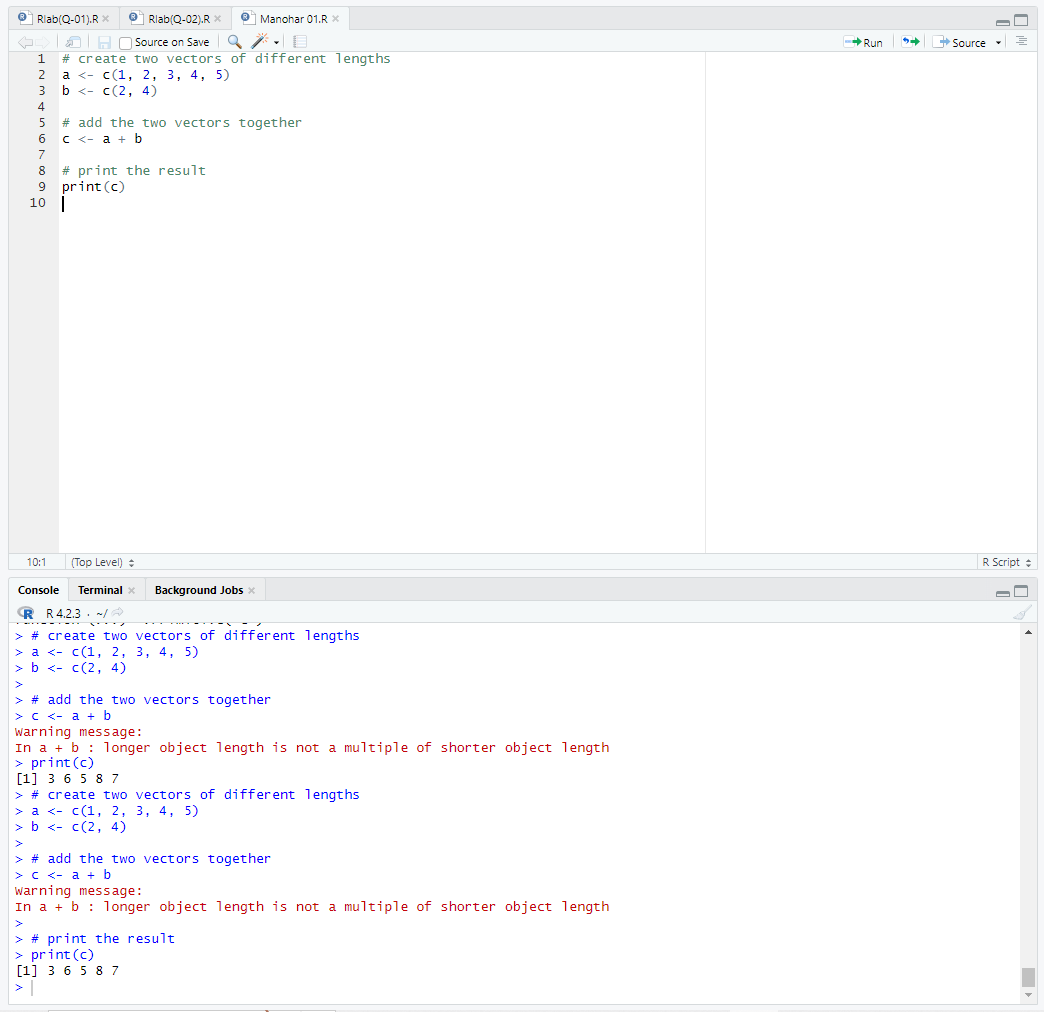
# add the two vectors together

c <- a + b

# print the result

print(c)

**OUTPUT:**

****

1. **Demonstrate the usage of apply function in R**

**Source Code:**

# create a matrix

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

# print the matrix

print(m)

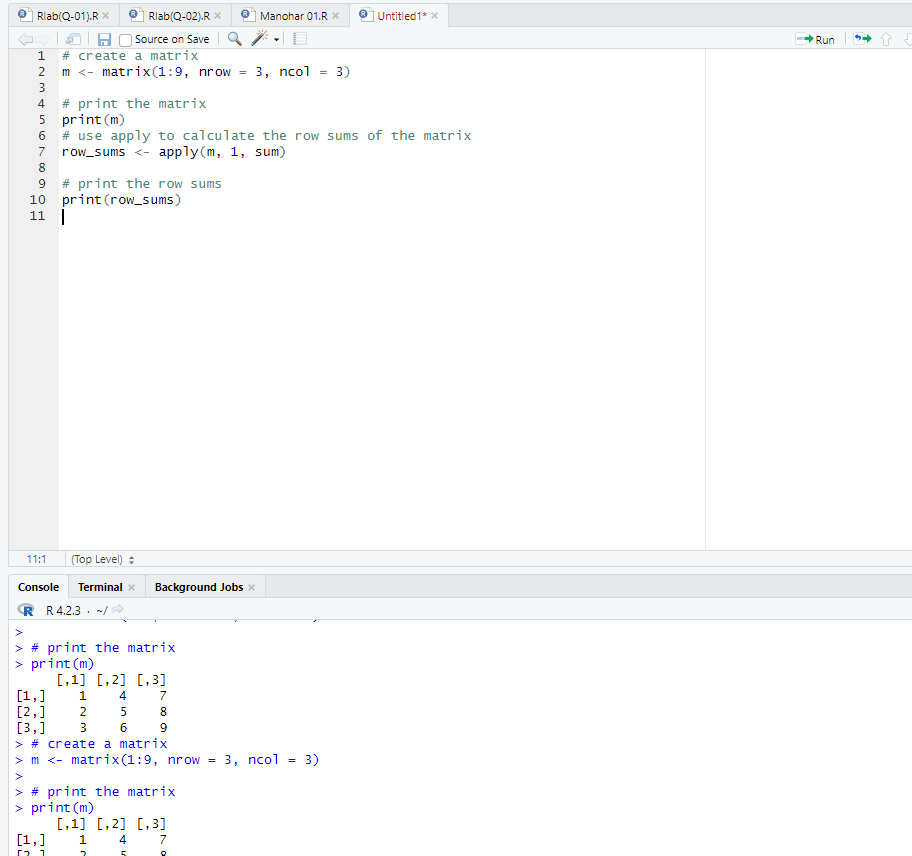
# use apply to calculate the row sums of the matrix

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

# print the row sums

print(row\_sums)

**OUTPUT:**



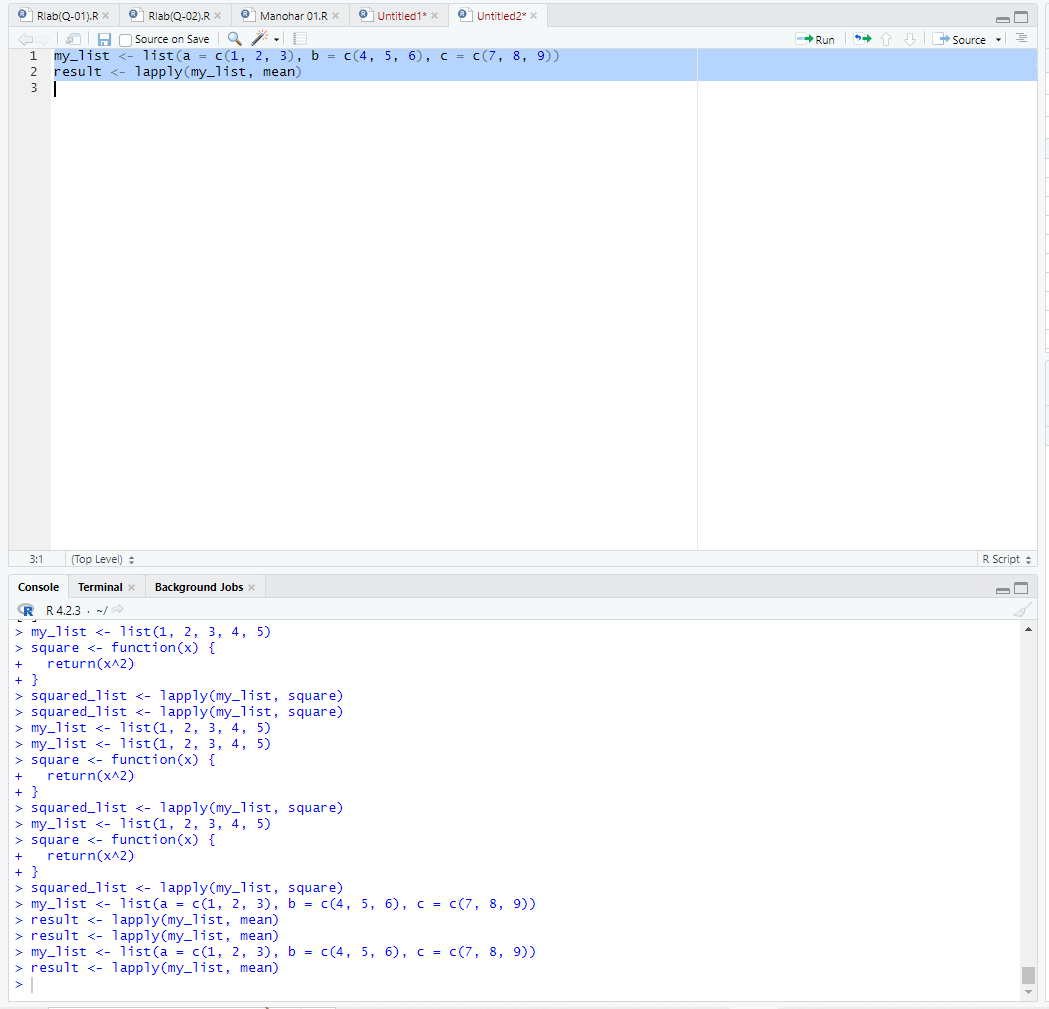
1. **Demonstrate the usage of lapply function in R**

**Source Code:**

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

result <- lapply(my\_list, mean)

**OUTPUT:**

****

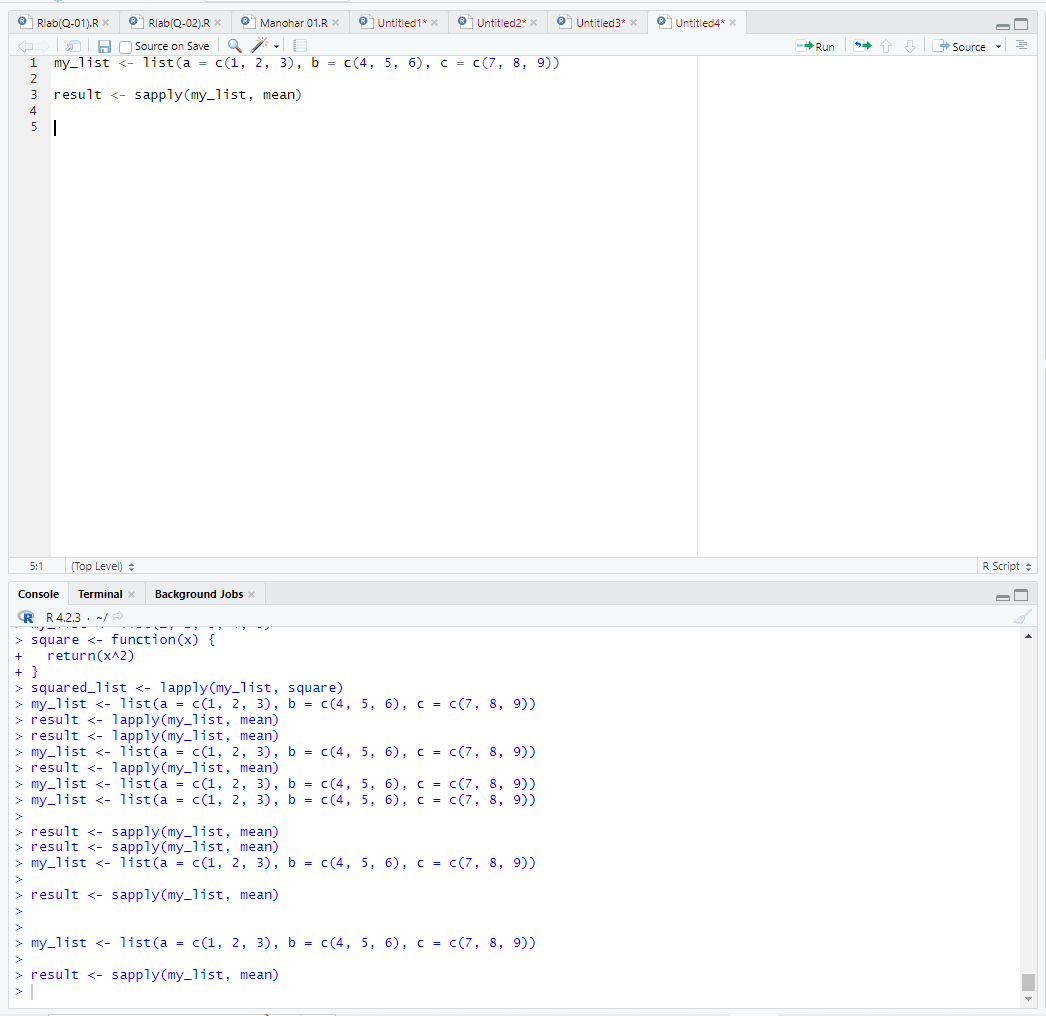
1. **Demonstrate the usage of sapply function in R**

**Source Code:**

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

result <- sapply(my\_list, mean)

**OUTPUT:**

****

1. **Demonstrate the usage of tapply function in R**

**Source Code:**

# Sample data

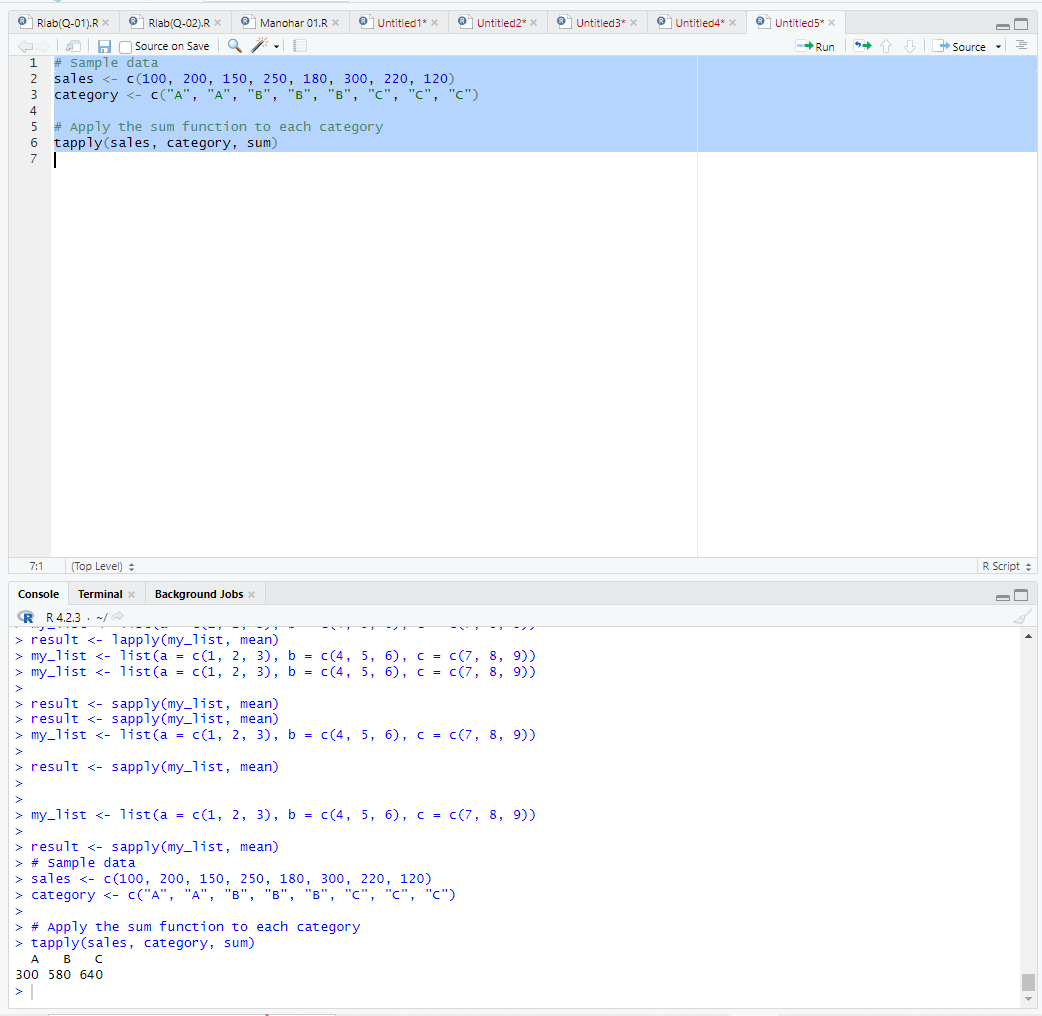
sales <- c(100, 200, 150, 250, 180, 300, 220, 120)

category <- c("A", "A", "B", "B", "B", "C", "C", "C")

# Apply the sum function to each category

tapply(sales, category, sum)

**OUTPUT:**

****

1. **Demonstrate the usage of mapply function in R**

**Source Code:**

# Sample data

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

y <- c(6, 7, 8, 9, 10)

# Define a custom function

myfun <- function(a, b) {

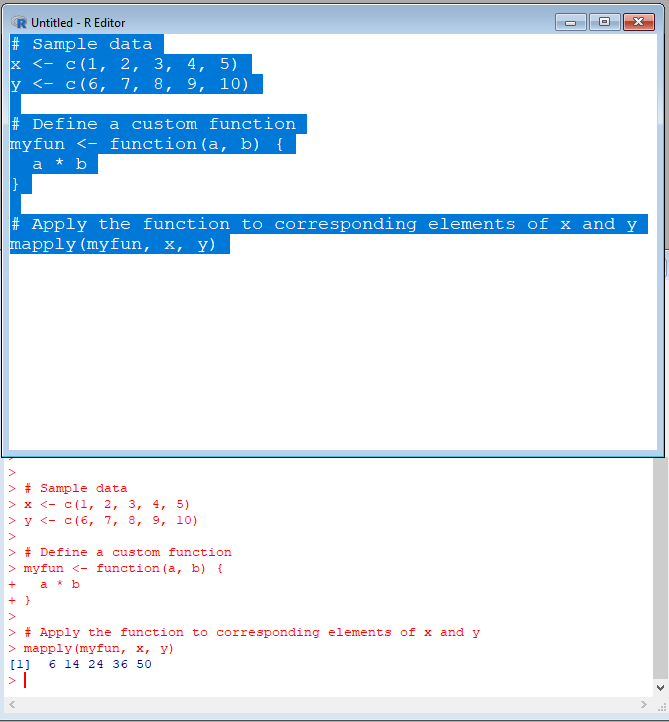
a \* b

}

# Apply the function to corresponding elements of x and y

mapply(myfun, x, y)

**OUTPUT:**

****

1. **Sum of Natural Numbers using Recursion**

**Source Code:**

# Define a recursive function to calculate sum of n natural numbers

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

if(n == 1) {

return(1)

} else {

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

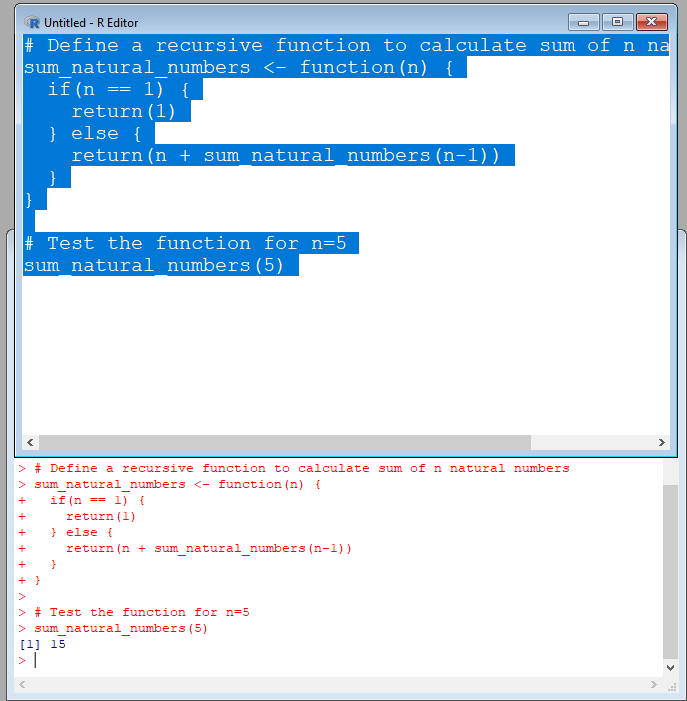
}

}

# Test the function for n=5

sum\_natural\_numbers(5)

**OUTPUT;**

****

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

**Source Code:**

# Define a recursive function to generate Fibonacci sequence

fibonacci <- function(n) {

if(n <= 1) {

return(n)

} else {

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

}

}

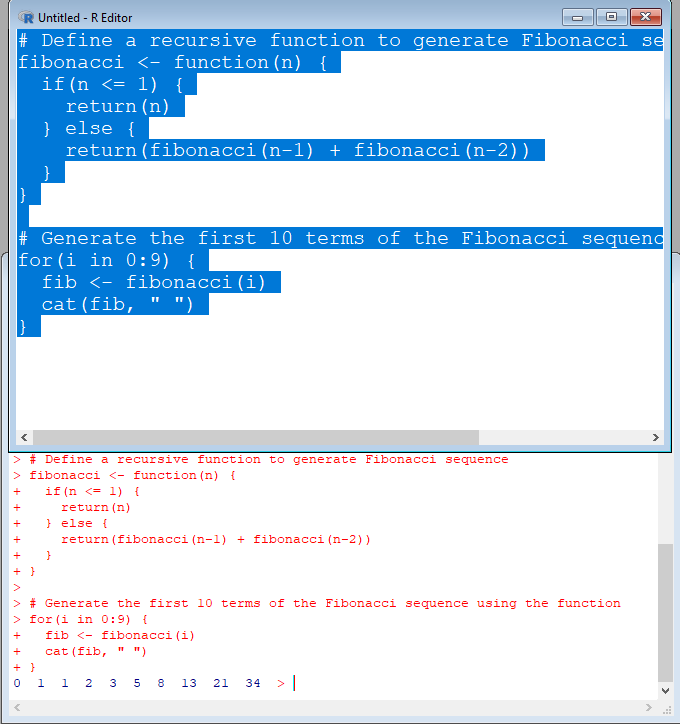
# Generate the first 10 terms of the Fibonacci sequence using the function

for(i in 0:9) {

fib <- fibonacci(i)

cat(fib, " ")

}



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

**Source Code:**

# Define a recursive function to calculate factorial of n

factorial <- function(n) {

if(n <= 1) {

return(1)

} else {

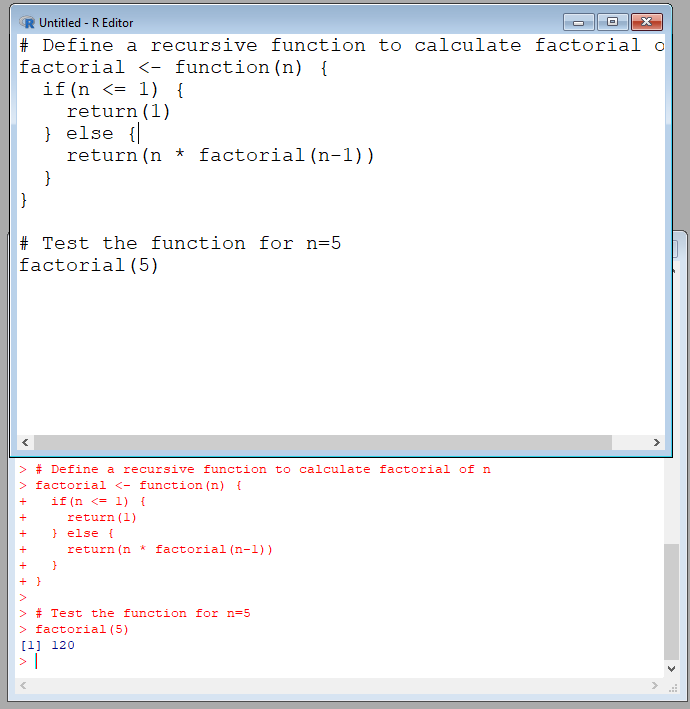
return(n \* factorial(n-1))

}

}

# Test the function for n=5

factorial(5)



**10.Consider two vectors: x=seq(1,43,along.with=Id)**

**y=seq(-20,0,along.with=Id)**

**Create a data frame ‘df’ as shown below.**

**&gt;df**

**Id Letter x y**

**1 1 a 1.000000 -20.000000**

**2 1 b 4.818182 -18.181818**

**3 1 c 8.636364 -16.363636**

**4 2 a 12.454545 -14.545455**

**5 2 b 16.272727 -12.727273**

**6 2 c 20.090909 -10.909091**

**7 3 a 23.909091 -9.090909**

**8 3 b 27.727273 -7.272727**

**9 3 c 31.545455 -5.454545**

**10 4 a 35.363636 -3.636364**

**11 4 b 39.181818 -1.818182**

**12 4 c 43.000000 0.000000**

**Source Code:**

# Create the x and y vectors

x <- seq(1, 43, along.with = Id)

y <- seq(-20, 0, along.with = Id)

# Create the Id and Letter vectors

Id <- rep(1:4, each = 3)

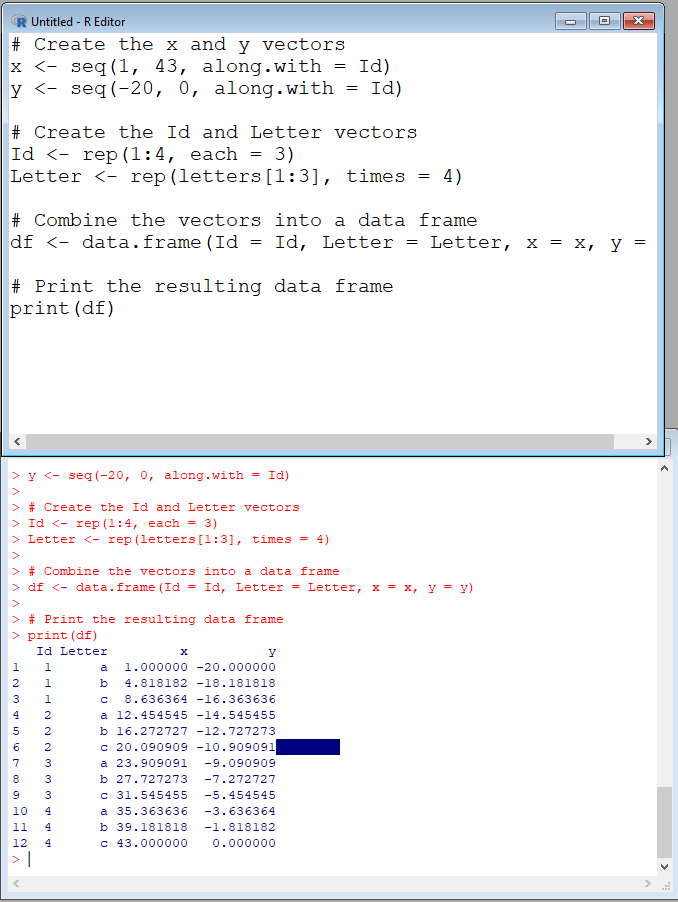
Letter <- rep(letters[1:3], times = 4)

# Combine the vectors into a data frame

df <- data.frame(Id = Id, Letter = Letter, x = x, y = y)

# Print the resulting data frame

print(df)



**12.** **Create two data frame df1 and df2:**

**&gt; df1**

**Id Age**

**1 1 14**

**2 2 12**

**3 3 15**

**4 4 10**

**&gt; df2**

**Id Sex Code**

**1 1 F a**

**2 2 M b**

**3 3 M c**

**4 4 F d**

**From df1 and df2 create M:**

**&gt;M**

**Id Age Sex Code**

**1 1 14 F a**

**2 2 12 M b**

**3 3 15 M c 4 4 10 F d**

**Source Code:**

# Create the 'df1' data frame

df1 <- data.frame(Id = 1:4, Age = c(14, 12, 15, 10))

# Create the 'df2' data frame

df2 <- data.frame(Id = 1:4, Sex = c("F", "M", "M", "F"), Code = c("a", "b", "c", "d")

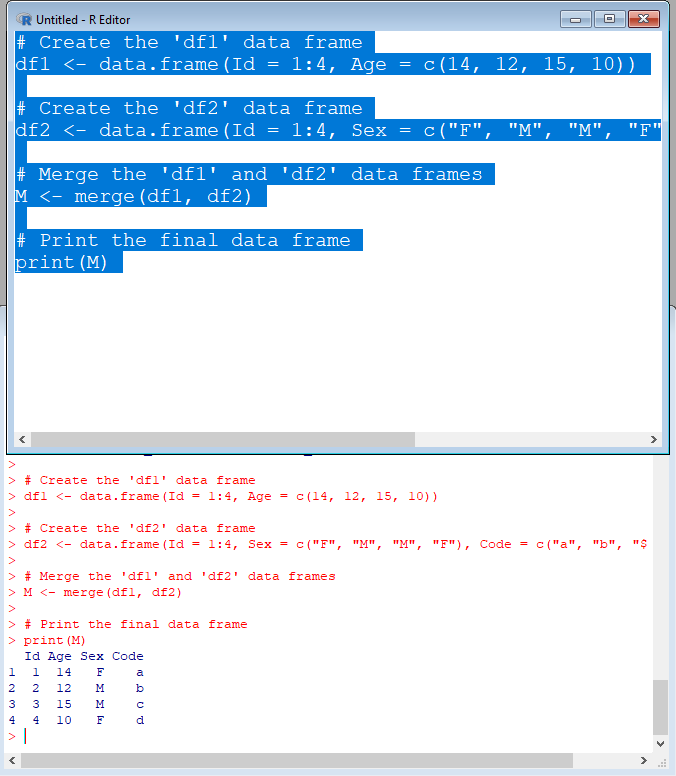
# Merge the 'df1' and 'df2' data frames

M <- merge(df1, df2)

# Print the final data frame

print(M)

**OUTPUT:**

****

**13.** **Create a data frame df3:**

**&gt; df3 id2**

**score 1 4**

**100**

**2 3 98**

**3 2 94**

**4 1 99**

**From M (used in Exercise-3) and df3 create N:**

**Id Age Sex Code score**

**1 1 14 F a 99**

**2 2 12 M b 94**

**3 3 15 M c 98 4 4 10 F d 100**

**Source Code:**

# Create the 'df1' data frame

df1 <- data.frame(Id = 1:4, Age = c(14, 12, 15, 10))

# Create the 'df2' data frame

df2 <- data.frame(Id = 1:4, Sex = c("F", "M", "M", "F"), Code = c("a", "b", "c", "d"))

# Merge the 'df1' and 'df2' data frames

M <- merge(df1, df2)

# Print the final data frame

print(M)

# create df3

df3 <- data.frame(id2 = c(4, 3, 2, 1), score = c(100, 98, 94, 99))

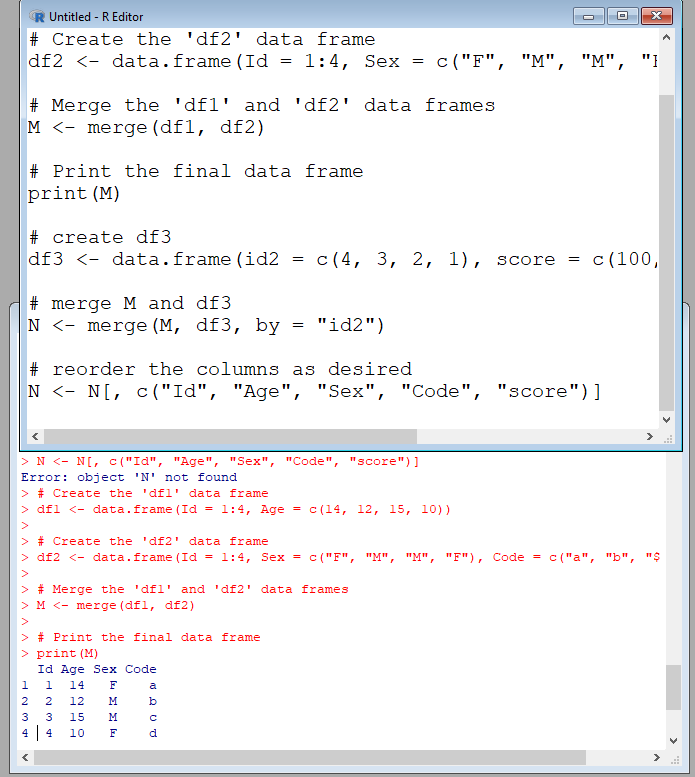
# merge M and df3

N <- merge(M, df3, by = "id2")

# reorder the columns as desired

N <- N[, c("Id", "Age", "Sex", "Code", "score")]

**OUTPUT:**

****

**14.** **Consider the previous one data frame N:**

**1) Remove the variables Sex and Code**

**2) From N, create a data frame:**

**values ind**

**1 1 Id**

**2 2 Id**

**3 3 Id**

**4 4 Id**

**5 14 Age**

**6 12 Age**

**7 15 Age**

**8 10 Age**

**9 99 score**

**10 94 score**

**11 98 score**

**12 100 score**

**Source Code:**

library(tidyr)

# Remove Sex and Code columns

N <- N[, c("Id", "Age", "score")]

# Reshape the data frame from wide to long format

N\_long <- gather(N, key = "variable", value = "value", -Id)

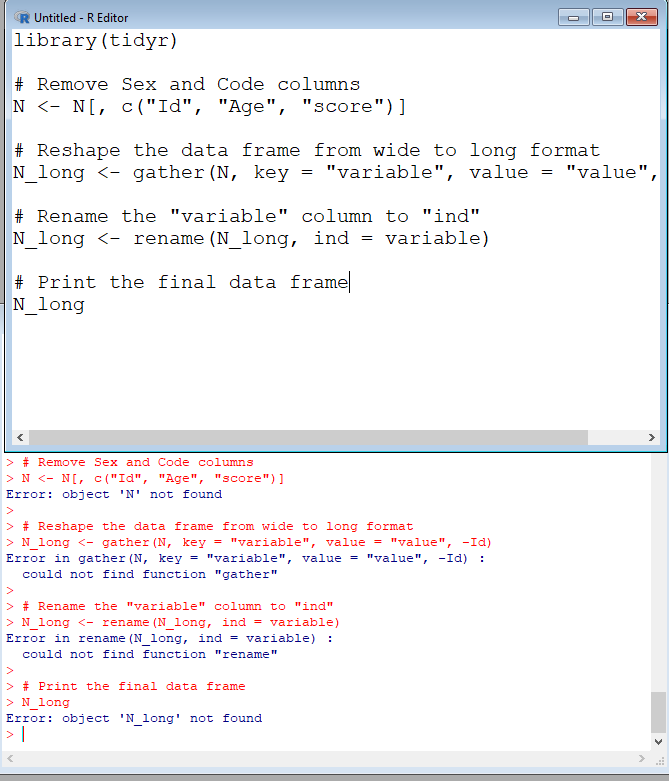
# Rename the "variable" column to "ind"

N\_long <- rename(N\_long, ind = variable)

# Print the final data frame

N\_long

**OUTPUT:**

****

**15.** **For this exercise, we’ll use the (built-in) dataset trees.**

**a) Make sure the object is a data frame, if not change it to a data frame.**

**b) Create a new data frame A:**

**&gt;A**

**Girth Height Volume**

**mean\_tree 13.24839 76 30.17097**

**min\_tree 8.30000 63 10.20000**

**max\_tree 20.60000 87 77.00000**

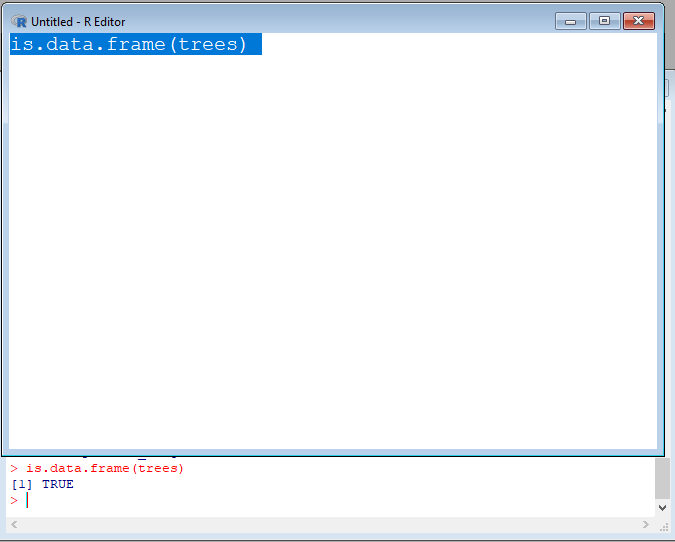
**sum\_tree 410.70000 2356 935.30000**

**Source Code:**

**(a)**

is.data.frame(trees)

**OUTPUT:**

****

**(b)**

# Compute the mean, min, max, and sum for each column

stats <- c(mean = "mean", min = "min", max = "max", sum = "sum")

# Compute the summary statistics for each column

summary\_stats <- aggregate(trees, by = list("mean\_tree", "min\_tree", "max\_tree", "sum\_tree"), stats)

# Rename the Group.1 columns to be the desired row names

rownames(summary\_stats) <- summary\_stats$Group.1

summary\_stats$Group.1 <- NULL

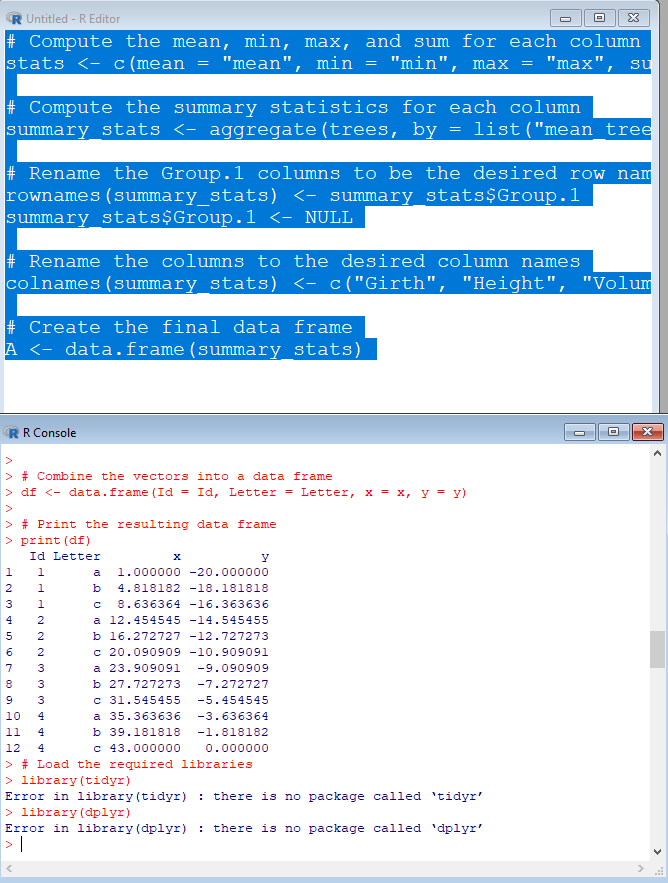
# Rename the columns to the desired column names

colnames(summary\_stats) <- c("Girth", "Height", "Volume")

# Create the final data frame

A <- data.frame(summary\_stats)

**OUTPUT:**

****