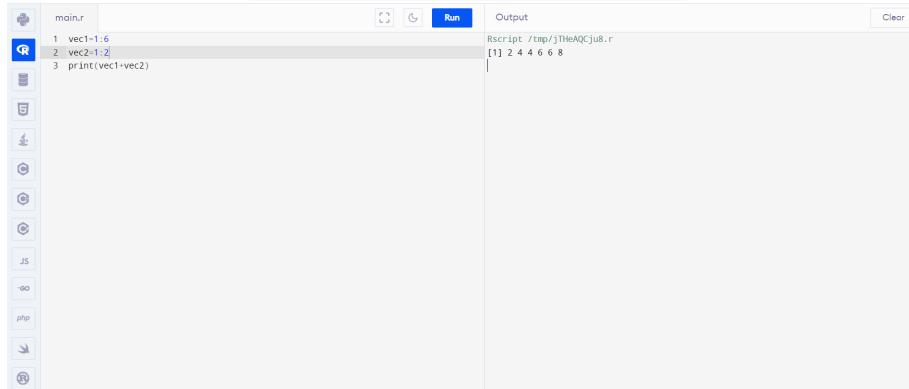


SAVEETHA SCHOOL OF ENGINEERING
SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES
ITA 0443 - STATISTICS WITH R PROGRAMMING FOR REAL TIME PROBLEM
DAY 2 – LAB EXERCISES

Reg No: 191911211

Name: K.Abhishek

1. IMPLEMENTATION OF VECTOR RECYCLING, APPLY FAMILY & RECURSION



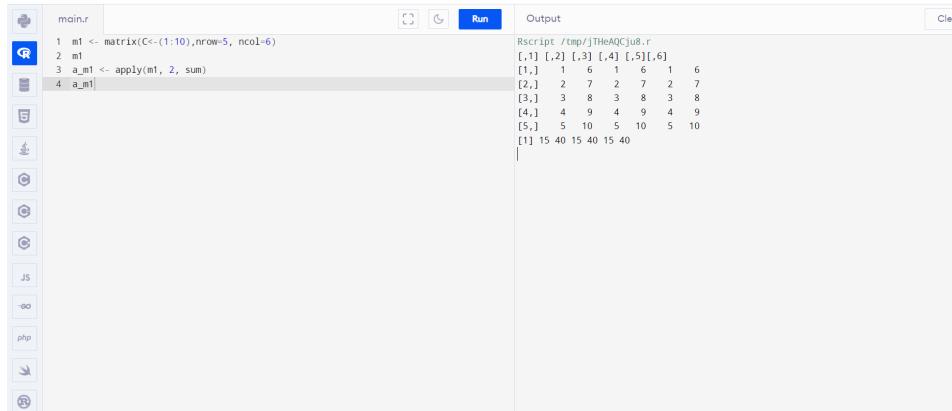
The screenshot shows the RStudio interface. On the left is a file browser with icons for various file types. The main area has two tabs: 'main.r' and 'Output'. The 'main.r' tab contains the following R code:

```
1 vect<-1:6
2 vec2<-1:2
3 print(vect+vec2)
```

The 'Run' button is highlighted in blue. The 'Output' tab shows the results of running the script:

```
Rscript /tmp/jTHeAQcju8.r
[1] 2 4 4 6 6 8
```

2. Demonstrate the usage of apply function in R



The screenshot shows the RStudio interface. On the left is a file browser with icons for various file types. The main area has two tabs: 'main.r' and 'Output'. The 'main.r' tab contains the following R code:

```
1 m1 <- matrix(1:(1:10),nrow=5, ncol=6)
2 m1
3 a_m1 <- apply(m1, 2, sum)
4 a_m1
```

The 'Run' button is highlighted in blue. The 'Output' tab shows the results of running the script:

```
Rscript /tmp/jTHeAQcju8.r
[,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    6    1    6    1    6
[2,]    2    7    2    7    2    7
[3,]    3    8    3    8    3    8
[4,]    4    9    4    9    4    9
[5,]    5   10    5   10    5   10
[1] 15 40 15 40 15 40
```

3. Demonstrate the usage of l apply function in R

The screenshot shows the RStudio interface. On the left is a file browser with a tree view. The main area has a tab labeled "main.r". The code in the editor is:

```
1 movies <- c("SPYDERMAN", "BATMAN", "VERTIGO", "CHINATOWN")
2 movies_lower <- lapply(movies, tolower)
3 str(movies_lower)
```

The "Run" button is highlighted in blue. To the right is the "Output" pane, which displays the results of the R script:

```
Rscript /tmp/jTHeAQcju8.r
List of 4
$ :chr "spyderman"
$ :chr "batman"
$ :chr "vertigo"
$ :chr "chinatown"
```

4. Demonstrate the usage of s apply function in R

The screenshot shows the RStudio interface. On the left is a file browser with a tree view. The main area has a tab labeled "main.r". The code in the editor is:

```
1 dt <- cars
2 lmn_cars <- lapply(dt, min)
3 smn_cars <- sapply(dt, min)
4 lmn_cars
```

The "Run" button is highlighted in blue. To the right is the "Output" pane, which displays the results of the R script:

```
Rscript /tmp/jTHeAQcju8.r
$speed
[1] 4
$dist
[1] 2
```

5. Demonstrate the usage of t apply function in R

The screenshot shows the RStudio interface. On the left is a file browser with a tree view. The main area has a tab labeled "main.r". The code in the editor is:

```
1 data(iris)
2 tapply(iris$Sepal.Width, iris$Species, median)
```

The "Run" button is highlighted in blue. To the right is the "Output" pane, which displays the results of the R script:

```
Rscript /tmp/jTHeAQcju8.r
setosa versicolor virginica
      3.4          2.8          3.0
```

6. Demonstrate the usage of m apply function in R

The screenshot shows a Jupyter Notebook interface. On the left, there is a file browser sidebar with icons for various file types. The main area has a tab labeled "main.r". Inside the "main.r" cell, the following R code is written:

```
1 vec1 <- c(1, 2, 3, 4)
2 vec2 <- c(2, 4, 6, 8)
3 vec3 <- c(3, 6, 9, 12)
4 mapply(function(val1, val2, val3) val1*val2*val3, vec1, vec2, vec3)
5
6
```

At the top right, there are buttons for "Run" and "Output". The "Output" section displays the results of the R script:

```
Rscript /tmp/jTheAQcju8.r
[1] 6 48 162 384
```

7. Sum of Natural Numbers using Recursion

The screenshot shows a Jupyter Notebook interface. On the left, there is a file browser sidebar with icons for various file types. The main area has a tab labeled "main.r". Inside the "main.r" cell, the following R code is written:

```
1 sum<-function(n){
2   if (n<=1){
3     return(n)
4   }else{
5     return(n+sum(n-1))
6   }
7 }
8 sum(7)
```

At the top right, there are buttons for "Run" and "Output". The "Output" section displays the results of the R script:

```
Rscript /tmp/N2hv1rq8rf.r
[1] 28
```

A progress bar at the bottom indicates "Establishing secure connection...".

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

The screenshot shows a code editor interface with a toolbar at the top. The file tab is labeled "main.r". The code in the editor is:

```
1
2 fibonacci <- function(n) {
3   if (n == 0) {
4     return(0)
5   } else if (n == 1) {
6     return(1)
7   } else {
8     return(fibonacci(n-1) + fibonacci(n-2))
9   }
10 }
11
12 # Test the function by generating the first 10 numbers in the sequence
13 for (i in 0:9) {
14   print(fibonacci(i))
15 }
```

The output window shows the results of running the script:

```
Rscript /tmp/N2hvIrq8rf.r
[1] 0
[1] 1
[1] 1
[1] 2
[1] 3
[1] 5
[1] 8
[1] 13
[1] 21
[1] 34
```

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

The screenshot shows a code editor interface with a toolbar at the top. The file tab is labeled "main.r". The code in the editor is:

```
1
2 factorial <- function(n) {
3   if (n == 0) {
4     return(1)
5   } else {
6     return(n * factorial(n - 1))
7   }
8 }
9 n <- 7
10 result <- factorial(n)
11 cat("The factorial of", n, "is", result)
12
```

The output window shows the results of running the script:

```
Rscript /tmp/N2hvIrq8rf.r
The factorial of 7 is 5040
```

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

```

The screenshot shows a web-based R compiler interface. On the left, there's a sidebar with icons for various programming languages like Python, Java, C, C++, etc. The main area has tabs for 'main.r' and 'Output'. The 'main.r' tab contains the following R script:

```

1 Id <- rep(1:4, each = 3)
2 x<-seq(1,43,along.with=Id)
3 y<-seq(-20,0,along.with=Id)
4 Letter<-rep(letters[1:3],4)
5
6
7 df <- data.frame(Id,Letter,x,y)
8 df

```

The 'Output' tab shows the results of running the script:

```

Rscript /tmp/N2hvTng8rf.r
  Id Letter      x      y
1 1       a 1.000000 -20.000000
2 1       b 4.818182 -18.181818
3 1       c 8.636364 -16.263636
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 1.000000 -20.000000 4.818182 -18.181818
8.636364 -16.363636 4 2 12.454545 -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 35.363636 -3.636364 39.181818 -1.818182
43.000000 0.000000

```

```

R Gui (32-bit)
File Edit Packages Windows Help
R Console
> print(col_sums)
Error in print(col_sums) : object 'col_sums' not found
> a_ml
Error: object 'a_ml' not found
> df
function (x, df1, df2, ncp, log = FALSE)
{
  if (missing(ncp))
    .Call(C_df, x, df1, df2, log)
  else .Call(C_dnf, x, df1, df2, ncp, log)
}
<bytecode: 0x095e6d90>
<environment: namespace:stats>
>      cy = y[3])
Error: unexpected ')' in "
> print(df)
function (x, df1, df2, ncp, log = FALSE)
{
  if (missing(ncp))
    .Call(C_df, x, df1, df2, log)
  else .Call(C_dnf, x, df1, df2, ncp, log)
}
<bytecode: 0x095e6d90>
<environment: namespace:stats>
> |
4

```

```

C:\Users\abhis\OneDrive\Documents\2 lab.R - R Editor
import pandas as pd
import numpy as np

# Define the original data frame
df = pd.DataFrame({'Id': [1, 2, 3, 4],
                   'a': [1, 2, 3, 4],
                   'b': [5, 6, 7, 8],
                   'c': [9, 10, 11, 12]})

# Calculate the new values for the data frame
df['x'] = df['a'] * 11.45455
df['ay'] = df['a'] * -2.72727
df['bx'] = df['b'] * -5.43636
df['by'] = df['b'] * 3.43636
df['bx'] = df['b'] * -1.81818
df['cy'] = df['c'] * -0.00000

# Rename the columns to match the desired output
df = df.rename(columns={'x': 'x.ay.bx.cy.c',
                       'a': 'Id'})

# Reorder the columns to match the desired output
df = df[['Id', 'x.ay.bx.cy.c', 'x', 'ay', 'ax', 'by', 'bx', 'cy']]

# Display the resulting data frame
print(df)

```

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

```

```

main.r
1 # Define df1
2 df1 <- data.frame(Id = c(1, 2, 3, 4),
3                     Age = c(14, 12, 15, 10))
4
5
6 # Define df2
7 df2 <- data.frame(Id = c(1, 2, 3, 4),
8                     Sex = c("F", "M", "M", "F"),
9                     Code = c("a", "b", "c", "d"))
10
11 # Merge df1 and df2 to create M
12 M <- merge(df1, df2, by = "Id")
13
14 # Display the resulting data frame
15 M
16

```

Output

```

Rscript /tmp/N2hvIrq8rf.r
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
2	2	12	M	b
3	3	15	M	c
4	4	10	F	d

```

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

# Merge M and df3 to create N
N <- merge(M, df3, by.x = "Id", by.y = "id2")

# Display the resulting data frame
N

```

Output:

	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

```

```
main.r
1 data(iris)
2 tapply(iris$Sepal.Width, iris$Species, median)
3
```

Output

```
Rscript /tmp/jTHeAQcju8.r
setosa versicolor virginica
3.4 2.8 3.0
```

Exercise 6

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

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

The screenshot shows a browser window with multiple tabs open. The active tab is 'Programiz R Online Compiler' at programiz.com/r/online-compiler/. The code editor contains an R script named 'main.r' with the following content:

```

1 # Load the trees dataset
2 data(trees)
3
4 # Check if it's a data frame, convert if not
5 if(!is.data.frame(trees)) {
6   trees <- as.data.frame(trees)
7 }
8
9 # Create the data frame A
10 A <- data.frame(
11   Girth = c(mean(trees$Girth), min(trees$Girth), max(trees$Girth), sum
12     (trees$Girth)),
13   Height = c(mean(trees$Height), min(trees$Height), max(trees$Height), sum
14     (trees$Height)),
15   Volume = c(mean(trees$Volume), min(trees$Volume), max(trees$Volume), sum
16     (trees$Volume))
17 )
18
19 # Name the rows
20 rownames(A) <- c("mean_tree", "min_tree", "max_tree", "sum_tree")
21
22 # Print the result
23 A

```

The output window shows the command 'Rscript /tmp/BOSXfIuw3v.r' followed by the output of the R script, which includes the calculated statistics for Girth, Height, and Volume.

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

The screenshot shows the R code for Exercise 7:

```

# Order the data frame by the first column
A <- A[order(A$Girth), ]

# Rename the row names
rownames(A) <- c("mean", "min", "max", "tree")

# Print the result
A

```

Below the code, a note says: "This should output the following data frame:" followed by a table showing the expected output:

	Girth	Height	Volume
mean	8.30000	63.00000	10.20000
min	13.24839	76.00000	30.17097
max	20.60000	87.00000	77.00000
tree	410.70000	2356.00000	935.30000

Exercise 8

Create an empty data frame with column types:

>df
 IntsLogicals Doubles Characters
 (or 0-length row.names)

The screenshot shows a browser window with multiple tabs open. The active tab is 'Online R Compiler' at programiz.com/r/online-compiler/. The interface includes a sidebar with file icons for various languages like Python, Java, C, C++, etc., and a main area for code input and output. The code in the input field is:

```
main.r
1 # Create an empty data frame with column types
2 df <- data.frame(
3   Intslogicals = integer(),
4   Doubles = numeric(),
5   Characters = character(),
6   stringsAsFactors = FALSE
7 )
8
9 # Set row names to 0-length
10 rownames(df) <- NULL
11
12 # Print the result
13 df
14 |
```

The output panel shows the R script output:

```
Rscript /tmp/05SwAyA53X.r
[1] Intslogicals    Doubles      Characters
<0 rows> (or 0-length row.names)
```

The system tray at the bottom shows weather (84°F Partly cloudy), system icons, and a date/time indicator (03-05-2023 20:48).

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.

The screenshot shows a web-based R compiler interface. The top navigation bar includes tabs for 'Day 2 Lab Manual_part2 - Google', 'Consider the previous one data...', 'Online R Compiler', and 'Upload files - kilari08/r-program'. Below the navigation is the Programiz logo and 'R Online Compiler' text. On the left, there's a sidebar with icons for various languages: Python, R, C, C++, JS, GO, PHP, and R. The main area has a 'main.r' tab open. The code editor contains the following R script:

```
1 # Create an empty data frame with column types
2 df <- data.frame(
3   IntsLogicals = integer(),
4   Doubles = numeric(),
5   Characters = character(),
6   stringsAsFactors = FALSE
7 )
8
9 # Set row names to 0-length
10 rownames(df) <- NULL
11
12 # Print the result
13 df
14
```

Next to the code editor is a 'Run' button. To the right is an 'Output' panel showing the results of the R script:

```
Rscript /tmp/05SwAyA53X.r
[1] IntsLogicals Doubles      Characters
<0 rows> (or 0-length row.names)
```

Exercise 10

Use the (built-in) dataset Titanic.

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

```

main.r
1 # Load the Titanic dataset
2 data(Titanic)
3
4 # Convert the table to a data frame
5 df <- as.data.frame(Titanic)
6
7 # Filter the data frame to include only the desired rows and columns
8 df_filtered <- df[df$Class == "1st" & df$Survived == "No", c("Sex", "Age", "Freq"
   )]
9
10 # Replace the levels in the Age column with "Child" and "Adult"
11 df_filtered$Age[df_filtered$Age == "1"] <- "Child"
12 df_filtered$Age[df_filtered$Age == "2"] <- "Adult"
13
14 # Print the final data frame
15 df_filtered
16
17

```

Output

```

Rscript /tmp/05SwAyA53X.r
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*”.

The screenshot shows a browser window with multiple tabs open. The active tab is 'programiz.com/online-compiler'. The interface includes a sidebar with language icons (Python, R, C, C++, JS, GO, PHP, SWIFT, RUST) and a main area divided into 'Code' and 'Output' panes. The 'Code' pane contains an R script named 'main.r' with the following content:

```
1 # Create the buildings data frame
2 buildings <- data.frame(location=c(1, 2, 3), name=c("building1", "building2",
   , "building3"))
3
4 # Create the data data frame
5 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))
6
7 # Merge the two data frames by the "location" variable
8 buildingStats <- merge(buildings, data, by="location")
9
10 # Print the merged data frame
11 buildingStats
12
```

The 'Output' pane displays the resulting R script output:

```
Rscript /tmp/05SwAyA53X.r
location    name survey efficiency
1          1 building1     1      51
2          1 building1     2      258
3          2 building2     1      64
4          2 building2     2       7
5          3 building3     1      70
6          3 building3     2      80
```

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.

The screenshot shows a Windows desktop environment. At the top, there's a taskbar with several open windows: "Day 2 Lab Manual_part2 - Google Sheets", "Consider the previous one data...", "Online R Compiler", and "Upload files - kilari08/r-program...". Below the taskbar is a browser window for "programiz.com/r/online-compiler/". The main area of the browser shows an R script named "main.r" and its output. The script creates two data frames, merges them, and prints the result. The output is an R script and a table showing building statistics.

```

main.r
1 # Create the buildings data frame
2 buildings <- data.frame(location=c(1, 2, 3), name=c("building1", "building2",
   , "building3"))
3
4 # Create the data data frame
5 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))
6
7 # Merge the two data frames by the corresponding variables
8 buildingStats <- merge(buildings, data, by.x="location", by.y="LocationID")
9
10 # Print the merged data frame
11 buildingStats
12
13

```

location	name	survey	efficiency
1	building1	1	51
2	building1	2	58
3	building2	1	64
4	building2	2	71
5	building3	1	70
6	building3	2	80

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.

This screenshot is similar to the one above, showing the same desktop environment and browser setup. However, it includes a promotional banner for "LOOKING TO LEARN PROGRAMMING?" from Programiz, which says "Start your programming journey with Programiz AT NO COST." The R script and its output remain the same as in the first screenshot.

```

main.r
1 # Create the buildings data frame
2 buildings <- data.frame(location = c(1, 2, 3), name = c("building1", "building2",
   , "building3"))
3
4 # Create the data data frame
5 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))
6
7 # Merge the two data frames without specifying the key variable
8 buildingStats <- merge(buildings, data)
9
10 # Print the merged data frame
11 buildingStats
12

```

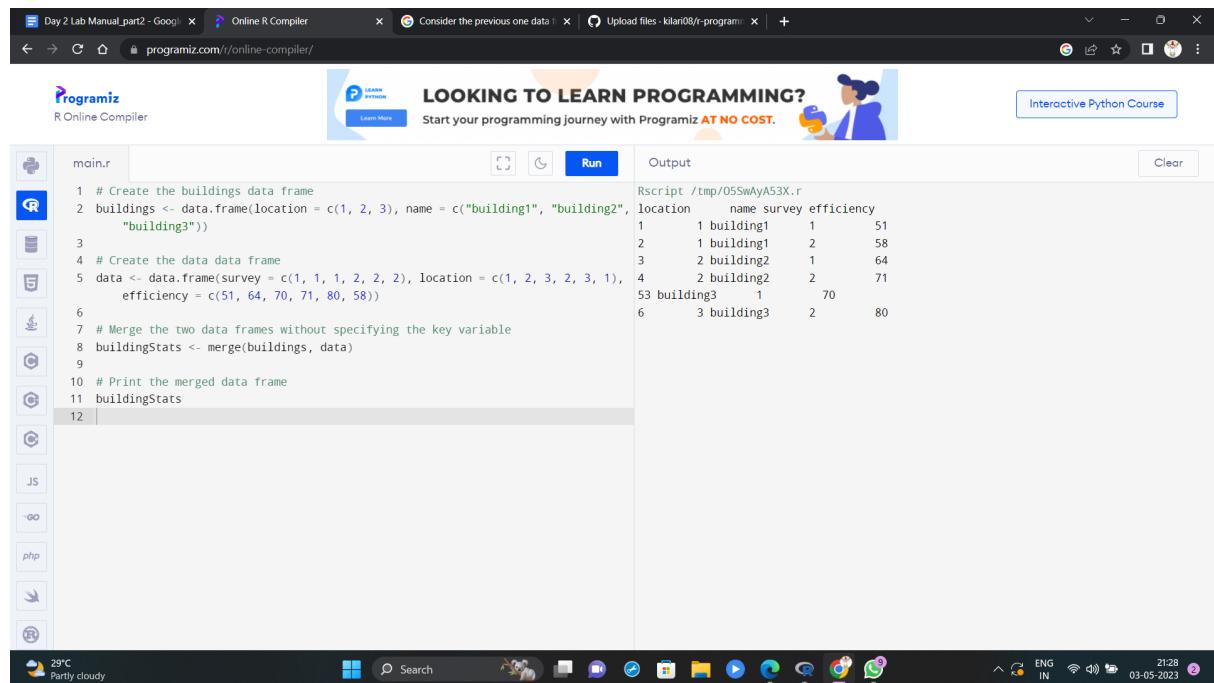
Exercise 12b)OuterJoin:

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”?



The screenshot shows a web-based R compiler interface. The code in the editor is:

```
main.r
1 # Create the buildings data frame
2 buildings <- data.frame(location = c(1, 2, 3), name = c("building1", "building2",
   "building3"))
3
4 # Create the data data frame
5 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))
6
7 # Merge the two data frames without specifying the key variable
8 buildingStats <- merge(buildings, data)
9
10 # Print the merged data frame
11 buildingStats
12
```

The output window shows the resulting data frame:

	location	name	survey	efficiency
1	1	building1	1	51
2	1	building1	2	58
3	2	building2	1	64
4	2	building2	2	71
53	building3	1	70	
6	3	building3	2	80

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, “allBuildings”.

The screenshot shows the RGui interface. In the foreground, the R Console window displays an error message:

```

if (missing(ncp))
  .Call(C_dfn, x, df1, df2, log)
else .Call(C_dnf, x, df1, df2, ncp, log)
> N
Error: object 'N' not found
> Nmerge(M,df3,by=x=="Id",by.y="id2")
Error in merge(M, df3, by.x = "Id", by.y = "id2") : obj
> N
Error: object 'N' not found
> N
Error: object 'N' not found
> N_long
Error: object 'N_long' not found
> g
Error: object 'g' not found
> allBuildings <- rbind(buildings, buildings2)
Error in rbind(buildings, buildings2) : object 'buildin
> allBuildings <- rbind(buildings, buildings2)
Error in rbind(buildings, buildings2) : object 'buildin
> |

```

In the background, the R Editor window titled 'C:\Users\abhis\OneDrive\Documents\2lab.R - R Editor' shows some code:

```

buildings <- data.frame(location = c(1, 2, 3), name = c("building1", "building2",
buildings2 <- data.frame(location = c(5, 4, 6), name = c("building5", "building4"
allBuildings <- rbind(buildings, 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.

The screenshot shows the RGui interface. In the foreground, the R Console window displays an error message:

```

.Call(C_dfn, x, df1, df2, log)
else .Call(C_dnf, x, df1, df2, ncp, log)
> N
Error: object 'N' not found
> Nmerge(M,df3,by=x=="Id",by.y="id2")
Error in merge(M, df3, by.x = "Id", by.y = "id2") : obj
> N
Error: object 'N' not found
> N
Error: object 'N' not found
> N_long
Error: object 'N_long' not found
> g
Error: object 'g' not found
> allBuildings <- rbind(buildings, buildings2)
Error in rbind(buildings, buildings2) : object 'buildin
> allBuildings <- rbind(buildings, buildings2)
Error in rbind(buildings, buildings2) : object 'buildin
> allBuildings <- rbind(buildings, buildings2)
Error in rbind(buildings, buildings2) : object 'buildin
> buildings3_new <- buildings3[, c("location", "name")]
Error: object 'buildings3' not found
> |

```

In the background, the R Editor window titled 'C:\Users\abhis\OneDrive\Documents\2lab.R - R Editor' shows the creation of the new dataframe:

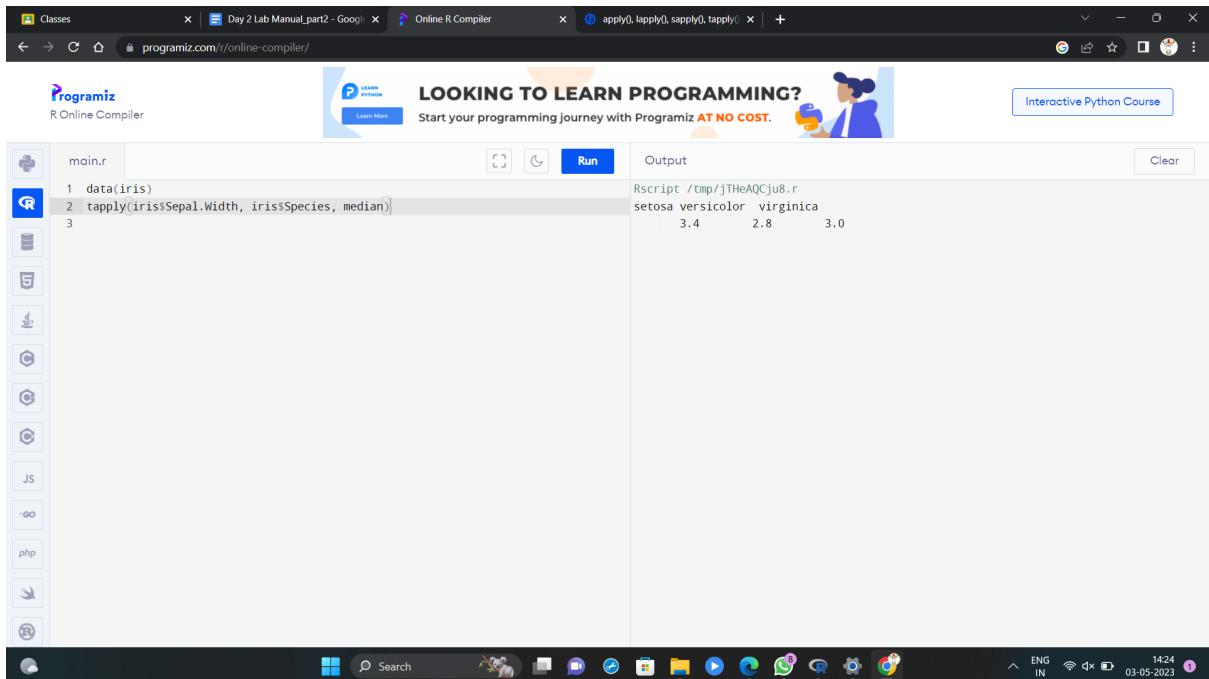
```

buildings3 <- data.frame(location = c(7, 8, 9), name = c("building7", "building8",
buildings3_new <- buildings3[, c("location", "name")]

```

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.



The screenshot shows a browser window with multiple tabs open. The active tab is 'Day 2 Lab Manual_part2 - Google' and the URL is 'programiz.com/r/online-compiler/'. The page displays an R script editor with the file 'main.r' containing the following code:

```
1 data(iris)
2 tapply(iris$Sepal.Width, iris$Species, median)
3
```

Below the code editor is a 'Run' button. To the right, there is an 'Output' panel showing the results of the R script execution:

```
Rscript /tmp/jTHeAQcju8.r
setosa versicolor virginica
      3.4          2.8          3.0
```

The browser's address bar shows the URL 'programiz.com/r/online-compiler/'. The bottom of the screen shows a Windows taskbar with various icons and the system tray indicating the date and time as '03-05-2023 14:24'.

RESHAPE FUNCTION IN R

Exercise: 16

Construct the following data frame ‘country’.

	countries	value.population_in_million	value.gdp_per capita
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.

The screenshot shows an R online compiler interface. On the left, there's a file browser with 'main.r' selected. The code in 'main.r' is:

```

library(tidyverse)
# create the countries data frame
countries <- data.frame(
  country = c("A", "B", "C"),
  population_in_million = c(100, 200, 120),
  gdp_per capita = c(2000, 7000, 15000)
)
# reshape from wide to long
countries_long <- gather(countries, key = "variable", value = "value", -country)

```

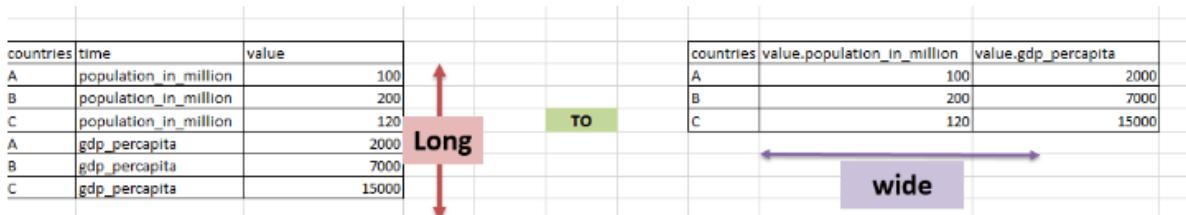
The output window shows the resulting long data frame:

\$ country	variable	value
1 A	population_in_million	100
2 B	population_in_million	200
3 C	population_in_million	120
4 A	gdp_per capita	2000
5 B	gdp_per capita	7000
6 C	gdp_per capita	15000

Below the code editor, there's a diagram illustrating the transformation. It shows a 'wide' data frame on the left and a 'long' data frame on the right. A green arrow labeled 'TO' points from the wide frame to the long frame. The wide frame has columns 'countries', 'population_in_million', and 'gdp_per capita'. The long frame has columns 'countries', 'time', and 'value'. The 'time' column in the long frame corresponds to the variable names in the wide frame.

- 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:



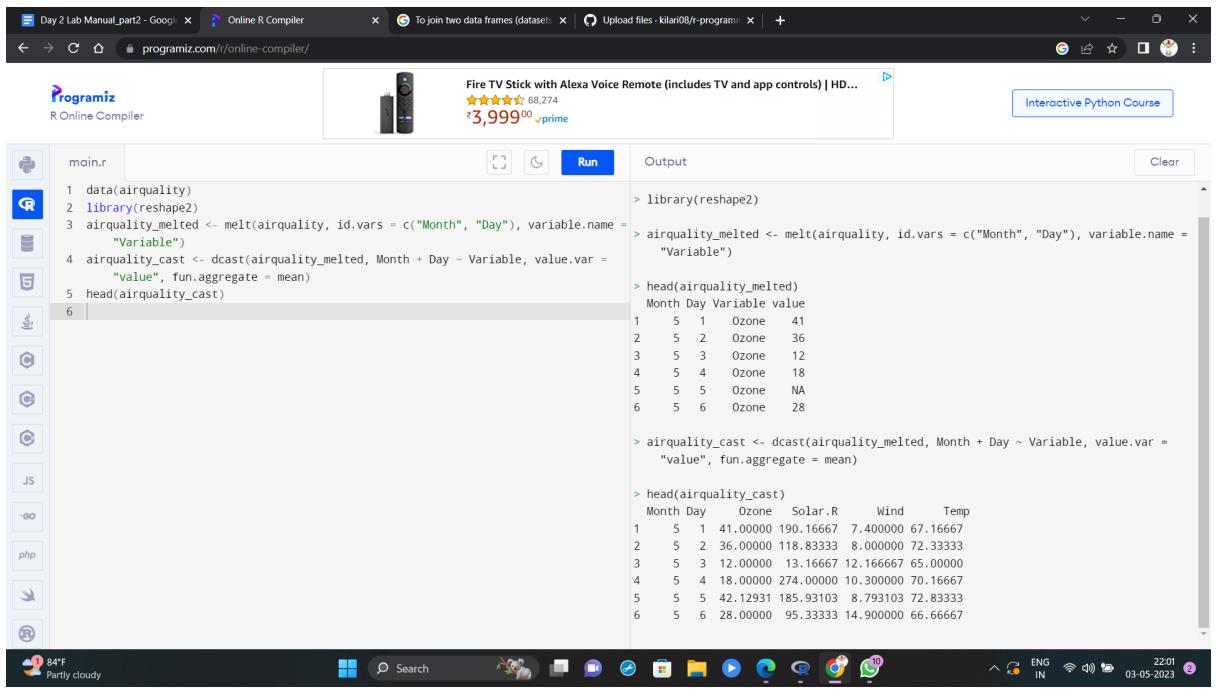
- data (country_w_to_L) which is in long format, is passed to reshape function
- idvar is the variable which need to be left unaltered, which is “countries”
- timevar are the variables that needs to converted to wide format
- v.names are the value variable

- 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 ?



The screenshot shows a web-based R environment. At the top, there are tabs for 'Day 2 Lab Manual_part2 - Google', 'Online R Compiler', 'To join two data frames (dataset)', and 'Upload files - kiran08/r-program'. Below the tabs, there's a banner for a Fire TV Stick with Alexa Voice Remote. The main area has a sidebar with icons for Python, R, C, C++, Go, JS, and PHP. The code editor window contains the following R script:

```

main.r
1 data(airquality)
2 library(reshape2)
3 airquality_melted <- melt(airquality, id.vars = c("Month", "Day"), variable.name =
  "Variable")
4 airquality_cast <- dcast(airquality_melted, Month + Day ~ Variable, value.var =
  "value", fun.aggregate = mean)
5 head(airquality_cast)
6

```

The output window shows the results of the R commands:

```

> library(reshape2)
> airquality_melted <- melt(airquality, id.vars = c("Month", "Day"), variable.name =
  "Variable")
> head(airquality_melted)
  Month Day Variable value
1     5    1   Ozone    41
2     5    2   Ozone    36
3     5    3   Ozone    12
4     5    4   Ozone    18
5     5    5   Ozone     NA
6     5    6   Ozone    28

> airquality_cast <- dcast(airquality_melted, Month + Day ~ Variable, value.var =
  "value", fun.aggregate = mean)

> head(airquality_cast)
  Month Day Ozone Solar.R   Wind   Temp
1     5    1 41.00000 190.16667 7.400000 67.16667
2     5    2 36.00000 118.83333 8.000000 72.33333
3     5    3 12.00000 13.16667 12.166667 65.00000
4     5    4 18.00000 274.00000 10.300000 70.16667
5     5    5 42.12931 185.93103 8.793103 72.83333
6     5    6 28.00000 95.33333 14.900000 66.66667

```

The status bar at the bottom indicates the system is 'Partly cloudy' at 34°F.

8 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.

```
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
```

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

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

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

