Introduction to version control with git

Rachael Stickland

Data Wrangler, The Alan Turing Institute

21 June 2023

AI for Multiple Long-term Conditions
Research Support Facility

Check installation of git and GitHub Desktop

Follow the installation instructions previously shared

Everyone needs:

1. GitHub Desktop installed:

   https://desktop.github.com

2. Three tutorial data files downloaded to your computer:

   https://github.com/aim-rsf/training/tree/main/version-control/super-sandwich-survey

Introduction to instructor

Rachael Stickland

BSc in Psychology & **Research Assistant**

- File naming for version control

PhD in Neuroscience & Neuroimaging

- Used GitHub to access *other people's* data analysis code & keep track of versions

Postdoctoral Research Fellow (Neuroimaging)

- Used GitHub regularly to work with my past & future selves
- Led creation of GitHub workflow for small research lab
- Contributed to open science GitHub projects
- Started to share analysis code alongside research publications

Data Wrangler at The Alan Turing Institute

- Started using GitHub for project management
- There is always more to learn …

Learning objectives

- understand what version control is
- understand why version control is useful
- understand what git is
- understand where you can use git
- be able to explain what these words mean in a git context: ***repo***, ***stage***, ***commit***, ***history***, ***branch (main)***, ***amend/undo/revert***
- be able to apply the above words/commands within a git GUI to version control your files

Theory and background

Version control

What is it

Why use it

Challenges

Solutions

Solutions (git)

git and GitHub

Version control is an approach to **recording changes** in a file or set of files **over time** so that you and your collaborators can: track their history, review any changes, and go back to earlier versions.

Version control

Why is version control useful?

It's a great data management and documentation tool

- Work with the latest version of a file
- Backup previous versions of files
- Save edit history (it's clear what you did & you have the option to revert back)
- Test out new features
- Collaborate with others and manage projects

Version control

What is it

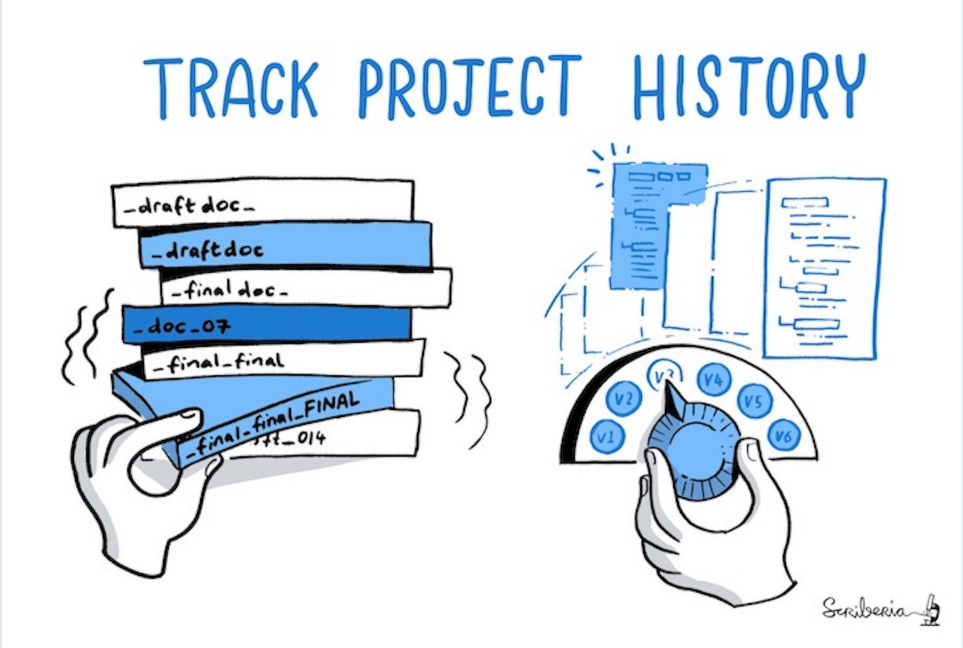
Why use it

Challenges

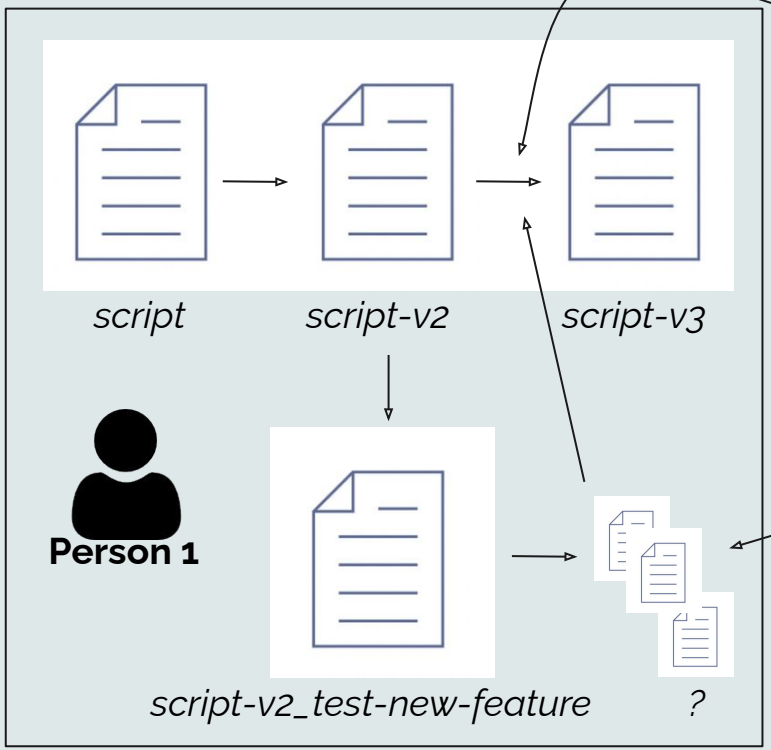
Solutions

Solutions (git)

git and GitHub



The Turing Way project illustration by Scriberia
DOI: 10.5281/zenodo.3332807.
Used under a CC-BY licence.



Person 3

Suggests a change but doesn't edit the files

?

Person 2

Helps with testing new feature

Person 1, 2 & 3

Meta document

Project history

Next steps

Resources

Bugs

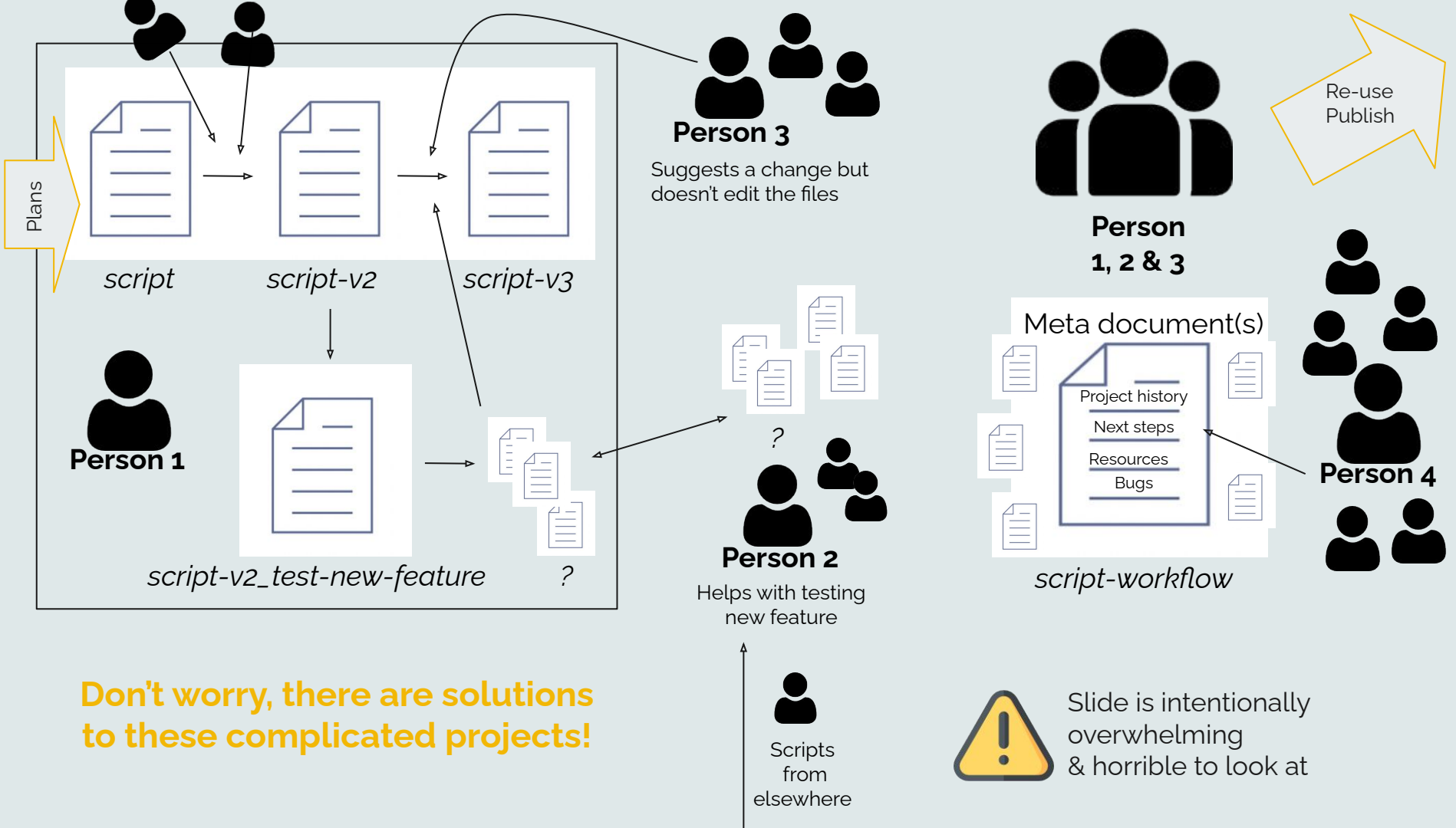
script-workflow

Person 4

Time has passed

- Where is my latest version of that script?
- When did I incorporate that new feature?
- Who made that change and why?
- Collaborator asked me to run script-v2 …

Slide is intentionally overwhelming & horrible to look at


Plans
script
script-v2
script-v3
Person 1
script-v2_test-new-feature
?
Person 3
Suggests a change but doesn't edit the files
?
Person 2
Helps with testing new feature
Scripts from elsewhere
Person 1, 2 & 3
Re-use
Publish
Meta document(s)
Project history
Next steps
Resources
Bugs
Person 4
script-workflow
Don't worry, there are solutions to these complicated projects!
Slide is intentionally overwhelming & horrible to look at

Version control

Versions in multiple files

A straightforward way to version files is by adding versions or explanatory labels to the file name:

v1, v2, v3 etc. draft, commentsAB, etc.

You are probably already doing some form of version control

This may be easy for one file or one user but it can be messy for lots of files and users. More sophisticated version control systems may be needed

Versions in a single file

Some software record all the changes in a file, allowing you to explore its history:

- Basic: Google Drive, Dropbox, Overleaf
- Advanced: Subversion, git

Versions in a single file

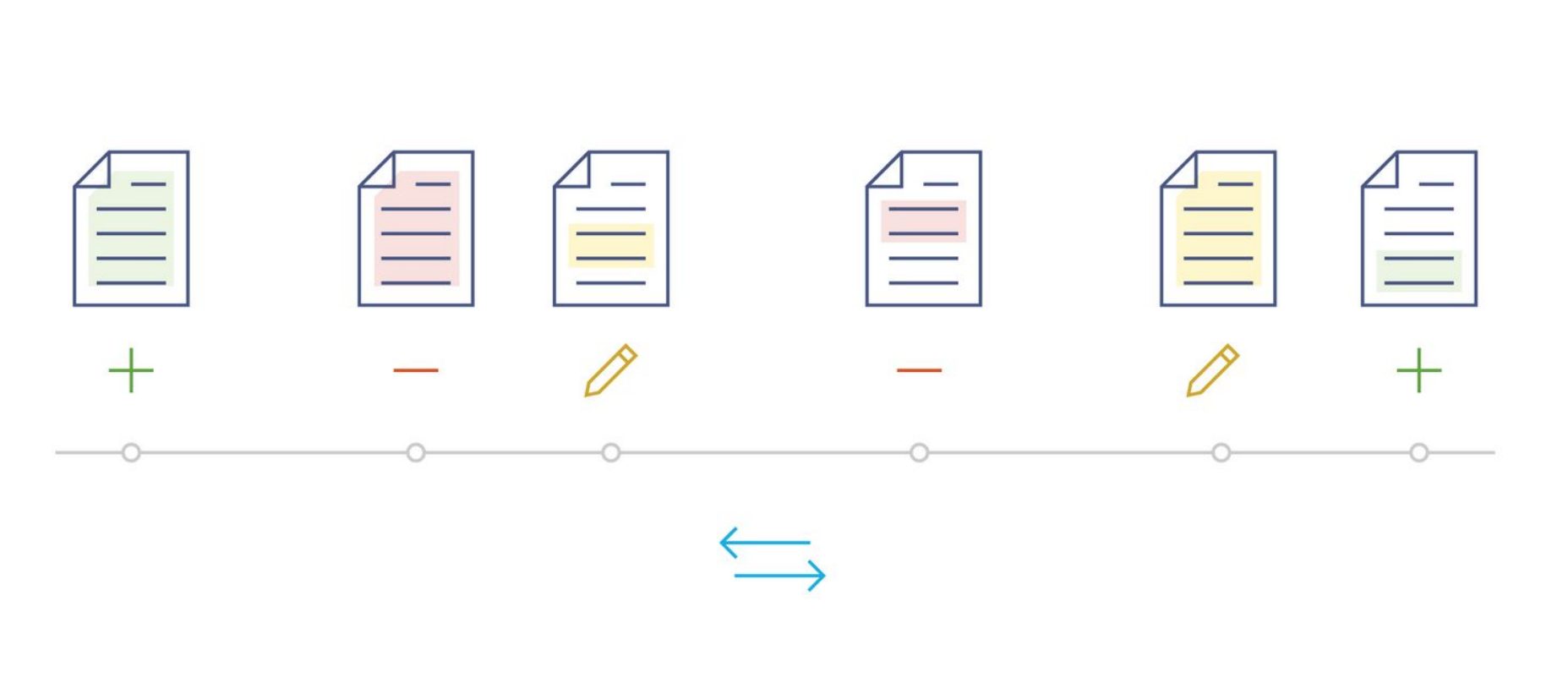


Image from Chapter 3 of the *Mozilla Science Lab's Study Group Orientation handbook*, used under a Mozilla Public License Version 2.0.

Version control

What is it

Why use it

Challenges

Solutions

Solutions (git)

git and GitHub

Key features of a version control system (VCS)

(SCM)
Source
Code
Management

When we edit a file, we want to record:

1. The content of the edit
2. When it was edited
3. Why it was edited
4. Who edited it

Snapshots of your project will be tracked with a unique code (recording features 1-4). A new file isn't saved, only the changes made to the file.

File revisions at each snapshot can be compared, restored (and sometimes) merged.

Version control

What is it

Why use it

Challenges

Solutions

Solutions (git)

git and GitHub

https://git-scm.com

Git is a distributed version control system

- There are others, but Git is the most widely used
- Created by the Linux development community in 2005
- Git is software, mostly written in C

Selling points of Git:

- Free, Fast, Open source
- Distributed not centralised (everyone has a copy of whole project)
- Most operations need only local files and resources
- History of changes (who, when, why, what)
- Retrieve previous workflow
- Foundation to the collaboration tools (GitLab, GitHub)

Git takes snapshots of your project (when you chose), then compares to the previous snapshot & asks you to record why you are making this change.

Version control

https://git-scm.com

Okay, why the funny name?

The name "git" was given by Linus Torvalds when he wrote the very first version

Torvalds is known for creation/development of the Linux kernel

> *I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'git'.* https://en.wikipedia.org/wiki/Git

"git" can mean different things, depending on your mood.

See the **README.md** file at **https://github.com/git/git** for multiple reflections on the name

Version control

What is it

Why use it

Challenges

Solutions

Solutions (git)

git and GitHub

A free **version control system** that lets you manage and keep track of your file history and retrieve previous workflows, **locally** on your computer.

A popular **cloud-based** hosting service for sharing projects you are tracking with git. Many features for **collaborating** with others. Good for backups.

How do I interact with git (locally)?

Many many options

Command line / terminal
We won't be covering this approach but if you are comfortable with the command line check out tutorials & git cheat sheets we've shared

Extensions to existing software
Your favourite text editor or analytical software is very likely to have a git extension/plugin, often via a GUI.
Examples: Visual Studio Code, R Studio

Specific Git clients (applications with a GUI)
Examples: GitKraken, GitHub Desktop

GUI: Graphical User Interface

Misconception 1:

`git` == GitHub

Misconception 2:

`git` is only for code

Misconception 3:

`git` is only for collaborating

Misconception 4:

`git` does not ‘auto-save’

Common challenge:

`Strange new words` 

Break & Questions

Interactive part

Let's  going!

Using GitHub Desktop we will:

Create a new repository
A repository (or repo) is similar to a folder/directory, where all files are stored for a project. It includes files used by *git* to track changes.

See how *git* interacts with us when we:

- **Edit** an existing file
- **Add** a new file
- Edit **multiple files** at the same time
- **Delete** a file
- **Rename** a file
- **View history** of our repo
- **Revert** to a previous state

Git calls taking a snapshot (saving your work at a moment in time) a **commit**

Open up GitHub Desktop
&
Follow along

See HackMD for detailed steps

Technical concepts

git concepts & commands: setting up

- **repo** is short for **repository**. Similar to a folder/directory, this is where all files are stored for a project, including files git uses to track changes

- First we ask git to pay attention to our files (**initialize** a repo, **git init**)

- Git will give us a default **branch** to work in which is called **main**
  - we will return to branches in lesson 2, covering why you might want to create another branch of a different name
  - moving between branches uses the **checkout** command

git concepts & commands: making changes

- **initializing** a git repository creates a subdirectory named **.git** that contains all necessary repository files and **logs**. But nothing is being automatically tracked! We need to take a snapshot …

- When we make a change to a file (addition, deletion, edit) we add it to a staging area (**git add**), to get it ready for the snapshot to be taken. We can view how it differs to the previous snapshot.

- A snapshot in git language is called a **commit**. This is like pressing the save button.
  - Once we've made all the changes we want, we take a snapshot of the staging area. We write a **commit message** explaining the motivation for the changes.

git concepts & commands: managing commits

- Viewing the git **logs** shows the history of the repository
    - Each commit will have four main features attached: a message, a time, a username, and a unique ID
- Using commit features we can **checkout** to previous states, and **reset** or **revert** previously applied changes

Summary of lesson 1

Summary

- **Version control** allows us to record changes in a set of files over time
- **Version control is useful** as a data management tool, allowing us to track history, review changes and go back to earlier versions
- **Git** is one of the most widely used version control systems
- **You can use git locally** via the command line, software extensions or git clients (e.g. GitHub Desktop)
- **GitHub Desktop** allows you to interact with git locally using a Graphical User Interface (and facilitates collaboration with others - lesson 2 & 3)

Extra considerations

- There are best practices when setting up a new repo (e.g. *README* file, *LICENSE* file, *.gitignore* file) that we did not cover in detail - pick that up in lesson 2

- GitHub Desktop is one of the most accessible git clients for getting started but …
  - It has limitations (staging less explicit, lacks some advance features, less interchangeable)
  - Use the best tool for your workflow (e.g. command line, software extensions)

- Notable limitations of git
  - Binary files (some spreadsheets, images) - particularly if large
  - What about datasets?

Resources to help

.gitattributes - tell git how to handle certain files

IDEs to manage databases (e.g. **Eclipse)** can have **version control plug-ins**

Git Large File Storage - https://git-lfs.com
Data Version Control - https://dvc.org
TortoiseSVN - https://tortoisesvn.net

Prep for lesson 2

Broad curriculum overview

Lesson 2 & 3
GitHub & Collaboration

Working locally and remotely
Working with yourself & others
Using GitHub Desktop and
https://github.com

Lesson 1
Version Control (with git)

Working locally
Working with yourself
Using GitHub Desktop

Potential confusion

We used a tool called GitHub Desktop. This allows us to build towards Lesson 2 & 3, however in Lesson 1 we only used GitHub desktop locally (i.e. we did not connect to https://github.com)

Further resources

git cheat sheets and **training manuals:**
https://training.github.com

GitHub Desktop documentation:
https://docs.github.com/en/desktop

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

AIM-RSF GitHub repository lists lots of resources and tutorials:

https://github.com/aim-rsf/training/tree/main/version-control

Acknowledgments

Acknowledgments

Eirini Zormpa

- Workshop organization
- Assistance with lesson planning
- Previous slides: https://aim-rsf.github.io/training/github-intro/github-intro-slides

Content / inspiration taken from

- https://docs.github.com/en/desktop
- https://git-scm.com
- https://annakrystalli.me/rrresearch/06_git.html
- https://github.com/git-guides

Icons

flaticon

www.flaticon.com/free-icons


THANK YOU!!
Scribbria