**Get GitHub**

**Why?**

* It’s a great place to store sequential versions of a project in a safe place and in a really organized way
* It’s ideal for science collaborators

**Some Notes on GitHub:**

"Repositories" (“repos” for short) are basically the GitHub equivalent of an RStudio "Project" - it's just a place where you can easily store all information/data/etc. related to whatever project you're working on.

When we create a repo in GitHub, we can have it communicate with a Project in RStudio:

* We can easily get (pull) information from GitHub to RStudio
* We can push information from RStudio to GitHub where it is safely stored and/or our collaborators can access it. The beauty is that it keeps a complete history of updated versions that can be accessed/reviewed by you and anyone with access to the repository.

Under the free version of GitHub, you can create an unlimited number of private repos, which can be shared among up to 3 collaborators (you can upgrade for more collaborators). This can be very important when you're working on projects with confidential data/analyses/etc. You can also create an unlimited number of public repos with no restriction on the number of collaborators. We'll be using publicly available repos.

**Let’s get started:**

**Step 1.** **Start organized!!**

First, make a folder on your computer called 'github'. I have mine in my home directory. Just have a really clear place where you're going to put your git-related business.

**Step 2. Make a GitHub account:** [**https://github.com**](https://github.com)

**Step 3. Get someone else’s stuff on GitHub into your own repo, modify it in RStudio, and push it back**

Sometimes, you'll want to use a model/data/etc. that someone else has stored on GitHub. Let's say that's a person you don't know, but they have a public repo because they want a lot of people to use their model/data/etc. How can we get information from \*their repo\* into \*our github\* and then \*work with it in RStudio\* before \*sending out updated version back to github\*? That's what we'll do first.

This requires a couple of steps:

1. **FORK.** As you can imagine, people you don't know usually don't want you to be able to update their files without them knowing it. That's why you can't actual change a file in a 'master branch' unless you've been added as a collaborator by the "owner". So instead of using and changing their originals, you need to ***fork*** their repo first. This essentially makes a copy of all information in their repo and stores it as your own.

* In GitHub, search for the ‘144l\_students’ repo and select it
* Press '**Fork**' in the top right
* Now navigate back to your GitHub account (if you click on your user picture in the top right, it'll bring down a menu where you can select "Your repositories"), and you'll see the forked repo exists in your world where you can mess with it

2. **CLONE.** Once you've forked a repo into your own GitHub, you probably want to start working with it in RStudio. How?

The easiest way is to get the repo URL from GitHub, THEN create the corresponding project in RStudio that it'll communicate with. When you're in your forked repo that you want to work with:

* Click on 'Code' in green
* Select and Copy the ENTIRE URL
* Open a new RStudio session
* Choose New Project > \*\*Version Control\*\* > \*\*Git\*\* > Paste the URL from GitHub into the 'Repository URL' bar > Make sure you're in the subdirectory you want to be in > Select 'Open in a new session' > Create project!

Now we’re working in an RStudio project and all of the files that WERE in the repo you forked are now in your working directory. We \*\*also\*\* have version control power now!

What does that mean? Let's make some changes in RStudio and find out.

* Open up the README.md file
* Make some changes (e.g. type in your name and the date)
* Save the file

3. **STAGE/COMMIT/PUSH.** Now your project in RStudio has been updated, but we want this new 'version' of the Project to be stored in GitHub. We follow the process of Stage > Commit > Push to send versions back to GitHub.

* Click on the 'Git' tab (up by Environment/History)
* Select files waiting to be committed (stage all of them)
* Click 'Commit'
* Add a short commit message (required), then press 'Commit'
* No error = working
* Press 'Push' - the green 'UP' icon to send the commits back to GitHub
* Refresh your GitHub repo, and see that the changes YOU made now exist in your repo. Cool! Cloud storage and version control!

**Extra Stuff:**

**Making your own GitHub repo/RStudio project from scratch**

You don't always want to use a repo that someone else has created. Usually, you'll have some data that you want to analyze in RStudio, and you might want to use GitHub to help with version control + collaboration.

The easiest way is to CREATE THE REPO FIRST, then follow the steps above (copy/paste URL for GitHub repo to set up R project with version control).

**Collaborating**

Sometimes, we'll want to be collaborate on a paper or data analysis work. If that's the case, we'll want to add a collaborator.

\*\*IMPORTANT: Collaborators will be able to push/pull changes directly to/from the repo you share with them, so make sure it's someone you want to allow!\*\*

Once you have confirmed that you want to add someone as a collaborator:

* Figure out what name or username or email they have associated with their GitHub account
* While in the Github repo that you want to add them to as a collaborator, click on 'Settings'
* Click 'Manage Access' and enter your password
* Click ‘Invite a collaborator’
* Enter their name(s) (any of the above to search)
* When you find the right one, click 'Add [username] to [repo]'
* And now they can clone and push/pull directly, and you can both be working on the same project