Happy Git and Github for the useR
Session 04 - Git fundamentals

Boook club R-Ladies Bergen, R-Ladies Den Bosch, R-Ladies Amsterdam

Book by Jenny, presentation by Michelle

Program for today

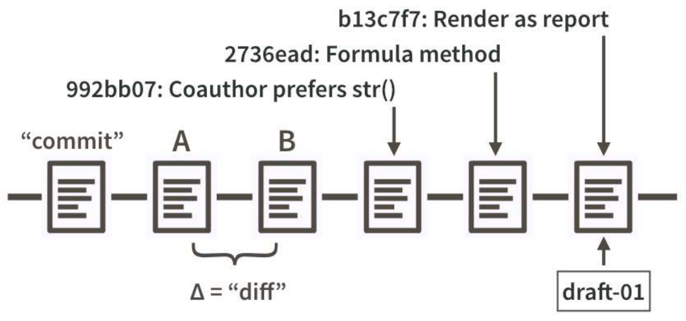
- Some Git basics
    - 20 Repo, commit, diff, tag
    - 21 Git commands
    - 22 Branches
    - 23 Remotes
    - 24 Refs

How Git works, concepts, applying it to data science

Chapter 20: Repo, commit, diff, tag

Terms

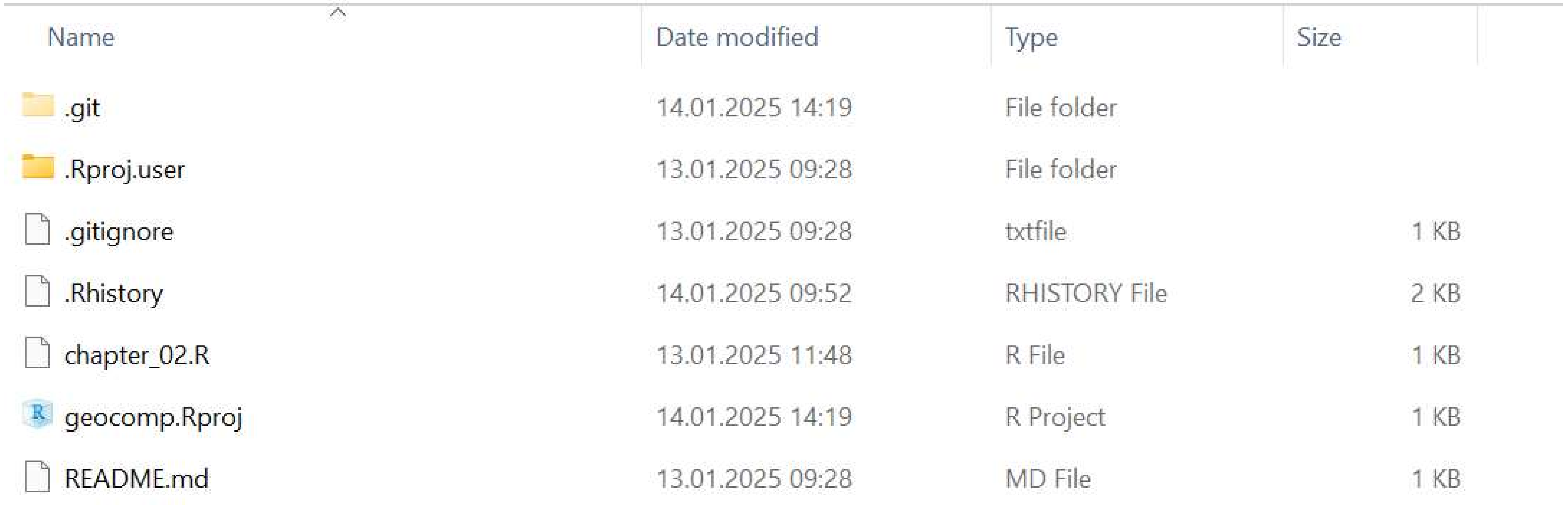
- **Repo** (repository): set of files
- **Commit**: snapshot of current version of files in a project/repo
- **Diff**: differences between one commit and another commit. *Each Git version of a file is an accumulation of diffs*
- **SHA** (Secure Hash Algorithm): a string of 40 letters and numbers assigned by Git to a commit to uniquely identify it
- **Tag**: a name you can assign to a version, e.g. "v.1.0.3" or "draft-01"



Advice

For each project:

- Assign it to one local directory
- Make it an RStudio project



Name	Date modified	Type	Size
.git	14.01.2025 14:19	File folder	
.Rproj.user	13.01.2025 09:28	File folder	
.gitignore	13.01.2025 09:28	txtfile	1 KB
.Rhistory	14.01.2025 09:52	RHISTORY File	2 KB
chapter_02.R	13.01.2025 11:48	R File	1 KB
geocomp.Rproj	14.01.2025 14:19	R Project	1 KB
README.md	13.01.2025 09:28	MD File	1 KB

- Make it a Git repository (see previous book section)

Workflow

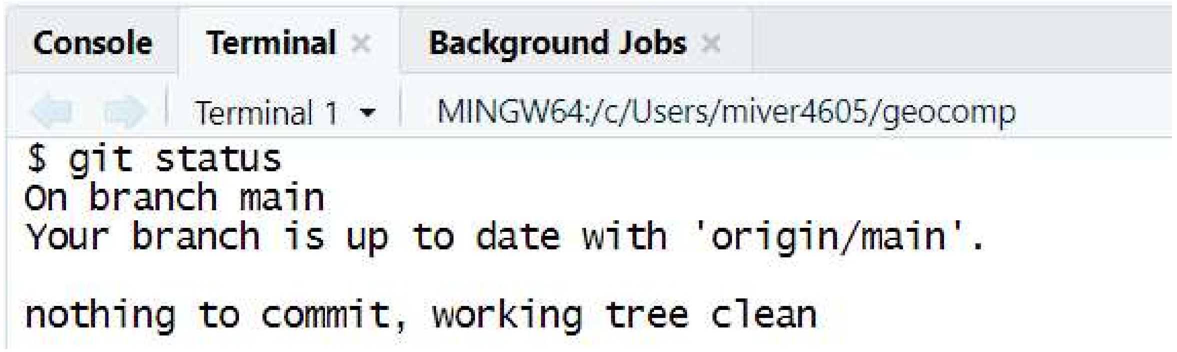
- Work on your files locally
- Periodically make a commit
  - When a "significant" stage is reached
  - Include a short commit message motivating the change
- Periodically push commits to GitHub - makes current version of repo accessible to others
  - First pull so that you have the updated remote version

Workflow

Chapter 21: Git commands

Can you remember/guess what these commands do?

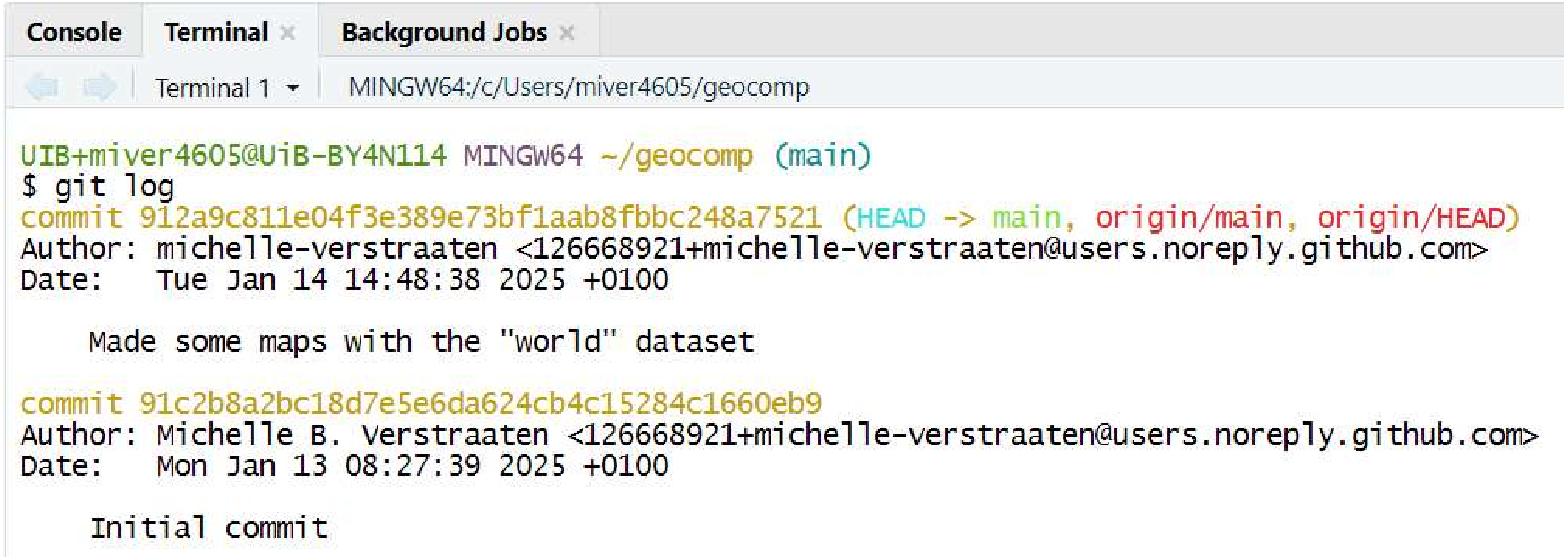
- `git clone https://github.com/jennybc/happy-git-with-r.git`
- `git remote --verbose`
- `git add foo.txt` : add foo.txt to the index (staging area)
- `git commit --message "A commit message"`
- `git status`



```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

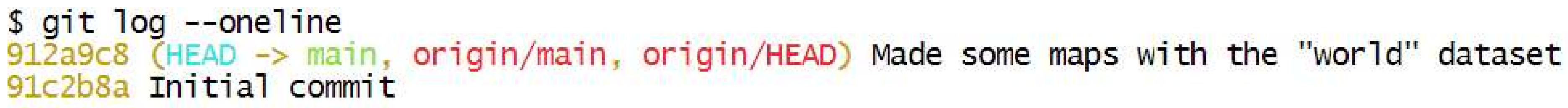
Can you remember/guess what these commands do?

- `git log`



```
UIB+miver4605@UiB-BY4N114 MINGW64 ~/geocomp (main)
$ git log
commit 912a9c811e04f3e389e73bf1aab8fbbc248a7521 (HEAD -> main, origin/main, origin/HEAD)
Author: michelle-verstraaten <126668921+michelle-verstraaten@users.noreply.github.com>
Date:   Tue Jan 14 14:48:38 2025 +0100

    Made some maps with the "world" dataset

commit 91c2b8a2bc18d7e5e6da624cb4c15284c1660eb9
Author: Michelle B. Verstraaten <126668921+michelle-verstraaten@users.noreply.github.com>
Date:   Mon Jan 13 08:27:39 2025 +0100

    Initial commit
```

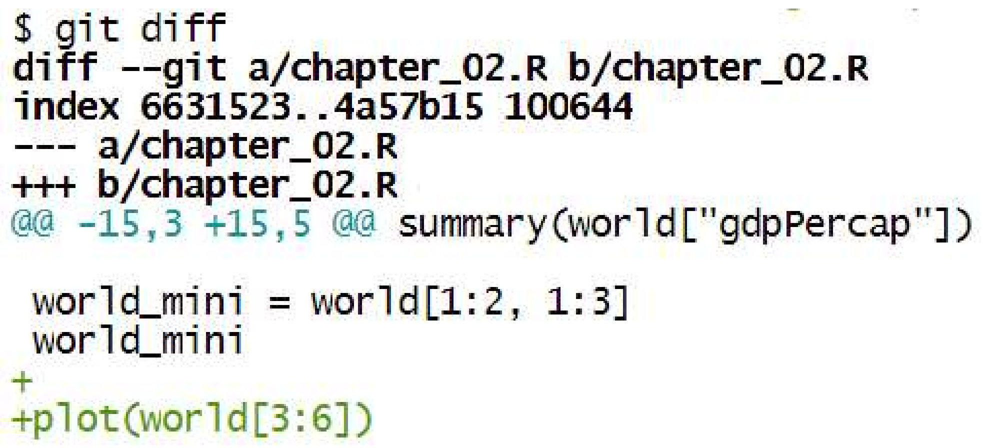
- `git log --oneline`



```
$ git log --oneline
912a9c8 (HEAD -> main, origin/main, origin/HEAD) Made some maps with the "world" dataset
91c2b8a Initial commit
```

Can you remember/guess what these commands do?

- `git diff`



```
$ git diff
diff --git a/chapter_02.R b/chapter_02.R
index 6631523..4a57b15 100644
--- a/chapter_02.R
+++ b/chapter_02.R
@@ -15,3 +15,5 @@ summary(world["gdpPercap"])

 world_mini = world[1:2, 1:3]
 world_mini
+
+plot(world[3:6])
```

- The rest of the list is covered in the next few chapters

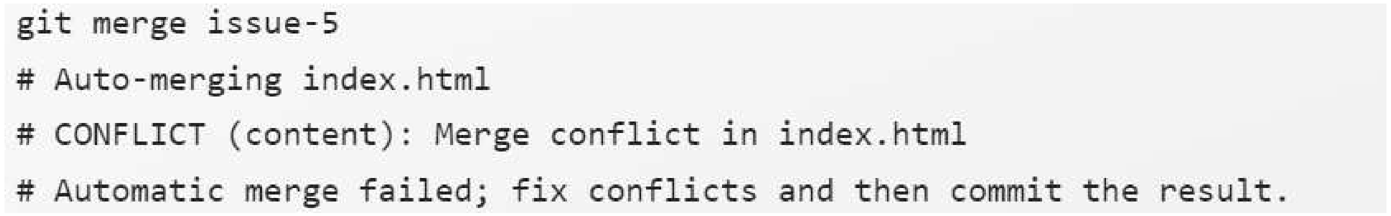
Chapter 22: Branches

Branching and merging

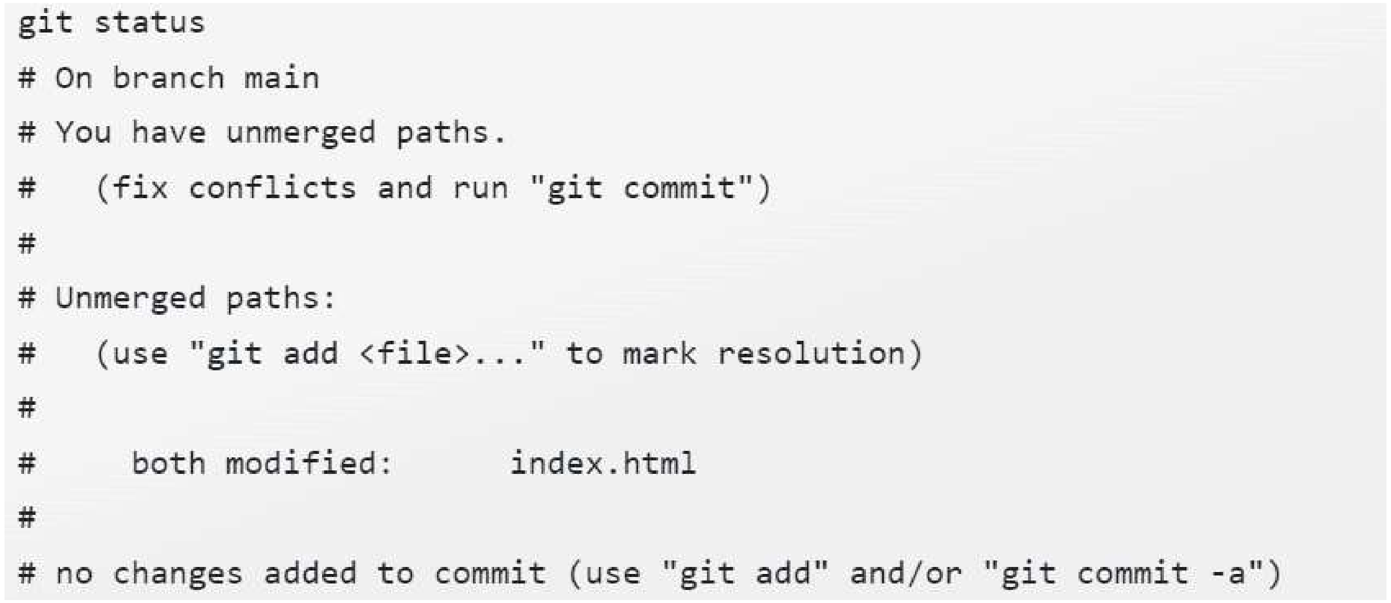
- For parallel work or experimenting with new features without interfering with the main project
- `git branch issue-5`
- `git checkout issue-5`
- `git checkout -b issue-5`
- Switching branch when you have incomplete work:
  - `git commit --all -m "WIP"`
  - `git checkout main`
  - `git checkout issue-5`
  - `git reset HEAD^`

Merging and handling conflicts

- `git checkout main`
- `git merge issue-5`

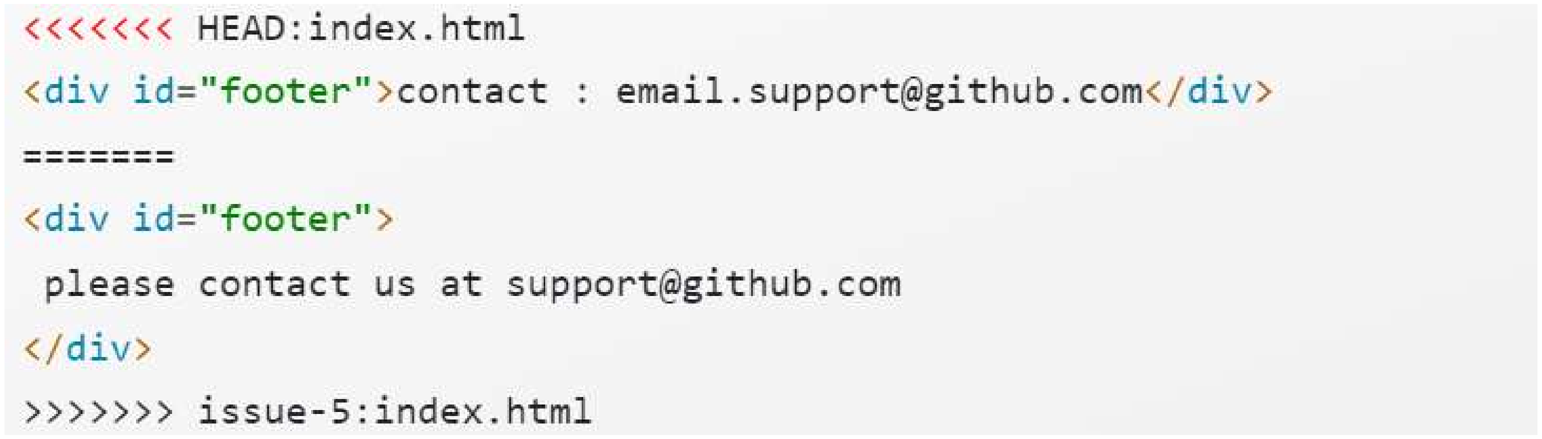


```
git merge issue-5
# Auto-merging index.html
# CONFLICT (content): Merge conflict in index.html
# Automatic merge failed; fix conflicts and then commit the result.
```

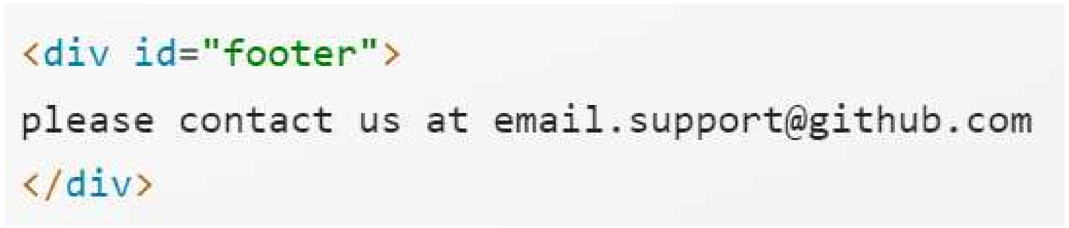


```
git status
# On branch main
# You have unmerged paths.
#   (fix conflicts and run "git commit")
#
# Unmerged paths:
#   (use "git add <file>..." to mark resolution)
#
#     both modified:      index.html
#
# no changes added to commit (use "git add" and/or "git commit -a")
```

Merging and handling conflicts



```
<<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=======
<div id="footer">
 please contact us at support@github.com
</div>
>>>>>>> issue-5:index.html
```



```
<div id="footer">
please contact us at email.support@github.com
</div>
```

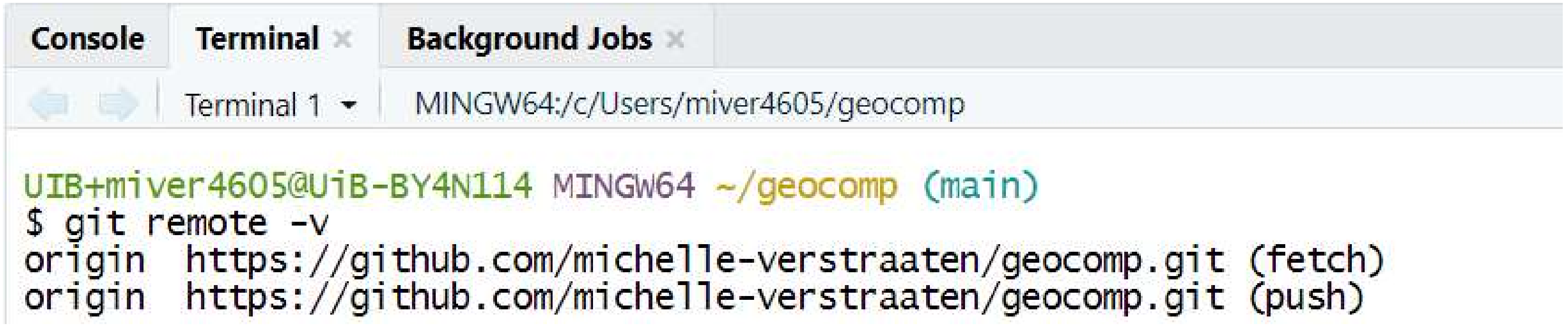
- `git add index.html`
- `git commit`

- If something goes wrong: `git merge --abort`
- More info: https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

Chapter 23: Remotes

Remotes

- Remote repositories are hosted on a network (not your local version)



```
UIB+miver4605@UiB-BY4N114 MINGW64 ~/geocomp (main)
$ git remote -v
origin  https://github.com/michelle-verstraaten/geocomp.git (fetch)
origin  https://github.com/michelle-verstraaten/geocomp.git (push)
```

- `git clone`
- `git remote add happygit https://github.com/jennybc/happy-git-with-r.git`
- Adding a second remote is useful when you have forked and cloned a repo and want to pull changes from the original repository (not your forked remote) - this second remote is usually nicknamed `upstream`:
- `git remote add upstream https://github.com/TRUE_OWNER/REPO.git`

Fetching and pushing

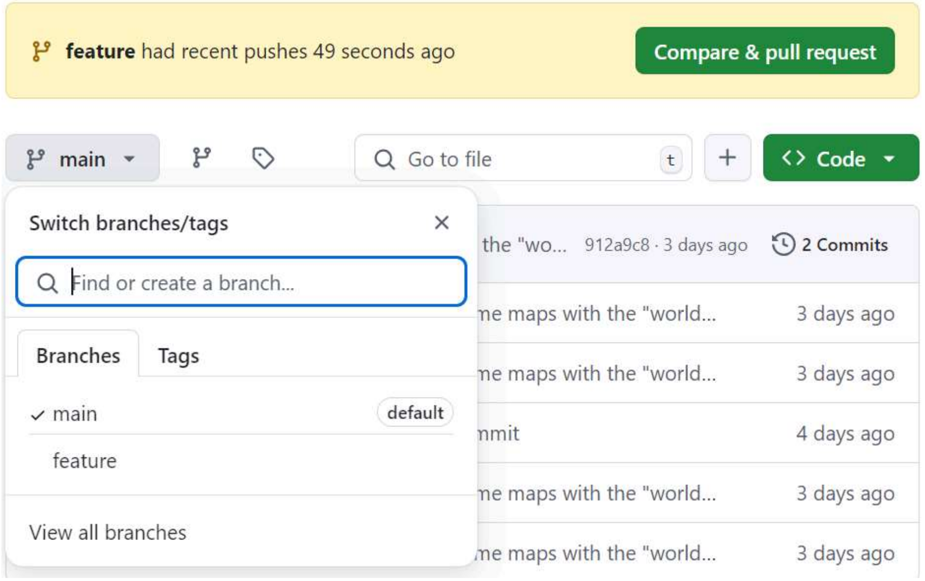
- `git fetch happygit`: downloads the remote commits to your local repo without changing the local branch
- `git fetch` + `git merge` ≈ `git pull`
- Git pull vs git fetch: https://www.youtube.com/watch?v=T13gDBXarj0

-
  ```
  # push my local changes to the origin remote's main branch
  git push origin main
  ```

Upstream tracking branches

```
UIB+miver4605@UiB-BY4N114 MINGW64 ~/geocomp (feature)
$ git push
fatal: The current branch feature has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin feature
```

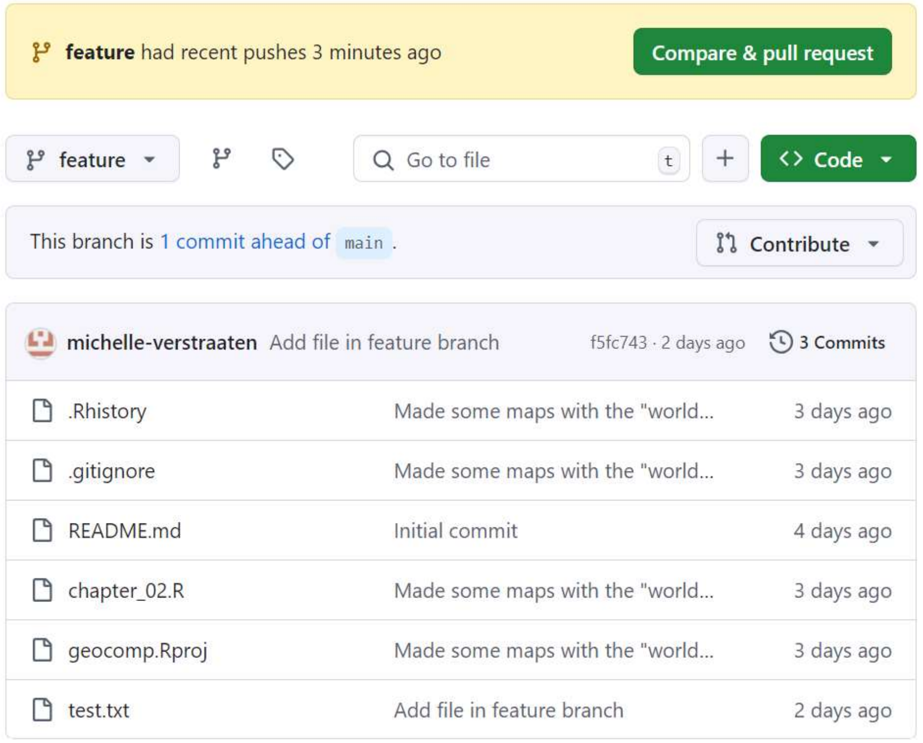
- `git config --global push.default current`


feature had recent pushes 49 seconds ago — Compare & pull request

main | Go to file | Code

Switch branches/tags
Find or create a branch...
Branches | Tags
✓ main — default
feature
View all branches

the "wo... 912a9c8 · 3 days ago — 2 Commits
me maps with the "world... — 3 days ago
me maps with the "world... — 3 days ago
mmit — 4 days ago
me maps with the "world... — 3 days ago
me maps with the "world... — 3 days ago


feature had recent pushes 3 minutes ago — Compare & pull request

feature | Go to file | Code

This branch is 1 commit ahead of `main`. — Contribute

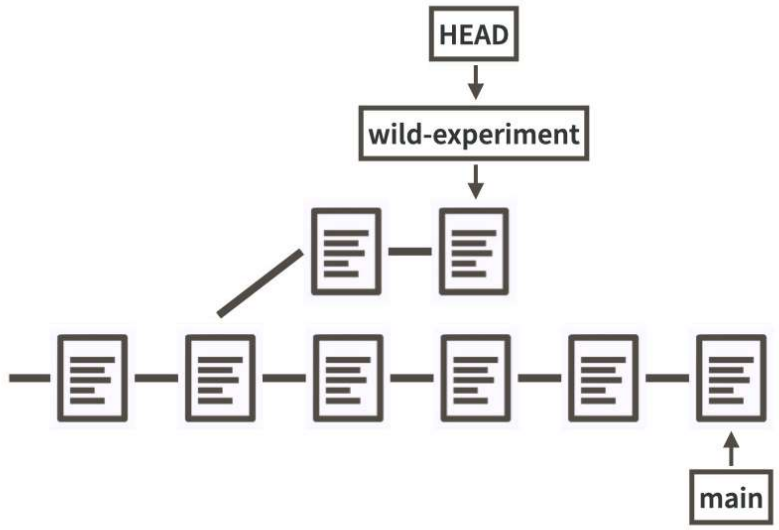
michelle-verstraaten Add file in feature branch — f5fc743 · 2 days ago — 3 Commits

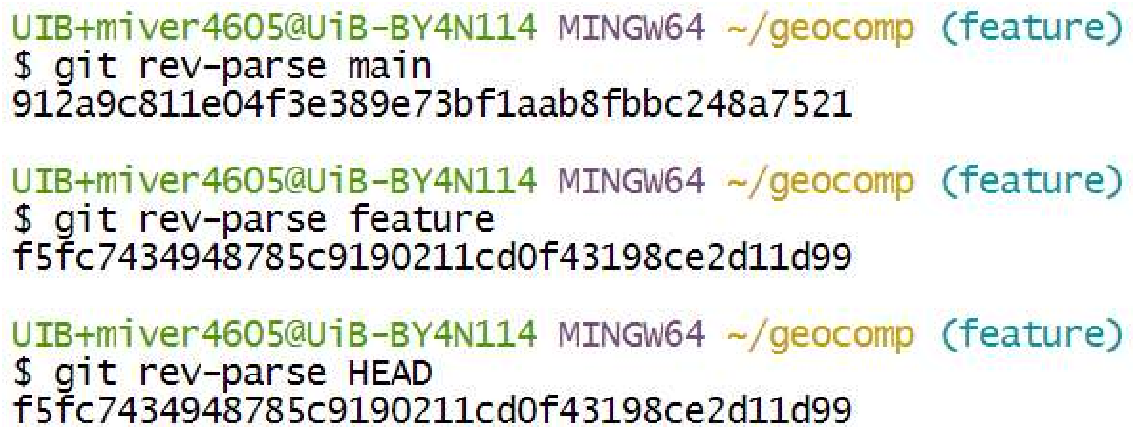
.Rhistory	Made some maps with the "world...	3 days ago
.gitignore	Made some maps with the "world...	3 days ago
README.md	Initial commit	4 days ago
chapter_02.R	Made some maps with the "world...	3 days ago
geocomp.Rproj	Made some maps with the "world...	3 days ago
test.txt	Add file in feature branch	2 days ago

Chapter 24: Refs

What are refs?

- References to specific commits (like pointers in programming). Examples:
  - a branch name
  - HEAD (a *symbolic ref*)
  - a tag (e.g., v1.4.2)





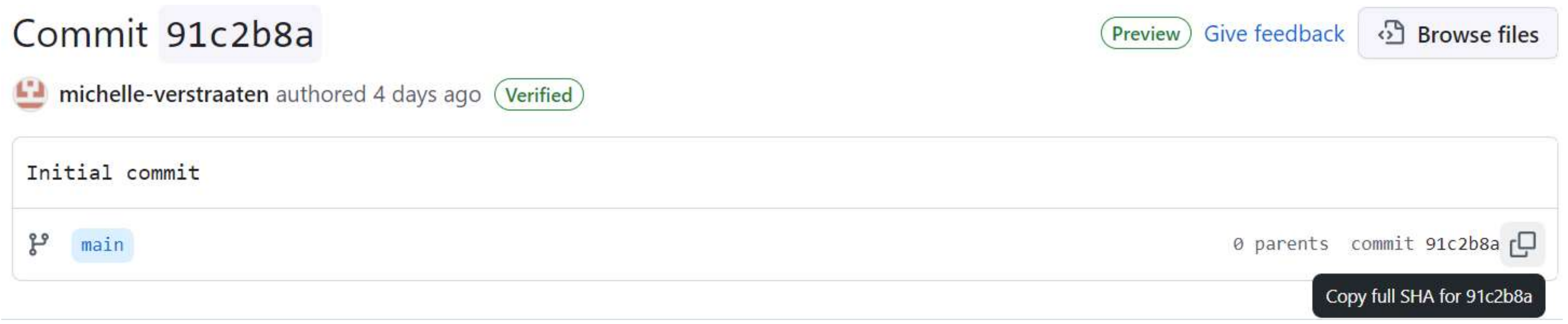
```
UIB+miver4605@UiB-BY4N114 MINGW64 ~/geocomp (feature)
$ git rev-parse main
912a9c811e04f3e389e73bf1aab8fbbc248a7521

UIB+miver4605@UiB-BY4N114 MINGW64 ~/geocomp (feature)
$ git rev-parse feature
f5fc7434948785c9190211cd0f43198ce2d11d99

UIB+miver4605@UiB-BY4N114 MINGW64 ~/geocomp (feature)
$ git rev-parse HEAD
f5fc7434948785c9190211cd0f43198ce2d11d99
```

- Use refs in commands like `git diff`, `git reset` and `git checkout`

Relative refs

- `HEAD~1` or `HEAD^`: the commit just before `HEAD`
- `HEAD~3` or `HEAD^^^`: three commits before `HEAD`
- See more: https://git-scm.com/docs/gitrevisions

Copying a specific SHA is easy in visual Git tools like GitHub and GitKraken



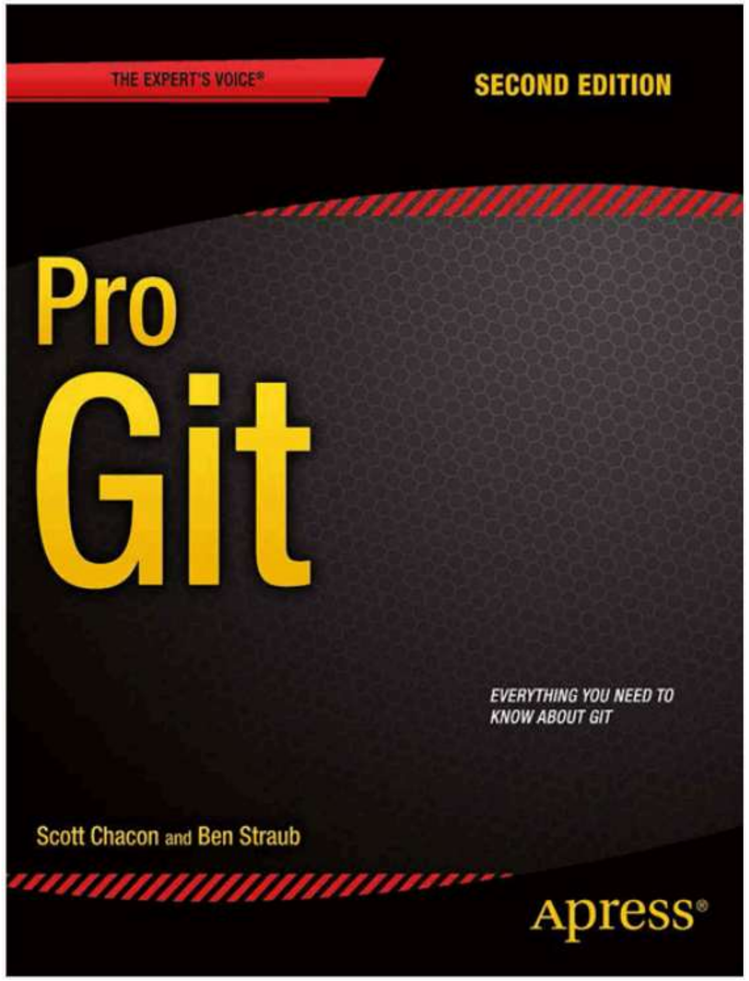
Commit 91c2b8a

michelle-verstraaten authored 4 days ago Verified

```
Initial commit
```

main 0 parents commit 91c2b8a

Pro Git


THE EXPERT'S VOICE®
SECOND EDITION
Pro
Git
EVERYTHING YOU NEED TO KNOW ABOUT GIT
Scott Chacon and Ben Straub
Apress®

https://git-scm.com/book/en/v2

The end of the session 4!

- Meetup for the Chapters
  - R-Ladies Amsterdam
  - R-Ladies Bergen
  - R-Ladies Den Bosch
- We need YOU as a presenter!