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:192125001

Name:R. Aishwarya

IMPLEMENTATION OF VECTOR RECYCLING, APPLY FAMILY & RECURSION

1. Demonstrate Vector Recycling in R.

coding: x <-

c(1, 2, 3) y <-

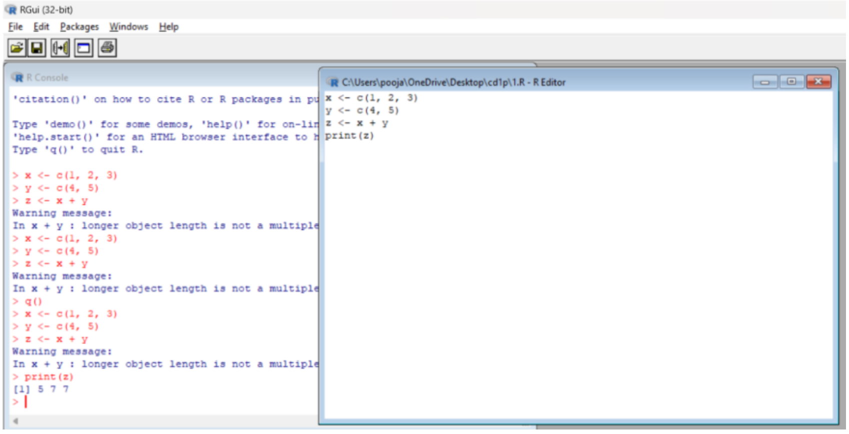
c(4, 5)

z <- x + y

print(z)

output:

[1] 5 7 7



RGui (32-bit)
File Edit Packages Windows Help
R Console
C:\Users\pooja\OneDrive\Desktop\cd1p\1.R - R Editor

```
'citation()' on how to cite R or R packages in pu

Type 'demo()' for some demos, 'help()' for on-lin
'help.start()' for an HTML browser interface to h
Type 'q()' to quit R.

> x <- c(1, 2, 3)
> y <- c(4, 5)
> z <- x + y
Warning message:
In x + y : longer object length is not a multiple
> x <- c(1, 2, 3)
> y <- c(4, 5)
> z <- x + y
Warning message:
In x + y : longer object length is not a multiple
> q()
> x <- c(1, 2, 3)
> y <- c(4, 5)
> z <- x + y
Warning message:
In x + y : longer object length is not a multiple
> print(z)
[1] 5 7 7
>
```

```
x <- c(1, 2, 3)
y <- c(4, 5)
z <- x + y
print(z)
```

2. Demonstrate the usage of apply function in

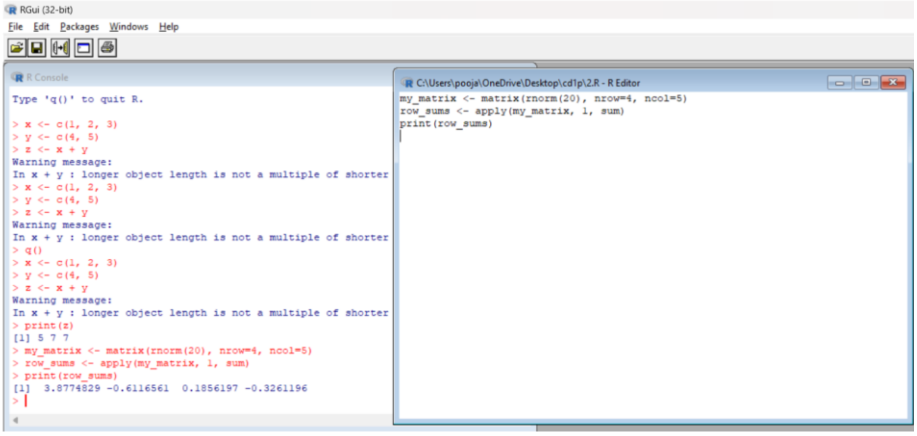
R coding:

my_matrix <- matrix(rnorm(20), nrow=4, ncol=5)

row_sums <- apply(my_matrix, 1, sum)

print(row_sums) output:

3.8774829 -0.6116561 0.1856197 -0.3261196



RGui (32-bit)
File Edit Packages Windows Help
R Console
C:\Users\pooja\OneDrive\Desktop\cd1p\2.R - R Editor

```
Type 'q()' to quit R.

> x <- c(1, 2, 3)
> y <- c(4, 5)
> z <- x + y
Warning message:
In x + y : longer object length is not a multiple of shorter
> x <- c(1, 2, 3)
> y <- c(4, 5)
> z <- x + y
Warning message:
In x + y : longer object length is not a multiple of shorter
> q()
> x <- c(1, 2, 3)
> y <- c(4, 5)
> z <- x + y
Warning message:
In x + y : longer object length is not a multiple of shorter
> print(z)
[1] 5 7 7
> my_matrix <- matrix(rnorm(20), nrow=4, ncol=5)
> row_sums <- apply(my_matrix, 1, sum)
> print(row_sums)
[1]  3.8774829 -0.6116561  0.1856197 -0.3261196
> 
```

```
my_matrix <- matrix(rnorm(20), nrow=4, ncol=5)
row_sums <- apply(my_matrix, 1, sum)
print(row_sums)
```

3.   Demonstrate the usage of lapply function in

R coding:

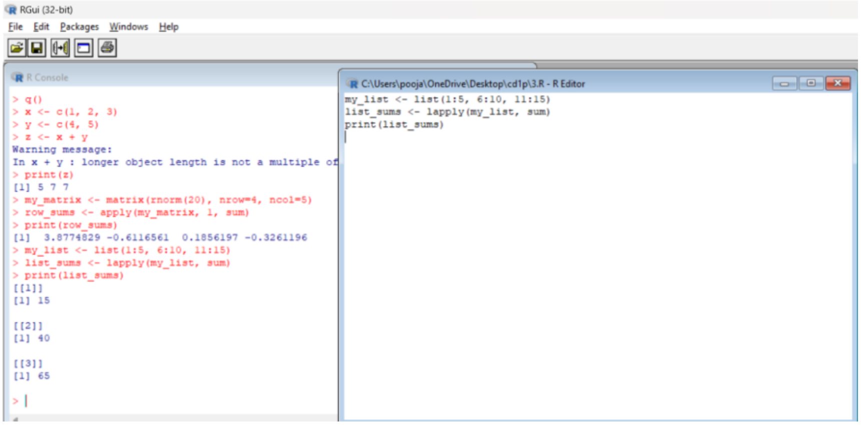
my_list <- list(1:5, 6:10, 11:15)

list_sums <- lapply(my_list, sum)

print(list_sums) output: [1] 15

[1] 40

[1] 65



RGui (32-bit)
File Edit Packages Windows Help
R Console
C:\Users\pooja\OneDrive\Desktop\cd1p\3.R - R Editor

```
> q()
> x <- c(1, 2, 3)
> y <- c(4, 5)
> z <- x + y
Warning message:
In x + y : longer object length is not a multiple of
> print(z)
[1] 5 7 7
> my_matrix <- matrix(rnorm(20), nrow=4, ncol=5)
> row_sums <- apply(my_matrix, 1, sum)
> print(row_sums)
[1]  3.8774829 -0.6116561  0.1856197 -0.3261196
> my_list <- list(1:5, 6:10, 11:15)
> list_sums <- lapply(my_list, sum)
> print(list_sums)
[[1]]
[1] 15

[[2]]
[1] 40

[[3]]
[1] 65

> 
```

```
my_list <- list(1:5, 6:10, 11:15)
list_sums <- lapply(my_list, sum)
print(list_sums)
```

4. Demonstrate the usage of sapply function in R

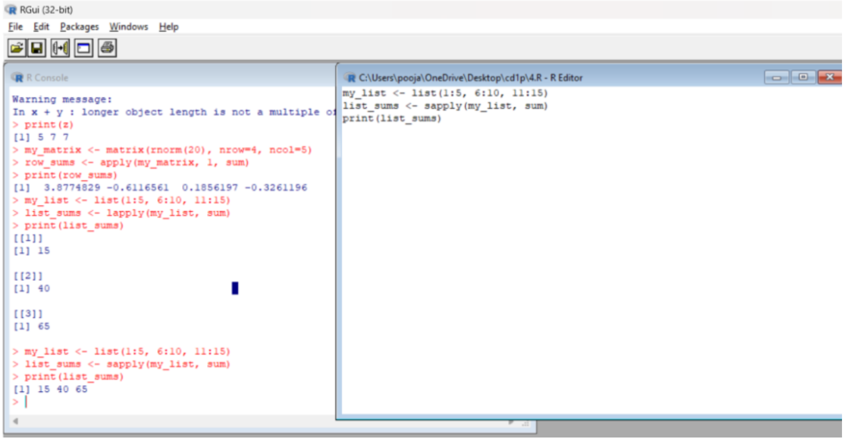
coding:

my_list <- list(1:5, 6:10, 11:15)

list_sums <- sapply(my_list, sum)

print(list_sums) output:

15 40 65


RGui (32-bit)
File Edit Packages Windows Help
R Console
C:\Users\pooja\OneDrive\Desktop\cd1p\4.R - R Editor

```
Warning message:
In x + y : longer object length is not a multiple of
> print(z)
[1] 5 7 7
> my_matrix <- matrix(rnorm(20), nrow=4, ncol=5)
> row_sums <- apply(my_matrix, 1, sum)
> print(row_sums)
[1]  3.8774829 -0.6116561  0.1856197 -0.3261196
> my_list <- list(1:5, 6:10, 11:15)
> list_sums <- lapply(my_list, sum)
> print(list_sums)
[[1]]
[1] 15

[[2]]
[1] 40

[[3]]
[1] 65

> my_list <- list(1:5, 6:10, 11:15)
> list_sums <- sapply(my_list, sum)
> print(list_sums)
[1] 15 40 65
>
```

```
my_list <- list(1:5, 6:10, 11:15)
list_sums <- sapply(my_list, sum)
print(list_sums)
```

5. Demonstrate the usage of tapply function in

R coding:

exam_scores <- data.frame(student = c("Alice", "Bob", "Charlie",

"Alice", "Bob", "Charlie"), score = c(80, 85, 90, 75, 70, 85)

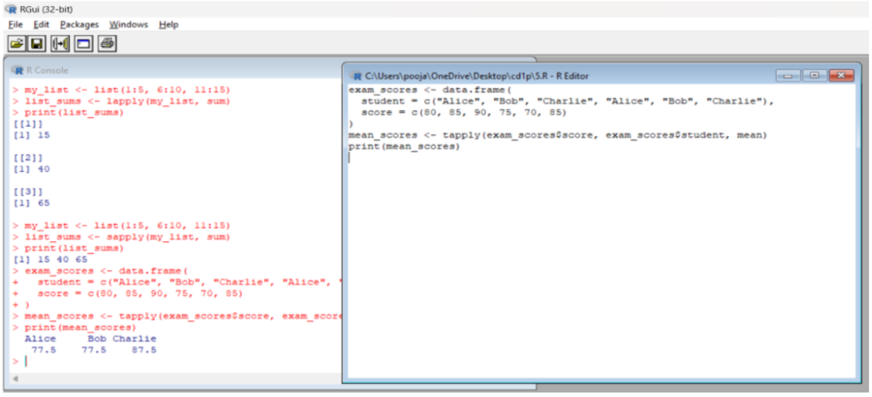
)

mean_scores <- tapply(exam_scores$score, exam_scores$student, mean)

print(mean_scores) output:

Alice Bob Charlie

77.5 77.5 87.5


RGui (32-bit)
File Edit Packages Windows Help
R Console
C:\Users\pooja\OneDrive\Desktop\cd1p\5.R - R Editor

```
> my_list <- list(1:5, 6:10, 11:15)
> list_sums <- lapply(my_list, sum)
> print(list_sums)
[[1]]
[1] 15

[[2]]
[1] 40

[[3]]
[1] 65

> my_list <- list(1:5, 6:10, 11:15)
> list_sums <- sapply(my_list, sum)
> print(list_sums)
[1] 15 40 65
> exam_scores <- data.frame(
+   student = c("Alice", "Bob", "Charlie", "Alice", "
+   score = c(80, 85, 90, 75, 70, 85)
+ )
> mean_scores <- tapply(exam_scores$score, exam_score
> print(mean_scores)
  Alice     Bob Charlie 
   77.5    77.5    87.5 
>
```

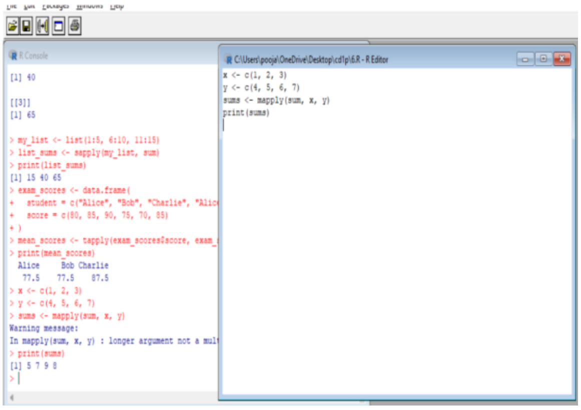
```
exam_scores <- data.frame(
  student = c("Alice", "Bob", "Charlie", "Alice", "Bob", "Charlie"),
  score = c(80, 85, 90, 75, 70, 85)
)
mean_scores <- tapply(exam_scores$score, exam_scores$student, mean)
print(mean_scores)
```

6. Demonstrate the usage of mapply function

in R coding: x <- c(1, 2, 3) y <- c(4, 5, 6, 7) sums <-

mapply(sum, x, y) print(sums) output:

5 7 9 8



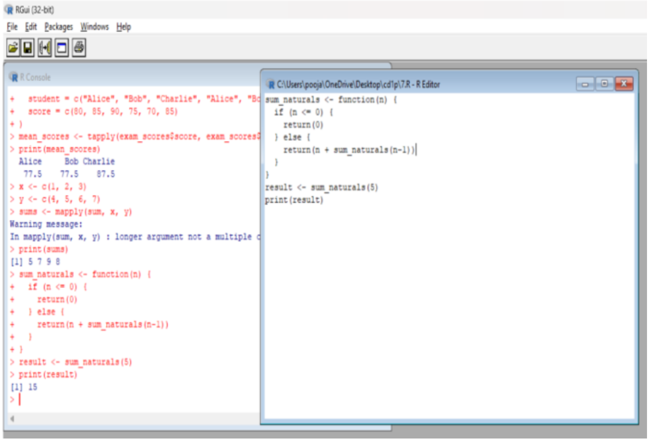
```
[1] 40

[[3]]
[1] 65

> my_list <- list(1:5, 6:10, 11:15)
> list_sums <- sapply(my_list, sum)
> print(list_sums)
[1] 15 40 65
> exam_scores <- data.frame(
+   student = c("Alice", "Bob", "Charlie", "Alic
+   score = c(80, 85, 90, 75, 70, 85)
+ )
> mean_scores <- tapply(exam_scores$score, exam_
> print(mean_scores)
  Alice     Bob Charlie
   77.5    77.5    87.5
> x <- c(1, 2, 3)
> y <- c(4, 5, 6, 7)
> sums <- mapply(sum, x, y)
Warning message:
In mapply(sum, x, y) : longer argument not a mul
> print(sums)
[1] 5 7 9 8
> 
```

```
x <- c(1, 2, 3)
y <- c(4, 5, 6, 7)
sums <- mapply(sum, x, y)
print(sums)
```

7. Sum of Natural Numbers using Recursion

coding:

```
sum_naturals <- function(n) {
  if (n <= 0) {
    return(0)
  } else { return(n +
    sum_naturals(n-1))
  }
} result <-
sum_naturals(5)
print(result) output:
15
```



RGui (32-bit)
File Edit Packages Windows Help
R Console
C:\Users\pooja\OneDrive\Desktop\cd1p\7.R - R Editor

```
+   student = c("Alice", "Bob", "Charlie", "Alice", "Bo
+   score = c(80, 85, 90, 75, 70, 85)
+ )
> mean_scores <- tapply(exam_scores$score, exam_scores$
> print(mean_scores)
  Alice     Bob Charlie 
   77.5    77.5    87.5 
> x <- c(1, 2, 3)
> y <- c(4, 5, 6, 7)
> sums <- mapply(sum, x, y)
Warning message:
In mapply(sum, x, y) : longer argument not a multiple c
> print(sums)
[1] 5 7 9 8
> sum_naturals <- function(n) {
+   if (n <= 0) {
+     return(0)
+   } else {
+     return(n + sum_naturals(n-1))
+   }
+ }
> result <- sum_naturals(5)
> print(result)
[1] 15
> 
```

```
sum_naturals <- function(n) {
  if (n <= 0) {
    return(0)
  } else {
    return(n + sum_naturals(n-1))
  }
}
result <- sum_naturals(5)
print(result)
```

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

output:

```
fibonacci <- function(n) {
  if (n <= 1) {
    return(n)
  } else { return(fibonacci(n-1) +
    fibonacci(n-2))
  }
} for (i in 0:9)
{ result <-
fibonacci(i)
print(result)
}
```

output:

```
0
1
1
2
```

3

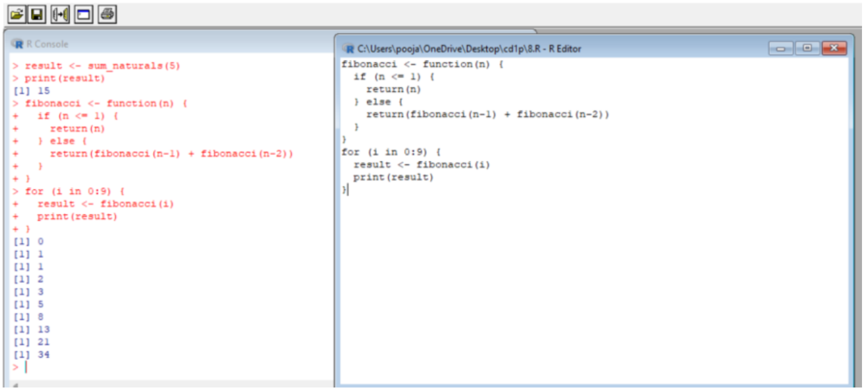
5

8

13

21

34



```
> result <- sum_naturals(5)
> print(result)
[1] 15
> fibonacci <- function(n) {
+   if (n <= 1) {
+     return(n)
+   } else {
+     return(fibonacci(n-1) + fibonacci(n-2))
+   }
+ }
> for (i in 0:9) {
+   result <- fibonacci(i)
+   print(result)
+ }
[1] 0
[1] 1
[1] 1
[1] 2
[1] 3
[1] 5
[1] 8
[1] 13
[1] 21
[1] 34
> 
```

```
fibonacci <- function(n) {
  if (n <= 1) {
    return(n)
  } else {
    return(fibonacci(n-1) + fibonacci(n-2))
  }
}
for (i in 0:9) {
  result <- fibonacci(i)
  print(result)
}
```

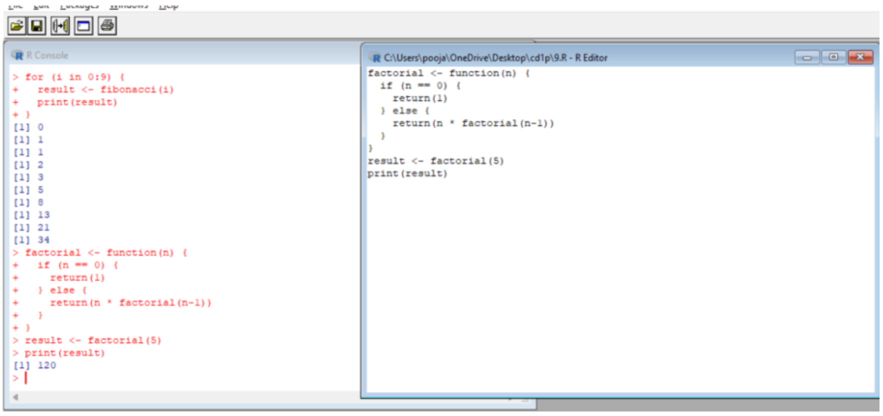
9. Write a program to find factorial of a number in R using recursion.

coding:

```
factorial <- function(n) {

  if (n == 0) {

    return(1)
  } else { return(n *

    factorial(n-1))

  } } result <-

factorial(5)

print(result)
```

output:

120



```
> for (i in 0:9) {
+   result <- fibonacci(i)
+   print(result)
+ }
[1] 0
[1] 1
[1] 1
[1] 2
[1] 3
[1] 5
[1] 8
[1] 13
[1] 21
[1] 34
> factorial <- function(n) {
+   if (n == 0) {
+     return(1)
+   } else {
+     return(n * factorial(n-1))
+   }
+ }
> result <- factorial(5)
> print(result)
[1] 120
>
```

```
factorial <- function(n) {
  if (n == 0) {
    return(1)
  } else {
    return(n * factorial(n-1))
  }
}
result <- factorial(5)
print(result)
```

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
```

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
```

Exercise 4
Create a data frame df3:

```
> 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
```

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
```

Exercise 6

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:

```
>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
```

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
IntsLogicals Doubles Characters
```

(or 0-length row.names)

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
```

1) look at duplicated elements using a provided R function.
2) keep only the unique lines on XY using a provided R function.

Exercise 10

Use the (built-in) dataset Titanic.

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

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.

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))

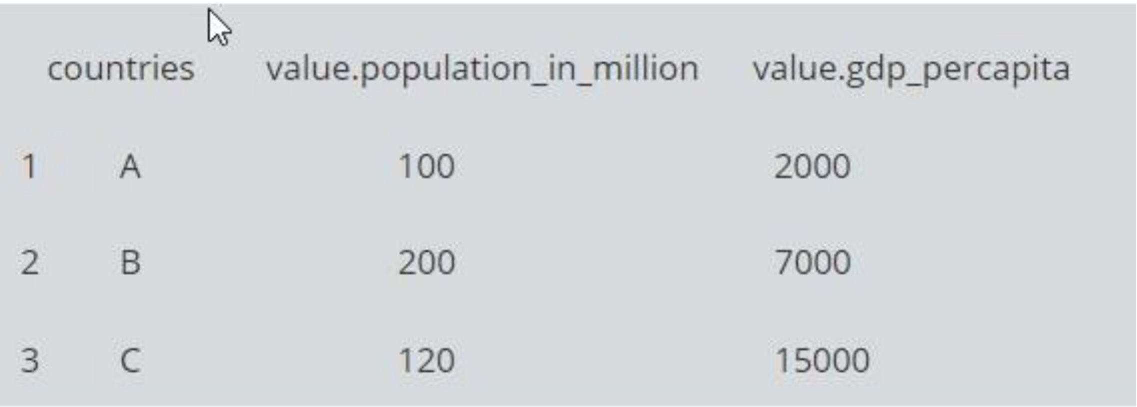
Create a new buildings3 without the extra variables.

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

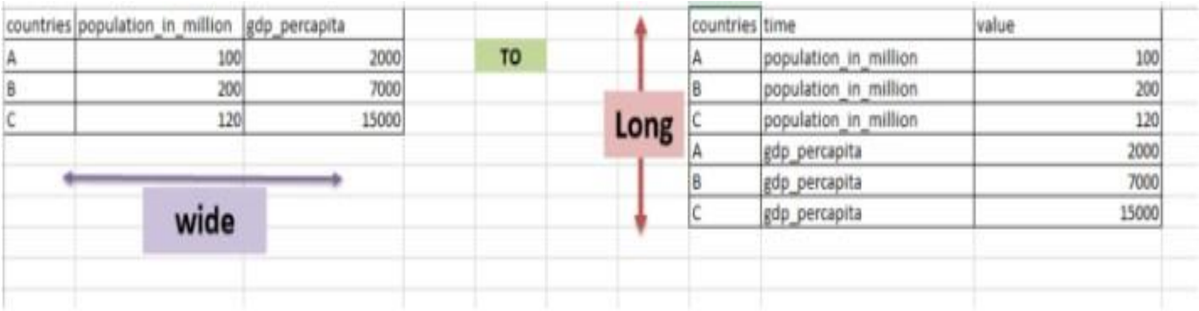
Construct the following data frame ‘country’.



	countries	value.population_in_million	value.gdp_percapita
1	A	100	2000
2	B	200	7000
3	C	120	15000

a) Reshape in R from wide to long:

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

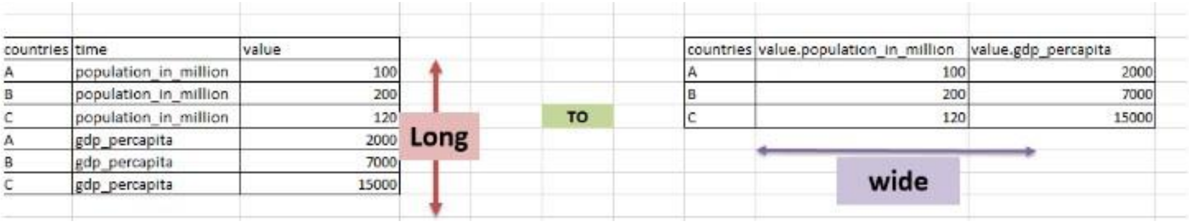


countries	population_in_million	gdp_percapita
A	100	2000
B	200	7000
C	120	15000

countries	time	value
A	population_in_million	100
B	population_in_million	200
C	population_in_million	120
A	gdp_percapita	2000
B	gdp_percapita	7000
C	gdp_percapita	15000

- 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.
- new.row.names is used to assign row names to the resultant dataset • direction is, to which format the data needs to be transformed

b) Reshape in R from long to wide:



countries	time	value
A	population_in_million	100
B	population_in_million	200
C	population_in_million	120
A	gdp_percapita	2000
B	gdp_percapita	7000
C	gdp_percapita	15000

countries	value.population_in_million	value.gdp_percapita
A	100	2000
B	200	7000
C	120	15000

- 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

7. MELTING AND CASTING IN R

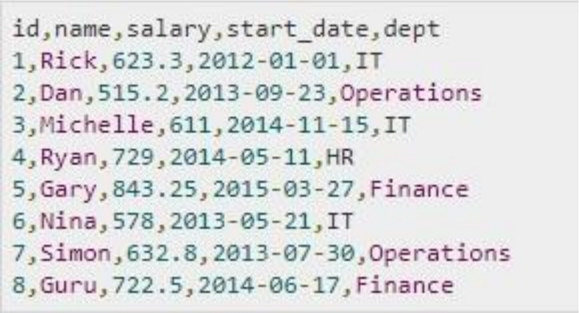
Exercises 17 :

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 ?

8 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.



```
id,name,salary,start_date,dept
1,Rick,623.3,2012-01-01,IT
2,Dan,515.2,2013-09-23,Operations
3,Michelle,611,2014-11-15,IT
4,Ryan,729,2014-05-11,HR
5,Gary,843.25,2015-03-27,Finance
6,Nina,578,2013-05-21,IT
7,Simon,632.8,2013-07-30,Operations
8,Guru,722.5,2014-06-17,Finance
```

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

3. Analyze the CSV File and compute the following.

a. Get the maximum salary

b. Get the details of the person with max salary

c. Get all the people working in IT department

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

e. Get the people who joined on or after 2014

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