Reproducible research #2

More on GitHub

GitHub

GitHub helps you work with others on code. There are two main ways you can do this:

- **Collaborating:** Different people have the ability to push and pull directly to and from the same repository. When one person pushes a change to the repository, other collaborators can immediately get the changes by pulling the latest GitHub commits to their local repository.
- **Forking:** Different people have their own GitHub repositories, with each linked to their own local repository. When a person pushes changes to GitHub, it only makes changes to his own repository. The person must issue a pull request to another person's fork of the repository to share the changes.

Issues

Each original GitHub repository (i.e., not a fork of another repository) has a tab for “Issues”. This page works like a Discussion Forum.

You can create new “Issue” threads to describe and discuss things that you want to change about the repository.

Issues

Issues can be closed once the problem has been resolved. You can close issues on the “Issue” page with the “Close issue” button.

If a commit you make in RStudio closes an issue, you can automatically close the issue on GitHub by including “Close #[issue number]” in your commit message and then pushing to GitHub.

For example, if issue #5 is “Fix typo in section 3”, and you make a change to fix that typo, you could make and save the change locally, commit that change with the commit message “Close #5”, and then push to GitHub, and issue #5 in “Issues” for that GitHub repository will automatically be closed, with a link to the commit that fixed the issue.

Pull request

You can use a *pull request* to suggest changes to a repository that you do not own or otherwise have the permission to directly change.

You can also use pull requests within your own repositories. Some people will create a pull request every time they have a big issue they want to fix in one of their repositories.

In GitHub, each repository has a “Pull requests” tab where you can manage pull requests (submit a pull request to another fork or merge in someone else’s pull request for your fork).

Pull request

Take the following steps to suggest changes to someone else's repository:

1. Fork the repository
2. Make changes (locally or on GitHub)
3. Save your changes and commit them
4. Submit a pull request to the original repository
5. If there are not any conflicts and the owner of the original repository likes your changes, he or she can merge them directly into the original repository. If there are conflicts, these need to be resolved before the pull request can be merged.

Merge conflicts

At some point, you will get *merge conflicts*. These happen when two people have changed the same piece of code in two different ways at the same time.

For example, say Rachel and are both working on local versions of the same repository, and I change a line to `mtcars[1, ]` while Rachel changes the same line to `head(mtcars, 1)`. Rachel pushes to the GitHub version of the repository before I do.

When I pull the latest commits to the GitHub repository, I will have a merge conflict for this line. To be able to commit a final version, I'll need to decide which version of the code to use and commit a version of the file with that code.

Merge conflicts

Merge conflicts can come up in a few situations:

- You pull in commits from the GitHub branch of a repository you've been working on locally.
- Someone sends a pull request for one of your repositories.

Merge conflicts

If there are merge conflicts, they'll show up like this in the file:

```
<<<<<<< HEAD
mtcars[1, ]
=======
head(mtcars, 1)
>>>>>>> remote-branch
```

To fix them, search for all these spots in files with conflicts, pick the code you want to use, and delete everything else.

Merge conflicts

For the example conflict, I might change the file from this:

```
<<<<<<< HEAD
mtcars[1, ]
=======
head(mtcars, 1)
>>>>>>> remote-branch
```

To this:

```
head(mtcars, 1)
```

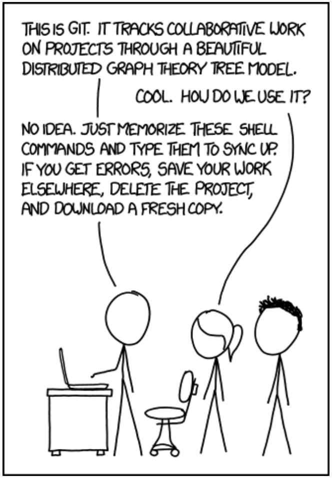
Then you can save and commit the file.

Find out more

If you'd like to find out more, Hadley Wickham has a great chapter on using git and GitHub with RStudio in his *R Packages* book:

http://r-pkgs.had.co.nz/git.html

Final note on git


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Source: xkcd