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 (3 pts)

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:

   ```
   mkdir lastname_ps3
   cd lastname_ps3
   git init
   ```

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:

   ```
   echo "# YOUR NAME HERE" > problemset3.R
   git add problemset3.R
   git commit -m "initial commit"
   ```

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:

   `https://github.com/Rucla-ed/lastname_ps3`

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:

   ```
   git remote add remote_ps3 git@github.com:Rucla-ed/lastname_ps3.git
   ```

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

   ```
   git remote -v
   ```

   Copy the output of this command here:

   ```
   remote_ps3  git@github.com:Rucla-ed/lastname_ps3.git (fetch)
   remote_ps3  git@github.com:Rucla-ed/lastname_ps3.git (push)
   ```

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

   ```
   We get an error because this is a new local branch that we're trying to push to the remote. We need
   ```

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

   ```
   git push --set-upstream remote_ps3 master
   ```

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

   ```
   mkdir plots
   ```

Part II: Branches & Scatterplots (4 pts)

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

   ```
   git checkout -b dev

   # OR

   git branch dev
   git checkout dev
   ```

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

   ```
   git branch -av
   ```

   Copy the output of this command here:

   ```
   * dev                        9b65500 initial commit
     master                     9b65500 initial commit
     remotes/remote_ps3/master 9b65500 initial commit
   ```

   What does the `*` indicate?

   ```
   It indicates which branch you're currently on.
   ```

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`. For how to export your plots, please refer to Section 4 in the lecture - you can use either method.

   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.*)

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


#head(df_school)
#names(df_school)
#glimpse(df_school)

sapply(df_school, function(x) sum(is.na(x))) #function to count missing obs
```

```
##         state_code        school_type            ncessch               name
##                  0                  0                  0                  0
##            address               city           zip_code          pct_white
##                  0                  0                  0                  0
##          pct_black       pct_hispanic          pct_asian     pct_amerindian
##                  0                  0                  0                  0
##          pct_other       num_fr_lunch     total_students      num_took_math
##                  0                158                  0               4103
##      num_prof_math       num_took_rla       num_prof_rla avgmedian_inc_2564
##               4251               4099               4219                624
##   visits_by_110635   visits_by_126614   visits_by_100751        inst_110635
##                  0                  0                  0                  0
##        inst_126614        inst_100751
##                  0                  0
```

```
df_alabama <- df_school %>%
    mutate(visited = ifelse(visits_by_100751 > 0, 1, 0)) %>%
    filter(state_code == inst_100751) %>%
    select(school_type, ncessch, name, total_students, avgmedian_inc_2564, visits_by_100751, visited)
```

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 (hint: use `xlab()` and `ylab()`)

    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`.

```
png('plots/scatterplot_grayscale.png')
ggplot(data = df_alabama, aes(x = total_students, y = avgmedian_inc_2564)) +
    geom_point() +
    xlab("Total enrollment") + ylab("Median average income")
```

```
## Warning: Removed 21 rows containing missing values (geom_point).
```

```
dev.off()
```

```
## pdf
##   2
```

6. Underneath the code for your previous graph, create another scatterplot. This one should look the same as the previous one, except the points are now colored based on whether the high school has been visited or not by the university.

   - X-axis: `total_students`
   - Y-axis: `avgmedian_inc_2564`
   - Color: `visited` (hint: don't forget to wrap the variable in `as.factor()`)
   - Label the axes
   - 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`.

```
png('plots/scatterplot_color.png')
ggplot(data = df_alabama, aes(x = total_students, y = avgmedian_inc_2564, color = as.factor(visited))) +
    geom_point() +
    xlab("Total enrollment") + ylab("Median average income") +
    scale_color_discrete(name = "Recruitment Visits", labels = c("No visits", "Visits"))
```

```
## Warning: Removed 21 rows containing missing values (geom_point).
```

```
dev.off()
```

```
## pdf
##   2
```

7. Underneath the code for your previous graph, create yet another scatterplot. This one should look similar to the previous one, except there should now be 2 separate subplots, one for public HS and one for private HS. In other words, facet your scatterplot by school type.

   - X-axis: `total_students`
   - Y-axis: `avgmedian_inc_2564`
   - Color: `visited`
   - Label the axes
   - Label the legend
   - 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`.

```
png('plots/scatterplot_facet.png')
ggplot(data = df_alabama, aes(x = total_students, y = avgmedian_inc_2564, color = as.factor(visited))) +
    geom_point() +
    xlab("Total enrollment") + ylab("Median average income") +
    scale_color_discrete(name = "Recruitment Visits", labels = c("No visits", "Visits")) +
    facet_wrap(~ school_type)
```

```
## Warning: Removed 21 rows containing missing values (geom_point).
```

```
dev.off()
```

```
## pdf
##   2
```

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

   Write the commands you used:

   ```
   git checkout master
   git merge dev
   ```

   What type of merge is this?

   `Fast-forward merge`

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

   ```
   git log
   ```

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

Part III: Bar charts (5 pts)

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

   ```
   git checkout dev
   ```

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`.

```
#Filtering to Midwest
df_midwest <- df_school %>%
  filter(state_code %in% c("OH", "MI", "IN", "WI", "IL", "MN", "IA", "MO", "ND", "SD", "NE", "KS"))

png('plots/barchart_geombar.png')
ggplot(data = df_midwest, aes(x = state_code)) +
    geom_bar() +
    xlab("State") + ylab("Number of HS")
dev.off()
```

```
## pdf
##   2
```

3. Now, underneath the code for your previous graph, create the same bar chart but this time 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`.

```
png('plots/barchart_geomcol.png')
df_midwest %>%
    count(state_code) %>%
    ggplot(aes(x = state_code, y = n)) +
    geom_col() +
    xlab("State") + ylab("Number of HS")
dev.off()
```

```
## pdf
##   2
```

4. Underneath the code for your previous graph, create yet another bar chart. This one should look similar to the previous one, but this time only include 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`.

```
png('plots/barchart_visited.png')
df_midwest %>%
    filter(visits_by_110635 + visits_by_126614 + visits_by_100751 > 0) %>%
    count(state_code) %>%
    ggplot(aes(x = state_code, y = n)) +
    geom_col() +
    xlab("State") + ylab("Number of HS")
dev.off()
```

```
## pdf
##   2
```

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

   ```
   git checkout master
   git merge dev
   ```

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:

   ```
   git cat-file -p 9465bdf4b7aff703afa8e81dc6ec597c8db3e896
   ```

   Copy the content of the commit object here:

   ```
   tree 661a8295c39fadf9bd8ddb957146d3206f63f0dc
   parent 7e00aa20e41243d4898c4b45cbb30c7b2cd7484f
   author cyouh95 <25449416+cyouh95@users.noreply.github.com> 1586684528 -0700
   committer cyouh95 <25449416+cyouh95@users.noreply.github.com> 1586684528 -0700

   bar chart 3
   ```

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:

   ```
   git cat-file -p 661a8295c39fadf9bd8ddb957146d3206f63f0dc
   ```

   Copy the content of the tree object here:

```
040000 tree f97f99249a4a54f7b796ce270b9beea2cb95c01a    plots
100644 blob eed2e202ae47c28209ee61f2aca436a01f2cdf4a    problemset3.R
```

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):

   ```
   git cat-file -p eed2e202ae47c28209ee61f2aca436a01f2cdf4a
   ```

   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:

   ```
   git cat-file -p f97f99249a4a54f7b796ce270b9beea2cb95c01a
   ```

   Copy the content of the tree object here:

   ```
   100644 blob 5366b499b14fa8764418391c3d573cd251917d36    barchart_geombar.png
   100644 blob 5366b499b14fa8764418391c3d573cd251917d36    barchart_geomcol.png
   100644 blob 68225bc8c8700c7f6fbead43d0f3ccf1413843e0    barchart_visited.png
   100644 blob dff69d16b910dce50cb8a57ad885e8c3f02a9395    scatterplot_color.png
   100644 blob 578c6c2ee92727d06bf60fac902b3b6bba4d0bca    scatterplot_facet.png
   100644 blob a5fb7b683137558c1c5d67f37b86521e8a447389    scatterplot_grayscale.png
   ```

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:

   ```
   git cat-file -p 7e00aa20e41243d4898c4b45cbb30c7b2cd7484f
   ```

   Copy the content of the parent commit object here:

   ```
   tree 9fd06217383f5641212f09e3126e773388d9a793
   parent d69a25d10e514aee78b06fef1a11b6fc1369835a
   author cyouh95 <25449416+cyouh95@users.noreply.github.com> 1586684299 -0700
   committer cyouh95 <25449416+cyouh95@users.noreply.github.com> 1586684472 -0700

   bar chart 2
   ```

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:

    ```
    git cat-file -p 9fd06217383f5641212f09e3126e773388d9a793
    ```

    Output:

    ```
    040000 tree fc6cbc8c8878581ff465599c9ed08bf1dc09c9cf    plots
    100644 blob 8f3bf9c1700d9dc995426c2e12f7d8a4e49cba46    problemset3.R
    ```

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

    ```
    git cat-file -p 8f3bf9c1700d9dc995426c2e12f7d8a4e49cba46
    ```

    Command to view tree object for `plots` directory:

    ```
    git cat-file -p fc6cbc8c8878581ff465599c9ed08bf1dc09c9cf
    ```

    Output:

    ```
    100644 blob 5366b499b14fa8764418391c3d573cd251917d36    barchart_geombar.png
    100644 blob 5366b499b14fa8764418391c3d573cd251917d36    barchart_geomcol.png
    100644 blob dff69d16b910dce50cb8a57ad885e8c3f02a9395    scatterplot_color.png
    100644 blob 578c6c2ee92727d06bf60fac902b3b6bba4d0bca    scatterplot_facet.png
    ```

```
100644 blob a5fb7b683137558c1c5d67f37b86521e8a447389    scatterplot_grayscale.png
```

Command to view commit object for `parent` commit:

```
git cat-file -p d69a25d10e514aee78b06fef1a11b6fc1369835a
```

Output:

```
tree 64693f2b3b252b2ef07ba5a27f1921bd25e84300
parent d63647c51cef86fa1bf247113efbb1a7c7e55b9f
author cyouh95 <25449416+cyouh95@users.noreply.github.com> 1586684232 -0700
committer cyouh95 <25449416+cyouh95@users.noreply.github.com> 1586684232 -0700

bar chart 1
```

Part IV: Resolving merge conflicts (5 pts)

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

   ```
   git checkout dev
   ```

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`.

```
png('plots/dev_scatterplot.png')
ggplot(data = df_alabama, aes(x = total_students, y = avgmedian_inc_2564, color = as.factor(visited))) +
    geom_point() +
    geom_smooth() +
    xlab("Total enrollment") + ylab("Median average income") +
    scale_color_discrete(name = "Recruitment Visits", labels = c("No visits", "Visits")) +
    facet_wrap(~ school_type)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning: Removed 21 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 21 rows containing missing values (geom_point).
```

```
dev.off()
```

```
## pdf
##   2
```

3. Next, modify your code for the `geom_col()` bar chart from Part III, Q4 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`.

```
png('plots/dev_barchart.png')
df_midwest %>%
    filter(visits_by_126614 + visits_by_100751 > 0) %>%
    count(state_code) %>%
    ggplot(aes(x = state_code, y = n)) +
    geom_col() +
```

```
    xlab("State") + ylab("Number of HS")
dev.off()
```

```
## pdf
##   2
```

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

   `git checkout master`

5. Now, on the `master` branch, modify your code for the `geom_col()` bar chart from Part III, Q4 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`.

```
png('plots/master_barchart.png')
df_midwest %>%
    filter(visits_by_110635 + visits_by_126614 > 0) %>%
    count(state_code) %>%
    ggplot(aes(x = state_code, y = n)) +
    geom_col() +
    xlab("State") + ylab("Number of HS")
dev.off()
```

```
## pdf
##   2
```

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

   `git merge dev`

   What type of merge is this?

   `3-way merge`

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:

```
<<<<<<< HEAD
    filter(visits_by_110635 + visits_by_126614 > 0) %>%
=======
    filter(visits_by_126614 + visits_by_100751 > 0) %>%
>>>>>>> dev
```

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:

   `git merge --abort`

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:

```
git checkout dev
git merge master
```

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:

    ```
    git add problemset3.R
    git commit -m "merge dev and master"
    ```

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?

    ```
    bar chart 3
    ```

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

    ```
    Because the `modify bar chart on master` commit was made on the `master` branch, its parent commit
    ```

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

    ```
    git push --set-upstream remote_ps3 dev
    ```

Part V: I got issues (2 pts)

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 pt)

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?