Introduction to Git and GitHub

Temporal Ecology Lab

August 11, 2025

1 Version Control in the Lab

Version control is an important practice in software engineering to track changes in project files, mostly code. In the lab, tracking and documenting changes are important as we develop lots of R code for data processing and analysis as well as for manuscript revisions. We usually apply version control to individual projects in the form of repositories. A repository is basically a folder in your laptop and all of its contents. We have a general structure for each repository as below:

- analyses/ - files for data processing, analysis, and presentation
  - figures/ - files for visualizing data
  - input/ - raw data files used for analysis, should be a subset of data/
  - output/ - files for all processed data, whether intermediate or final
  - code files
- data/ - data as collected or received, files are never modified unless actively collecting data
- docs/ - notes, meeting logs, manuscript files

Bigger projects can have a more complicated directory structure, but the base structure is retained. We also use an online host to store these repositories such that we can designate multiple individuals to make changes to a single repository, tracking changes from all collaborators.

2 Git and GitHub

Our version control system of choice is Git and the online repository host GitHub for collaboration and project management. The subsections below provide a guide on how to start using Git and GitHub. Note, Git and GitHub have a lot more features than those enumerated below, but they are enough for our purposes. Go through this exercise first to start working on code-based projects in the lab, and pursue understanding Git and GitHub more after, if desired.

2.1 Installing Git

1. Install Git using instructions over at https://git-scm.com/downloads for your operating system.
2. Make a GitHub account over at https://github.com/

3. Set your username and email address which are used to identify authors of changes, also used by GitHub to associate changes to corresponding accounts

4. (Recommended) Set a system for authentication so inputting credentials for connecting to GitHub is only required for the first one, Git Credential Manager (GCM) for HTTPS for local laptops (SSH for lab servers, link here)

2.2 Making repositories

1. Navigate to the folder to be tracked as a repository

2. Apply version control through 'git init'

3. Make a remote repository at GitHub

4. Set the GitHub repository as the remote repository of the local repository through 'git remote'

5. If the remote repository exists already, navigate to the folder where the local repository folder is desired and clone the repository through 'git clone'

2.3 Making changes

Tracked changes in Git are packaged as commits, similar to a version number (or ID), which can accommodate diverging and converging versions. There is an intermediate step between changes and commits usually called a staging area. Changes must first be moved to the staging area then made into combined into a single commit.

1. Make changes (edit existing file, make new file, etc.) to repository

2. Move changes to staging area through 'git add'

3. Make a commit through 'git commit'

4. If moving a change backwards, use 'git restore'

5. If unmaking a commit, use 'git reset'

2.4 Getting changes from and making changes to remote repository

1. (Recommended) Start each session of making changes by retrieving changes applied to the remote repository through 'git pull'

2. After making commits locally, apply these changes to the remote repository through 'git push'

2.5 Working with branches

Branches are separate workspaces which share commit histories until the point of divergence. They are usually used when making changes to parts of the project without making changes to the main branch. We don't usually work with them but knowing a bit about branches can help. The related commands are below:

- git branch
- git checkout
- git switch

2.6 Resolving merge conflicts

When working with multiple collaborators, there can be conflicts when trying to apply changes to the remote repository. For example, a collaborator pushed a commit to the remote repository while changes are being made to a local repository. When the time comes to push these commits to the remote repository, there may be conflicts in the changes. Usually, these are first resolved locally before being applied to the remote repository. The related commands are below:

- git rebase
- git merge
- git fetch