**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**

19021002

SUDHARSANAM.J

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

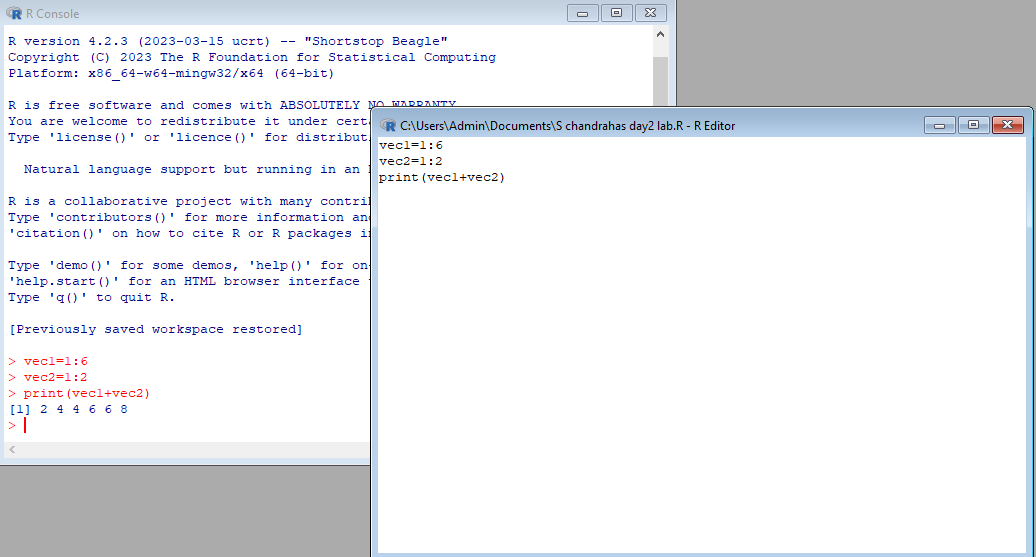
**1. Demonstrate Vector Recycling in R.**

**A. So, the repetition of small length vector as long as completion of operation on long length vector is known as vector recycling.**

**vec1=1:6**

**vec2=1:2**

**print(vec1+vec2)**



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

**A.**

**The apply() function lets us apply a function to the rows or columns of a matrix or data frame.**

**SOURCE CODE:**

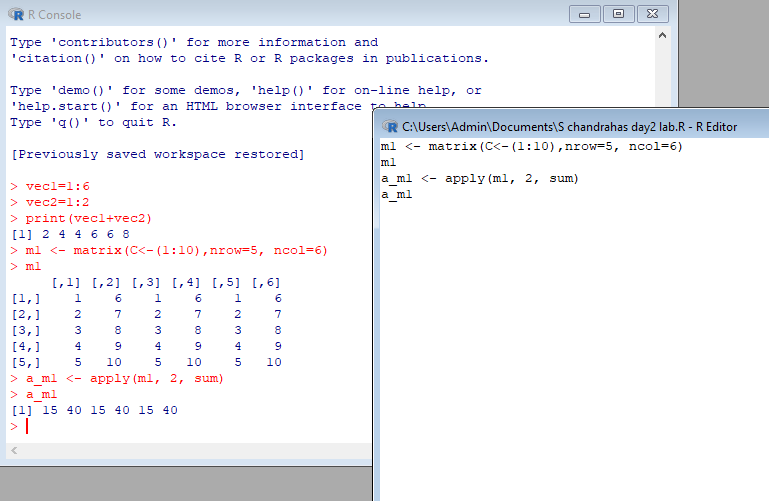
**i]k8**

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

**m1**

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

**a\_m1**



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

**A.**

**The lapply() function helps us in applying functions on list objects and returns a list object of the same length. The lapply() function in the R Language takes a list, vector, or data frame as input and gives output in the form of a list object. Since the lapply() function applies a certain operation to all the elementnts**

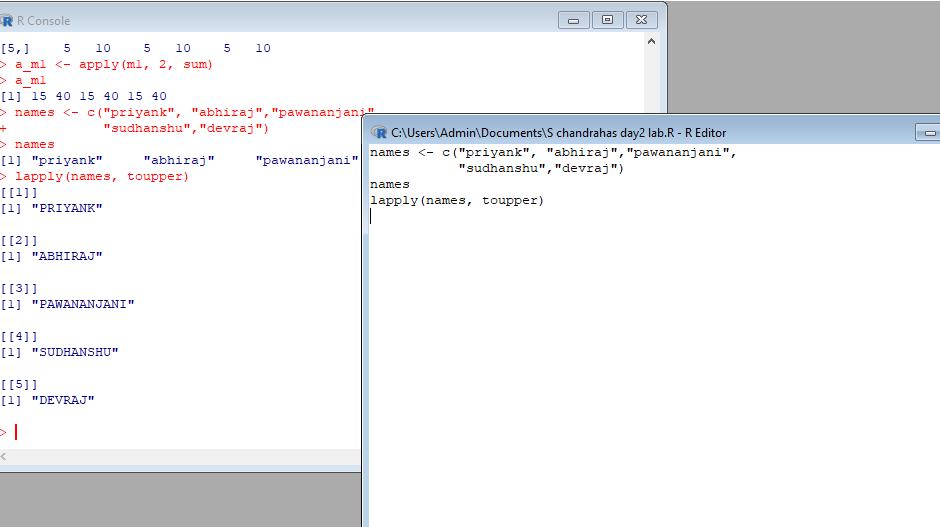
SOURCE CODE

names <- c("priyank", "abhiraj","pawananjani",

"sudhanshu","devraj")

names

lapply(names, toupper)



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

**A.**

The sapply() function helps us in applying functions on a list, vector, or data frame and returns an array or matrix object of the same length. The sapply() function in the R Language takes a list, vector, or data frame as input and gives output in the form of an array or matrix object

SOURCE CODE:

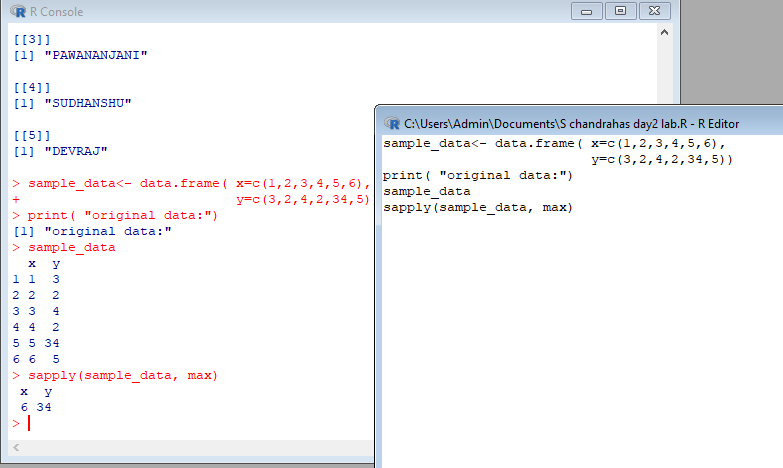
sample\_data<- data.frame( x=c(1,2,3,4,5,6),

y=c(3,2,4,2,34,5))

print( "original data:")

sample\_data

sapply(sample\_data, max)



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

**A)The tapply() helps us to compute statistical measures (mean, median, min, max, etc..) or a self-written function operation for each factor variable in a vector. It helps us to create a subset of a vector and then apply some functions to each of the subsets**

**SOURCE CODE:**

**library(tidyverse)**

**head(diamonds)**

**tapply(diamonds$price, diamonds$cut, mean)**

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

**A.** **SOURCE CODE:**

**myfunction <- function(a, b, c) {**

**a\*b + c**

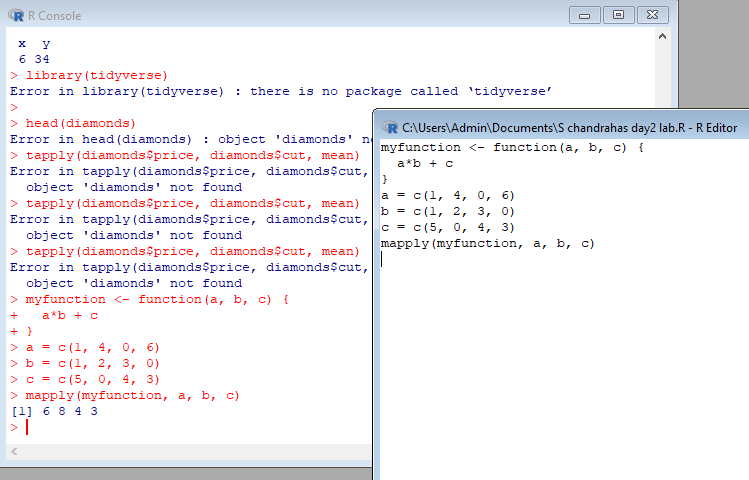
**}**

**a = c(1, 4, 0, 6)**

**b = c(1, 2, 3, 0)**

**c = c(5, 0, 4, 3)**

**mapply(myfunction, a, b, c)**



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

**A.**

SOURCE CODE:

recur\_sum <- function(n) {

if(n <= 1) {

return(n)

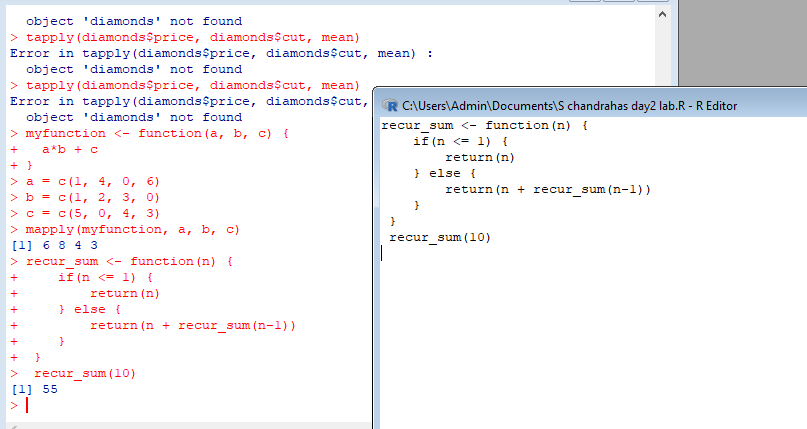
} else {

return(n + recur\_sum(n-1))

}

}

recur\_sum(10)



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

**A.**

SOURCE CODE:

recur\_fibo <- function(n) {

if(n <= 1) {

return(n)

} else {

return(recur\_fibo(n-1) + recur\_fibo(n-2))

}

}

nterms = as.integer(readline(prompt="How many terms? "))

if(nterms <= 0) {

print("Plese enter a positive integer")

} else {

print("Fibonacci sequence:")

for(i in 0:(nterms-1)) {

print(recur\_fibo(i))

}

}

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

A.

SOURCE CODE:

recur\_factorial <- function(n) {

if(n <= 1) {

return(1)

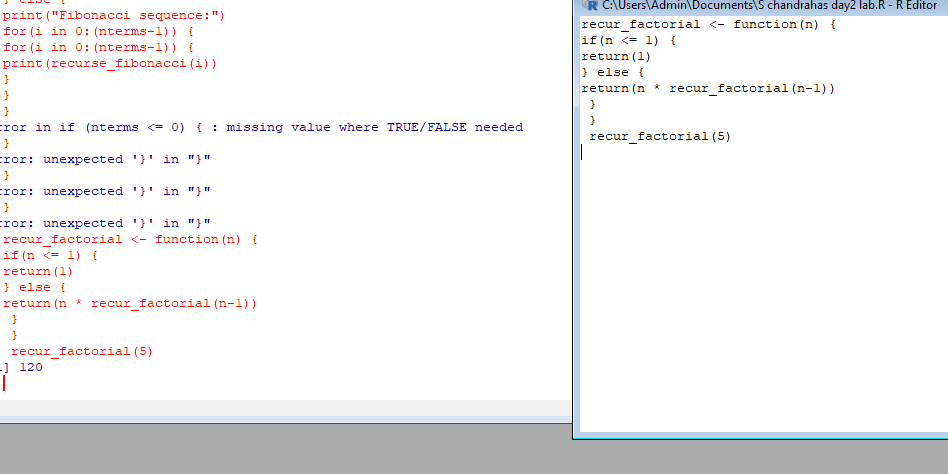
} else {

return(n \* recur\_factorial(n-1))

}

}

recur\_factorial(5)



**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

SOURC CODE

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

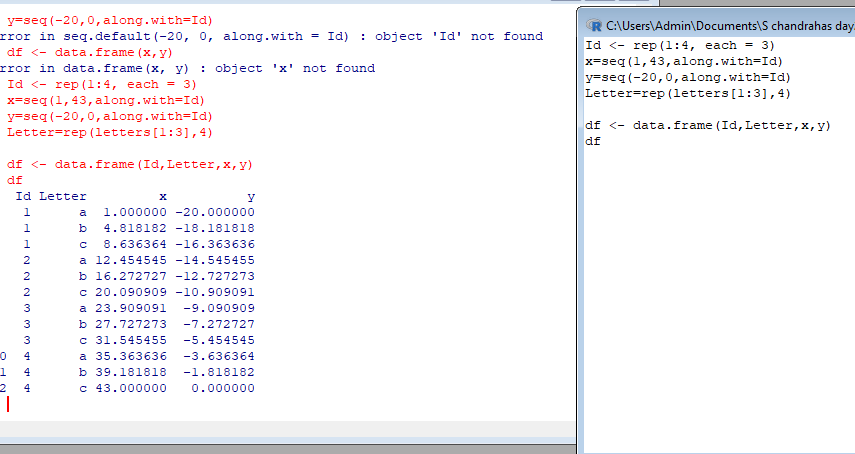
x=seq(1,43,along.with=Id)

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

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

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

df



**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 14
* 2 12
* 3 15
* 4 10

> df2

Id Sex Code

* 1 F a
* 2 M b
* 3 M c
* 4 F d

From df1 and df2 create M:

>M

Id Age Sex Code

* 1 14 F a
* 2 12 M b
* 3 15 M c 4 4 10 F d

SOURCE CODE

id <- c(1,2,3,4)

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

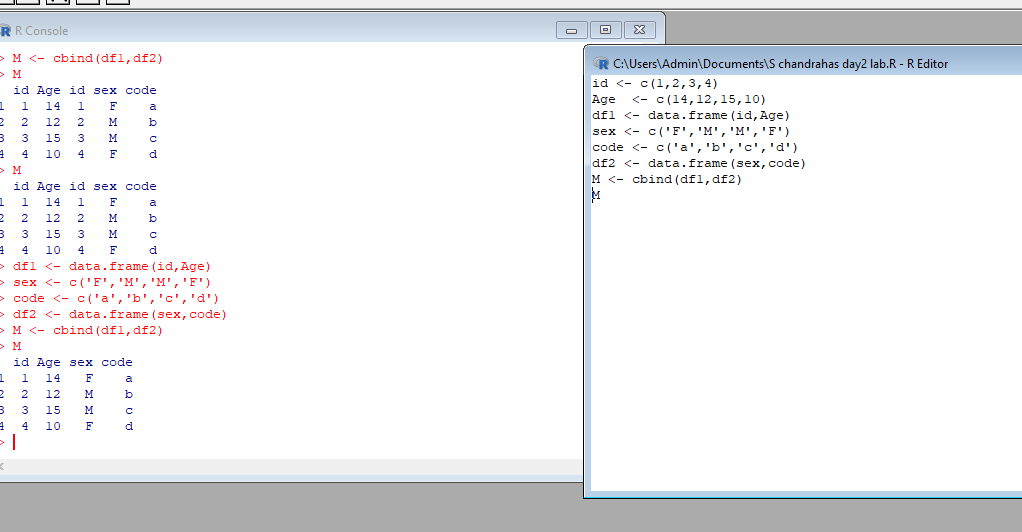
df1 <- data.frame(id,Age)

sex <- c('F','M','M','F')

code <- c('a','b','c','d')

df2 <- data.frame(sex,code)

M <- rbind(df1,df2)



**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 14 F a 99
* 2 12 M b 94
* 3 15 M c 98 4 4 10 F d 100
* Sourc code

id <- c(1,2,3,4)

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

df1 <- data.frame(id,Age)

sex <- c('F','M','M','F')

code <- c('a','b','c','d')

df2 <- data.frame(sex,code)

M <- cbind(df1,df2)

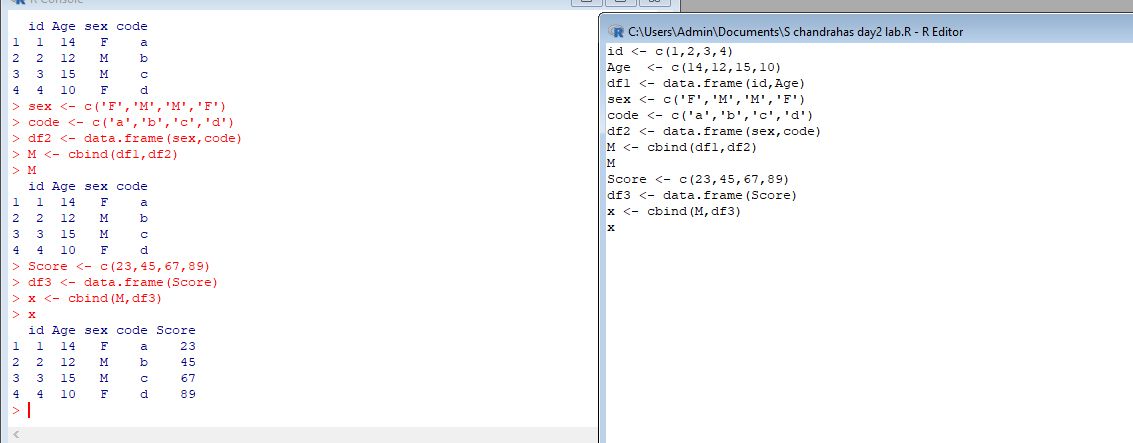
M

Score <- c(23,45,67,89)

df3 <- data.frame(Score)

x <- cbind(M,df3)

x

* 

**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 Id
* 2 Id
* 3 Id
* 4 Id
* 14 Age
* 12 Age
* 15 Age
* 10 Age
* 99 score
* 94 score
* 98 score
* 100 score

SOURCE CODE

id <- c(1,2,3,4)

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

df1 <- data.frame(id,Age)

sex <- c('F','M','M','F')

code <- c('a','b','c','d')

df2 <- data.frame(sex,code)

M <- cbind(df1,df2)

M

Score <- c(23,45,67,89)

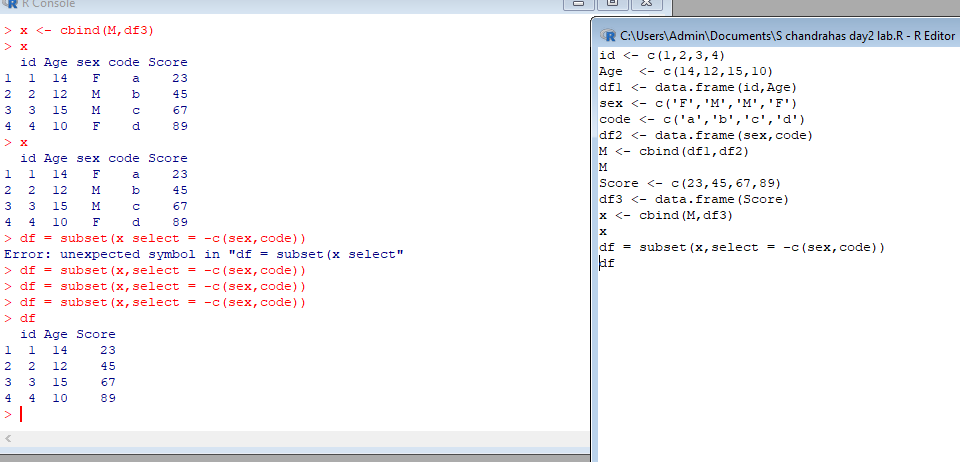
df3 <- data.frame(Score)

x <- cbind(M,df3)

x

df = subset(x,select = -c(sex,code))

df

* 

**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

SOURCE CODE

A<-data.frame()

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

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

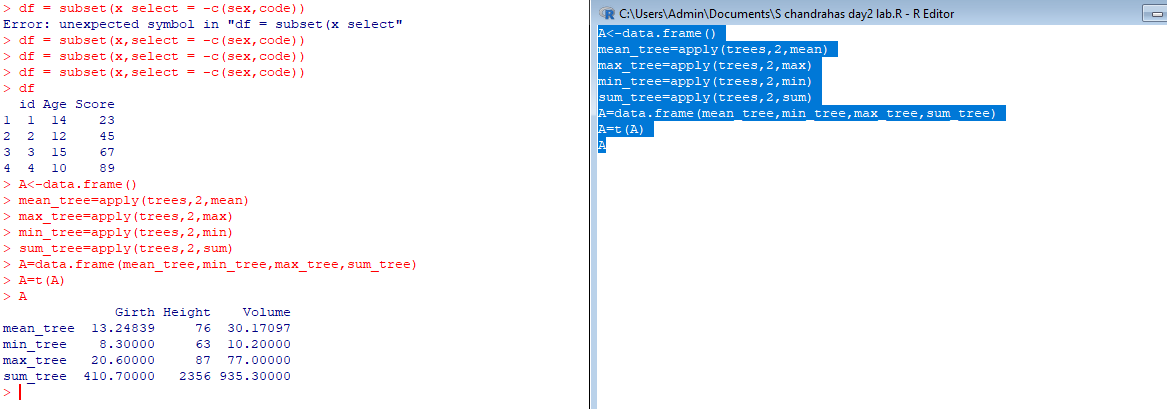
min\_tree=apply(trees,2,min)

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

A=data.frame(mean\_tree,min\_tree,max\_tree,sum\_tree)

A=t(A)

A



**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

**Exercise 8**

Create an empty data frame with column types:

>df

df <- data.frame()

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 0
* 2 3
* 3 2
* 1 0
* 4 5
* 5 9
* 2 3

* look at duplicated elements using a provided R function.
* keep only the unique lines on XY using a provided R function.
* Source code

X <- c(1,2,3,1,4,5,2);

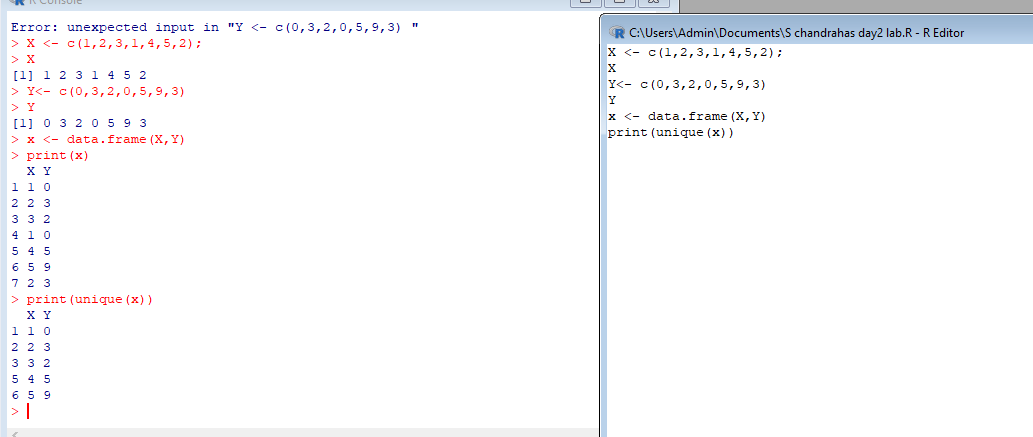
X

Y<- c(0,3,2,0,5,9,3)

Y

x <- data.frame(X,Y)

print(unique(x))



**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.
2. 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

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”.

A> -----------> buildingStats <- merge(buildings, data, by="location")

**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.

A>----buildingStats <- merge(buildings, data, by.x="location", by.y="LocationID")

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.

A>

buildingStats <- merge(buildings, data)

Since both dataframes have a common variable called "location", the merge function will automatically join the frames by this 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”.

A) buildingStats <- merge(buildings, data, by="location", all=TRUE)

**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.**

**A)**

**buildingStats <- merge(buildings, data, by="location", all.x=TRUE)**

**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.

A)

buildingStats <- merge(buildings, data, by="location", all.y=TRUE)

**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”?

A) buildingStats <- merge(buildings, data, by=NULL)

**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”.

A) allBuildings <- rbind(buildings, buildings2)

The resulting dataframe "allBuildings" will have 6 rows, with locations 1, 2, and 3 from "buildings", and locations 5, 4, and 6 from "buildings2".

**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.

A)

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

**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.

A)buildings$extraVariable <- NA

buildings2$extraVariable <- NA

buildings <- rbind(buildings, buildings3)

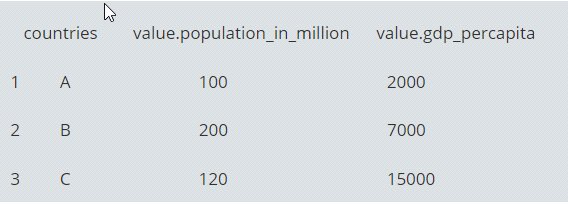
buildings2 <- rbind(buildings2, buildings3)

The variable "extraVariable" is added to both "buildings" and "buildings2" with NA values. The rows from "buildings3" are appended to the bottom of both dataframes.

**RESHAPE FUNCTION IN R**

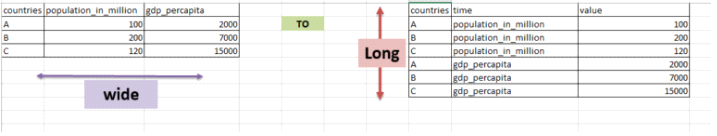
**Exercise: 16**

Construct the following data frame ‘country’.

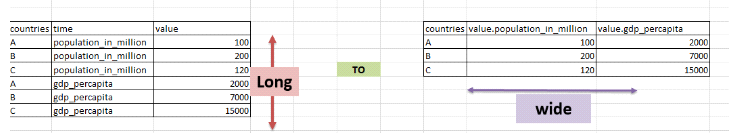


1. **Reshape in R from wide to long:**

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
* **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 ?

A.

library(tidyr)

data(airquality)

airquality\_long <- airquality %>%

pivot\_longer(cols = c("Ozone", "Solar.R", "Wind", "Temp"),

names\_to = "variable", values\_to = "value")

head(airquality\_long)

1. Melt airquality data and specify month and day to be “ID variables” ?

A.

airquality\_long <- airquality %>%

pivot\_longer(cols = c("Ozone", "Solar.R", "Wind", "Temp"),

names\_to = "variable", values\_to = "value") %>%

separate(Date, c("Month", "Day"), sep = "\\s+")

head(airquality\_long)

1. Cast the molten airquality data set .

A.

airquality\_wide <- airquality\_long %>%

pivot\_wider(names\_from = variable, values\_from = value)

head(airquality\_wide)

1. Use cast function appropriately and compute the average of Ozone, Solar.R , Wind and temperature per month ?

A.

library(reshape2)

data(airquality)

airquality\_avg <- cast(melt(airquality, id.vars = c("Month", "Day")),

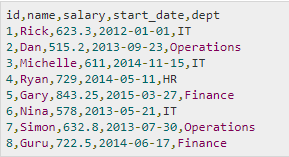
Month ~ variable, mean)

head(airquality\_avg)

1. **FILE MANUPULATION IN R**

**Exercise 18**

* 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.



A.

Name, Age, Gender, City

John, 28, Male, New York

Emily, 24, Female, Los Angeles

Michael, 35, Male, Chicago

Sarah, 31, Female, Houston

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

A.

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

1. Analyze the CSV File and compute the following.
2. Get the maximum salary

A.

max\_salary <- max(data$salary)

1. Get the details of the person with max salary

A.

person\_with\_max\_salary <- subset(data, salary == max\_salary)

1. Get all the people working in IT department

**A.**

IT\_employees <- subset(data, department == "IT")

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

**A.** high\_paid\_IT\_employees <- subset(IT\_employees, salary > 600)

1. Get the people who joined on or after 2014

**A.**

recent\_employees <- subset(data, year\_joined >= 2014)

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

A.

recent\_employees <- subset(data, year\_joined >= 2014)

write.csv(recent\_employees, file = "output.csv", row.names = FALSE)