



Git and GitHub for R Projects with RStudio

Lukas M. Weber, PhD

Assistant Professor
Department of Biostatistics
Boston University

September 18, 2025

Overview

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



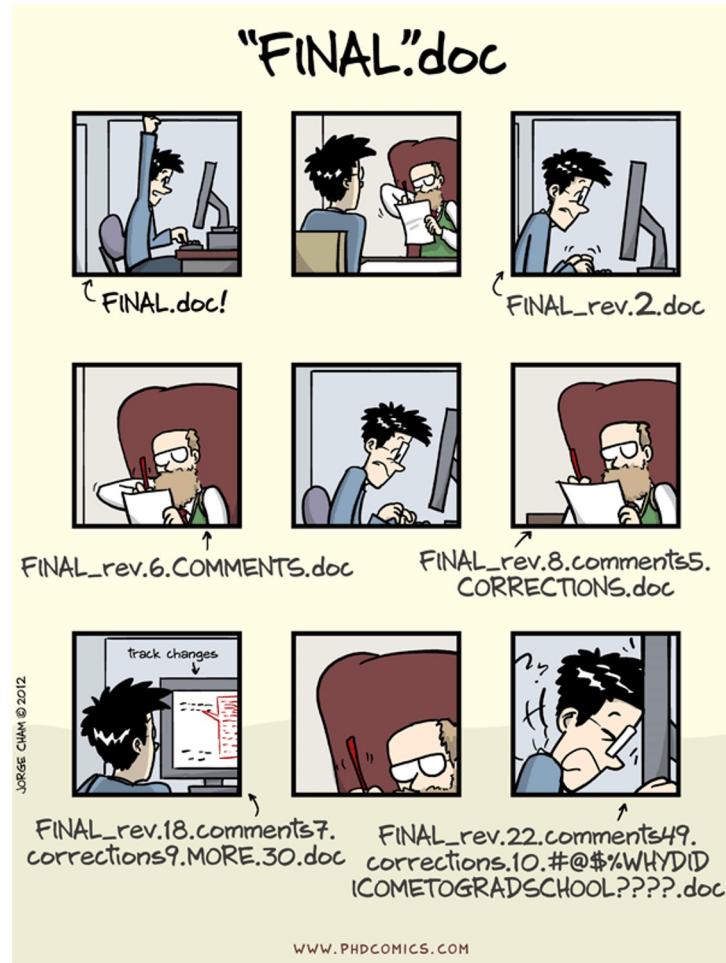
Git and GitHub

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

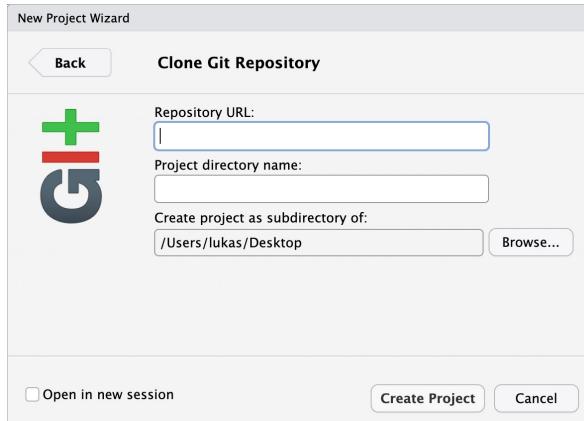
- available from command-line as well as (more recently) interfaces in development environments such as RStudio
- developed in 2005 by Linus Torvalds (creator of Linux operating system) as an improvement to earlier version control systems such as svn

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



```
Last login: Thu Sep 18 13:28:11 on ttys000
lukas@dot1x-nat-10-222-25-220 git-demo-Nov2024 % git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
lukas@dot1x-nat-10-222-25-220 git-demo-Nov2024 % git remote -v
origin https://github.com/lmweber/git-demo-Nov2024 (fetch)
origin https://github.com/lmweber/git-demo-Nov2024 (push)
lukas@dot1x-nat-10-222-25-220 git-demo-Nov2024 %
```

GitHub

The screenshot shows the GitHub profile page for 'lmweber'. It features a large circular profile picture of Lukas Weber. Below the profile picture is his name, 'Lukas Weber', and his title, 'Assistant Professor, Department of Biostatistics, Boston University'. A 'Edit profile' button is visible. On the left, there's a sidebar with links for Overview, Repositories (113), Projects, Packages, Stars (23), and a search bar. The main area displays a grid of pinned repositories: 'OSTA' (Public), 'focus-c' (Public), 'mSVG' (Public), 'mSVG-analyses' (Public), 'diffcyt' (Public), and 'cytometry-clustering-comparison' (Public). At the bottom, it shows '885 contributions in the last year' with a heatmap visualization and 'Contribution settings' dropdown. The heatmap shows contributions from September 2023 to September 2024, with colors indicating contribution frequency (green for less, dark blue for more).

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:

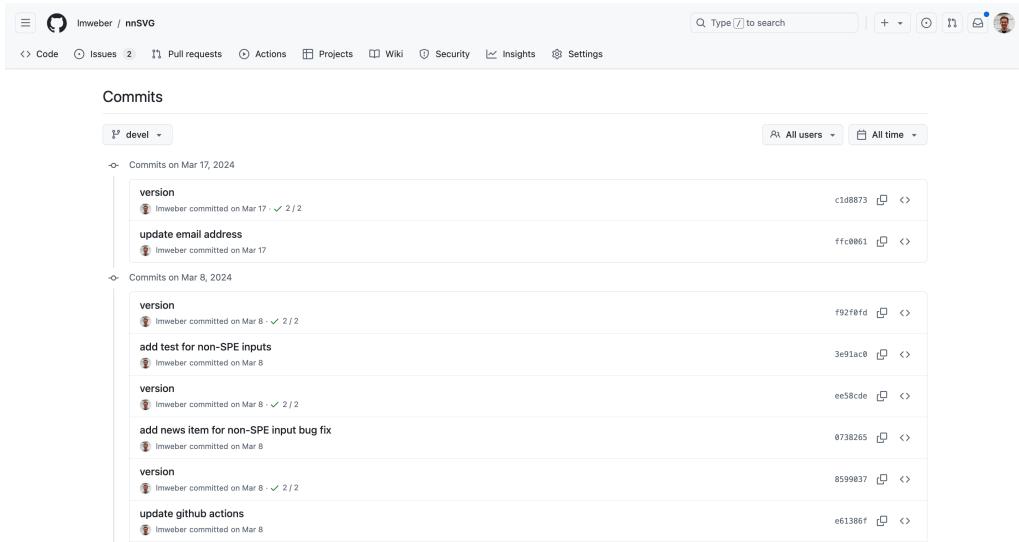
Software Carpentry lesson: <https://swcarpentry.github.io/git-novice/>



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



The screenshot shows the GitHub commit history for the repository `lmweber/nnSVG`. The commits are filtered by the branch `devel`. The history is organized into two main sections: commits made on March 17, 2024, and commits made on March 8, 2024.

Commits on Mar 17, 2024:

- version**: Imweber committed on Mar 17 - ✓ 2 / 2. Commit hash: `c1d8873`.
- update email address**: Imweber committed on Mar 17. Commit hash: `ffc0061`.

Commits on Mar 8, 2024:

- version**: Imweber committed on Mar 8 - ✓ 2 / 2. Commit hash: `f92f0fd`.
- add test for non-SPE inputs**: Imweber committed on Mar 8. Commit hash: `3e91ac0`.
- version**: Imweber committed on Mar 8 - ✓ 2 / 2. Commit hash: `ee58cde`.
- add news item for non-SPE input bug fix**: Imweber committed on Mar 8. Commit hash: `0738265`.
- version**: Imweber committed on Mar 8 - ✓ 2 / 2. Commit hash: `8599837`.
- update github actions**: Imweber committed on Mar 8. Commit hash: `e61386f`.

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

The screenshot shows the GitHub repository page for 'nnSVG' owned by 'lmweber'. The page includes a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main content area displays the repository's details: 'Public', '5 Branches', '2 Tags', and a commit history. The commit history lists 283 commits from 'lmweber' over the past 8 months, with descriptions like 'update github actions' and 'fix Rbuildignore for png image'. To the right, there are sections for 'About' (describing nnSVG as a scalable method for identifying spatially variable genes), 'Releases' (none published), 'Packages' (none published), and 'Contributors' (3). The bottom right corner of the image has the number '9'.

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>

The screenshot shows the GitHub repository page for 'nnSVG' owned by 'lmweber'. The repository is public and has 5 branches. The main code tab is selected, showing a list of commits. The most recent commit is from 'lmweber' on 'version' at 'c1d8873' 8 months ago, which updated GitHub Actions. Other commits include fixing R buildignore, updating DESCRIPTION, and adding unit tests. The repository has 13 stars, 4 forks, and 9 watches. It includes sections for Releases, Packages, and Contributors.

Code | **Issues** 2 | **Pull requests** | **Actions** | **Projects** | **Wiki** | **Security** | **Insights** | **Settings**

nnSVG Public

Code | **Issues** 2 | **Pull requests** | **Actions** | **Projects** | **Wiki** | **Security** | **Insights** | **Settings**

lmweber version ✓ c1d8873 · 8 months ago 283 Commits

.github update github actions 8 months ago

R use && instead of & to fix bug in if statement if assayNam... 8 months ago

inst add news item for non-SPE Input bug fix 8 months ago

man update examples and add unit tests including values last year

tests add test for non-SPE inputs 8 months ago

vignettes additional documentation for errors when dataset contain... last year

.Rbuildignore fix Rbuildignore for png image 2 years ago

.gitignore fix DESCRIPTION 3 years ago

DESCRIPTION version 8 months ago

LICENSE update license 2 years ago

NAMESPACE warning when dataset contains rows and/or columns of z... last year

README.md update citations last year

About

nnSVG: scalable method to identify
spatially variable genes (SVGs) in
spatially-resolved transcriptomics data

Readme | **MIT license** | **Cite this repository** | **Activity** | **13 stars** | **4 watching** | **9 forks**

Releases

No releases published | [Create a new release](#)

Packages

No packages published | [Publish your first package](#)

Contributors 3

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>

The screenshot shows the GitHub Issues page for the repository 'lmweber / OSTA'. The top navigation bar includes links for Code, Issues (8), Pull requests, Discussions, Actions, Projects (1), Wiki, Security, Insights, Labels, and Settings. A search bar is located at the top right. Below the navigation, there is a search bar with the query 'is:issue state:open' and buttons for Labels, Milestones, and New issue. The main content area displays a list of issues. At the top of this list are filters for Open (8) and Closed (83) issues. The first issue listed is '#209 Update Visium HD in Chap 6 Example Datasets', which was opened by 'estellad' last week. There is a comment icon next to the issue title.

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")
```

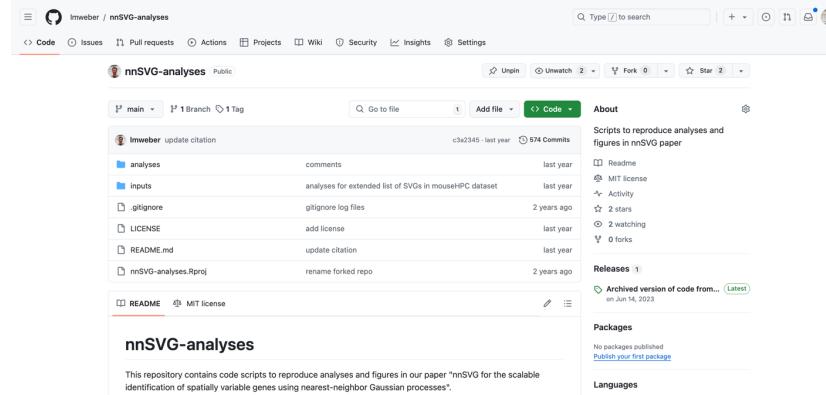
→ Example: <https://github.com/lmweber/nnSVG>

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

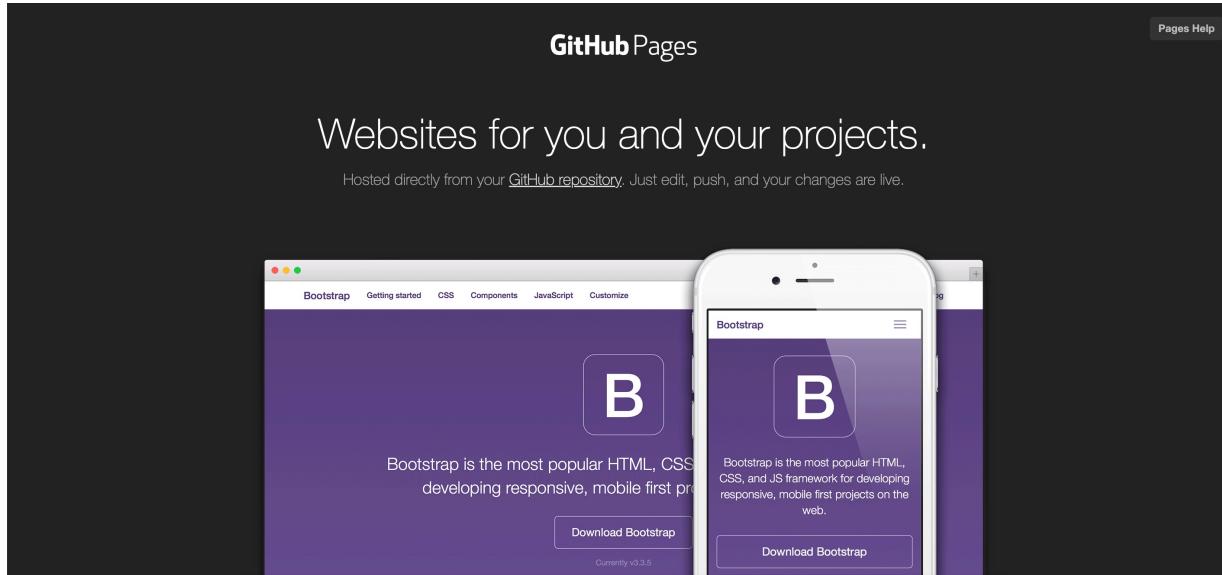
Working with Git and GitHub makes it easier to follow good coding practices

- 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

Lukas Weber's GitHub profile page:

- Pinned:**
 - OSTA** (Public) - "Orchestrating Spatial Transcriptomics Analysis with Bioconductor" book. (TeX, 88 stars, 37 forks)
 - locus-c** (Public) - Code to reproduce analyses in our paper on the landscape of gene expression in the human locus coeruleus (LC). (HTML, 4 stars)
 - nnSVG** (Public) - nnSVG: scalable method to identify spatially variable genes (SVGs) in spatially-resolved transcriptomics data. (R, 21 stars, 10 forks)
 - nnSVG-analyses** (Public) - Scripts to reproduce analyses and figures in nnSVG paper. (R, 3 stars)
 - diffcyt** (Public) - R package for differential discovery analyses in high-dimensional cytometry data. (R, 21 stars, 12 forks)
 - cytometry-clustering-comparison** (Public) - R scripts to reproduce analyses in our paper comparing clustering methods for high-dimensional cytometry data. (R, 45 stars, 14 forks)
- Contributions:** 885 contributions in the last year. Contribution settings: 2025 (selected), 2024, 2023, 2022, 2021.

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

The screenshot shows the GitHub interface for creating a new repository. At the top, there's a navigation bar with icons for search, issues, pull requests, and other repository management. Below the bar, the title 'New repository' is displayed next to a GitHub logo. On the left, a sidebar lists '1 General' and '2 Configuration'. The main area is titled 'Create a new repository' and contains instructions: 'Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository.](#)' and 'Required fields are marked with an asterisk (*).'. Under 'General', the 'Owner' dropdown is set to 'Imweber' and the 'Repository name' field is filled with 'git-demo-Sep2025'. A note below the field says 'git-demo-Sep2025 is available.' Under 'Description', there's a text input with the placeholder 'Demo repository for git / GitHub lesson' and a character count of '39 / 350 characters'. In the 'Configuration' section, the 'Choose visibility' dropdown is set to 'Public'. There are also notes about choosing who can see and commit to the repository.

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

The screenshot shows a GitHub repository page for 'git-demo-Sep2025'. The page includes a search bar, navigation links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the header, there's a profile picture, the repository name 'git-demo-Sep2025' (Public), and buttons for Pin, Watch (0), Fork (0), Star (0). A large blue banner at the top provides instructions: 'Start coding with Codespaces' (Add a README file and start coding in a secure, configurable, and dedicated development environment) with a 'Create a codespace' button; 'Add collaborators to this repository' (Search for people using their GitHub username or email address) with an 'Invite collaborators' button; and 'Quick setup — if you've done this kind of thing before' (Set up in Desktop, HTTPS, SSH, URL: https://github.com/lmweber/git-demo-Sep2025.git) with a note: 'Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.' Below this is a section for command-line cloning with the following code:

```
echo "# git-demo-Sep2025" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/lmweber/git-demo-Sep2025.git
git push -u origin main
```

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

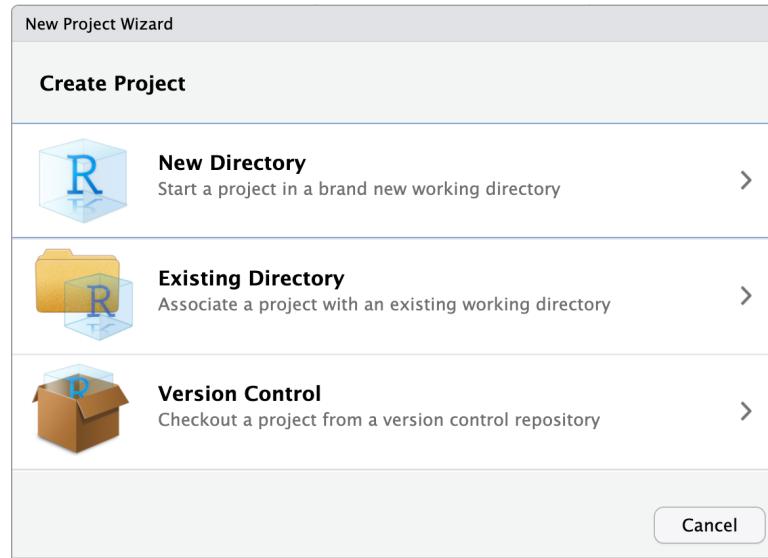
Create RStudio project

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

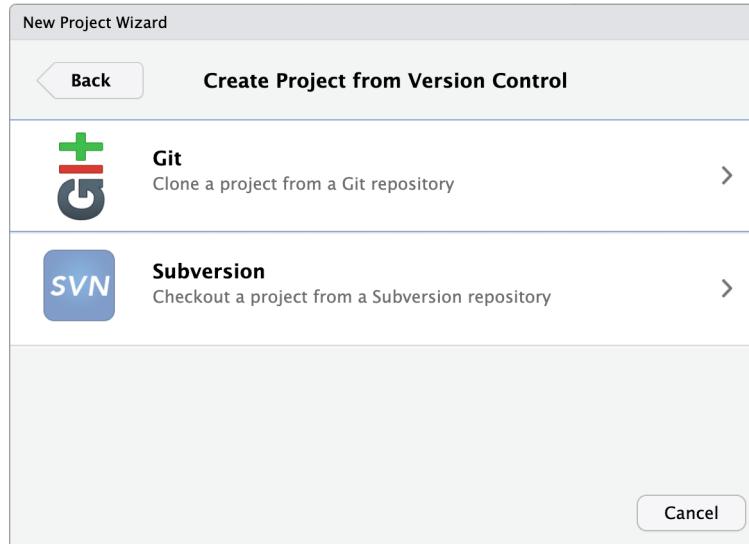
Now create an RStudio “Project” and link the repository from GitHub

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

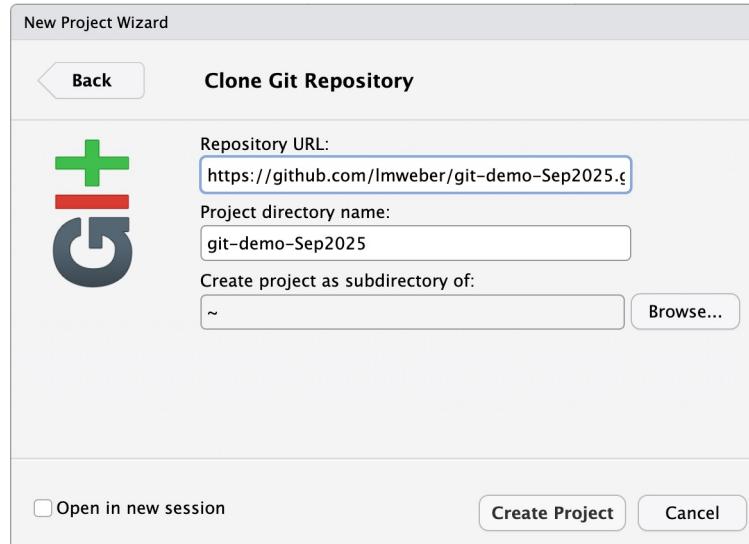
Now create an RStudio “Project” and link the repository from GitHub

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

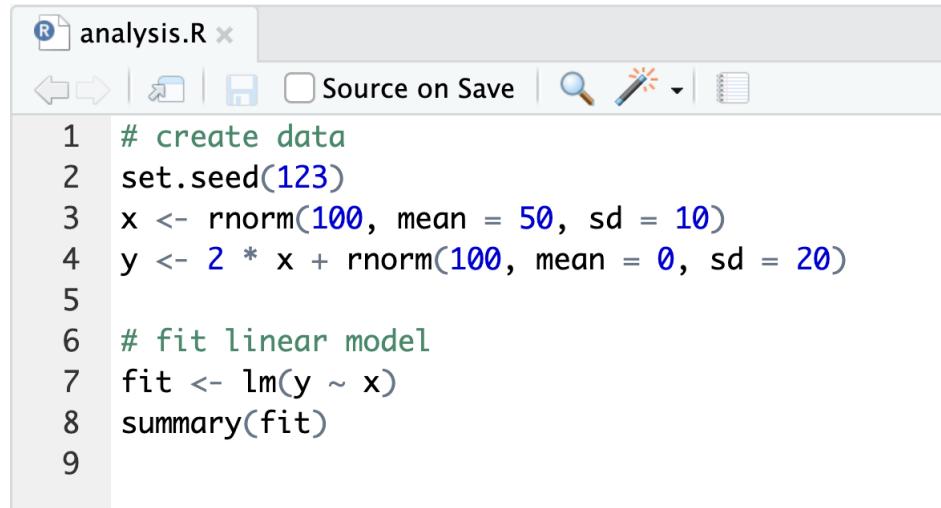


Add files to repository

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



```
analysis.R x
Source on Save | 🔎 ⚔ | 📝

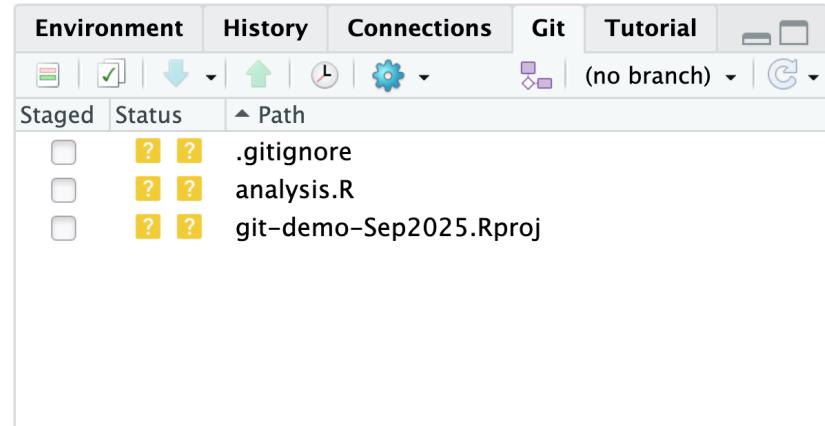
1 # create data
2 set.seed(123)
3 x <- rnorm(100, mean = 50, sd = 10)
4 y <- 2 * x + rnorm(100, mean = 0, sd = 20)
5
6 # fit linear model
7 fit <- lm(y ~ x)
8 summary(fit)
9
```

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”

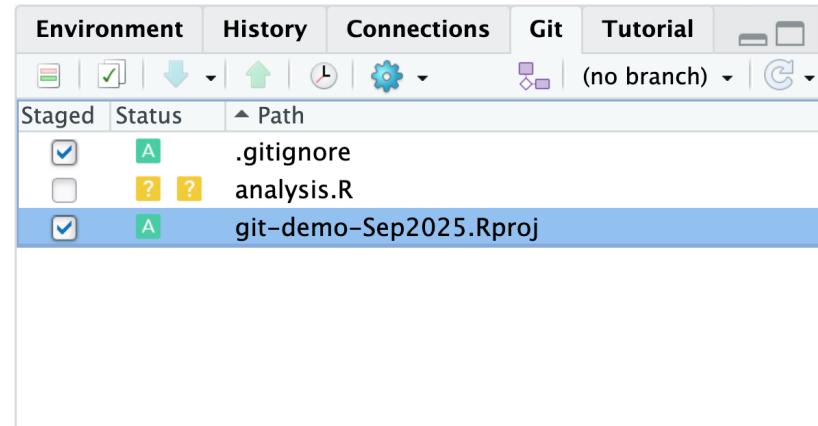
- “pull” commits from remote repository to local repository (i.e. updating local repository)

Add files to repository

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

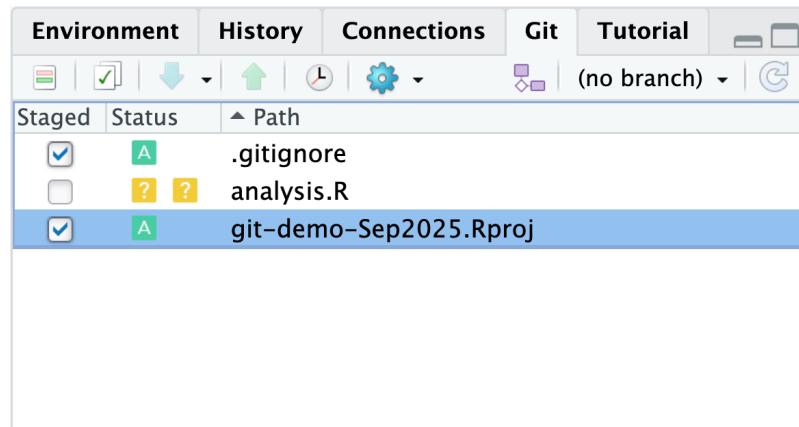
→ Demo

- 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



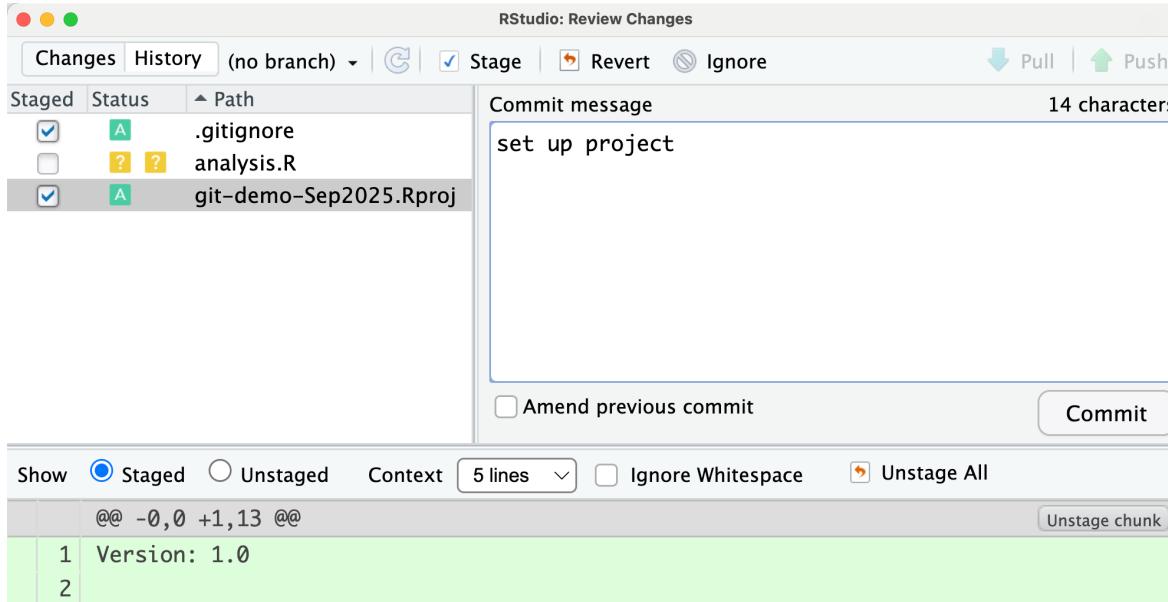
Add files to repository

- 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)



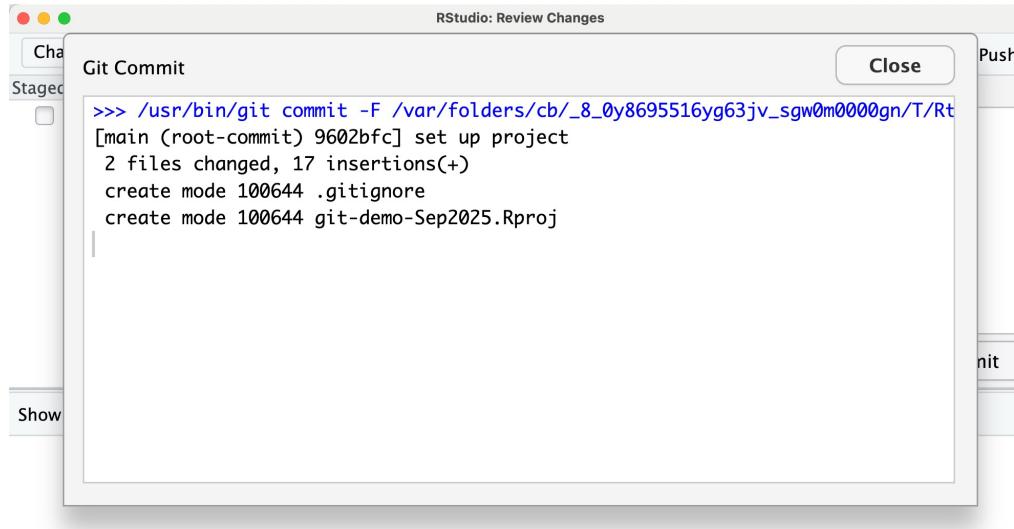
Add files to repository

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



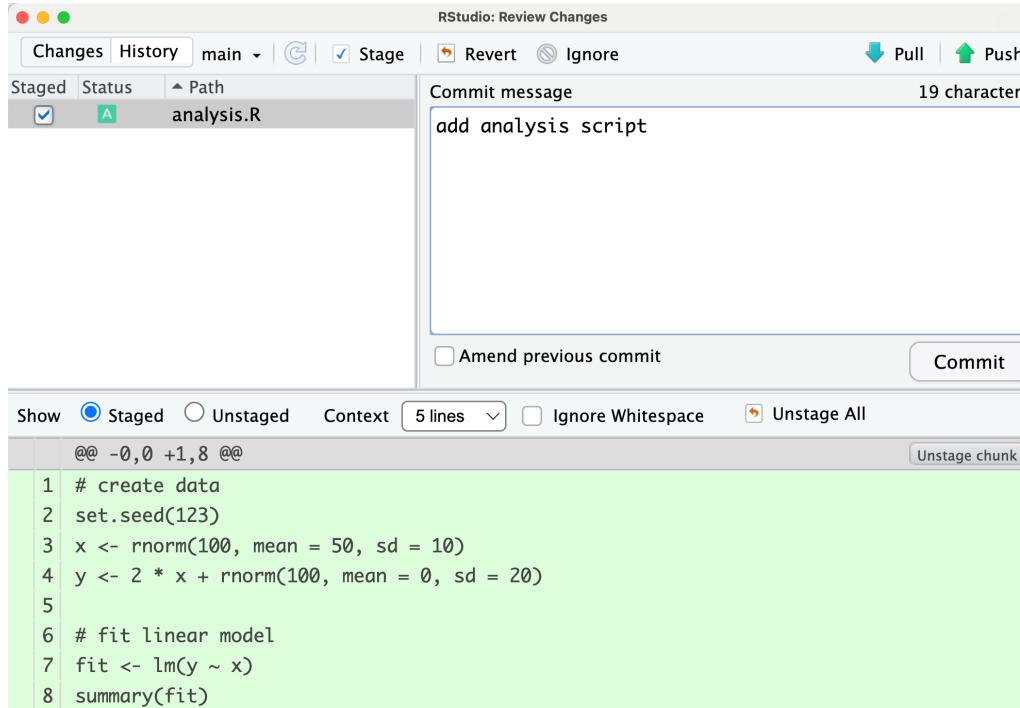
Add files to repository

- 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



The screenshot shows the RStudio interface for managing version control changes. The main window title is "RStudio: Review Changes". The top menu bar includes "Changes", "History", "main", "Stage", "Revert", "Ignore", "Pull", and "Push". The "Changes" tab is selected. In the center, there's a "Commit message" field containing "add analysis script" with a character count of "19 characters". Below it is a checkbox for "Amend previous commit" and a large "Commit" button. At the bottom, there are filtering options: "Show Staged" (selected), "Unstaged", "Context", "5 lines", "Ignore Whitespace", and "Unstage All". A preview area shows the first 8 lines of the "analysis.R" file:

```
@@ -0,0 +1,8 @@
1 # create data
2 set.seed(123)
3 x <- rnorm(100, mean = 50, sd = 10)
4 y <- 2 * x + rnorm(100, mean = 0, sd = 20)
5
6 # fit linear model
7 fit <- lm(y ~ x)
8 summary(fit)
```

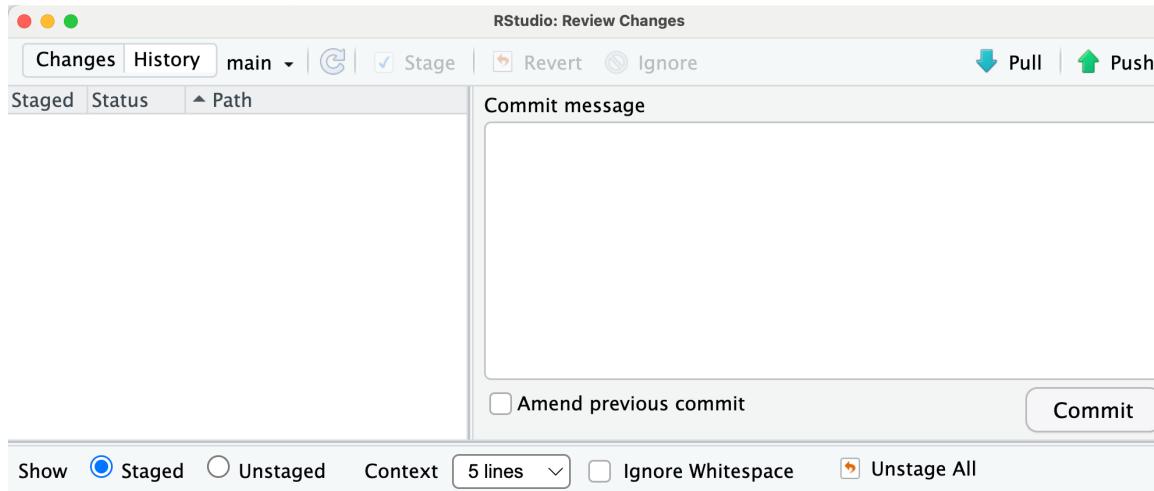
Add files to repository

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



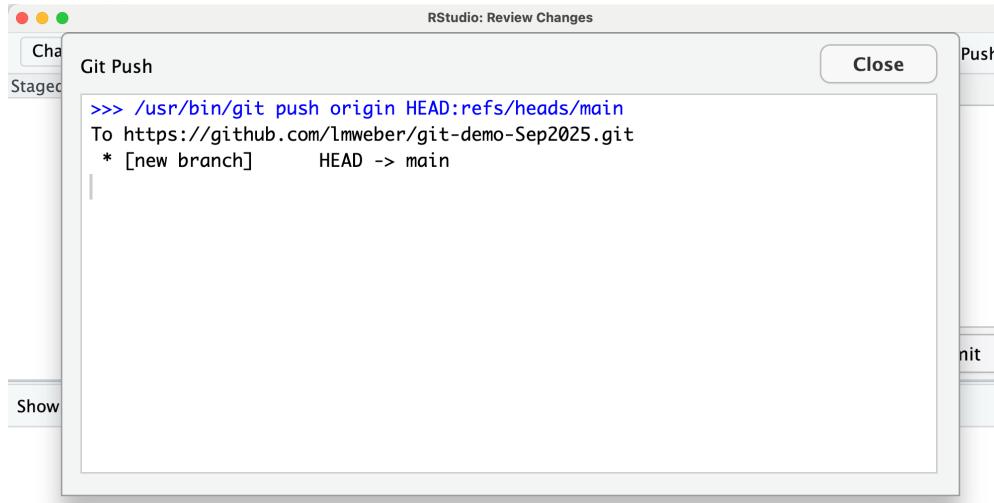
Add files to repository

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



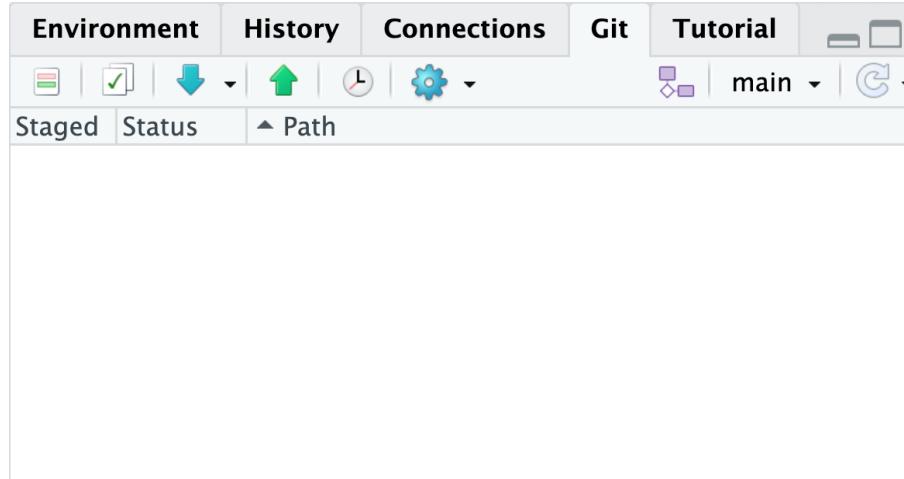
Add files to repository

- 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

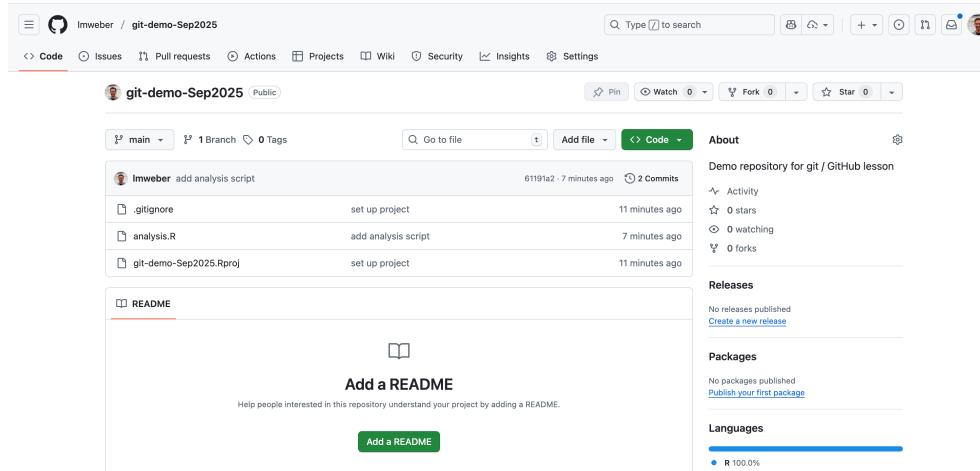
The screenshot shows a GitHub repository page for 'git-demo-Sep2025' owned by 'Imweber'. The 'Code' tab is selected. Recent commits include:

- Imweber add analysis script (61191a2 · 7 minutes ago)
- .gitignore set up project (11 minutes ago)
- analysis.R add analysis script (7 minutes ago)
- git-demo-Sep2025.Rproj set up project (11 minutes ago)

The repository has 1 branch and 0 tags. It has 2 commits, 0 stars, 0 watching, and 0 forks. There are no releases or packages published. The repository is 100% R.

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!