Version Control using Git in R

CDSI COMPUTATIONAL AND DATA SYSTEMS INITIATIVE

Dr. Tim Elrick | 16 April 2024

What you will learn today

- recap: how to connect to Git, GitHub and GitHub repositories
- how to use versioning in R with Git
- how to collaborate in R using Git

Prerequisites

- having installed Git
- enabled Git in RStudio
- have your GitHub token readily available or have it saved in R *(HTTPS access to GitHub)*

Recap: Basic Git commands using the terminal

`git status`	checks status of Git
`git init`	creates a Git database for current folder
`git add <file name>`	adds file to staging pool
`git add .`	adds all files in folder to staging pool
`git commit -m "My commit message"`	adds staged files to Git database

Let's recap how this works in RStudio…

Recap: Basic Git commands connecting to GitHub

Prerequisites: Empty GitHub repo *and* RProject exists

`git remote add origin https://github.com/un/rn.git`	connects Git to GitHub
`git push -u origin main`	initial push from local to online repo

Let's recap how this works in RStudio…

un = username, *rn* = repo name
main = local main branch, *origin* = GitHub main branch

CDSI COMPUTATIONAL AND DATA SYSTEMS INITIATIVE

Recap: Connect RStudio to GitHub using `usethis`

Prerequisites: RProject exists

```r
usethis::use_git() # enables Git for current RProject
# now make your first commit, then
usethis::use_github() # creates repo on GitHub with RProject name
```

Recap: Create RProject from GitHub repo

Prerequisite: GitHub repo exists

Create a *folder* in the **terminal**, then use the following code:

`git clone https://github.com/un/rn.git`

Alternatively, run the following code using the `usethis` package in **R**:

```r
usethis::create_from_github("https://github.com/un/rn.git",
                            "path/to/local/folder")
```

Versioning with Git – first steps

Every time, you create a ***new commit*** you create a *new version* of your project.

We can see all versions in our local Git either

… in **RStudio** by clicking on `History` 🕒

… *alternatively*, in the **terminal** using

`git log`	shows the history log of commits
`git log --oneline`	same as above, but in one-liners
`git log --date=human`	shows dates in the log in a nicer form

Versioning with GitHub – first steps

We can *push* a new commit to our online GitHub repo – or *pull* the latest version.

Prerequisite: RProject is connected to GitHub repo

`git pull`	downloads latest changes from GitHub to local RProject, if there are any
`git push`	uploads latest changes (commits) from local RProject to GitHub, if there are any

A common workflow

1. Open your **RProject** and check if your `Git` tab is clean *(i.e. no file shows up in the staging area/index)*
2. Start working on your *RScript* or *Quarto doc*.
3. When you reached a significant step in your coding,
   a. save your file, then
   b. **stage** and **commit** with message `wip` (*work in progress*)
   c. don’t **push** yet!

A common workflow

4. Continue working until you reached the next significant step,
   a. save your file, then
   b. **stage** and **amend commit** *(without changing the commit message)*
   Note, in the **terminal** this would be:
   ```
   git commit --amend --no-edit
   ```
   c. don’t **push** yet!

A common workflow

5. When you finally achieve a *milestone*:

   a. Save your file, then

   b. **stage** and **amend commit** *(change your message to what you achieved)*

   c. Now **push**!

Check differences between versions

1. To check differences in versions between **locally saved files** and **committed files**

- either click on `Diff` Diff in **RStudio**
- or in the **terminal** use `git diff`

2. To check differences in versions between **two different commits**

- in the **terminal** use `git diff <hash1> <hash2> [<file name>]` *(where the* hashes *are at least the first 4 characters of two different commit hashes)*

Going back in time

You can

1. **look at** an older version/commit
2. **return** to an older version/commit (but keep your file versions in the working directory)
3. **return** to an older version/commit (and dispose your file versions in the working directory)

Older versions – *look at*

- in **RStudio**, go to the `Git` tab and click on `History`
- in the **terminal** with
  - `git show <hash> <file name>` *(shows version of the file of commit hash)*
  - `git show HEAD~ <file name>` *(shows the previous version of the file)*
  - `git show HEAD~`*n* `<file name>` *(shows the n-th previous version of the file)*

Note, HEAD~ means the pointer in your commit tree is moved back to the *parent commit*.
HEAD~2 means the pointer is moved back to the *grandparent commit*.

Older versions – *return (but keep files)*

Returning to older versions/commits only works in the **terminal**.

- to undo your last commit, use: `git reset --soft HEAD~` It will put the committed files back into the *staging area/ index*. This is useful, if you forgot to *amend your commit* or if you want to *add more files* to your commit.
- to undo your last commit and *unstage* the committed file, use: `git reset HEAD~`
- *note, you can use `reset` specifying a filename or path, e.g. if you want to amend only one file out of a commit with multiple files.*

Older versions – *return (and dispose)*

- to undo your last commit in a way that you get back to where you left when you committed *(i.e. deleting changes you have been working on since)*, use:
  `git reset --hard HEAD~`
- to undo your last commit for a specific file and delete all the changes for that file since then, use:
  `git checkout HEAD~ filename`
- in **RStudio**
  - by clicking on `Diff` Diff,
  - selecting the file to reset, and
  - then clicking on `Discard all` Discard All

Working with branches – first steps

If you want to try something out, you can *create a branch* of your repository

::: {.incremental}

- in **RStudio** by clicking on
- in the **terminal** with `git branch <branch name>`

:::. . .

To *switch to a branch*

Note, when you create a branch your new branch will look like the last commit.

- in **RStudio** click on `main` next to ⧉ and select the *branch*
- in the **terminal** use `git switch <branch name>`

Working with branches

To delete a branch, use: `git branch -d <branch name>`.

To merge a branch into `main`,

- you first switch to *main* with `git switch main`,
- then `git merge <branch name>`.

Note, you can only delete branches that are merged.

Note, you can recover a deleted branch with `git checkout -b <branch name> <hash>` *(using the hash displayed when you deleted it)*

Use someone else’s GitHub repository

- in your **browser** go to https://github.com/AGeographer/mock_project
- copy the HTTPS URL and change back to **RStudio** and use
  - either `usethis::create_from_github()`
  - or `File > New Project > Version Control > Git`
  - or in the **terminal** type `git clone yourRepo.git`

Cooperating with others

If you want to see if the GitHub repo has different versions than your local repo,

- check with `git fetch`
- followed by `git diff head...origin`

If you want to update your local repo, use `git pull`.

Note, if two versions of the same file exists, Git will auto-merge them at first, but you need to resolve the conflict then.

Last resort

Prerequisite: you had pushed your Git repo to GitHub.

If your Git repository is screwed up (which might happen when you start working with Git),

- rename your local folder,
- then either `usethis::create_from_github()` in **R**
- or `git clone yourRepo.git` in the **terminal**

Resources

- Posit developer Jenny Bryan’s https://happygitwithr.com
- Alex Douglas, et al. 2024 An Introduction to R, chapter on version control: https://intro2r.com/github_r.html