**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**ITA 0443 - STATISTICS WITH R PROGRAMMING FOR REAL TIME PROBLEM**

**DAY 2 – LAB EXERCISES**

**Reg No: 192124028**

**Name: KOVI SAI GANESH**

**IMPLEMENTATION OF VECTOR RECYCLING, APPLY FAMILY & RECURSION**

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

**# creating vector with**

**# 1 to 6 values**

**vec1=1:6**

**# creating vector with 1:2**

**# values**

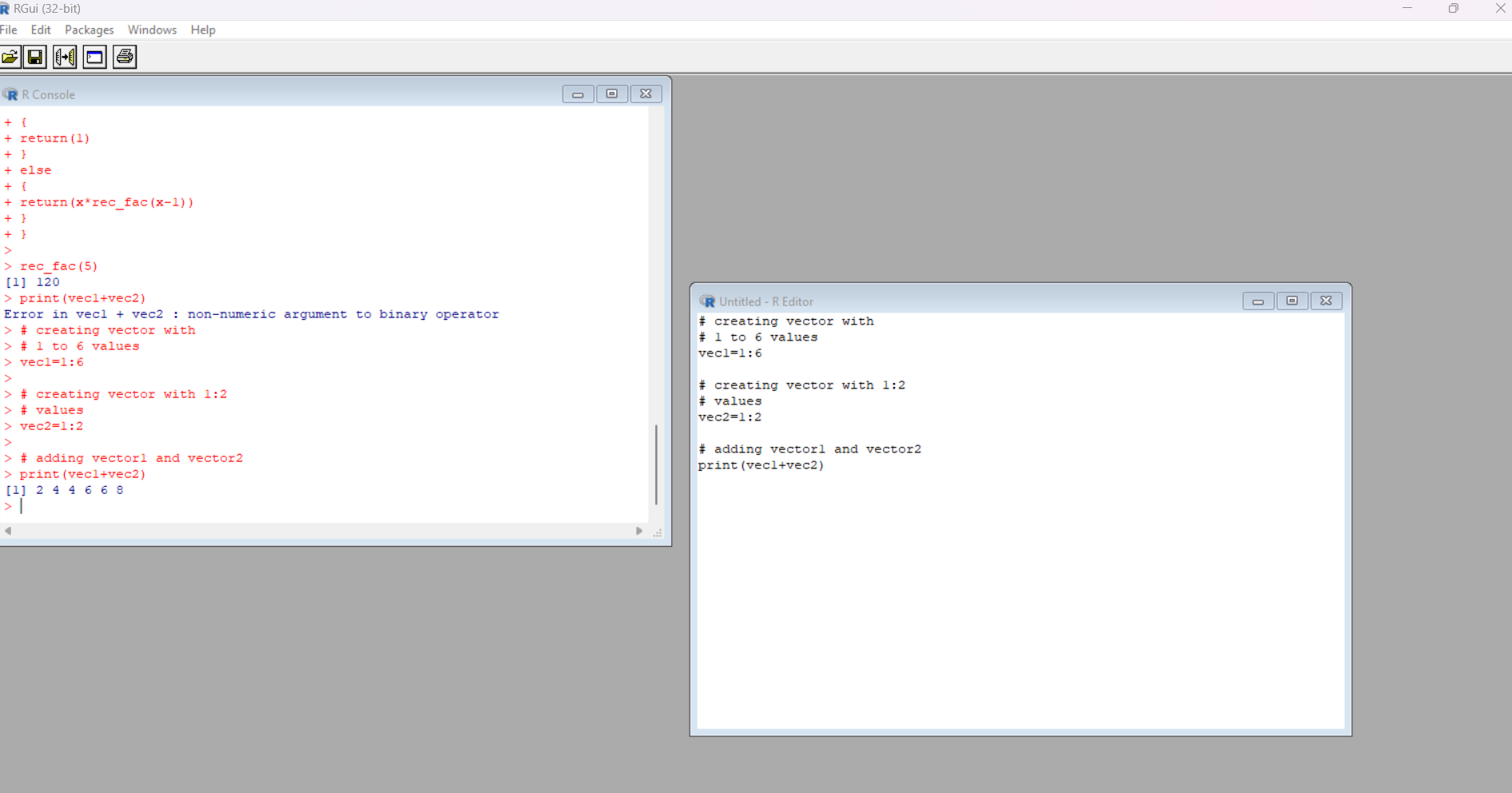
**vec2=1:2**

**# adding vector1 and vector2**

**print(vec1+vec2)**

**> print(vec1+vec2)**

**[1] 2 4 4 6 6 8**

****

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

**m1 <- matrix(C<-(1:10),nrow=5, ncol=6)**

**m1**

**a\_m1 <- apply(m1, 2, sum)**

**a\_m1**

**> a\_m1**

**[,1] [,2] [,3] [,4] [,5] [,6]**

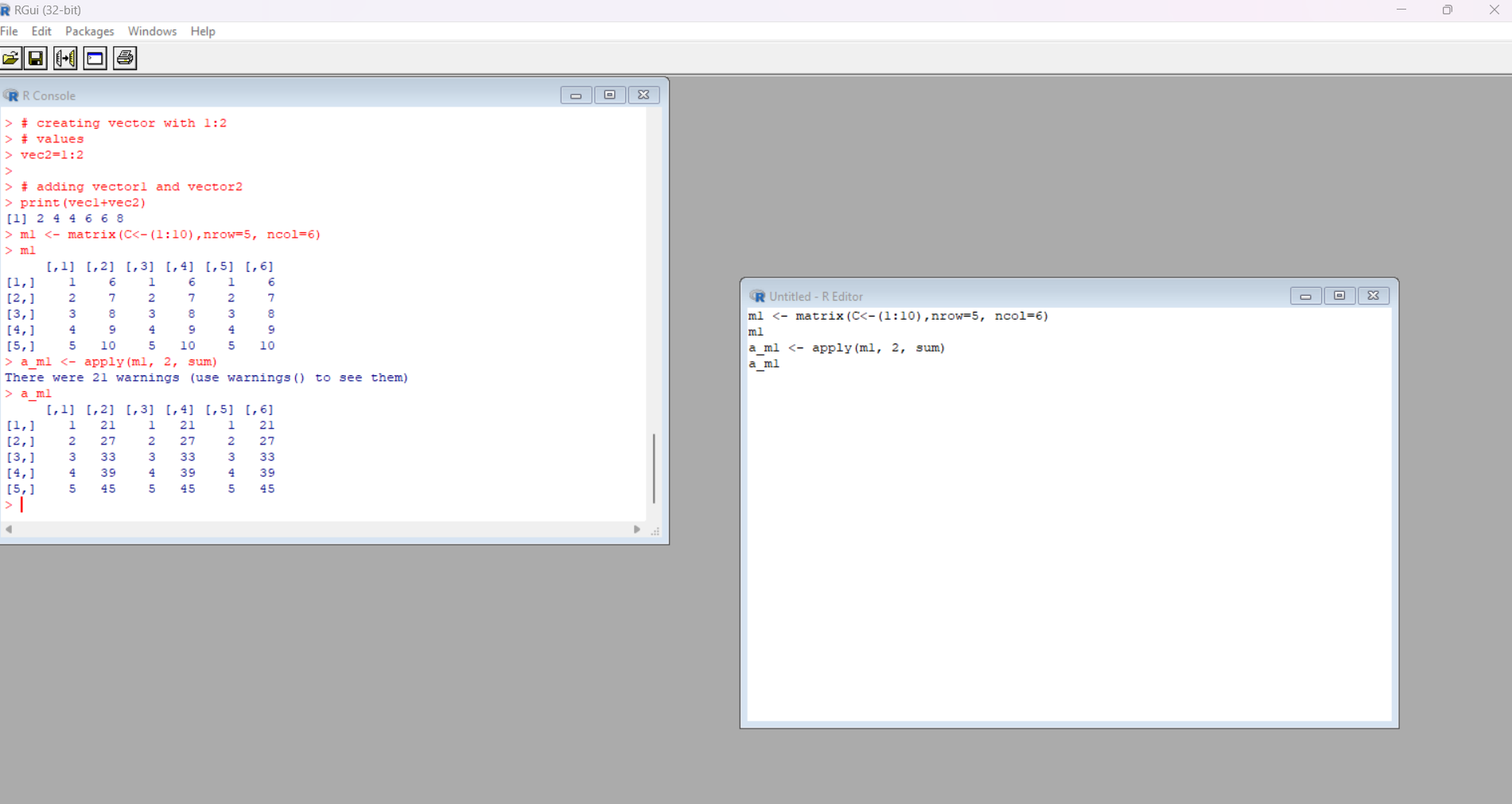
**[1,] 1 21 1 21 1 21**

**[2,] 2 27 2 27 2 27**

**[3,] 3 33 3 33 3 33**

**[4,] 4 39 4 39 4 39**

**[5,] 5 45 5 45 5 45**

****

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

**movies <- c("SPYDERMAN","BATMAN","VERTIGO","CHINATOWN")**

**movies\_lower <-lapply(movies, tolower)**

**str(movies\_lower)**

**> str(movies\_lower)**

**output:**

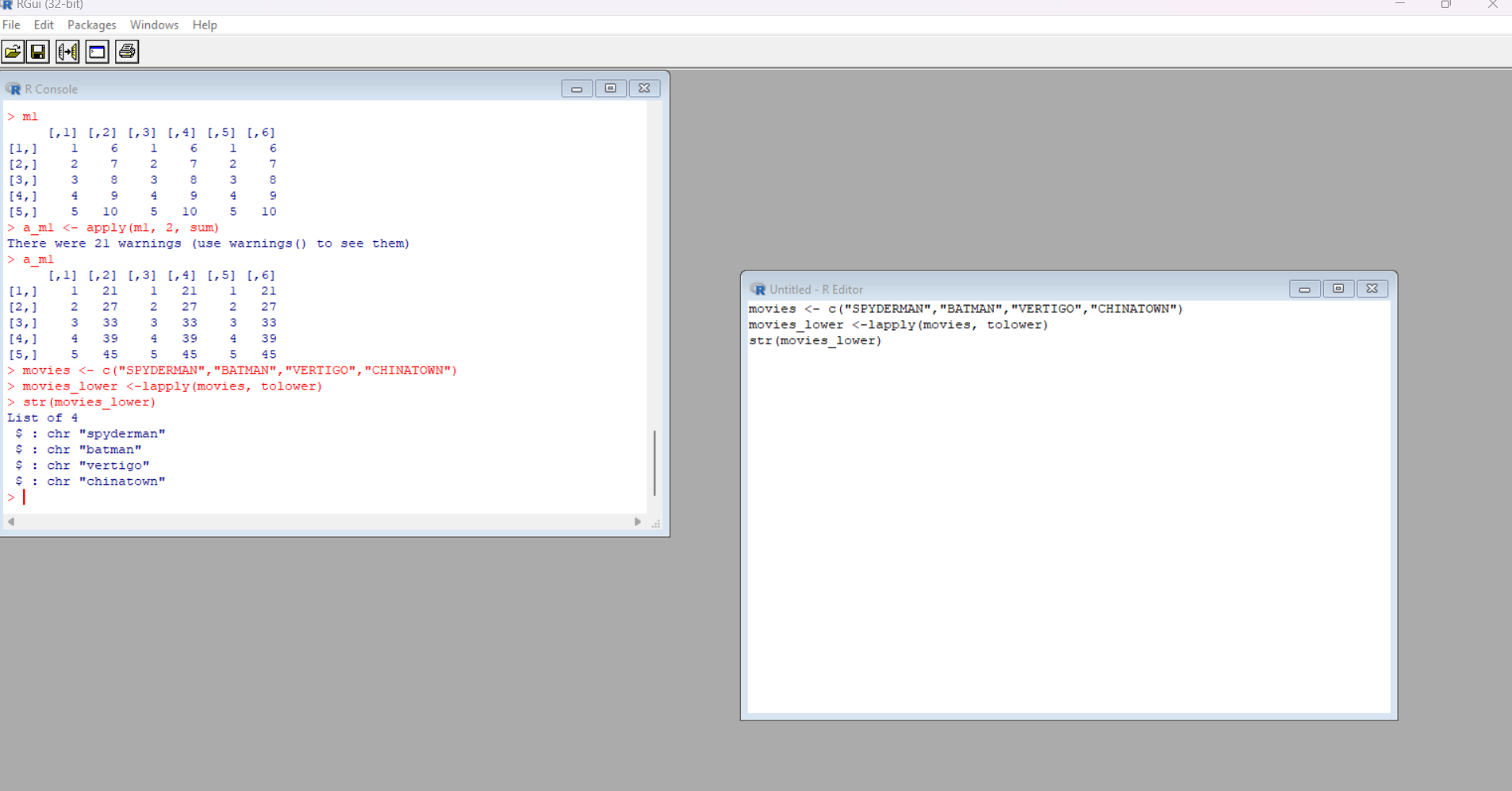
**List of 4**

**$ : chr "spyderman"**

**$ : chr "batman"**

**$ : chr "vertigo"**

**$ : chr "chinatown"**

****

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

**dt <- cars**

**lmn\_cars <- lapply(dt, min)**

**smn\_cars <- sapply(dt, min)**

**lmn\_cars**

**output :**

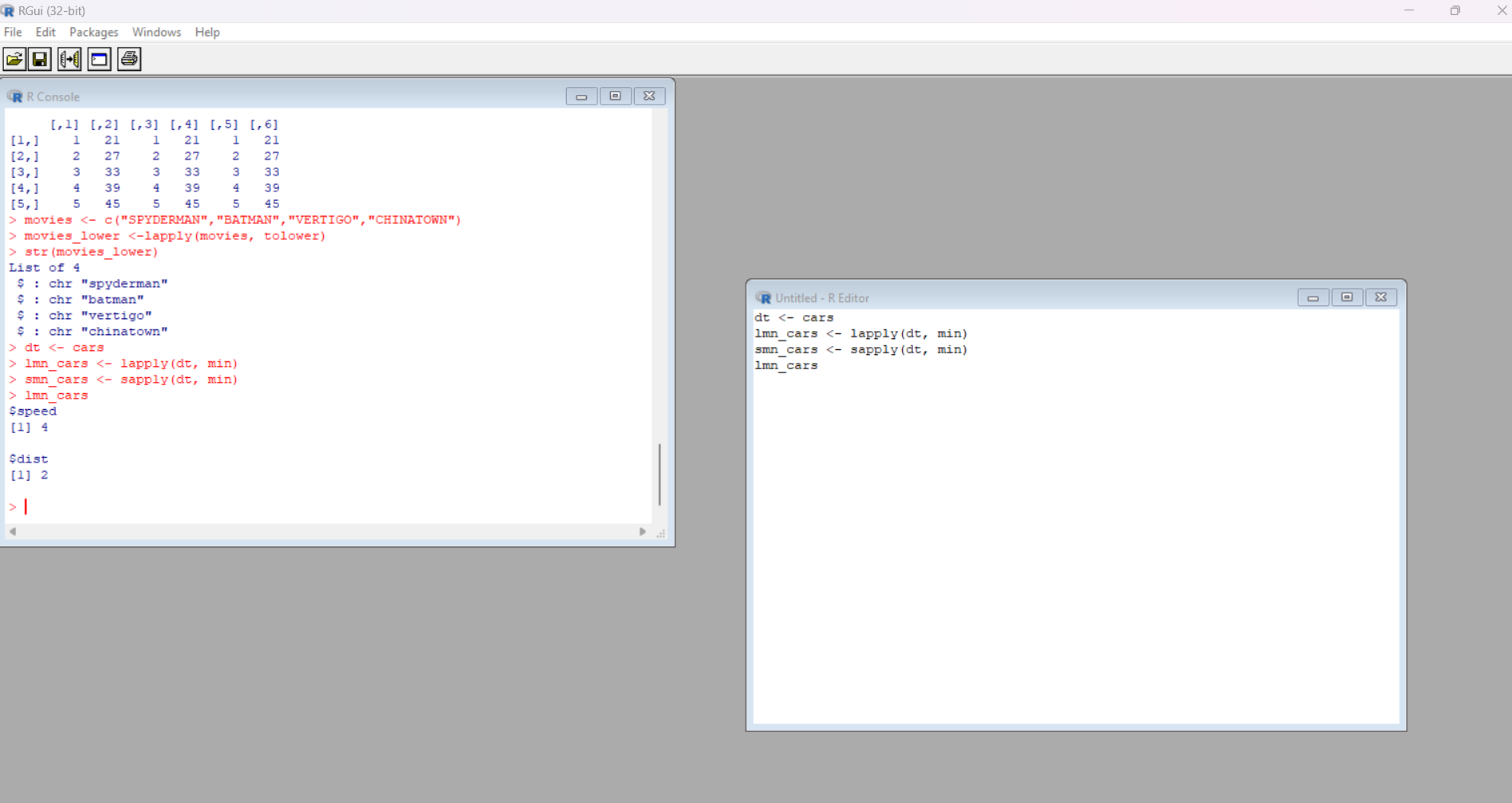
**> lmn\_cars**

**$speed**

**[1] 4**

**$dist**

**[1] 2**

****

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

**data(iris)**

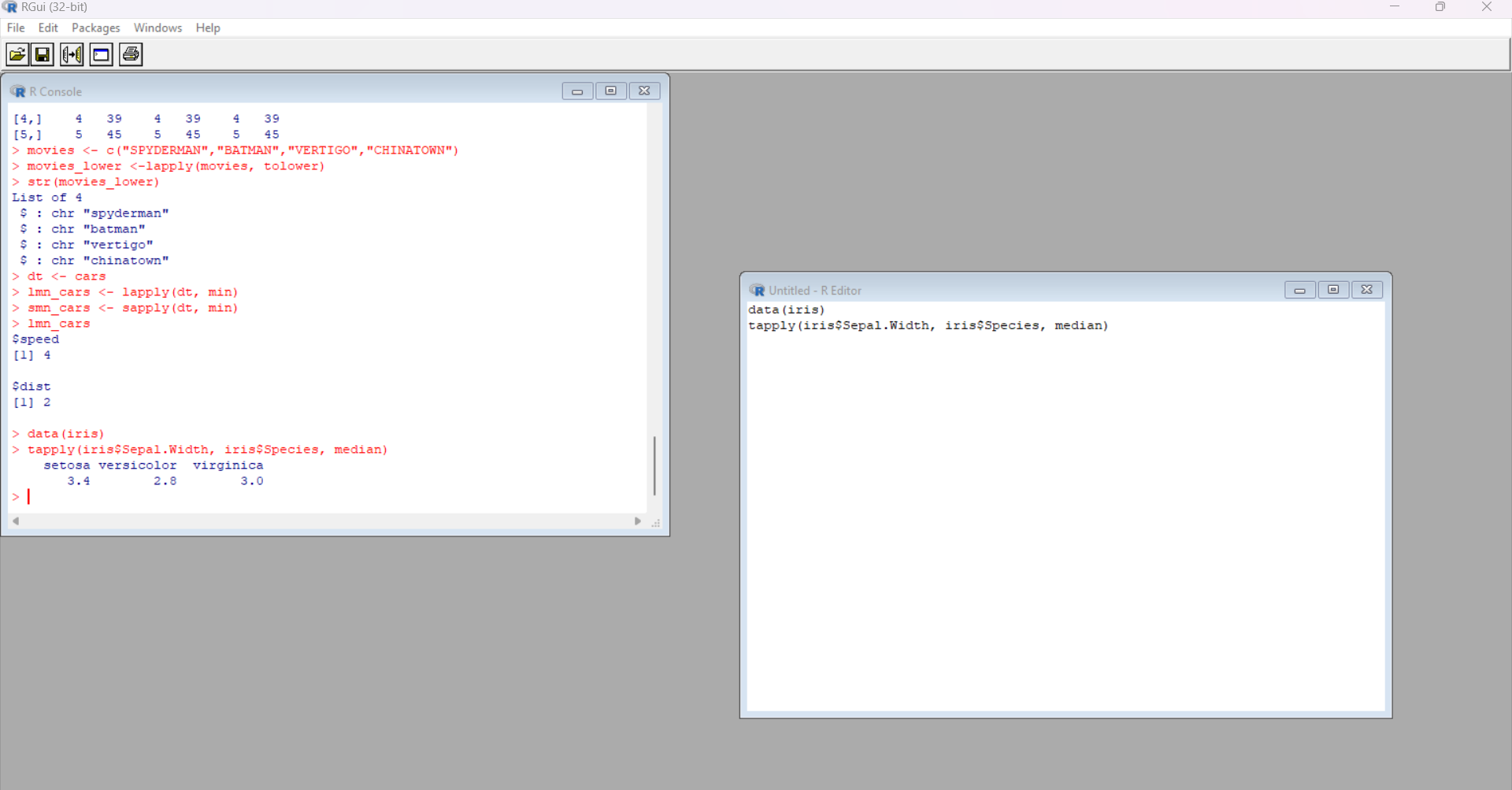
**tapply(iris$Sepal.Width, iris$Species, median)**

**OUTPUT:**

**tapply(iris$Sepal.Width, iris$Species, median)**

**setosa versicolor virginica**

**3.4 2.8 3.0**

****

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

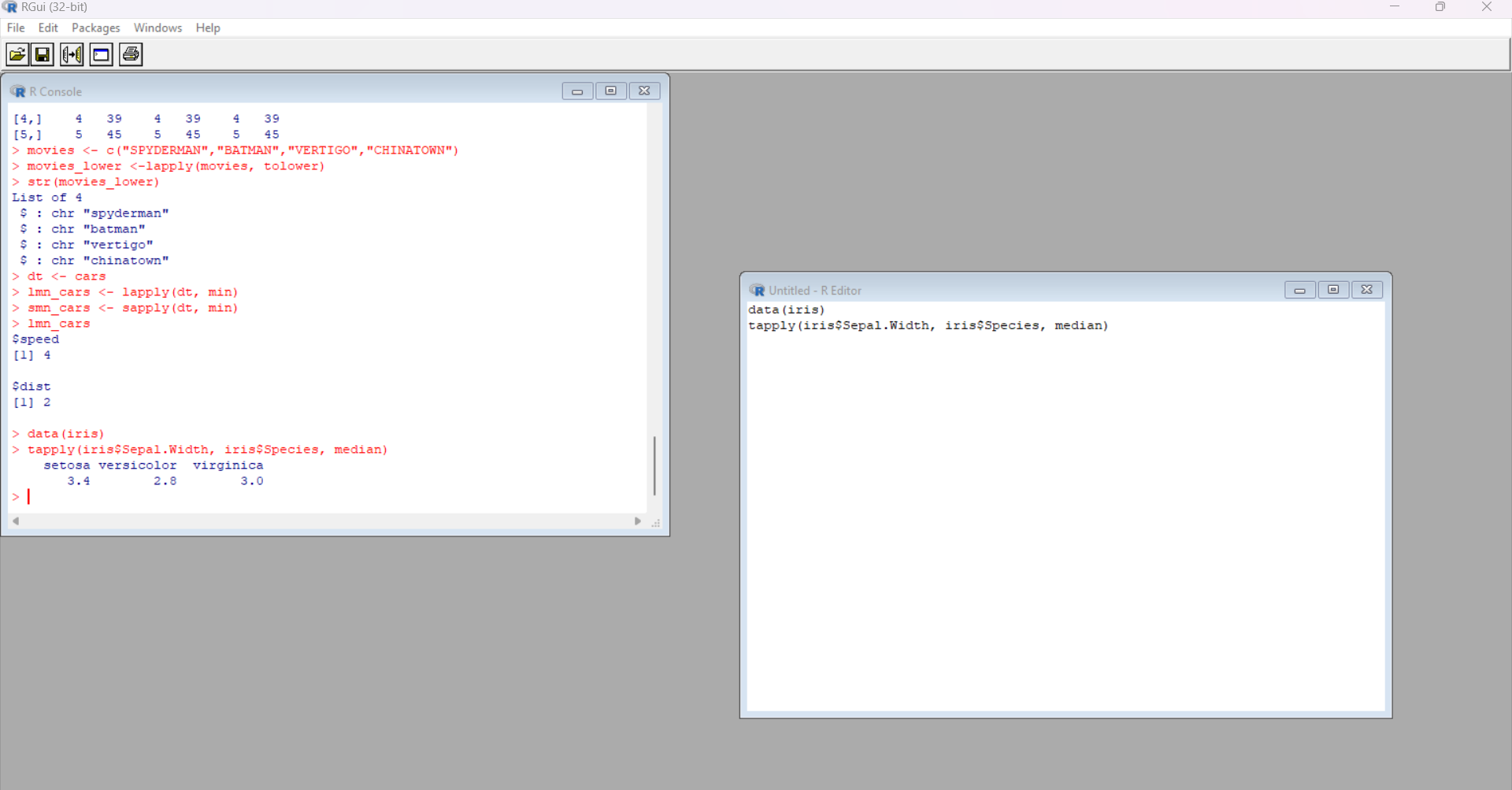
**A = list(c(1, 2, 3, 4))**

**> B = list(c(2, 5, 1, 6))**

**> result = mapply(prod, A, B)**

**> print(result)**

**[1] 1440**

****

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

**sum<-function(n){**

**if (n<=1){**

**return(n)**

**}else{**

**return(n+sum(n-1))**

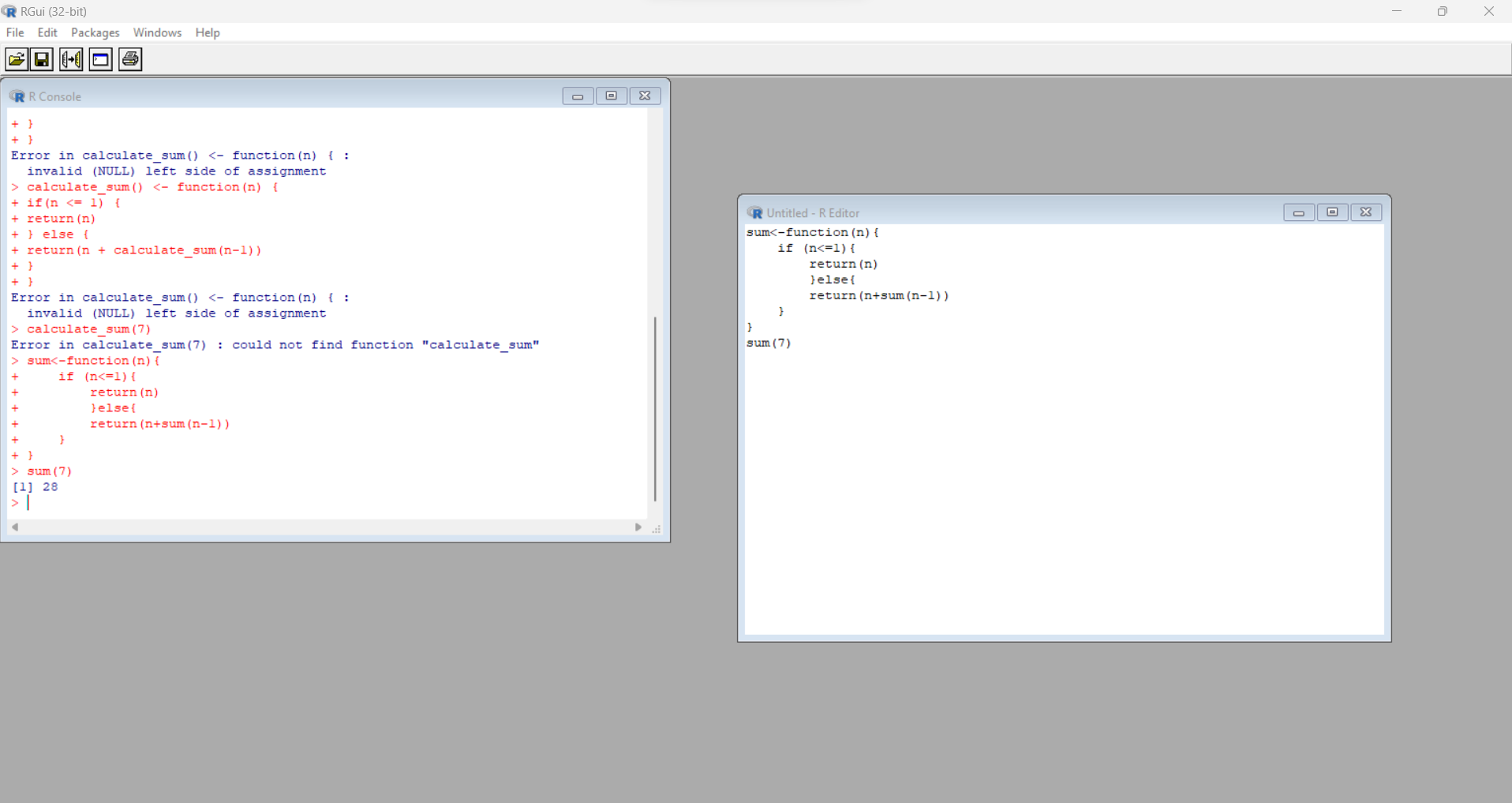
**}**

**}**

**sum(7)**

**OUTPUT :**

**> sum(7)**

****

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

**Fibonacci <- numeric(10)**

**Fibonacci[1] <- Fibonacci[2] <- 1**

**for (i in 3:10) Fibonacci[i] <- Fibonacci[i - 2] + Fibonacci[i - 1]**

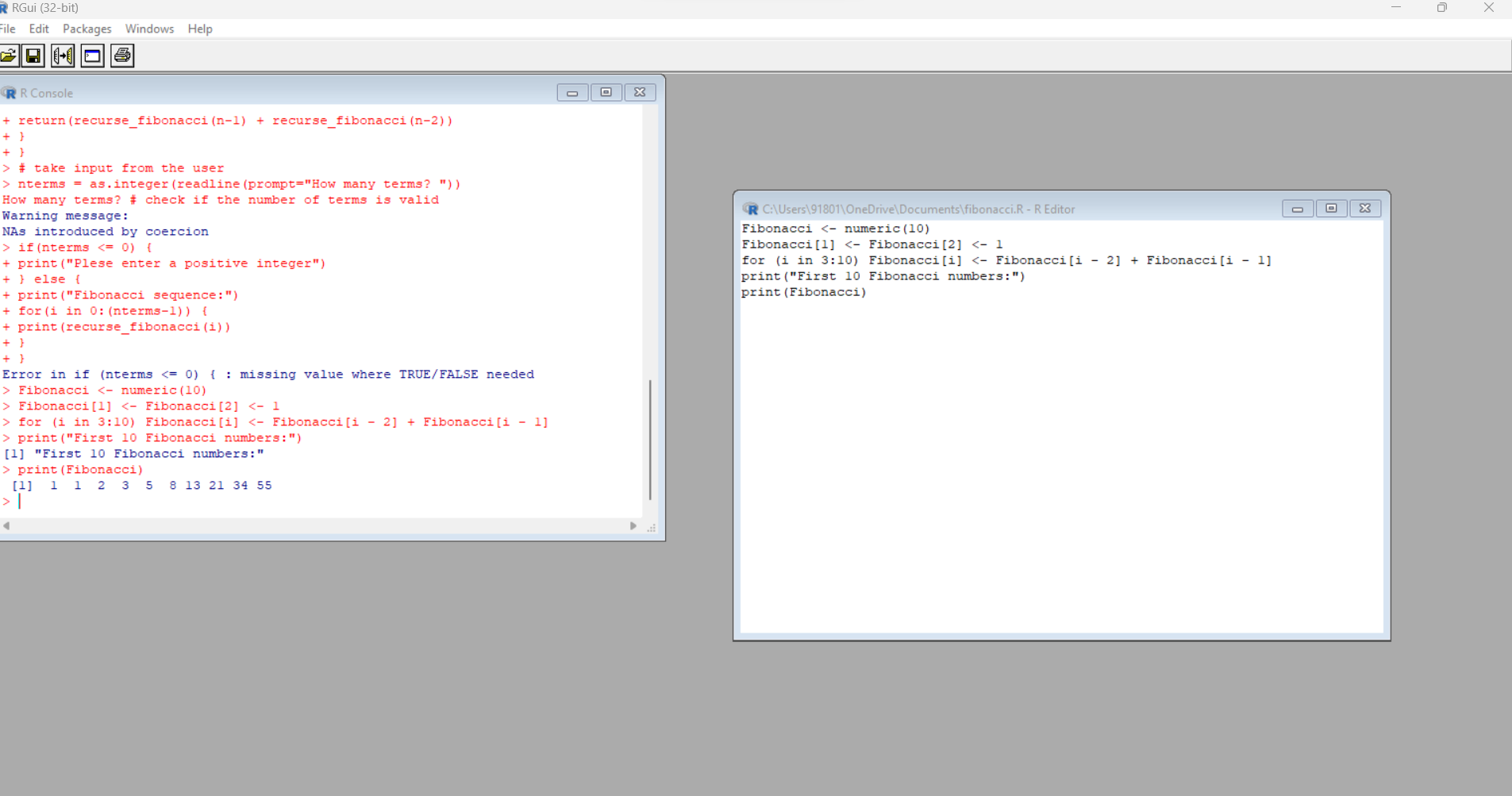
**print("First 10 Fibonacci numbers:")**

**print(Fibonacci)**

**OUTPUT:**

**> rec\_fac(5)**

**[1] 120**

****

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

**rec\_fac <- function(x){**

**if(x==0 || x==1)**

**{**

**return(1)**

**}**

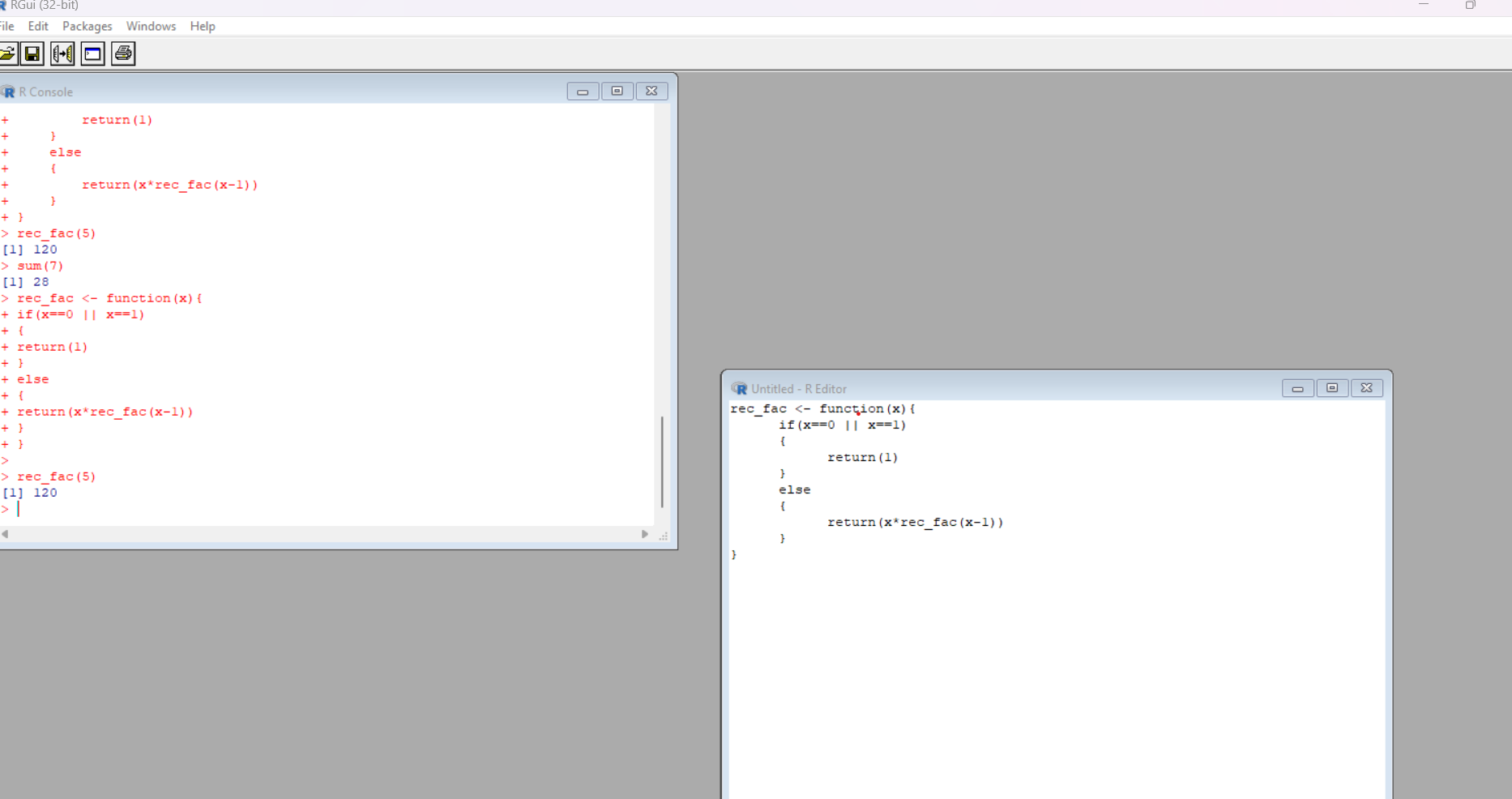
**else**

**{**

**return(x\*rec\_fac(x-1))**

**}**

**}**

****

**CREATION AND MANIPULATION OF DATAFRAMES IN R**

**Exercise 1**

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.

>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

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

CODE :

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

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

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

OUTPUT :

> 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

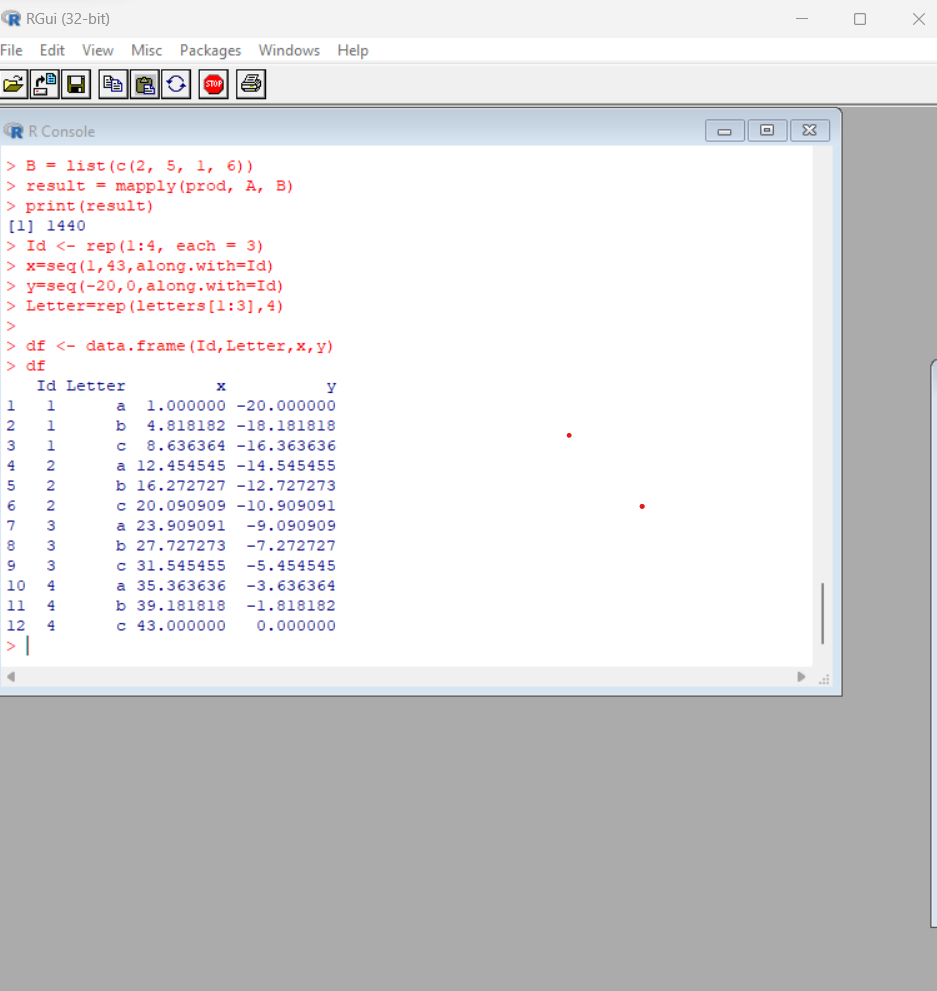
8 3 b 27.727273 -7.272727

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



**Exercise 2**

Using the data frame ‘df’ in Exercise1, Construct the following data frame. Id x.ay.ax.by.bx.cy.c 1 1 1.00000 -20.000000 4.818182 -18.181818 8.636364 -16.363636 4 2 12.45455 -14.545455 16.272727 -12.727273 20.090909 -10.909091 7 3 23.90909 -9.090909 27.727273 -7.272727 31.545455 -5.454545 10 4 35.36364 -3.636364 39.181818 -1.818182 43.000000 0.000000

**Exercise 3**

Create two data frame df1 and df2:

> df1

Id Age

1. 1 14
2. 2 12
3. 3 15
4. 4 10

> 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:

>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

CODE :

Id <- c(1:4)

> Age <- c(14,12,15,10)

> df1 <- data.frame(Id,Age)

>

> Sex <- c("F","M","M","F")

> Code <- letters[1:4]

> df2 <- data.frame(Id,Sex,Code)

> M <- merge(df1,df2, by = "Id")

>

> M

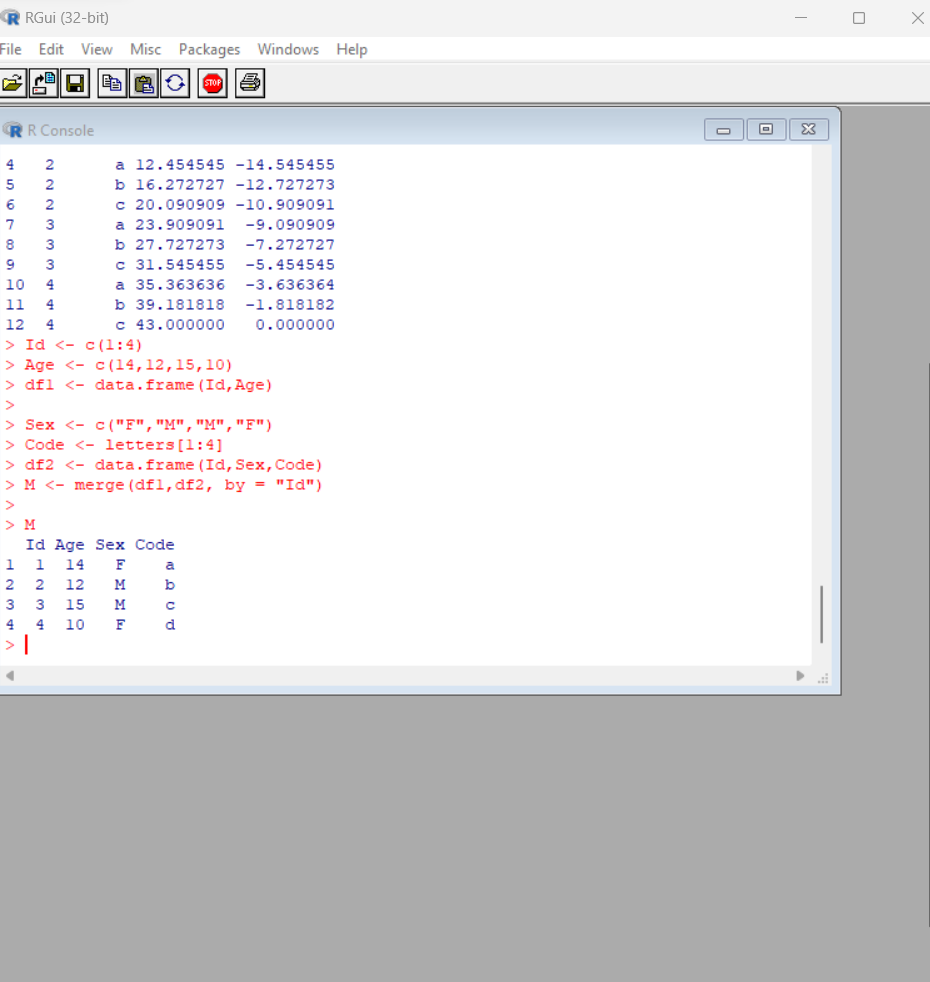
Id Age Sex Code

1 1 14 F a

2 2 12 M b

3 3 15 M c

4 4 10 F d



## **Exercise 4**

Create a data frame df3:

> df3 id2 score 1 4 100

1. 3 98
2. 2 94
3. 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

CODE :

id2 <- 4:1

> score <- c(100,98,94,99)

> df3 <- data.frame(id2,score)

>

> N=merge(M,df3,by.x='Id',by.y='id2')

> N

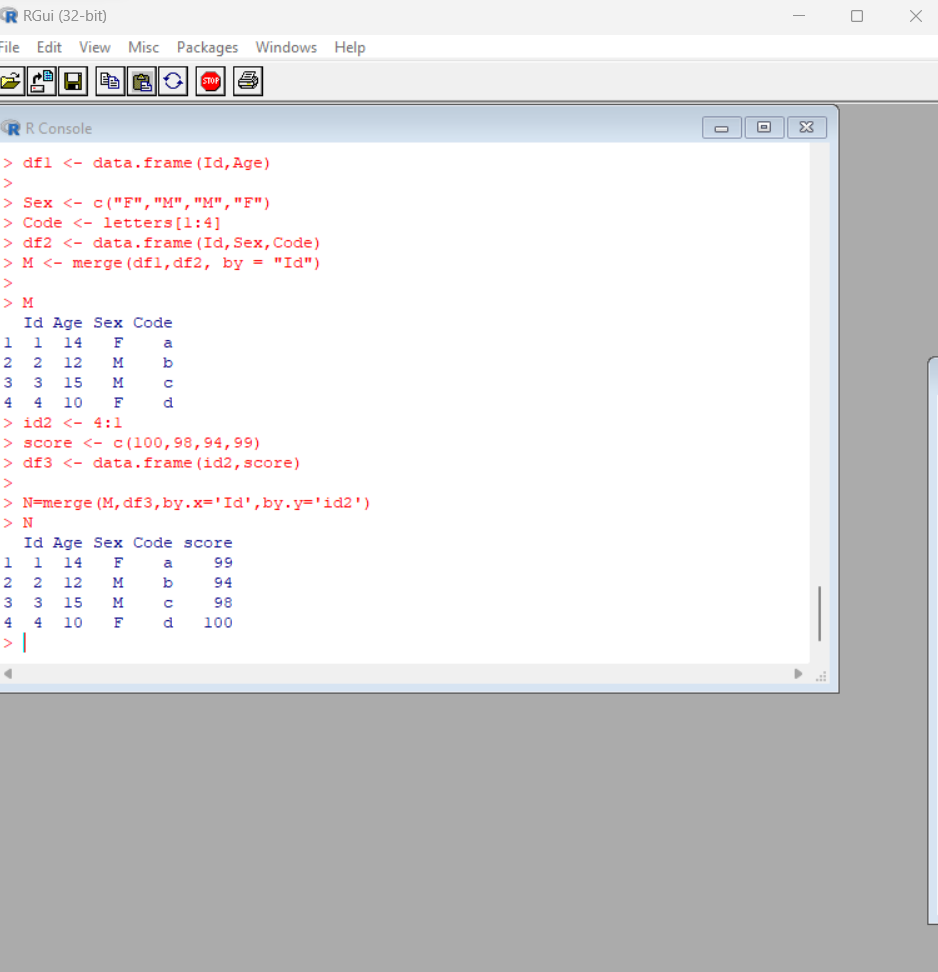
Id Age Sex Code score

1 1 14 F a 99

2 2 12 M b 94

3 3 15 M c 98

4 4 10 F d 100



## **Exercise 5**

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

CODE :

> N[,c("Sex")]=NULL

> N[,c("Code")]=NULL

> stack(N)

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

> N

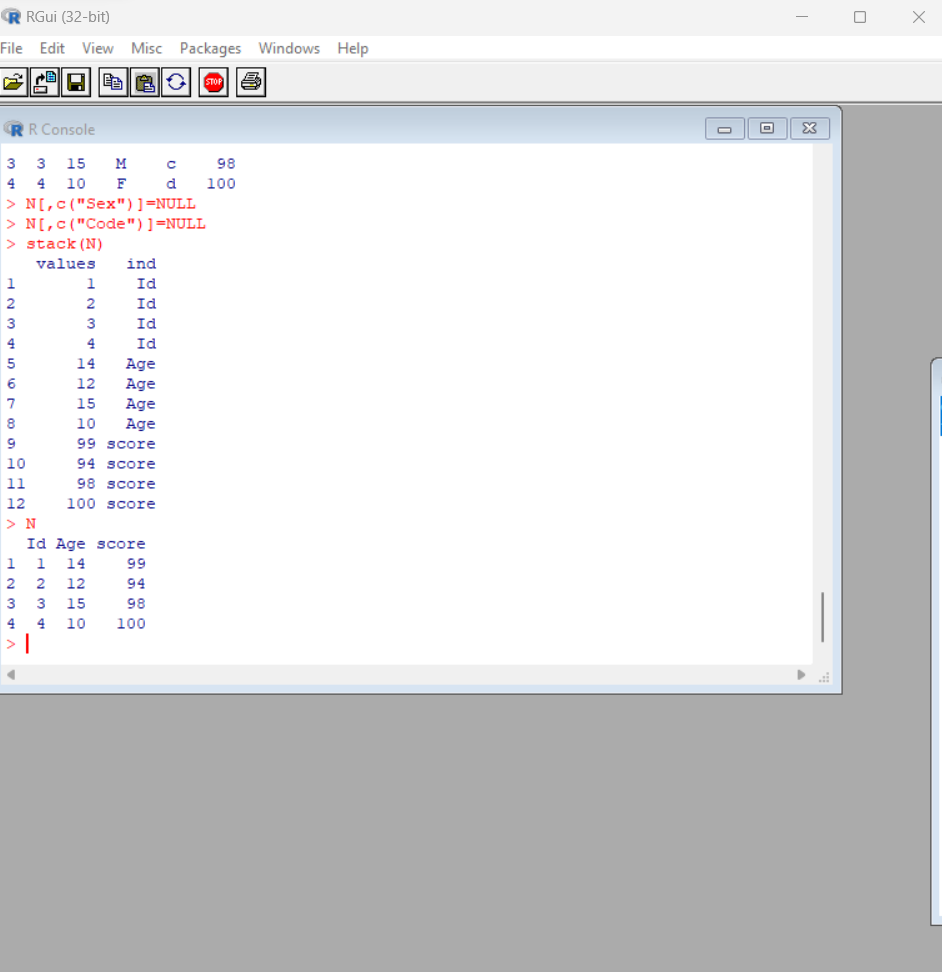
Id Age score

1 1 14 99

2 2 12 94

3 3 15 98

4 4 10 100



## **Exercise 6**

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

1. Make sure the object is a data frame, if not change it to a data frame.
2. Create a new data frame A:

>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

> A <- trees

> mean\_tree=apply(trees,2,mean)

> max\_tree=apply(trees,2,max)

> min\_tree=apply(trees,2,min)

> sum\_tree=apply(trees,2,sum)

There were 50 or more warnings (use warnings() to see the first 50)

>

> A=data.frame(mean\_tree,min\_tree,max\_tree,sum\_tree) # The expected table is the transpose of A.

Error in data.frame(mean\_tree, min\_tree, max\_tree, sum\_tree) :

arguments imply differing number of rows: 3, 31

>

> A <- t(A)

>

> A

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]

Girth 8.3 8.6 8.8 10.5 10.7 10.8 11.0 11.0 11.1 11.2 11.3 11.4 11.4

Height 70.0 65.0 63.0 72.0 81.0 83.0 66.0 75.0 80.0 75.0 79.0 76.0 76.0

Volume 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 24.2 21.0 21.4

[,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]

Girth 11.7 12.0 12.9 12.9 13.3 13.7 13.8 14.0 14.2 14.5 16.0 16.3

Height 69.0 75.0 74.0 85.0 86.0 71.0 64.0 78.0 80.0 74.0 72.0 77.0

Volume 21.3 19.1 22.2 33.8 27.4 25.7 24.9 34.5 31.7 36.3 38.3 42.6

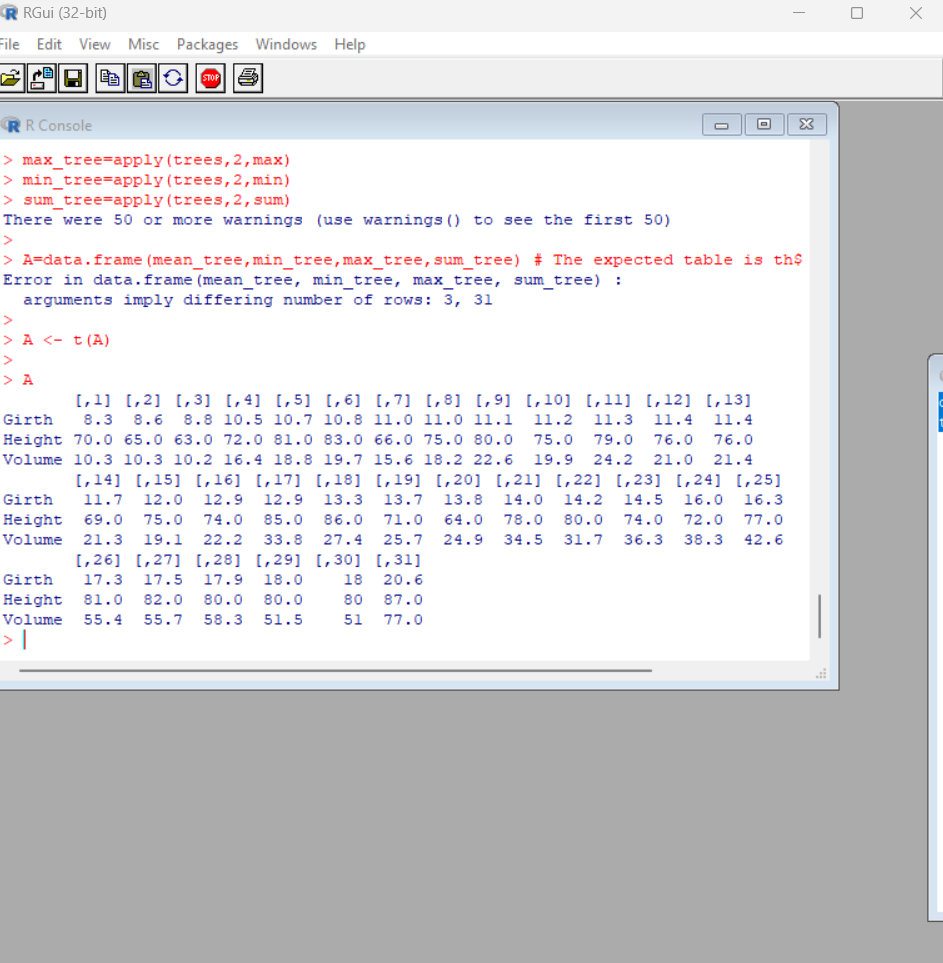
[,26] [,27] [,28] [,29] [,30] [,31]

Girth 17.3 17.5 17.9 18.0 18 20.6

Height 81.0 82.0 80.0 80.0 80 87.0

Volume 55.4 55.7 58.3 51.5 51 77.0

>



## **Exercise 7**

Consider the data frame A:

1)Order the entire data frame by the first column.

2)Rename the row names as follows: mean, min, max, tree

> A[order(A[,1]),]

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]

Girth 8.3 8.6 8.8 10.5 10.7 10.8 11.0 11.0 11.1 11.2 11.3 11.4 11.4

Volume 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 24.2 21.0 21.4

Height 70.0 65.0 63.0 72.0 81.0 83.0 66.0 75.0 80.0 75.0 79.0 76.0 76.0

[,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]

Girth 11.7 12.0 12.9 12.9 13.3 13.7 13.8 14.0 14.2 14.5 16.0 16.3

Volume 21.3 19.1 22.2 33.8 27.4 25.7 24.9 34.5 31.7 36.3 38.3 42.6

Height 69.0 75.0 74.0 85.0 86.0 71.0 64.0 78.0 80.0 74.0 72.0 77.0

[,26] [,27] [,28] [,29] [,30] [,31]

Girth 17.3 17.5 17.9 18.0 18 20.6

Volume 55.4 55.7 58.3 51.5 51 77.0

Height 81.0 82.0 80.0 80.0 80 87.0

> row.names(A)

[1] "Girth" "Height" "Volume"

> row.names(A) <- c("mean","min","max","tree")

Error in dimnames(x) <- dn :

length of 'dimnames' [1] not equal to array extent

>

> row.names(A) <- c("mean","min","max")

> a

Error: object 'a' not found

> A

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]

mean 8.3 8.6 8.8 10.5 10.7 10.8 11.0 11.0 11.1 11.2 11.3 11.4 11.4 11.7

min 70.0 65.0 63.0 72.0 81.0 83.0 66.0 75.0 80.0 75.0 79.0 76.0 76.0 69.0

max 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 24.2 21.0 21.4 21.3

[,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]

mean 12.0 12.9 12.9 13.3 13.7 13.8 14.0 14.2 14.5 16.0 16.3 17.3

min 75.0 74.0 85.0 86.0 71.0 64.0 78.0 80.0 74.0 72.0 77.0 81.0

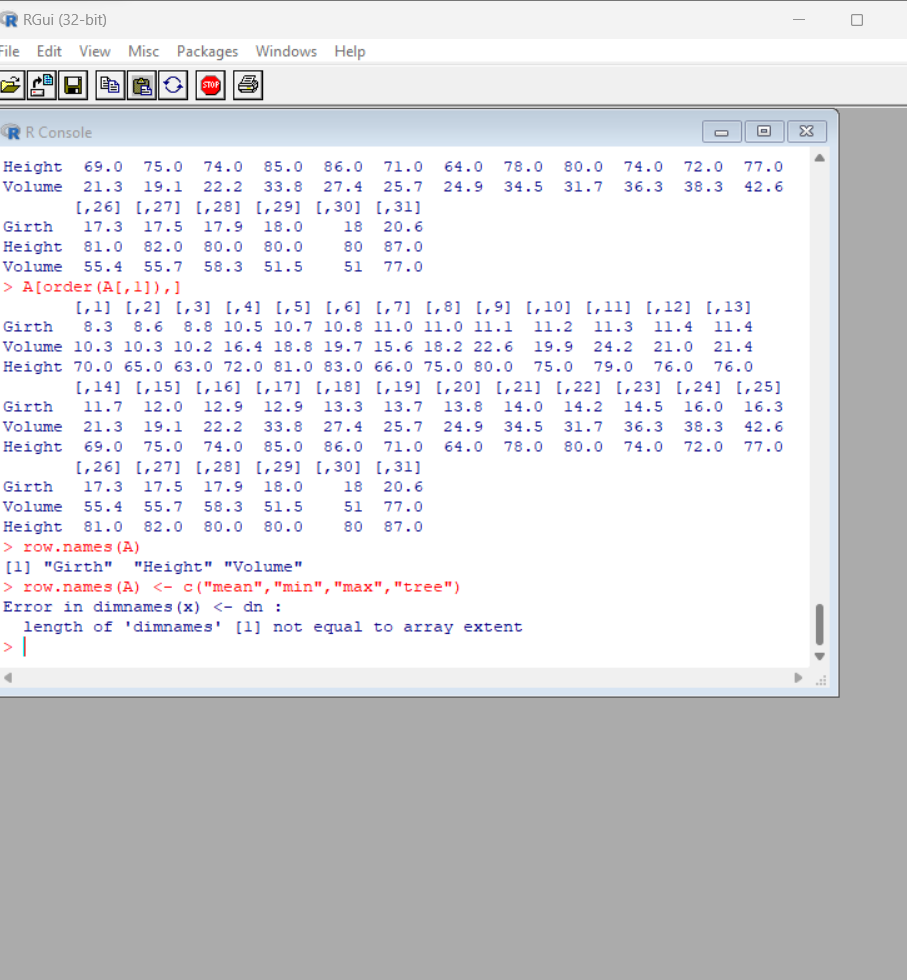
max 19.1 22.2 33.8 27.4 25.7 24.9 34.5 31.7 36.3 38.3 42.6 55.4

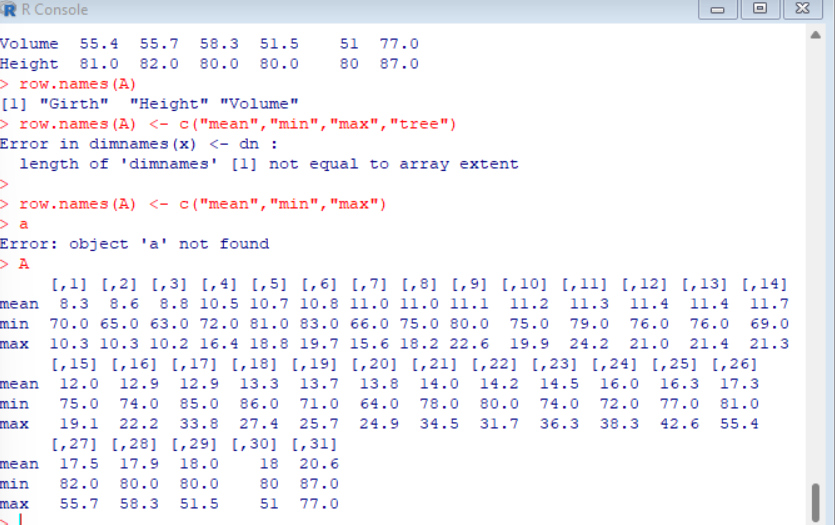
[,27] [,28] [,29] [,30] [,31]

mean 17.5 17.9 18.0 18 20.6

min 82.0 80.0 80.0 80 87.0

max 55.7 58.3 51.5 51 77.0





## **Exercise 8**

Create an empty data frame with column types:

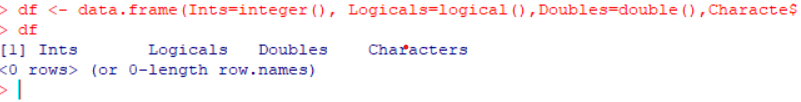
>df

IntsLogicals Doubles Characters

(or 0-length row.names)

> df <- data.frame(Ints=integer(), Logicals=logical(),Doubles=double(),Characters=character())

> df



## **Exercise 9**

Create a data frame XY

X=c(1,2,3,1,4,5,2)

Y=c(0,3,2,0,5,9,3)

> XY

X Y

1. 1 0
2. 2 3
3. 3 2
4. 1 0
5. 4 5
6. 5 9
7. 2 3
8. look at duplicated elements using a provided R function.
9. keep only the unique lines on XY using a provided R function.

> XY <- data.frame(X=c(1,2,3,1,4,5,2),Y=c(0,3,2,0,5,9,3))

>

> XY

X Y

1 1 0

2 2 3

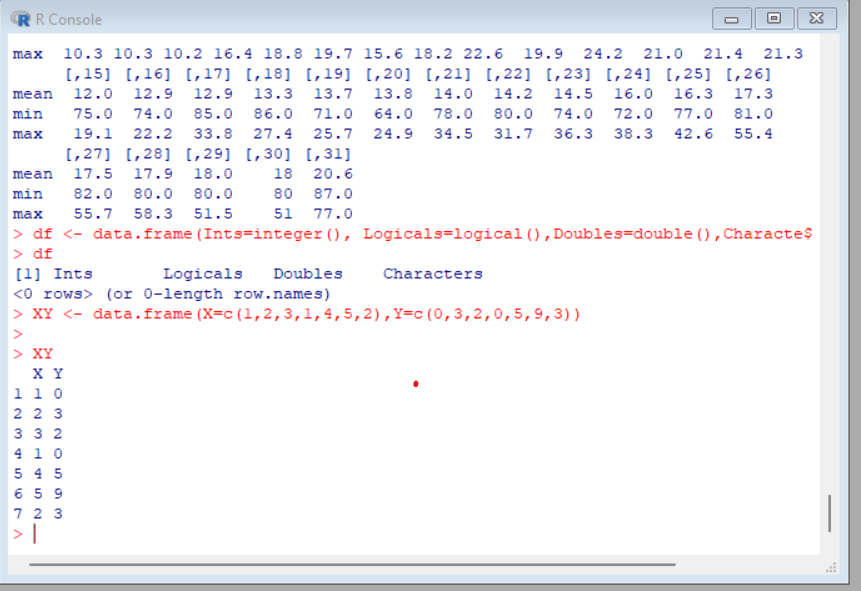
3 3 2

4 1 0

5 4 5

6 5 9

7 2 3



> duplicated(XY)

[1] FALSE FALSE FALSE TRUE FALSE FALSE TRUE

unique(XY)

X Y

1 1 0

2 2 3

3 3 2

5 4 5

6 5 9

## **Exercise 10**

Use the (built-in) dataset Titanic.

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

> str(Titanic)

'table' num [1:4, 1:2, 1:2, 1:2] 0 0 35 0 0 0 17 0 118 154 ...

- attr(\*, "dimnames")=List of 4

..$ Class : chr [1:4] "1st" "2nd" "3rd" "Crew"

..$ Sex : chr [1:2] "Male" "Female"

..$ Age : chr [1:2] "Child" "Adult"

..$ Survived: chr [1:2] "No" "Yes"

> Tit <- data.frame(Titanic)

> Tit

Class Sex Age Survived Freq

1 1st Male Child No 0

2 2nd Male Child No 0

3 3rd Male Child No 35

4 Crew Male Child No 0

5 1st Female Child No 0

6 2nd Female Child No 0

7 3rd Female Child No 17

8 Crew Female Child No 0

9 1st Male Adult No 118

10 2nd Male Adult No 154

11 3rd Male Adult No 387

12 Crew Male Adult No 670

13 1st Female Adult No 4

14 2nd Female Adult No 13

15 3rd Female Adult No 89

16 Crew Female Adult No 3

17 1st Male Child Yes 5

18 2nd Male Child Yes 11

19 3rd Male Child Yes 13

20 Crew Male Child Yes 0

21 1st Female Child Yes 1

22 2nd Female Child Yes 13

23 3rd Female Child Yes 14

24 Crew Female Child Yes 0

25 1st Male Adult Yes 57

26 2nd Male Adult Yes 14

27 3rd Male Adult Yes 75

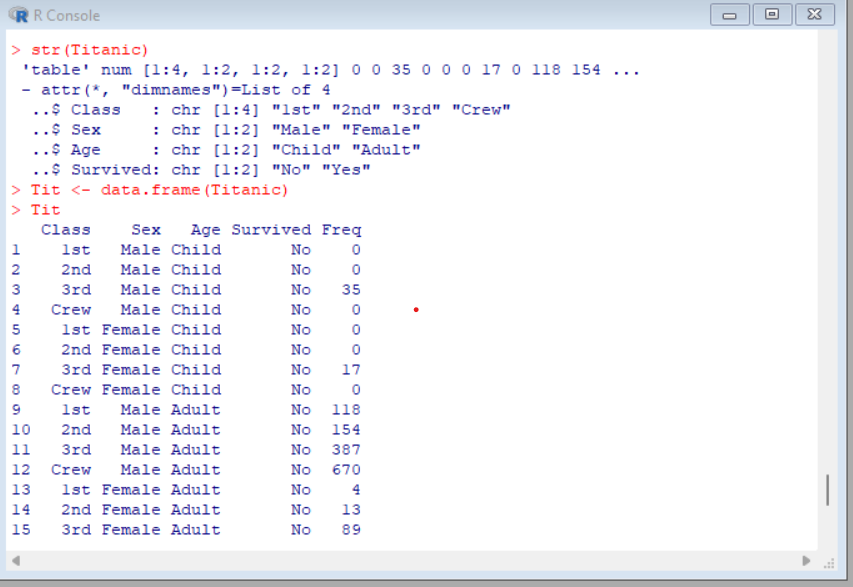
28 Crew Male Adult Yes 192

29 1st Female Adult Yes 140

30 2nd Female Adult Yes 80

31 3rd Female Adult Yes 76

32 Crew Female Adult Yes 20



1. Define a data frame with value 1st in Class variable, and value NO in Survived variable and variables Sex, Age and Freq.

Sex Age Freq

1 Male Child 0

5 Female Child 0

9 Male Adult 118

13 Female Adult 4

> df <- subset(Tit, subset = Class=='1st' & Survived=='No',select=c(Sex,Age,Freq))

>

> df

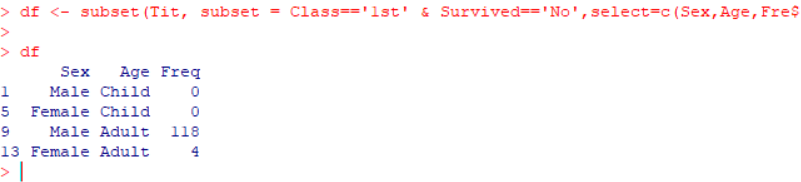
Sex Age Freq

1 Male Child 0

5 Female Child 0

9 Male Adult 118

13 Female Adult 4



MERGING DATAFRAMES

## **Exercise 11 a)**

Create the following dataframes to merge:

buildings<- data.frame(location=c(1, 2, 3), name=c("building1", "building2","building3"))

data <- data.frame(survey=c(1,1,1,2,2,2),location=c(1,2,3,2,3,1),efficiency=c(51,64,70,7,80,58))

The dataframes, *buildings*and *data*have a common key variable called, “location”.

Use the merge() function to merge the two dataframes by “location”, into a new dataframe,“buildingStats”.

> buildings <- data.frame(location=c(1, 2, 3), name=c("building1", "building2", "building3"))

> data <- data.frame(survey=c(1,1,1,2,2,2), location=c(1,2,3,2,3,1),

+

+ efficiency=c(51,64,70,71,80,58))

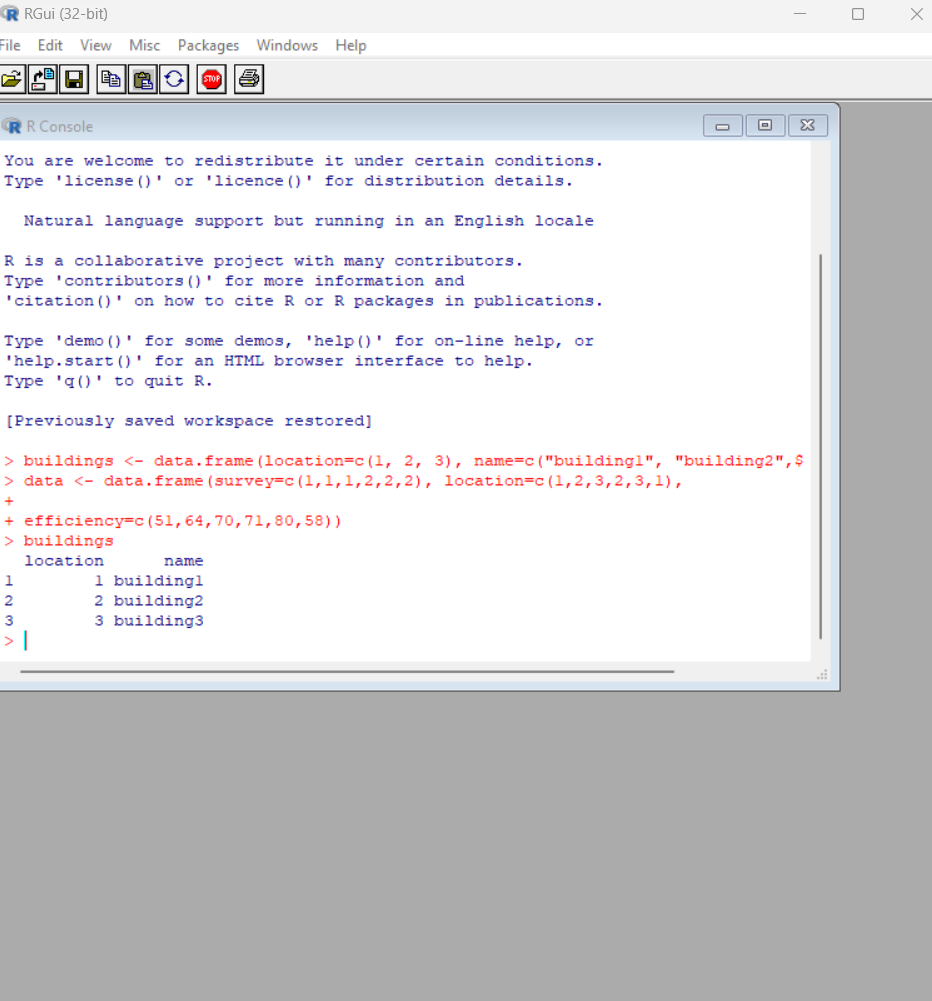
> buildings

location name

1 1 building1

2 2 building2

3 3 building3



## **Exercise 11 b)**

Give the dataframes different key variable names:

buildings<- data.frame(location=c(1, 2, 3), name=c("building1","building2", "building3"))

data <- data.frame(survey=c(1,1,1,2,2,2), LocationID=c(1,2,3,2,3,1), efficiency=c(51,64,70,71,80,58))

The dataframes, buildings and data have corresponding variables called, location, and LocationID. Use the merge() function to merge the columns of the two dataframes by the corresponding variables.

> buildings <- data.frame(location=c(1, 2, 3), name=c("building1", "building2", "building3"))

>

> data <- data.frame(survey=c(1,1,1,2,2,2), LocationID=c(1,2,3,2,3,1),

+

+ efficiency=c(51,64,70,71,80,58))

> buildings

location name

1 1 building1

2 2 building2

3 3 building3

DIFFERENT TYPES OF MERGE IN R

**Exercise 12a)InnerJoin:**

The R merge() function automatically joins the frames by common variable names. In that case, demonstrate how you would perform the merge in **Exercise 11a** without specifying the key variable.

**Exercise 12b)OuterJoin:**

Merge the two dataframes from **Exercise 11a**. Use the “all=” parameter in the merge() function to return all records from both tables. Also, merge with the key variable, “location”.

**Exercise 12c)Left Join:**

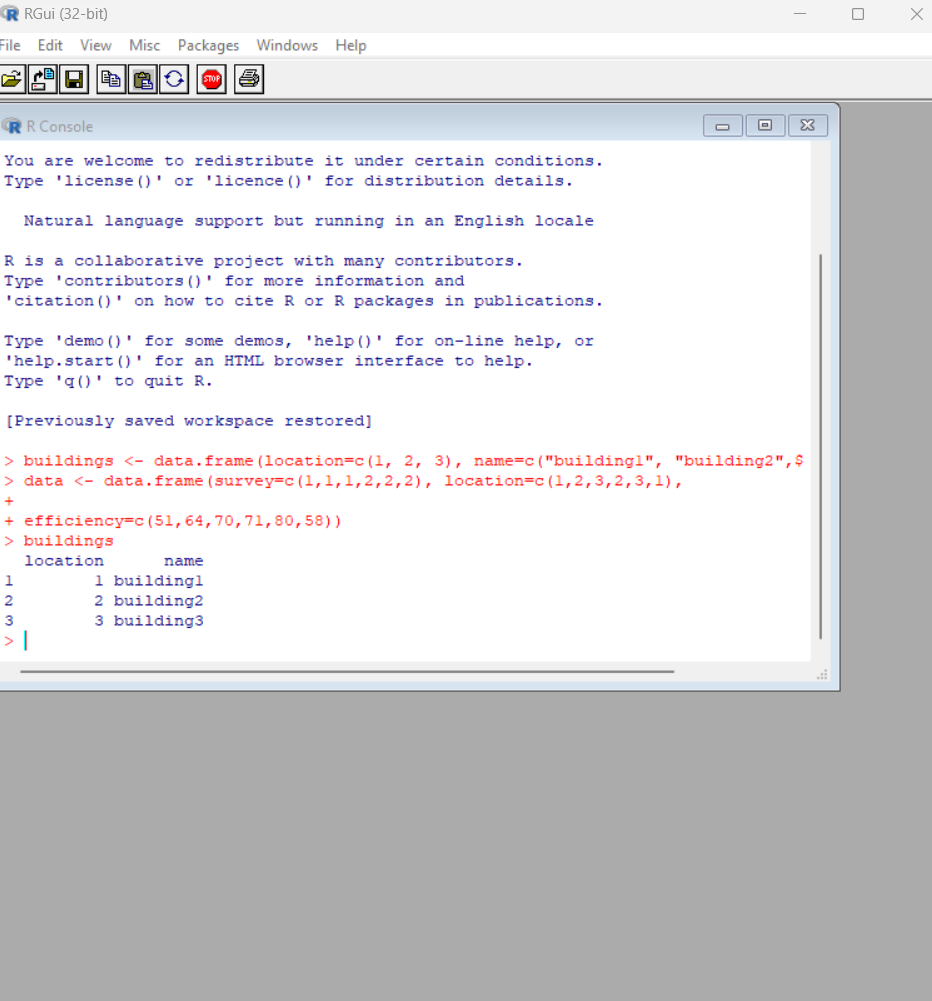
Merge the two dataframes from **Exercise 11a**, and return all rows from the left table. Specify the matching key from **Exercise 11a.**

**Exercise 12d)Right Join:**

Merge the two dataframes from **Exercise 11a,** and return all rows from the right table. Use the matching key from **Exercise 11a** to return matching rows from the left table.

**Exercise 12e)Cross Join:**

Merge the two dataframes from **Exercise 11a**, into a “Cross Join” with each row of “buildings” matched to each row of “data”. What new column names are created in “buildingStats”?



### **Exercise 13MergingDataframe rows:**

To join two data frames (datasets) vertically, use the rbind function. The two data frames must have the same variables, but they do not have to be in the same order.

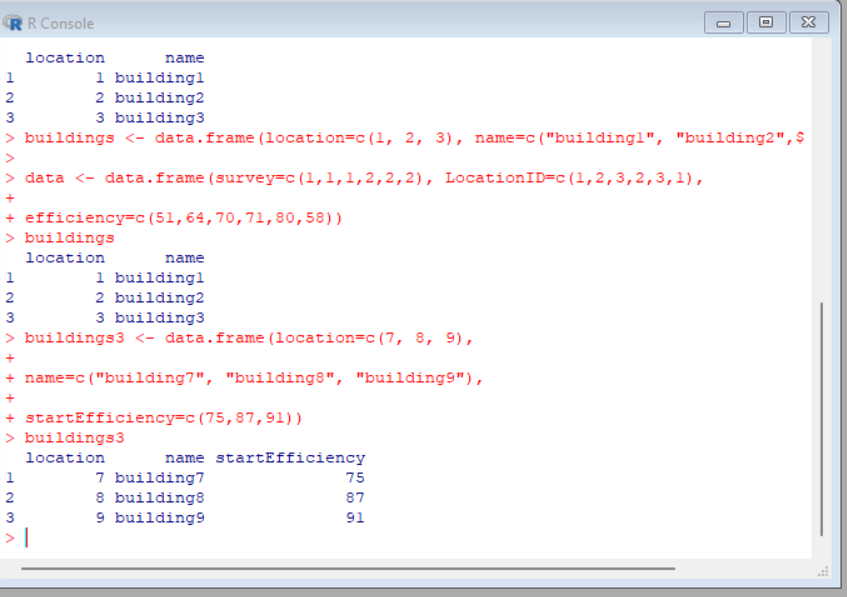
Merge the rows of the following two dataframes:

buildings<- data.frame(location=c(1, 2, 3), name=c("building1",

"building2", "building3"))

buildings2 <- data.frame(location=c(5, 4, 6), name=c("building5", "building4", "building6"))

Also, specify the new dataframe as, “allBuidings”.

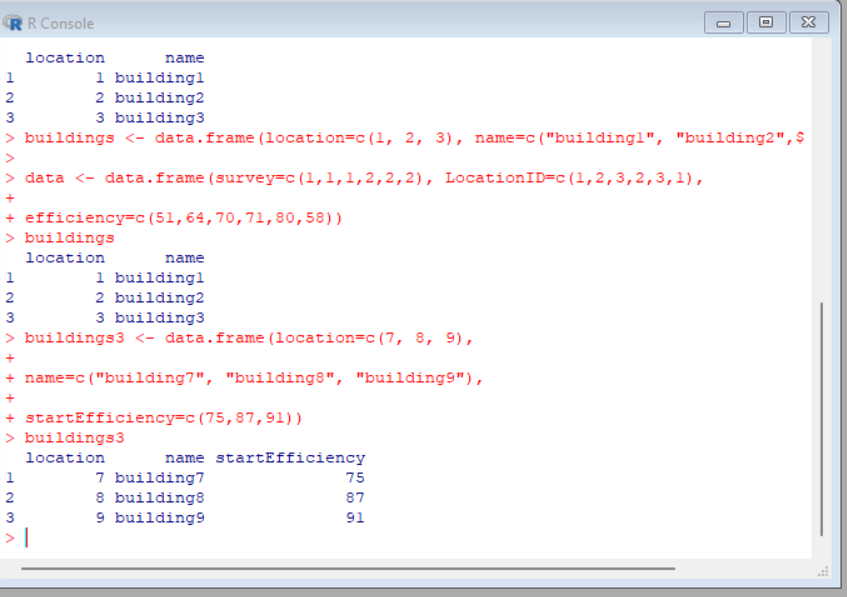


### **Exercise 14**

Create a new dataframe, buildings3, that has variables not found in the previous dataframes.

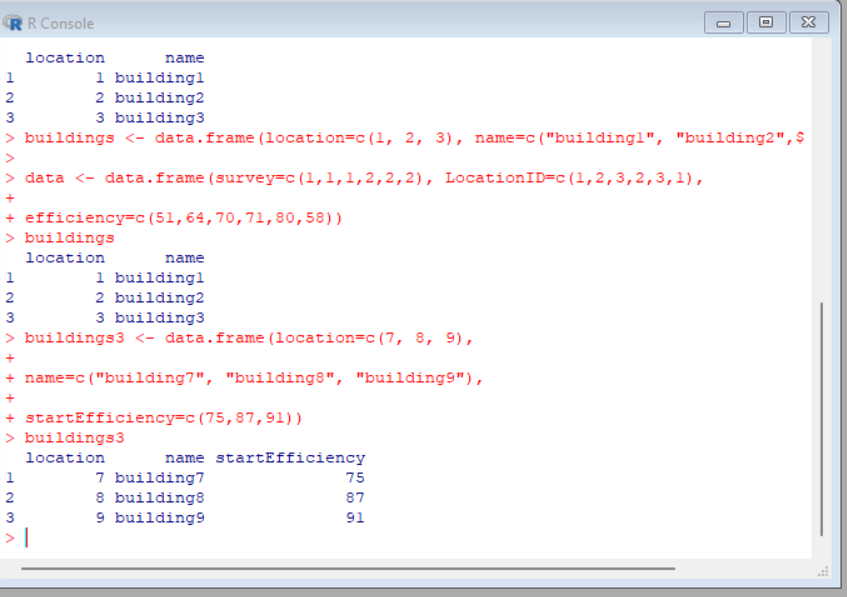
buildings3 <- data.frame(location=c(7, 8, 9), name=c("building7", "building8", "building9"), startEfficiency=c(75,87,91))

Create a new buildings3 without the extra variables.



### **Exercise 15**

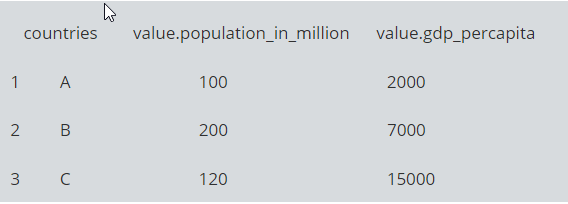
Instead of deleting the extra variables from buildings3. append the buildings, and buildings2 with the new variable in buildings3, **(from Exercise 14).** Set the new data in buildings and buildings2 , (**from Exercise 13)**, to NA.



**RESHAPE FUNCTION IN R**

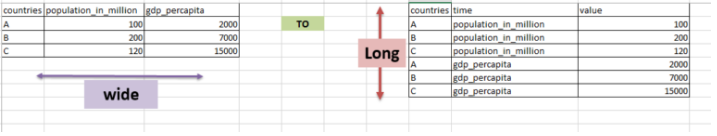
**Exercise: 16**

Construct the following data frame ‘country’.



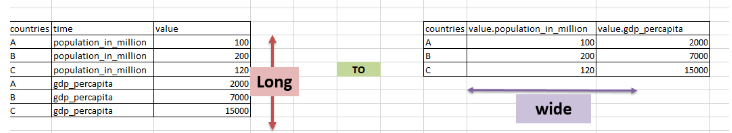
#### **Reshape in R from wide to long:**

Reshape the above data frame from wide to long format in R.



* data frame “country” is passed to reshape function
* idvar is the variable which need to be left unaltered which is “countries”
* varying are the ones that needs to converted from wide to long
* v.names are the values that should be against the times in the resultant [data frame](http://www.datasciencemadesimple.com/data-frame-in-r/).
* new.row.names is used to assign row names to the resultant dataset
* direction is, to which format the data needs to be transformed

1. **Reshape in R from long to wide:**



* data (country\_w\_to\_L) which is in long format,  is passed to reshape function
* idvar is the variable which need to be left unaltered, which is “countries”
* timevar are the variables that needs to converted to wide format
* v.names are the value variable
* direction is, to which format the data needs to be transformed
  1. **MELTING AND CASTING IN R**

**Exercises 17 :**

1. Melt airquality data set and display as a long – format data ?
2. Melt airquality data and specify month and day to be “ID variables” ?
3. Cast the molten airquality data set .
4. Use cast function appropriately and compute the average of Ozone, Solar.R , Wind and temperature per month ?

> x = data.frame(subject = c("John", "Mary"),

+ time = c(1,1),

+ age = c(33,NA),

+ weight = c(90, NA),

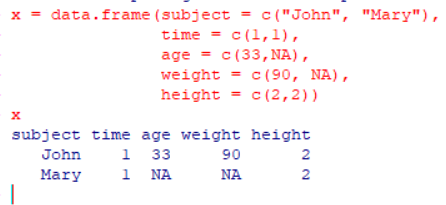
+ height = c(2,2))

> x

subject time age weight height

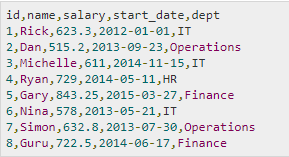
1 John 1 33 90 2

2 Mary 1 NA NA 2

1. 
   * 1. **FILE MANUPULATION IN R**

**Exercise 18**

1. Consider the following data present. Create this file using windows notepad . Save the file as **input.csv** using the save As All files(\*.\*) option in notepad.



1. Use appropriate R commands to read **input.csv** file.

### Analyze the CSV File and compute the following.

### Get the maximum salary

### Get the details of the person with max salary

### Get all the people working in IT department

### Get the persons in IT department whose salary is greater than 600

### Get the people who joined on or after 2014

cannot open file 'input.csv': No such file or directory

> print(data)

survey LocationID efficiency

1 1 1 51

2 1 2 64

3 1 3 70

4 2 2 71

5 2 3 80

6 2 1 58

> data <- read.csv("input.csv")

> print(is.data.frame(data))

[1] TRUE

> print(ncol(data))

[1] 3

> print(nrow(data))

[1] 6

### Get the people who joined on or after 2014 and write the output onto a file called output.csv

