Problem Set 4

Overview:

In this problem set you will be practicing project organization and workflow by creating a private repository with your homework groups. We will not be asking for your git commands and output– you do not have to paste your git commands in R markdown and can simply use your command line interface. We will be using the **rscorecard** packing in R to practice working with APIs, manipulating data, and creating plots using **ggplot**. Additionally, we will ask you to create a pull request and assign a person in your group to review your work and offer suggestions. Once you have made the suggested revisions, the person you assigned as a reviewer will review and accept your changes and merge to the master branch. We encourage you all to communicate with your group by creating issues on your shared repository.

Part I: Command Line & Organizating Project/Script

1. Have one member of your group create a new private repository on GitHub here and name it `<team_name>_ps4`. Under the `Add .gitignore` option, select `R`, and then click `Create repository`.

   Invite the other group members as collaborators under `Settings` > `Manage access` > `Invite teams or people`. All members will clone this repository to their local machines.

2. Create a new RStudio project, setting this repository directory as the project working directory. Note that your R Console and RStudio Terminal will start up in this directory now.

   In your RStudio Terminal, check `git status`. You will see that an `.Rproj` file has been generated. For the purposes of this class, you can add `.Rproj` to `.gitignore`.

   Have a second member of your group add this to `.gitignore`, then add/commit this change and push to the remote. All other members will pull this change.

3. You can begin working individually now. Remember it is good practice to work on a branch.

   Create a new branch called `dev_[initials]` and switch to it.

4. In your project directory, create a directory called `plots`.

   Inside `plots`, create another sub-directory called `<last_name>_plots`. This sub-directory will contain any plots you create.

5. In your project directory, create another directory called `data`. This will contain any data files you save.

6. In your project directory, create a file called `<last_name>_script.R`.

   Open this file in RStudio to edit. Use the problem set template provided here.

7. In your R script, create 2 variables `data_dir` and `plot_dir` that will store the file paths to the respective directories. Remember to write the paths relative to the project directory.

8. Next, install (if necessary) and import the following packages in your R script: `tidyverse` and `rscorecard`

Part II: College Scorecard API

1. You will be using the rscorecard package to fetch data from the U.S. Department of Education College Scorecard API.

   College Scorecard collects institution-level data on various post-secondary institutions. Spend some time reading through the documentation here.

2. Read through the README file for `rscorecard` here and obtain an API key from here. Follow the example and use `sc_key()` to save your key at the top of your R script.

   Try copying and running the example code from the README file to make sure you are able to request data and access the data dictionary.

3. Spend some time reading about the commands in `rscorecard` here and looking at some examples here.

4. You can also download the data dictionary here. The variables available for this dataset are listed under the `VARIABLE NAME` column in the `institution_data_dictionary` tab. You must turn the variable name to lowercase before using it in the `rscorecard` functions.

   Alternatively, you can use `sc_dict()` to search for variable names. Run `?sc_dict` in your R Console to see more information. For example, you can specify if you want to search the `variable` or `description` column using the `search_col` argument.

5. In `<last_name>_script.R`, fetch the following data and store it inside the variable `full_df`:

   - Filter
     - Include only institutions that predominantly offers bachelor's degree
     - Do not include any service schools or institutions located outside the 50 states (ie. exclude schools whose `region` is classified as `U.S. Service Schools` or `Outlying Areas`)
   - Select
     - Unitid
     - Institution name
     - Institution control
     - Institution state
     - In-state tuition and fees
     - Out-of-state tuition and fees
     - Median earnings of students working and not enrolled 10 years after entry

6. Perform some further data manipulations and save the resulting dataframe in `df`:

   - Add a column called `school_type` that is `1` if the institution is public and `2` if it is private
   - Add a column called `tuitionfee_diff` that is the difference between in-state and out-of-state tuition
   - Add a column called `tuitionfee_diff_pct` that is the percentage increase between the in-state and out-of-state tuition (don't have to multiply by 100)

7. Use `saveRDS()` to save your `df` in a file called `<last_name>_rscorecard.RDS` in the `data_dir`. Add this data file and commit with the message `add rscorecard data`.

Part III: ggplot

1. Still on the `dev_[initials]` branch, use `full_df` to create the following scatterplot in `<last_name>_script.R`:

   - X-axis: Out-of-state tuition
   - Y-axis: Median earnings
   - Color: School type
   - Add smoothed prediction lines for each school type

- Label your graph

   Save your plot as `out_of_state_tuition_earnings.png` in the `plot_dir`. Add this plot and commit with the message `add scatterplot (out-of-state tuition vs earnings)`.

2. Underneath the code for your previous graph, create the same scatterplot but this time use the in-state tuition instead of out-of-state.

   Save your plot as `in_state_tuition_earnings.png` in the `plot_dir`. Add this plot and commit with the message `add scatterplot (in-state tuition vs earnings)`.

3. Now, filter your `df` to include only public universities for creating the following bar plot:

   - X-axis: State
   - Y-axis: Average `tuitionfee_diff_pct` for public universities in that state
   - Label your graph

   Save your plot as `diff_in_tuition_by_state.png` in the `plot_dir`. Add this plot and commit with the message `add barplot (diff in tuition)`.

4. Switch back to the `master` branch and merge in your data file and plots from `dev_[initials]`. Push to the remote repository. You may need to pull first if your group members pushed their work to `master` first.

   Delete the `dev_[initials]` branch.

Part IV: Pull request

1. Create a new local branch called `<last_name>_plot` off the `master` branch and switch to it.

2. Use the `rscorecard` package to pull any data you'd like. Save the data in a file called `<last_name>_custom.RDS` in the `data_dir`.

   Add the data file and commit with the message `add custom data`.

3. Using that data, create a plot of your choice. Save your plot as `<last_name>_custom.png` in the `plot_dir`.

   Add this plot and commit with the message `add custom plot`.

4. Push your `<last_name>_plot` branch to the remote. Remember you'll need to set the upstream branch during this initial push.

5. On GitHub, create a pull request to have the changes from your `<last_name>_plot` branch merged to `master`. Assign one of your group members as the reviewer. Do not assign the person who has assigned you as their reviewer, so that everyone will get a chance to review for someone else.

6. When you are assigned as reviewer by one of your group members, navigate to their pull request and take a look at their plot.

   Under the `Files` tab, click on `Review changes` and select the `Request changes` option. Leave a comment stating what you'd like to see changed about the plot, then click `Submit review`.

7. Once you receive feedback on your own plot, make the requested changes and push your revised plot to the branch. In your commit message, you can reference the pull request by including `#<pull_request_number>`.

   Re-request a review from your reviewer.

8. Finally, once you receive a re-request for your review on your group member's plot, make sure to approve their new changes, then click on `Merge pull request` to merge the changes into `master`.

Part V: I got issues

1. Navigate to the issues tab for the **rclass2** repository here.

   You can either:

   - Create a new issue posting a question you have about the class/problem set (assign instructors)
   - Answer a question that another student posted
   - Create a new issue posting about something new you learned or figured out from this class
     - If you choose this option, please mention the other members of your team and assign yourself

   Paste the link to the issue you contributed to as a comment in `<last_name>_script.R`.

   Please make sure to close the issue once your question has been resolved or within 1 week.

Part VI: Wrapping up

1. Finally, create another branch `<last_name>_script` and switch to it. Add and commit your R script (`<last_name>_script.R`) and push this branch to the remote repository.
2. How much time did you spend on this problem set?