Git basics - Review

Git basics - Overview

- Quick intro
- Explain will cover “idealized” local workflow
- But not perfect or correct - they will have to use their own judgment

- Focused recap
- Recap some core concepts
- Questions? Just shout them out
- Bit of a code-along
- Pair-programming practice

What even is Git?

- Many IDEs have “Git integration”
- Don’t need Github to practice anything

- Version Control System (VCS)
- Command line tool…
- Git != Github

Why use Git?

- Versioning = easy rollback
- Collaboration: single source of truth
- Cite reproducibility when relevant

- Version history and recoverability

- Takes “snapshots” of your project
  - I.e. `commits`
- Collaboration: multiple people, one codebase
- Reproducibility and audit trail

Key concepts

- Emphasize staging vs working tree vs HEAD
- Branch: independent line of development
- Remote: hosted copy (e.g., GitHub)
- Add diagrams?
  - .git folder
  - git cycle
- Index vs working tree vs HEAD

Questions - shout out the definitions of the following

Easy ones first...

- Repository (repo)
- Add
- Commit
- Push
- Pull

Trickier ones...

- Clone
- Fork
- Where is your repo?
- Branch
- Staging area
- Remote

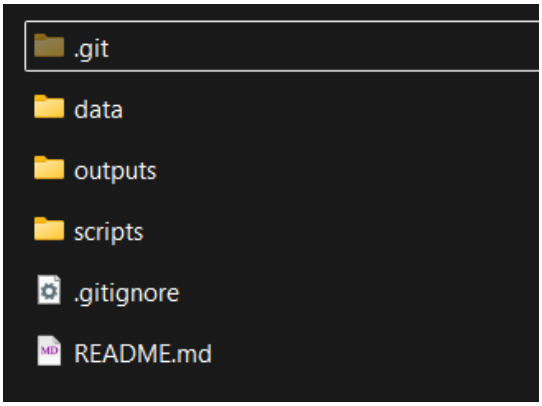

Figure 1: .git folder

"Commit cycle"

- Edit > stage > commit (repeat)
- Use git status and git log frequently
- Show one tiny example live

- Edit files
- Add to the index
- Commit with a message
- Check the status
- (Push when ready)
- Repeat

Edit > stage > commit (repeat)

Other important commands / actions

- Reminder to check git status before committing
- `init` -> create repo
  - This is done on Github.com
- `clone` -> copy remote to local

  - use RStudio New Project interface

- `status` -> show changes
  - See the Git pane
- `add` -> "stage" changes
  - (aka add them to the "index")
- `commit` -> save a snapshot of your projects current state
- `log` -> See the history window

Branching basics

- Glossing over the technical details
- Branch = movable pointer to a commit
  - Parallel but "independent"
- Name branches clearly; keep them focused
- Merge when ready; resolve conflicts if necessary

- Branch -> a parallel line of work / development
- Create using **RStudio Branch button**
- Switch branches with the **RStudio Branch dropdown**
- "Merge" branches using **GitHub.com**
- Keep branches small and focused

Branch example

- Course-notes website
- Developed by several people all working on different branches.
- Can work on branches where there is unfinished work without impacting the live website.
- Same goes for, e.g. a data analysis project
- https://www/ddi-talent.github.io/website-ug-data-science/
- https://www.github.com/ddi-talent/website-ug-data-science/

Undoing things

Can be fraught!

- Three different "levels" of undo
  - `revert`
  - `reset`
  - `restore`
- We will take a look at all three together

Quick demo

Code along…