Instructions
Assignment 3

ICPSR: Data Visualization (Prof. Cooper)

Week 2-3

- Date: June 29 - July 5, 2021 (Due Monday, July 5, 9:00 PM)
- Time: Due July 5th, 2021, 9:00 PM

What should you do in Assignment 3?

- In RStudio, choose "File" > "New File" > "R Script". Then, save the file as "Session05_script.R".
- Work on Exercises below.
- You should upload your R script by 9:00 pm Monday, July 5th to Canvas ("Assignments" > "Assignment 3").

Notes:

- Normally I would ask you to not write or type your answers *within your script*. But feel free to write some comments (with `#`) for your own memo. At this stage I encourage you to write reasonably extensive comments in your own version of your scripts so that you remember what it is you did and why you did it. Scripts to be read by others, under normal circumstances, should not be quite so heavyhanded with comments.
- Never use `T` and `F` as synonyms for `TRUE` and `FALSE`, because `T` and `F` can be redefined and could cause confusion and error. See `https://www.r-bloggers.com/r-tip-avoid-using-t-and-f-as-synonyms-for-true-and-false/`
- Each time I ask you to create a new object, please follow the creation of that object by printing the object. Under normal circumstances you would not print out every object in your R script, whether using proper workflow on your own or sharing scripts with other researchers and collaborators. It will be useful for all of us *in the context of this course*. You can check your work constantly by seeing the output and I can grade more efficiently by seeing the output.

- If I use a particular verb or reference a particular operator or function that you do not immediately recognize, there are many resources out there to help you to discover the solution. The process of search and discovery for basic coding solutions is a part of programming, and **R** has many online resources for you. Google is your friend, along with a very large number of sources for **R** support.

Exercises

Set up the top of your R script with your name, the name of the course, the term, and the date using comments. There are different style guides for comments. For your homeworks, please use `#` (followed by one space) to add a comment to your code. Begin with `##` (followed by one space) for entirely commented lines.

```
> ## Data Visualization, ICPSR Summer 2021
> ## Assignment 3
> ##
> ## Name: YOUR NAME
> ## Date: June 29 - July 5, 2021
```

For this entire assignment you will use three datasets: the World Income Inequality Dataset (WIID), the old IMDB data (`movies_metadata.csv`) and the new IMDB movie ratings datasets (`imdb.tsv` and `ratings.tsv`).

In this assignment you'll start fixing your labels, themes, and colors. For the time being, please use `theme_minimal` as your default theme. Fix all your labels; you can do so within one `labs` line within your plot. When you start using your own colors, use hex codes (for example, go to `www.color-hex.com`) unless I specifically suggest otherwise. There are almost 17,000,000 colors available to you in hex code, so be picky about your colors.

For some of your work today you might have to deal with missing data points. That's life as a researcher. There are many ways to deal with this. For now we are going to remove the "NA" missing data points inside the statistical functions like `mean`, `median`, `min`, or `max` with the `na.rm = TRUE` argument.

1. First, let's work with the IMDB dataset.

   (a) Open up the IMDB dataset and name it `movie`. Check the `head` and `tail` to make sure it loaded correctly.

   (b) Let's plot a histogram. I'd like to look at the differences in IMDB scores of black-and-white films and color films. You'll have to distinguish the levels of the `color` variable with an aesthetic. In `geom_histogram` the default position is stacked, meaning the categories of the variable and their counts will be stacked on top of each other. As you can see, this may not be ideal. We'll fix the `position` argument in the next problem.

   (c) Let's fix the previous histogram. It would be better to make a histogram where there are two separate histograms for the categories of the `color` variable that are not stacked. In addition, because there are significantly fewer black-and-white films, we should present within-category proportions or densities rather than counts. Investigate `geom_histogram` with

the helper function. Specifically, pay attention how to change the default histogram's `position` argument and how to change the default `stat` from counts to proportions/densities. (*Hint: the default count stat would be replaced by a density stat, both of which would represent variation on the y-axis!*)

2. WIID Data.

   (a) Let's try a `density` plot of income inequality (`gini_reported`) in Asia. Separate the density plot into density plots by subregion (using an aesthetic). The title should be something like "Income Inequality in Asia." Fix all labels. Use the colors of your choice for filling in the separate density curves. To see the different density curves more clearly, drop the `alpha` level down to something like 0.2 or 0.3.

   (b) For the UN region Europe, for all years, I want you to plot each country's mean Gini Index score. This should be one dot per country. I would like you to make a scatterplot of country on one axis (thus, a discrete axis) and its mean Gini Index score (a continuous variable) on the other. I would also like you to flip the axes, reorder the values, and suppress/remove the axis title for country. This should produce a pretty cool, simple, clean, and modern graphic. You may also need to reduce the size of the axis tick labels in a `theme` layer.

   (c) Let's make another interesting plot. Subset to Africa. Let's make a bar plot of the difference between each country's average Gini Index score and the average Gini index score of the entire continent. Flip the axes again. Color the bars with color hex codes of your choice.

   (d) Make a histogram of Gini index values of all countries and years. Set the color aesthetic of the histogram by UN region. Drop `alpha` to 0.6 as well. Create your own qualitative color palette using hex codes.

   (e) I want to see the median Gini index score by year for each of the subregions of the Americas (UN Regions). Start with a scatter plot and make it pretty transparent, adding a color aesthetic for sub-region. Then also show me a smoothed line of the same information with plot with a separate line colored for each subregion. For a bonus, add a light gray line that averages over all the other lines. Adjust colors and alpha to where it looks nice. Use hex code colors.

3. IMDB Data Again: Some Counting and a New Filtering Option

   (a) Show me all the different languages represented in the `movie` dataset with the `distinct` function.

   (b) Count the the movies (using `count`) by `language`. Sort them, please, with the `arrange` function.

   (c) Count the movies both by `language and color`. Sort them by both variables with `arrange`, please.

(d) I am going to introduce a function to you for this problem, the logical operator `%in%`. It determines whether the elements of the first vector are also contained in the second vector. To practice with it, make a vector called `vector1` that gets values 1 through 10 in sequence. Then create a `vector2` that gets the values 1, 4, and 11.

(e) To practice and understand the `%in%`, type `vector1 %in% vector2` and see what R prints. Then, do it in reverse, typing `vector2 %in% vector1` and printing it. Pay attention to the difference; this is how you understand functions.

(f) Obviously, most of the movies in this database are in English. Boring! Using a plot/picture and not words, count the number of films in French, Spanish, Hindi, and Mandarin. To do so, use the `%in%` operator to filter for those lanaguages only and then provide a bar plot.

4. Pulling Apart and Creating String/Character Variables (Old IMDB Data Again)

(a) In the next few problems I am inviting you to use the `stringr` package inside `tidyverse`. This package helps you work with character or string variables to find patterns that you might want to highlight, analyze, subset, or change altogether. There is an impressive set of functions inside `stringr` for these purposes, pretty much all set up in a `str_SOMETHING()` format, where `SOMETHING` is replaced with a number of verbs and adjectives. These functions work to apply rules and find patterns *inside* vectors. Download the **RStudio** cheatsheet on `stringr` if you have not yet done so. Just download the cheatsheet and get free points!

(b) First, let's go back to our `count` function. I want to know if different genres have different average IMDB scores. To start, Let's just count the different genres in the dataset with the `count` function.

(c) Wow, that's messy. It makes sense that a movie might be in multiple categories, but the coding done here is kind of ridiculous. Let's find which movies are considered action movies. Use `str_detect()` somehow to create a new variable `action` and then count the action versus non-action movies. There's a really useful function called `ifelse` you might consider, although it is not necessary. You could also try `case_when`. Once you have the new variable, perform a count.

(d) Perform another `count` like above, but do so for animated movies.

(e) I want to know whether comedies are rated higher than dramas on average in their IMDB scores over time. More generally, I want to know how the IMDB scores for the average comedy or drama evolve by year. Keep in mind that some movies will be considered both, and that is no problem. I suggest you simply have a "Comedy-Drama" category for those. Please show me with a fairly transparent scatterplot underneath layer and with

a smoother line on top by category. Use your own hex code colors. Move the legend to the bottom of the panel. Fix all labels.

5. The IMDB database we have been using is a fairly small sample of what is currently available. For this fifth series of problems you will be using the new, updated data files I have provided you. They are `imdb.tsv` and `ratings.tsv`. These files are significantly larger than the old IMDB database!

   (a) This IMDB database has *millions* of observations and is more representative of the broader array of options available to consumers. I have provided a couple of new and rather large datasets for you that have been freshly pulled from IMDB. Read in the `imdb.tsv` and `ratings.tsv` files from your Canvas Files/Assignments/Week 4 Module. The `.tsv` files read just like `.csv` files; the `readr` command for importing these files is `read_tsv()`. Just for your information, `tsv` stands for "tab-separated values," while `csv` stands for "comma-separated values."

   (b) Let's start with the `imdb.tsv` file. I wonder if the genres variable has improved since a few years ago. Let's investigate. Use a simple `count` function to do so.

   (c) We are going to need to `join` the two datasets together to be able to get what we want for analysis and visualization. A suite of these functions exists in `dplyr`. I would start with `full_join`. Use the helper file to investigate; all you need is one ID variable from both data sets on which to match. Join the `ratings.tsv` and `imdb.tsv` datasets using the film ID in the `by` argument.

   (d) There seem to be many observations that have no average IMDB scores (now called `AverageScore`). I think we should look to see why. Use the `is.na` function somehow to figure this out. Specifically, let's try investigating the variable `titleType`. Show me how many missing values exist within each title type.

   (e) Ahh. Most, though certainly not all, of the missing IMDB scores are from TV episodes. For the final plot with these data you will subset to films with IMDB scores. Let's create a line plot comparing scores of two of the title types, full-length movies and"shorts," films shorter than 40 minutes including credits. Instead of working with means, however, let's make the underneath layer is a bit different. Please estimate the following quantities: the median IMDB score (don't be confused that the IMDB score is also an "average" score across individuals who vote), the 25th percentile (also known as the `quantile` at .25), and the 75th percentile (also known as the `quantile` at .75). Create these in one mutate function.

      i. Use a geom called a `geom_segment` for your underneath layer and a smoother line as your top layer for the plot. Remember, we are comparing full-length films against shorts. The underneath segments should be vertical bars for each year, ranging from the 25th to the

75th percentile, representing the IQR (interquartile range) for each year. Set the alpha parameter to about 0.4 for the segments.

ii. The top layer of the plot should be smoother lines. Please shrink down the `span` of the smoother lines to about 0.3.

iii. Set one color, for short films, to a relatively dark, mustard-like yellow using hex codes. Pick a good medium-intensity blue for full-length films.

iv. Move the legend inside the plot to negative space on the bottom right of the plotting panel. Fix all labels.

6. One last WIID problem. (Somewhat tricky!)

   (a) For this problem, I want to know how each UN region's individual Gini scores (all the observations in one region) compares to the total distribution of observed Gini scores in the dataset (all the observations in all the regions). Please facet a scatterplot of individual Gini index scores by UN region. Cut out the years where only a few countries have observations; perhaps filter for all years after 1940. Change all labels to look nice and professional. Use the `theme_minimal` again. Here's a slightly tricky addition: add ALL the points (all regions) to each facet (the region-specific points) *underneath* in a light gray color. Reduce the size of the points a bit, and potentially add the slightest bit of transparency. Adjust the regional colors so they stand out from the gray, with different hues and relatively high intensity.