# Exercise: Visualizing Associations and Mapping Aesthetics

**Overview**

This exercises provides some practice creating data visualizations to communicate associations among variables and is designed to be completed in 75 minutes or fewer. Although you should have time to complete the parts, if you are unable, *as with all exercises, you are encouraged to complete it outside of class time* so that you are able to incorporate your experiences and knowledge into future exercises. You may need to consult course reading materials located at the course site as some elements may not have been covered in the basic content contained in associated videos.

This exercise assumes you have an understanding of `{dplyr}` functions like `select()`, `filter()`, `mutate()`, and `summarize()` and that you have been practicing using these functions as part of your *course allocation time outside of class*. Also assumed is a basic understanding of `{ggplot2}` functions like `ggplot()`, `geom_point()`, as `aes()`.

**This exercise focuses on:**

- Intentional File Creation and File Management
- Creating point plots
- Mapping variables to aesthetics
- Workflow and Reproducibility
- Version control and Git

**This exercise uses:**

- Your RStudio Git version-control project for your team project (presumably completed)
- `{here}`, `{dplyr}`, `{ggplot2}` functions and functions from relevant `Base R` libraries (e.g., `read.csv()`, `readRDS()`, `saveRDS()`, etc.)
- Your knowledge from past in-class exercises, videos, homework, etc.

**Note:** If your RStudio project for your team is not created, you will work in your `dataviz-exercises` project and use a different data set, for example `ggplot2::mpg` or `ggplot2::diamonds`. Do not use class time to set up your team project.

# Part 1: Creating a Point Plot: Mapping a Numeric Variable to Aesthetics

Using a data set, identify some relevant numeric and non-numeric variables. Create a point plot with a numeric variable along the y-axis and a factor/character variable along the x-axis. The character variable could represent a grouping variable from a previous exercise. Map a numeric variable to an aesthetic other than x or y.

1. Create an `.R` script file (Recommendation: Start with `starter_script_file.R`)
2. Name the file appropriately based on the goal of the three parts (review briefly). Add either your name or initials as a prefix to the file name. Do not use spaces.
3. Load your libraries and functions you may need.
4. Write code to read your raw data file using `{here}` paths. Assign the data frame an object name.
5. Write code to create a simple point plot using `geom_point()`. Map a numeric variable to an aesthetic other than x or y.
6. Investigate the plot visually (not quantitatively) to determine whether there are associations or patterns in the data worth noting to investigating more thoroughly. If so, make some notes to reference later.
7. Save your plot script file.

Make sure that your script file is saved in the appropriate project directory so that other team members will know where they are located.

# Part 2: Creating a Point Plot: Mapping a Non-Numeric Variable to Color

Using either the same variables or a different set of variables, create a point plot with a numeric variable along the y-axis and a non-numeric (e.g., factor/character variable along the x-axis). The non-numeric variable could represent a grouping variable from a previous exercise. Map a non-numeric variable to the *color* aesthetic.

1. In a new section of your `.R` script, describe what this new code will do.
2. Write code to create a *new* point plot with a non-numeric variable mapped to the color aesthetic.
3. Re-save your plot script file.
4. Investigate the plot visually (not quantitatively) to determine whether there are associations or patterns in the data worth noting to investigating more thoroughly. If so, make some notes to reference later.
5. Consider whether the plot appears too busy and complicated. Consider other ways to visualize the data. Make a list of suggestions to reference later.

*Hint:* If you have not done so, now may be the time to start determining the official color palette for your organization and decide whether such a palette would be useful for your visualizations; not all colors and color combinations are useful.

# Part 3: Creating a Point Plot: Mapping a Different Non-Numeric Variable to Color

Using the same variables used in Part 2, create a point plot with a numeric variable along the y-axis and a non-numeric variable along the x-axis. You will now map a *different* character/factor variable that contains a different number of unique groups to the *color* aesthetic to explore the differences in the data as well as see the differences in the color palette applied. If you do not have a second variable, filter your data frame to keep or remove a group so that you a different number of groups before you pass it to `ggplot()`.

1. In a new section of your `.R` script, describe what this new code will do.
2. Write code to create a *new* point plot with a new or modified non-numeric variable mapped to the color aesthetic.
3. Re-save your plot script file.
4. Investigate the plot visually (not quantitatively) to determine whether there are associations or patterns in the data worth noting to investigating more thoroughly. If so, make some notes to reference later.
5. Consider whether the plot appears too busy and complicated. Would changing something to the plot make it easier to understand? Consider other ways to visualize the data. Make a list of suggestions to reference later.

# Part 4: Practicing with Git

Remember that your Git commands will be in the terminal and not the RStudio console.

## Checkout your Feature Branch

If you are not working on a feature branch of your team project, get on it now. To verify your branch, use `git branch` in your terminal. If you are on *main*, checkout your feature branch using `git checkout` and then set the upstream.

- `git branch`
- `git checkout <your-branch>`
-

## Staging, Committing, and Pushing to the Remote Repository

1. *Stage* Your File(s)

- Stage your `.R` script file created
- Use `git add <path-to-file-and-file-name>`
- Ensure that you add the correct file(s), including character casing

2. *Commit* Change(s) with a Message

- Commit chang(e) with a clear message
- Use `git commit -m "added script for ...."`

3. *Push* Change(s)

- Push your local repository change to the remote repository on GitHub
- Use `git push` or `git push origin <your-branch-name>` (to ensure you are not pushing from main)

4. Check your Remote Branch on GitHub

- Go to your GitHub account and under **code** toggle to see your branch history.

# Other Practice Ideas

You can also practice mapping and setting aesthetics as the process for all plots created in the future will be identical.

- map other aesthetics to explore the data; set aesthetics to customize the visualization
- create other multiclass scatter plots
- add a `geom_smooth()` layer to a plot that maps variables to x and y only
- add a `geom_smooth()` layer to a plot that maps variables to x, y, and another aesthetic