Problem Set 3

Overview:

In this problem set we will be practicing more git/GitHub workflow basics, manipulating data in R, creating plots using **ggplot**, working with git branches and merging, and creating GitHub issues. We are asking you to create a git repository on your local computer which you will later connect to a remote repository on GitHub. This local repository will have an **.R** file where you will read in data, clean and manipulate this data, and use **ggplot** to create plots (e.g. scatterplots and bar plots). You will save your plots in a **plots** folder you create inside your repository. We will practice branching and merging by making changes to the plots you create in part II and III using different branches. We will go through the steps of staging our changes, committing, and merging our branches. You may encounter merge conflicts when collaborating on repositories in this class or outside of this class. For that reason, we will ask you to resolve a merge conflict.

This problem set is longer than previous problem sets, however, you will notice there is a lot of overlap between the problem sets. We strongly encourage you to work with your group and/or reach out when you are encountering issues. Post questions related to this problem set on the rclass repo issues tab and assign the instructors @ozanj, @mpatricia01, @cyouh95.

Part I: Command line & Git/GitHub

1. You know the drill! Using your command line interface (CLI) (e.g. Git Bash, terminal), create a new folder called **lastname_ps3**. Change directory into the **lastname_ps3** folder and initiate it as a git repository.

   Write the commands you used here:

2. Use the `echo` command to output the text `"# YOUR NAME HERE"` and redirect it using `>` to a file called `problemset3.R`. Add this file and make a commit.

   Write the commands you used here:

3. Log in to your GitHub account online and create a new private repository here: https://github.com/organizations/Rucla-ed/repositories/new

   Name it **lastname_ps3** and do NOT initialize it with a `README.md` file. Paste the link to your repository here:

4. Add your newly created repository as a remote for your local **lastname_ps3** repository. Name the remote repo `remote_ps3` rather than `origin`. Write the command you used here:

5. List out the connected remote. Use the option that will display both the remote name and URL. Write the command you used here:

   Copy the output of this command here:

6. If you try pushing your changes with just `git push`, why will you get an error?

7. Write the command to properly push your changes to the remote:

8. Lastly, create a sub-directory called `plots` via the command line. Write the command you used here:

Part II: Branches & Scatterplots

1. Create a new branch called `dev` and switch to it. Write the command(s) you used:

2. List out all your branches (local & remote) as well as details on latest commits. Write the command you used:

   Copy the output of this command here:

   What does the `*` indicate?

3. Over the next few sections, you will begin writing R code for data manipulation and creating plots using `ggplot`. Please make sure all your R code goes in `problemset3.R`.

   Open `problemset3.R` in RStudio and load in the following data on off-campus recruiting events by public universities:

   ```
   load(url("https://github.com/Rucla-ed/rclass2/raw/master/_data/recruiting/recruit_school_somevars.R
   ```

   Each observation (row) in the data is a high school. The columns are various characteristics of the high school. There are also columns indicating the number of times the high school has been visited by a public university:

   - `visits_by_100751` = University of Alabama
   - `visits_by_126614` = University of Colorado Boulder
   - `visits_by_110635` = UC Berkeley

4. Perform the following data manipulations and save the resulting dataframe in an object to use later:

   - Create a 0/1 dummy variable called `visited` that indicates whether the high school received a visit from the university of your choice (`0`=received no visits, `1`=received 1 or more visits)
   - Filter observations to keep only high schools that are located in the same state as the university (hint: see `state_code` for high school state code and `inst_[univ]` for university state code)
   - Subset your dataframe to include the following variables: `school_type`, `ncessch`, `name`, `total_students`, `avgmedian_inc_2564`, `visits_by_[univ]`, `visited`

   (*Note: This is the same data manipulation you did in Part III of Problem Set 2. Feel free to copy your code over from there.*)

5. Use the dataframe from the previous step to create a scatterplot of total enrollment by median household income.

   - X-axis: `total_students`
   - Y-axis: `avgmedian_inc_2564`
   - Label the axes

Export your plot as a PNG named `scatterplot_grayscale.png` and save it in the `plots` folder.

Add `problemset3.R` and your plot, then make a commit with the message `scatterplot 1`.

6. Update your code from the previous step to have different point colors based on whether the high school has been visited or not by the university.
   - Color: `visited`
   - Label the legend (hint: use `scale_color_discrete()`)

   Export your plot as a PNG named `scatterplot_color.png` and save it in the `plots` folder.

   Add `problemset3.R` and your plot, then make a commit with the message `scatterplot 2`.

7. Update your code from the previous step to have 2 subgraphs, one for public HS and one for private HS. In other words, facet your scatterplot by school type.
   - Facet: `school_type`

   Export your plot as a PNG named `scatterplot_facet.png` and save it in the `plots` folder.

   Add `problemset3.R` and your plot, then make a commit with the message `scatterplot 3`.

8. Switch back to the `master` branch. Then, merge in the changes from `dev` to `master`.

   Write the commands you used:

   What type of merge is this?

9. Check the commit log. Write the command you used:

   Note that the commits made on the `dev` branch are now part of the commit history on the `master` branch.

Part III: Bar charts

1. Switch back to the `dev` branch. Write the command(s) you used:

2. You'll now continue to edit `problemset3.R` for the next few steps. Use the `df_school` dataframe to create a new dataframe `df_[region]` filtering for states in one US region and create a bar chart displaying number of high schools in each state. Use `geom_bar()`.
   - Create a new dataframe df_[region] filtering for a region of the US.
     - Midwest = state_code %in% c("OH", "MI", "IN", "WI", "IL", "MN", "IA", "MO", "ND", "SD", "NE", "KS")
     - South = state_code %in% c("DE", "MD", "VA", "WV", "KY", "NC", "SC", "TN", GA, FL, "AL", "MS", "AR", "LA", "TX", "OK")
     - West = state_code %in% c("MT", "ID", "WY", "CO", "NM", "AZ", "UT", "NV", "CA", "OR", "WA", "AK", "HI")
     - Northeast = state_code %in% c("ME", "NH", "VT", "MA", "RI", "CT", "NY", "NJ", "PA")
   - X-axis: `state_code`
   - Y-axis label: Number of HS

- Label the axes

Export your plot as a PNG named `barchart_geombar.png` and save it in the `plots` folder.

Add `problemset3.R` and your plot, then make a commit with the message `bar chart 1`.

3. Now, recreate the bar chart in the previous step but use the `geom_col()` function.

   Export your plot as a PNG named `barchart_geomcol.png` and save it in the `plots` folder.

   Add `problemset3.R` and your plot, then make a commit with the message `bar chart 2`.

4. Update your code from the previous step to have the bar chart include only high schools that received at least 1 visit from any of the 3 universities.

   Export your plot as a PNG named `barchart_visited.png` and save it in the `plots` folder.

   Add `problemset3.R` and your plot, then make a commit with the message `bar chart 3`.

5. Switch back to the `master` branch and merge in `dev`. Write the commands you used:

6. Check the commit log and find the hash of the last commit. Use the `git cat-file` command to print out the contents of that commit object. Write the command you used:

   Copy the content of the commit object here:

7. Based on the previous output, locate the hash of the `tree` object. Use the `git cat-file` command to print out the contents of that object. Write the command you used:

   Copy the content of the tree object here:

8. Based on the previous output, what is the hash of the blob object for the `problemset3.R` file? Use the `git cat-file` command to print out the contents of that object. Write the command you used (no need to copy output):

   What is the hash of the tree object for the `plots` directory? Use the `git cat-file` command to print out the contents of that object. Write the command you used:

   Copy the content of the tree object here:

9. Going back to the content of the commit object in question 6, locate the hash of the `parent` commit object. Use the `git cat-file` command to print out the contents of that object. Write the command you used:

   Copy the content of the parent commit object here:

10. Repeat steps 7 to 9 for the parent commit found in the previous step and write your responses below. Understand the differences between the outputs here and the outputs you got from steps 7 to 9.

    Command to view `tree` object of parent commit:

    Output:

    Command to view blob object for `problemset3.R` file (no need to copy output):

    Command to view tree object for `plots` directory:

    Output:

    Command to view commit object for `parent` commit:

    Output:

Part IV: Resolving merge conflicts

1. Switch back to the `dev` branch. Write the command(s) you used:

2. Go back to `problemset3.R` and modify your code for the scatterplot from Part II, Q7 to overlay smooth prediction lines.

   Export your plot as a PNG named `dev_scatterplot.png` and save it in the `plots` folder.

   Add `problemset3.R` and your plot, then make a commit with the message `modify scatterplot on dev`.

3. Next, modify your code for the `geom_col()` bar chart from Part III to include only high schools that received at least 1 visit from either the University of Alabama or the University of Colorado Boulder.

   Export your plot as a PNG named `dev_barchart.png` and save it in the `plots` folder.

   Add `problemset3.R` and your plot, then make a commit with the message `modify bar chart on dev`.

4. Switch back to the `master` branch. Write the command you used:

5. Now, on the `master` branch, modify your code for the `geom_col()` bar chart from Part III to include only high schools that received at least 1 visit from either the UC Berkeley or the University of Colorado Boulder.

   Export your plot as a PNG named `master_barchart.png` and save it in the `plots` folder.

   Add `problemset3.R` and your plot, then make a commit with the message `modify bar chart on master`.

6. Merge in the changes from `dev` to `master`. Write the command you used:

What type of merge is this?

7. Uh oh, you've run into a merge conflict! But don't panic. Start by running `git status` to confirm that the conflict arised in `problemset3.R`.

   Use `cat` to print out the contents of `problemset3.R`. Notice that the code for adding smooth prediction lines to the scatterplot was able to merge in just fine. Only the code modifying the bar chart is marked as conflicted because it is modified on both branches.

   Copy the conflicted lines (as indicated by Git's markers) here:

8. Maybe you're not ready to incorporate all changes to the `master` branch just yet. What is the command to abort the merge and return the branches back to their original states? Run the command and write it here:

9. Phew! But you decide you still want to combine changes from both branches at some point. So let's try the merge again, but this time on the `dev` branch.

   Switch to the `dev` branch and merge in the `master` branch. Write the commands you used:

10. You run into the same merge conflict, but this time, let's try resolving the conflict. Open up `problemset3.R` in RStudio and delete the markers that Git added. You decide to go with the `master` branch version of the line that filtered high schools based on visits from UC Berkeley and the University of Colorado Boulder.

    After you finish resolving the conflicts, add the R script and make a commit with the message `merge dev and master`. Write the commands you used:

11. Check the commit log. Note that the commit made on the `master` branch is now part of the commit history on the `dev` branch.

    Use the `git cat-file` command to print out the contents of the commit object for the commit whose message is `modify bar chart on master`. Find the hash of the parent commit and look for it in the commit log. What is the commit message of the parent commit?

    Note that the parent commit in this case is not just the previous commit in the commit log. Why is that?

12. Lastly, push the `dev` branch to the remote. Don't forget to set the upstream branch during this initial push.

Part V: I got issues

1. Navigate to the issues tab for the **rclass2** repository here: https://github.com/Rucla-ed/rclass2/issues

   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 here:

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

Part VI: Wrapping up

1. Finally, add and commit this file you are working on (`problemset3.Rmd`) to your repository (`master` branch) and push to the remote repository.
2. How much time did you spend on this problem set?