2.9 APPLYING R ON DATASETS

In this section you will be introduced to using the basic functions and operations of R in processing data, your baptism of fire. You will also be introduced to some of the functions that are commonly used in R, particularly in Data Processing as we go along with this learning session.

To start, let's create a simple dataset of 31 records of students with Jame, Age and Score. We can do this in severals ways:

- Create the dataset using the ***data.frame function***.
- Create the list in any spreadsheet application, save the file in a CSV format then load it in R using the ***read_csv2 function***.
- Create the dataset in MS Excel and use the ***read_excel function*** *from the* ***readxl*** *Library*.
- Create the dataset in a JSON format (.json) and use the ***fromJSON function*** *from the* ***rjson*** *Library*.

The list goes on... and mostly, it will need you to install Libraries and understand the functions to make it work.

But with this learning session, we will stay within the base functions that are included within the R Base Packages. And with this, we can use either the ***read_csv function*** or create the dataset using the ***data.frame function***. And before anything else, we need to have a data that we can work on. So, let's use a "*made-up*" dataset of student with Names, Age and Score to simulate a test score record of random students whose ages are between 19 to 23 years of age and a score ranging from 60 to 100.

Sample Dataset

(you can generate your own records)

Name	Age	Score
Alice	20	85
Bob	21	78
Charlie	19	92
David	22	65
Eva	20	88
Frank	23	95
Grace	21	70
Henry	20	76
Ivy	19	90
Jack	22	60
Kate	21	82
Leo	20	91
Mia	23	87
Noah	21	79
Oliver	20	84
Paul	22	68
Quinn	19	93
Ruby	21	77
Sam	20	89
Tina	22	64
Uma	21	81
Vic	20	86
Will	23	90
Xena	19	95
Yara	21	72
Zack	20	80
Anna	22	67
Ben	21	74
Carla	20	88
Derek	23	85
Ella	21	78

You can copy the sample dataset to your favorite spreadsheet application and save it as a CSV file (*i.e. sampleData.csv)*, *just your the default settings*. Once saved, all you need to do is to open your *R Console* or *R Studio IDE* and run the following command:

1. Check your Current Working Directory:

```
> getwd()

# this command allows you to get the current working directory of your R
# this will surely be set to a default location, your User Directory
```

2. Set your New Working Directory:

```
> setwd()

# this command will allow you set the working directory for your R
# let's make sure to specify a working directory for our R session
# replace MyWorkingDirectory with your preferred directory

> setwd('MyWorkingDirectory')
```

3. Load your dataset from a CSV File:

```
> students <- read.csv('sampleData.csv', header = TRUE)
```

4. If all goes well, your dataset, named **students**, will be loaded to R. Then we now try to view our dataset.

```
> students
# this will just directly print values of the dataset (students)

> head(students)
# display the first 6 records of the dataset

> tail(students)
# this will directly print the last 6 records of the dataset
```

If you were able to display the dataset in step #4, then you are all set for data processing. But, before we do that, let's do an alternative process in creating a dataset. And take note, this will require a bit more work but it is a good way to better understand dataset preparation using **data.frame** and **vectors**. So, let's do some manual dataset creation.

1. Create each records (Name, Age & Score) using the **vector** data structure and combine function, ***c()***.

```
> Name <- c("Alice", "Bob", "Charlie", "David", "Eva", "Frank", "Grace", "Henry",
           "Ivy", "Jack","Kate", "Leo", "Mia", "Noah", "Oliver", "Paul","Quinn",
           "Ruby", "Sam", "Tina", "Uma", "Vic", "Will", "Xena", "Yara", "Zack",
           "Anna", "Ben", "Carla", "Derek", "Ella")

> Age <- c(20, 21, 19, 22, 20, 23, 21, 20, 19, 22, 21, 20, 23, 21, 20, 22,
          19, 21, 20, 22, 21, 20, 23, 19, 21, 20, 22, 21, 20, 23, 21)

> Score <- c(85, 78, 92, 65, 88, 95, 70, 76, 90, 60, 82, 91, 87, 79,
            84, 68, 93, 77, 89, 64, 81, 86, 90, 95, 72, 80, 67, 74,
            88, 85, 78)

# each of the created vectors represent the columns of the student dataset
```

2. Create the dataset (students) using the ***data.frame()*** function.

```
> students <- data.frame(Name, Age, Score)

# this command combines all 3 vectors into a dataframe (students)
# take note on the arrangement of the vectors
```

3. Test your dataset by displaying the values. Refer to Step #4 in page 79.

Now that we have a dataset that we can work on, let's inspect it first to better understand what we are dealing with before doing something with it.

- ***Display the Dataset Structure***

```
> str(students)
# str() is a function that displays the structure of the Object
```

- ***Check the Dimension of the Dataset***

```
> dim(students)
# dim() displays the dimensions of the Object
```

- ***Draw out the names of the Columns***

```
> names(students)
# names() will list down the Names of the Object (Columns)
```

- ***Get the General Description of the Dataset***

```
> summary(students)
# summary() will list down a General Description of the Object
# the displayed result will vary based on the Object itself
```

- ***Plot a General Visualization of the Dataset***

(more on this data exploration and visualization, Module 3)

```
> plot(students)
# plot() will list draw a general visualization based on the records and
columns of the Object.
# this will allow you to assess which variables would be best fitted to be
processed and evaluated more to better describe the dataset
```

Now let's use some investigate further on our dataset and use the Basic Concepts of R to process the data. Let's start by

asking some questions to better understand the dataset and we will implement some R codes to get the results.

1. How many records does the dataset contain?

```
> dim(students)
# dim() displays 2 values, rows and columns

> nrow(students)
# nrow() displays the number of rows in an object.
# ncol() displays the number of columns in an object.
# dim() displays both
```

Answer: ____

2. How many students in each Age?

```
> table(students$Age)
# table() displays the count of each unique value of a specific column
```

Answer: 19 20 21 22 23
___ ___ ___ ___ ___

3. What are the unique ages represented in the dataset?

```
> unique(students$Age)
# unique() displays each unique value of a specific column
```

Answer: ___ ___ ___ ___ ___

4. What is the average score of the students?

```
> mean(students$Score)
# mean() calculates and displays the average of the values of the column
```

Answer: _____

5. Which student has the highest score?

```
> max(students$Score)
```
*# **max()** displays maximum value of the provided data*

```
> students[which.max(students$Score), ]
```
*# **which.max()** searches for the maximum value and returns the complete record (row) of the dataset using the index value*

Answer: _____

6. Which student has the lowest score?

```
> min(students$Score)
```
*# **min()** displays minimum value of the provided data*

```
> students[which.min(students$Score), ]
```
*# **which.min()** searches for the minimum value and returns the complete record (row) of the dataset using the index value*

Answer: _____

7. What is the median age of the students?

```
> median(students$Age)
```
*# **median()** searches and displays the median value of the provided data*

Answer: _____

8. How many students scored above 80?

```
> number_of_students_above_80 <- sum(students$Score > 80)
```
*# **sum()** when used this way will behave as a count() in other languages*

Answer: _____

9. What is the age range of the students (oldest and youngest)?

```
> range(students$Age)
```
*# **range()** displays the minimum and maximum value of a specific dataset*
*# **min()** and **max()** can also be used. But these are 2 different commands*

10. Are there any students with the same score? If so, how many?

```
> table(students$Score)
```
 # this is the simplest command to execute, find the score with a value greater than 1 (one) and that's it

```
> myTab <- table(students$Score)
> myTab_duplicate <- myTab[myTab > 1]
> nrow(myTab_duplicate)
```
 # this is a bit complicated but all it does is create a table of unique values of Score and a count of repetition then at the second command, it removes all records whose count is not greater than 1.
 # the last command counts the number of rows in your Object.

```
> sum(duplicated(students$Score))
```
*# **duplicated()** counts the number of items that has a duplicated value*
*# using **sum()** as a counter*

Answer: _____

11. How many students fall within specific age groups (e.g., 18-20, 21-23)?

```
> myTab_by_Age <- cut(students$Age, breaks = c(18, 20, 23), right = FALSE)
> myTab_count <- table(myTab_by_Age)
```
*# **cut()** categorizes the ages based on the provided parameters in the **breaks**,*
*# the **right** parameters specifies if the right most value is included or not*
*# with the given category, running the **table()**, counts the number of occurrences each category repeats itself, in other words, **count**.*

Answer: [18,20) [20,23)

12. What percentage of students scored above the average score?

```
> mean(students$Score > mean(students$Score)) * 100
```

*# **mean()** is the average of the Score.*
 # this process will collate all Scores that are greater than the average score and then computes the percentage of the Scores above the average score.

Answer: _____

13. List down the all the students and put a remark Passed when the score is 75 or above and Failed if not.

```
> for (i in 1:nrow(students)) {
    if (students$Score[i] >= 75) {
        cat(students$Name[i], "Passed.\n")
    }
    else {
        cat(students$Name[i], "Failed.\n")
    }
}
```

14. Create additional column in the **students** dataset named **Grade** where **A** is given to Scores from *90* and above, **B** from 80 to 89, **C** for *Scores below 80*

```
> students$Grade <- ifelse(students$Score >= 90, "A",
                           ifelse(students$Score >= 80, "B", "C"))
```

*# creating a new column in a dataset is as simple as calling the **students** dataset and placing a new column name **Grade (students$Grade)***

HANDS-ON ACTIVITY # 4

Well then, now that we have learned how to use the basic commands in R to process data, let us now emulate what we have done in this session.

Scenario:

You are provided a dataset with at least 3 columns and 40 records. You are then asked to describe the dataset and provide some data processing operation an produce a valuable result.

Data: *You can create a dataset on your own or find some simple datasets online*

Assessing and Describing a Dataset:

Objective: Apply some analysis to describe your dataset and present some valuable data to further description of the said dataset.

Steps:

1. Create your dataset.

2. Load your dataset to R (*refer to pages 77 to 80*).

3. Use the basic R commands that will describe the dataset (*refer to page 81*).

4. Once you are done with the basic dataset description, do some data processing to better evaluate and process the information in your dataset. (*use the example questions found in pages 81 to 85 as reference*).

Paste or write your R Scripts below.
(Should you be using a CSV file for your dataset, please include a copy within this document)

>

(Add more sheets when needed)

>

(Add more sheets when needed)