Exercises for 2nd session : Part 2

For these exercises, you will need to have the `tidyverse` package installed and loaded.

```
install.packages("tidyverse")
library(tidyverse)
```

Part 1: Simple Exercises

These exercises focus on applying one or two functions at a time to understand their core purpose. We will use the built-in `iris` and `mtcars` datasets.

Exercise 1: Selecting Specific Columns with `select()`

Context: You have the `iris` dataset and only need to work with the columns related to petal measurements.
Tasks:

1. Load the `iris` dataset (it's built-in).
2. Create a new data frame called `iris_petal` that contains only the `Petal.Length`, `Petal.Width`, and `Species` columns using the select() method
3. Display the first 6 rows of your new `iris_petal` data frame.

Exercise 2: Filtering Rows with `filter()`

Context: You are analyzing the `mtcars` dataset and want to isolate cars that are powerful.
Tasks:

1. Load the `mtcars` dataset.
2. Create a new data frame called `powerful_cars` that contains only the cars with horsepower (`hp`) greater than 200.
3. How many cars are in the `powerful_cars` data frame?

Exercise 3: Sorting a Data Frame with `arrange()`

Context: You need to sort the `iris` dataset to easily find the plants with the longest and shortest sepals.
Tasks:

1. Create a new data frame called `iris_sorted` by arranging the `iris` dataset by `Sepal.Length` in descending order.
2. Display the top 5 rows of `iris_sorted` to see the flowers with the longest sepals.
3. Now, create another data frame `iris_sorted_asc` arranged by `Sepal.Length` in ascending order and display its top 5 rows.

Exercise 4: Creating a New Column with `mutate()`

Context: The `mtcars` dataset has weight (`wt`) in thousands of pounds. You need to convert this to kilograms.
Tasks:

1. Create a new data frame called `mtcars_metric`.
2. Add a new column to it called `wt_kg` which is the `wt` column multiplied by 453.592 (since `wt` is in 1000s of lbs, this is a simplification).
3. Display the `mpg`, `wt`, and the new `wt_kg` columns for the first 5 cars.

Exercise 5: Combining `filter()` and `select()` with the Pipe `%>%`

Context: You want to find the petal measurements for only the 'virginica' species in the `iris` dataset.
Tasks:

1. Using the pipe (`%>%`), start with the `iris` dataset.
2. First, `filter` the data to include only the rows where `Species` is "virginica".
3. Then, `select` only the `Petal.Length` and `Petal.Width` columns.
4. Store the result in a data frame called `virginica_petals` and display it.

Exercise 6: Basic Grouping with `group_by()` and `summarise()`

Context: You want to calculate the average sepal length for each species in the `iris` dataset.
Tasks:

1. Start with the `iris` dataset.
2. Use `group_by()` to group the data by the `Species` column.
3. Use `summarise()` to calculate the mean of `Sepal.Length` for each species. Name the new summary column `avg_sepal_length`.
4. Store the result in `species_avg_sepal` and display it.

Exercise 7: Using `apply()` on Numeric Data

Context: You want to calculate the column means for only the numeric columns in the `iris` dataset.
Tasks:

1. Create a new data frame `iris_numeric` that contains only the first four (numeric) columns of the `iris` dataset.
2. Use the `apply()` function on `iris_numeric` to calculate the mean of each column. (Hint: The second argument to `apply()` should be `2` to specify columns).
3. Display the resulting vector of means.

Exercise 8: Basic Aggregation with `aggregate()`

Context: You want to find the average horsepower (`hp`) for each cylinder type (`cyl`) in the `mtcars` dataset using a base R function.
Tasks:

1. Use the `aggregate()` function.
2. The first argument should be the column you want to summarize (`mtcars$hp`).
3. The second argument should be a list of grouping variables (in this case, `list(cylinder_type = mtcars$cyl)`).
4. The third argument should be the function to apply (`mean`).
5. Display the result.

Exercise 9: Using the `by()` Function

Context: You want to get a statistical summary (`summary()`) of miles per gallon (`mpg`) for each gear count (`gear`) in the `mtcars` dataset.
Tasks:

1. Use the `by()` function.
2. The first argument should be the data vector to be summarized (`mtcars$mpg`).
3. The second argument should be the grouping factor (`mtcars$gear`).
4. The third argument should be the function to apply (`summary`).
5. Display the result.

Exercise 10: A Simple `dplyr` Chain

Context: Find the average horsepower for 4-cylinder cars that get over 25 mpg.
Tasks:

1. Start with the `mtcars` dataset and the pipe (`%>%`).
2. `filter()` for cars where `cyl` is 4 AND `mpg` is greater than 25.
3. Use `summarise()` to calculate the `mean()` of the `hp` column.
4. Display the result.

Part 2: Complicated Exercises

These exercises require combining multiple functions and thinking through a multi-step analysis. We will use the `airquality` and `starwars` datasets.

Exercise 11: Advanced Filtering and Selection

Context: From the `starwars` dataset, you need to find all non-human characters who are from Tatooine or Naboo and are taller than 100 cm. You are only interested in their name, species, homeworld, and height.
Tasks:

1. Start with the `starwars` dataset.

2. `filter()` the data for characters where `species` is not "Human" AND `homeworld` is either "Tatooine" or "Naboo" (use `%in%`) AND `height` is greater than 100.
3. `select()` only the `name`, `species`, `homeworld`, and `height` columns.
4. Store the result in `specific_characters` and display it.

Exercise 12: Conditional Column Creation with `mutate()` and `case_when()`

Context: You want to categorize the months in the `airquality` dataset into seasons.
Tasks:

1. Start with the `airquality` dataset.
2. Use `mutate()` to create a new column called `Season`.
3. Inside `mutate()`, use `case_when()` to assign seasons based on the `Month` column:
   - Months 6, 7, 8 are "Summer".
   - Months 9, 10 are "Autumn".
   - All other months are "Spring" (since the data only covers May-Sept).
4. Display the `Month` and new `Season` columns for a sample of rows to check your work.

Exercise 13: Advanced Grouping and Summarization

Context: For each species in the `starwars` dataset with at least two members, find the number of characters, the average height, and the heaviest mass.
Tasks:

1. Start with the `starwars` dataset.
2. `group_by()` the `species` column.
3. Use `summarise()` to calculate three new columns:
   - `count = n()` (to count characters in each group).
   - `avg_height = mean(height, na.rm = TRUE)`.
   - `max_mass = max(mass, na.rm = TRUE)`.

4. After the summary, `filter()` the results to keep only the species where `count` is greater than 1.
5. `arrange()` the final result in descending order of `count`.

Exercise 14: Using `apply()` with a Custom Function

Context: You want to calculate the range (max - min) for each of the numeric measurements in the `iris` dataset.
Tasks:

1. Create a numeric matrix `iris_matrix` from the first four columns of `iris`.
2. Write a simple custom function `calculate_range <- function(x) { max(x) - min(x) }`.
3. Use `apply()` on `iris_matrix` to apply your `calculate_range` function to each column.
4. Display the result.

Exercise 15: Advanced `aggregate()` with a Formula

Context: Using the `mtcars` dataset, calculate both the mean and standard deviation of `mpg` and `hp`, grouped by both `cyl` and `gear`.
Tasks:

1. Use the `aggregate()` function with its formula interface.
2. The formula should group `mpg` and `hp` by `cyl` and `gear`. (Hint: `cbind(mpg, hp) ~ cyl + gear`).
3. The data argument is `mtcars`.
4. The function argument should calculate both mean and standard deviation. This is tricky. A common way is to apply a function that returns a vector, like `function(x) c(mean = mean(x), sd = sd(x))`.
5. Display the result.

Exercise 16: Comparing `by()` with a `dplyr` Chain

Context: Find the median horsepower (`hp`) for each number of carburetors (`carb`) in the `mtcars` dataset. Solve this problem in two different ways.
Tasks:

1. **Method 1 (Base R):** Use the `by()` function to apply the `median()` function to the `hp` column, grouped by the `carb` column.
2. **Method 2 (Tidyverse):** Use a `dplyr` chain. Start with `mtcars`, `group_by(carb)`, and then `summarise(median_hp = median(hp))`.
3. Display the results from both methods and observe the difference in their output format.

Exercise 17: Full Data Pipeline - Finding the Hottest Day

Context: In the `airquality` dataset, find the date (Month and Day) with the highest temperature. Also, include the Wind and Ozone values for that day.
Tasks:

1. Start with the `airquality` dataset.
2. Use `arrange()` to sort the entire dataset by `Temp` in descending order.
3. Use `head(1)` to select the top row, which will be the hottest day.
4. `select()` the `Month`, `Day`, `Temp`, `Wind`, and `Ozone` columns.
5. Display the final one-row data frame.

Exercise 18: Full Data Pipeline - Character Analysis

Context: Find the three heaviest characters for each gender in the `starwars` dataset. Exclude characters with unknown mass or gender.
Tasks:

1. Start with the `starwars` dataset.
2. `filter()` out any rows where `mass` or `gender` is `NA`.
3. `group_by()` the `gender` column.
4. `arrange()` the data by `mass` in descending order *within each group*.
5. Use `slice_head(n = 3)` to select the top 3 rows from each gender group.
6. `select()` the `name`, `gender`, and `mass` columns to display the result clearly.

Exercise 19: Combining Base R `aggregate` with `dplyr`

Context: First, calculate the average `mpg` for each `cyl` and `am` (transmission type) combination using `aggregate`. Then, use `dplyr` to rename the columns and filter for only the combinations with an average `mpg` over 20.
Tasks:

1. Use `aggregate()` with the formula `mpg ~ cyl + am` on the `mtcars` dataset with the `mean` function. Store the result in `agg_result`.
2. Convert `agg_result` into a tibble using `as_tibble()`.
3. Use a `dplyr` pipe (`%>%`) on the tibble.
4. Use `rename(avg_mpg = mpg)` to make the column name more descriptive.
5. `filter()` the results to keep only rows where `avg_mpg` is greater than 20.
6. Display the final, cleaned-up summary table.

Exercise 20: Open Problem Solving

Context: A client wants to know the average Body Mass Index (BMI) for the three most common species in the `starwars` dataset. BMI is calculated as mass/(height_in_meters)2.
Tasks:

1. Start with the `starwars` dataset.
2. First, identify the three most common species. (Hint: `count(species, sort = TRUE)` and `head(3)`).
3. `filter()` the original `starwars` dataset to keep only the characters belonging to those top three species.
4. Use `mutate()` to calculate two new columns: `height_m` (height in meters) and `bmi`.
5. `group_by()` species.
6. `summarise()` to find the `avg_bmi = mean(bmi, na.rm = TRUE)`.
7. Display the final result.