2. Reproducible Science with Git and GitHub

Z620: Quantitative Biodiversity, Indiana University

OVERVIEW

During QB, you will have access to a dedicated Mac desktop computer located in Fine Arts 004. On this classroom computer, you will use the **cloud version** of RStudio (https://posit.cloud), which is accessible from anywhere and comes pre-installed with Git. You will use Git and GitHub on the lab computer to manage QB **worksheets**, which contain in-class **exercises** and **assignments** that will be submitted via GitHub. Today, you will learn how to **fork** and **clone** a GitHub repository. Additionally, you will practice how to **commit** and **push** changes, enabling you to later retrieve your work using the **fetch** and **merge** commands. In QB, assignments will be submitted to instructors for evaluation through a **pull request** on GitHub.

Using Git on Posit Cloud

1) Join the QB-2025 Workspace on Posit

1. Sign in to your free Posit.cloud (https://posit.cloud/plans/free) account.
2. On Slack and Canvas, we will provide you with an invitation you to join the QB-2025 Workspace. Click this link and accept the invitation.
3. After successfully joining, you should see in the left navigation bar, under **Spaces**, both *Your Workspace* and *QB-2025*. Navigate to the *QB-2025* space by clicking it.

The difference between *Your Workspace* and the *QB-2025 Workspace* lies in the available computational resources. When using a free Posit.cloud account (i.e., *Your Workspace*), memory is limited to 1 GB RAM. Some of the code we will execute in QB requires more. In the *QB-2025 Workspace*, you will be allocated additional RAM to support the completion of your assignments.

Please note that while files in *Your Workspace* are private and associated with your free account, instructors can view all projects created in the QB-2025 Workspace. This is because we serve as the administrators of "QB-2025," and privacy settings do not apply in this shared environment. Last, the Posit Cloud subscription charge is proportional to usage. Therefore, we request that you only use the QB-2025 Workspace for class-related work.

2) Create an empty R project & prepare Git

1. In the *QB-2025 Workspace*, click the upper right button that says New Project. Select "New RStudio Project", which will create an empty project. Name this project "QB2025_Lastname" (e.g., QB2025_Smith).

2. By default, all new projects are set to "Private." Please ensure projects remain private so that **only you have editing access**. Assignments will still be published publicly via Git, but keeping projects private helps prevent accidental edits to the wrong assignment, especially since Posit.Cloud autosaves automatically.

3. While Git comes pre-installed with Posit Cloud, we are going to make a few changes to the default settings of Git (configurations) so your Posit.Cloud workspace and Git interact with each other more smoothly.

Inside your new R project (QB2025_LastName), you will see just below the file toolbar an area to type with three tabs: “Console”, “Terminal”, and “Background Jobs”.

4. Select “Terminal”.

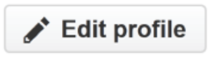
The terminal, which is often called the “command line”, allows a user to directly interact with the computing system through UNIX commands. This can be a useful way to change the default “under-the-hood” settings of a computer or to initiate code.

5. Type the following lines into the terminal, line by line. Press enter at the end of each line to submit the line to the terminal. For variables like user.name and user.email, make sure to put in your own information!

```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
git config --global push.default simple
git config --global credential.helper store
git config --global core.editor "nano"
git config --global core.autocrlf input
```

3) Open your Git Account

1. Navigate to https://github.com.
2. Sign in with your Username and Passphrase.
3. On the top right of your screen, click on your Username.
4. Click on the Edit Profile Icon to edit your profile.



You are now ready to *Git* !!!

4) Fork and Clone a Repo

1. Navigate to the QB student organization: https://github.com/QBstudents.
2. Click on the repository (repo) with your name.
3. Fork your repo by clicking on the Fork icon in the top right of your screen:

   You should now see the repo on your GitHub page.

4. Clone the repo into your empty RStudio project (named “Github”) using the command line (Terminal). Replace “username” with your GitHub username and “repo” with the name of the repo in the QBstudent organization (e.g., QB2025_Smith).

```
git clone https://github.com/username/repo
cd ./repo
git status
```

The repo should have downloaded into your Posit cloud project. The status should stay "all up to date".

5. Check and update remote repo location. The following commands will set the location of your **upstream remote repository** on the QB Course GitHub site. (In the example below, you will need to replace "repo" with the name of your QB Repository.)

```
git remote -v
git remote add upstream https://github.com/QBstudents/repo
git remote -v
```

You can copy and paste the URL for your upstream repo from the GitHub website.

6. Using RStudio, open and edit the README.md file contained in your student repository. Immediately following the text that already exists in this file, supply the following: Enter your name and email address. Write a short biography (~1 paragraph) and share something about yourself that we might not otherwise know. List three to five course expectations. You can learn more about Markdown formatting (e.g., making ordered lists) here: https://guides.github.com/features/mastering-markdown/. When you are done, save the close the document.

5) GitHub Permissions

Before going any further, we need to set up our GitHub permissions. Somewhat recently, GitHub made the decision to move away from passwords. Now, automatic token authentication is used. Here are the instructions for setting this up. 1. Go to your GitHub site (e.g., https://github.com/jaytlennon).

2. Click on your avatar in the upper right hand corner.
3. Click on "Settings".
4. Click on "Developer settings" on the left-hand side towards the bottom.
5. Click on "Personal access tokens" on the left-hand side.
6. Choose the "Tokens (classic)" option.
7. You want to assign "read and write" permissions for all repositories.
8. Assign a token name and description for 90 days.
9. After generating token, make sure you copy and save; you will need this for next step.

In the process above, token generation and verification is linked to Duo, which can be enabled in "Settings" under "Passwords and authentication". Duo will generate a GitHub associated six-digit code that you will be asked to supply to github.com

6) Commit and Push

Now that your permissions are established, we can commit our changes and pushing them upstream to the remote repository with the following instructions:

1. You just changed the README.md file. To record this, we are going to create a **commit** or “revision” with the following code. The `status` function identifies changes you made, the `add` command adds a file to the “staging area”, and the `commit` command sends the changes to your repository along with a message (-m).

   ```
   git status
   git add ./README.md
   git commit -m "Updated README.md with student information"
   ```

2. Now we will **push** the changes to GitHub. But before we do this, we always want to check for any changes that others have made; otherwise we can create **conflicts** that need to be fixed. We check for these changes using the git commands **fetch** and **merge** as follows:

   ```
   git fetch upstream
   git merge upstream/main
   git push origin
   git status
   ```

After “git fetch upstream”, Terminal should prompt you for username and password. Username is your github username (ex: “jaytlennon”). The password is the **personal access token** you just generated, not the password you use to login to github.

You should now see the repo, including your recent changes, on your GitHub webpage.

7) Pull Request

After pushing your commit, you need to follow up by making a **Pull Request**.

1. Navigate to your GitHub webpage to make sure that the file (e.g., README.md) was uploaded correctly. If so, submit a pull request to submit your file to the course instructors.

   The course instructors can now merge and see your changes.

8) Getting Up to Date

2. To get new worksheets, you will *pull* your upstream repository to your local computer using the **fetch** and **merge**. This will allow any updates your instructors have made to be merged with your local documents. In addition to pulling your upstream repo, you always want to push any updates to your origin.

   ```
   git status
   git fetch upstream
   git merge upstream/main
   git push origin
   git status
   ```

During this course, you will receive **worksheets** and submit all **assignments** using these methods. In addition, you will use Git and GitHub to contribute to assignments in class and on your personal computers.

9) Basic Git and GitHub glossary:

The terms below will be used throughout this class. Some are commands that you will type into the terminal window. All are defined with respect to how we will be using Git and GitHub in this class.

Term	Meaning
repository	The collection of files and folders that compose a project.
upstream	Refers to the central repository, e.g., the version managed by your instructors.
origin	Refers to your on-line version of the class repository.
fork	Create a version of the class repository in your IU-GitHub account.
clone	Create a copy of 'origin' on your computer. This is the version you will edit.
fetch	Download changes that have been made to an online repository.
merge	Merge changes with your local version.
pull	'fetch' + 'merge'. 'pull' executes 'fetch' and 'merge' but give less freedom of control.
staging area	A file that Git uses to store information about your changes.
add	Having edited, removed, or added files, this command will add your changes to the staging area.
commit	Having added your changes, this command will save a picture of what your files looked like at that moment.
push	Having committed your changes, this command will update 'origin'.
pull request	Having updated 'origin', request that these changes be pulled into 'upstream'