Name: _____ PUID _____

STAT 350 Worksheet #1

In this exercise, we will analyze a dataset from a study by Potvin, Lechowicz, and Tardif (1990), which examined how the grass species *Echinochloa crus galli* responds to environmental changes. Data were collected from twelve plants, six from Quebec, Canada, and six from Mississippi, USA. Half of the plants from each location were subjected to an overnight chilling treatment, while the others were left unchilled.
CO2 uptake rates were measured at seven levels of ambient CO2 concentration for each plant, resulting in 84 observations. This dataset provides an opportunity to practice core statistical skills, including working with categorical and numerical data and creating visualizations such as histograms and boxplots.

> Potvin, C., Lechowicz, M. J. and Tardif, S. (1990) "The statistical analysis of ecophysiological response curves obtained from experiments involving repeated measures", *Ecology*, **71**, 1389–1400.

The **CO2** dataset is included in the base R installation, making it readily accessible for analysis.
To load and explore this dataset, run the following R command: `data(package = "datasets", "CO2").`
After running the command, a data variable named **CO2** should appear in your environment (visible in the top-right pane) as a *promise*. A *promise* is a part of R's lazy loading process to delay the evaluation of something until it is needed.

The `help` command provides comprehensive documentation for R's functionality, including datasets, functions, and packages. This includes detailed explanations of variables, their meanings, measurement units (if applicable), and often usage examples. To learn more about the dataset and its structure and properties, use the `help` command as follows: `help(CO2).`

1) Using the information from the help documentation (visible in the bottom-right pane), list all the variables in the dataset and specify the measurement units where applicable (e.g., feet, miles, etc.).

The `View` command in R provides a convenient way to visually explore a dataset's structure and contents in a spreadsheet-like format. This is especially useful for getting an overview of the data, including variable names, data types, and sample values. To open a dataset in the viewer, use the View command as follows:
`View(CO2).`

2) Answer the following questions regarding the data.

- How many observations (rows) are in the CO2 dataset?
- Identify the variables in the dataset and classify them. Specify which variables are **qualitative** (categorical) and which are **quantitative** (numerical).
- For each qualitative variable specify whether the variable's measurement scale is **nominal** or **ordinal**. List all possible values (categories) the variable can take.
- For each quantitative variable specify identify whether the variable's measurement scale is **interval** or **ratio**. Provide a brief explanation for your classification, considering factors such as the presence of a meaningful zero or the type of measurement.

Another useful command to quickly explore **variables** in R is the `table` function. This function generates a summary of the frequency of each category within a variable, providing a straightforward way to understand the distribution of a **qualitative variable**.
To use the `table` function, you need to specify the variable of interest from your dataset by passing it as an argument in the following format: `table(data_frame$variable_name).`
Replace `data_frame` with the name of your dataset (e.g., `CO2`) and `variable_name` with the name of the variable you want to examine. The dollar sign ($) is used to access a specific column (or variable) within a data frame.

3) Run this command on each of the **qualitative variables** and provide the frequency counts for each of the possible categories for each qualitative variable. Also run the `table` command on the **quantitative variables**. Why might one of the quantitative variables exhibit a specific pattern, such as repeated values or fixed intervals? Based on the context of the dataset or the nature of the variable, hypothesize why this pattern occurs.

The primary variable suitable for numerical and graphical exploration is the `uptake` variable, as highlighted in your analysis for question 3. It is important to explore this variable further, examining how its behavior varies across the different values of the other variables. This exploration will provide deeper insights into potential relationships and patterns in the data.

4) Perform a numerical and graphical univariate analysis of the `uptake` variable. Compute and report statistics for the central tendencies and spread of the `uptake` variable. Be sure to use proper notation for each statistic (e.g., $\bar{x}$ for the sample mean). Obtain a **histogram** and a **modified boxplot** for this variable. Describe what you observe regarding the distribution using the histogram and modified boxplot.

In R, functional programming simplifies iterating over data and applying operations with concise and declarative code. For example, `tapply` applies a function to subsets of a vector, grouped by a qualitative variable (see `help(tapply)` for details).

5) Using the `tapply` function in R we can easily compute statistics by groups. Compute the mean and standard deviation of the `uptake` as a function of the `Type` variable, i.e., run the following commands `tapply(CO2$uptake, CO2$Type, mean)` and `tapply(CO2$uptake, CO2$Type, mean)` and save the output as variables. Report your output here.

Now, we are going to graphically visualize the distribution of the sample `uptake` rates as a function of `Type`. First, we will create a side-by-side boxplot.

6) Use the ggplot2 package to plot `uptake` on the y-axis and `Type` on the x-axis, with each box representing one of the two Type categories (Quebec and Mississippi). Label the axes appropriately and ensure the title reflects the purpose of the graph. Describe any differences you observe between the distributional characteristics of `uptake` across the categories of `Type`.

While a boxplot summarizes key statistics like medians and outliers, a histogram reveals the shape of the data, including skewness, multimodality, and distribution of values across its range.

7) Create a histogram of `uptake` as a function of `Type` using ggplot2. First, use `tapply` to calculate the mean and standard deviation of `uptake` for each `Type`. Then, use `ifelse` to compute a new column, `normal.density`, by applying the normal density formula conditionally, based on whether the `Type` is **Quebec** or **Mississippi**. Plot uptake on the x-axis and density on the y-axis, using `facet_wrap` to create separate panels for each `Type`. Overlay the histogram with a **red kernel density curve** and a **blue normal density curve** to compare distributions between Quebec and Mississippi. Describe any differences you observe between the distributional characteristics of `uptake` across the categories of `Type`.

The relationship between `uptake` and `conc` (ambient CO2 concentration) is an important aspect of the dataset, as it reflects how plants respond to varying levels of CO2 in their environment. Unlike `Type`, which has only two categories, `conc` includes multiple distinct categories, representing different experimental conditions. To analyze this relationship, we will create visualizations to explore how `uptake` varies across these values of `conc` and discuss challenges and remedies when plotting data with multiple numerical categories.

8) Create a side-by-side boxplot to visualize the distribution of `uptake` as a function of `conc` using ggplot2. Plot `uptake` on the y-axis and `conc` on the x-axis. What do you observe about the boxplot? Why does this approach not work as intended? Hint: Consider how the `conc` variable is being treated in the plot.

9) Remedy the issue observed by converting the conc variable into a factor:
`CO2$conc <- as.factor(CO2$conc)`.
Recreate the side-by-side boxplot after applying this change. Describe any differences you observe between the distributional characteristics of `uptake` across the categories of `conc`.

Histograms with Density Curves for `conc`: Unlike `Type`, where we could use `ifelse` to compute normal density curves for two categories, working with more than two categories makes this approach cumbersome. Instead, we use the `mapply` function to calculate the normal density for each observation, dynamically referencing the mean and standard deviation for its specific concentration level. First, use `tapply` to calculate the mean and standard deviation of `uptake` for each `conc` value. Next, write a function that computes the normal density for a given `uptake` and `conc` value using statistics (xbar, s).

Using the function you just created in conjunction with the `mapply` function obtain the density values for the normal curve and store it in your data.

```
CO2$normal.density <- mapply(your_function, CO2$uptake, CO2$conc)
```

10) Create a histogram of `uptake` as a function of `conc` using ggplot2. Plot `uptake` on the x-axis and density on the y-axis, using `facet_wrap` to create separate panels for each `Type` and add separate ploting rows `facet_wrap(~ conc, nrow = 2)`. Overlay the histogram with a **red kernel curve** and a **blue normal density curve** to compare distributions across concentration levels. Describe any differences you observe between the distributional characteristics of `uptake` across the categories of `conc`.

For Reference:

`ifelse`

- **Purpose**: Vectorized conditional function that evaluates a condition and returns one value if **TRUE** and another if **FALSE**.
- **Usage** **ifelse(condition, value_if_true, value_if_false)**
- **Why Used**: Straightforward for binary conditions.
- **Alternative**: Tthe **dplyr** library provides more flexibility for multiple conditions using the **case_when** function.

`tapply`

- **Purpose**: Applies a function (e.g., mean, sd) to subsets of a vector grouped by a categorical variable.
- **Usage** **tapply** (**vector, grouping_variable, function**)
- **Why Used**: It is simple and efficient for grouped calculations with categorical variables.
- **Alternative**: In the **dplyr** library, the same task can be achieved using **group_by** and **summarise**.

`mapply`

- **Purpose**: Applies a function to multiple arguments simultaneously.
- **Usage** **mapply** (**function, arg1, arg2, …**)
- **Why Used**: Flexible for dynamic, row-wise calculations involving multiple variables.
- **Alternative**: In libraries like **dplyr** or **purrr** packages can achieve similar results with **mutate** or **map2**.