**Git and GitHub Tutorial**

R Workshop, Emory University

Spring 2025

Adee Weller and Eddy Yeung

What is Git?

* Git is a free, open-source version control system that helps developers track changes to their code. It's the most widely used version control system in the world. Every coder has their own codebase and can work on them at any time. You create branches, merge them, and delete (and undelete!) them with ease. This makes it very easy to track changes and revert to older versions of the code.
* It is particularly popular with software developers and collaborative teams. It is not yet widely popular with political scientists (who are scared of coding) but it **Should Be** (and is becoming more popular)!
* It is open-source and free.
* Git comes with a terminal called GitBash.

What is GitHub?

* GitHub is a cloud-based hosting service and website that helps software developers store, track, and collaborate on projects. It's a social network for programmers that encourages collaboration and sharing.
* It used Git to move files and track changes while multiple people work on the same project simultaneously.
* Software developers use it to show off their code and work ethic through their profiles.
* All files are visible by all, increasing transparency and honesty.
* You can also make free websites through it, and there are many tutorials online to help with its use.
* Also has GitHub CoPilot, and AI/LLM for coding questions.
* You need git to operate GitHub.

Go to GitHub, and show the page, click on a profile, play around on the site and get familiar with it.

Then go to [Adee’s R workshop page](https://github.com/adeeweller/R_Workshop) – that’s where we’ll be focusing.

**Opening Git**

Think of a repository or ‘repo’ as a folder where all your code and data are kept. You can share this folder and edit it as you wish. You can either take local files and make them into a repo, or you can clone one from an existing Git Repository. There will be two repos – one in the cloud and one on your device.

1. Open up the Bash on your device. For me, I type ‘bash’ into the search bar and hit enter.
   1. For a Mac, this is ‘terminal’ where GitBash is accessed
   2. (You could also use any other terminal of your choice, but I will focus on GitBash for PC and Terminal for Mac for this tutorial.)

**Configure Git**

1. Add your name and email in the global settings:
   1. git config --global user.name “Adee Weller”
   2. git config --global user.email [adee.weller@emory.edu](mailto:adee.weller@emory.edu)
   3. git config --global init.default branch main
      1. this sets up the default branch that Git will use
   4. If you need help for any command, try <command> -h or help <command>
   5. To clear the terminal, just type ‘clear’

**Creating a Local Git Repository**

1. Move to the working directory where you want your project to be stored:
   1. cd pathfile
      1. if the folder doesn’t exist yet, make one:
         1. mkdir foldername
   2. for PCs, folders end in ‘/’
   3. for Macs, folders are just words
2. Initialize Git repo:
   1. git init
      1. this will initialize an empty Git Repo ‘/.git/’
3. Check if that worked:
   1. git status

**Make a new file**

1. Create a new blank file to practice moving code.
   1. touch newfile\_**AW**.txt – creates a blank file named newfile.txt in your wd
      1. Instead of **AW**, put your own initials
2. Check if this worked:
   1. ls

**Add that file to the staging environment**

Think of the staging environment like you’re putting the code in the wings of a stage, ready to be ‘pushed’ in front of an audience, but still behind the curtains. That ‘push’ or **commit** is a record of what changes you are making when you push code from the staging environment onto the cloud. In the past, this was sometimes called the ‘index’.

1. Add the file you made to the staging environment:
   1. git add <filename>
      1. git add newfile\_AW.txt
2. You can check if this worked:
   1. git status
      1. This should show that one change is ready to be committed
      2. Any other changes there will be tracked
3. To unstage a file:
   1. git rm --cached <filename>
4. To track all the files in the working directory:
   1. git add .
5. To organize files that Git ignores altogether (such as sensitive data or identifying information), create a “.gitignore.txt” file with the names or types of the files to be ignored. There is a lot of information online for suppressing different file types according to your own needs.

**Commit Changes**

1. Commit the changes with a comment to tell collaborators (and yourself) what changes are being made:
   1. git commit -m “blank file”