Digital Research Toolkit for Linguists

Week 14: Text editors, Git(Hub) basics

Anna Pryslopska

July 8, 2024

Psycholinguistics and Cognitive Modeling Lab

Exam

Exam date

Monday, July 29th 14:00–15:00, Kepler 17 (K2), M 17.52 (KP1705M 17.52)
Same time, different room! Should be in Campus.

Earlier exam date in justifiable cases

Wednesday, July 24th 11:00–12:00, Kepler 17 (K2), M 17.81 (KP178OGAM 17.81)
Different time, same room. Should be in Campus.

Term paper

You will receive new data on a different subject. Task: clean, analyze, summarize, do literature research, write a 15 page report in LaTeX, upload data to Git (Quarto report, tex files, plots etc.)

Homework

Homework

Find the changes I made in the big project

```
diff -r big_project_1 big_project_2
```

`main.tex`	backmatter → blackmatter
`title.tex`	Prysłopska → Pryslopska
`mypreamble.sty`	german → french
`chapter1.tex`	1 space → 2 spaces
`chapter2.tex`	341,755 → 341.755
`chapter2.tex`	three → 3
`chapter3.tex`	ten → 10

Any others were by mistake but good on you for catching them!

Homework

How many times does the word “Tagblatt” appear in the files **`corpus1.txt`**, **`corpus2.txt`**, and **`corpus3.txt`**? Count lines.

```
grep -wc "Tagblatt" corpus1.TXT corpus2.TXT
corpus3.TXT
```

```
corpus1.TXT 41
corpus2.TXT 34
corpus3.TXT 34
```

Homework

Count all the lines and instances of “die/Die/DIE” in the corpora.

```
grep -iw die corpus1.TXT | wc
```

file	lines	instances	bytes
corpus1.TXT	325	16651	124855
corpus2.TXT	466	26786	197282
corpus3.TXT	466	26773	197176

```
grep -iwo die corpus1.TXT | wc
```

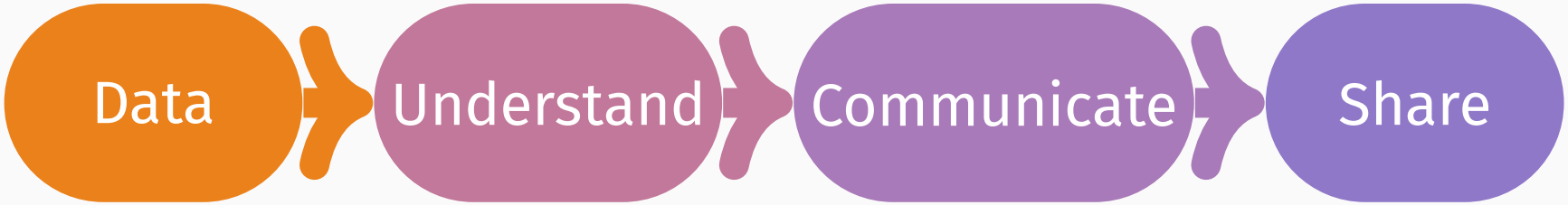
file	lines	instances	bytes
corpus1.TXT	325	685	2740
corpus2.TXT	466	1025	4100
corpus3.TXT	466	1026	4104

Homework

What are the differences between `corpus2.txt` and `corpus3.txt`

`corpus2.txt`	`corpus3.txt`
Eisbärbaby	Eisbär-Baby
—	Dabei nutzen in Ägypten…
Die ganzen “Skandälchen”…	—
Bagnčres-de-Luchon	Bagnères-de-Luchon
Künstler-Bohčme	Künstler-Bohème
30.11.2013	30.12.2013

Questions?



R & RStudio, packages, data types, formats, encoding

import from workspace, assign values, operations, clean, filter, arrange, select, merge, group, summarize, export, visualize

document, research, create clean and beautiful reports

connect, collaborate, backup

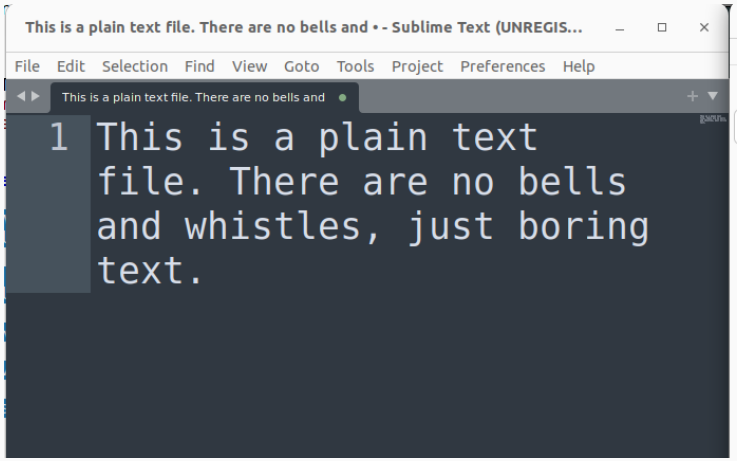
Table of contents

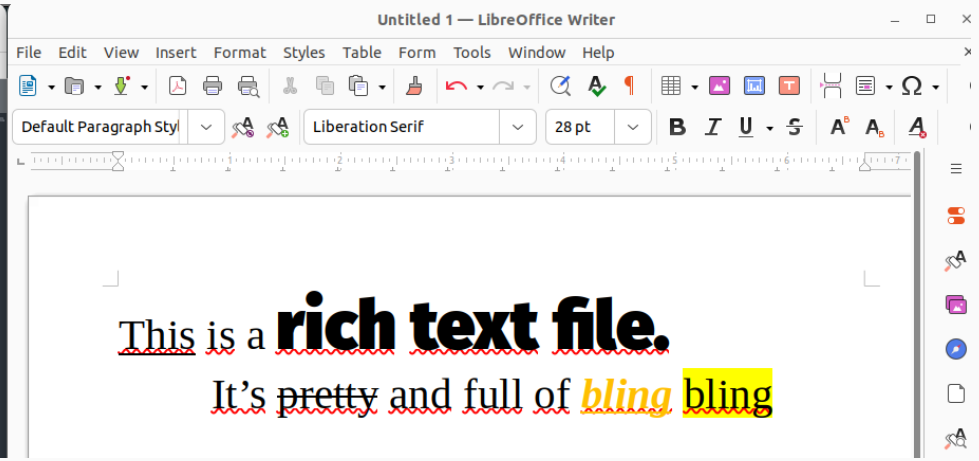
Text editors

What is a text editor?

Text editors allow you to write, open, edit, and create plain text files.

Plain vs. rich text: No formatting (typeface, size, weight, style etc.), nothing unexpected. Only text.





What is a text editor?

- Most portable way of sharing files.
- Useful for previewing data or writing code.
- Highlight any syntax (e.g. R, markdown, TEX)
- You can open any file in a text editor.
- Usually, nothing unexpected will happen.

What good is a plain text file?

Simplicity

No learning curve, minimalist interface, no distractions

Performance

Lightweight, fast even with large files, easy on the computer

Portability

Text files are readable across operating systems, compatible with everything, virtually the same since 1950s

Customization

Plugins and scripts can change an editor to a personal assistant

What good is a plain text file?

No Lock-In

No proprietary formats, independent, can be edited with any editor

Version Control Friendly

Easy to track changes and merge

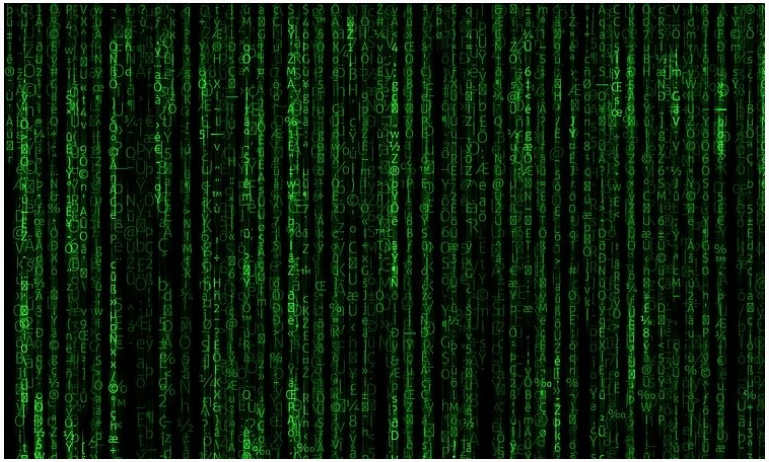
Security

Lower risk of viruses and embedded malicious code, unlikely to be targets

Focus on Content

Focus on writing and coding without formatting distractions, encourages good practices and clear documentation

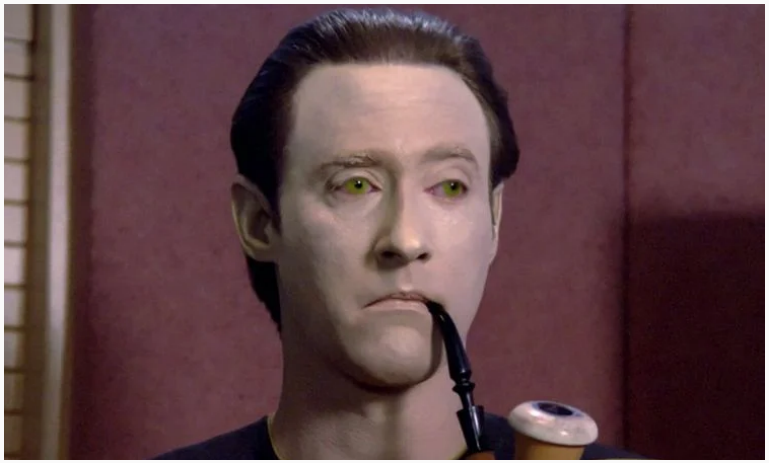
What use is a plain text file?



Writing and editing code.

System administration.

Creating quick notes, drafts, documentation.



Writing, manipulating, processing text data.

Which editor to chose?

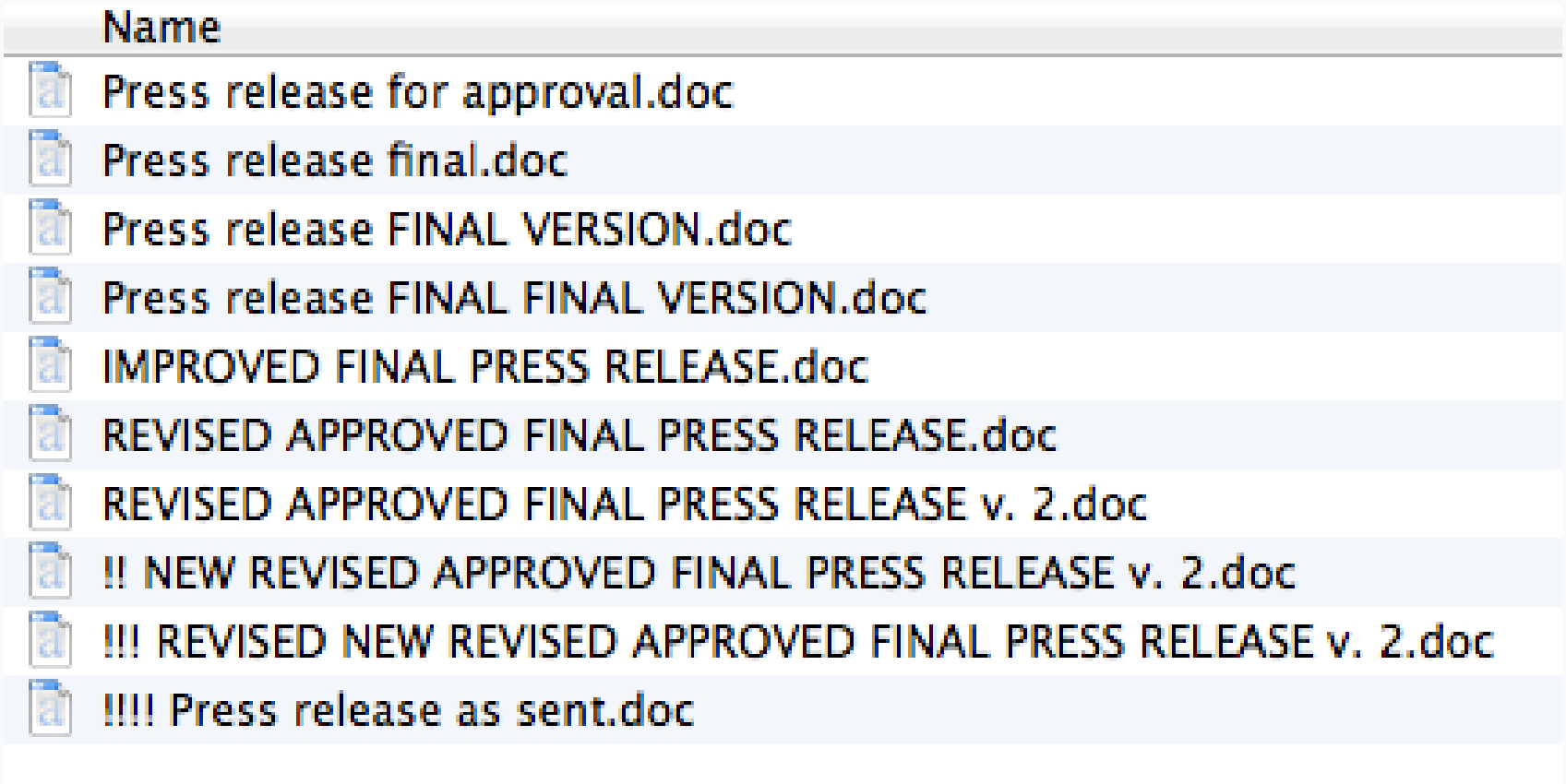
Opinions on text editors are like butts: everyone has one.

Which do I have to install?

1. **Sublime text** `https://www.sublimetext.com/`
   beautiful, simple, free but will nag you to buy it
2. **Microsoft VS Code** `https://code.visualstudio.com/`
   good for coding, git integration, overkill
3. **Brackets** `https://brackets.io/`
   integrates with photoshop, visualizations, but it's Adobe...
4. **Notepad++** `https://notepad-plus-plus.org/`
   step up from plain Notepad, good for beginners
5. Honorable mentions for more overkill:
   - **Emacs** `https://www.gnu.org/software/emacs/`
   - **Vim** `https://www.vim.org/`

Version control

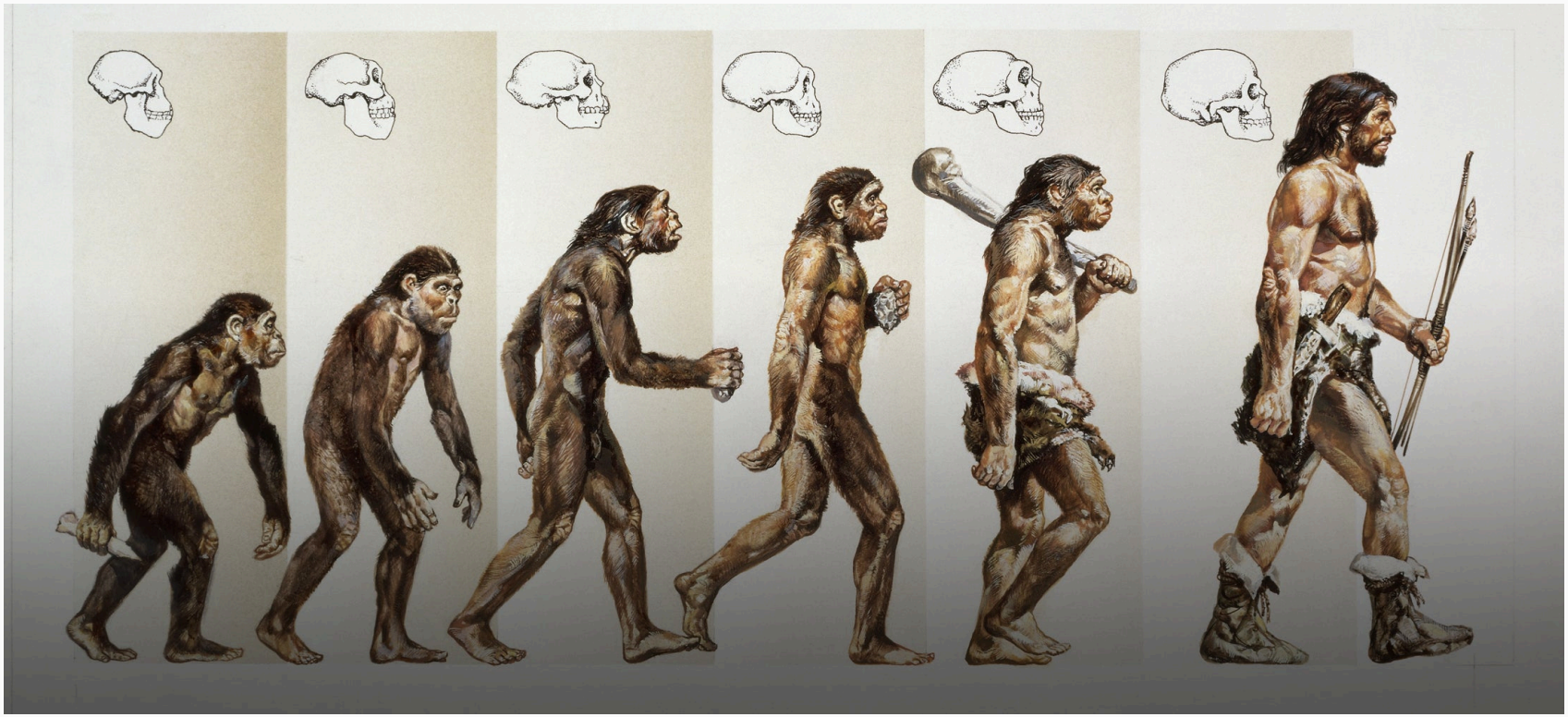
Does this look familiar?



Popular choices: date, initials, version nr, final, copy, reviewed

Renaming, saving as, adding dates and initials etc. to file names may work for small personal project but breaks down fast with collaboration and big projects.

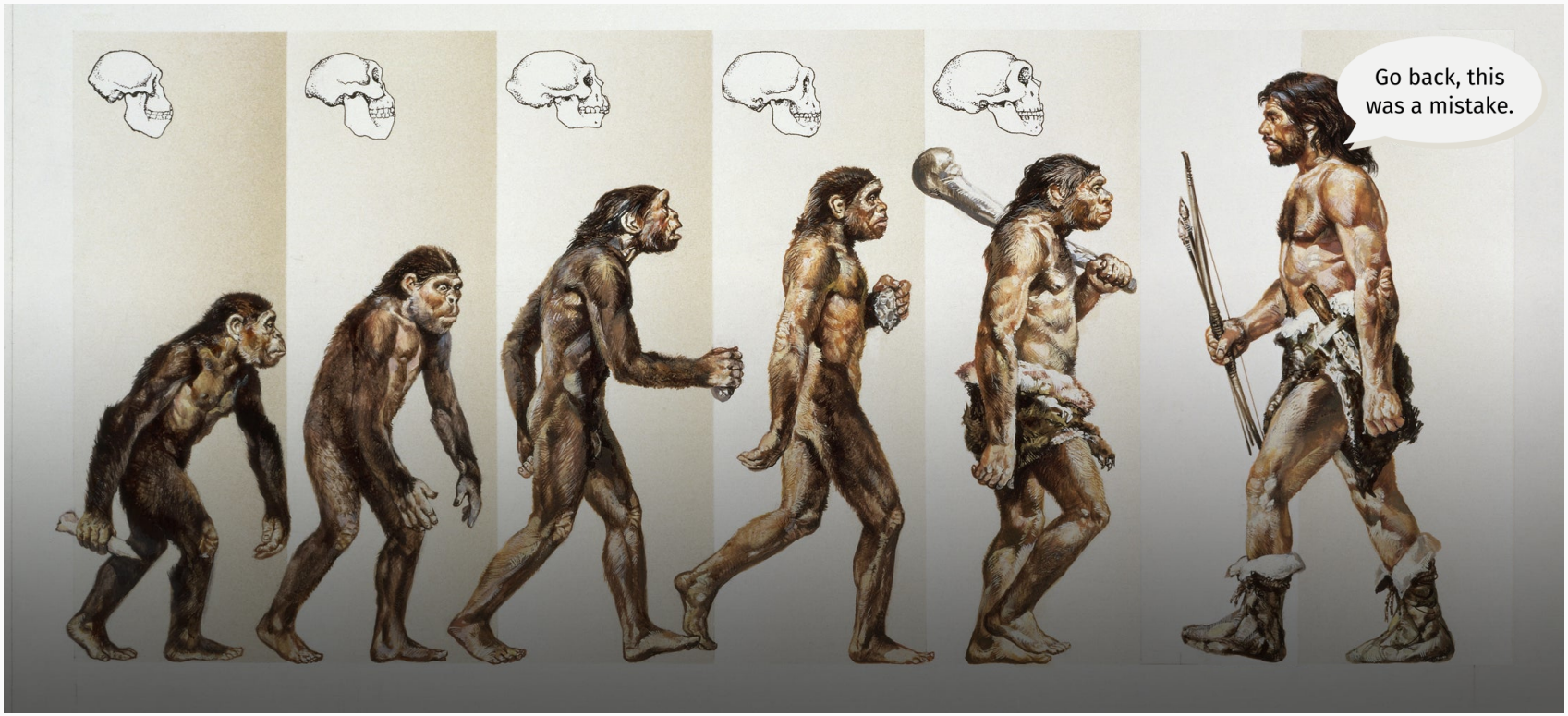
Version control



Gettyimages

Version control helps you track changes over time.

Version control



Gettyimages

When you mess up you can **go back** to when things worked.

Why use version control?

History and tracking

Keeps a detailed record of all stages of the project. Allows tracking who made which change and why.

Documentation

Documents changes, reasons for changes, and related discussions.

Backup and restore

A backup for the project. In case of mistakes, data loss, or sabotage, the previous version(s) can be easil restored.

Why use version control?

Experimentation

You can try out new ideas without affecting the core project. Branches can be safely created, removed, or integrated as needed.

Collaboration

Multiple people can work on the same project simultaneously and merge their changes without many conflicts. Different groups can work on different parts of a project.

Remote work

Many teammates can work from different locations and on different machines with different OS and synchronize their work.

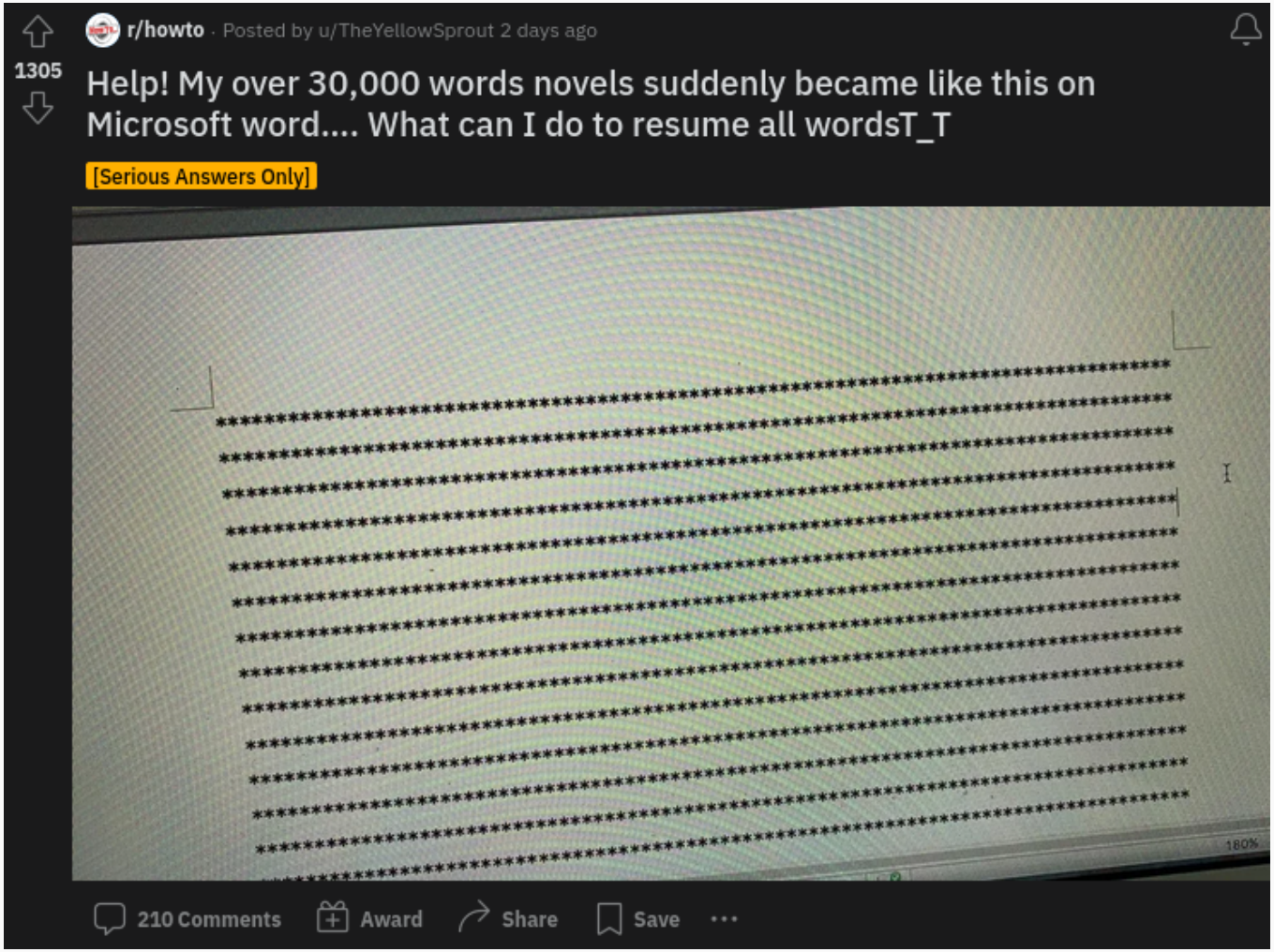
Tracking changes

`diff` is great locally and you can use DiffChecker → paid + online + data security?

Multiple people working on the same project in parallel makes files messy, fast:

- Collaborator 1 is writing the intro
- Collaborator 2 is adding the statistics
- Collaborator 3 is summarizing the results
- Proofreader is checking for errors
- ...

Backup

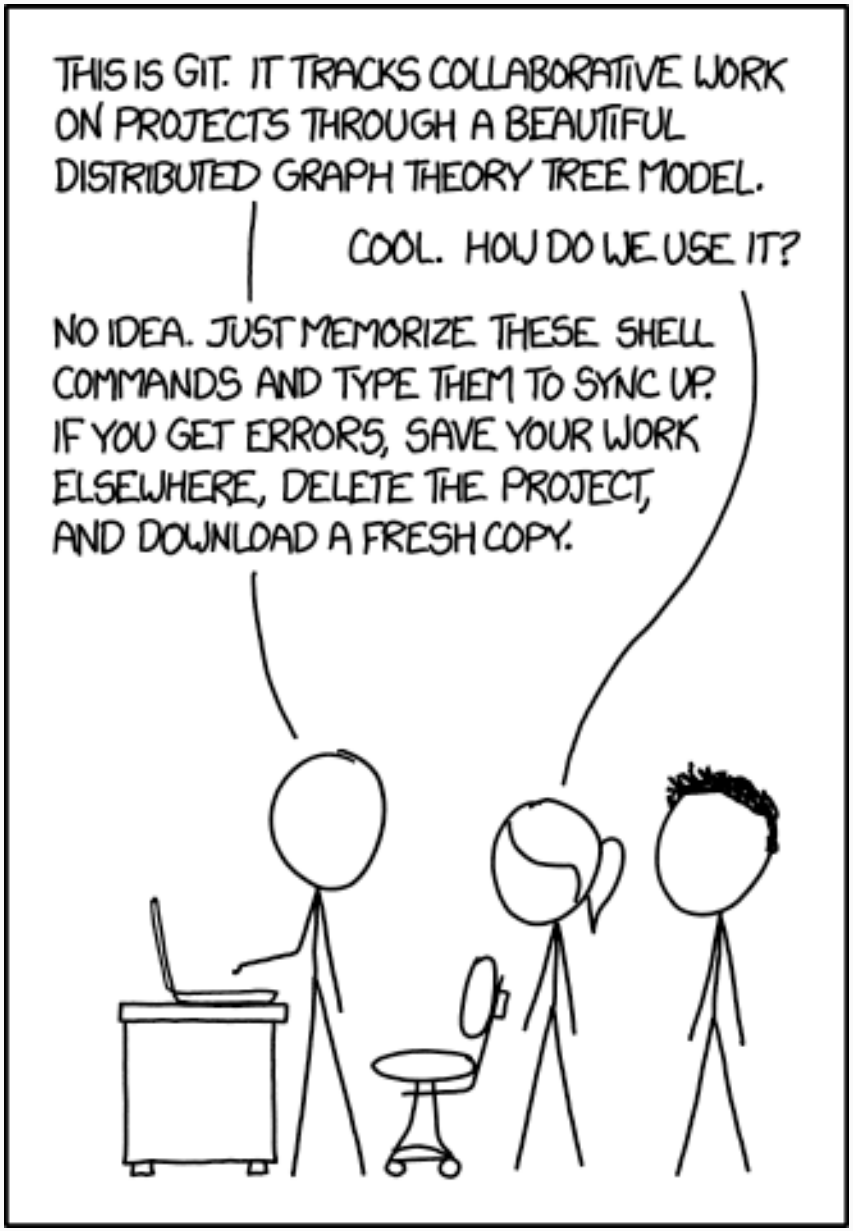


Remember to backup important files locally and in the cloud!

Git and GitHub

Why learn Git?

- Keep track of your data
- Synchronize across devices
- Collaborate effectively
- Resolve errors and fix mistakes
- Parallelize processes
- Backup and store files
- Remember, revert, and restore (short- and long-term)
- Be more employable as (data) scientists and developers
- It’s free and open source


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Git, GitHub, and GitLab

Git

(🇬🇧 slang: worthless, foolish, unpleasant person) an OS version control software that works locally on your computer. Great for individual use.

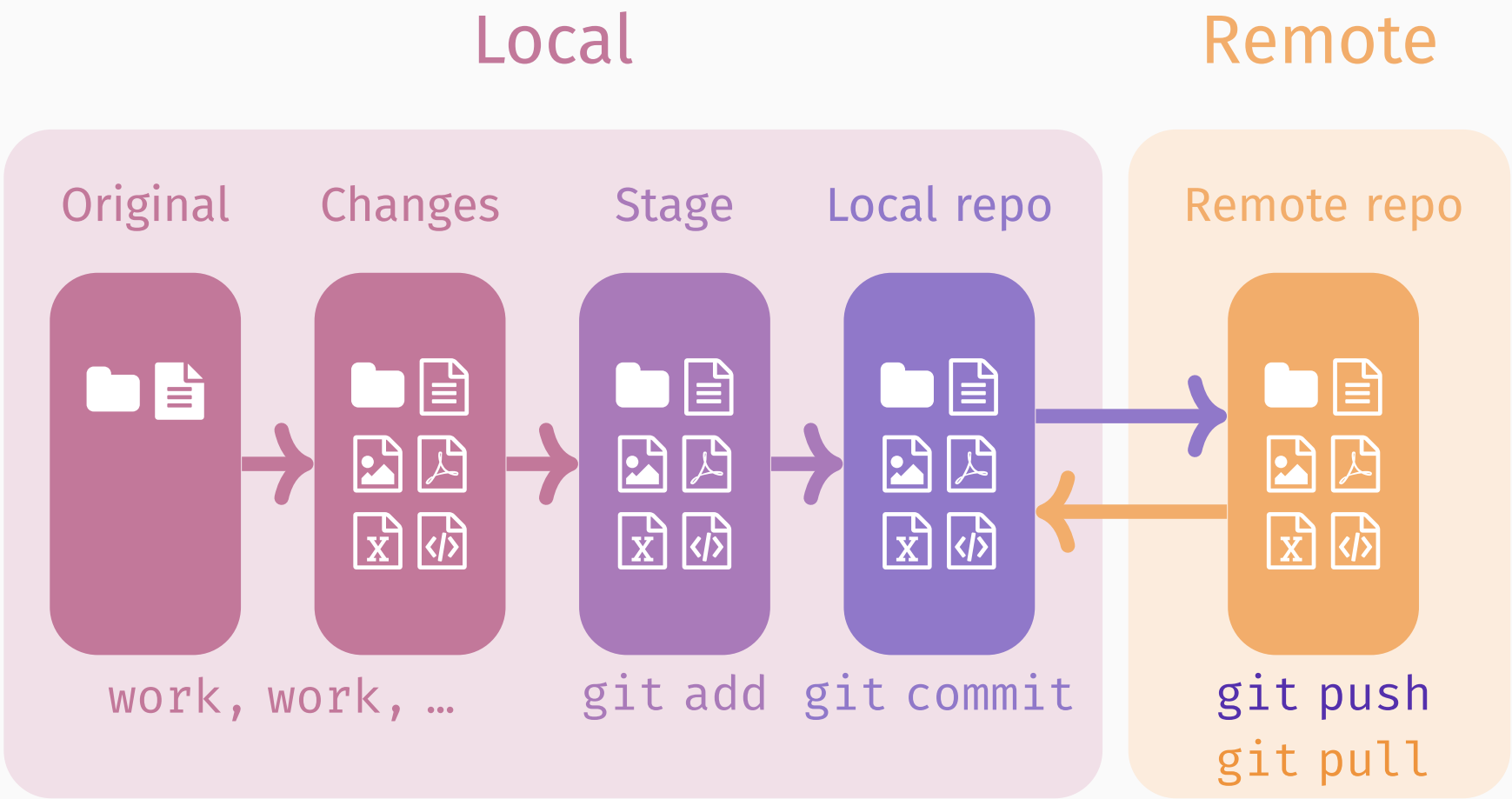
GitHub

the most popular code hosting platform that makes collaborating using git easier. LinkedIn and Twitter for developers, owned by Microsoft. Uni Stuttgart has their own GitHub server `https://github.tik.uni-stuttgart.de/`

GitLab

an alternative code hosting platform that has somewhat different user permissions and includes Continuous Integration (no need to pull → this will be explained soon).

Workflow


Local
Remote
Original
Changes
Stage
Local repo
Remote repo
work, work, …
git add
git commit
git push
git pull

Glossary: Setup

Repository/repo a place where something is stored and managed.

Server the computer storing the repository.

Client the computer connecting to the repository.

Working set/copy local directory/place where you make changes.

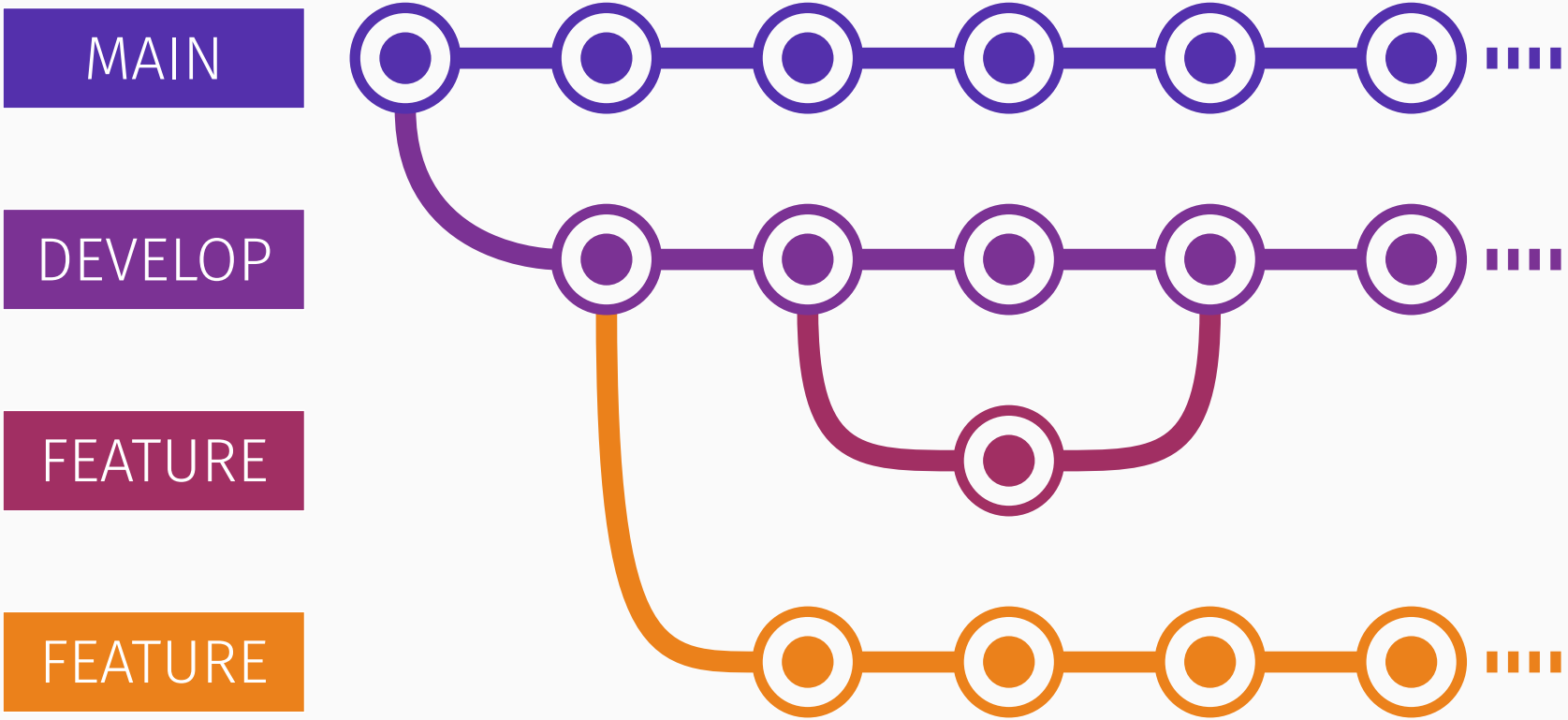
Trunk/main/master the main location for the data in the repository.

Fork is a copy of a repository that you can change to your liking.

Branch a parallel version of the repository at this time.

Merge combine a (feature) branch with the main branch/trunk.

Tree structure



Glossary: Actions (must know)

`add` tell git to track untracked files.

`commit` change to tracked files. Has a unique ID (≈ save as) and keeps record of changes. Usually contains a message, i.e. description of changes.

`status` what are the changes since the last commit?

`push` send changes to a remote repository.

`pull` get changes from a remote repository.

Glossary: Actions

`changelog` list of all changes.

`branch` (make a) parallel version of a repository.

`diff` difference in changes between files.

`merge` combining changes from two branches.

`reset` undo/rollback the last commit (move backwards).

`revert` undo/cancel a change but make a new commit (move forwards).

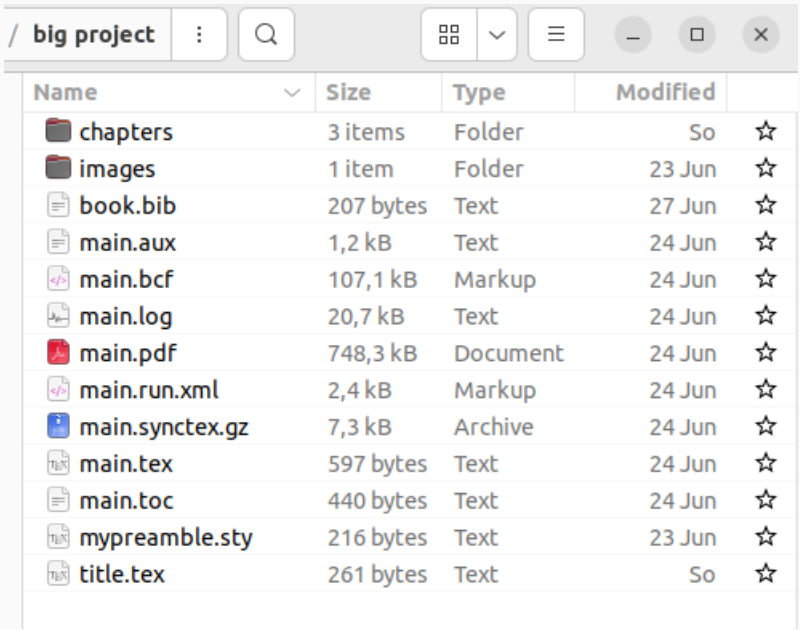
Getting started

Install git and set it up

1. Download git e.g.
   `https://github.com/git-guides/install-git`
2. Open a terminal
3. Give yourself a name:
   ```
   git config --global user.name "Your Name"
   ```
   - which program
   - (topic) do what and where
   - (variable) details please
   - your name or alias
4. Insert your email address:
   ```
   git config --global user.email "mail@example.com"
   ```

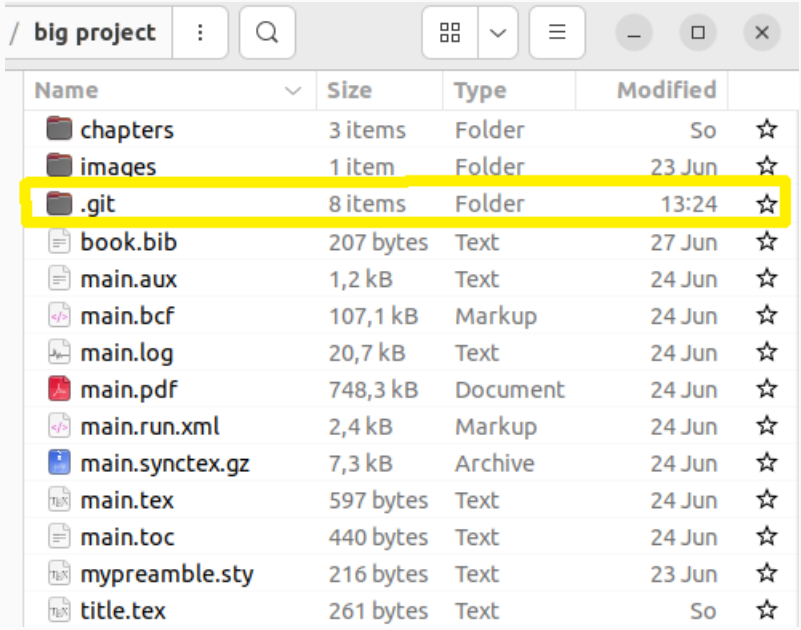
First repo

Navigate via command line (`cd`) or open in terminal.



Initialize git:

`git init`



What's going on

`git status`

```
anna@AP-UniSTR-Laptop:~/Desktop/ERT4H/big project$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        book.bib
        chapters/
        images/
        main.aux
        main.bcf
        main.log
        main.pdf
        main.run.xml
        main.synctex.gz
        main.tex
        main.toc
        mypreamble.sty
        title.tex

nothing added to commit but untracked files present (use "git add" to track)
```

What to track

`git add FILE`

```
anna@AP-UniSTR-Laptop:~/Desktop/ERT4H/big project$ git add main.tex
anna@AP-UniSTR-Laptop:~/Desktop/ERT4H/big project$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   main.tex

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        book.bib
        chapters/
        images/
        main.aux
        main.bcf
        main.log
        main.pdf
        main.run.xml
        main.synctex.gz
        main.toc
        mypreamble.sty
        title.tex
```

Keep all records

`git add .`

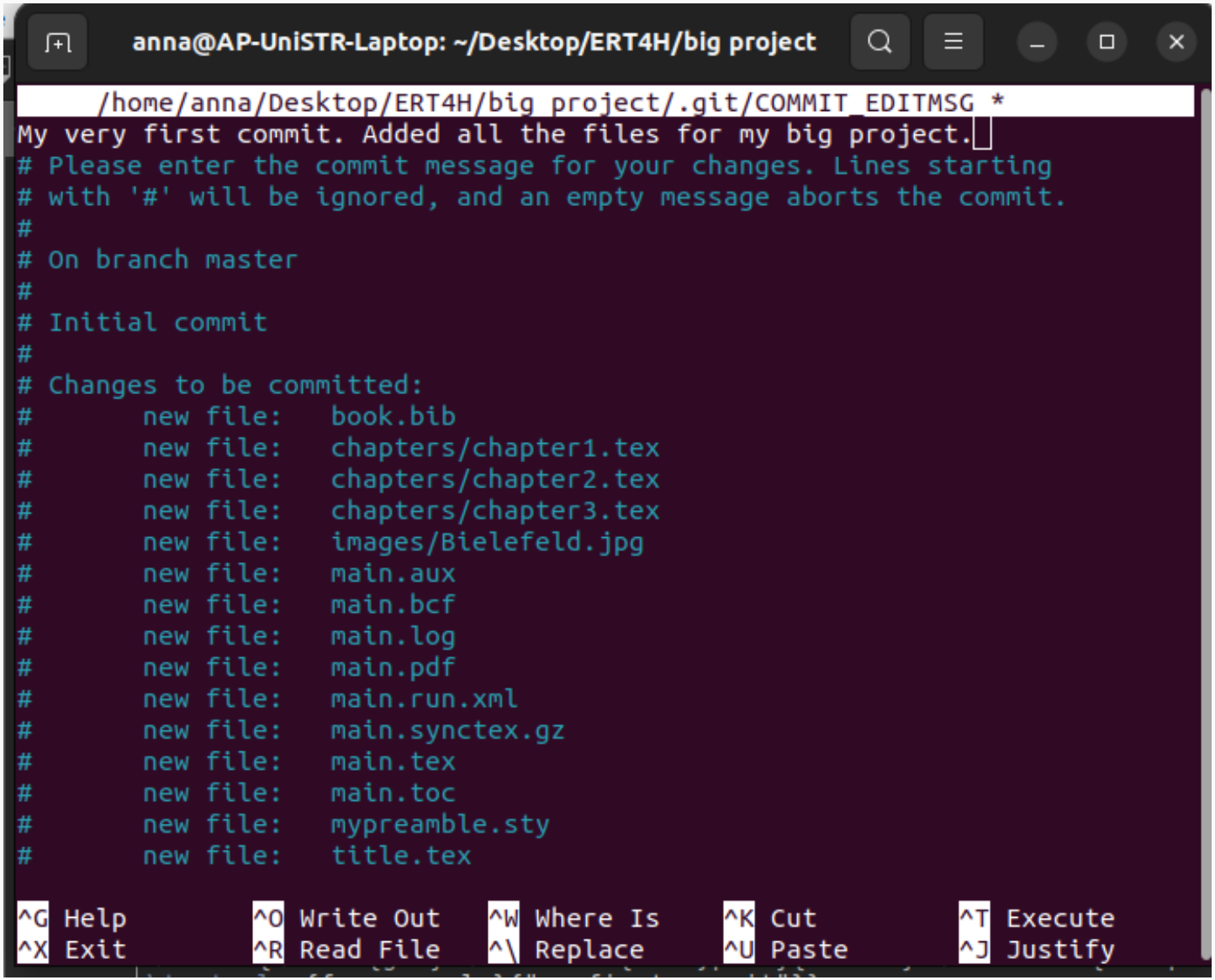
```
anna@AP-UniSTR-Laptop:~/Desktop/ERT4H/big project$ git add .
anna@AP-UniSTR-Laptop:~/Desktop/ERT4H/big project$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   book.bib
        new file:   chapters/chapter1.tex
        new file:   chapters/chapter2.tex
        new file:   chapters/chapter3.tex
        new file:   images/Bielefeld.jpg
        new file:   main.aux
        new file:   main.bcf
        new file:   main.log
        new file:   main.pdf
        new file:   main.run.xml
        new file:   main.synctex.gz
        new file:   main.tex
        new file:   main.toc
        new file:   mypreamble.sty
        new file:   title.tex
```

What have I done?

```
git commit
```



Changes

```
anna@AP-UniSTR-Laptop:~/Desktop/ERT4H/big project$ git commit
[master (root-commit) 0cf6617] My very first commit. Added all the files fo
r my big project.
 15 files changed, 3244 insertions(+)
 create mode 100644 book.bib
 create mode 100644 chapters/chapter1.tex
 create mode 100644 chapters/chapter2.tex
 create mode 100644 chapters/chapter3.tex
 create mode 100644 images/Bielefeld.jpg
 create mode 100644 main.aux
 create mode 100644 main.bcf
 create mode 100644 main.log
 create mode 100644 main.pdf
 create mode 100644 main.run.xml
 create mode 100644 main.synctex.gz
 create mode 100644 main.tex
 create mode 100644 main.toc
 create mode 100644 mypreamble.sty
 create mode 100644 title.tex
```

```
git commit -m "My very first commit. Added all the files for my big project."
```

What’s going on

```
anna@AP-UniSTR-Laptop:~/Desktop/ERT4H/big project$ git status
On branch master
nothing to commit, working tree clean
```


MUCH
LATER

Round 2. Fight!

git status

```
anna@AP-UniSTR-Laptop:~/Desktop/ERT4H/big project$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        chapters/chapter4.tex

nothing added to commit but untracked files present (use "git add" to track)
```

git add .

```
anna@AP-UniSTR-Laptop:~/Desktop/ERT4H/big project$ git add chapters/chapter4.tex
anna@AP-UniSTR-Laptop:~/Desktop/ERT4H/big project$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   chapters/chapter4.tex
```

git commit

```
anna@AP-UniSTR-Laptop:~/Desktop/ERT4H/big project$ git commit -m "Added chapter 4"
[master 0cac288] Added chapter 4
 1 file changed, 9 insertions(+)
 create mode 100644 chapters/chapter4.tex
```

Captain's log. Stardate 20220711

`git log`

```
anna@AP-UniSTR-Laptop:~/Desktop/ERT4H/big project$ git log
commit 0cac2880b8bb89a4eb9eac482705bd250ed9c6f5 (HEAD -> master)
Author: ac140358 <anna.pryslopska@ling.uni-stuttgart.de>
Date:   Sat Jul 9 21:48:05 2022 +0200

    Added chapter 4

commit 0cf6617eb06dd6c513f2965b452ad76650544195
Author: ac140358 <anna.pryslopska@ling.uni-stuttgart.de>
Date:   Sat Jul 9 21:33:30 2022 +0200

    My very first commit. Added all the files for my big project.
```

Documentation

Your first commit should be a `readme.md` file.

```
1   # Big project
2
3   What is this? This project is authored by Anna
    Prysłopska.
4
5   This is my Big Project. This can be my BA thesis, a
    novel, the next big app that Facebook will buy (I don't
    subscribe to their rebranding), or anything else.
6
7   ## Table of Contents
    Table of Contents (out of date)
8   - [Installation](#installation)
9   - [Usage](#usage)
10  - [Configuration](#configuration)
11  - [Examples](#examples)
12  - [Contributing](#contributing)
13  - [Contact](#contact)
```

Git and GitHub best practices

Commit often

Make small commits often to make tracking changes and finding mistakes easy.

Review changes before committing

Use `git diff` to check what changes were made to the files you're committing.

Meaningful messages

Write clear and concise commit messages.

✖ "Add files"

✔ "Added participant information to the methods section"

Avoid committing large or personal files

Exclude unnecessary files and sensitive information. Keep it minimal and use `.gitignore`. GitHug will reject files that are too large.

Branch out

Use branches when collaborating, adding new features, fixing bugs, and experimenting. Keep the main branch stable.

Documentation

Keep the documentation up to date (e.g. `readme.md`), include information about setting up, dependencies, examples, contact information, known issues etc.

Git and GitHub best practices

Others: regularly review code, protect branches, track issues, organize milestones, document in a wiki

Questions?

Summary

- ✔ Text editors and their uses
- ✔ Version control
- ✔ Git basics
- ⏩ More git, GitHub, SSH

Read more

Git guide: `https://github.com/git-guides/`

Another git guide: `http://rogerdudler.github.io/git-guide/`

Git tutorial: `http://git-scm.com/docs/gittutorial`

Another git tutorial: `https://www.w3schools.com/git/`

Git cheat sheets: `https://training.github.com/`

Ask questions: `https://stackoverflow.com`

Homework assignment

Homework assignment due July 12th at 15:30

- Complete assignment 12 ($\rightarrow$ ILIAS)