Name: Xian Hui B. Cheng

Section: BSIT 3-5

Activity 6: Using APPLY Family FUNCTIONS

You are given a dataset with students' grades in three subjects: Math, Science, and English.

```
students <- data.frame(
  Name = c("John", "Alice", "Bob", "David", "Emma"),
  Math = c(85, 90, 78, 92, 88),
  Science = c(82, 88, 76, 95, 85),
  English = c(89, 91, 80, 88, 94)
)
```

1. Using the apply() function, calculate the average score for each student across all three subjects (Math, Science, and English). Add a new column to the students dataset that contains the average score.

Note: you can also not use a function. We just used it to apply D.R.Y Principle

```
calculate_average <- function(data){
  apply(data[, c("Math", "Science", "English")], 1, mean)
}

students$Average <- calculate_average(students)
print(students)
```

Other Option:

```
students$Average <- apply(students[, c("Math", "Science", "English")], 1, mean)
print(students)
```

Output:

```
   Name Math Science English  Average
   John   85      82      89 85.33333
  Alice   90      88      91 89.66667
    Bob   78      76      80 78.00000
  David   92      95      88 91.66667
   Emma   88      85      94 89.00000
```

Explanation:

apply()

- This is used to apply to the rows or columns in the data frame

1st argument: ***students[, c("Math", "Science", "English")]***

- This is used to select the columns Math, Science, and English from the data frame students.

2nd argument: **1**

- This specifies row-wise operations which allows to focus on the row

3rd argument: **mean**

- This applies the function mean in order to compute the mean for each row

2. Using the lapply() function, calculate the mean score for each subject (Math, Science, English). Display the result as a list where each element represents the mean score of a subject.

```
mean_score <- lapply(students[, c("Math", "Science", "English")], mean)
print(mean_score)
```

Output:

```
$Math
[1] 86.6

$Science
[1] 85.2

$English
[1] 88.4
```

Explanation:

lapply()

- This applies a function to each element. However the difference between apply() and lapply() is that lapply() are used for lists or vectors and **outputs a list.**

1st argument: ***students[, c("Math", "Science", "English")]***

- This is used to select the columns Math, Science, and English from the data frame students.

2nd argument: ***mean***

- lapply() applies the mean function to each of the element in the column provided (Math, Science, English)

3. Using the sapply() function, calculate the total score for each subject (Math, Science, and English). Display the result as a vector where each element represents the total score of a subject.

```
total_score <- sapply(students[, c("Math", "Science", "English")], sum)
print(total_score)
```

Output:

```
 Math Science English
  433     426     442
```

Explanation:

sapply()

- Similar to apply() and lapply()), this applies a function to each element. However, it simplifies the output of lapply() into a vector or matrix, it also gives a simplified output.

1st argument: ***students[, c("Math", "Science", "English")]***

- This is used to select the columns Math, Science, and English from the data frame students.

2nd argument: ***sum***

- sapply() applies the sum function to each of the element in the column provided (Math, Science, English) to get the total score then simplify result into a **vector**

4. Using the apply() function, calculate the highest score in each subject (Math, Science, and English). Use apply() to perform the operation on the columns (i.e., find the maximum value in each column).

```
highest_score <- apply(students[, c("Math", "Science", "English")], 2, max)
print(highest_score)
```

Output:

```
 Math Science English
   92      95      94
```

Explanation:

apply()

1st argument: ***students[, c("Math", "Science", "English")]***

- This is used to select the columns Math, Science, and English from the data frame students.

2nd argument: **2**

- This specifies column-wise operations which allows it to focus on the column. Since 1 is for row and 2 is for column.

3rd argument: **max**

- This applies the function max to each of the values in each column. ***max*** is used to calculate the maximum value of each column.

5. Using lapply(), create a new list where each element contains a vector of scores greater than or equal to 85 for each subject. For example, in the "Math" subject, the result should be a list of scores in that subject that are greater than or equal to 85.

```
passing_grade <- lapply(students[, c("Math", "Science", "English")], function(grades) {
  grades [grades >= 85]
})

print(passing_grade)
```

Output:

```
$Math
[1] 85 90 92 88

$Science
[1] 88 95 85

$English
[1] 89 91 88 94
```

Explanation:

lapply()

1st argument: ***students[, c("Math", "Science", "English")]***

- This is used to select and iterated over the columns Math, Science, and English from the data frame students.

2nd argument: ***function(grades)***

- This is an anonymous function wherein it will be applied to each subject column
    - ***grades*** - input vector that represents each value for every column (subject)
    - The condition grades[grades >= 85] is the one used to filter if the specific grade in that iteration is greater or equal to 85
    - ***subsetting*** - TRUE or FALSE is used to filter the original grades vector. Only elements where condition is TRUE will be added.

Subject	Grades	Condition	Filtered Grades
Math	85, 90, 78, 92, 88	TRUE, TRUE, FALSE, TRUE, TRUE	85, 90, 92, 88
Science	82, 88, 76, 95, 85	FALSE, TRUE, FALSE, TRUE, TRUE	88, 95, 85
English	89, 91, 80, 88, 94	TRUE, TRUE, FALSE, TRUE, TRUE	89, 91, 88, 94

6. Using sapply(), calculate the standard deviation of scores for each subject. Display the result as a vector where each element is the standard deviation of a subject.

```
sd_score <- sapply(students[, c("Math", "Science", "English")], sd)
print(sd_score)
```

Output:

```
    Math  Science  English
5.458938 7.049823 5.224940
```

Explanation:

sapply()

1st argument: ***students[, c("Math", "Science", "English")]***

- This is used to select the columns Math, Science, and English from the data frame students.

2nd argument: ***sd***

- sapply() applies the sd function to each of the element in the column provided (Math, Science, English) to get the standard deviation of each then simplify result into a **vector** (column)

7. Using the apply() function, create a new column in the students dataset that contains "Pass" if the average score of the student is greater than or equal to 85, and "Fail" if it is less than 85. Use apply() to check each student's average score and assign the respective result.

```
students$Rating <- apply(students[, c("Math", "Science", "English")], 1, function(grades
  ifelse (mean(grades) >= 85, "Pass", "Fail")
})

print(students)
```

Output:

```
  Name Math Science English  Average Rating
  John   85      82      89 85.33333   Pass
 Alice   90      88      91 89.66667   Pass
   Bob   78      76      80 78.00000   Fail
 David   92      95      88 91.66667   Pass
  Emma   88      85      94 89.00000   Pass
```

Explanation:

apply()

1st argument: ***students[, c("Math", "Science", "English")]***

- This is used to select the columns Math, Science, and English from the data frame students.

2nd argument: **1**

- This specifies row-wise operations which allows to focus on the row. Since 1 is for row and 2 is for column

3rd argument: **function(grades)**

- This is used to calculate the mean of the grades for each column and categorize them.

  ifelse(mean(grades) >= 85, "Pass", "Fail")
- In here, we used a ternary operator to check of the condition
- **mean(grades)** was used to calculate the average score for each element in the columns selected

students$Rating

- The $ operator was used to add a new column called Rating to the students data frame

Name	Grades (Math, Sci, Eng)	Average Score (mean)	mean(grades) >= 85	Rating
John	85, 82, 89	85.33	TRUE	Pass
Alice	90, 88, 91	89.67	TRUE	Pass
Bob	78, 76, 80	78.00	FALSE	Fail
David	92, 95, 88	91.67	TRUE	Pass
Emma	88, 85, 94	89.00	TRUE	Pass

8. Create a new list using lapply() that contains the count of scores greater than or equal to 90 for each subject. Then, use the apply() function to find the total number of students with scores greater than or equal to 90 across all subjects.

```
high_score <- lapply(students[, c("Math", "Science", "English")], function(grades) {
  sum(grades >= 90)
})

print(high_score)
```

Output:

```
$Math
[1] 2

$Science
[1] 1

$English
[1] 2
```

Explanation:

lapply()

- This applies a function to each element and iterates over them. However the difference between apply() and lapply() is that lapply() are used for lists or vectors and **outputs a list.**
- This is used in this case to count the number of scores greater than or equal to 90 for each subject.

1st argument: ***students[, c("Math", "Science", "English")]***

- This is used to select the columns Math, Science, and English from the data frame students.

2nd argument: ***function(grades)***

- ***grades >= 90*** - this checks if the grades are greater than or equal to 90.
- ***sum(grades >= 90)*** - sum is used to count the number of TRUE values that the logical vector returned.

Subject	grades	grades >= 90	Count
Math	85, **90**, 78, **92**, 88	FALSE, **TRUE**, FALSE, **TRUE**, FALSE	2
Science	82, 88, 76, **95**, 85	FALSE, FALSE, FALSE, **TRUE**, FALSE	1
English	89, **91**, 80, 88, **94**	FALSE, **TRUE**, FALSE, FALSE, **TRUE**	2