Data Science Methods, Lab for Week 2: Git

Your Name

2020-09-07

This lab introduces you to navigating histories on GitHub, `git checkout`, and `git revert`, as well as the git side of the workflow we'll use for labs.

Summary:

1. Use `git checkout [hash]` to go back to a previous commit.
2. Use `git revert [hash]` to undo history back to a previous commit.
3. The git side of the lab workflow goes like this:
   a. Fork and clone the lab repo from https://github.com/data-science-methods.
   b. Follow the instructions in `lab.R`
   c. When you're finished, push the results back up to GitHub and file a pull request for the original repo.

Problem 1

This problem sets up some changes in the repo that we can browse in GitHub's history view.

1. Navigate to the GitHub repo for this lab assignment, https://github.com/data-science-methods/lab-w2-git. Fork the lab, then clone it to your working machine. Open the file `lab.R` in a *new* R session.

2. Edit `lab.R`. You don't need to do anything in particular, but they should be trackable changes. Add and commit the changes to your working machine. In the commit message, put something like "problem 1 step 2."
3. Edit `lab.R` again, making another round of changes. Add and commit the changes to your working machine. In the commit message, put something like "problem 1 step 3."
4. Push the repo back up to GitHub.

Problem 2

Now we'll review the history on GitHub, and retrieve an identifier for a commit from the history.
1. Open your fork of the lab repo on GitHub. The URL will be something like https://github.com/username/lab-w2-git.
2. Find and click on the history button, above the file list and to the right. (I wasn't able to find a keyboard shortcut for this. :-( )

3. Look through the history for your commit from problem 1 step 2 (the first round of edits). Click on the *comment* to view the *diff*, or the changes made by that commit.

4. Back on the history page, find the *hash* on the right side. It should be a series of 7 letters and numbers. This is an ID code that git uses to track each individual commit. Click on the clipboard icon to copy the hash.

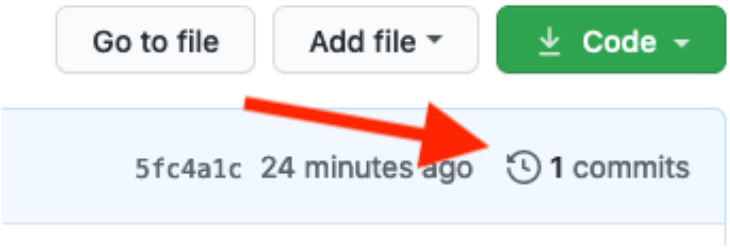

Figure 1: The history button in GitHub is immediately above the file list, on the right

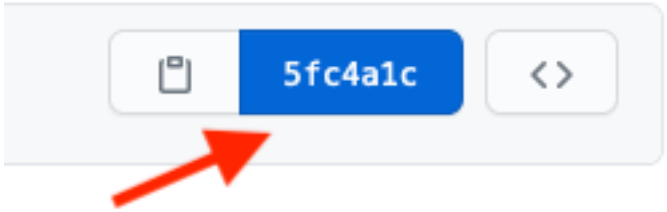

Figure 2: The hash for a commit is the series of 7 letters on the right side of the entry in the history view.

Problem 3

1. Back on your working machine, enter the following into the command line, pasting your hash in place of the example.

```
$ git checkout 5fc4a1cbe6331e1a38a70ed49d7bfc8182461bc2
```

2. Look at `lab.R`. Your edits from problem 1 step 3 are gone!
3. Check `git status`. Note that it says `HEAD detached at [hash]` in red. This means that making commits right now will cause serious problems. (To confirm: git won't let us make commits with a detached HEAD.)
4. To get out of this "detached HEAD" state, do the following:

```
$ git checkout main
```

5. Confirm that your edits from problem 1 step 3 have been restored and that `git status` isn't warning you about a detached HEAD.

Problem 4

Suppose that the changes from problem 1 step 3 caused things to break, and we want to permanently go back to the previous state of things (ie, after we finished problem 1 step 2). We do that using `git revert`.

1. In the GitHub history, find the hash for the commit that *caused* the breaking changes.
   - This is the commit from problem 1 *step 3*, not step 2.
   - Note that this is different from the commit we used in Problem 5.
2. Do the following:

```
$ git revert [hash for problem commit]
```

3. Confirm that the changes from problem 1 step 3 have been undone. Push to GitHub, and examine the history view.

Problem 5

When you want to contribute to someone else's code — or when you're finished with the labs in this course — you can't just push your changes back to the original repository. Instead you file a *pull request* (often abbreviated as *PR*), which is a request that the owner of the original repository integrate your proposed changes.

1. Make sure that all of your work has been pushed up to GitHub.
2. Open up your copy of the repository on GitHub. Go to the "pull requests" tab, and then click on the green "New pull request" button.

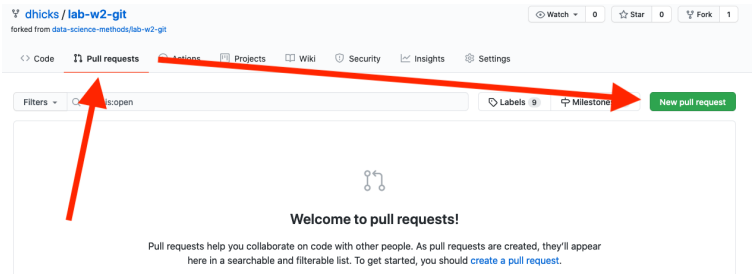

Figure 3: Location of the pull requests tab and new pull request button

3. The New Pull Request screen will show you "base repository" (the original repository, in this case the copy owned by the course account) and your "head repository" (containing the changes you want to suggest). You can also review the changes with the diff view before hitting the green "Create pull request" button.

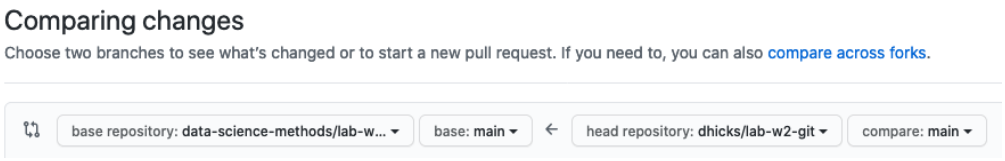

Figure 4: The New Pull Request screen indicates the base and head repositories

4. Finally, you need to add a comment explaining your pull request. What do these changes do, and how do they benefit the project?
   - For this course, you can just comment that you've completed the lab.
   - The R side of the workflow will automatically check your work after you file a pull request. But I'll explain that in the R lab for this week.