Computing Assignment 3

Due Feburary 17, before class.

A Prerequisite

You should have done the following as part of Computing Assignment 2, but just in case, I'm including it here. I strongly recommend that your capitalization, spelling, and punctuation match mine, so if it doesn't, now would be a good time to fix it. For example, `POLS 209` is different from `pols-209` and `Data` is different from `data`.

1. Create a `pols-209` folder on your computer, this can be wherever you like (e.g., Desktop, Dropbox).
2. In that folder, create a `data` subfolder.

Go to the course webpage and save the `state-legislators` data file to your `data` folder. You may choose whichever file type you prefer. This data set contains an ideology score for each member of each state House of Representatives from 1993 to 2014. It also contains variables that indicate the year, the state, and the legislator's party.

The Assignment

Write an R script that does the following, thoroughly commenting your code along the way:

1. Completes any prerequisites for the actions below, such as setting the working directory, loading needed packages, and loading the `state-legislators` data.
2. Uses `group_by()` and `summarize()` to calculate the average `ideology_score` within each combination of year, state, and party. (The grouping variables are `year`, `state`, and `party`.)
3. Uses `ggplot()` to create **a single figure**–a line plot with the year along the x-axis and the average ideology score along the y-axis. Includes the following modifications:
   - Uses `group = state` to link the variable `state` with the group aesthetic. `group` works similarly to `color`–it creates separate lines for each state in the data frame–but the lines will all be the same color. There are simply too many states to make a color legend useful.
   - Applies a facet by party, producing three separate lines plots within the same figure. Each of the three figures represents one political party.
   - Supplies an `alpha` argument to `geom_line()` so that the lines do not completely obscure each other. Experiment a bit with values between 0 and 1 to find the value that works best. I usually start with `alpha = 0.5` and adjust from there.
   - Improves the labels for the x-axis and y-axis (using `labs()`).
   - Includes nice title, subtitle, and caption (also using `labs()`). You choose. Be creative.
   - Uses a different theme than the default (again, be creative). Experiment with at least a couple before settling.
4. In a comment, write 3-5 sentences discussing **one** interesting pattern in shown in the figure. You should experiment with highlighting your entire comment and clicking *Code*, *Reflow Comment*.

Note: Once you compile your script into a notebook (see below), your figure dimmensions might be off (i.e., your figure might seem too tall). That's okay. RStudio tries to choose a good height and width, but it sometimes chooses poorly. For your papers, you'll want beautiful figures, but for this assignment, the default figure is fine.

Once the script is written, you should **save it** to a convenient spot on your computer. (I suggest a folder `R` inside your `pols-209` folder.) Remember that you'll be writing several scripts this semester, so keep them organized.

Comments on Grading

We grade the assignments using the following rubric:

1. Specification (40%): The code correctly performs all desired actions.
2. Comments (40%): The code is thoroughly and neatly commented.
3. Readability (20%): The code is neatly written and includes appropriate use of white space to make the code easily readable.

Here are some other suggestions:

- We recommend that you use comments to number the questions. This helps us understand what question you are trying to answer when the code differs from our expectation.
- You should usually exclude unnecessary code. For example, you should avoid loading unneeded packages and including lines of code that performs actions the assignment does not ask for.

Submitting Your Work

In case of technical difficulties, I don't want you to spend a lot of time figuring out how to submit your work. If you can't figure it out, just bring a hard copy to class. We'll sit down and work through the process so it's smooth and easy next time.

1. With your R script open, click "File", "Compile Notebook..." Or just click the little white notebook icon.
2. Under "Notebook Output Format," select MS Word. MS Word is the best choice for 95% of you, but HTML or PDF occassionally work better.
3. In a web browser, go to the eCampus page for POLS 209.
4. Click "Submit Computing Assignments" under "Course Materials." Click "Computing Assignment 3."
5. To the right of "Attach File," click "Browse My Computer."
6. Navigate to the file containing your R script and you'll find a file with the same name but the extension ".docx" (if you chose MS Word) rather than ".R". Select the ".docx" file.
7. Click "Submit." If this doesn't work, see footnote 1.

I expect you to submit the assignment on eCampus *before* class. However, I have given you until noon in case you encounter technical difficulties, but see footnote 1.

Troubleshooting

If your code does work when you run it, but does not work when compiling it into a notebook, please check the following:

1. You set your working directory *in the script*. To do this, simply set the working directory to `pols-209` via point-and-click (by clicking *Session*, *Set Working Directory*, *Choose Directory...*). R runs a command in the console, something like `setwd("~/Dropbox/classes/pols-209")`. Copy this command onto the top of your script.
2. You load all necessary packages *in the script*. RStudio begins a new R session to compile a notebook, so even if you have all packages loaded in your current session, compiling a notebook will fail if all necessary packages are not loaded in the script.

I expect you to submit the assignment on eCampus *before* class. However, I have given you until noon in case you encounter technical difficulties.