Quantitative Research Workflow

Francisco Villamil

UC3M – February 13, 2024

Thinking about workflow

- How you organize your coding projects: data, output, integration between different things (e.g. R and Latex)
  - → (Note on R Markdown)
- How to code better
- Learning how to use a computer properly

Why?

1. **Automate stuff:** you spend a lot of time on the computer so make it work for you
2. **Avoid errors:** we should not trust ourselves

Problems

- Version control: `final.docx`, `final2.docx`, `finalFINAL.docx`
- Going back to old (or not so old) projects and...
  - → change something in one 5000-line R file
  - → doesn't run because file is missing, where's the file?
  - → need to change some *constant* and spend a whole day looking for it
- Re-running versions of graphs and tables in the final dissertation
- Avoiding this things:
  journals.sagepub.com/doi/10.1177/2053168022112645 4

Problems

- Version control: `final.docx`, `final2.docx`, `finalFINAL.docx`
- Going back to old (or not so old) projects and...
  - → change something in one 5000-line R file
  - → doesn't run because file is missing, where's the file?
  - → need to change some *constant* and spend a whole day looking for it
- Re-running versions of graphs and tables in the final dissertation
- Avoiding this things:
  journals.sagepub.com/doi/10.1177/2053168022112645 4

Problems

- Version control: `final.docx`, `final2.docx`, `finalFINAL.docx`
- Going back to old (or not so old) projects and...
  - → change something in one 5000-line R file
  - → doesn't run because file is missing, where's the file?
  - → need to change some *constant* and spend a whole day looking for it
- Re-running versions of graphs and tables in the final dissertation
- Avoiding this things:
  journals.sagepub.com/doi/10.1177/2053168022112645 4

Problems

- Version control: `final.docx`, `final2.docx`, `finalFINAL.docx`
- Going back to old (or not so old) projects and...
  - → change something in one 5000-line R file
  - → doesn't run because file is missing, where's the file?
  - → need to change some *constant* and spend a whole day looking for it
- Re-running versions of graphs and tables in the final dissertation
- Avoiding this things:
  journals.sagepub.com/doi/10.1177/2053168022112645 4

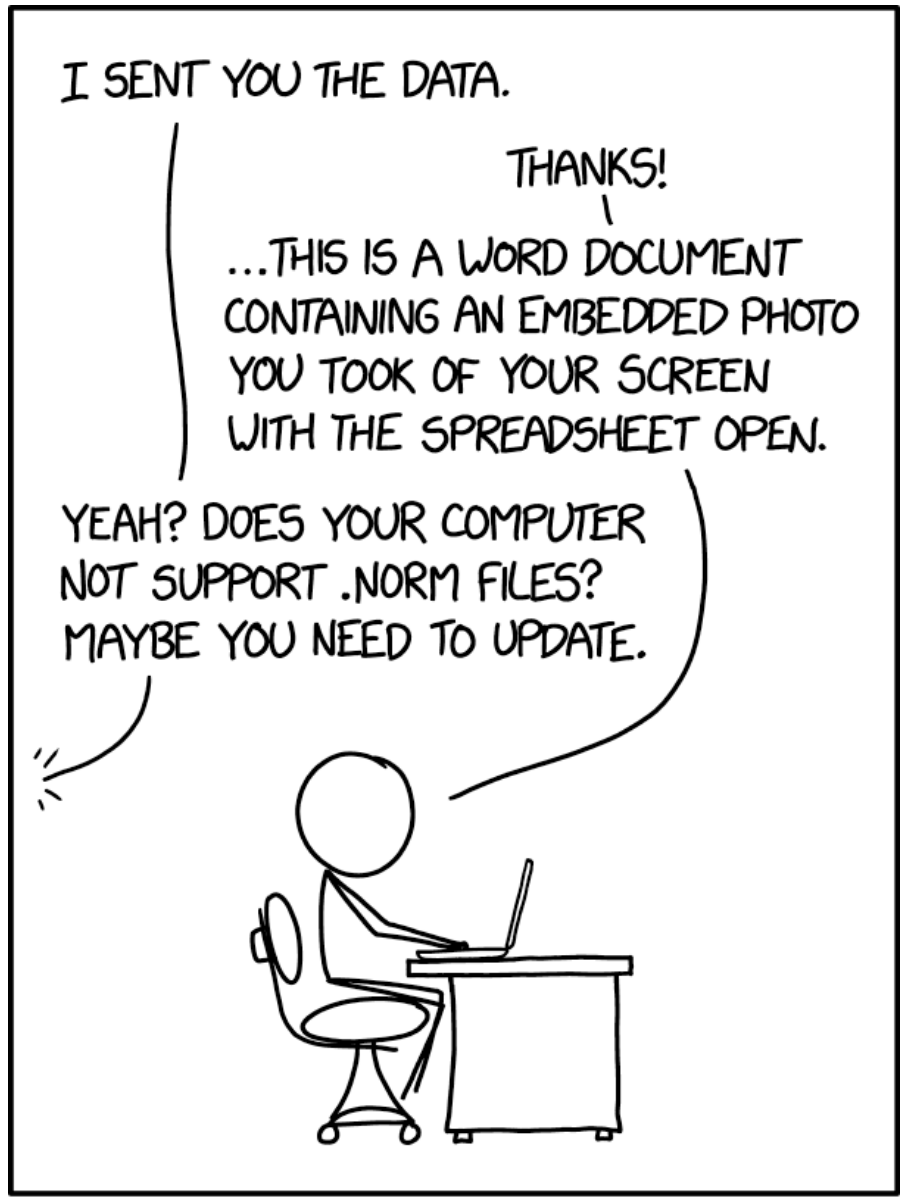
Main ideas

1. Working with **plain text** files
2. Organizing coding and empirical projects
3. Coding better (automating, defining constants, checks & warnings...)
4. Use version control (git)
5. Use your tools better: learn some command line, customize computer, choose a text editor...

Some resources

- Hadley Wickham's R Style guide (and the whole Advanced R book later on)
- Kieran Healy's *The Plain Person's Guide to Plain Text Social Science*: https://plain-text.co/
  - → Although `emacs` is perhaps a bit too hardcore
- The best Git course I know is this: https://gitexercises.fracz.com/
- MIT's *The Missing Semester of Your CS Education*: https://missing.csail.mit.edu/
- Software Carpentry's lessons: https://software-carpentry.org/lessons/
  - → Especially Unix Shell and Version Control with Git
- **github.com/franvillamil/workflow_example**

What is **plain text**?



SINCE EVERYONE SENDS STUFF THIS

Roadmap

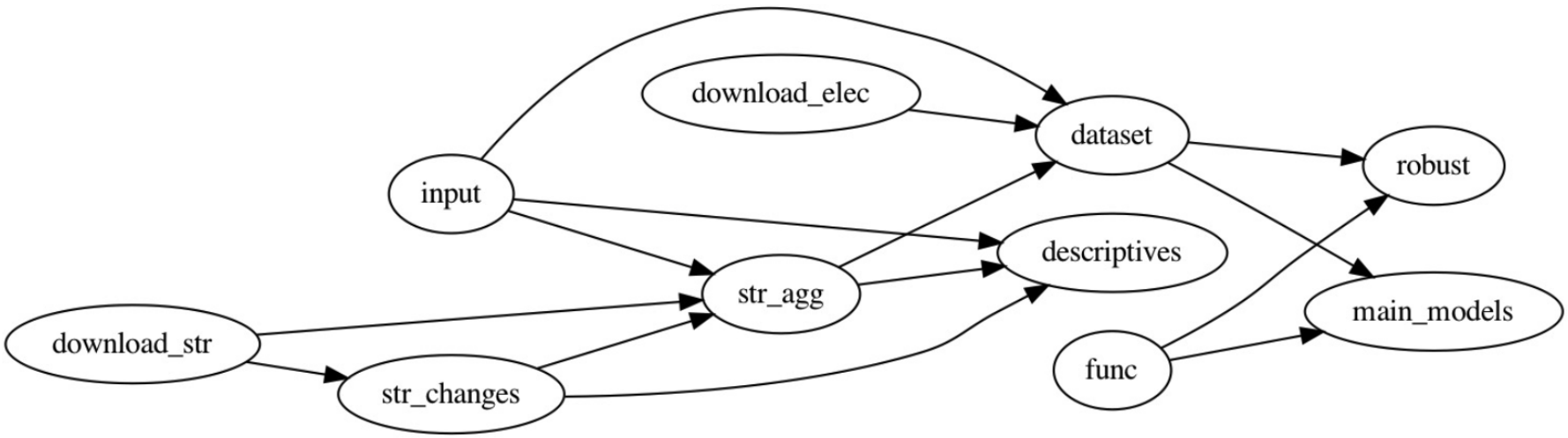
Coding better and organizing data projects

Using computers

Coding projects: tasks as folders

- This applies especially to the R part of projects
- Do not create one huge R code file, use different files for different tasks
- You want to do the same with the folder structure
  - → **Especially with R output!**
- (Optionally, you can use *Makefile*, but it can be problematic)

Coding projects: tasks as folders



Coding projects extra: filenaming

- Not a lot of things here, but just think about how you name files or folders

1. Do not use spaces
2. Ideally use some standards (e.g. `1_clean-file.R`, `2_merge.R` ...)

- **DON'T:** `First file.csv`, **definitely not:** `Datos educación.csv`
  - → good: `first_file.csv`, `datos_educacion.csv` etc

Coding projects, extra: integrate

- Put together `Latex` and `R` parts (and/or Python, etc)
- If you organize the folder as I said before, it's pretty much solved
- Overleaf example
- What you get with this? Avoid mistakes, makes your life easier...

Coding projects, extra: Makefile

- Example
- https://makefiletutorial.com/

Writing code I: automate via functions

- Main idea: you should not write the same code *twice*
- Use functions to automate everything: producing plots, analyses (e.g. predicted probabilities), recurrent actions, etc
- Why? Avoids mistakes, easier to debug, cleaner files
- Examples:
  - → https://github.com/franvillamil/ethnicity_africa/tree/master/func
  - → https://github.com/franvillamil/streets_vox/tree/master/func

Writing code II: write checks and warnings

- As you write code, always write checks using `stop()` or `warning()`, e.g.:
  - → if a new data frame is built from merging, what should be the number of rows in the final df? or columns?
  - → should two objects be identical?
  - → do we have duplicated values by some ID?
  - → do you expect ‘‘145‘‘ or 145 (character vs integer)?
  - → ...

```r
## Load

# Pre-invasion data
pre_data = read_dta("data/test_revisado.dta", encoding = "latin1") %>%
  left_join(read_dta("data/soft_300.dta", encoding = "latin1")[, c("response_id", "TS")]) %>%
  mutate(date = as.Date(str_sub(TS, 1, 10), "%m/%d/%Y"), post = 0) %>%
  rename(trust_army = Q49) %>%
  rename(Q41 = Q41_h)# labeling error

# Post-invasion data
post_data = read_dta("data/test5_revisado.dta", encoding = "latin1") %>%
  mutate(date = as.Date(str_sub(ts, 1, 10), "%m/%d/%Y"), post = 1) %>%
  # filter(date <= as.Date("2022-03-10")) %>%
  rename(trust_army = Q46)

## Checks
if(!identical(attr(pre_data$Q11, 'label'), attr(post_data$Q11, 'label')) &
  identical(attr(pre_data$Q42, 'label'), attr(post_data$Q42, 'label')) &
  identical(attr(pre_data$Q48, 'label'), attr(post_data$Q48, 'label')) &
  identical(attr(pre_data$Q43, 'label'), attr(post_data$Q43, 'label')) &
  identical(attr(pre_data$trust_army, 'label'), attr(post_data$trust_army, 'label')) &
  identical(attr(pre_data$Q47, 'label'), attr(post_data$Q47, 'label'))){stop("!")}
```

```r
mentions_by_url = function(filename, keywords){

  # Set up
  month = gsub("output/webs_(\\d+-\\d+)\\.rds", "\\1", filename)
  df = url_df[url_df$month == month,]

  # Read
  raw = readRDS(filename)

  # Check
  if(length(raw) != nrow(df)){stop("diff length df/raw! (1)")}
```

Writing code II: write checks and warnings

- Also try to minimize errors, e.g. that you have visuals of real output, e.g.:

1. I use `print()` all the time to show length of stuff, number of missing data, etc
2. `modelsummary` vs `stargazer` example
   → journals.sagepub.com/doi/10.1177/2053168022112645 4
   → github.com/franvillamil/streets_vox/blob/master/robust/rob.R

Roadmap

Coding better and organizing data projects

Using computers

Plain text

- Quicker and easier to work with
- Cross-platform and does not depend on proprietary software
- Much better for the things you want to do
  - → You can use version control on it
  - → Closer to how machines work it - so easier for whatever related to machines (e.g. syncing two computers) → example1, example2
  - → It's a base ingredient you can convert into whatever (e.g. with `R`, `LaTeX`, etc)

Customizing your computer

- https://franvillamil.github.io/posts/setup_macos.html
- https://github.com/franvillamil/templates
- https://github.com/franvillamil/configfiles
- Examples: mdtopdf/docxtopdf, baserepos, Spectacle, ...

Code editor

- Choose and get used to some code editor
  - → You're probably using the editor in `RStudio`, that's fine, but there are reasons to use better and more general tools
- You can customize these so suit your needs, e.g.:
  - → Edit & run languages you use (`R`, `Latex`, whatever)
  - → Small stuff that saves time, like snippets
  - → Navigate a project
  - → And much more complicated stuff we're not going to talk about and that I do not know so much about
- I use Sublime Text: https://www.sublimetext.com/
- Anyway, **don't use MS Word** (as much as possible)

Using the command line

- Think of it as the language to communicate with the OS
- No need that you become a computer wizard, but I personally think it pays off to learn a little bit
- Why?
  - → Automate stuff in the computer (e.g. from updating local files to converting .docx into pdf)
  - → Navigate and work with files faster
  - → Version control, installing stuff, solving issues
  - → Virtual machines
- **Note:** Unix/Mac vs Windows

Version control (Git)

- `final1.docx`, `finalfinal.docx`... but in a proper way

Version control (Git)

- `final1.docx`, `finalfinal.docx`... but in a proper way
- You want to keep control of all versions of a file, something like MS Word's 'Tracked changes' but just much better

Version control (Git)

- `final1.docx`, `finalfinal.docx`... but in a proper way
- You want to keep control of all versions of a file, something like MS Word's 'Tracked changes' but just much better
  - → Keep a time machine of all versions of a file

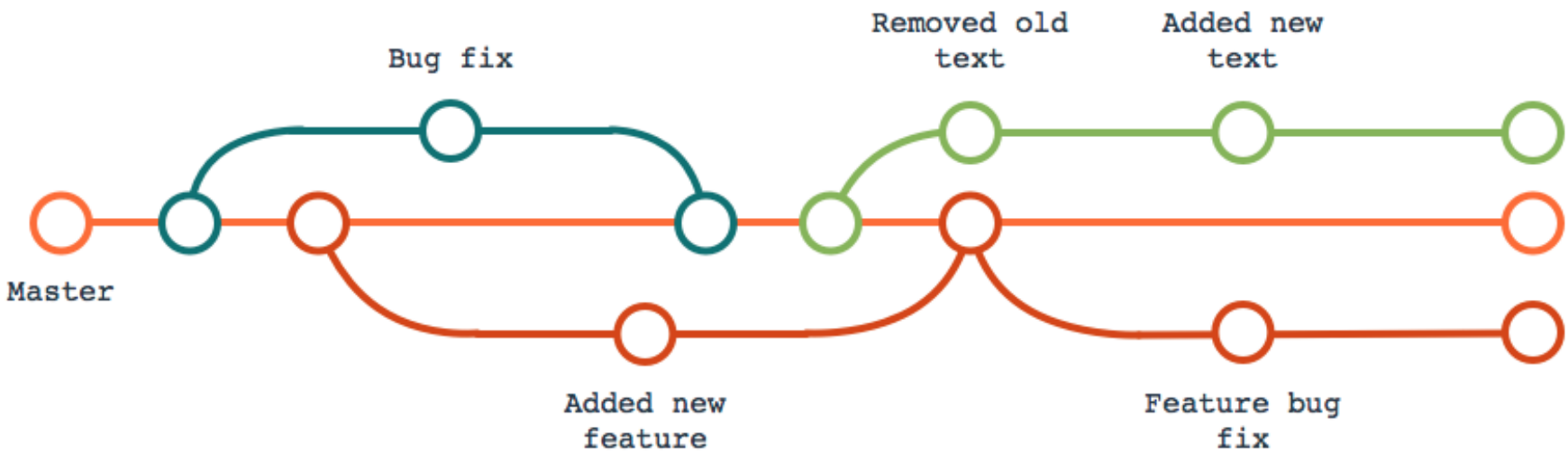
Version control (Git)

- `final1.docx`, `finalfinal.docx`... but in a proper way
- You want to keep control of all versions of a file, something like MS Word's 'Tracked changes' but just much better
  - → Keep a time machine of all versions of a file
  - → Allow collaboration between different people (or between two computers)

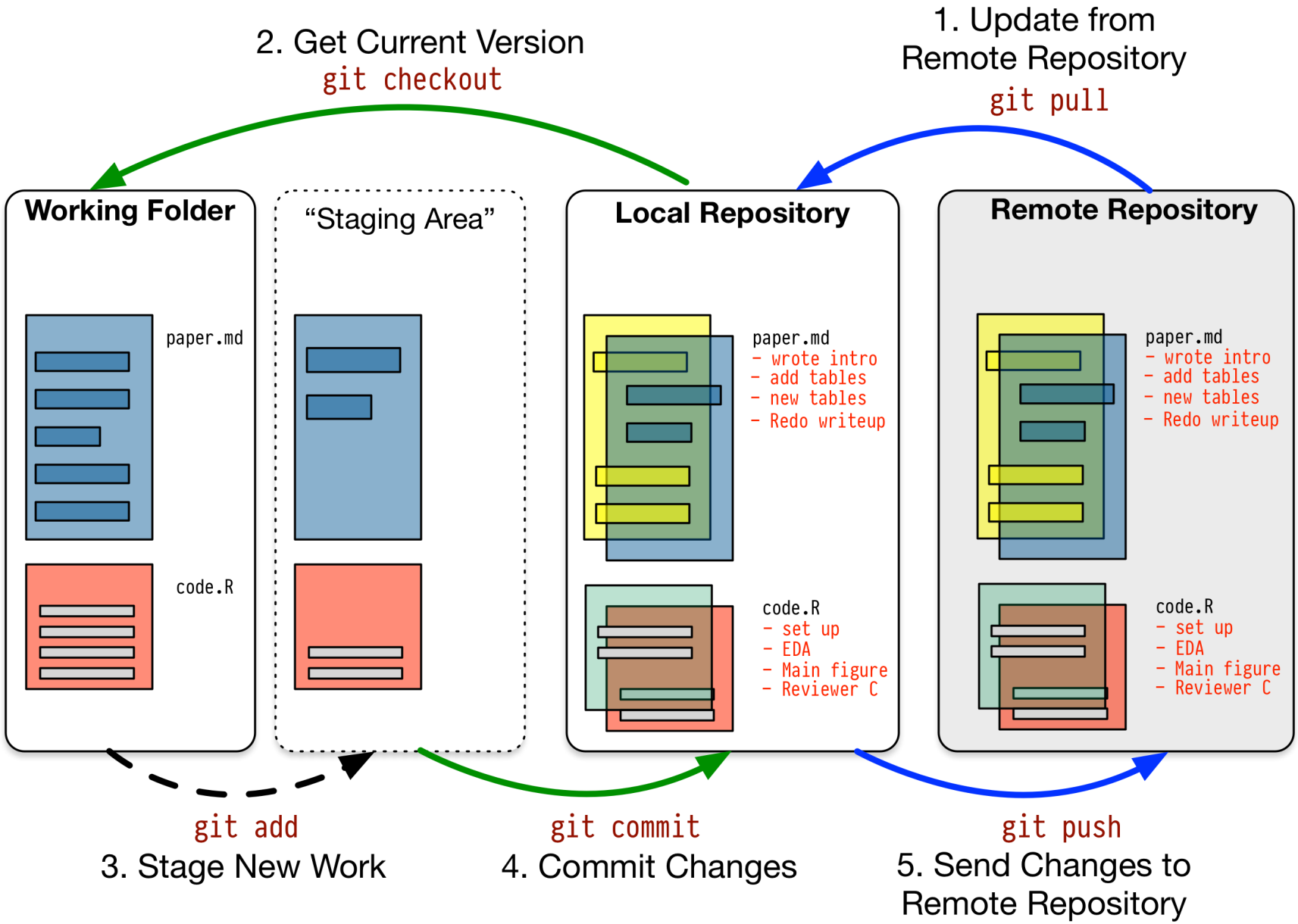
Version control (Git)

- `final1.docx`, `finalfinal.docx`... but in a proper way
- You want to keep control of all versions of a file, something like MS Word's 'Tracked changes' but just much better
  - → Keep a time machine of all versions of a file
  - → Allow collaboration between different people (or between two computers)
- There is more than one system, but most people use `Git` (and Github)

Version control



Version control



Version control - a note

- Version control works **much** better if you work with other people who also use version control, which is often not the case (at least not mine)
- Yet, there are two advantages to use it in my view:
  - → Obvious one: keep older versions of a file
  - → If you work with two computers, perhaps Google Drive/Dropbox do not work that well
  - → Virtual machines (e.g. Google Cloud Computing, Amazon Web Services)