Lab 2 - GitHub

Learning goals

In this lab, you are expected to learn/practice the following skills:

- Forking a repository on GitHub
- Git workflow: clone/commit/push
- Using pull requests (PRs)

Updating a single repo

One of the most common tasks that people use git for is for collaborating. While in general team members organize such that there is no overlapping editing of the files, `git` is (usually) smart enough to avoid clashes when multiple edits are done in the same document[1]. To show this, we will do a collaborative edit of a file!

We will be working with the repository https://github.com/USCbiostats/PM566-whoami

Step 1: Fork the project to create your own repo

Not a term/command actually available in Git, forking is a feature available in GitHub (as in other services) that allows users to create copies of other people's projects to propose changes (i.e. make **pull requests**, i.e. "I have this great update for your project! Would you like to add it by *pulling it* into your repo?").

To start, you just need to use the Fork button available on the main page of the repository you would like to contribute to[2]:

Once you "Fork" a project, GitHub will automatically:

1. Create a copy (using `git clone`) of that project in your account.

2. Set up a pipeline to generate pull requests for the original repository.

Once you have a copy of the project in your account, you can proceed by "downloading it" to your computer. You can do this using either the command line (Terminal) or the GitHub Desktop app.

Command Line

You can download your version of the `whoami` repository using the `git clone` command. You will need to copy the URL from **your version** of the repository, available under the “Code” button. For example, if your github user name is `statsnerd` and the repository name is `PM566-whoami`, you could use the following in your command line

```
cd where/you/want/to/download/the/thing
git clone https://github.com/statsnerd/PM566-whoami.git
```

And if you have your ssh credentials set up, you can do instead

```
cd where/you/want/to/download/the/thing
git clone git@github.com:statsnerd/PM566-whoami.git
```

This way you will get a copy of the repository in your local machine. Now, let’s see how can we update the project!

GitHub Desktop

Alternatively, you can open GitHub Desktop, click the Current Repository tab in the upper left, click the Add button, and select Clone Repository. If you are signed in to your GitHub account, this should open a dialog box allowing you to search your existing repositories. Find your version of the `PM566-whoami` repo and Clone it.

Step 2: Modifying the corresponding line

If you got the correct copy, you should find a very simple repository with only two files: `CODE_OF_CONDUCT.md` and `README.md`. The first file is a general code of conduct for the project, which we do not need to edit. The second file is the one that we will be playing with. The README file, which happens to be a Markdown file, contains, or at least will contain, your and your team members’ biographies. Here is what you need to do:

1. Find the line with your name.
2. In that single line (i.e. not spanning multiple lines), write something about yourself, e.g. “I am from XYZ, I love doing ABC, …”.
3. (optional) if you feel like it, add at the end of the line a picture of yourself (or avatar) using either html or markdown. This will require you to include the figure in the `images` folder of the repo, unless you provide a link to a picture online.
4. Commit the changes and push the changes to your repo using `git commit` and `git push`, e.g.

```
        git commit -a -m "[A short but meaningful message]"
        # git add [your-avatar.png] ... if you need to add a picture
        git push
```

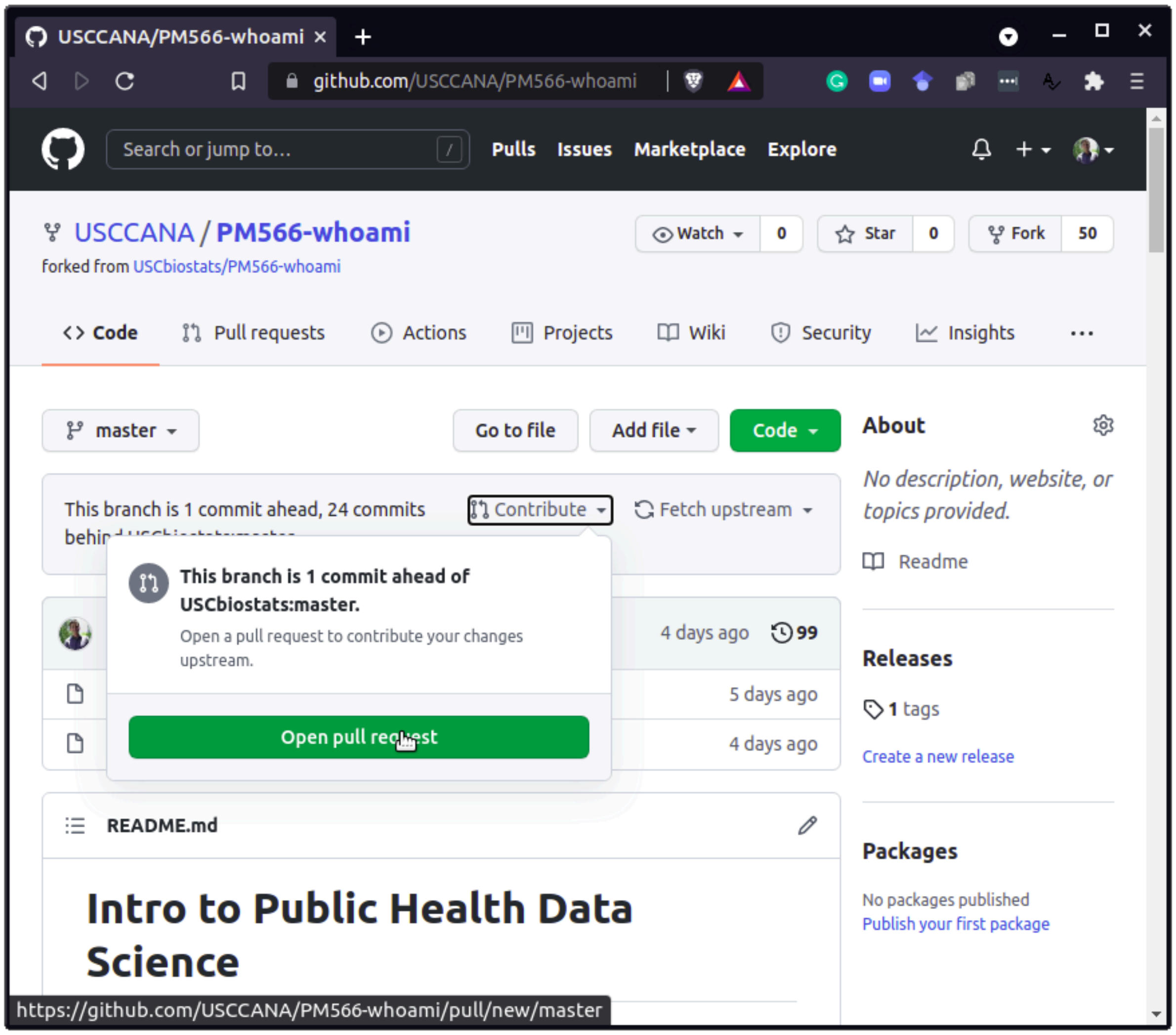
Or using the “Commit [n] file(s) to main” and “Push origin” buttons in GitHub Desktop.

You have now updated **your** online version of the `PM566-whoami` repo and are one step closer to make your first Pull Request. We will see how that happens in the next part.

Step 3: Do the pull request

This is the final step. Overall, pull requests (PRs) are as complex as the proposed changes are. The PR that you are about to make should go smoothly, yet, any time that you make a new PR, the changes should be able to be `merged` in the original repository without conflicts. Conflicts may only appear if the proposed changes are out-dated with respect to the main repository, meaning that the main repository was modified *after* your fork and your proposed changes cannot be merged without generating conflicts[3]. For now, let's just look at the simple case.

To create the PR, you just need to go to your online copy of the project and click on "Contribute" then "Open Pull Request":



This branch is 1 commit ahead, 24 commits behind USCbiostats:master.

This branch is 1 commit ahead of USCbiostats:master.

Open a pull request to contribute your changes upstream.

No description, website, or topics provided.

No packages published

Intro to Public Health Data Science

https://github.com/USCCANA/PM566-whoami/pull/new/master

You can submit pull requests to the original repo from your copy of it via the "Contribute" button.

This will create a PR in the original repository. GitHub will automatically analyze the PR and check whether merging the PR to the master branch will result in a conflict or not. If all is OK, then the owner/admin of the repository can merge the PR. Otherwise, if there's a conflict, you can go back to your local repo, make the needed changes, commit the changes, and push the changes to your copy on GitHub. In this stage, the PR will automatically update to reflect the new changes you made in your copy of the project.

For more information, check out [Creating a pull request from a fork](#) on GitHub.

Submitting your lab

Once you have created a Pull Request and it has been accepted, you should see your text on the **original repository** (not just your personal forked copy). If you can see your text on the USCbiostats version, then congratulations, you're done with Lab 2!

Footnotes

1. Team-members could be working on the same file but editing different lines of code. If this is the case, after pull/push, git will integrate the changes without conflicts. ↩
2. For more details, take a look at the Forking Projects article in GitHub guides. ↩
3. More info about how to deal with conflicts in this very neat post on stackoverflow.com How to resolve merge conflicts in Git. GitHub also has a way to solve conflicts in PRs, but this is only available to the admins of target repo. More info here, ↩

University of Southern California
Department of Population and Public Health Sciences

LICENSE
View the source at GitHub.