# Day 3: Sharing plans, data and code

By David Souto

# Worksheet 8

Introduction to GitHub (20’)

Make sure GitHub Desktop is installed and you have a GitHub account.

The example we are going to use comes from a neuroimaging data. The community has developed conventions for folder structure based on common sense to facilitate sharing.

Here is the basic BIDS structure, which you can use as inspiration for your own datasets, in the absence of other community-specific conventions.

See also: <https://nceas.github.io/sasap-training/materials/reproducible_research_in_r_fairbanks/literate-analysis-with-rmarkdown.html#organizing-a-reproducible-research-folder>

Occasion to introduce standards for data.

Through example dataset. Each table could be assigned a task.

## Version control with GitHub

GitHub is an incredibly popular website to share code. It uses Git for version control, a command line tool. You can use Git through the command line and through a very handy app called GitHub Desktop.

A common assumption is that version control is only efficient / useful with larger projects. We will show this is not the case. Another misconception is that it is too much faff / complication for a simple project. This will also appear misguided today. We’ll start with level 1, which his sufficient to provide you with: the ability to share code, back up code, seamlessly synchronize code between computers, have a record of changes and return to working versions of the code.

### Setup

If you didn’t already.

### Level 1: It’s just me

#### Turn your workshop folder into a GitHub repository. …

We can integrate with RStudio by …

#### Note the information you can gather from launching GitHub Desktop. The Changes Tab, The History tab. Change something to one of the files in the repository. Observe what happens in the “Changes Tab”

#### Committing and commenting your changes. On the bottom left corner provide a summary of the change (necessary) and description (optional). You can now “Commit to main”. Meaning you can record changes in the “main” version of the code. This has to be specified because we can make parallel versions of the code, called branches (see Level XX).

#### Push / Pull. Observe now the new options you have on the right hand side. You have a suggestion to “push” changes to GitHub, you can check by “View on GitHub”.

Now change something on GitHub by pressing the edit button for the README.Rmd file. See the option now to “Pull” changes from GitHub to your local computer.

#### Let’s try to break things A red smiley with horns and teeth AI-generated content may be incorrect.

Go to GitHub and delete a line in the README.Rmd file. Then go to the local folder (View Explorer) and delete *a different* line in the local version of the file. Go to GitHub Desktop and do a pull. Observe the issue.

Go to View in RStudio to resolve the conflict. What are the options?

#### There is another pickle. We said it would be nice to have a structure with everything in one folder. But we don’t want to share the raw data via GitHub. We usually don’t have enough space among a myriad of other reasons. We also have code we are working on we don’t want to share.

Create a file called **.gitignore**, open this file with the Notepad or RStudio. Add folders to ignore, depending on your file structure, this way:

**data/**

**for\_your\_eyes\_only/**

#### Do a commit. Then View on GitHub. See whether the files have been synchronized or not.

Go to View in RStudio to resolve the conflict. What are the options?

8.2.6. Lastly, ideally you want short names for your repositories so you don’t have long paths. But things can get messy in your GDesktop if you have many repositories. Example: XXX

You can then use “aliases”. They are local to your GDesktop and can disambiguate names. Aliasies are italisized.

#### Well done. You can now use Git (through the GitHub interface) pretty much to the same level of sophistication as his creator Linus Torvalds (see XXX) without a line of code.

Every time you have a new project you create a new repository. This does not stop you from backing up your data if you are afraid of breaking things. The next levels might give you more confidence on that front.

### Level 2: Reuse code by cloning and forking things

We are going to clone a repository or repo.

We are going to clone

### Level 3: N = 2

What if we work with others?

#### Add your neighbour to your project, decide on what they can and cannot do

#### Create a branch called playground. Observe what happens on GDesktop and on GHub after you commit.

#### Create a merge request to merge the branch with main

Working on the code at the same time.

### Level 4: N >3

Let’s say your project has many contributors across the world. You want a way to manage and veto their contributions.

Let’s create a project where everyone is a contributor. Ask people to review code. Suggest a merger …

Advanced GitHub structure:

Try using the command line from GDesktop: