**Name:** Xian Hui B. Cheng
**Section:** BSIT 3-5

**Activity 4: Manipulating and Merging Data**

Manipulate and combine datasets using R. Write and test your R code for each question. Provide explanations for your steps and results.

1. Multi-Column Merge: Create two data frames: df1: Columns ID, Name, and Age. df2: Columns ID, Score, and Subject. Write R code to perform an inner join on ID and explain the resulting data frame structure.

```
df1 <- data.frame(
  id = c(1, 2, 8, 4, 5),
  name = c("Xian", "Marie", "Christine", "Brian", "Kyle"),
  age = c(20, 31, 20, 21, 50)
)

df2 <- data.frame(
  id = c(1, 2, 3, 4, 5),
  score = c(99, 95, 89, 75, 91),
  subject = c("Elective", "Operating System", "COBOL", "Research", "Math")
)

merged_data <- merge(df1, df2, by = "id")
print(merged_data)
```

Output:

```
 id  name age score          subject
  1  Xian  20    99         Elective
  2 Marie  31    95 Operating System
  4 Brian  21    75         Research
  5  Kyle  50    91             Math
```

Explanation:
1. Creating the two data frames with the assign columns using data.frame

2. ***merge(df1, df2, by = "id")*** - this was used in the result variable. This function matches the rows where the id column is present in df1 and df2

Result:
It only contains the rows with matching id values

| df1.ID | df2.ID | Match? | merged_data |
|---|---|---|---|
| 1 | 1 | Yes | ID=1, Name=Xian, Age=20, Score=99, Subject=Elective |
| 2 | 2 | Yes | ID=2, Name=Marie, Age=31, Score=95, Subject=Operating System |

| | | | |
|---|---|---|---|
| 8 | 3 | No | |
| 4 | 4 | Yes | ID=4, Name=Brian, Age=21, Score=75, Subject=Research |
| 5 | 5 | Yes | ID=5, Name=Kyle, Age=50, Score=91, Subject=Math |

2. Add a column to the merged data frame from Task 1 that dynamically categorizes Age into three groups:
"Young": Age <= 25
"Middle-aged": Age > 25 and <= 40
"Senior": Age > 40

```
merged_data$age_group <- ifelse(merged_data$age <= 25, "Young",
                                ifelse(merged_data$age > 25 & merged_data$age <= 40,"Middle-aged"
                                       ,"Senior"))

print(merged_data)
```

Output:

```
  id  name age score           subject   age_group
   1  Xian  20    99         Elective       Young
   2 Marie  31    95 Operating System Middle-aged
   4 Brian  21    75         Research       Young
   5  Kyle  50    91             Math      Senior
```

Explanation:
***age_group***
- We added a new column to merged_data called age_group which contains the result of ifelse()

***ifelse()***
- This is the vectorized function for conditional operators. It combines condition checking in a single function and returns a vectorized unit.
- This uses a **nested ifelse()** logic, where it checks the first condition, and moves on to the other ifelse if it evaluates FALSE

| Age | age <= 25 | age > 25 & age <= 40 | Action | age_group |
|---|---|---|---|---|
| 20 | TRUE | N/A | returns “Young” | “Young” |
| 31 | FALSE | TRUE | returns “Middle-aged” | “Middle-aged” |
| 21 | TRUE | N/A | returns “Young” | “Young” |

| | | | | |
|---|---|---|---|---|
| 50 | FALSE | FALSE | returns “Senior” | “Senior” |

3. Using the merged data, compute the average Score for each Age group created in Task 2. Explain the steps and provide the R code.

```
young_scores <- merged_data$score[merged_data$age_group == "Young"]
middle_aged_scores <- merged_data$score[merged_data$age_group == "Middle-aged"]
senior_scores <- merged_data$score[merged_data$age_group == "Senior"]

young_avg <- sum(young_scores) / length(young_scores)
middle_aged_avg <- sum(middle_aged_scores) / length(middle_aged_scores)
senior_avg <- sum(senior_scores) / length(senior_scores)

average_scores <- data.frame(
  age_group = c("Young", "Middle-aged", "Senior"),
  average_score = c(young_avg, middle_aged_avg, senior_avg)
)

print(average_scores)
```

Output

```
    age_group average_score
1       Young            87
2 Middle-aged            95
3      Senior            91
```

Explanation:

**merged_data$score[merged_data$age_group == “String”]**

- This extracts the scores from the rows of age_group and checks whether it’s Young, Middle-aged, or Senior. This is used to filter the scores by age group

**Calculating average**

- ***sum()*** **-** compute the total scores for each group.
- ***length()*** - counts the number of scores in each group. Since this is needed for the formula of average (total scores / total count)
- ***average*** - sum of scores that was calculated from sum() divided by the length from the scores

**average_scores**

- We then create a dataframe to combine the calculated average

4. Sort the merged data frame by Subject (ascending) and Score (descending). Write and explain the R code used.

```
sorted_data <- merged_data[order(merged_data$subject, -merged_data$score), ]
print(sorted_data)
```

Output:

```
 id  name age score          subject   age_group
  1  Xian  20    99         Elective       Young
  5  Kyle  50    91             Math      Senior
  2 Marie  31    95 Operating System Middle-aged
  4 Brian  21    75         Research       Young
```

Explanation:

**order()**

- This generates the rows indicated in its parameters in order. It's default order is ascending and to print the descending order we use the **negative sign**)

**Parameters of order**

merged_data$subject - this sorts the data frame by subject in ascending order which is the default behavior of order function

-merged_data$score **-** this sorts the data frame by score in descending order because of the negative sign which reverses it

5. Write an R function that accepts multiple data frames as input and combines them using rbind() after ensuring all have the same column names. Demonstrate the function with sample inputs.

```
combine_data_frames <- function(...) {
  data_frames <- list(...)

  column_names <- lapply(dfs, colnames)
  if (!all(sapply(column_names, function(x) all(x == column_names[[1]])))) {
    print("Column Names are not the same")
  }

  combined_df <- do.call(rbind, df)

  return(combined_df)
}
```

```r
# Sample Data Frames
df1 <- data.frame(
  id = c(1, 2, 3),
  name = c("Xian", "Marie", "Christine"),
  score = c(99, 95, 89)
)

df2 <- data.frame(
  id = c(4, 5, 6),
  name = c("David", "Kyle", "Brian"),
  score = c(92, 87, 89)
)
# Combine Data Frames
combined_df <- combine_data_frames(df1, df2)

# Print the Result
print(combined_df)
```

OUTPUT:

```
  id      name score
1  1      Xian    99
2  2     Marie    95
3  3 Christine    89
4  4     David    92
5  5      Kyle    87
6  6     Brian    89
```

EXPLANATION:

**combine_data_frames <- function(...)**

- This allows the function to accept any number of arguments with the (...) this allows us to accept any number of data frames as separate arguments

**data_frames <- list(...)**

- This is used to make all the data frames provided into a list, this is used to iterate over them to check if the column names exists and combine them

**column names extraction**

- lapply() - this is used to apply colnames() into each data frame in the list to create a list of column names for all the inputted data frames

**Making sure all column names are the same**

- sapply() - this is used to iterate over the column_names and check if each set of column names is identical to the first submitted column name (column_names[[1]])
- all(x == column_names[[1]]) - this compares every set of column names to the first set. If any of the set returns as FALSE then it will immediately return FALSE

- If FALSE: an error message will pop up and print(“Column Names are not the same”) will be printed

**Combining the data frames**

- **rbind()** - this combines data frames row-wise (since it’s call row bind). This stacks the data frames on top of one another. In order to use this all inputted data frames must have the same number of columns, column names, and same data types
- **do.call()** - this applies a function to a list of arguments. In this case, it was used to call rbind, and applied it to all the elements of the data frames (df). It basically pass all the arguments to rbind()