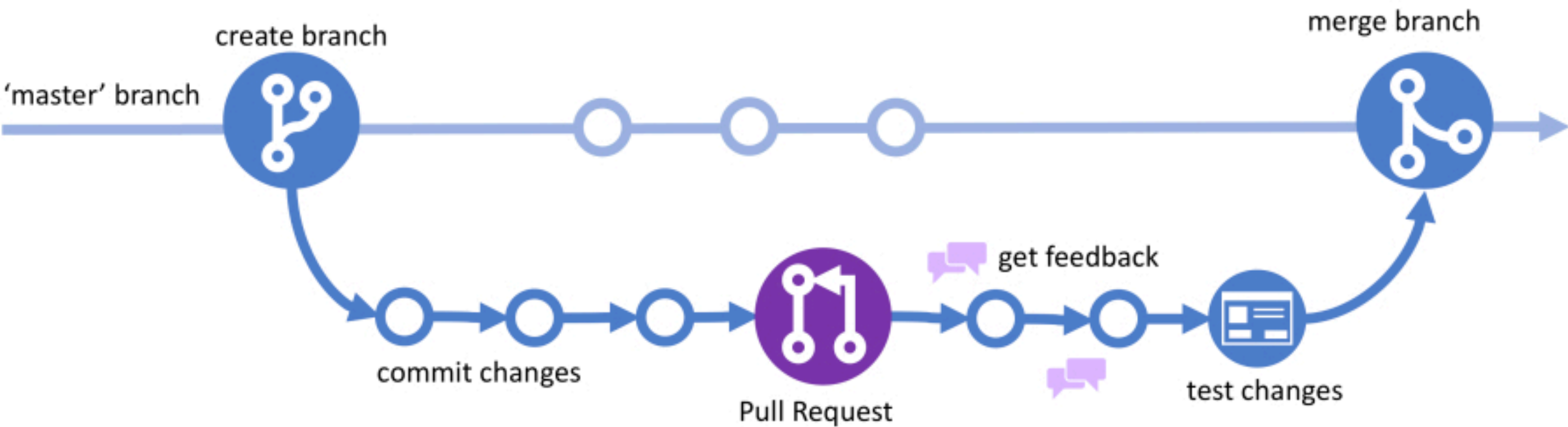


Version Control

Melbourne Statistical Consulting Platform
University of Melbourne
April 2024

GitHub Flow



Have you ever had this? If not, you will...

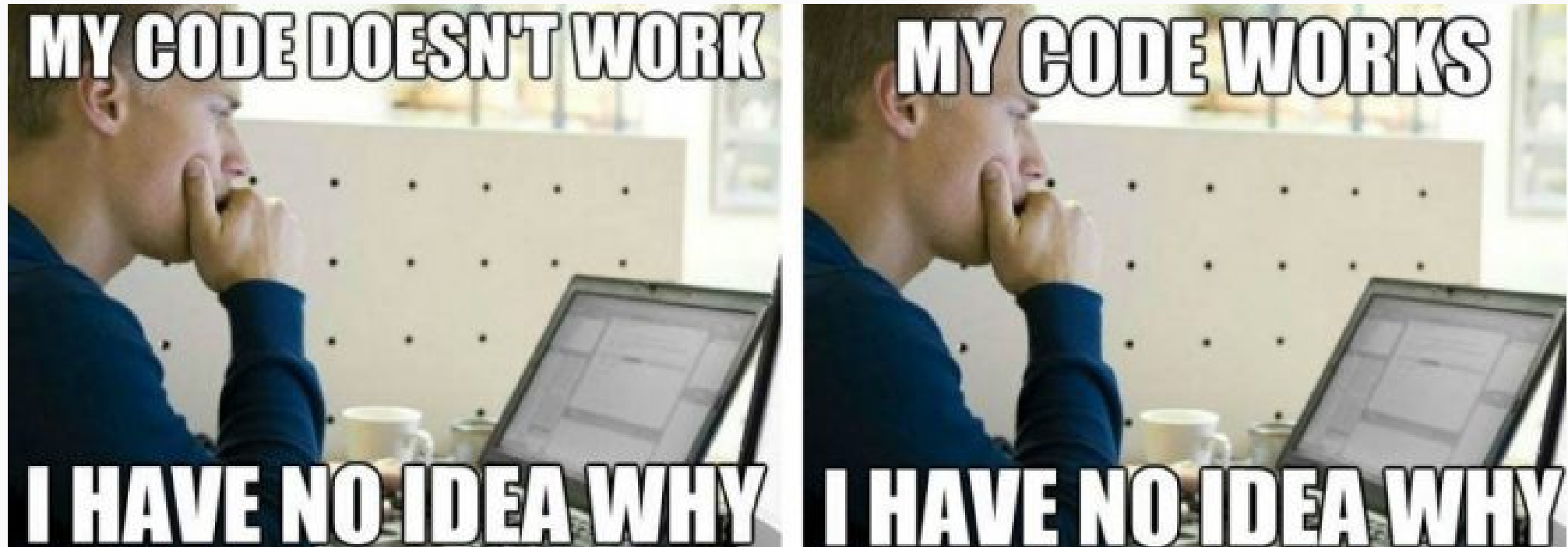


Image from meeum.com ([Webpage](#)).

Motivation



- My code is broken and I have no idea what I've done!
- I can run that script as it is today, but I don't know how to remake that figure from last year...
- The reviewer wants me to update that figure, but I've since modified the script and deleted that bit!
- We have different versions of the code running in our lab - it is simply a mess!
- I have no idea why I changed it -- there aren't any comments -- but I can see it changed the results...

Image from Jose Luis Navarro, published on Wikimedia.org ([Source](#)).

What is this version control thing...?!

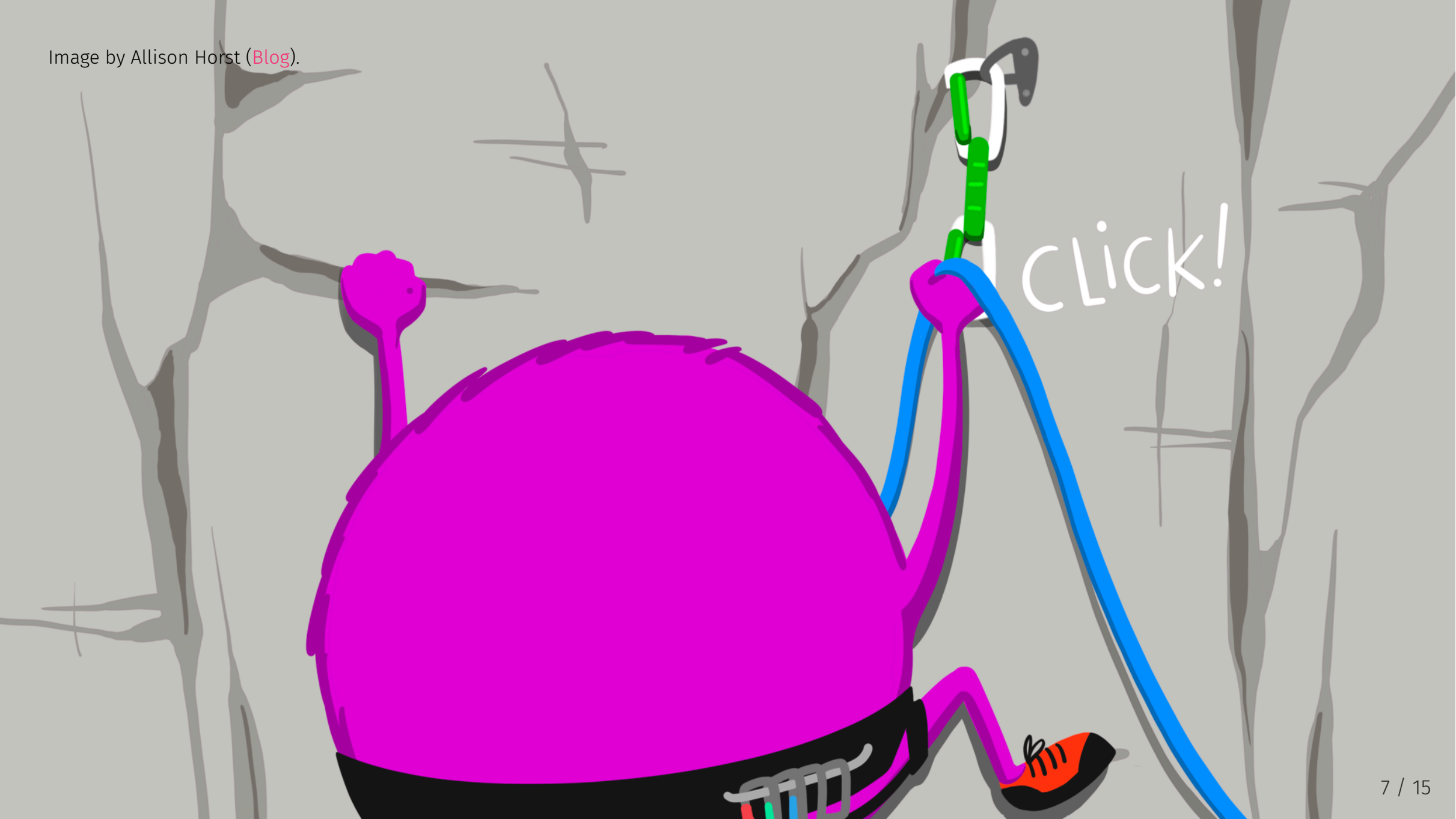
- It's like "track changes" but for text documents
- You can revert to or compare with old versions
- I think of it as a really valuable safety net, and safety nets allow you to be bolder
- Helps keep files tidy
 - Avoids many versions of the file: `analysis.Rmd`, `analysis_forSam.Rmd`, `analysis_Oct22.Rmd`, `analysis_old.Rmd`, `analysis_new.Rmd`, `analysis_new_new.Rmd`, `analysis_new_new_b.Rmd`, ...
 - You can delete sections of code that have already been logged/committed, rather than keeping lots of commented lines.
- It helps keep track of different versions of your work
- You can also use it for other things (e.g., Latex/markdown manuscripts)

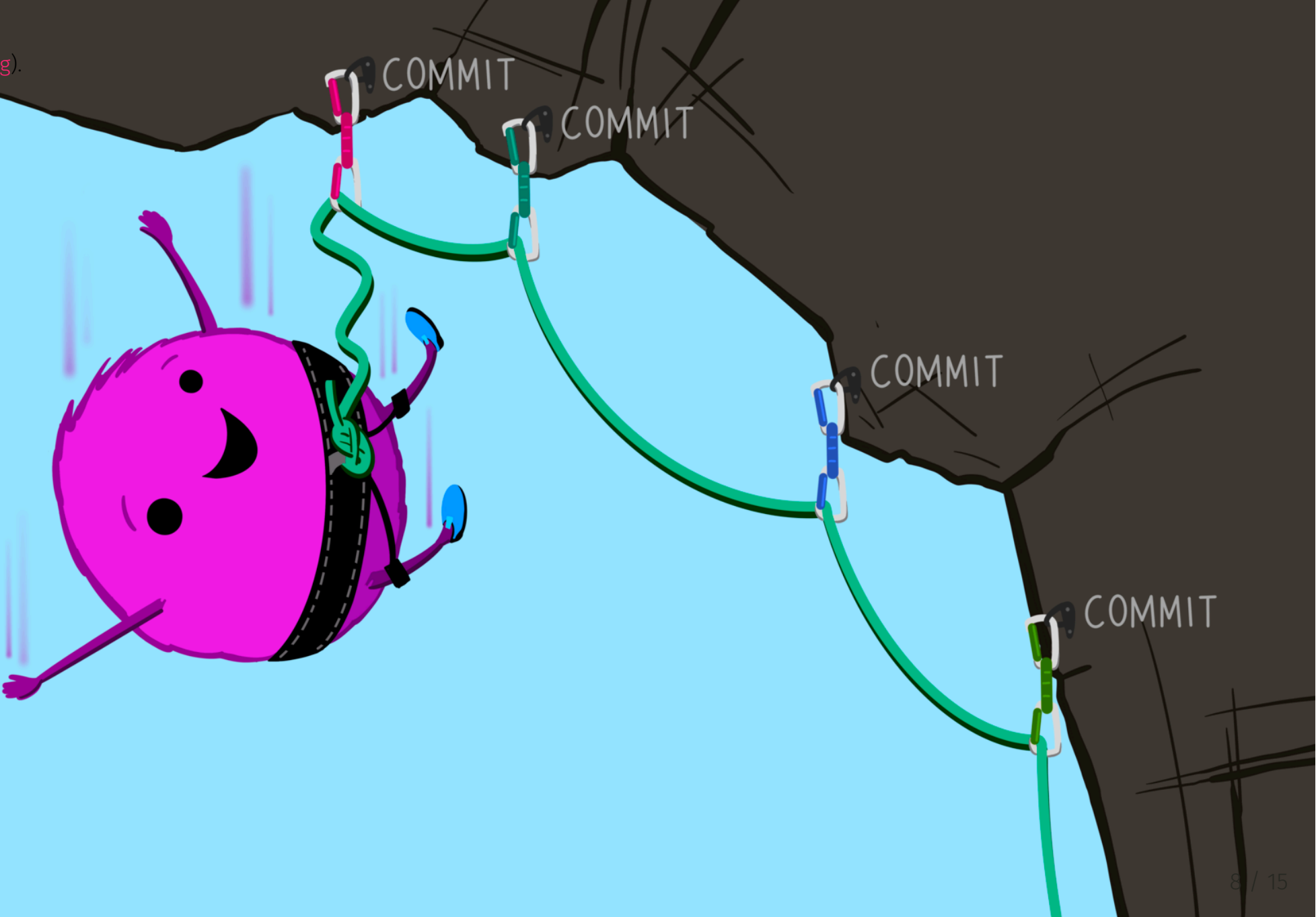
The simplest workflow

1. Initialise your *repository*
2. Add some files
3. *Commit* the new changes
 - Write a comment in the *commit message*
4. Make changes to them, add more files, remove files from repository
5. Commit your changes, again writing a comment

Repeat steps 4. and 5. as you keep working

Image by Allison Horst ([Blog](#)).





Version control software

- Distributed version control
 - Many repositories can work independently
 - Examples: *git*, mercurial
- Centralised version control
 - One central repository
 - Example: *subversion*
- RStudio can talk to git and subversion
 - Neither come packaged with RStudio: install them separately
 - <https://docs.posit.co/ide/user/ide/guide/tools/version-control.html>
- For single-user projects, it won't matter much which you use. Beyond that, you will need to review the options.

Good practices (1)

- Commit early, commit often
- Write descriptive messages

Example commit messages (or what to avoid doing)

Good:

Fixed bugs in `read_sample()`

- There were too many commas among the arguments
- Referred to a variable that had not been defined
- Now loads additional required libraries

To be avoided: `ASDF`, `fixed Scott's bug`, `Update 20th March 2024`

Good practices (2)

- Avoid tracking certain files
 - Output from the code (e.g., log-files, plot files or exported datasets produced by your scripts)
 - Big datasets
- Tell git to *ignore* certain files or groups of files
 - e.g. `*.jpg`, `*.html`, `data/`

Some things to be aware of

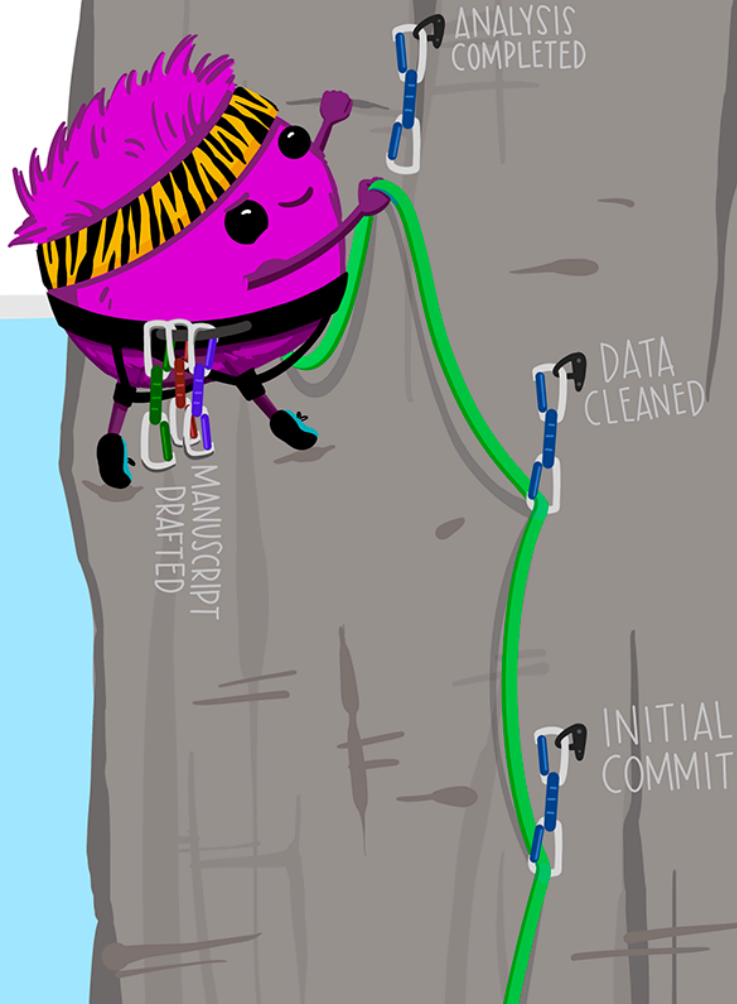
- Remote vs local repositories
 - Remote repos can be on the same computing system, or on the web (e.g., github, bitbucket)
 - Your local repository is the one you are working on
 - Good for collaboration, publishing your code, or as back-up
 - Can be public or private. There are education accounts to Github & Bitbucket
- Between changes and commits is *staging*
 - You *stage* your changes, then you commit them
 - You don't need to stage everything: maybe just some files

Image by Allison Horst ([Blog](#)).



Image by Allison Horst ([Blog](#)).

When working with GitHub, we can navigate with more obvious, safe, streamlined routes that let us focus on the science-y things we want to do...



...but working without GitHub can be disorienting, with too much time spent sifting through past work to figure out next steps forward.

