

Problem Set #5

Saanchi Shah

02/12/2023

Grade: /33

Overview

In this problem set, you will continue to practice with git/GitHub and also perform some simple data manipulations in R. This week, we are focusing on working with git branches and merging. You will also get some practice on how to resolve a merge conflict. Please read the instructions carefully as you complete this problem set and write down your answers where indicated. You won't be required to write down every command you run, only the ones we ask you to write down.

Part I: Setting up your project repository

/1.5

1. Similar to the last problem set, you will create a new **RStudio project** for this problem set. Name your directory `ps5_lastname_firstname` (fill in your name). Move this `problemset5.Rmd` you're working on into your newly created project directory, and initialize `ps5_lastname_firstname` as a git repository.

```
#git init
```

/1.5

2. Create the following directory structure for your `ps5_lastname_firstname` directory. Download the **Problem set R script template** available under the **Syllabus & Resources** section of the [class website](#) (or click [here](#)). Rename the downloaded `ps_template.R` to `ps5_script.R` and save it inside your `scripts/` folder.

```
ps5_lastname_firstname
|
|- plots/
|- scripts/
  |- ps5_script.R
```

/2

3. Open up your `ps5_script.R` script. Load the `tidyverse` library and create a directory path object called `plots_dir` for the `plots/` directory. Then using your **RStudio Terminal**, add your `ps5_script.R` script and commit with the message "add `ps5_script.R` on main".

/1

4. Now, head over to GitHub in your browser and create a new private repository in the **anyone-can-cook** organization [here](#). Name your repo **ps5_lastname_firstname** (fill in your name) and do **NOT** initialize it with a **README.md** or **.gitignore** file.

/1

5. Add your newly created repository as a remote for your local **ps5_lastname_firstname** repository. Name the remote repo **remote_ps5** rather than **origin**. Write the command you used here:

Command to add remote repository

```
git remote add remote_ps5_new git@github.com:anyone-can-cook/ps5_shah_saanchi.git
```

/1

6. List out the connected remote. Use the option that will display both the remote name and URL. Write the command you used as well as the output you see below:

Command to display remote info

```
git remote -v
```

Output

```
remote_ps5_new  git@github.com:anyone-can-cook/ps5_shah_saanchi.git (fetch)
remote_ps5_new  git@github.com:anyone-can-cook/ps5_shah_saanchi.git (push)
```

/1

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

ANSWER: Because the current branch has no upstream or no configured destination

/1

8. Write the command to properly push your changes to the remote for the first time:

Command to push to remote

```
git push --set-upstream main remote_ps5_new
```

Part II: Branching & merging

/1

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

Command(s) to create and switch to 'dev' branch

```
git branch dev
git checkout dev
```

/1.5

2. List out all your branches (local & remote) as well as details on latest commits. Write the command you used and the output you see. Also answer the questions below.

```
# Command to list detailed info on local and remote branches
git branch -a
```

```
# Output
* dev
  main
remotes/remote_ps5_new/main
```

In your output above, what does the * indicate? How many local branches do you currently have? How many remote branches?

ANSWER: * indicates the branch I am currently working with.

/0.5

3. In your `ps5_script.R` script, load in the data on off-campus recruiting events by public universities from the following URL: https://github.com/anyone-can-cook/rclass2/raw/main/data/recruiting/recruit_schools.csv. Each observation (row) in the `df_school` dataframe 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 each of the following public universities:

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

/1.5

4. Let's first perform some analysis on the University of Alabama. Create a new object called `df_univ` from `df_school` by performing the following data manipulations:
 - Create a 0/1 dummy variable called `visited` that indicates whether the high school received a visit from the University of Alabama (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 of Alabama (*Hint:* See `state_code` for high school state code and `inst_100751` for university state code)
 - Subset your dataframe to include only the following variables: `ncessch`, `total_students`, `avgmedian_inc_2564`, `visited`

/1.5

5. Copy the following code to your `ps5_script.R` and run it. This will save a plot called `scatterplot_alabama.png` to your `plots_dir` that shows the relationship between total enrollment and average median income of high schools in Alabama, colored by whether or not they received a visit by the University of Alabama.

In your **RStudio Terminal**, add your `ps5_script.R` and `scatterplot_alabama.png`, then make a commit with the message "add u of alabama plot on dev".

```
png(file.path(plots_dir, 'scatterplot_alabama.png'))
ggplot(data = df_univ, aes(x = total_students, y = avgmedian_inc_2564, color = as.factor(visited))) +
  geom_point() +
  xlab('Total enrollment') + ylab('Average median income') +
  scale_color_discrete(name = 'Recruitment Visits', labels = c('No visits', 'Visits'))
dev.off()
```

/1

6. Check the commit log on your dev branch and paste your output below. Notice what it says in parentheses next to each commit hash, regarding where each of your branches are at.

Commit history on 'dev' branch

On branch dev

Untracked files:

(use "git add <file>..." to include in what will be committed)

.DS_Store

.Rhistory

.Rproj.user/

ps5_shah_saanchi.Rproj

nothing added to commit but untracked files present (use "git add" to track)

/1

7. Now switch back to the main branch and check the commit history there. Paste your output below. Compare it to what you see in the previous question and make sure you understand what you see (no need to write down anything).

Commit history on 'main' branch

On branch main

Your branch is up to date with 'remote_ps5_new/main'.

Untracked files:

(use "git add <file>..." to include in what will be committed)

.DS_Store

.Rhistory

.Rproj.user/

ps5_shah_saanchi.Rproj

nothing added to commit but untracked files present (use "git add" to track)

/1

8. Merge in the changes from dev into main and write the command you used below. What type of merge is this?

```
# Command to merge changes from 'dev' into 'main'
git merge dev
```

```
# What type of merge is this?
Fast-forward merge
```

/1

9. Check the commit history on your `main` branch again and paste the output below. Again, make sure you understand what you see and how it compares with the previous steps.

To view commit history can use `git branch -v` or `git log`.

```
# Commit history on 'main' branch after merge
```

```
commit 95fb31e24af72b08fa2993c2d29358c995f4f925 (HEAD -> main, dev)
Author: SAANCHI SHAH <sshah15@g.ucla.edu>
Date:   Sun Feb 12 19:24:35 2023 -0800
```

```
    add u of alabama plot on dev
```

```
commit 7fb52a825494a25bfb8d49f5f1e9c9c4d82c16c9 (remote_ps5_new/main)
Author: SAANCHI SHAH <sshah15@g.ucla.edu>
Date:   Sun Feb 12 19:10:19 2023 -0800
```

```
    add ps5_script.R
```

/1

10. Still on your `main` branch, push your changes to the remote. Check the commit history yet again and paste the output below. You should see that the `dev`, local `main`, and remote `main` branches are all even (i.e., in-sync).

Code: could use `git branch -v` or `git log` to see the commits.

```
# Commit history on 'main' branch after pushing to remote
```

```
On branch main
```

```
Your branch is up to date with 'remote_ps5_new/main'.
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
  .DS_Store
```

```
  .Rhistory
```

```
  .Rproj.user/
```

```
  ps5_shah_saanchi.Rproj
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

Part III: Resolving merge conflicts

/2

1. Switch back to the `dev` branch. Open up `ps5_script.R` and modify your code from Part II, Q4 so that `df_univ` is based on University of Colorado Boulder instead of University of Alabama. (*Hint*: The `visited` column should now be based on `visits_by_126614`, and the high schools should be filtered to only those in the state of `inst_126614`)

Next, also modify your code from Part II, Q5 so that the plot is saved in a file called `scatterplot_cuboulder.png`. Run the code to save the new plot.

In your **RStudio Terminal**, add your `ps5_script.R` and `scatterplot_cuboulder.png`, then make a commit with the message "add cu boulder plot on dev".

/1

2. Now switch back to the `main` branch. If you look at `ps5_script.R`, you will see it still has the code for University of Alabama. Modify the code so that `df_univ` is now for UC Berkeley, and save the plot as `scatterplot_ucberkeley.png`.

In your **RStudio Terminal**, add your `ps5_script.R` and `scatterplot_ucberkeley.png`, then make a commit with the message "add uc berkeley plot on main".

/1

3. At this point, you have made an additional commit each to the `dev` and `main` branches, so the branches have diverged. Still on the `main` branch, try merging in the `dev` branch and write the command you used below. What type of merge is this?

```
# Command to merge changes from 'dev' into 'main'
```

```
#git merge dev
```

```
# What type of merge is this?
```

```
3-way merge
```

/1

4. Uh oh, you've run into a merge conflict! But don't panic. You remember there is a command to abort the merge and return the branches back to their original states. Run the command and write it below:

```
# Command to abort the merge
```

```
#git merge --abort
```

```
\textcolor{red}{\textbf{/1}}
```

5. Phew! Everything is back to the way it was. Now let's say you still want to combine changes from both

Switch to the `'dev'` branch, and merge the changes from `'main'` into `'dev'`. Write the command you used

Command to switch to dev branch

```
#git checkout dev
```

Command to merge changes from main into dev

```
#git merge main
```

```
\textcolor{red}{\textbf{/2}}
```

6. You run into the same merge conflict, but this time, let's try resolving the conflict. Start by running

Now open up 'ps5_script.R' in **RStudio**, and you should see that Git had added markers around the

After you finish resolving the conflicts, use your **RStudio Terminal** to add 'ps5_script.R' and make

Command to add ps5_script.R

```
git add scripts/ps5_script.R
```

Command to make commit

```
git commit -m "merge dev and main"
```

```
\textcolor{red}{\textbf{/3}}
```

7. Still on the 'dev' branch, check your commit history. Note that the commit you made on the 'main' branch

Using the 'git cat-file' command, print out the contents of the commit object for this "add uc berkeley

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

Command to print content of the commit object for the "add uc berkeley plot on

```
git cat-file -p255bfaf
```

Commit message of the parent commit

```
add u of alabama plot on dev
```

Parent commit hash: 95fb31e24af72b08fa2993c2d29358c995f4f925 # Why is the parent not just the previous commit listed in the log? Answer: Because we merged the two branches but had initially worked on the dev branch to save the alabama plot. This commit now has a parent in the dev branch. The merge has two parents now.

```
\textcolor{red}{\textbf{/1}}
```

8. Lastly, push the ‘dev’ branch to the remote. Don’t forget to set the upstream branch during this ini

Command to push to remote

““

Part IV: Create a GitHub issue

- Go to the [class repository](#) and create a new issue.
- Please refer to [rclass2 student issues readme](#) for instructions on how to post questions or things you’ve learned.
- You can either:
 - Ask a question that you have about this problem set or the course in general. Make sure to assign the instructors (@ozanj, @xochilhlopez, @joycehguy, @augias) and mention your team (e.g., @anyone-can-cook/your_team_name).
 - Share something you learned from this problem set or the course. Please mention your team (e.g., @anyone-can-cook/your_team_name).
- You are also required to respond to at least one issue posted by another student.
- Paste the url to your issue here: https://github.com/anyone-can-cook/rclass2_student_issues_w23/issues/217
- Paste the url to the issue you responded to here: https://github.com/anyone-can-cook/rclass2_student_issues_w23/issues/214

Knit to pdf and submit problem set

Knit to pdf by clicking the “Knit” button near the top of your RStudio window (icon with blue yarn ball) or drop down and select “Knit to PDF”

You will submit this problem set by pushing it to your repository. Make sure to push both the .Rmd and .pdf files.