Unix, Git, and GitHub Lab

Setup

0. Please watch the two YouTube videos before this lab. The links are: https://www.youtube.com/watch?v=hrTQipWp6co and https://www.youtube.com/watch?v=RGOj5yH7evk. For the second one, you can skip the part on SSH keys (20:30 through 25:27). Note that GitHub was updated and the primary branch is now called "main" not "master", so any time you see commands with "master" in them, you'd have to use "main".

1. Install **R**, RStudio, git, and Git Bash (if you use a Windows computer). Installation instructions for git can be found here: http://rafalab.dfci.harvard.edu/dsbook-part-1/productivity/installing-git.html.

2. Create an account on GitHub. Remember that the name should be professional and able to identify you reasonably well.

3. Open a terminal on your computer (Terminal app on Mac or Git Bash on Windows) and take a few minutes using the **cd** and **ls** commands to navigate through your system and list the contents of the directories.

Initializing Git and Syncing with GitHub

4. In the terminal, navigate to a directory where you want to create a folder for this class. Still in the terminal, create a new directory with the name "Math_3190" (without the quotes) and then navigate to that folder in the terminal.

5. In the terminal, create another new directory inside Math_3190 called "Math_3190_Assignments" and change directory to that new folder.

6. Now we can start using Git. First, we'll want to configure Git. Type each of the following, but edit the email and name.
   git config --global user.email "you@example.com"
   git config --global user.name "Your Name"
   Then type this line to make sure the default name for each branch is "main" rather than "master" to align with GitHub.
   git config --global init.defaultBranch main
   You only have to run those git config --global commands once and those will be saved whenever you use Git from now on.

7. We are going to initialize Git in this folder. Use the **git init** command.

8. Create the file README.md using the **touch** command. Then create the file .gitignore.

9. Edit the README.md file by adding a description about this being your repository for all the Math 3190 assignments. You can edit this in a text editor like TextEdit or Notepad or you can edit it in RStudio.

10. Now, enable the ability to view hidden files on your computer and edit the .gitignore file. Simply add a line that has ".DS_Store" without the quotes. Files that end in .DS_Store are created by some operating systems (like macOS) for file management. We don't want that file tracked by Git.

11. From the Math_3190_Assignments folder in then terminal, type **git status**. You'll see the README.md file and the .gitignore file.

12. Now we will want to add all those files to Git. Use **git add** to add them all individually or use the period to add them all. Then use **git status** again to see the changes.

13. Those files are now in the staging area. To save them, we need to commit them. Type **git commit -m "message"** and change the message to indicate that this is the initial commit and specify the files that were added. Then type **git status** again.

14. Now create two directories in this Math_3190_Assignments folder: Labs and Homework. In the Labs folder, add another folder called "Lab_1". Download and put these instructions in that Lab_1 folder.

15. Type **git status** again. You'll see the Labs folder, but probably will not see the Homework folder. Empty folders are not tracked by Git until there is something in them.

16. Add and commit again to indicate the Lab 1 has been added. Then type **git status** again.

17. Type **git log** to see a log of your commits. Copy the commit hash from your first commit. (Note: on Windows, you cannot use Command-c for this. You will need to highlight it then right click and select copy.) Then type **git checkout commit_hash**. Do not type "commit_hash". Instead, paste the commit hash from the first commit. (Again, windows users will need to right click and select paste.) You'll see that the Lab_1 folder should be gone. This has put you in a new temporary branch. Type **git branch** to see the two branches.

18. Now type **git log** again. Only one commit will show up. In order to see the second one, type **git log --all**. One commit will have (main) beside it and one will have (HEAD). The HEAD commit is the one you are looking at. The (main) commit is the most recent one on the original branch.

19. Type **git switch -c new_branch** to create a new branch from the temporary one. Any changes made here will not affect the main branch. Type **git branch** to see the two branches.

20. Using the terminal, copy the README.md file and paste it in the Homework folder. Then add and commit that change.

21. Type **git log --all --graph** to see the logs and the branches.

22. Now type **git checkout main** to switch back to the main branch. The READEME.md file should no longer be in the Homework folder and instructions should be back in the Lab_1 folder.

23. We are going to merge the branches now. Before we do, however, it is best to change the editor used in Git. Type **git config --global core.editor "nano"** to switch to a simpler text editor.

24. Now type **git merge new_branch** to merge the main branch with the new one. This will bring up an editor in which you will need to write a message about why you are merging. Using only your keyboard, type a message about this being done for Lab 1. Then exit by typing Control-x, then agree to save the changes, then press enter. The branches will now be merged. The README.md file should be in the Homework folder and the instructions should be in the Lab_1 folder.

25. We can now delete the new_branch by typing **git branch -d new_branch**. Then edit the README.md file in the Homework folder to indicate that this is the README for Homework folder. Then add and commit this change.

26. It is now time to bring GitHub into the mix. In GitHub, create a repository with the name Math_3190_Assignments. Do not add any files to it (not a README.md, .gitnore, or a license).

27. Back in the terminal, type
    git remote add origin https://github.com/username/Math_3190_Assignments
    Change the username part to your GitHub username.

28. Now type **git push -u origin main**. In the future, you will only need to type **git push** since the upstream is being set right now.

29. You will likely be asked to enter a username and password at this stage. The username is your GitHub username, but the password will be a token you have to generate. You can generate the token using these steps:
    a. Go to your GitHub home page: https://github.com/username.
    b. Click on your account icon on the top right and go down to "Settings".

c. On the left side, at the bottom, click on “Developer Settings”.
d. On the left, click on “Personal access tokens”.
e. Click on Tokens (classic). We will be using the classic tokens, not the fine-grained tokens.
f. Click on “Generate new token” and the “Generate new token (classic)”.
g. Select an expatriation time. 90 days is good.
h. Select “rep”, “workflow”, “gist”, and “user” under scopes.
i. Click on “Generate token”.
j. Copy and paste this token for your password. You won’t have to do this again for another three months.

More info can be found here.

30. Go back to your GitHub page and go to the Math_3190_Assignments repo. All of the files should be there.

Forking and Cloning

31. The final part of this lab is to fork a repository and the clone it to your local system. Navigate to my GitHub page: https://github.com/rbrown53. Click on Repositories and then on the mypackage repo. Near the top on the right side, click on Fork. You can keep all options as they are (add a description if you’d like) and then click on Create Fork. This will create your own mypackage repo with all of my code and files in there.

32. On GitHub, navigate to the mypackage repo in your repositories. Click on the green Code button, and then copy the url given there.

33. Back in the terminal, go back one directory to your Math_3190 folder (not the Math_3190_Assignments). Then type **git clone url**. Paste the url you copied instead of typing “url”. This will clone the repo on your system.

Pull Requests

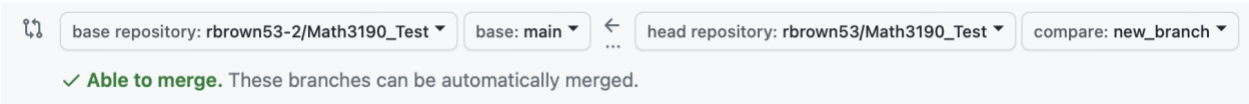
34. It is good practice to create a new branch when forking and cloning to keep the original branch a perfect snapshot of the forked repo. Create a new branch by going into the mypackage directory in the terminal and typing **git branch new_branch**. You don’t have to call it new_branch. You can call this new branch anything you’d like. Then use **git checkout new_branch** to move over to it. Alternatively, you could type **git checkout -b new_branch** to create it and move over to it in one command.

35. In the mypackage folder, add a new folder called “First_Last” with your name. In that folder, add a text document (with the extension .txt) that includes a couple sentences about how much experience you have had in the past with these tools we covered today

in the lab.

36. Once you have done that, add, commit, and push the changes to your forked GitHub repo. Since the GitHub repo does not have this new branch, you will have to use **git push -u origin new_branch** (or whatever you called the new branch).

37. Go to GitHub and to the forked mypackage repo. At the top, it should have a yellow message saying Compare & pull request. Click on that. This will take you to the Pull Request (PR) page. Near the top, you should see something that looks like the picture below. Of course, in your case, the left side will have rbrown53/mypackage and the right side will have your username. The arrow in the middle is indicating the repo on the right will be sent to the repo on the left.



Add a title to your PR that indicates what you did (added a new folder). Then add a description with more detail of what is in it. Then click on Create pull request. You can check out the Files changed page for a summary of what is different in the original and in the directory you requested be pulled.

38. Once that is done, I will approve the request. Then you can go back to your mypackage repository, select the main branch, and click the Sync fork button and then Update branch. This will align your fork with my original repo.

39. Now go back to the terminal, switch back to the main branch, and use **git pull**. This will update your local directory with GitHub, which is synced to my original repo.

40. Finally, you can delete the new branch since main is up to date. It is usually best to not delete this entire the PR has been accepted and then the main branch is updated. You can delete the new branch by typing **git branch -d new_branch**.

Finishing Up

41. Once you have finished all the above, go into your Math_3190_Assignments repo on GitHub. Go to the settings, go down to the Danger Zone, and change the visibility to private. It will ask you if you are sure several times. Then go back into settings, click on Collaborators on the left side, and click on Add people. Add me (rbrown53) as a collaborator. This will allow me to see your assignments repo, but will not allow anyone else to see it.