AN INTRODUCTION TO DATALAD

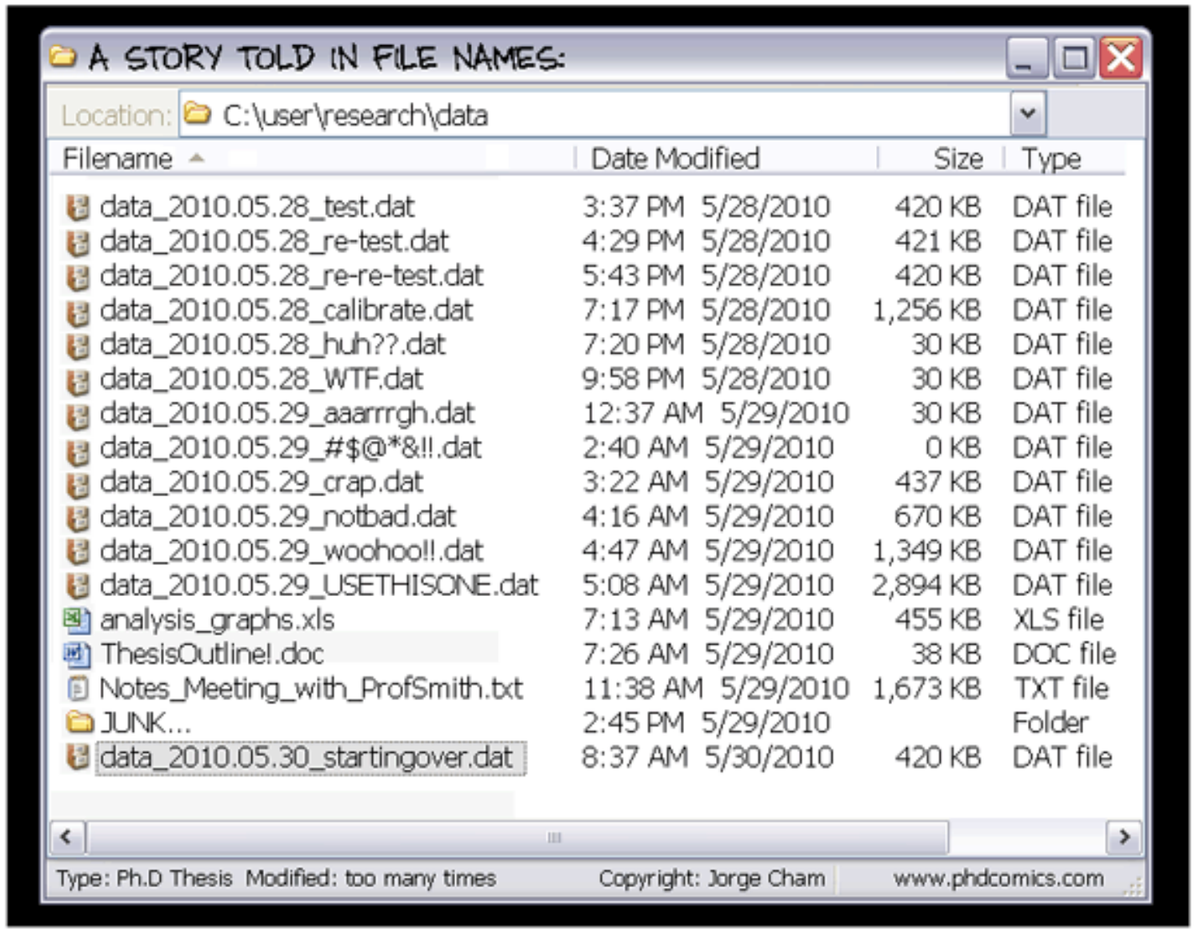
Adina Wagner
mas.to/adswa

Psychoinformatics lab,
Institute of Neuroscience and Medicine, Brain & Behavior (INM-7)
Research Center Jülich

Slides: DOI 10.5281/zenodo.10369776 (Scan the QR code)
Online Slides: files.inm7.de/adina/talks/html/osoh.html

LECTURE + LIVE CODING

- Live-demonstration of DataLad examples and workflows
- Materials incl. copy-paste code snippets and hands-on exercises at handbook.datalad.org/r.html?osoh


A STORY TOLD IN FILE NAMES:
Location: C:\user\research\data
Filename
Date Modified
Size
Type
data_2010.05.28_test.dat 3:37 PM 5/28/2010 420 KB DAT file
data_2010.05.28_re-test.dat 4:29 PM 5/28/2010 421 KB DAT file
data_2010.05.28_re-re-test.dat 5:43 PM 5/28/2010 420 KB DAT file
data_2010.05.28_calibrate.dat 7:17 PM 5/28/2010 1,256 KB DAT file
data_2010.05.28_huh??.dat 7:20 PM 5/28/2010 30 KB DAT file
data_2010.05.28_WTF.dat 9:58 PM 5/28/2010 30 KB DAT file
data_2010.05.29_aaarrrgh.dat 12:37 AM 5/29/2010 30 KB DAT file
data_2010.05.29_#$@*&!!.dat 2:40 AM 5/29/2010 0 KB DAT file
data_2010.05.29_crap.dat 3:22 AM 5/29/2010 437 KB DAT file
data_2010.05.29_notbad.dat 4:16 AM 5/29/2010 670 KB DAT file
data_2010.05.29_woohoo!!.dat 4:47 AM 5/29/2010 1,349 KB DAT file
data_2010.05.29_USETHISONE.dat 5:08 AM 5/29/2010 2,894 KB DAT file
analysis_graphs.xls 7:13 AM 5/29/2010 455 KB XLS file
ThesisOutline!.doc 7:26 AM 5/29/2010 38 KB DOC file
Notes_Meeting_with_ProfSmith.txt 11:38 AM 5/29/2010 1,673 KB TXT file
JUNK... 2:45 PM 5/29/2010 Folder
data_2010.05.30_startingover.dat 8:37 AM 5/30/2010 420 KB DAT file
Type: Ph.D Thesis Modified: too many times
Copyright: Jorge Cham
www.phdcomics.com

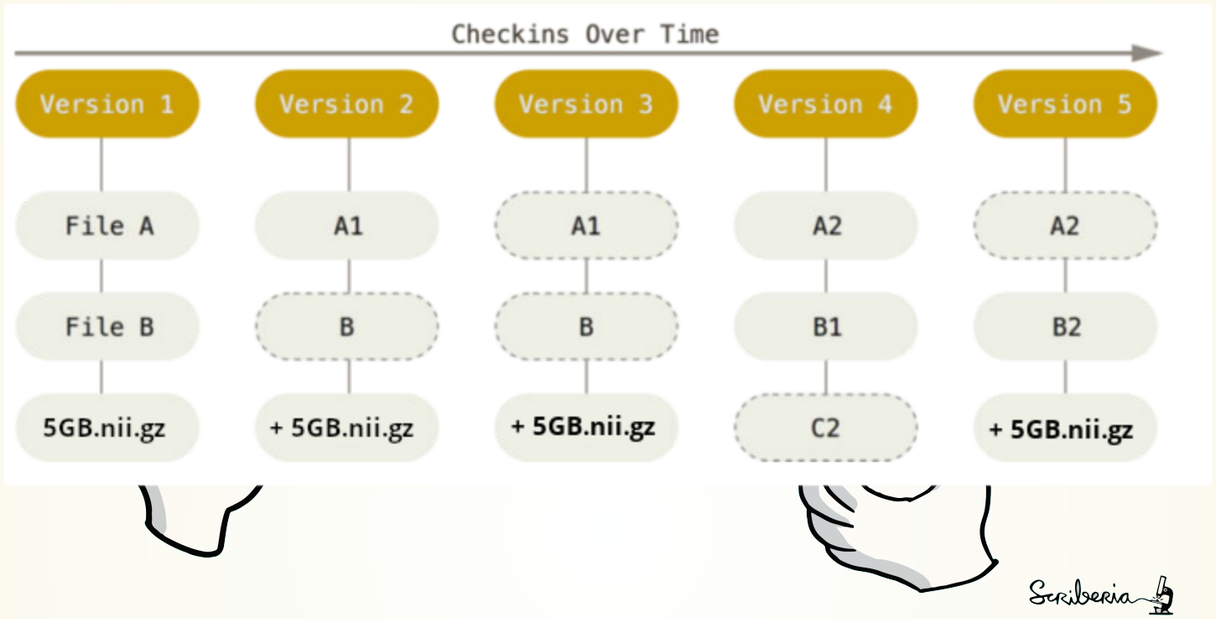
THE SAME, BUT FOR DATA:

```
--- /data/BnB1/DATA/download_data/eNKI ----------------
                        /..
    5.2 TiB [##########] /eNKI_unzipped
    3.3 TiB [######    ] /eNKI_redownload
    3.2 TiB [######    ] /eNKI_BIDSdownload
  724.2 GiB [#         ] /eNKI_20180806
  218.8 GiB [          ] /eNKI_aus_Raw_Data
```

(Yes, 13 TB of data. Yes, real-life example)

HELP! GIT TO THE RESCUE?

Sadly, Git does not handle large files well.



And repository hosting services refuse to handle large files:

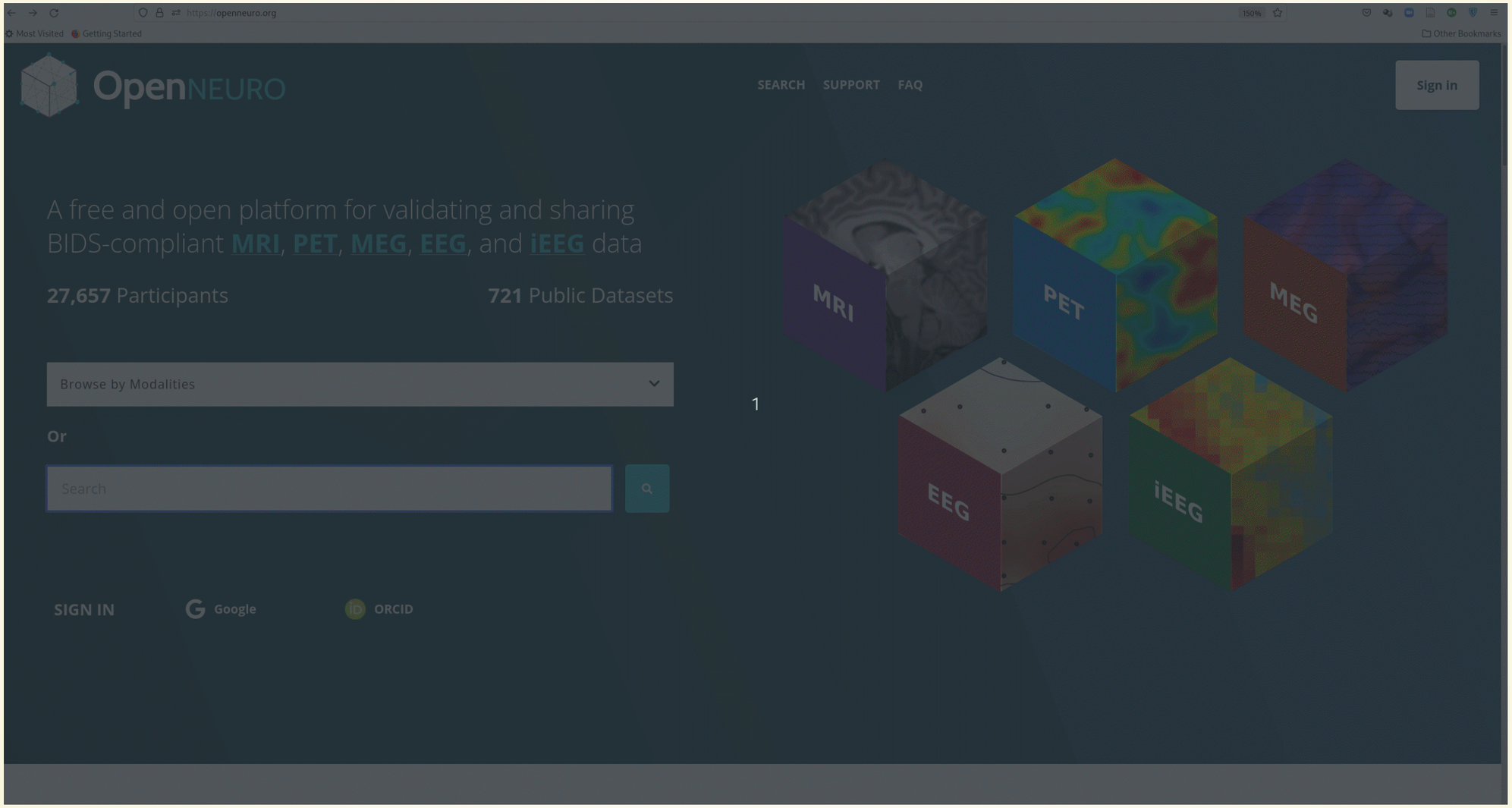
```
adina@muninn in /tmp/myresearch on git:master
❯ git push gh-adswa master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 497.87 KiB | 161.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: error: Trace: 64a78dd41ece8e5493fe33f97397a7a90ef9c91260ba32786970dbdcf5c4e0dd
remote: error: See http://git.io/iEPt8g for more information.
remote: error: File output.dat is 500.00 MB; this exceeds GitHub's file size limit of 100.00 MB
remote: error: GH001: Large files detected. You may want to try Git Large File Storage - https://git-lfs.github.com.
To github.com:adswa/myresearch.git
 ! [remote rejected] master -> master (pre-receive hook declined)
error: failed to push some refs to 'github.com:adswa/myresearch.git'
```

- A command-line tool, available for all major operating systems (Linux, macOS/OSX, Windows), MIT-licensed
- Build on top of Git and Git-annex
- **Allows...**
  ... version-controlling arbitrarily large content
  version control data and software alongside to code!
  ... transport mechanisms for sharing and obtaining data
  consume and collaborate on data (analyses) like software
  ... (computationally) reproducible data analysis
  Track and share provenance of all digital objects
  ... and much more
- Completely domain-agnostic

EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

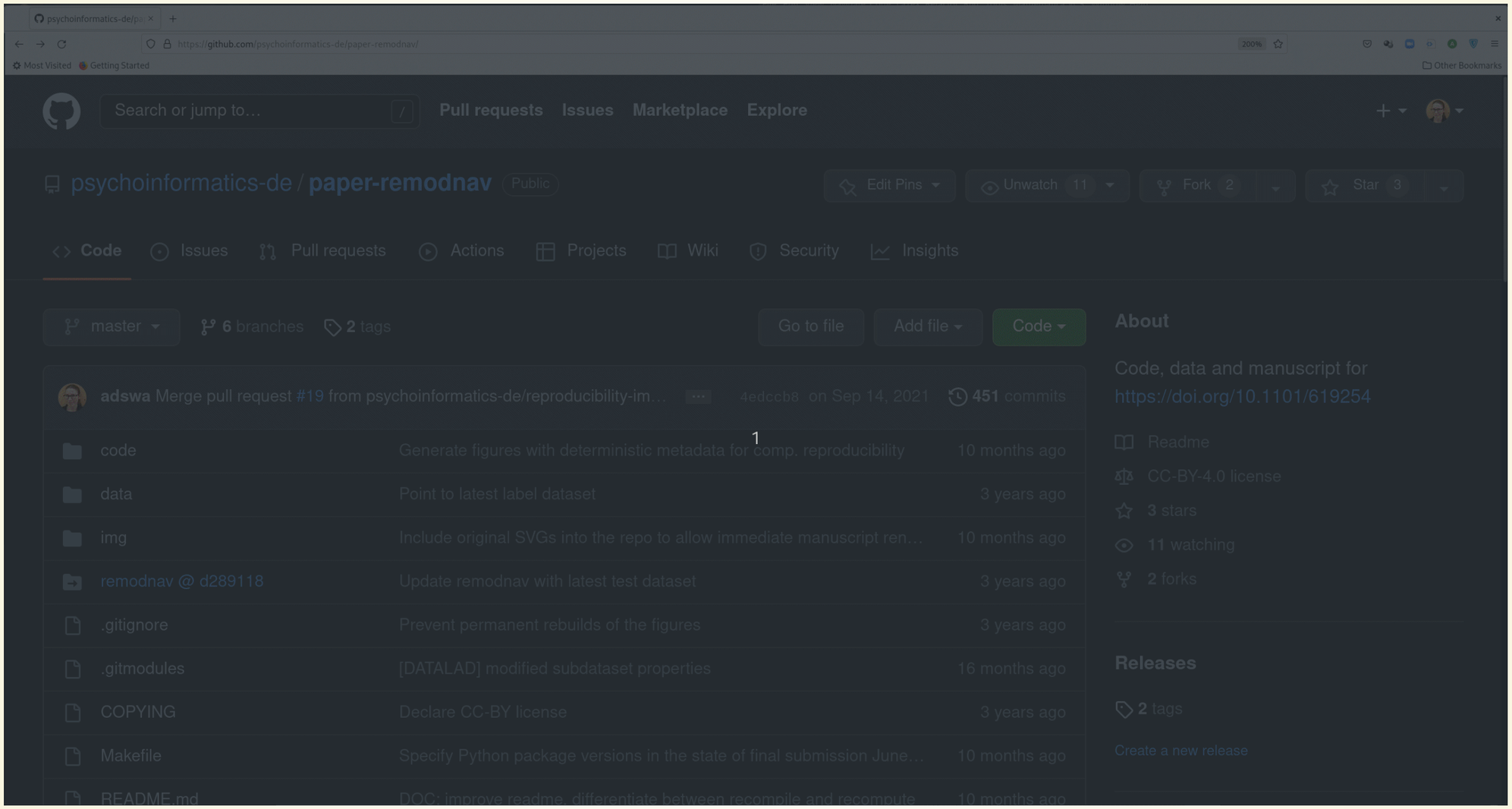
EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

- Behind-the-scenes **infrastructure component for data transport and versioning** (e.g., used by OpenNeuro, brainlife.io , the Canadian Open Neuroscience Platform (CONP), CBRAIN)



EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

- **Creating and sharing reproducible, open science**: Sharing data, software, code, and provenance



EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

USING DATALAD

- DataLad can be used from the command line

```
datalad create mydataset
```

- ... or with its Python API

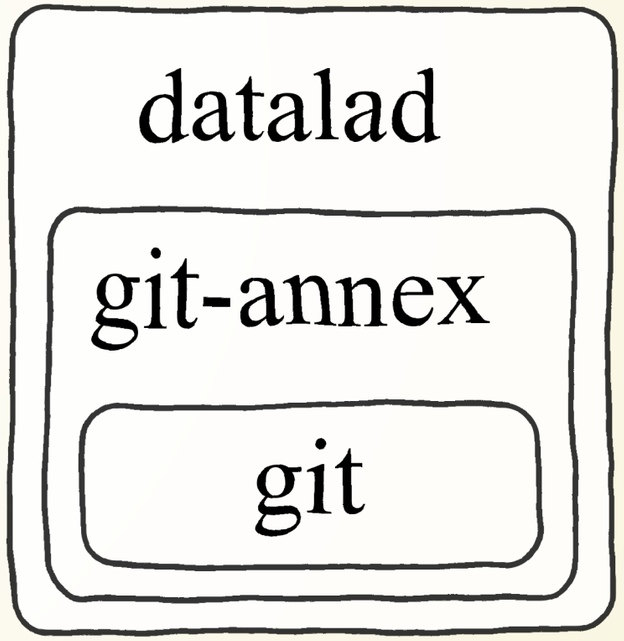
```
import datalad.api as dl
dl.create(path="mydataset")
```

- ... or with a slimmed-down graphical user interface
- ... and other programming languages can use it via system call

```
# in R
> system("datalad create mydataset")
```

DATALAD DATASETS

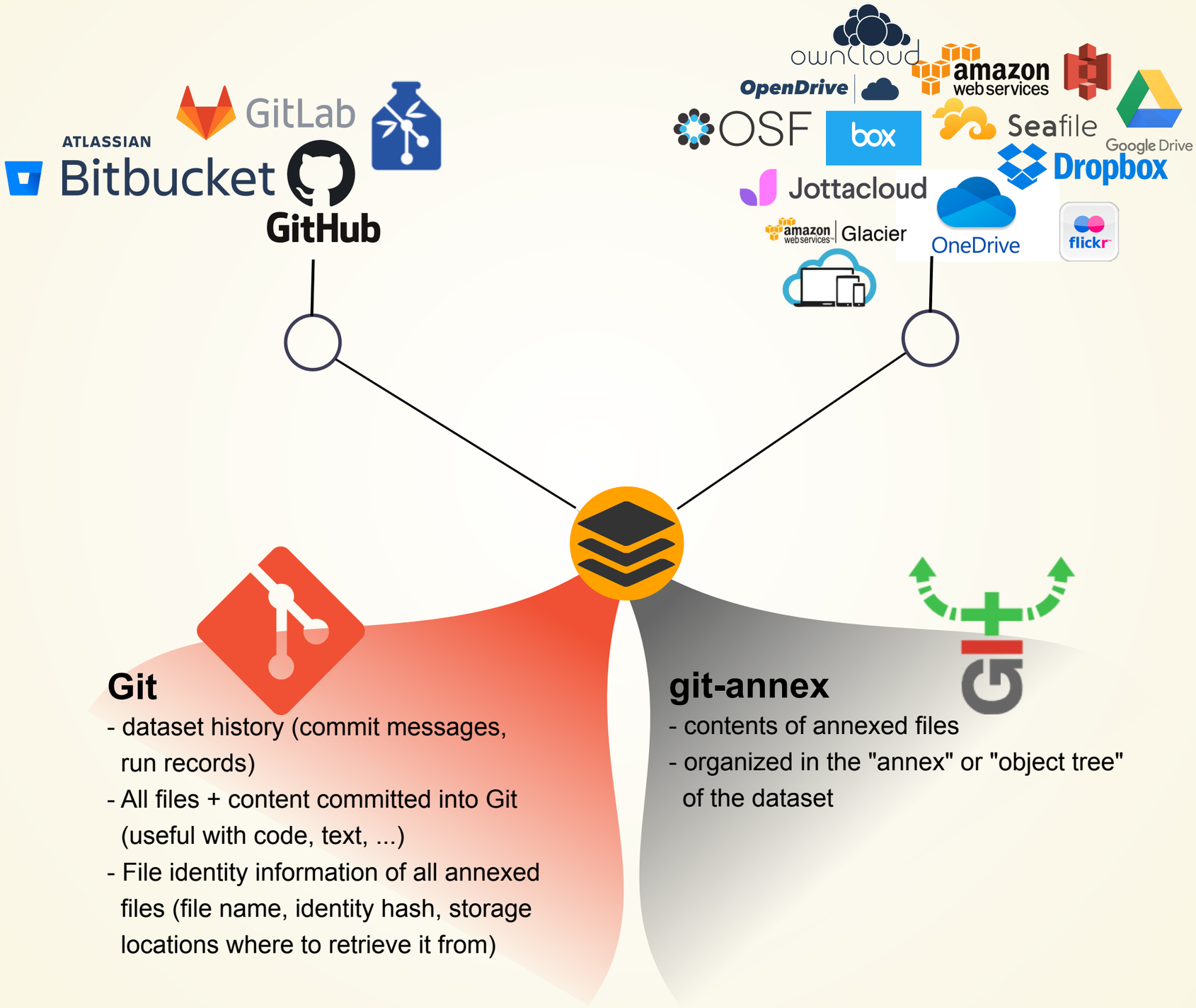
- DataLad's core data structure
  - Dataset = A directory managed by DataLad
  - Any directory of your computer can be managed by DataLad.



A DataLad dataset is a joined Git + git-annex repository

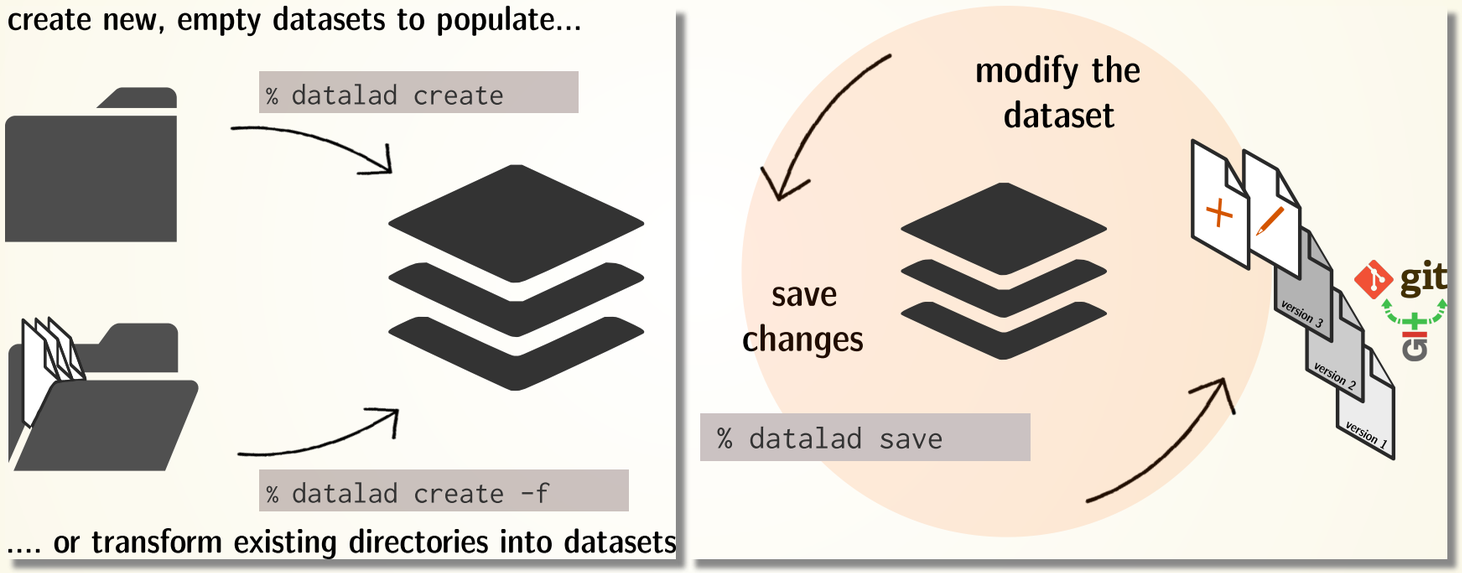
DISTRIBUTED VERSION CONTROL FOR DATA

DISTRIBUTED VERSION CONTROL FOR DATA



VERSION CONTROL

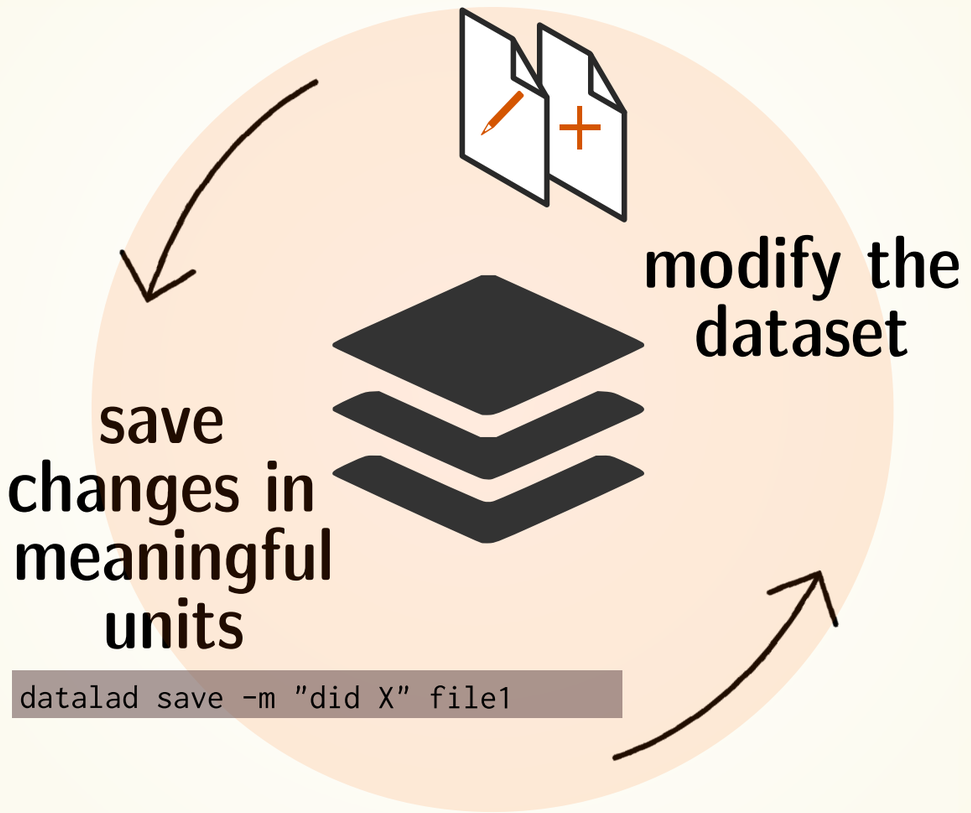
- DataLad knows two things: Datasets and files



- Every file you put into a in a dataset can be easily version-controlled, regardless of size, with the same command.

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!



Advice:

- Save meaningful units of change
- Attach helpful commit messages

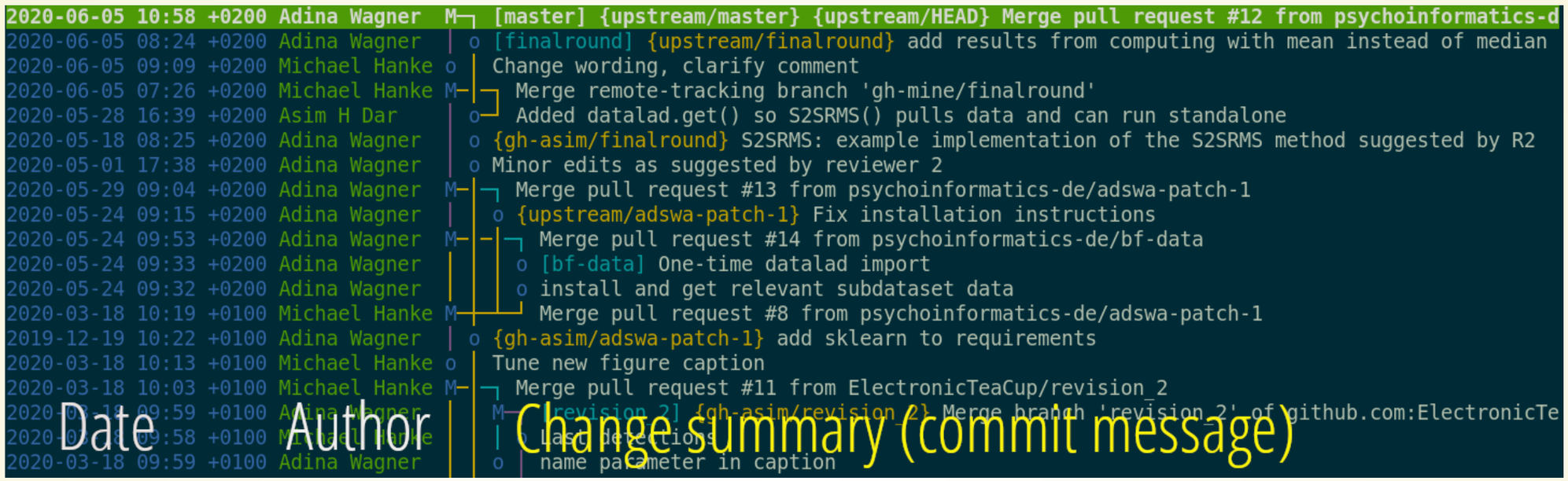
THIS MEANS: YOU CAN ALSO VERSION CONTROL DATA!

```
$ datalad save \
  -m "Adding raw data from neuroimaging study 1" \
  sub-*
  add(ok): sub-1/anat/T1w.json (file)
  add(ok): sub-1/anat/T1w.nii.gz (file)
  add(ok): sub-1/anat/T2w.json (file)
  add(ok): sub-1/anat/T2w.nii.gz (file)
  add(ok): sub-1/func/sub-1-run-1_bold.json (file)
  add(ok): sub-1/func/sub-1-run-1_bold.nii.gz (file)
  add(ok): sub-10/anat/T1w.json (file)
  add(ok): sub-10/anat/T1w.nii.gz (file)
  add(ok): sub-10/anat/T2w.json (file)
  add(ok): sub-10/anat/T2w.nii.gz (file)
[110 similar messages have been suppressed]
  save(ok): . (dataset)
  action summary:
add (ok: 120)
save (ok: 1)
```

copy

VERSION CONTROL

- Your dataset can be a complete research log, capturing everything that was done, when, by whom, and how



- Interact with the history:
- reset your dataset (or subset of it) to a previous state,
- throw out changes or bring them back,
- find out what was done when, how, why, and by whom
- Identify precise versions: Use data in the most recent version, or the one from 2018, or...
- ...

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

Configurations (**-c yoda**, **-c text2git**) are useful (details soon).

A dataset has a history to track files and their modifications.

Explore it with Git (**git log**) or external tools (e.g., **tig**).

`datalad save` records the dataset or file state to the history.

Concise **commit messages** should summarize the change for future you and others.

`datalad status` reports the current state of the dataset.

A clean dataset status (no modifications, not untracked files) is good practice.

CONSUMING DATASETS

- A dataset can be created from scratch/existing directories:

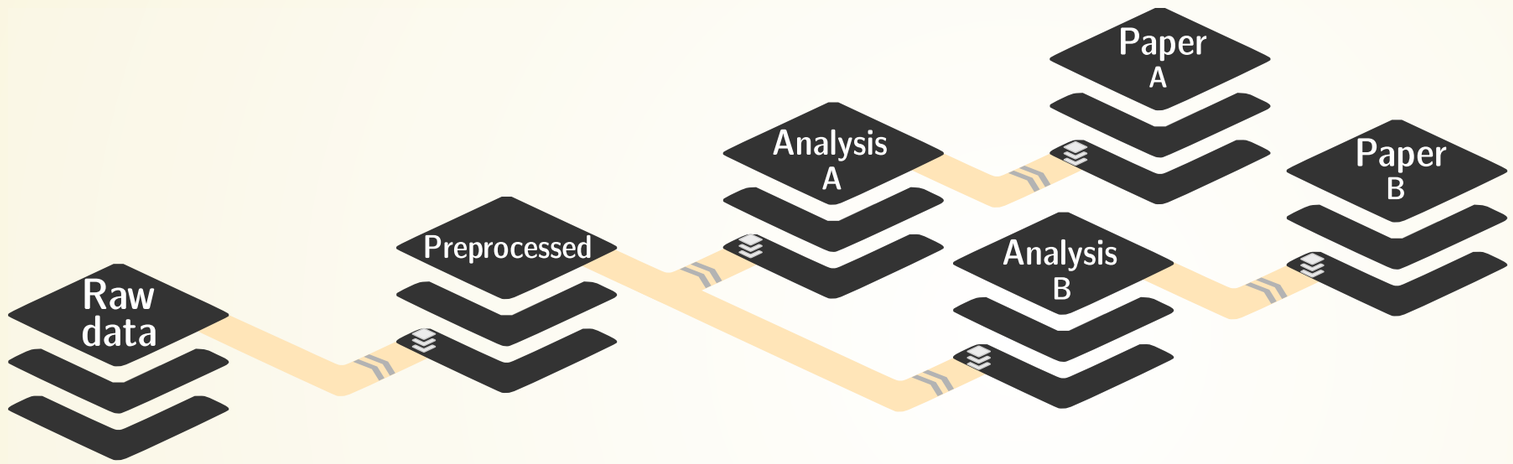
```
$ datalad create mydataset
[INFO] Creating a new annex repo at /home/adina/mydataset
create(ok): /home/adina/mydataset (dataset)
```

- but datasets can also be installed from paths or from URLs:

```
$ datalad clone https://github.com/datalad-datasets/human-connectome-project-opena
install(ok): /tmp/HCP (dataset)
```

DATASET NESTING

- Typically, Git repositories are cumbersome to link to eachother. DataLad provides seamless nesting mechanisms:



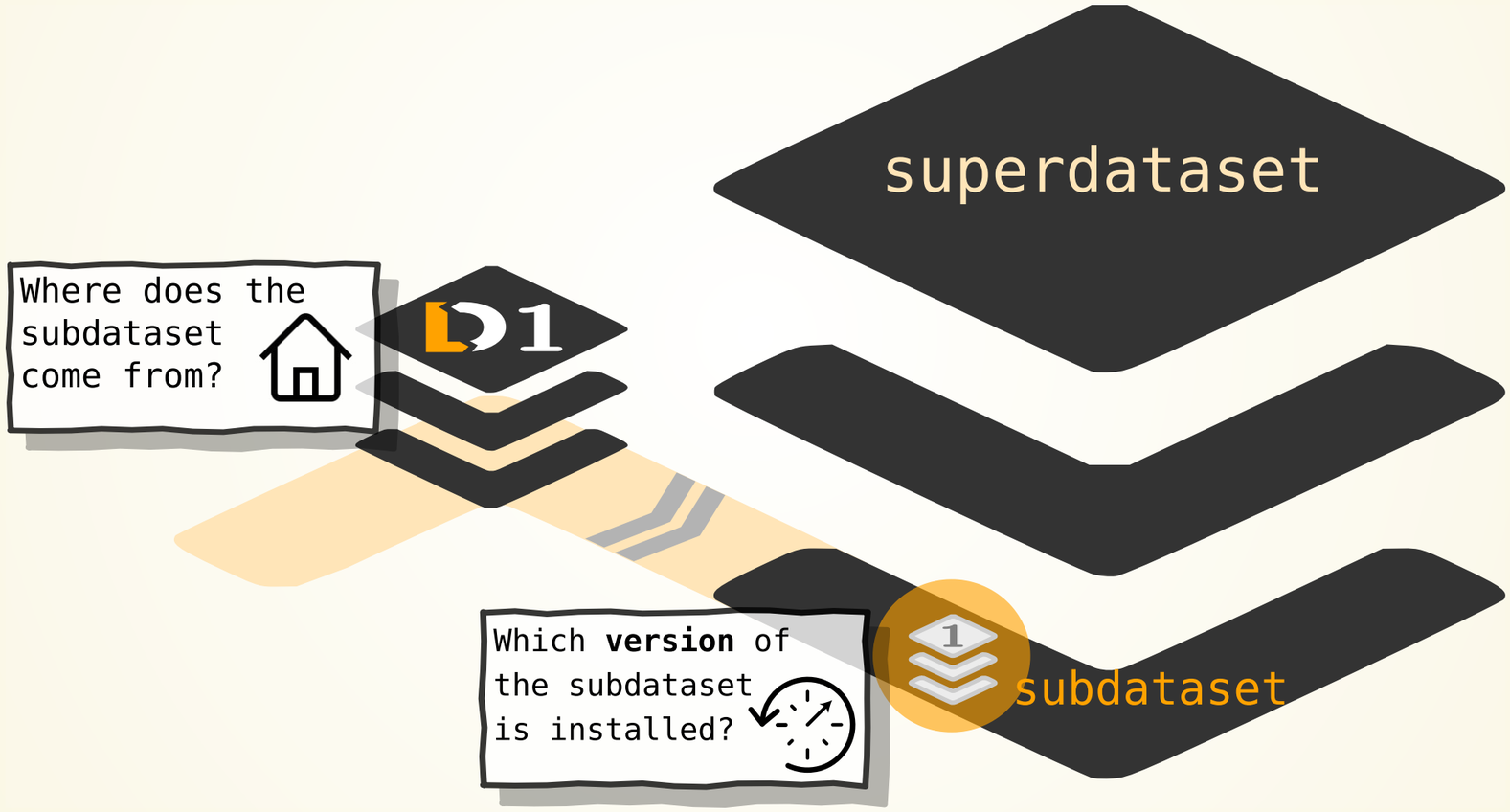
Nest modular datasets to create a linked hierarchy of datasets, and enable recursive operations throughout the hierarchy

- Modularizes research components for transparency, reuse, and access management
- Overcomes scaling issues with large amounts of files

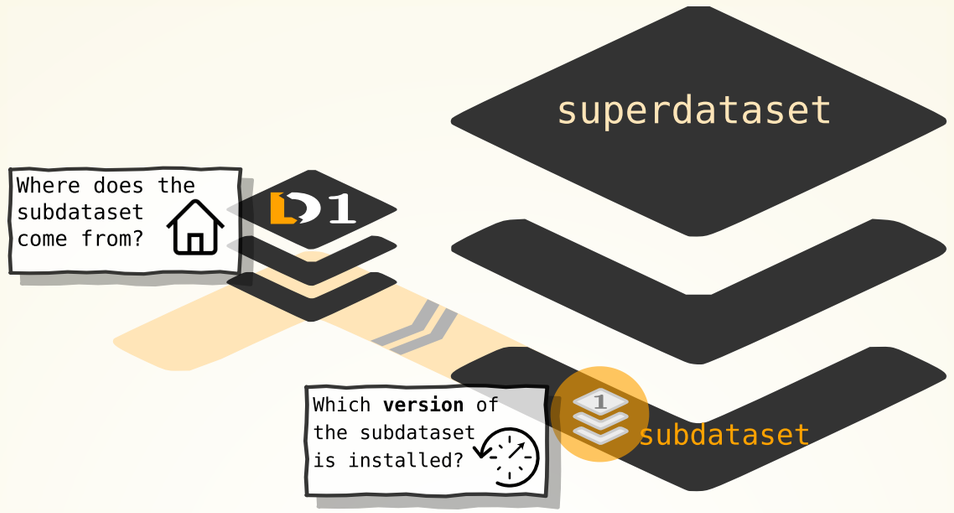
```
adina@bulk1 in /ds/hcp/super on git:master❱ datalad status --annex -r
  15530572 annex'd files (77.9 TB recorded total size)
  nothing to save, working tree clean
```

(github.com/datalad-datasets/human-connectome-project-openaccess)

DATASET NESTING



DATALAD: DATASET LINKAGE



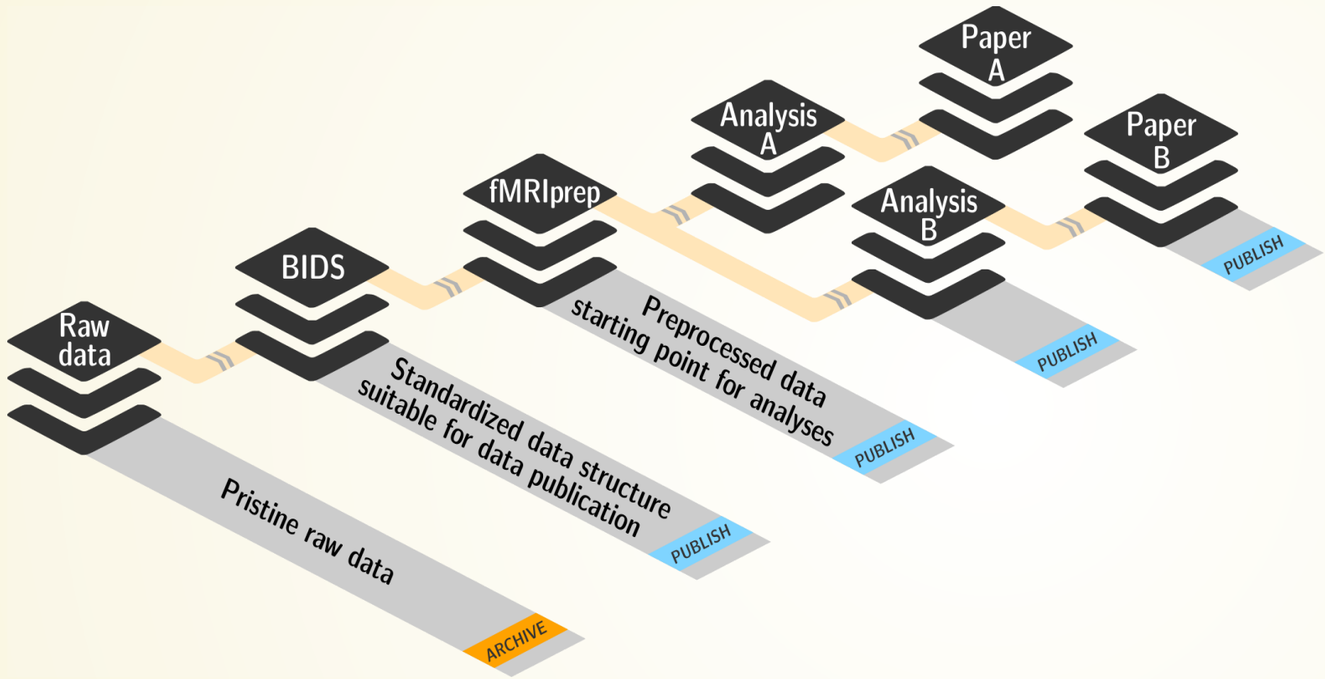

```
$ datalad clone --dataset . http://example.com/ds inputs/rawdata
```
copy

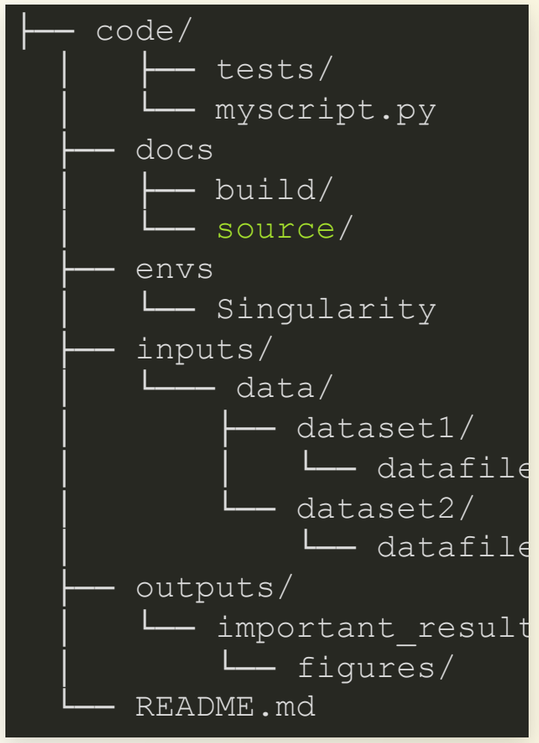
```
$ git diff HEAD~1
diff --git a/.gitmodules b/.gitmodules
new file mode 100644
index 0000000..c3370ba
--- /dev/null
+++ b/.gitmodules
@@ -0,0 +1,3 @@
+[submodule "inputs/rawdata"]
+       path = inputs/rawdata
+       url = http://example.com/importantds
diff --git a/inputs/rawdata b/inputs/rawdata
new file mode 160000
index 0000000..fabf852
--- /dev/null
+++ b/inputs/rawdata
@@ -0,0 +1 @@
+Subproject commit fabf8521130a13986bd6493cb33a70e580ce8572
```
copy

BASIC ORGANIZATIONAL PRINCIPLES FOR DATASETS

Keep everything clean and modular

- An analysis is a superdataset, its components are subdatasets, and its structure modular



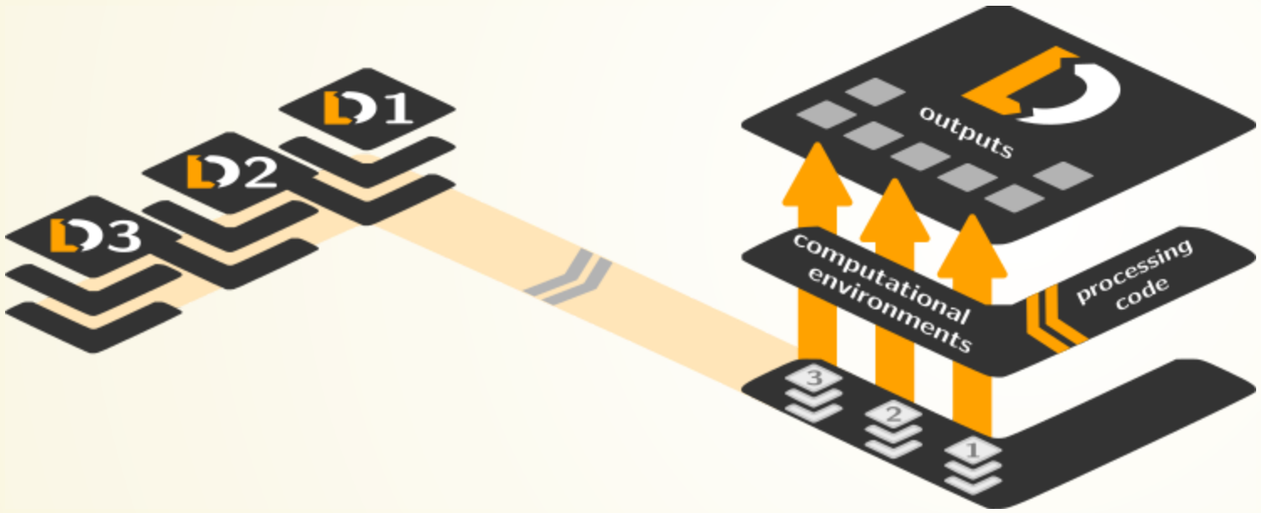


- do not touch/modify raw data: save any results/computations outside of input datasets
- Keep a superdataset self-contained: Scripts reference subdatasets or files with relative paths

BASIC ORGANIZATIONAL PRINCIPLES FOR DATASETS

Record where you got it from, where it is now, and what you do to it

- Link datasets (as subdatasets), record data origin
- Collect and store provenance of all contents of a dataset that you create



Document everything:

- Which script produced which output? From which data? In which software environment? …

Find out more about organizational principles in the YODA principles!

PLENTY OF DATA, BUT LITTLE DISK-USAGE

- Cloned datasets are lean. "Meta data" (file names, availability) are present, but **no file content**:

```
$ datalad clone git@github.com:psychoinformatics-de/studyforrest-data-phase2.git
  install(ok): /tmp/studyforrest-data-phase2 (dataset)
  $ cd studyforrest-data-phase2 && du -sh
  18M .
```

- file's contents can be retrieved on demand:

```
$ datalad get sub-01/ses-movie/func/sub-01_ses-movie_task-movie_run-1_bold.nii.gz          copy
  get(ok): /tmp/studyforrest-data-phase2/sub-01/ses-movie/func/sub-01_ses-movie_task-movie_run-1
```

- Have more access to your computer than you have disk-space:

```
# eNKI dataset (1.5TB, 34k files):                                                        copy
$ du -sh
  1.5G  .
# HCP dataset (80TB, 15 million files)
$ du -sh
  48G .
```

PLENTY OF DATA, BUT LITTLE DISK-USAGE

Drop file content that is not needed:

```
$ datalad drop sub-01/ses-movie/func/sub-01_ses-movie_task-movie_run-1_bold.nii.gz
drop(ok): /tmp/studyforrest-data-phase2/sub-01/ses-movie/func/sub-01_ses-movie_task-movie_run-1_
```

copy

When files are dropped, only "meta data" stays behind, and they can be re-obtained on demand. This allows disk-space aware computations:

Install your input data

➡ *get the data you need*

➡ *compute your results*

➡ *drop input data (and potentially all automatically re-computable results)*

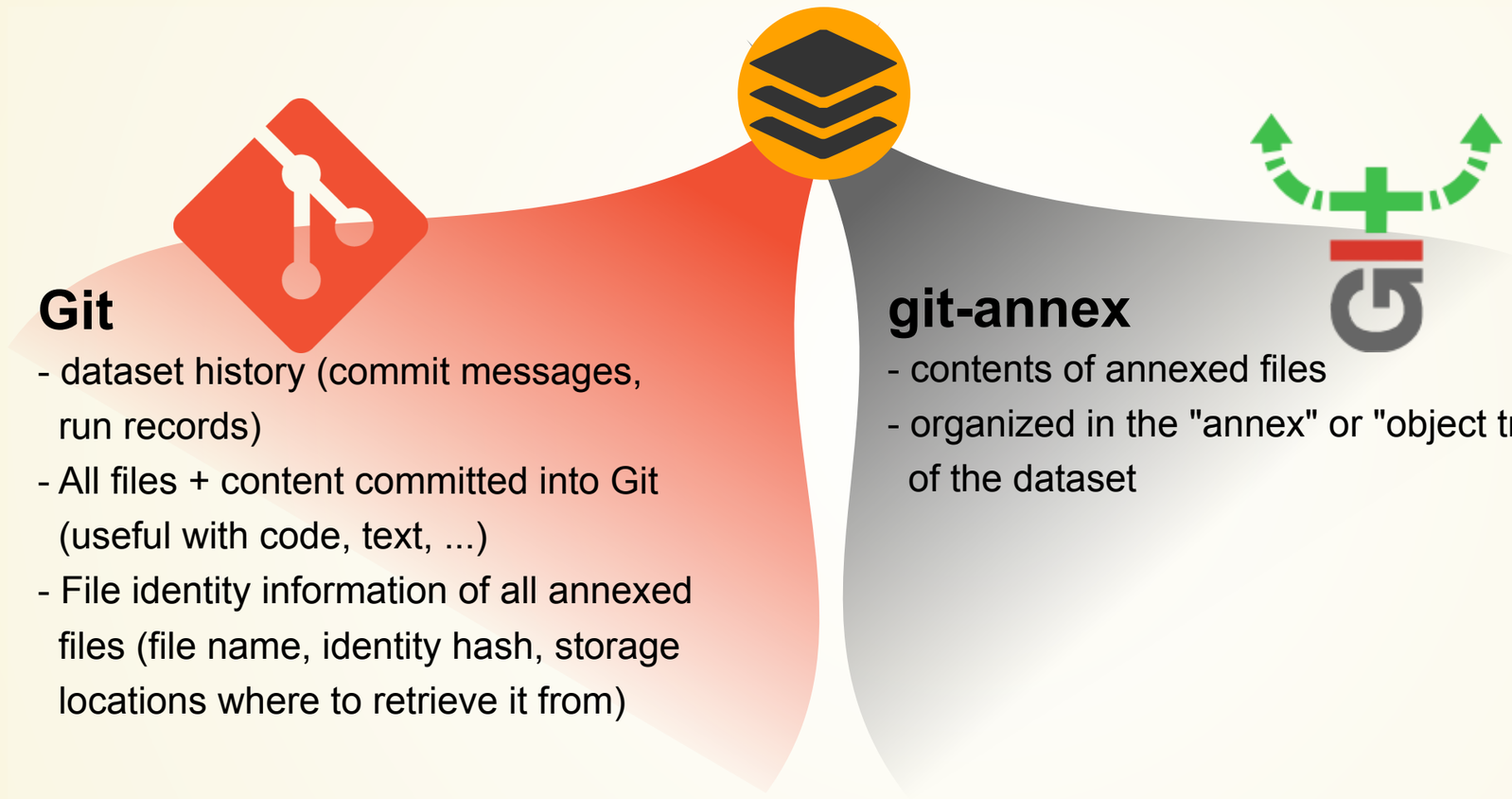
```
dl.get('input/sub-01')

[really complex analysis]

dl.drop('input/sub-01')
```

copy

GIT VERSUS GIT-ANNEX

Data in datasets is either stored in Git or git-annex

By default, everything is annexed, i.e., stored in a dataset annex by git-annex



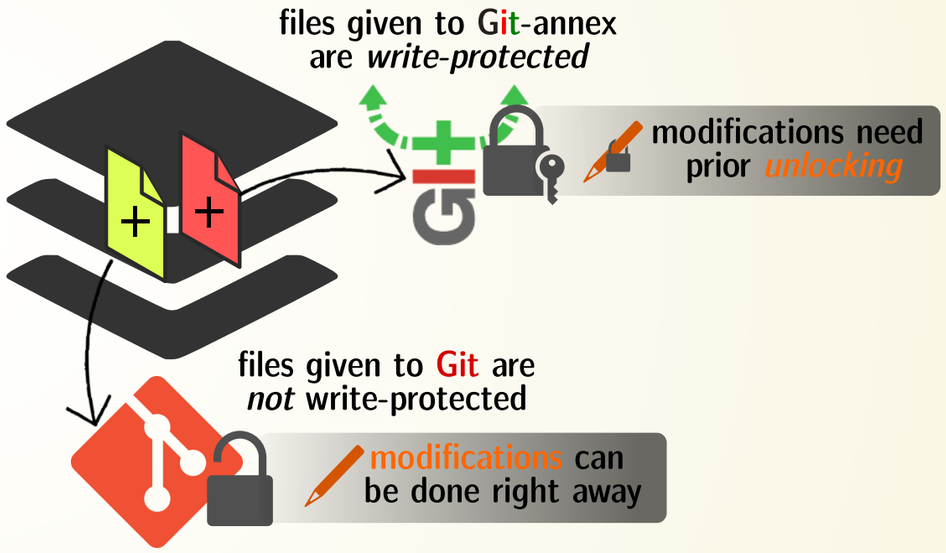
- With annexed data, only content identity (hash) and location information is put into Git, rather than file content. The annex, and transport to and from it is managed with **git-annex**

GIT VERSUS GIT-ANNEX

Git and Git-annex handle files differently: annexed files are stored in an annex. File content is hashed & only content-identity is committed to Git.

- Files stored in Git are **modifiable**, files stored in git-annex are **content-locked**

- Annexed contents are **not available** right after cloning, only content identity and availability information (as they are stored in Git). Everything that is annexed is **retrieved on demand** with `datalad get`.



Read this handbook chapter for details. :)

GIT VERSUS GIT-ANNEX

Users can decide which files are annexed:

- **Pre-made run-procedures**, provided by DataLad (e.g., `text2git`, `yoda`) or created and shared by users (Tutorial)
- Self-made configurations in `.gitattributes` (e.g., based on file type, file/path name, size, ...; rules and examples)
- Per-command basis (e.g., via `datalad save --to-git`)

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad clone` installs a dataset.

It can be installed "on its own": Specify the source (url, path, ...) of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

The **--dataset/-d** option needs a path to the root of the superdataset.

Only small files and metadata about file availability are present locally after an install.

To retrieve actual file content of annexed files, `datalad get` downloads file content on demand.

Datasets preserve their history.

The superdataset records only the version state of the subdataset.

REPRODUCIBLE DATA ANALYSIS

Your past self is the worst collaborator:

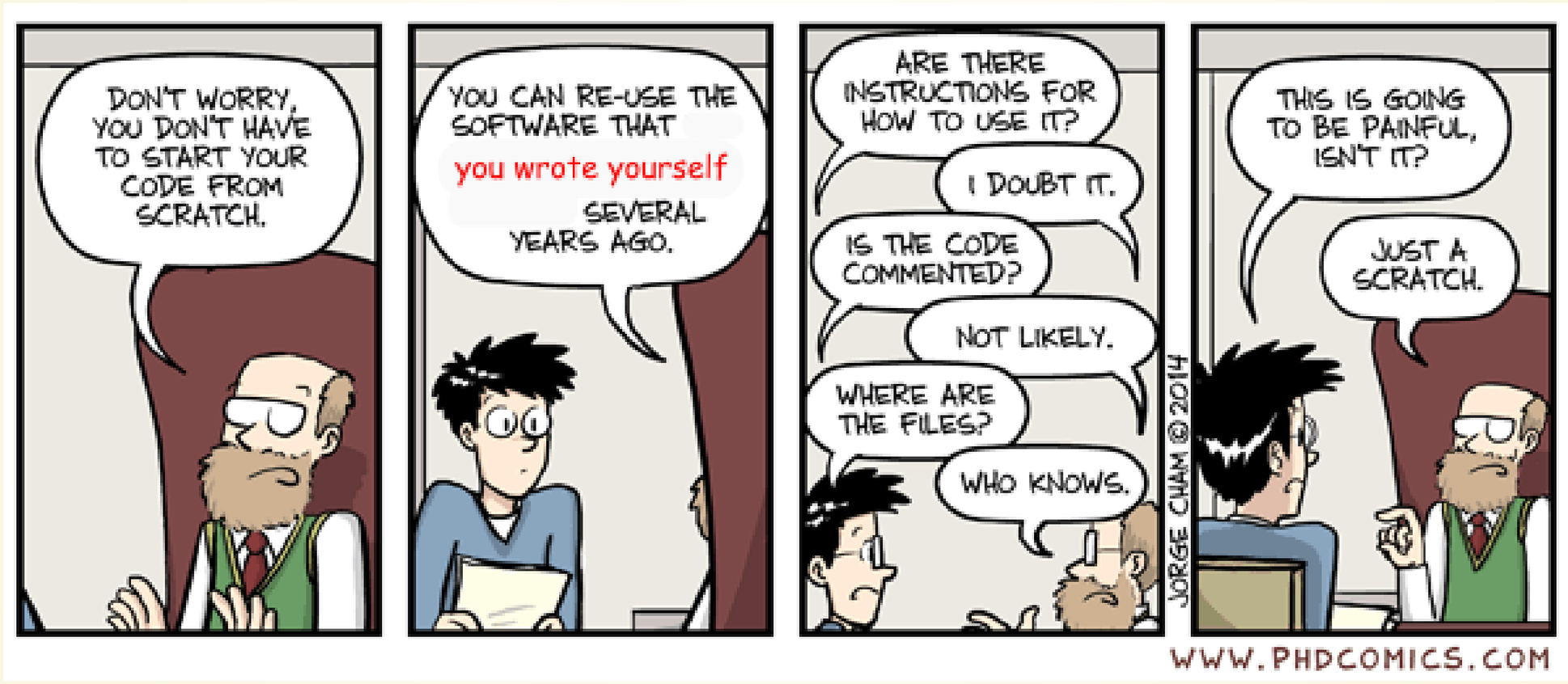
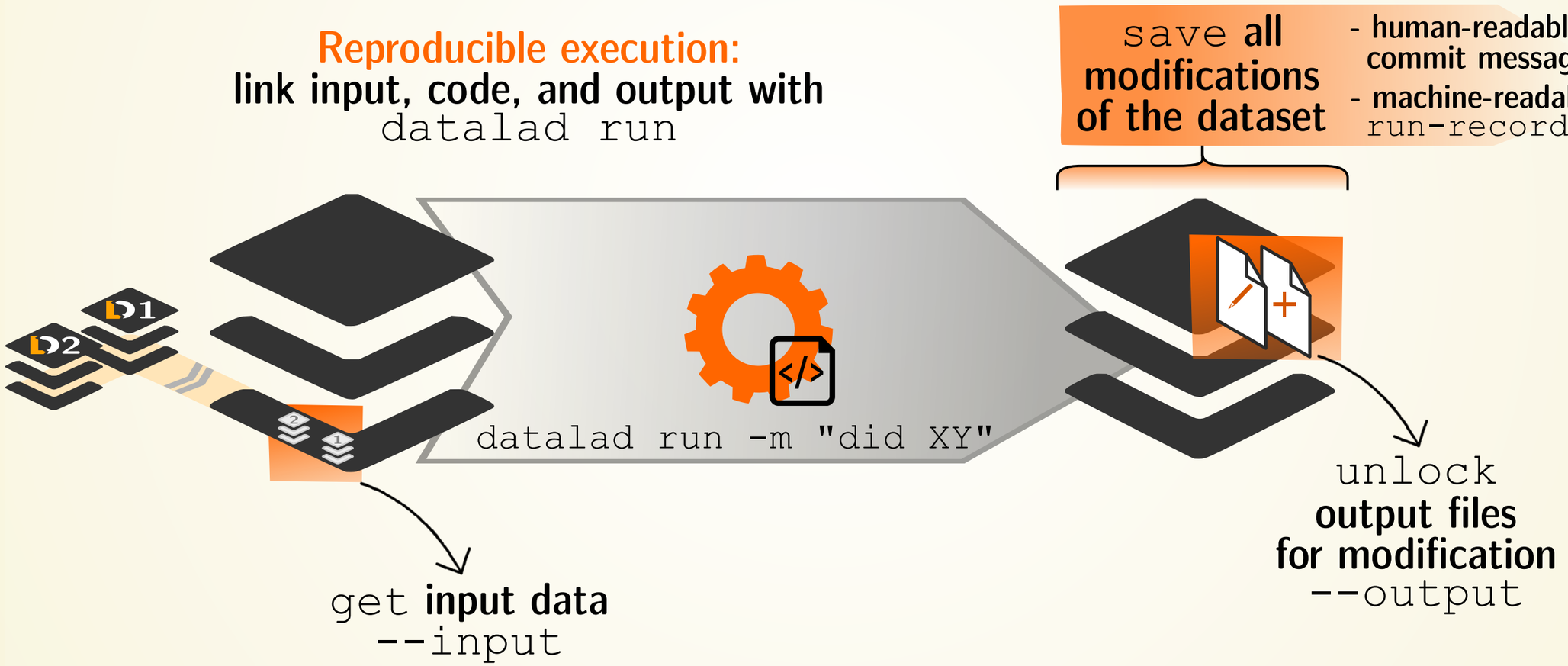


Image credit: Full comic at http://phdcomics.com/comics.php?f=1979

REPRODUCIBLE EXECUTION & PROVENANCE CAPTURE

datalad run

Reproducible execution:
link input, code, and output with
`datalad run`

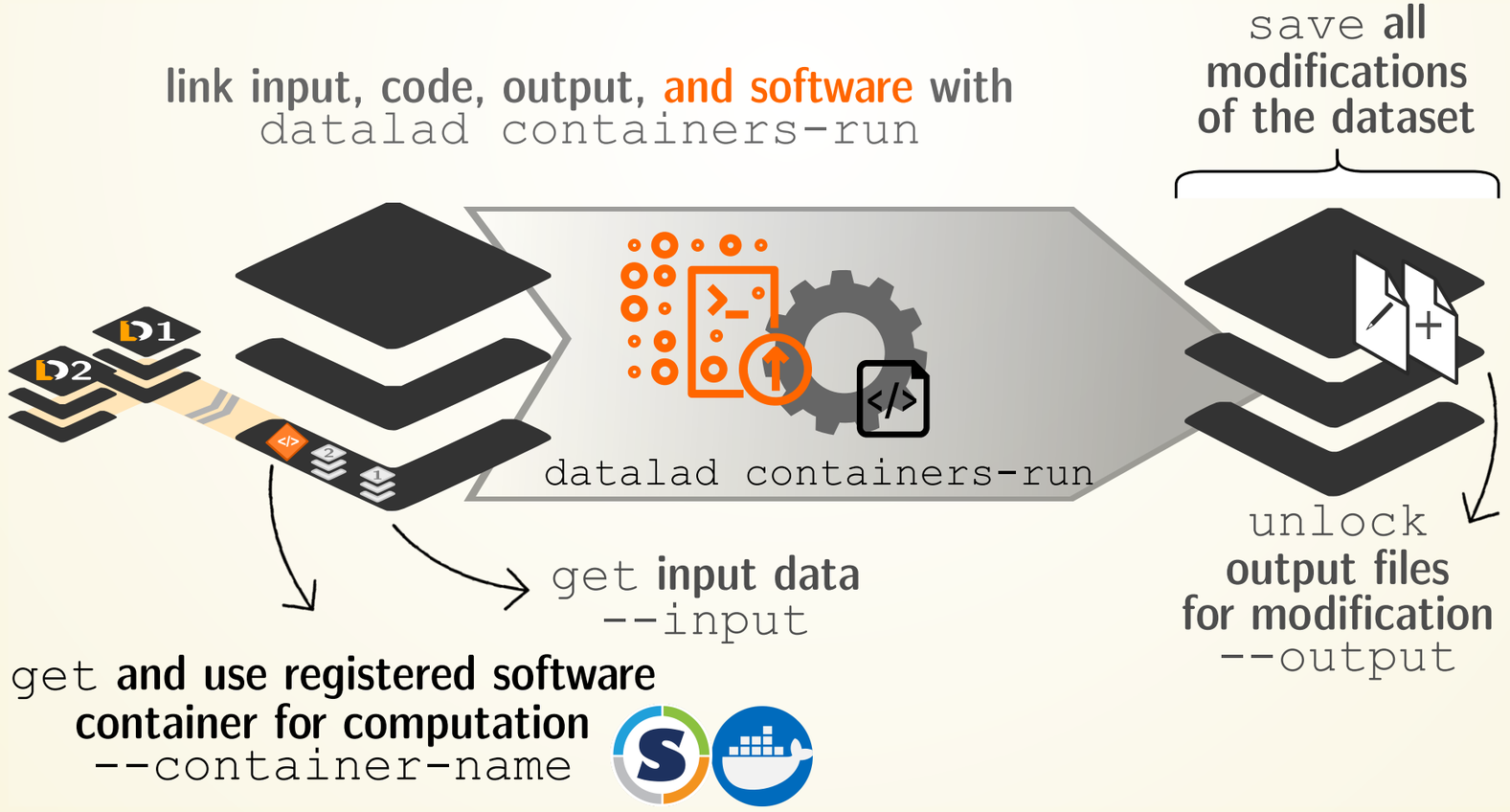


DATALAD RERUN

- `datalad rerun` is helpful to spare others and yourself the short- or long-term memory task, or the forensic skills to figure out how you performed an analysis
- But it is also a digital and machine-readable provenance record
- Important: The better the run command is specified, the better the provenance record
- Note: run and rerun only create an entry in the history if the command execution leads to a change.

COMPUTATIONALLY REPRODUCIBLE EXECUTION & PROVENANCE CAPTURE

- Code may fail (to reproduce) if run with different software
- Datasets can store (and share) software environments (Docker or Singularity containers) and reproducibly execute code inside of the software container, capturing software as additional provenance
  DataLad extension: `datalad-container`



SUMMARY - REPRODUCIBLE EXECUTION

`datalad download-url` obtains web content & records its origin.

It even takes care of saving the change.

`datalad run` records a command and its impact on the dataset.

All dataset modifications are saved - use it in a clean dataset.

Data/directories specified as `--input` are retrieved prior to command execution.

Use one flag per input.

Data/directories specified as `--output` will be unlocked for modifications prior to a rerun of the command.

Its optional to specify, but helpful for recomputations.

`datalad containers-run` can be used to capture the software environment as provenance.

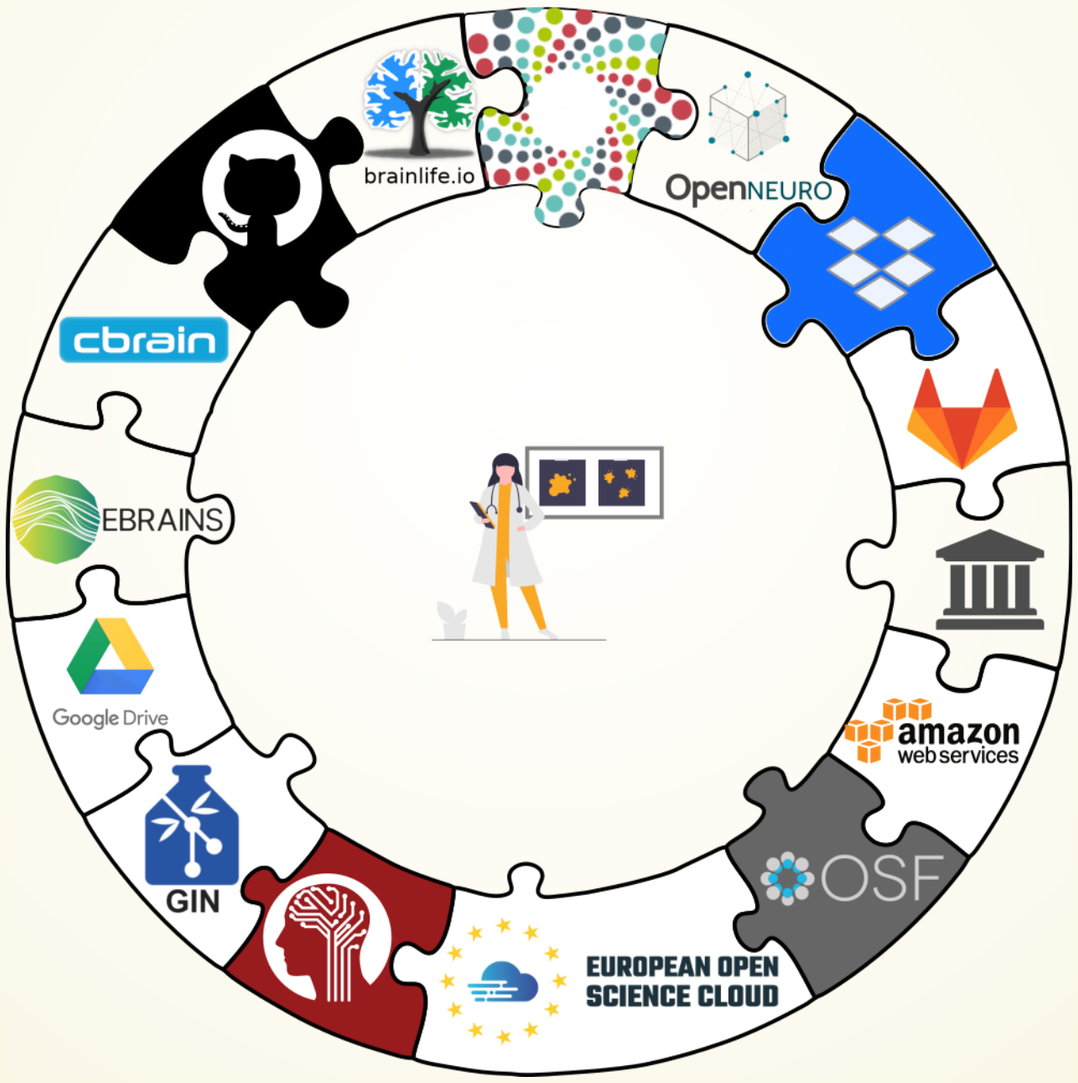
Its ensures computations are ran in the desired software set up. Supports Docker and Singularity containers

`datalad rerun` can automatically re-execute run-records later.

They can be identified with any commit-ish (hash, tag, range, ...)

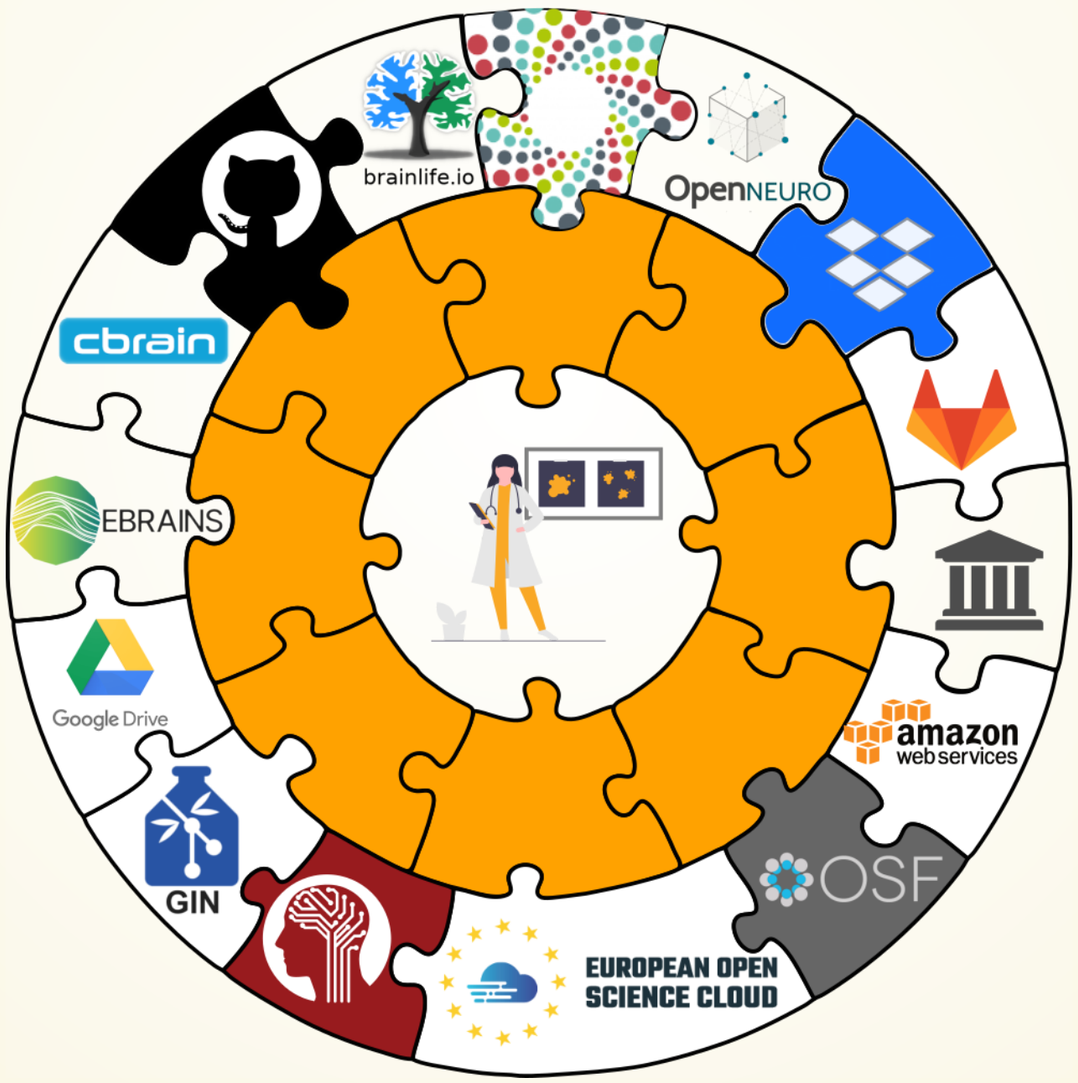
INTEROPERABILITY

- DataLad is built to maximize interoperability and use with hosting and storage technology



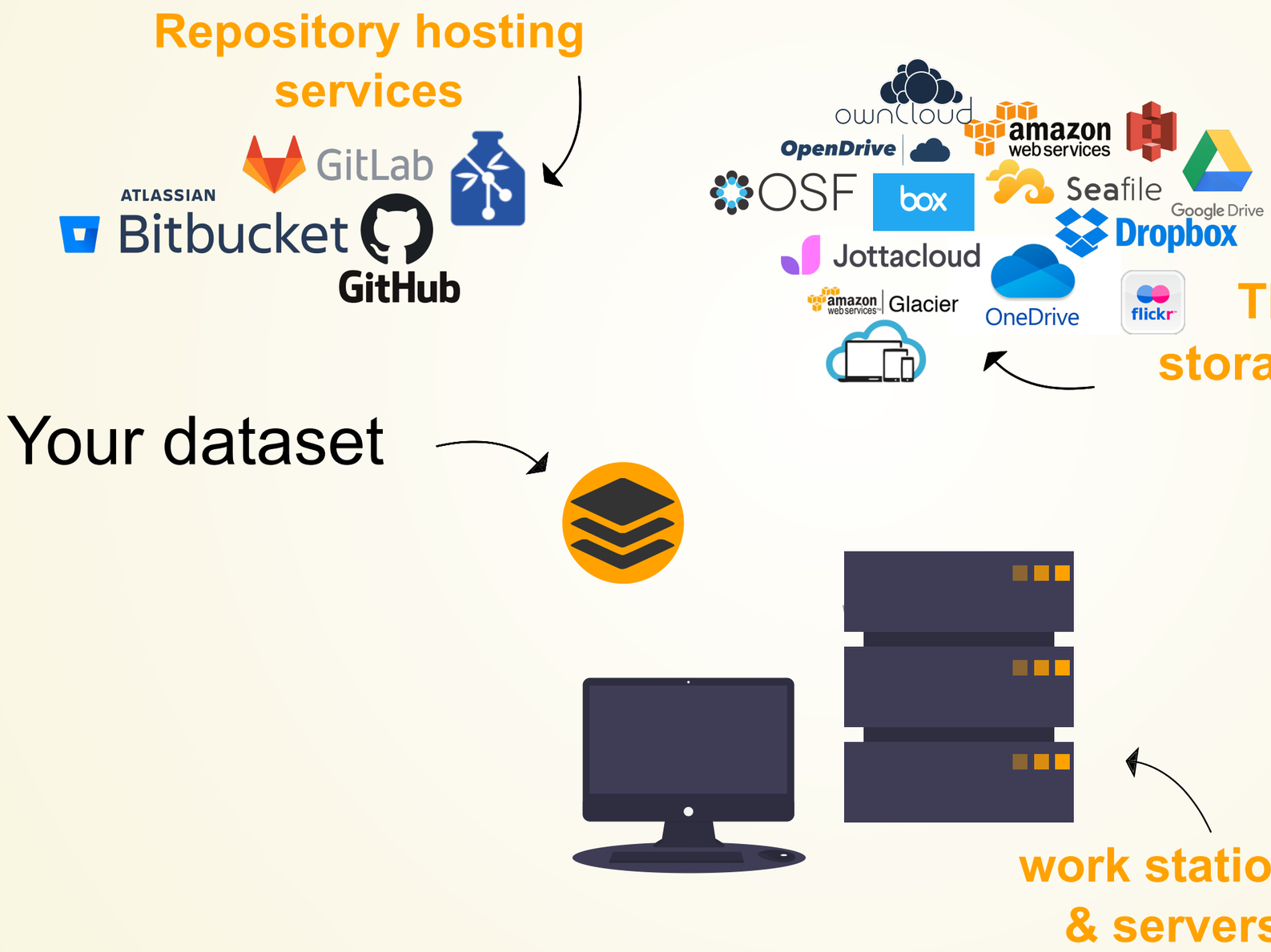
INTEROPERABILITY

- DataLad is built to maximize interoperability and use with hosting and storage technology



PUBLISHING DATASETS

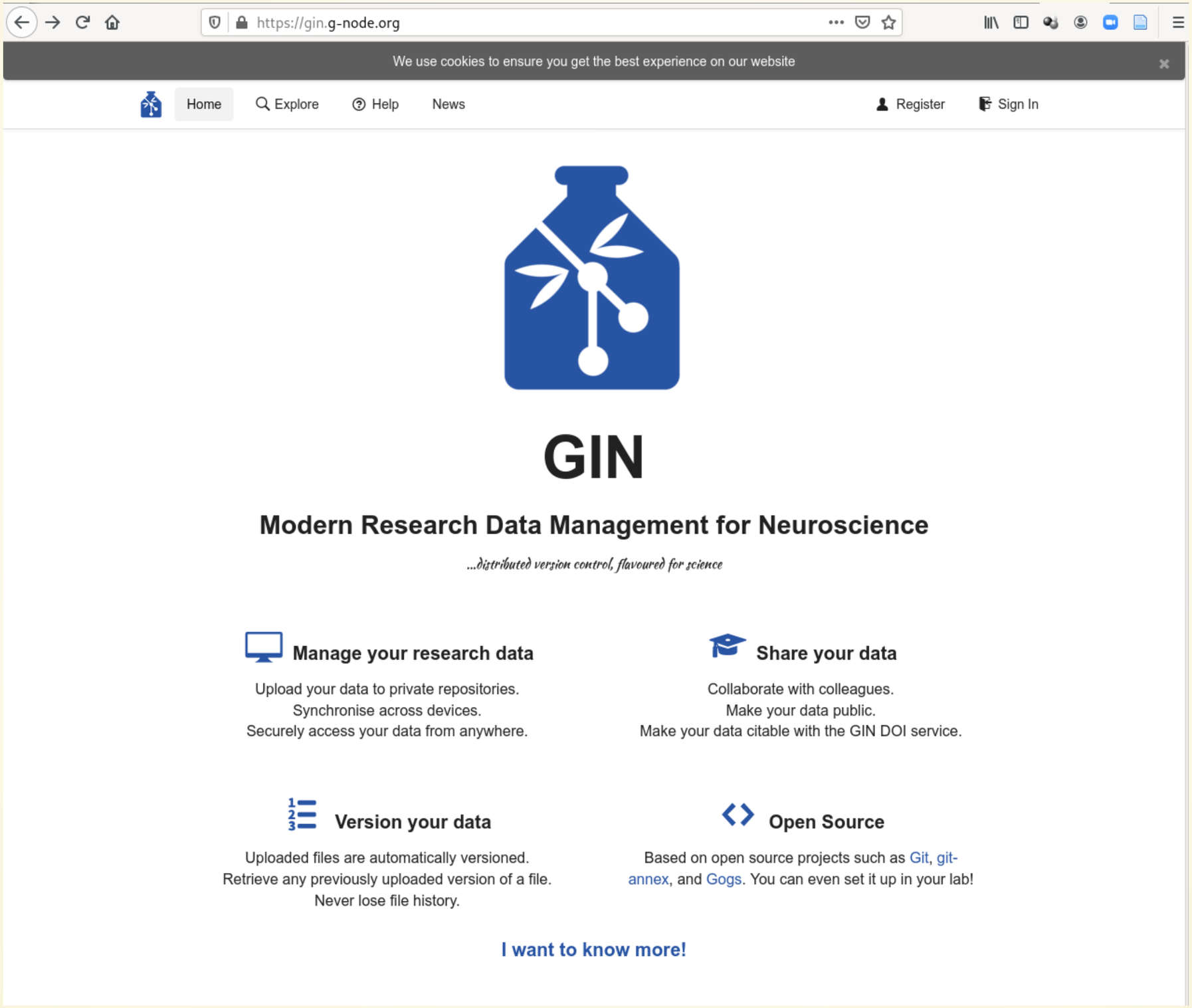
I have a dataset on my computer. How can I share it, or collaborate on it?
General information: handbook.datalad.org/r.html?publish



Today: Publishing a dataset to Gin

GIN.G-NODE.ORG

Gin is a free repository hosting service.



...PUBLISHING DATASETS

DataLad has convenience functions to create `sibling`-repositories on various infrastructure and third party services (GitHub, GitLab, OSF, WebDAV-based services, DataVerse, ...) , to which data can then be published with `push`.

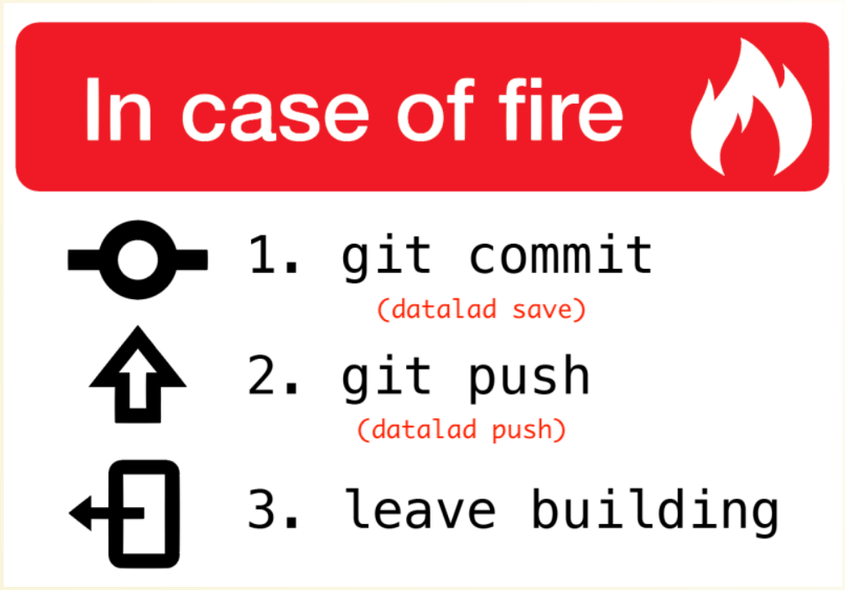
```
datalad create-sibling-gin example-analysis --access-protocol ssh
```
copy

You can verify the dataset's siblings with the `siblings` command:

```
datalad siblings
```
copy

And we can push our complete dataset (Git repository and annex) to GIN:

```
datalad push --to gin
```
copy



STEP-BY-STEP: WEBINTERFACE

STEP-BY-STEP: COMMAND LINE

WHY USE DATALAD?

- Mistakes are not forever anymore: Easy version control, regardless of file size
- Who needs short-term memory when you can have run-records?
- Disk-usage magic: Have access to more data than your hard drive has space
- Collaboration and updating mechanisms: Alice shares her data with Bob. Alice fixes a mistake and pushes the fix. Bob says "datalad update" and gets her changes. And vice-versa.
- Transparency: Shared datasets keep their history. No need to track down a former student, ask their project what was done.

ACKNOWLEDGEMENTS

Funders

NSF 1429999

germany-usa

SPONSORED BY THE Federal Ministry of Education and Research

BMBF 01GQ1411

JÜLICH Forschungszentrum

DFG

EUROPEAN UNION
European Regional Development Fund

cbbs center for behavioral brain sciences

SACHSEN-ANHALT
Ministerium für Wissenschaft und Wirtschaft

DataLad software & ecosystem

- Psychoinformatics Lab, Research center Jülich
- Center for Open Neuroscience, Dartmouth College
- Joey Hess (git-annex)
- *>100 additional contributors*

Collaborators

Human Brain Project

CONP PCNO

OpenNEURO

eBRAIN Health

MOTOR SFB 1451

brainlife.io

cbrain

VirtualBrainCloud

UNLOCKING THINGS

- `datalad run` "unlocks" everything specified as `--output`
- Outside of `datalad run`, you can use `datalad unlock`
- This makes annex'ed files writeable:

```
$ ls -l myfile
lrwxrwxrwx 1 adina adina  108 Nov 17 07:08 myfile -> .git/annex/objects/22/Gw/MD5E-s7--f447b20a7

# unlocking
$ datalad unlock myfile
unlock(ok): myfile (file)
$ ls -l myfile
-rw-r--r-- 1 adina adina    7 Nov 17 07:08 myfile  # not a symlink anymore!
```

- `datalad save` "locks" the file again

```
$ datalad save
add(ok): myfile (file)
action summary:
  add (ok: 1)
  save (notneeded: 1)

$ ls -l myfile
lrwxrwxrwx 1 adina adina 108 Nov 17 07:08 myfile -> .git/annex/objects/22/Gw/MD5E-s7--f447b20a7f
```

Some tools (e.g., MatLab) don't like symlinks. Unlocking or running matlab with "datalad run" helps!

REMOVING DATASETS

- As mentioned before, annexed data is write-protected. So when you try to `rm -rf` a dataset, this happens:

```
$ rm -rf mydataset                                                                copy
rm: cannot remove 'mydataset/.git/annex/objects/70/GM/MD5E-s27246--8b7ea027f6db1cda7af496e97d4eb
rm: cannot remove 'mydataset/.git/annex/objects/70/GM/MD5E-s35756--af496e97d4eb7c98b7ea027f6db1c
[...]
```

😱

- (If you accidentally ever do this, you need to apply write permissions recursively to all files)

```
$ chmod -R +w mydataset
$ rm -rf mydataset                  # success!
```

REMOVING DATASETS

- The correct way to remove a dataset is using `datalad remove`:

```
$ datalad remove -d ds001241
remove(ok): . (dataset)
action summary:
  drop (notneeded: 1)
  remove (ok: 1)
```

copy

- If a dataset contains file for which no other remote copy is known, you'll get a warning:

```
$ datalad remove -d mydataset
[WARNING] Running drop resulted in stderr output: git-annex: drop: 1 failed

[ERROR  ] unsafe; Could only verify the existence of 0 out of 1 necessary copies; Rather than dr
drop(error): interdisciplinary.png (file) [unsafe; Could only verify the existence of 0 out of 1
[WARNING] could not drop some content in /tmp/mydataset ['/tmp/mydataset/interdisciplinary.png']
drop(impossible): . (directory) [could not drop some content in /tmp/mydataset ['/tmp/mydataset/
action summary:
  drop (error: 1, impossible: 1)
```

copy

- In that case, use `--nocheck` to force removal:

```
$ datalad remove -d mydataset --nocheck                                    1 !
remove(ok): . (dataset)
```

copy

REMOVING DATASETS

- If a dataset contains subdatasets, `datalad remove` will also error:

```
$ datalad remove -d myds                                                    copy
drop(ok): README.md (file) [locking gin...]
drop(ok): . (directory)
[ERROR  ] to be uninstalled dataset Dataset(/tmp/myds) has present subdatasets, forgot --recursi
remove(error): . (dataset) [to be uninstalled dataset Dataset(/tmp/myds) has present subdatasets
action summary:
  drop (ok: 3)
  remove (error: 1)
```

- In that case, use `--recursive` to remove all subdatasets, too:

```
$ datalad remove -d myds --recursive                                        copy
uninstall(ok): input (dataset)
remove(ok): . (dataset)
action summary:
  drop (notneeded: 2)
  remove (ok: 1)
  uninstall (ok: 1)
```

- A complete overview of file system operations is in handbook.datalad.org/en/latest/basics/101-136-filesystem.html