Exercise: Color Scales and Palettes

Overview

This exercises provides some practice working with functions from **{colorspace}** and **{monochromeR|** and is designed to be completed in 75 minutes or fewer. In particular, you will be tasked with creating visualizations to which you will apply diverging colors, creating your own color palettes for a given number of colors, and examining how a plot looks to one with a form of color-vision deficient or colorblindness. 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 `{ggplot2}` functions like `ggplot()`, `geom_point()`, `geom_bar()`, and as `aes()`. Your ability to work through the exercise will also be influenced in part by your prior practice using these functions as part of your *course allocation time outside of class*.

This exercise focuses on:

- Creating visualizations
- Mapping variables to aesthetics
- Adding color scale layers to plots
- Determining unique instances in vectors

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.)
- Other Libraries: `{colorspace}` and `{monochromeR}`
- Your knowledge from past in-class exercises, videos, homework, etc. and corresponding modules from the course site.

Part 1: Diverging Palettes using {colorspace}

The **{colorspace}** library is one of the most flexible color palette libraries, so familiarizing yourself with its functions would be good to do for those wishing to create visualizations. Using a data set for your project, set up your data frame. in a way that would be appropriate for a *diverging color palette*. Using **{colorspace}**, create two plots based on the data, make one of the plots without a diverging color palette and another with diverging palette You may need to practice your data frame manipulation skills to tackle the problem. Use a palette other than one used in the class module in order to experiment. Diverging palettes are useful to show distance from a reference point. For example, values deviating from a mean value, median value, error measure, or other cutoff score, like an all-time-high, record, or performance cutoff.

For whatever function you use to achieve your goals, you will need to pass an argument to `palette`. Also, your `geom_*()` could require either `col` or `fill` so pay attention to the function naming convention. Be mindful also of the difference between numeric and non-numeric scales as evidenced by `scale_*_continuous_*()` versus `scale_*_discrete_()`. You may experience errors are part of learning to work with the function naming conventions.

Assign each plot to an object of some useful name in order to use in certain parts.

Part 2: Identifying Unique Instances in a Vector

Depending on the mutation of your variable for diverging colors, you may end up with many or few unique values in that vector. Determine how many colors you might need if you wanted to apply colors to each unique value (as if it were discrete) in that vector. Code that below.

Part 3: Diverging Color Palettes for Discrete Scales

When dealing with continuous values, your color scale and corresponding legend will display this effect. Sometimes, however, you will prefer your color palette be discrete rather than continuous. You will run into issues if you try to apply a discrete color scale to your numeric vector. If you don't have too many unique instances in the vector, modify your variable in your data frame so that you can apply a *discrete* complement to the diverging palette in Part 1. If you get stuck thinking about how to modify the variable, ask me for some help. If you have too many unique instances, consider filtering your data so that you can create a *discrete* complement to the diverging palette without it being too busy.

Part 4: Generating Color Palettes

Although some libraries are helpful for color palettes, you may need to generate your own. `monochromeR::generate_palette()` can help you here. **{monochromeR}** also contains functions for converting rgb colors to hex colors, just in case you have rgb colors but no hex version. I would recommend you automate this process and not have to search for matches online.

For these palettes, consider colors that may be relevant for your project, otherwise choose your own colors for this task. You should create the number of colors that you need based on the unique number of instance in your vector that you determined earlier.

```
generate_palette(
  colour,                   # the starting colour for the palette
  modification,             # options to "go_darker", "go_lighter",
                            #     "go_both_ways", or "blend".
  n_colours,                # the number of colors to return
  blend_colour = NULL,      # optional for blending colors
  view_palette = FALSE,     #
  view_labels = TRUE,       #
  ...
)
```

1. Use `monochromeR::generate_palette()` to create a color palette that starts with one of the colors and blends another color and assign the hex colors to a vector object. Pass the number of unique colors you will need are an argument to `n_colors`.
2. Create a second palette that starts with the same color but does not blend. Instead, make it **darker**.
3. Create a third palette that starts with the same color but does not blend. Instead, make it **lighter*.

You can see if the `"go_both_ways"` modification helps with the palette.

Just for Fun

Pass one of your color palettes to `colorspace::swatchplot(my-created-palette, cvd = TRUE)` to see what the colors will look like to someone with a variant of color deficiency.