BOSTON UNIVERSITY

Git and GitHub for R Projects with RStudio

Lukas M. Weber, PhD

Assistant Professor
Department of Biostatistics
Boston University

September 18, 2025

lmweber.org

Overview

1. Introduction to Git and GitHub
2. Use cases and examples
3. Interactive demo and exercises using RStudio

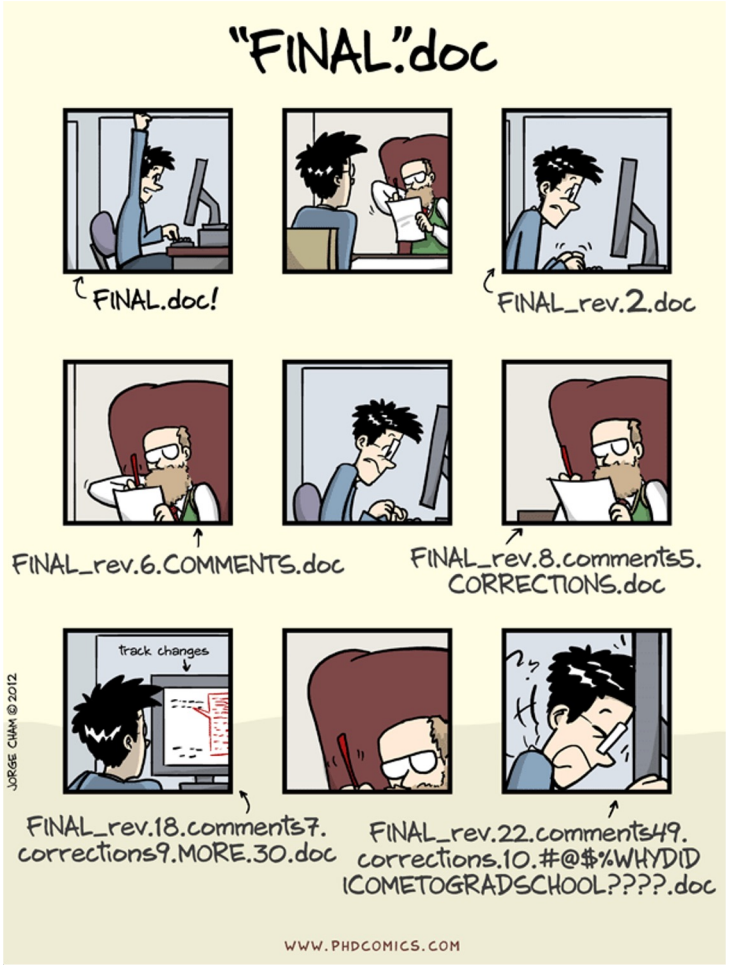
Git and GitHub

System for managing code

- version control (tracking changes)
- collaboration
- organizing projects
- publishing and sharing code
- reproducibility

References and additional materials:

- Software Carpentry lesson on "Version Control with Git": https://swcarpentry.github.io/git-novice/



Git and GitHub

Git and GitHub refer to two separate but related concepts:

Git: command-line utility

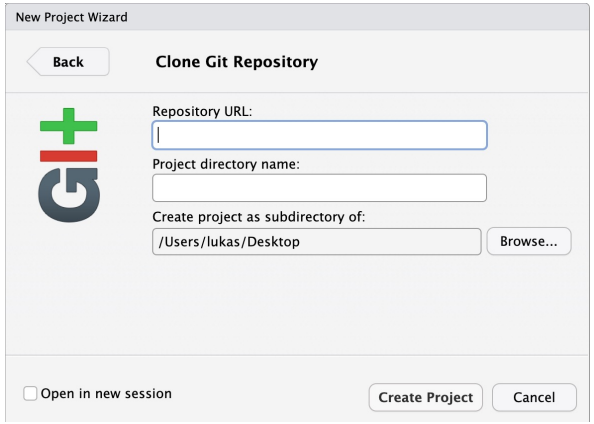
- available from command-line as well as (more recently) interfaces in development environments such as RStudio
- developed in 2005 by Linus Torvalds (creator of Linux operating system) as an improvement to earlier version control systems such as svn

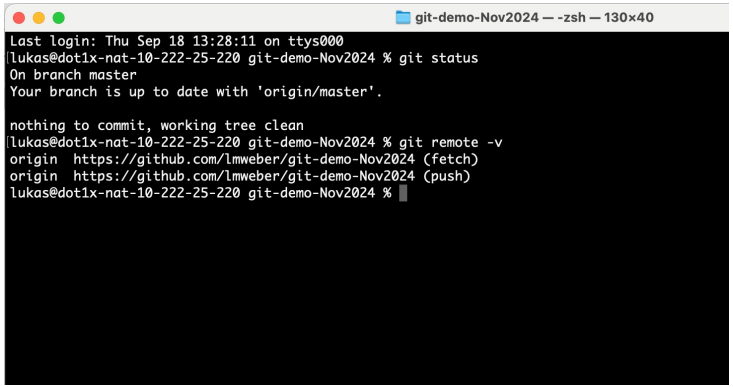
GitHub: commercial website and collaboration environment

- hosting service for Git repositories, allows collaboration by multiple users
- started in 2008 and acquired by Microsoft in 2018
- alternatives also exist (e.g. GitLab) but GitHub is most popular

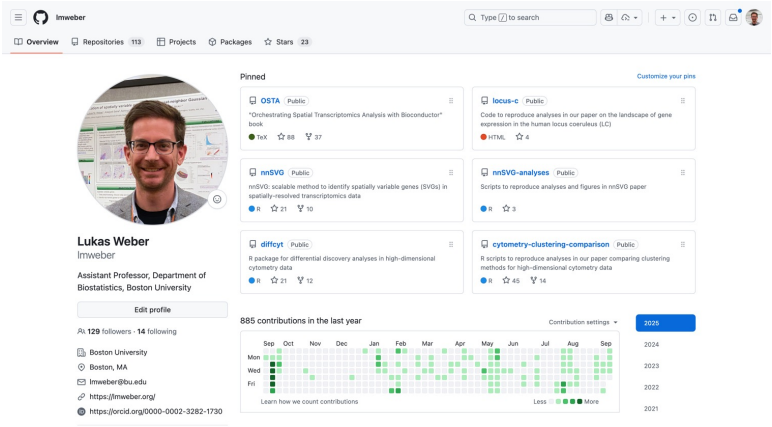
Git and GitHub

Git





GitHub



Why use Git and GitHub?

1. Version control

- version control lets you keep track of updates / edits to your code scripts (or text) over time
- nothing that is “committed” to version control is ever lost
- can go back to look up previous updates, undo changes, go back to an earlier version
- have a record of who made changes and when in a collaborative project

More details:

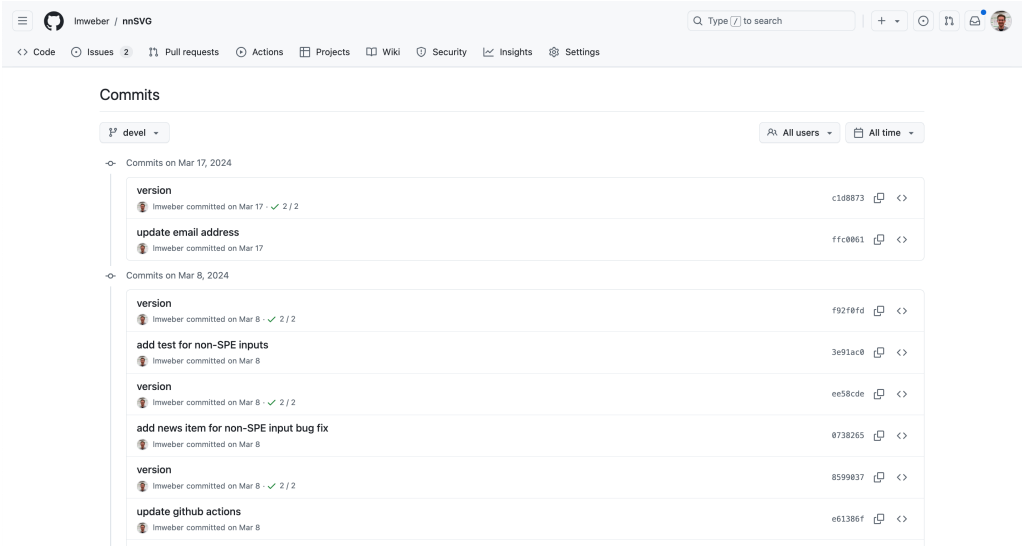
Software Carpentry lesson: https://swcarpentry.github.io/git-novice/

Terminology

“Commits”

- specific updates to code scripts
- includes a record of the author, email address, timestamp, and a “commit message”
- can look up “commit history” on GitHub

Example
https://github.com/lmweber/nnSVG

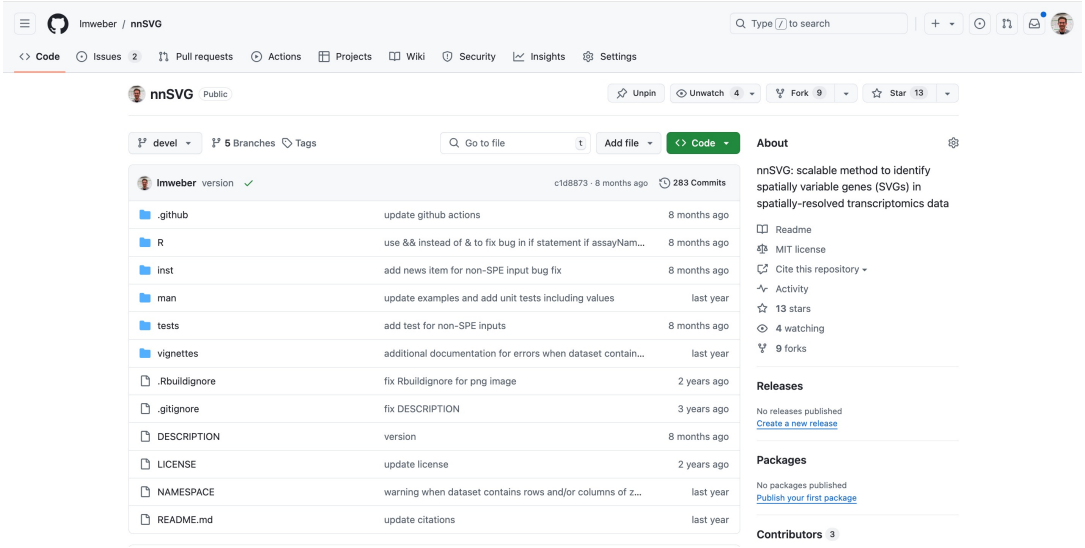


Terminology

“Repository”

- project containing one or more version-controlled files in a directory
- multiple people can contribute / collaborate
- can be stored locally (on your computer) and linked publicly on GitHub

Example
https://github.com/lmweber/nnSVG

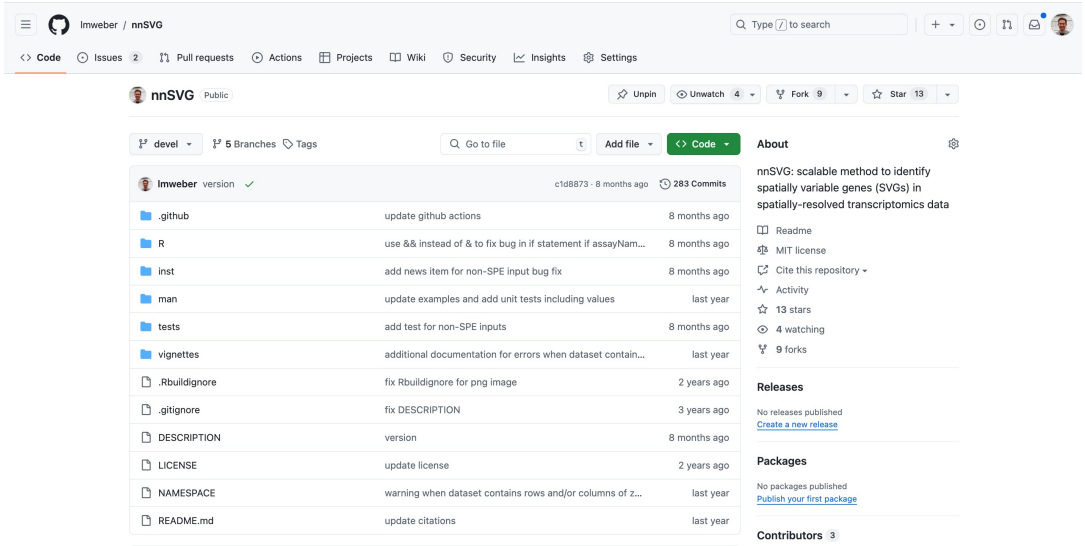


Terminology

“Local” vs. “remote” repositories

- local repository: stored in a directory on your computer, used for your own work
- remote repository (e.g. on GitHub): hosted on a server, enables collaboration, also used as backup or to keep track of your own projects

Example
https://github.com/lmweber/nnSVG



Terminology

“Push”

- share local changes to a remote repository
- updates the remote repository with the local commits

“Pull”

- retrieve changes from a remote repository
- updates the local repository with the changes from the remote repository

Questions and setup

Questions?

Set up your GitHub account if you have not already

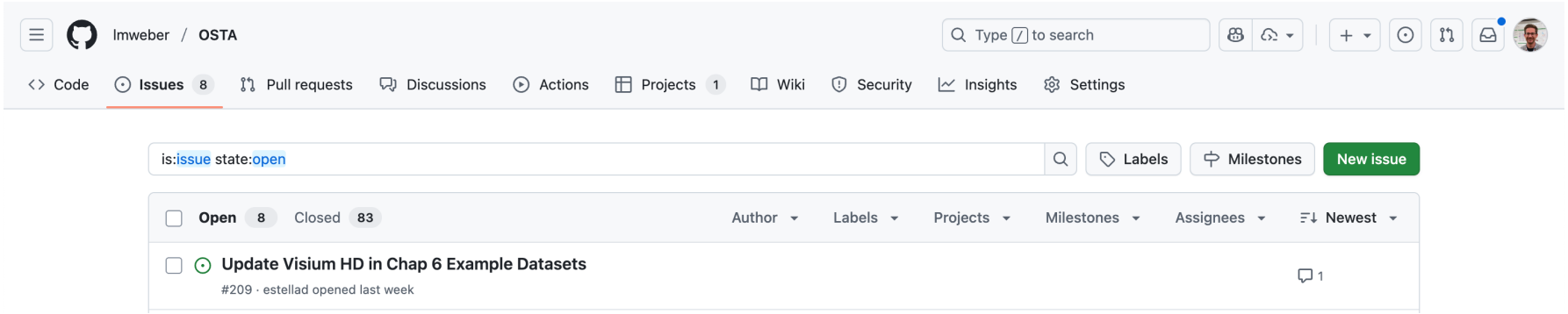
Use an email address that you will have long-term access to (e.g. Gmail)

Set up / update R and RStudio

2. Collaboration

- Multiple people can contribute to the same Git repository
- GitHub provides a convenient interface for communication via “Issues”

Example (GitHub Issues): https://github.com/lmweber/OSTA



3. Publishing code

R packages

- can store files for an R package in a GitHub repository
- can install R packages directly from GitHub

```
library(remotes)  # or library(devtools)

install_github("lmweber/nnSVG")
```

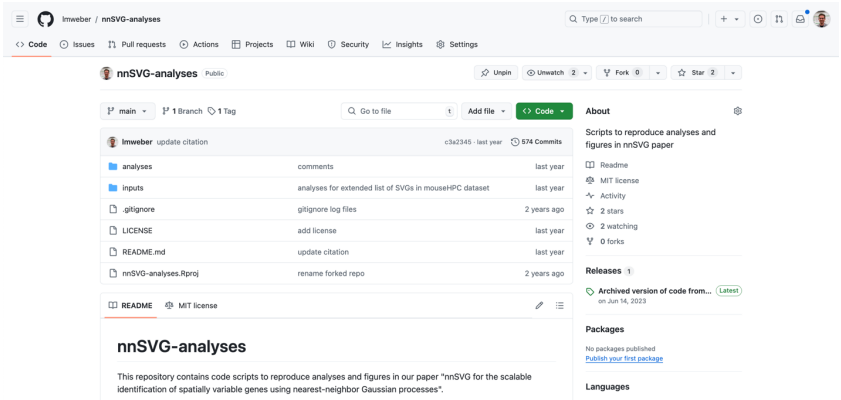
→ Example: https://github.com/lmweber/nnSVG

4. Reproducibility

Analysis code repositories

- store analysis code from a project
- can include link in published paper for reproducibility
- also useful as backup of important code

→ Example: https://github.com/lmweber/nnSVG-analyses



5. Coding best practices

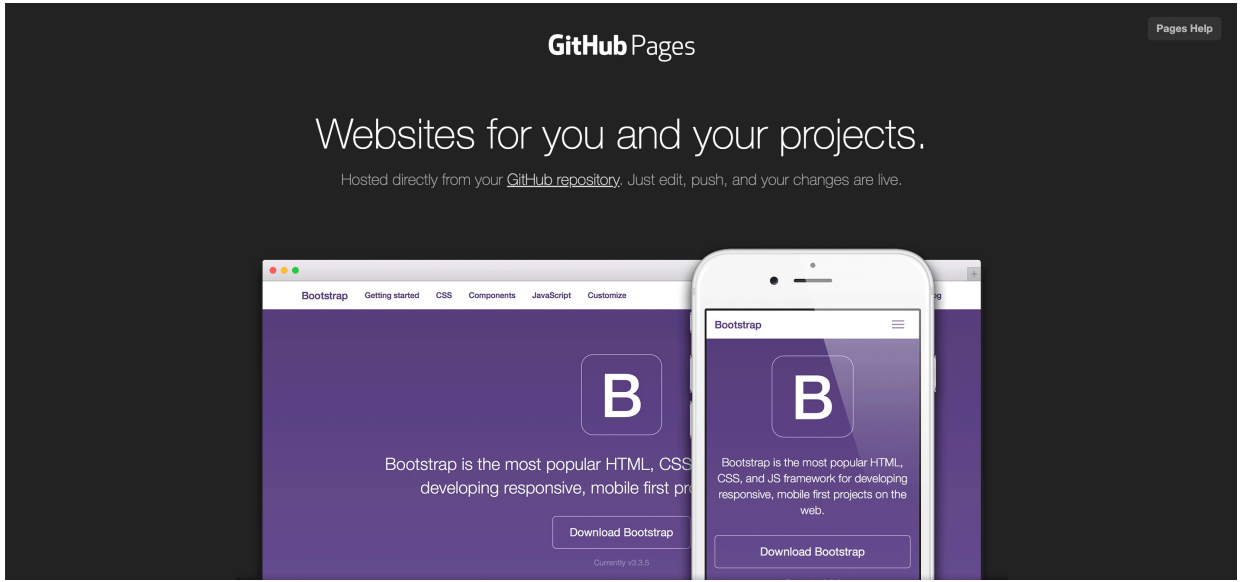
Working with Git and GitHub makes it easier to follow good coding practices

- Modularity: push “commits” to GitHub each time you have a small piece of code done (or at the end of the day / end of a coding session), instead of building up long / complicated scripts
- Reproducibility: private or public repositories
- Documentation: documentation files, readme pages

6. Websites and portfolio

GitHub Pages

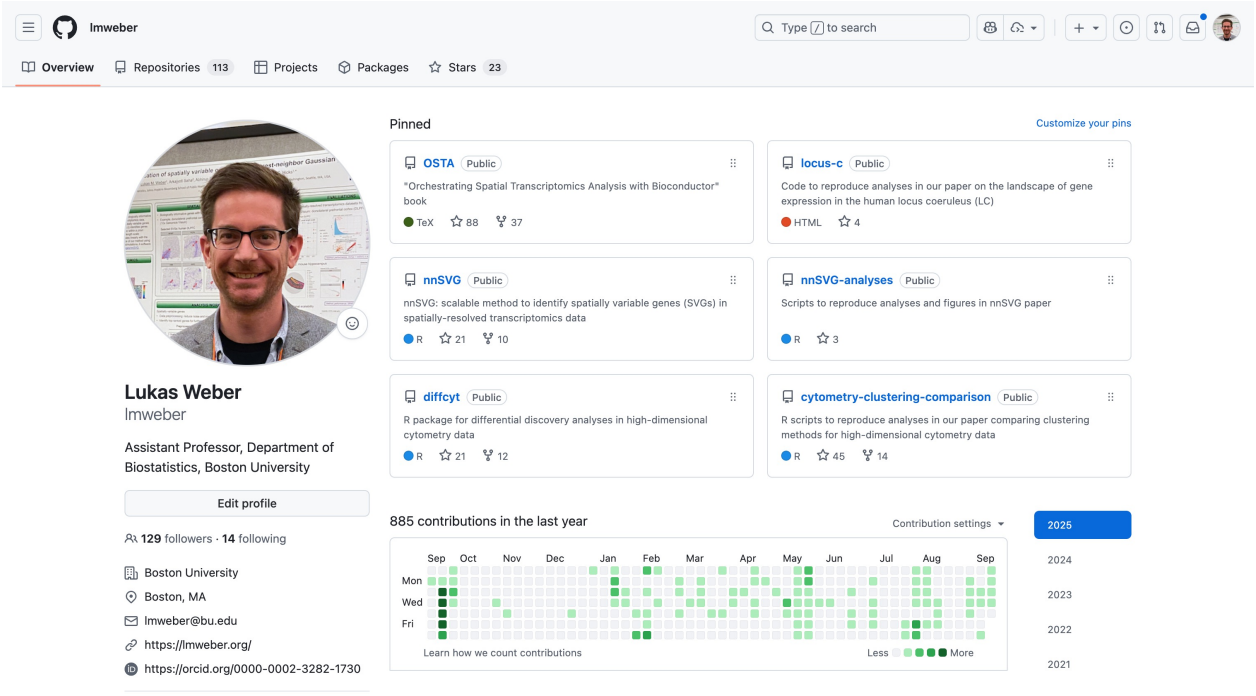
- GitHub provides a free hosting service for websites
- templates available



6. Websites and portfolio

GitHub profile page

- can use as portfolio of coding projects, e.g. for job applications



Interactive demo using RStudio

Setup

- Create GitHub account
- Install / update R and RStudio
- Set up Git locally on your computer

Setting up Git locally on a computer for the first time can sometimes be complicated

After Git is set up, we will access it in RStudio

Setup

Check if you already have Git installed on your computer

- type `git config --global --list` in RStudio Terminal window (or Terminal)

If not, follow Git installation instructions from Software Carpentry lesson: https://swcarpentry.github.io/git-novice/02-setup.html

You only really need to set up the following 3 settings:

- `git config --global user.name "Your Name"`
- `git config --global user.email "your@email.com"`
- `git config --global core.editor "nano"`

Setup

Next, enable authentication settings in your GitHub account

Generate a “Personal Access Token” (more secure alternative to password)

- Settings → Developer settings → Create Personal Access Token (classic)
  - Select “No expiration” for expiration date
  - Select all permissions for “repo”, “user”, and “workflow”
- Save this and paste it in like a password whenever Git asks for it (using “https” authentication)
- Alternatively, set up “ssh” authentication (more secure / more complicated to set up, but only need to set up once per computer)
  - Check your GitHub account has 2-factor authentication enabled (e.g. Google Authenticator app on your phone)
  - Create an “ssh key” in RStudio Git settings after setting up an RStudio project
  - Paste the “public” part of the ssh key into settings in your GitHub account
  - Check your local Git repository is using ssh authentication by typing “git remote -v”

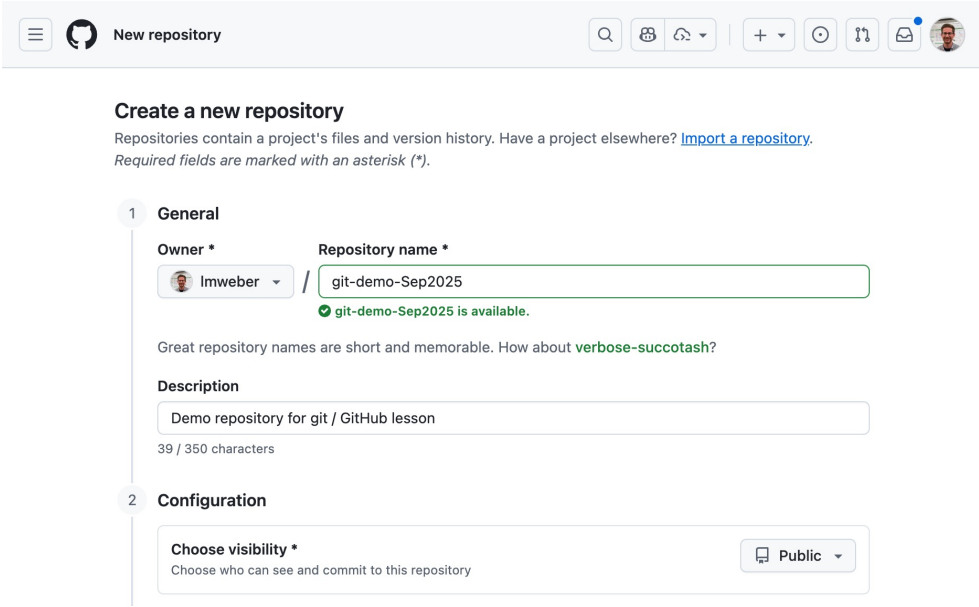
Create a repository on GitHub

Can either create a repository on GitHub and then download a copy on your computer, or create it locally and then link it on GitHub

We will do the first option (create it on GitHub)

→ Demo

Note various settings / options available



Create a repository on GitHub

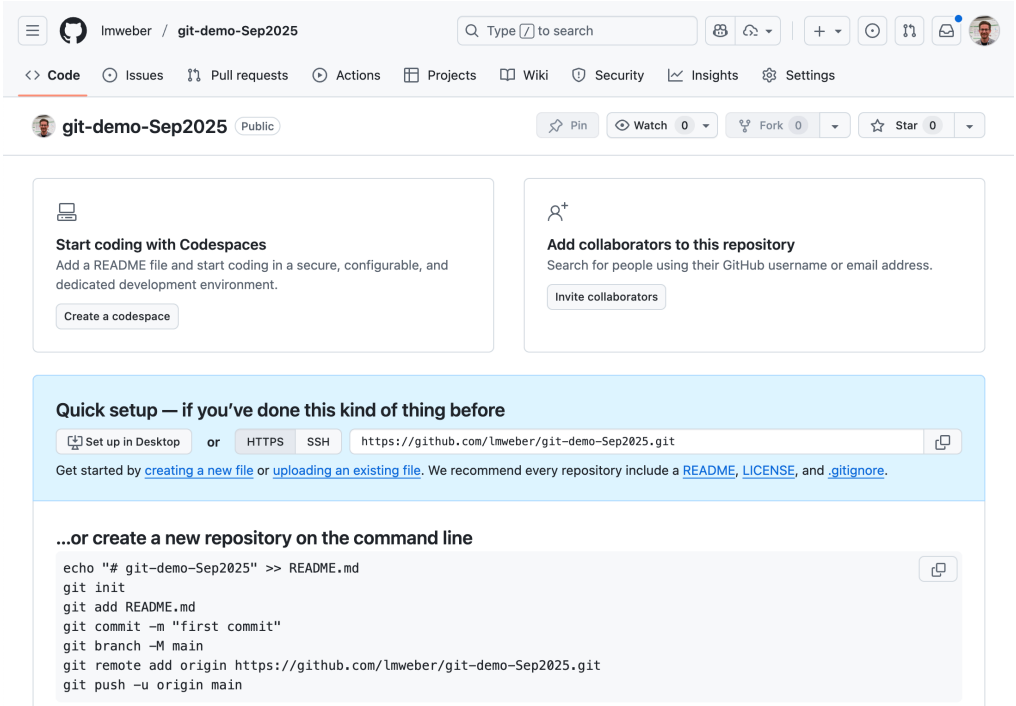
Can either create a repository on GitHub and then download a copy on your computer, or create it locally and then link it on GitHub

We will do the first option (create it on GitHub)

→ Demo

GitHub then provides instructions for how to “clone” the repository locally using the command line

We will use RStudio instead



More terminology

“Clone”

- download a local copy of a remote repository (e.g. from GitHub) onto your computer

“Fork”

- create a copy / duplicate of a remote repository, e.g. a copy of someone else’s repository in your own GitHub account, where you can make your own changes
- the new fork will branch off from the current version of the existing repository, so you can add changes while also keeping track of the original (“upstream”) repository

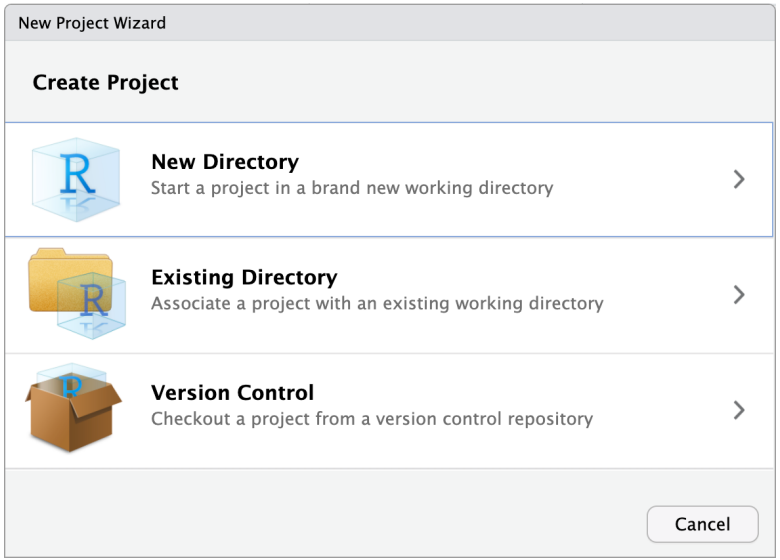
Create RStudio project

Now create an RStudio “Project” and link the repository from GitHub

RStudio projects are a convenient way to organize code and other files from a project in a single directory, e.g. multiple analysis scripts for a data analysis project

→ Demo

Select “Version Control” and “Git” to link a repository from GitHub



Create RStudio project

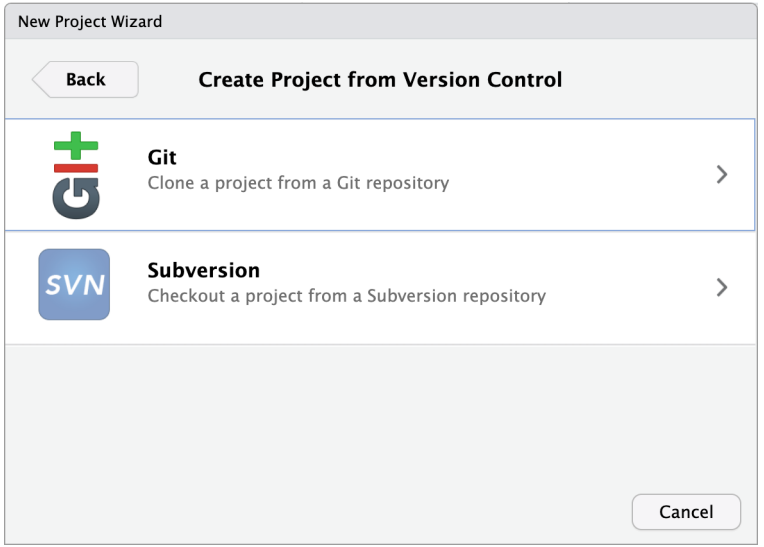
Now create an RStudio “Project” and link the repository from GitHub

RStudio projects are a convenient way to organize code and other files from a project in a single directory, e.g. multiple analysis scripts for a data analysis project

→ Demo

Select “Version Control” and “Git” to link a repository from GitHub

Paste in the repository URL



Create RStudio project

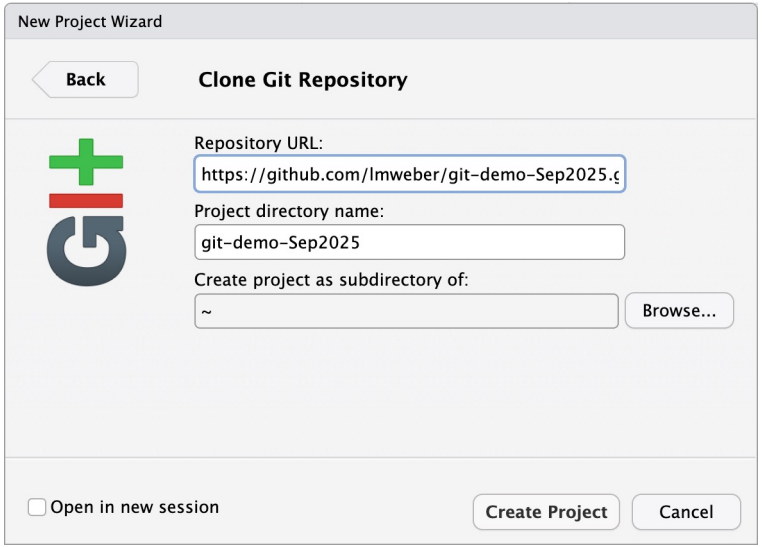
Now create an RStudio “Project” and link the repository from GitHub

RStudio projects are a convenient way to organize code and other files from a project in a single directory, e.g. multiple analysis scripts for a data analysis project

→ Demo

Select “Version Control” and “Git” to link a repository from GitHub

Paste in the repository URL

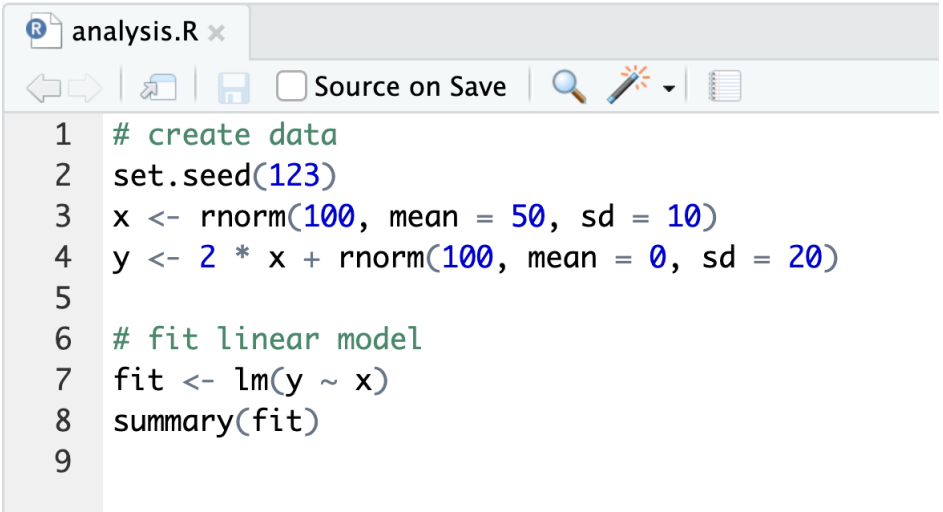


Add files to repository

Now we can add files locally in the project directory, and push them to the repository on GitHub

→ Demo

- File > New File > R Script
- Add some code and save file

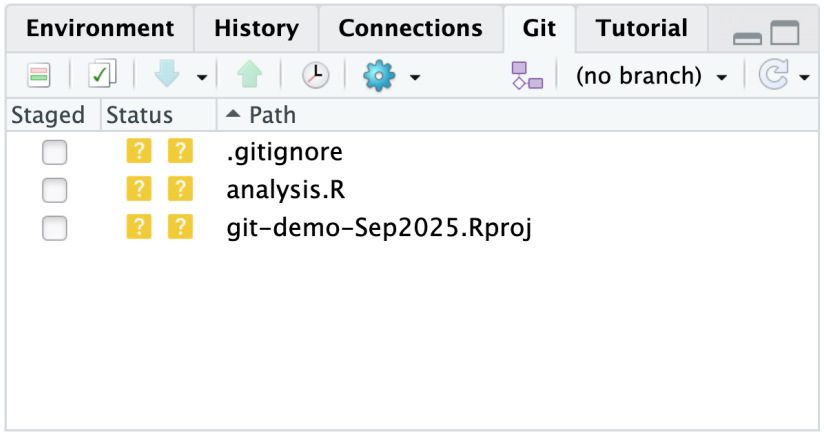


Add files to repository

Now we can add files locally in the project directory, and push them to the repository on GitHub

→ Demo

- Check changes
  - Open the “Git” tab in the top-right corner of RStudio
  - You should see the new R script file listed under “Unstaged Changes”
  - “Stage” the changes: prepares them to be included in the next commit



More terminology

“Stage”

- select which changes to include in the next commit, and prepare changes for the commit by adding them to the “staging area”

“Commit”

- record a snapshot of changes to the repository – includes the changes along with a record of author, email address, timestamp, and “commit message”

“Push”

- “push” commits from local repository to remote repository (i.e. updating remote repository)

“Pull”

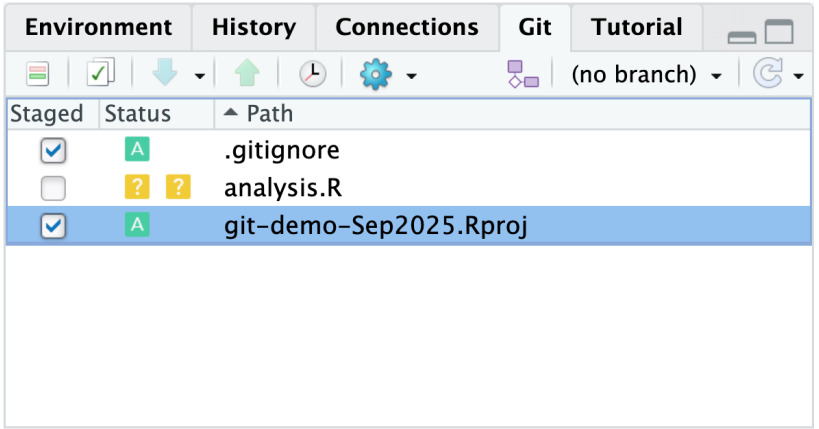
- “pull” commits from remote repository to local repository (i.e. updating local repository)

Add files to repository

Now we can add files locally in the project directory, and push them to the repository on GitHub

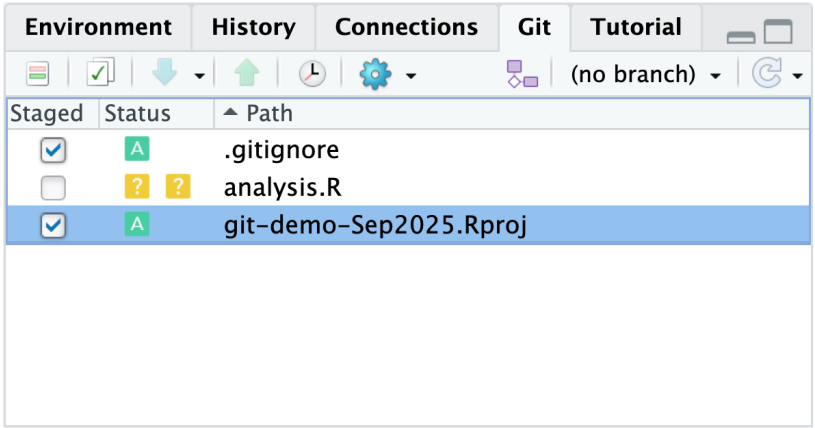
→ Demo

- Check changes
  - Open the “Git” tab in the top-right corner of RStudio
  - You should see the new R script file listed under “Unstaged Changes”
  - “Stage” the changes: prepares them to be included in the next commit



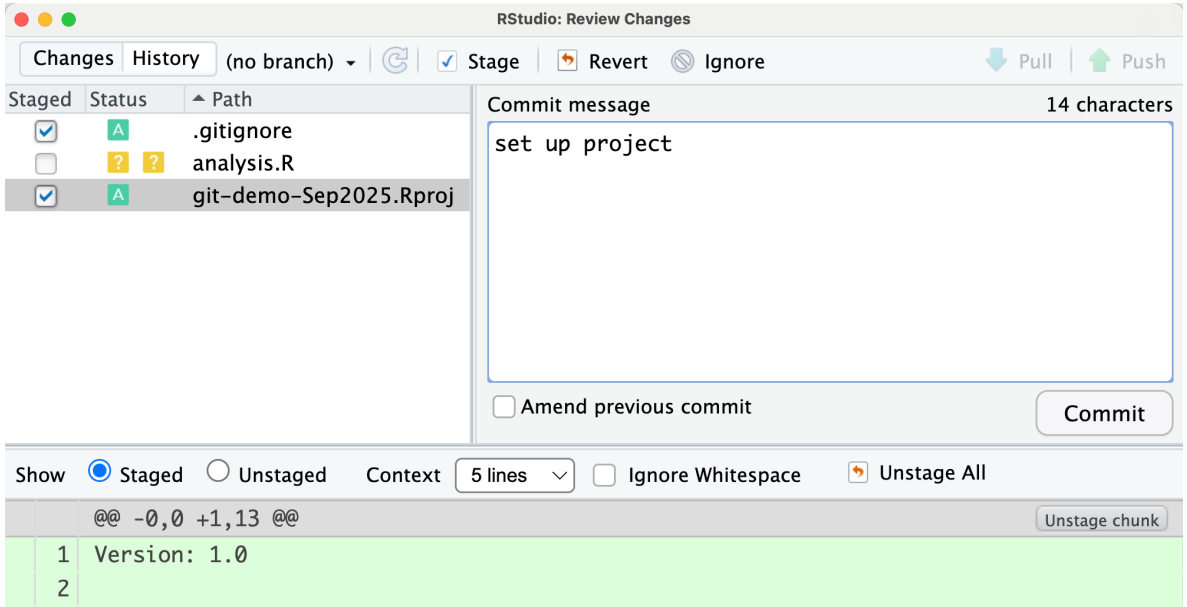
Add files to repository

- Save, stage, commit, and push the files
- RStudio also provides an opportunity to check the changes before committing and pushing ("Diff" button, or opens automatically)
- Note that we also have some additional files from creating the RStudio project, so we will split the updates into two commits (good coding practices: modularity)



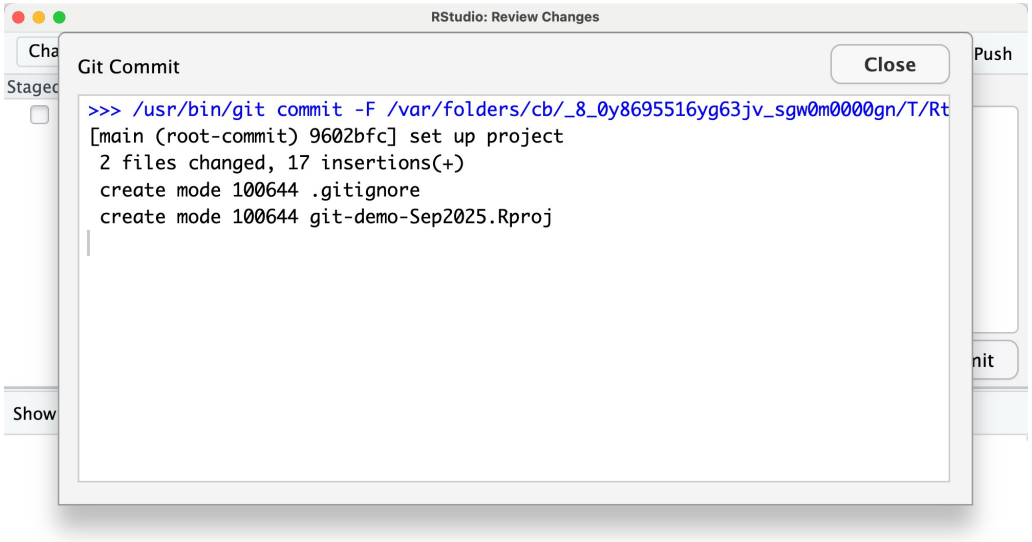
Add files to repository

- Save, stage, commit, and push the files
- Click on “Commit”, add a “Commit message”, then “Commit” again



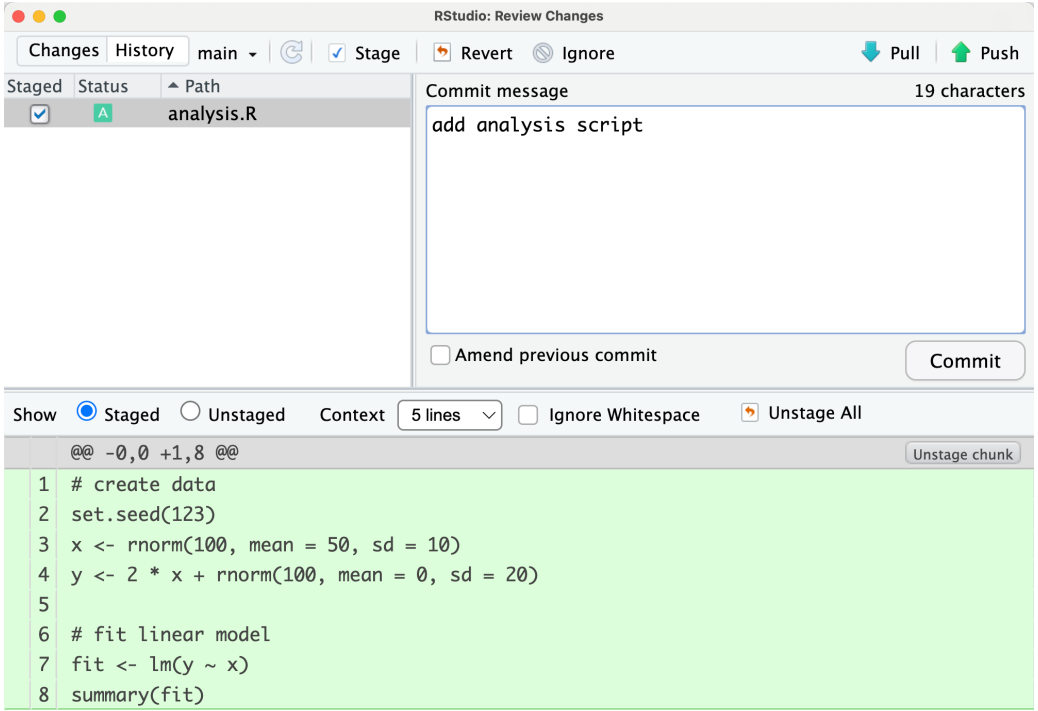
Add files to repository

- Save, stage, commit, and push the files
- Click on “Commit”, add a “Commit message”, then “Commit” again



Add files to repository

- Save, stage, commit, and push the files
- Click on “Commit”, add a “Commit message”, then “Commit” again

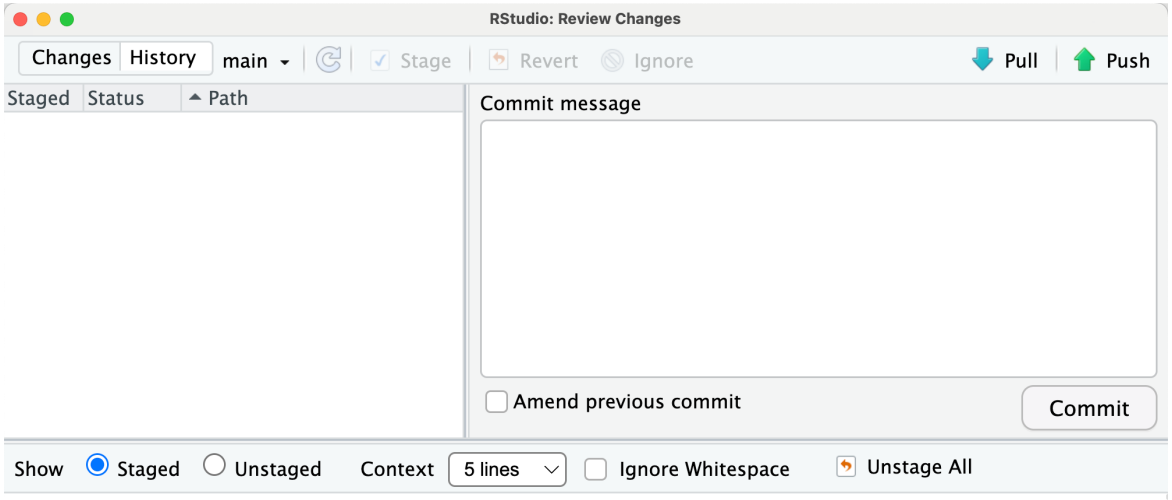


Add files to repository

- Save, stage, commit, and push the files
- Click on “Commit”, add a “Commit message”, then “Commit” again

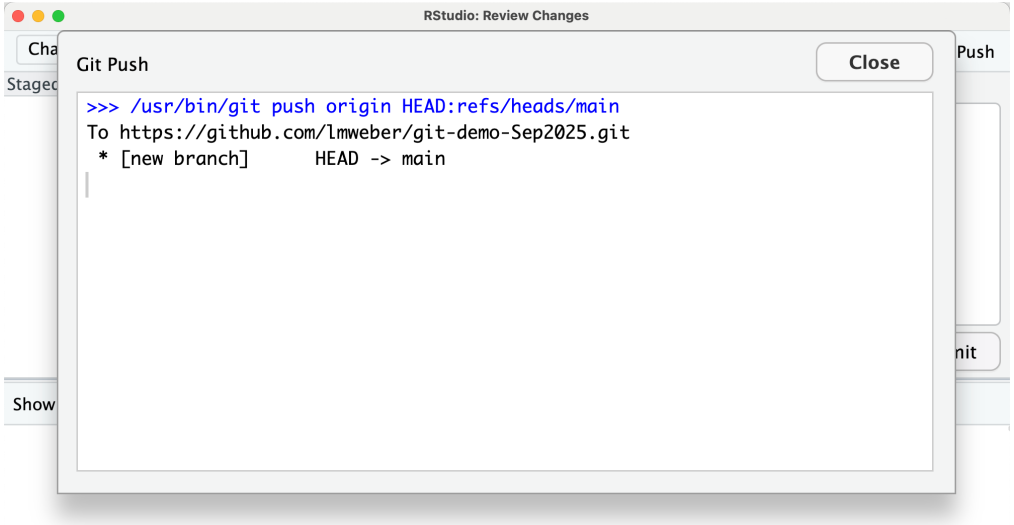
Add files to repository

- Save, stage, commit, and push the files
- Then click on “Push” (at the top-right)



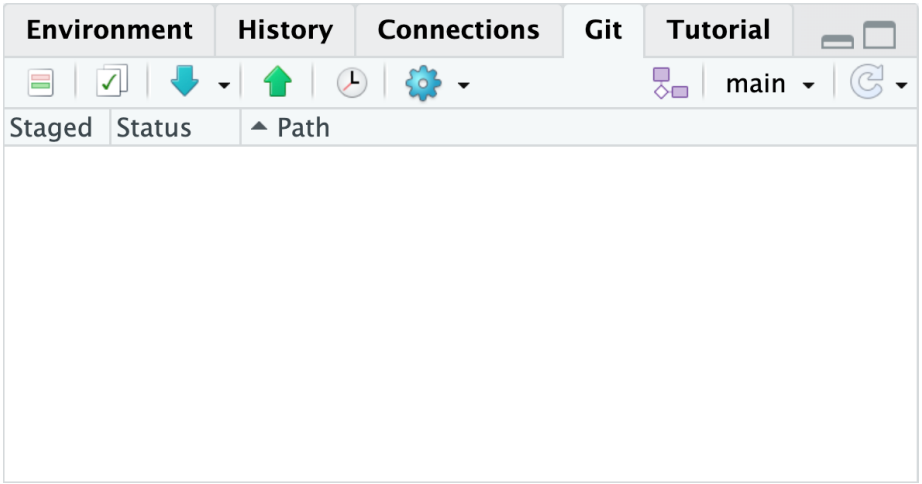
Add files to repository

- Save, stage, commit, and push the files
- Then click on “Push” (at the top-right)



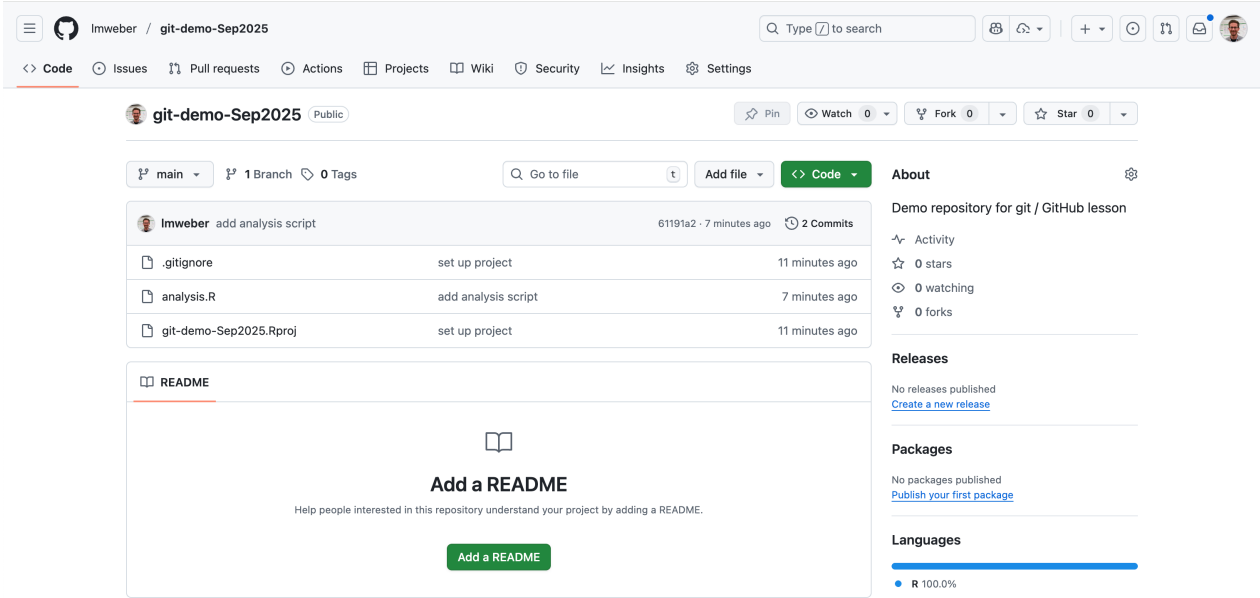
Add files to repository

- Now the “Staged” area in the Git panel shows no further changes in the repository
- Alternatively: type `git status` in RStudio Terminal



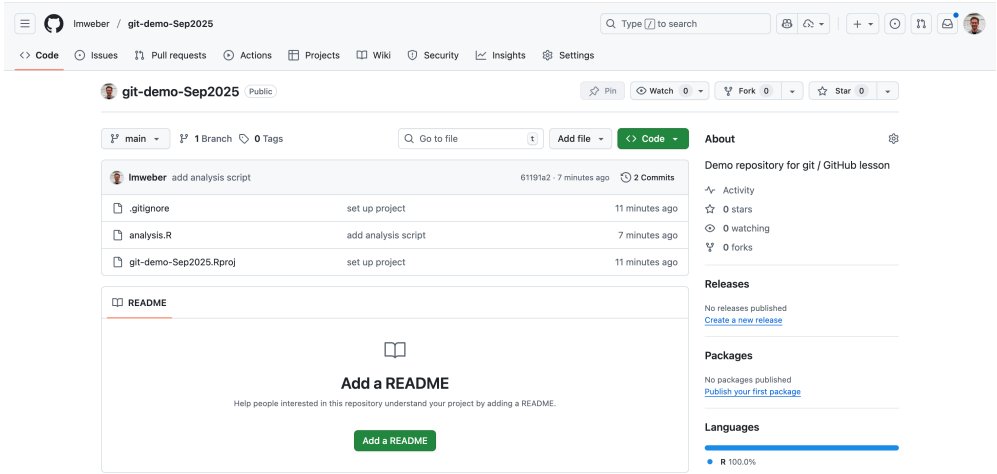
Add files to repository

- Congratulations! You have pushed your first commits to GitHub
- Check the updates on the GitHub repository page online



Add files to repository

- Congratulations! You have pushed your first commits to GitHub
- Check the updates on the GitHub repository page online



- More details: Software Carpentry lesson: https://swcarpentry.github.io/git-novice/03-create.html

Additional topics

- Pull changes from GitHub (e.g. changes added to repository by someone else)
- “Pull requests”
- “Branches”
- Merging and “merge conflicts”
- Commit history / git log
- Commit hashes (unique ID for every commit)
- Commit message style
- Types of files to commit (mainly code and text, small files)
- .gitignore file (e.g. exclude directories with large files: `data/`, `plots/`, etc)
- Using Git from command line (alternative to RStudio interface)

See Software Carpentry lesson for details: https://swcarpentry.github.io/git-novice/03-create.html

Exercises

1. Add additional code to create a plot to the analysis script, and push the changes to GitHub. Check that you can see the updated code and the commit history on GitHub.

2. Add the person sitting next to you as a “Collaborator” to your GitHub repository (you can find this in “Settings” for the repository – note this is in the repository settings, not your GitHub profile settings). Ask your collaborator to push a new commit to the repository from their laptop. Then pull these changes to your local repository on your laptop and check if your files are now updated.

3. Set up a repository for one of your courses that involves R programming, or for a research project you are working on. Add a code script for a current assignment or your current research work. Then add new commits / updates to this repository over the course of the semester to keep a record of your work. (Note that you can choose whether the repository is public or private in settings.)

Thank you!