DATA MANAGEMENT FOR NEUROIMAGING

WITH DATALAD

Adina Wagner
@AdinaKrik

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

Slide sources: https://github.com/datalad-handbook/datalad-course/
Slide archive: doi.org/10.5281/zenodo.6880616

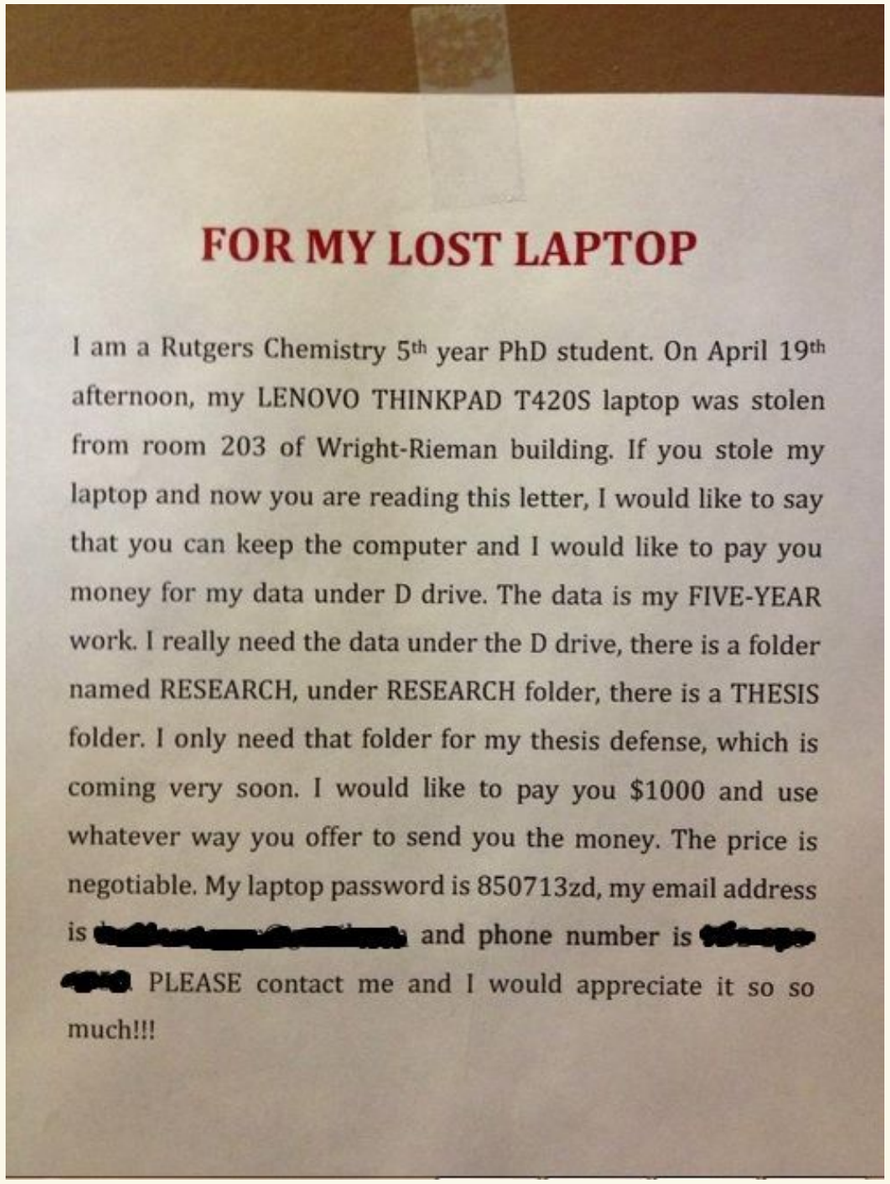
COMMON PROBLEMS IN SCIENCE

You write a paper & stay up late to generate good-looking figures, but you have to tweak many parameters and display options. The next morning, you have no idea which parameters produced which figures, and which of the figures fit to what you report in the paper.

Image credit: Illustration adapted from Scriberia and The Turing Way

COMMON PROBLEMS IN SCIENCE

Your research project produces phenomenal results, but your laptop, the only place that stores the source code for the results, is stolen or breaks



FOR MY LOST LAPTOP

I am a Rutgers Chemistry 5th year PhD student. On April 19th afternoon, my LENOVO THINKPAD T420S laptop was stolen from room 203 of Wright-Rieman building. If you stole my laptop and now you are reading this letter, I would like to say that you can keep the computer and I would like to pay you money for my data under D drive. The data is my FIVE-YEAR work. I really need the data under the D drive, there is a folder named RESEARCH, under RESEARCH folder, there is a THESIS folder. I only need that folder for my thesis defense, which is coming very soon. I would like to pay you $1000 and use whatever way you offer to send you the money. The price is negotiable. My laptop password is 850713zd, my email address is [redacted] and phone number is [redacted]. PLEASE contact me and I would appreciate it so so much!!!

COMMON PROBLEMS IN SCIENCE

A graduate student complains that a research idea does not work. Their supervisor can't figure out what the student did and how, and the student can't sufficiently explain their approach (data, algorithms, software). Weeks of discussion and mis-communication ensues because the supervisor can't first-hand explore or use the students project.

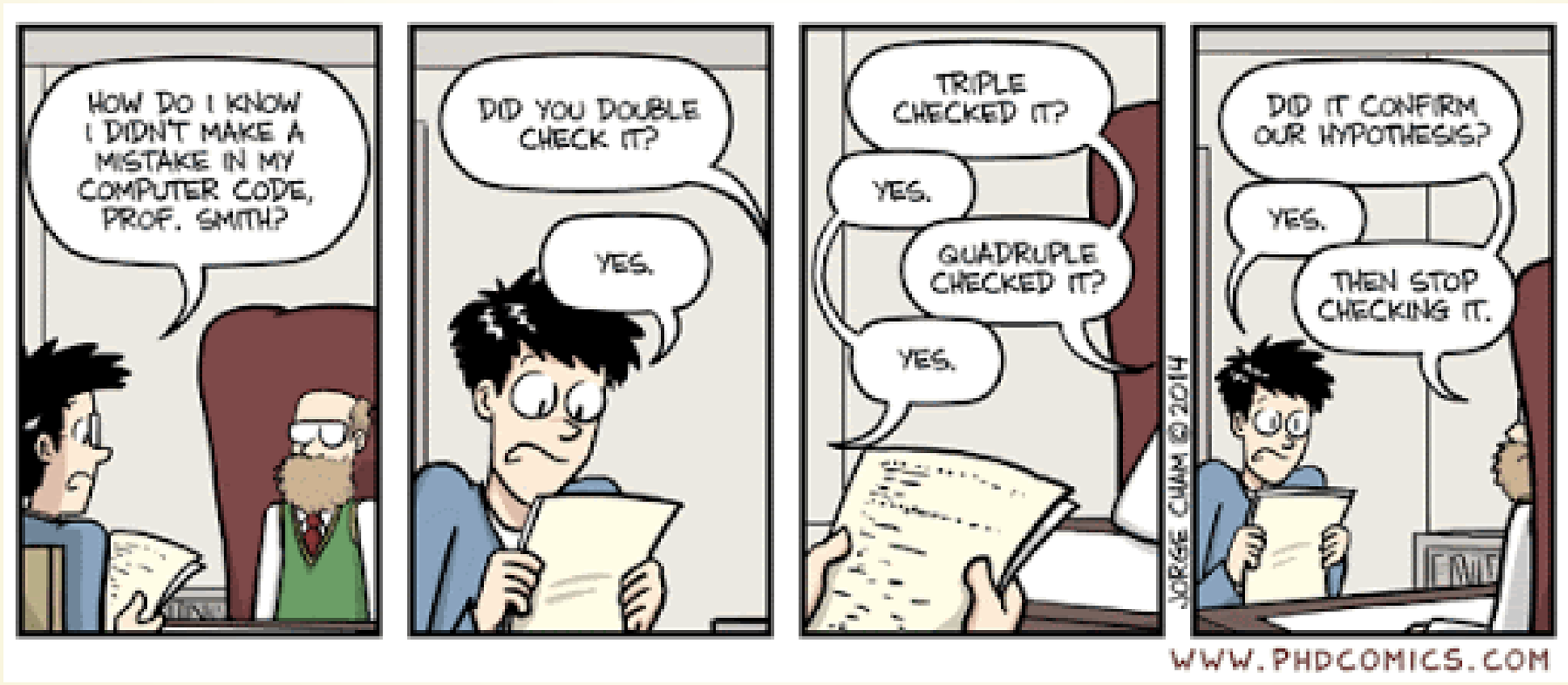


Image credit: http://phdcomics.com/comics.php?f=1693

COMMON PROBLEMS IN SCIENCE

You wrote a script during your PhD that applied a specific method to a dataset. Now, with new data and a new project, you try to reuse the script, but forgot how it worked.

Image credit: http://phdcomics.com/comics.php?f=1693

COMMON PROBLEMS IN SCIENCE

You try to recreate results from another lab's published paper. You base your re-implementation on everything reported in their paper, but the results you obtain look nowhere like the original.

~~COMMON~~ OLD PROBLEMS IN SCIENCE

All these problems were paraphrased from Buckheit & Donoho, **1995**

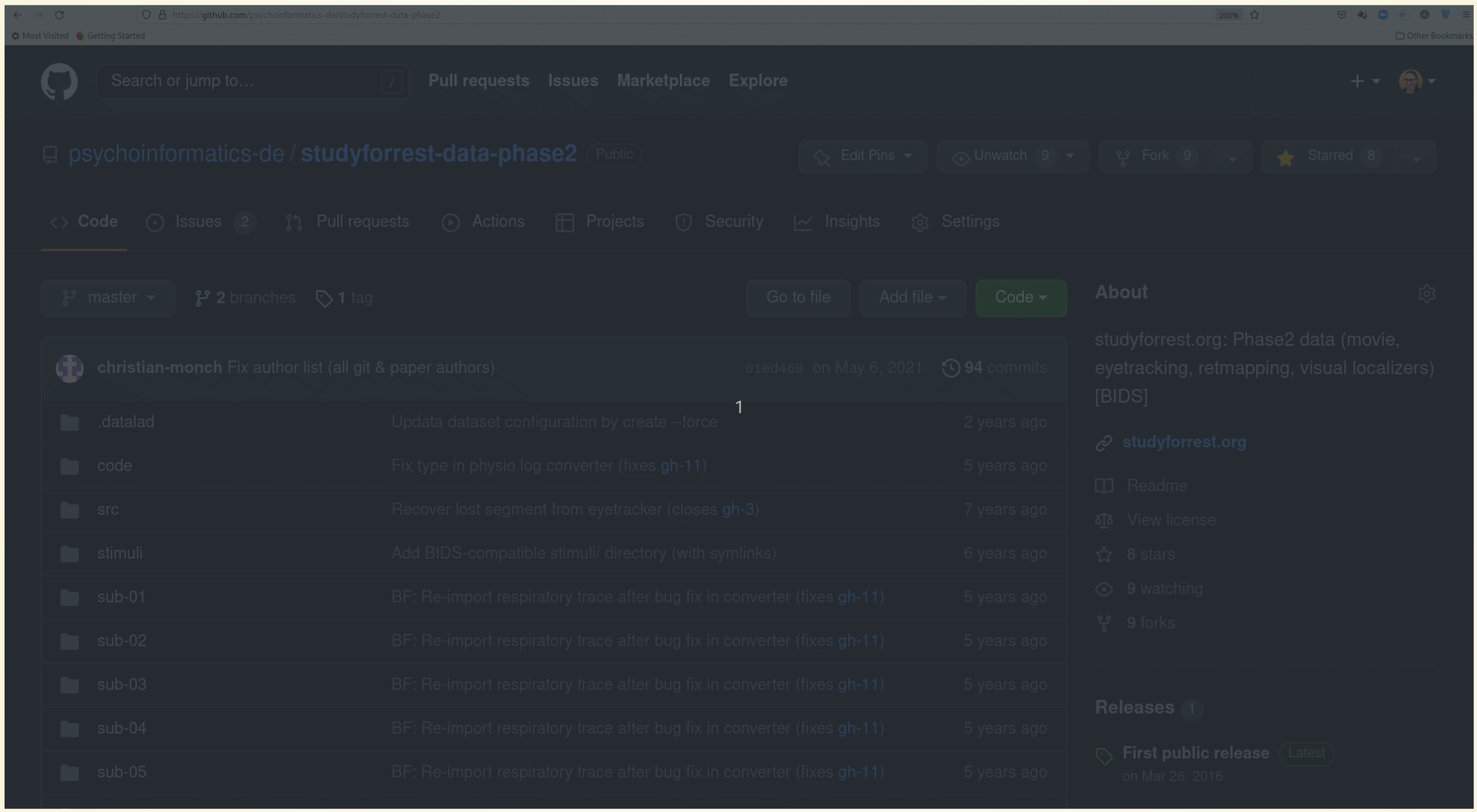
DataLad can help with small or large-scale data management

Free,
open source,
command line tool & Python API

Halchenko, Meyer, Poldrack, ... & Hanke, M. (2021). DataLad: distributed system for joint management of code, data, and their relationship. Journal of Open Source Software, 6(63), 3262.

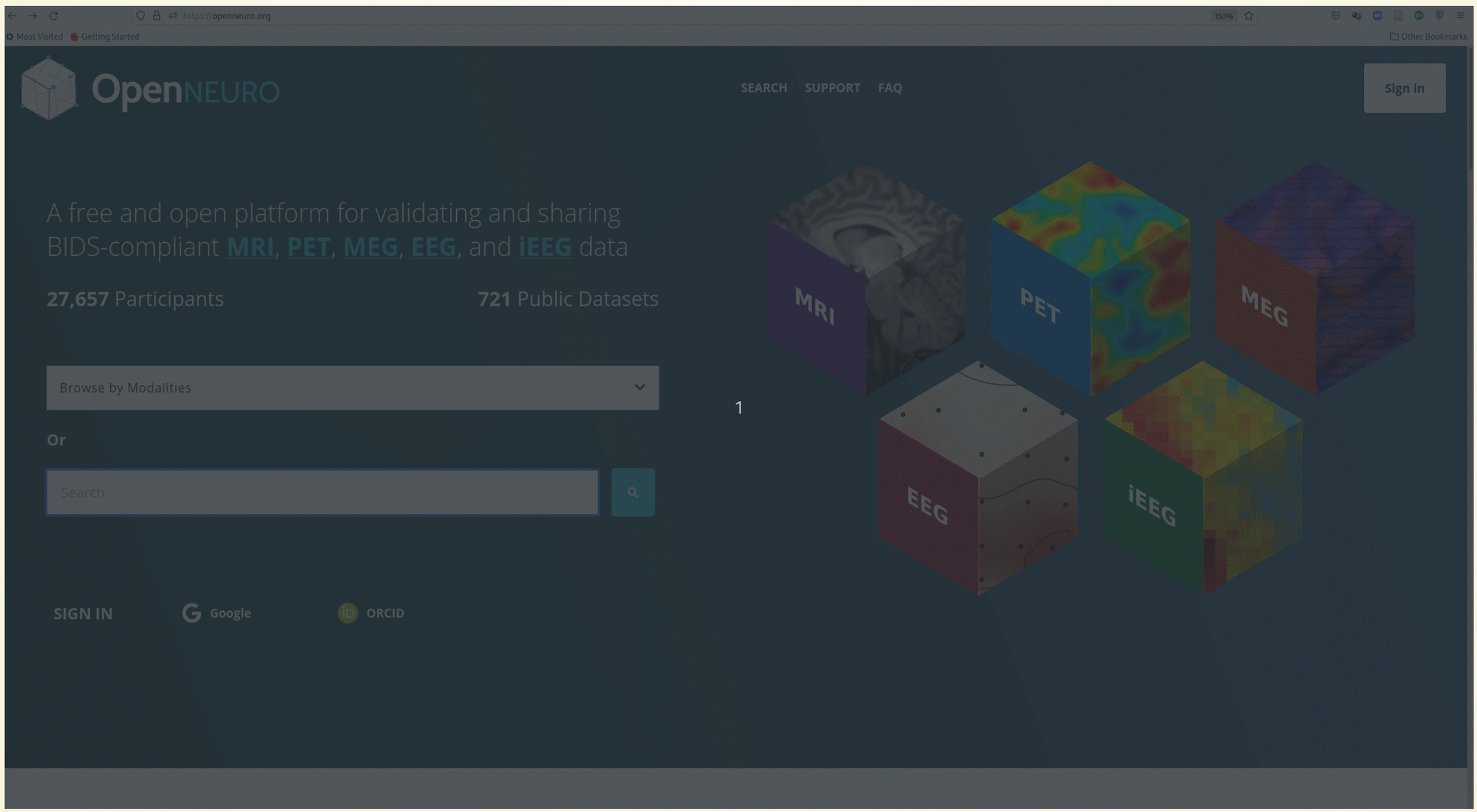
EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

- **Publish or consume datasets** via GitHub, GitLab, OSF, the European Open Science Cloud, or similar services



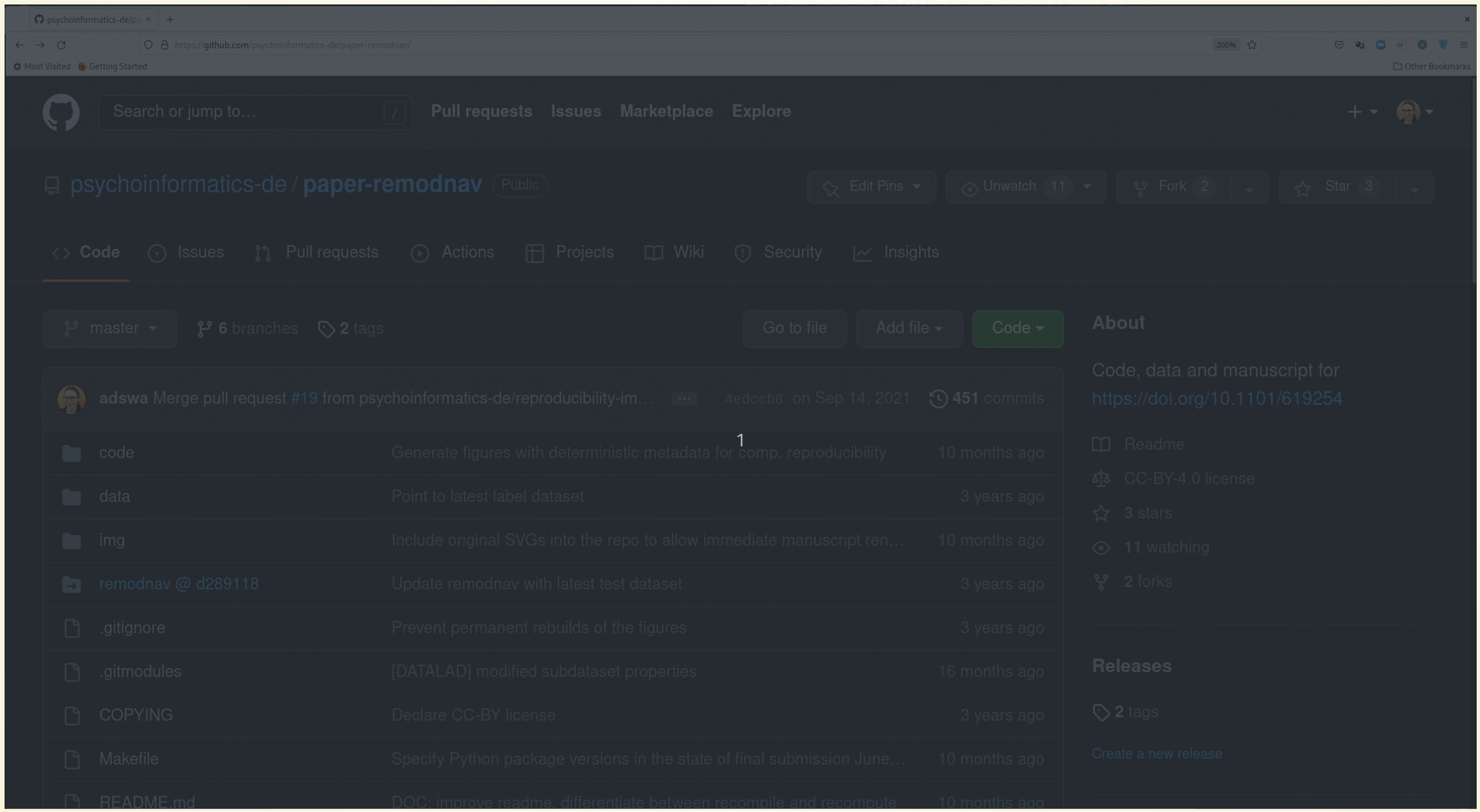
EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

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



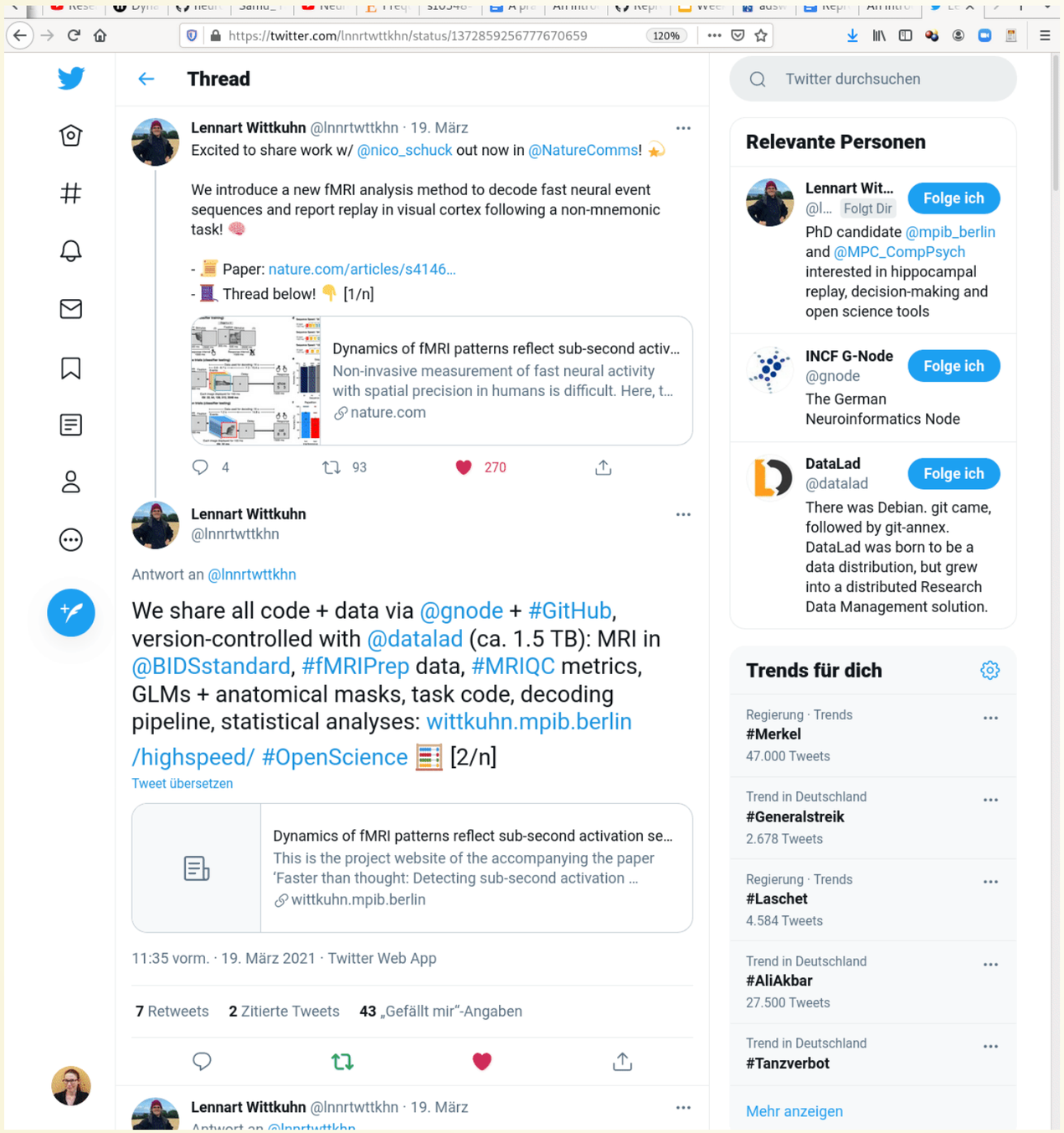
EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

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



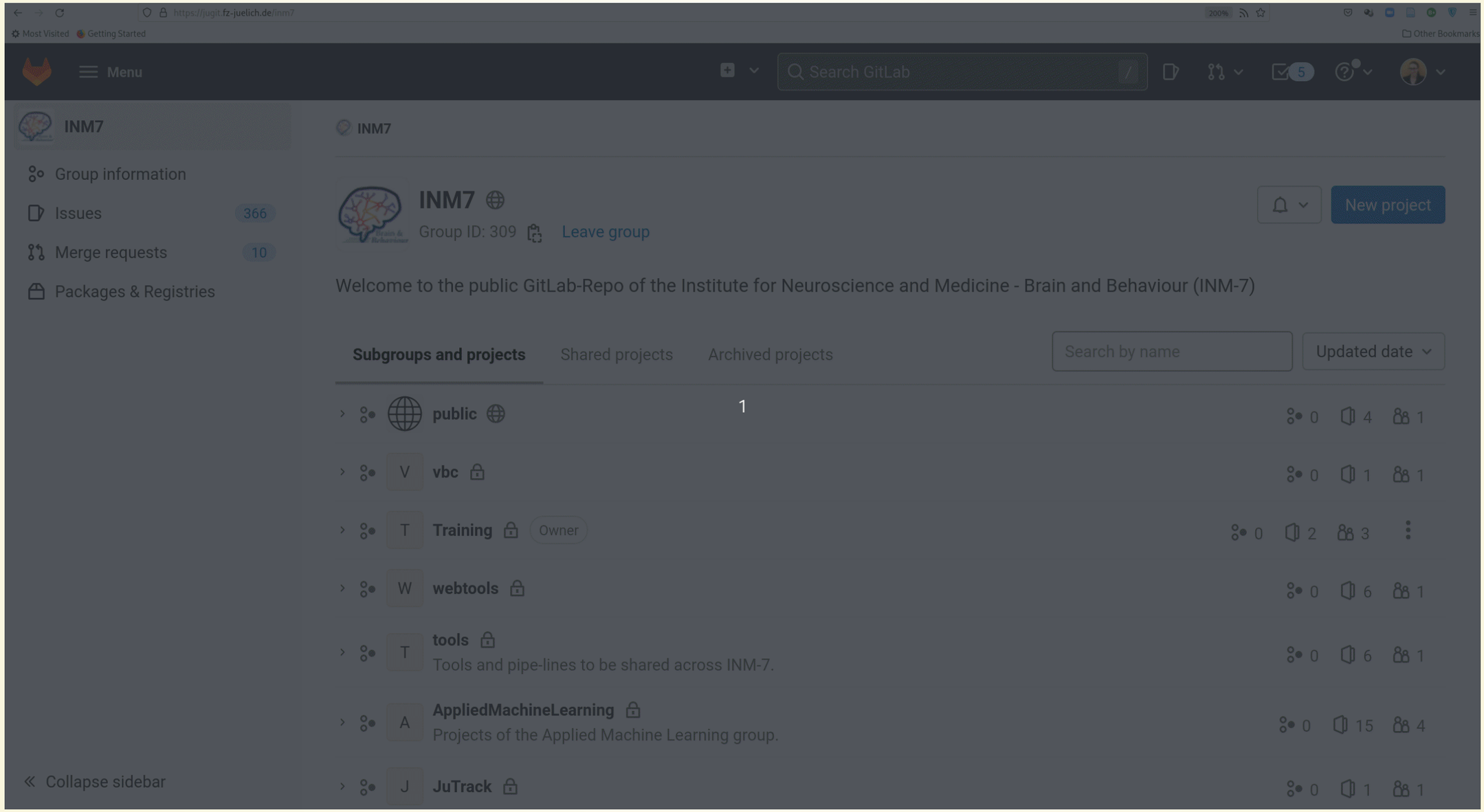
EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

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



EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

- **Central data management** and archival system



DataLad CORE FEATURES:

- Joint **version control** (Git, git-annex): version control data & software alongside your code
- **Provenance capture**: Create and share machine-readable, re-executable provenance records for reproducible, transparent, and FAIR research
- decentral **data transport** mechanisms: Install, share and collaborate on scientific projects; publish, upgrade, and retrieve their contents in a streamlined fashion on demand, and distribute files in a decentral network on the services or infrastructures of your choice

Code for hands-on: handbook.datalad.org

PREREQUISITES: TERMINAL

- DataLad can be used from the command line

```
datalad create mydataset
```

- ... or with its Python API

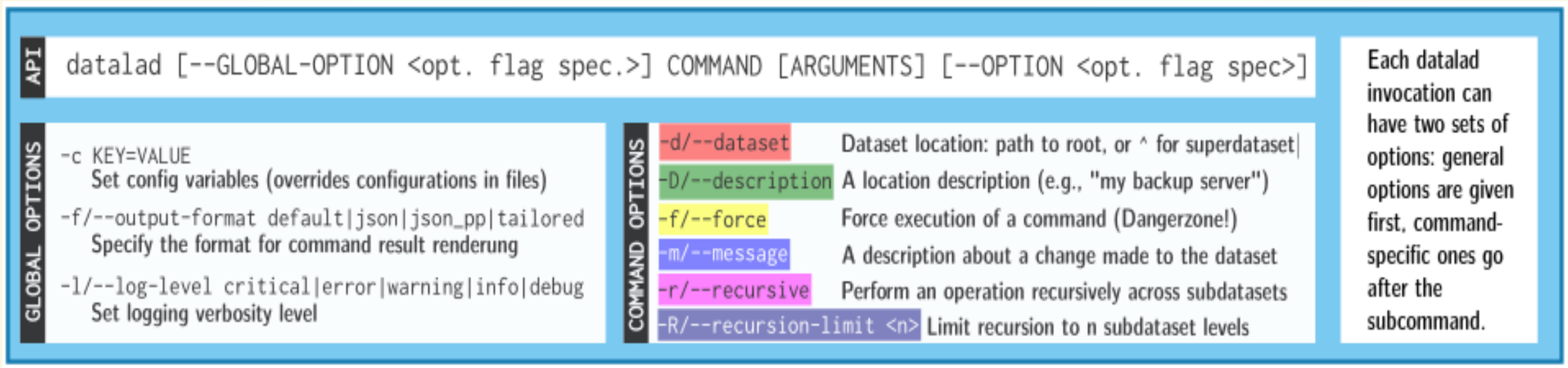
```
import datalad.api as dl
dl.create(path="mydataset")
```

- ... and other programming languages can use it via system call

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

PREREQUISITES: USING DATALAD

- Every DataLad command consists of a main command followed by a sub-command. The main and the sub-command can have options.



- Example (main command, subcommand, several subcommand options):

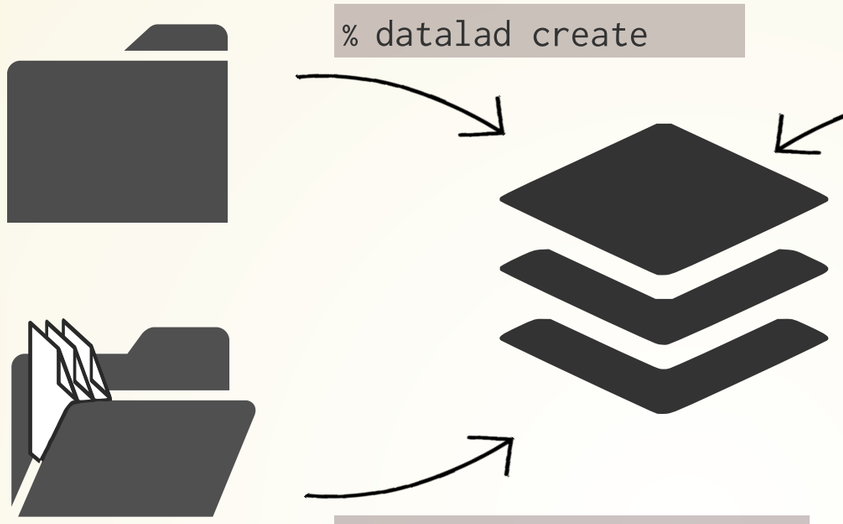
```
$ datalad save -m "Saving changes" --recursive
```

- Use *--help* to find out more about any (sub)command and its options, including detailed description and examples (*q* to close). Use *-h* to get a short overview of all options

```
$ datalad save -h
      Usage: datalad save [-h] [-m MESSAGE] [-d DATASET] [-t ID] [-r] [-R LEVELS]
                          [-u] [-F MESSAGE_FILE] [--to-git] [-J NJOBS] [--amend]
                          [--version]
                          [PATH ...]

Use '--help' to get more comprehensive information.
```

EVERYTHING HAPPENS IN DATALAD DATASETS

create new, empty datasets to populate...



```
% datalad create
```


```
% datalad clone
```

.... or get datasets from other places

```
% datalad create -f
```

.... or transform existing directories into datasets

- Look and feel like a directory on your computer
- content agnostic
- no custom data structures

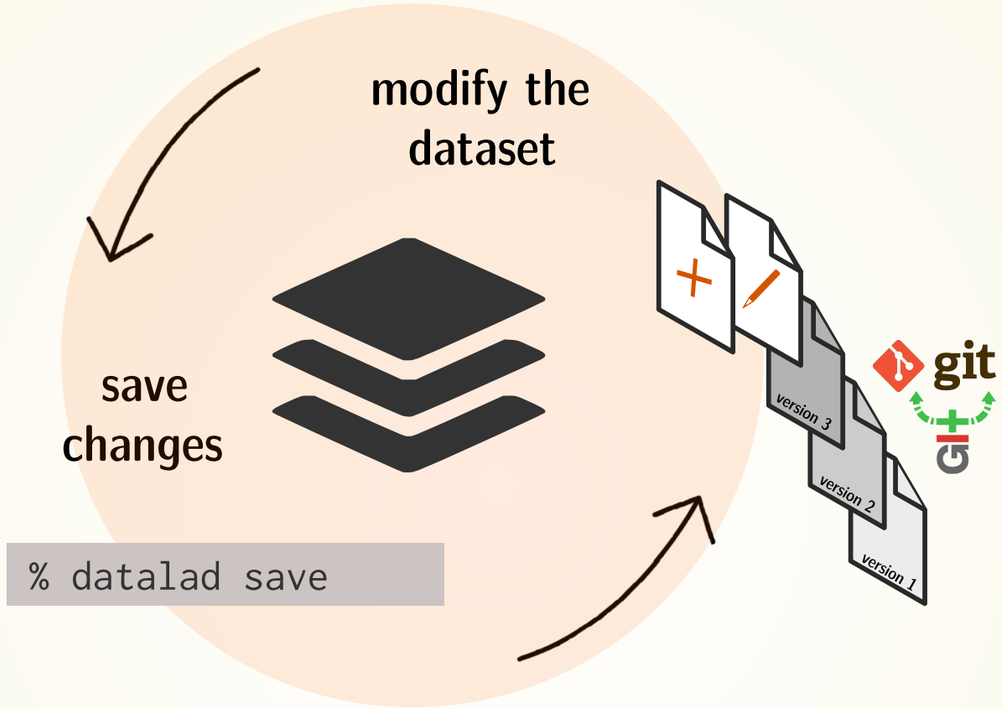
File viewer

```
adina@muninn ~/repos/papers/paper-remodnav on master!
➤ ls
code  remodnav  main.tex   references.bib  svjour3.cls
data  COPYING   Makefile   spbasic.bst
img   diff.aux  README.md  svglov3.clo
```

Terminal view

DATASET = GIT/GIT-ANNEX REPOSITORY

- version control files regardless of size or type



Stay flexible:

- Non-complex DataLad core API (easy for data management novices)
- Pure Git or git-annex commands (for regular Git or git-annex users, or to use specific functionality)

EXHAUSTIVE TRACKING

The building blocks of a scientific result are rarely static

Analysis code evolves

(Fix bugs, add functions, refactor, ...)

EXHAUSTIVE TRACKING

The building blocks of a scientific result are rarely static

Data changes

(errors are fixed, data is extended, naming standards change, an analysis requires only a subset of your data...)

EXHAUSTIVE TRACKING

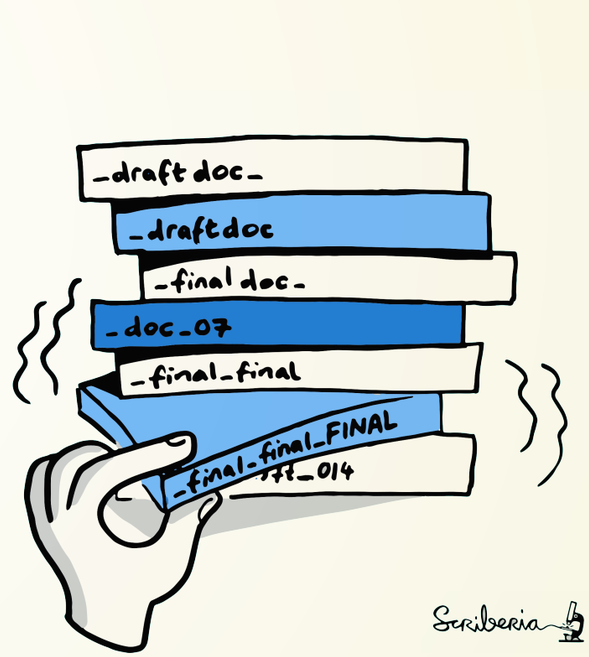
The building blocks of a scientific result are rarely static

Data changes (for real)

(errors are fixed, data is extended, naming standards change, ...)

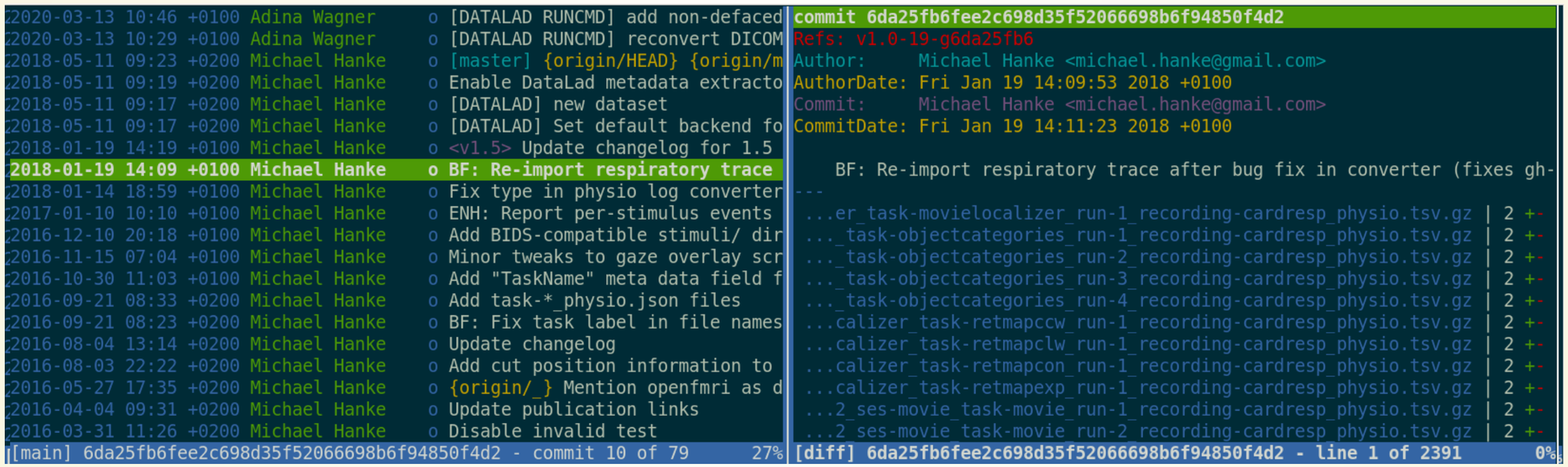
EXHAUSTIVE TRACKING

"Shit, which version of which script produced these outputs from which version of what data... and which software version?"



1. TRANSPARENCY - FOR DATA

Once you track changes to data with version control tools, you can find out *why* it changed, *what* has changed, *when* it changed, and *which version* of your data was used at which point in time.



DIGITAL PROVENANCE

= "The tools and processes used to create a digital file, the responsible entity, and when and where the process events occurred"

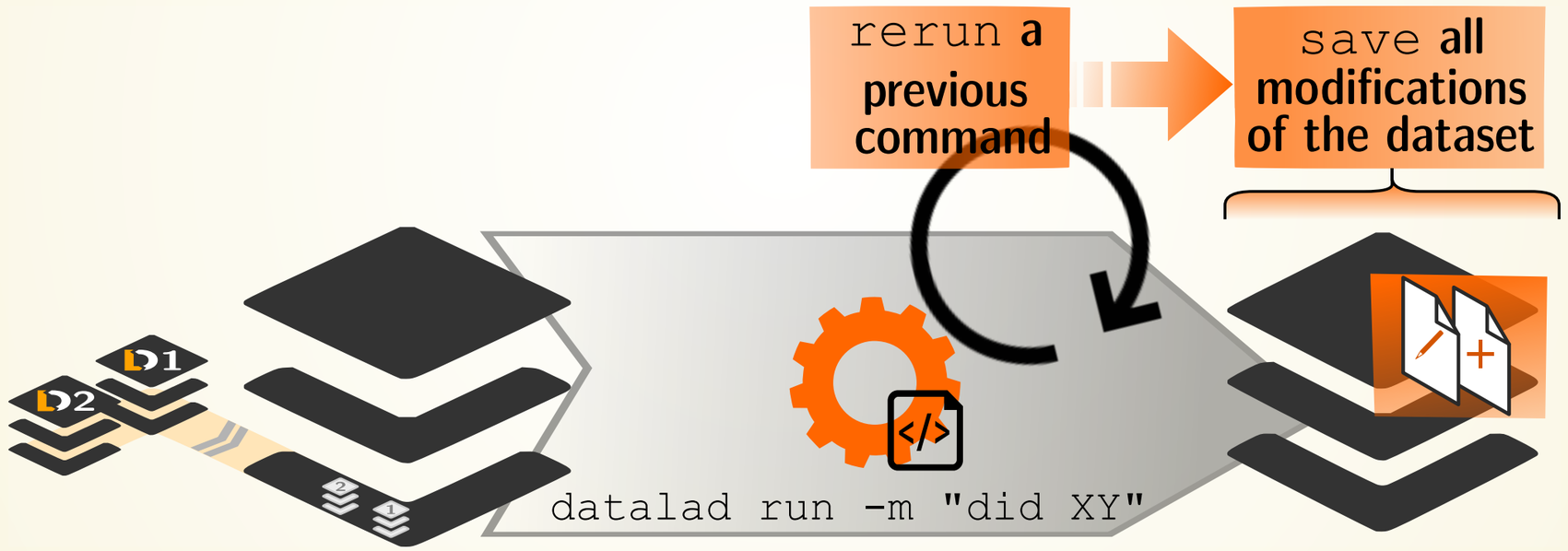
- Have you ever saved a PDF to read later onto your computer, but forgot where you got it from? Or did you ever find a figure in your project, but forgot which analysis step produced it?

PROVENANCE AND REPRODUCIBILITY

datalad run wraps around anything expressed in a command line call and saves the dataset modifications resulting from the execution

PROVENANCE AND REPRODUCIBILITY

datalad rerun repeats captured executions.
If the outcomes differ, it saves a new state of them.



SEAMLESS DATASET NESTING & LINKAGE

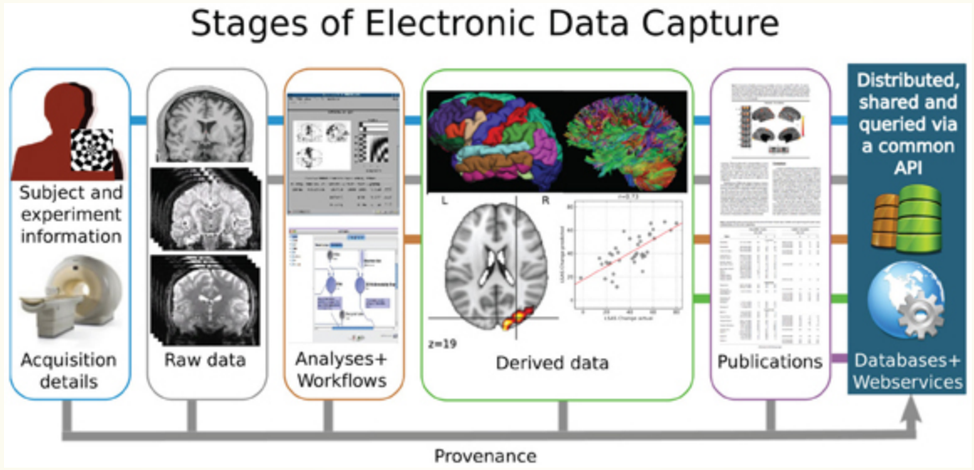
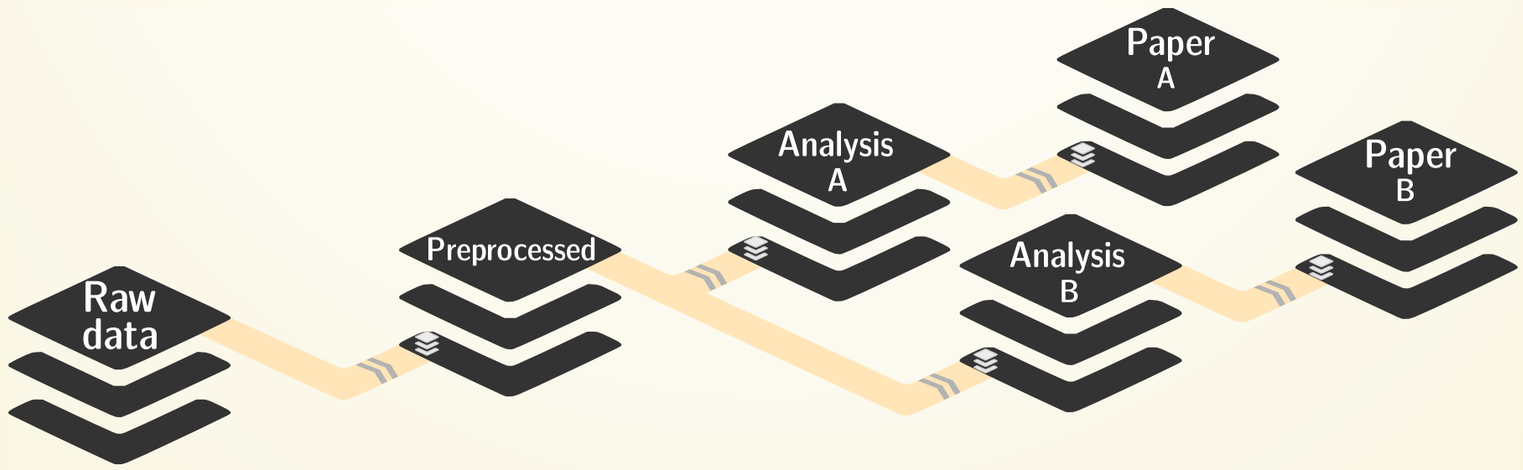
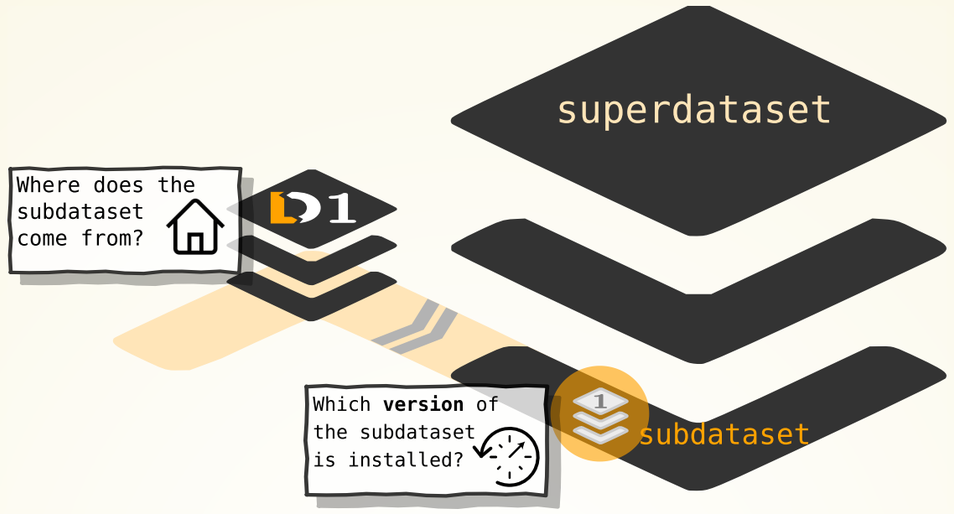


Image credit: Poline et al., 2011



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

SEAMLESS DATASET NESTING & LINKAGE




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

```
$ 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
+       datalad-id = 68bdb3f3-eafa-4a48-bddd-31e94e8b8242
+       datalad-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
```

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 .
```

- files' contents can be retrieved on demand:

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

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

```
# eNKI dataset (1.5TB, 34k files):
$ du -sh
1.5G  .
# HCP dataset (~200TB, >15 million files)
$ du -sh
48G .
```

PLENTY OF DATA, BUT LITTLE DISK-USAGE

Drop file content that is not needed:

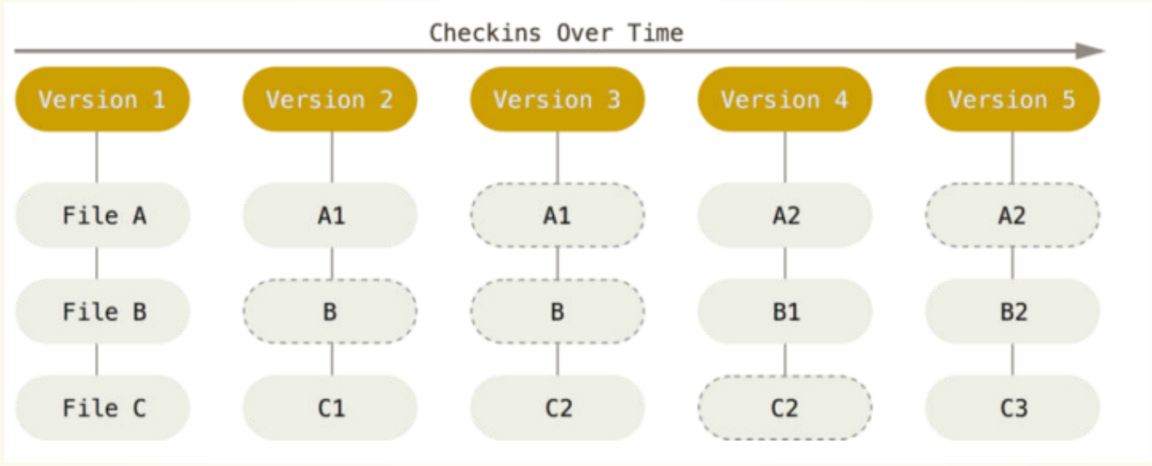
```
$ datalad drop sub-01/ses-movie/func/sub-01_ses-movie_task-movie_run-1_bold.nii.gz
drop(ok): /tmp/studyforrest-data-phase2/sub-01/ses-movie/func/sub-01_ses-movie_task-movie_run-1_
```

When files are dropped, only "meta data" stays behind, and they can be re-obtained on demand.

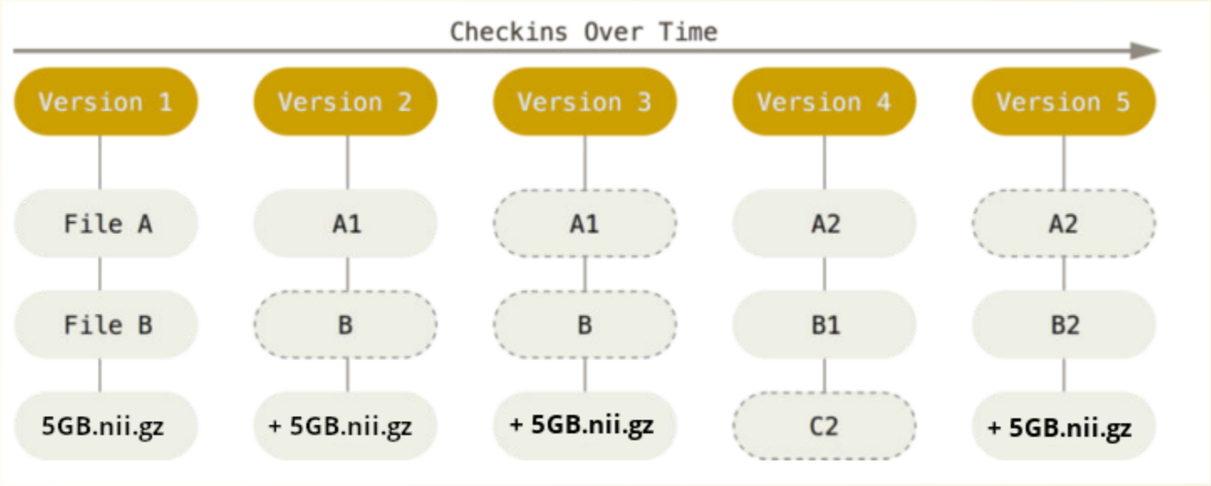
```
dl.get('input/sub-01')

[really complex analysis]

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

THERE ARE TWO VERSION CONTROL TOOLS AT WORK - WHY?

Git does not handle large files well.



THERE ARE TWO VERSION CONTROL TOOLS AT WORK - WHY?



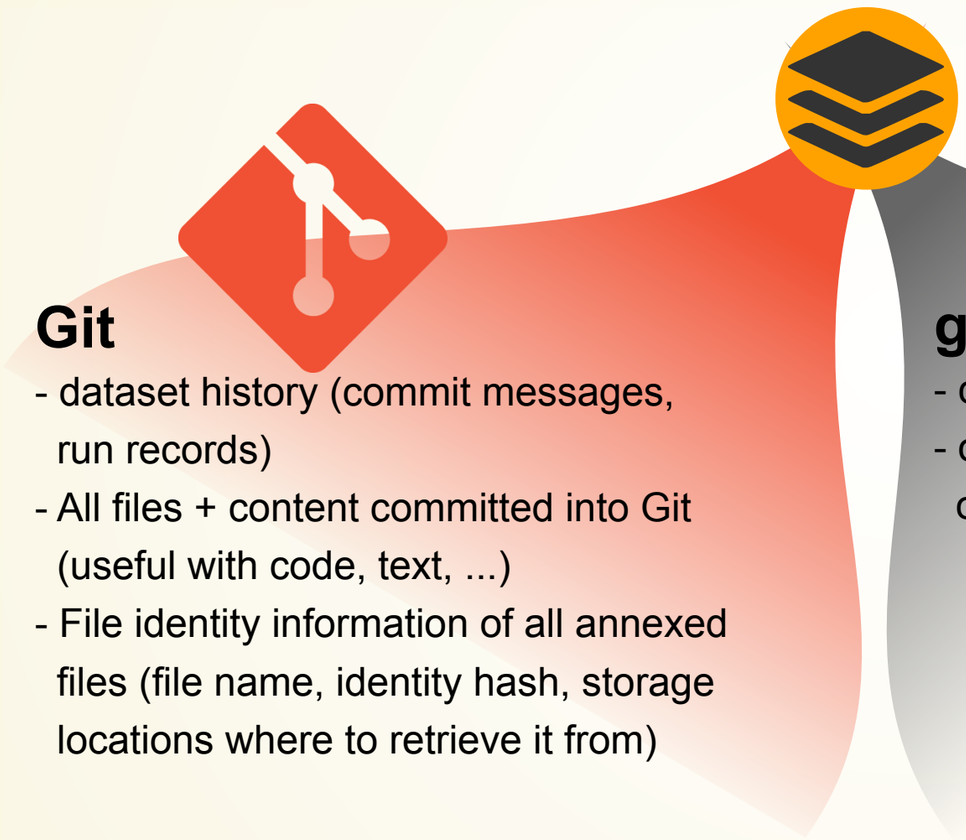
Git does not handle large files well.

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-l
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'
```

git-annex to the rescue! Let's take a look how it works

GIT VERSUS GIT-ANNEX



Git

- dataset history (commit messages, run records)
- All files + content committed into Git (useful with code, text, ...)
- File identity information of all annexed files (file name, identity hash, storage locations where to retrieve it from)

git-annex

- contents of annexed files
- organized in the "annex" or "object tree" of the dataset

DATASET INTERNALS

- Where the filesystem allows it, annexed files are symlinks:

```
$ ls -l sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz
lrwxrwxrwx 1 adina adina 142 Jul 22 19:45 sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz ->
../../.git/annex/objects/kZ/K5/MD5E-s24180157--aeb0e5f2e2d5fe4ade97117a8cc5232f.nii.gz/MD5E-s24180
--aeb0e5f2e2d5fe4ade97117a8cc5232f.nii.gz
```

(PS: especially useful in datasets with many identical files)

- The symlink reveals this internal data organization based on identity hash:

```
$ md5sum sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz
aeb0e5f2e2d5fe4ade97117a8cc5232f  sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz
```

- The (tiny) symlink instead of the (potentially large) file content is committed - version controlling precise file identity without checking contents into Git

```
diff --git a/sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz b/sub-02/func/sub-02_task-oneback_run-01_bold.nii.
new file mode 120000
index 0000000..398e7f1
--- /dev/null
+++ b/sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz
@@ -0,0 +1 @@
+../../.git/annex/objects/kZ/K5/MD5E-s24180157--aeb0e5f2e2d5fe4ade97117a8cc5232f.nii.gz/MD5E-s24180157--aeb0e5f2e2
```

- File contents can be shared via almost all standard infrastructure. File availability information is a decentral network. A file can exist in multiple different locations.

```
$ git annex whereis code/nilearn-tutorial.pdf
whereis code/nilearn-tutorial.pdf (2 copies)
        cf13d535-b47c-5df6-8590-0793cb08a90a -- [datalad]
        e763ba60-7614-4b3f-891d-82f2488ea95a -- jovyan@jupyter-adswa:~/my-analysis [here]

  datalad: https://raw.githubusercontent.com/datalad-handbook/resources/master/nilearn-tutorial.pd
```

GIT VERSUS GIT-ANNEX

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

By default, everything is annexed.

Two consequences:

- Annexed contents are not available right after cloning, only content identity and availability information (as they are stored in Git). Everything that is annexed needs to be retrieved with `datalad get` from whereever it is stored.
- Files stored in Git are modifiable, annexed files are protected against accidental modifcations

Git	**git-annex**
handles **small** files well (text, code)	handles **all** types and sizes of files well
file contents are in the Git history and will be **shared** upon git/datalad push	file contents are in the annex. Not necessarily shared
Shared with every dataset clone	**Can be kept private** on a per-file level when sharing the dataset
Useful: Small, non-binary, frequently modified, need-to-be-accessible (DUA, README) files	Useful: Large files, private files

Useful background information for demo later. Read this handbook chapter for details

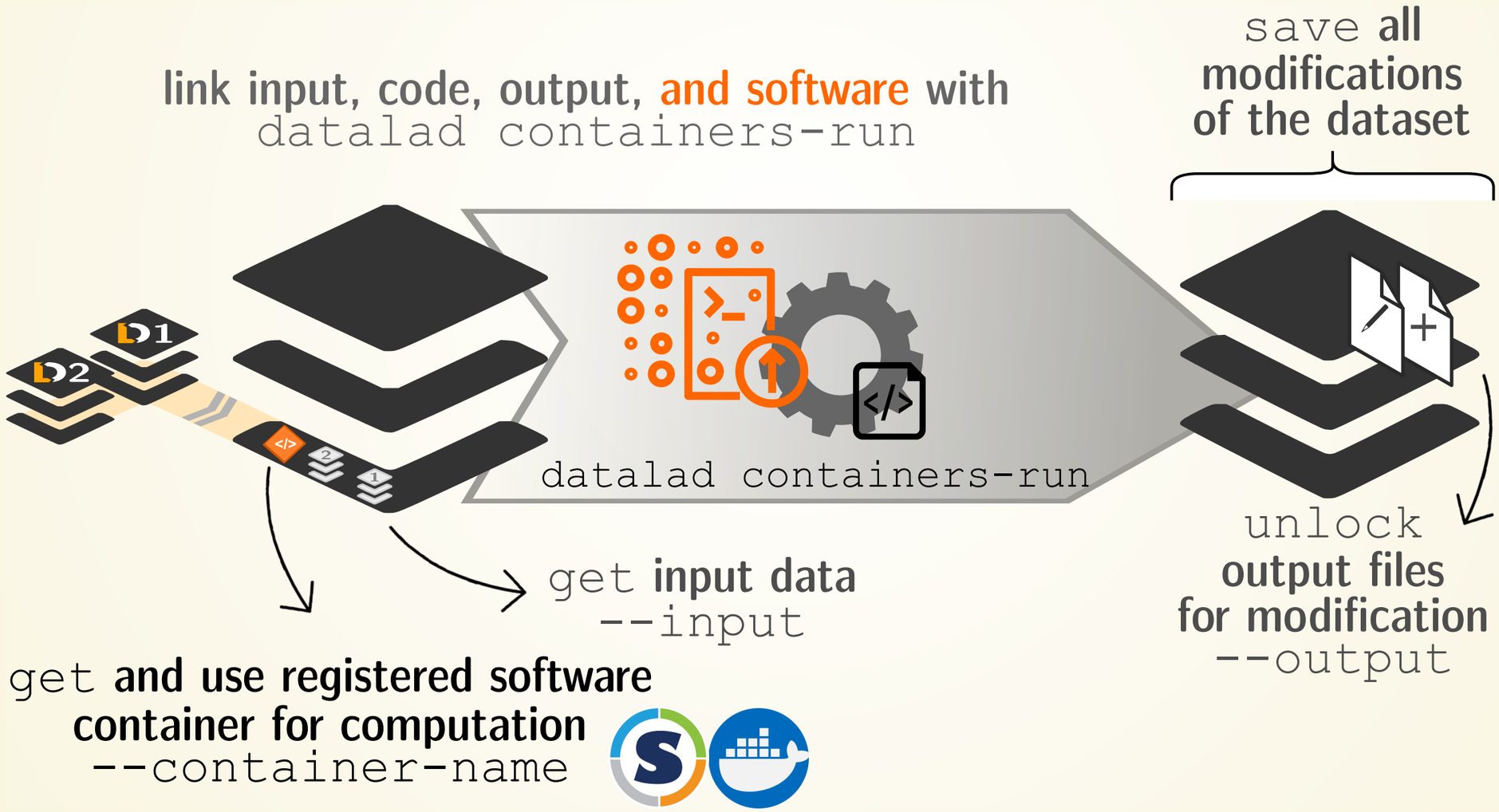
GIT VERSUS GIT-ANNEX

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`)

COMPUTATIONAL PROVENANCE

- The `datalad-container` extension gives DataLad commands to register software containers as "just another file" to your dataset, and **datalad containers-run** analysis inside the container, capturing software as additional provenance



SHARING DATASETS

Repository hosting services

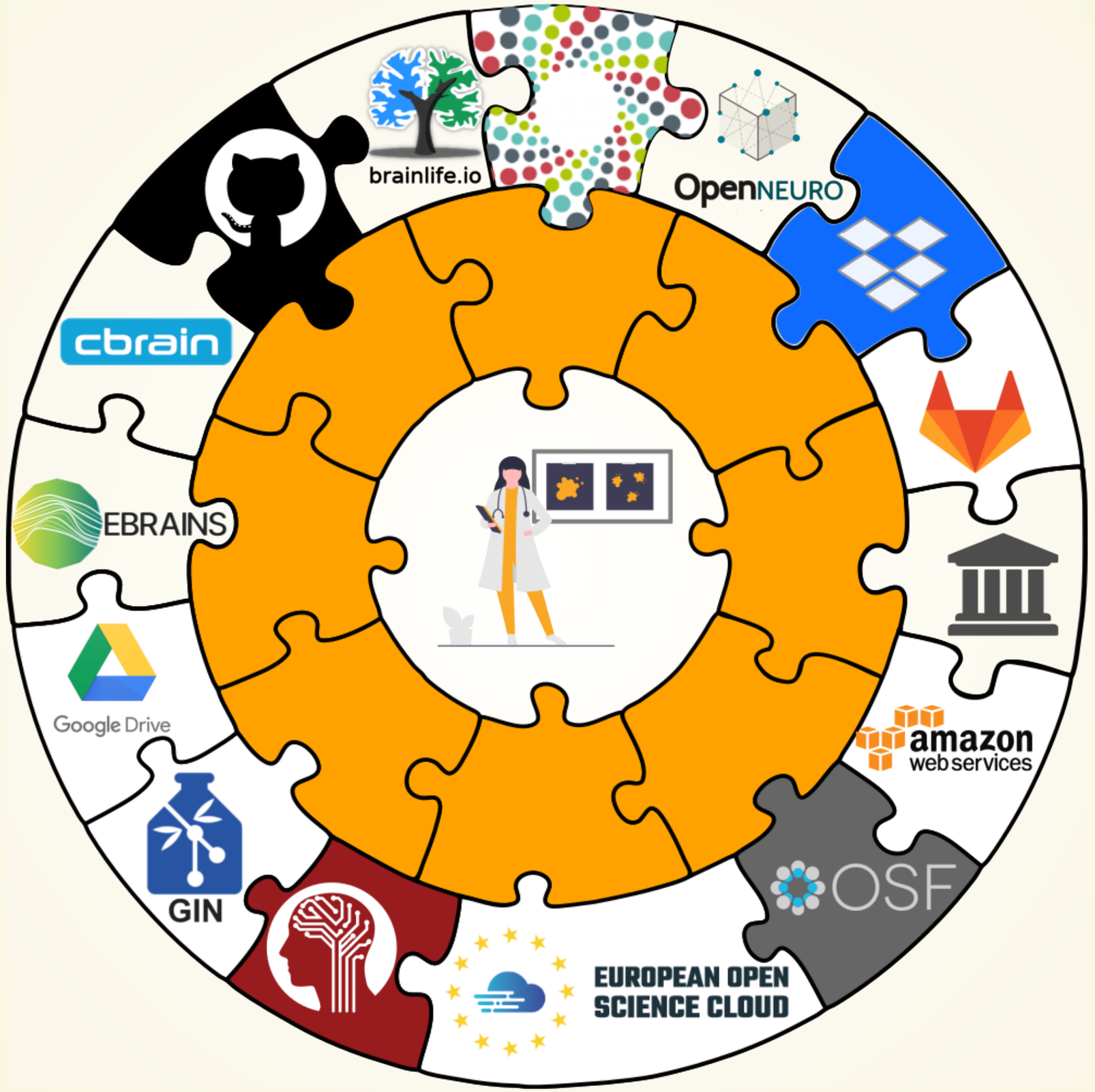
GitLab
ATLASSIAN Bitbucket
GitHub

ownCloud
OpenDrive
amazon web services
OSF
box
Seafile
Google Drive
Dropbox
Jottacloud
amazon web services Glacier
OneDrive
flickr

Third party storage providers

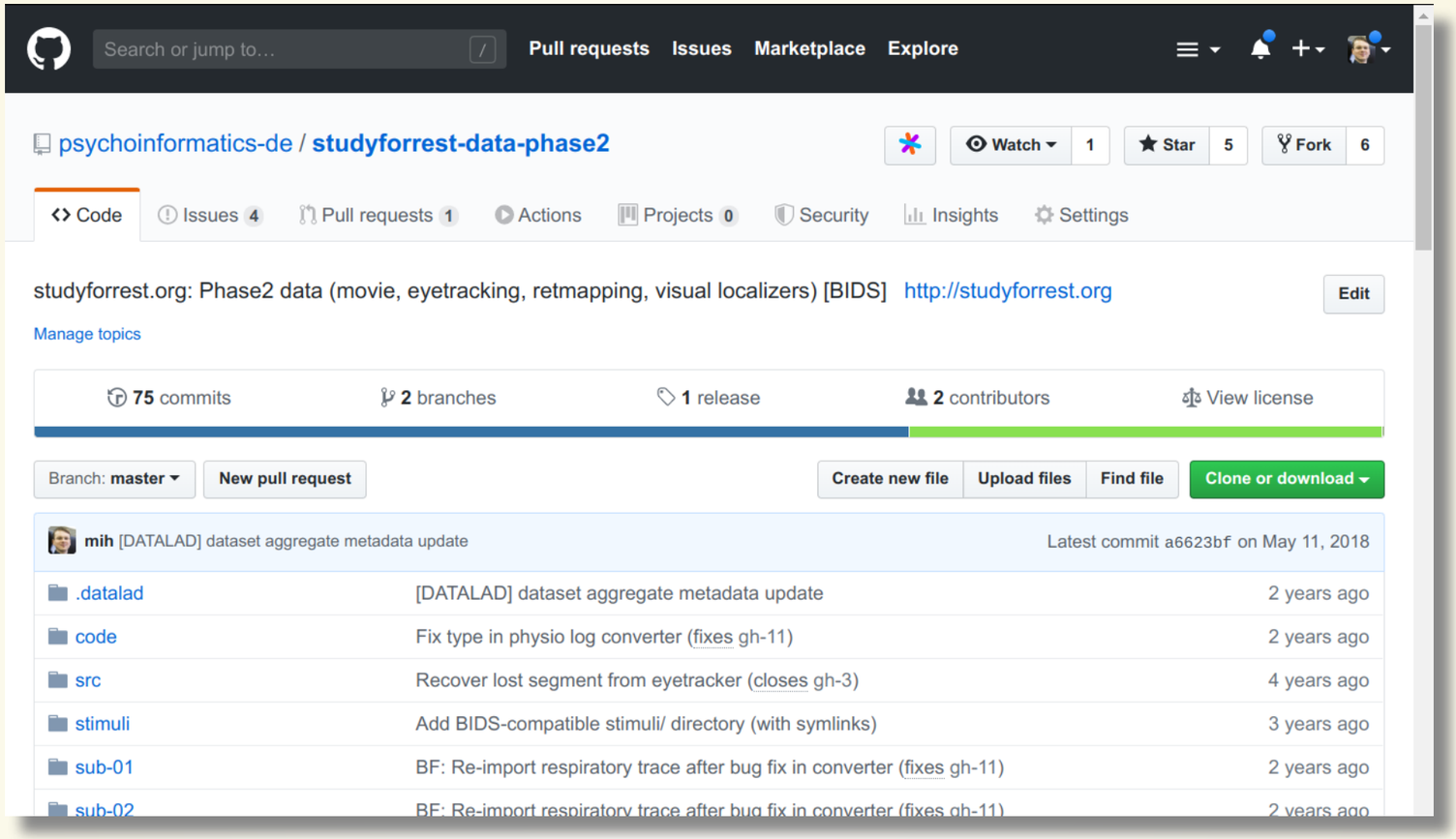
Your dataset

work stations & servers



Apart from **local computing infrastructure** (from private laptops to computational clusters), datasets can be hosted in major **third party repository hosting and cloud storage** services. More info: Chapter on Third party infrastructure.

SERVICES

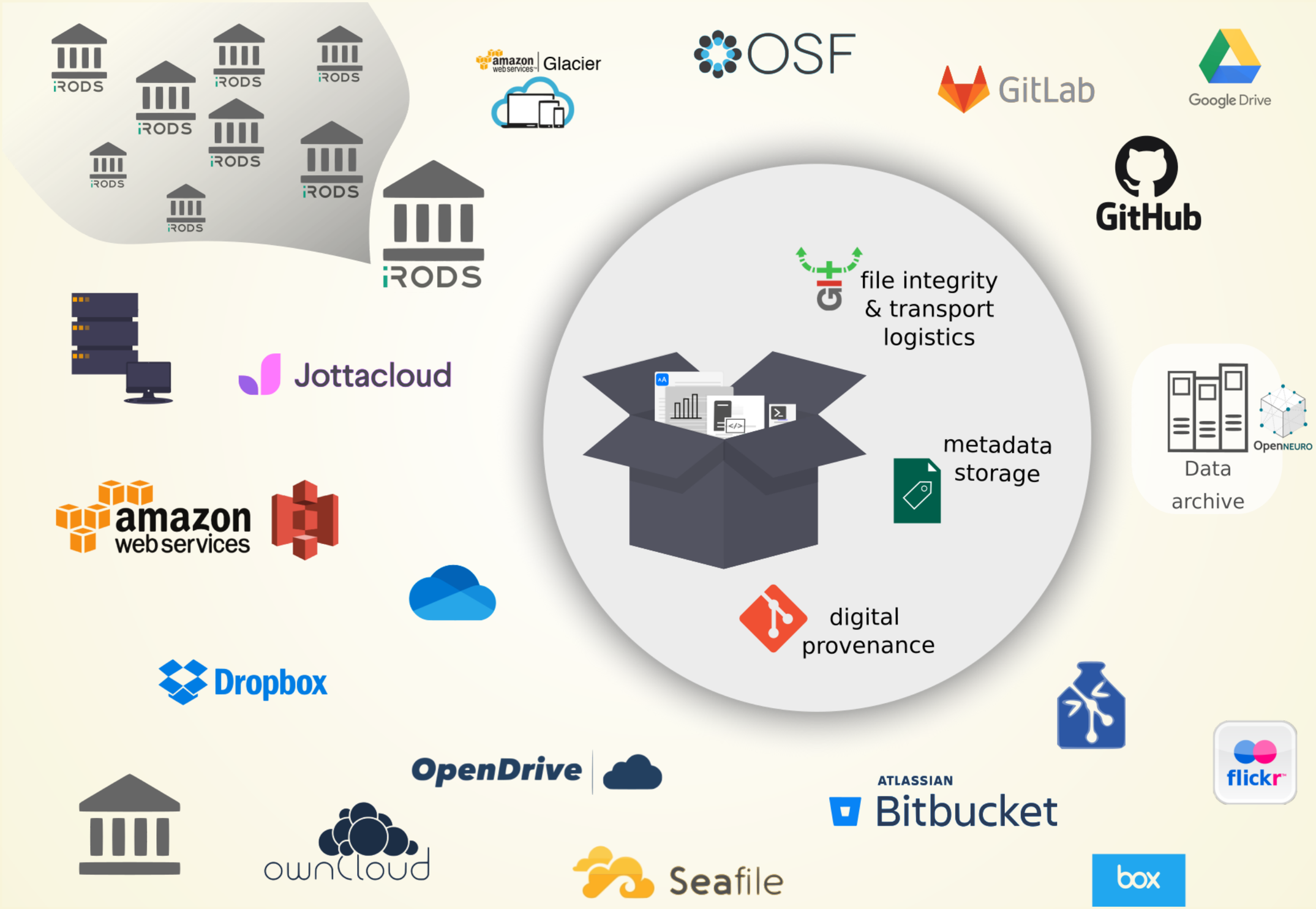


- make *the* difference for advertisment, discovery, convenience
- but imply gigantic dependencies
- often impossible to "take over"

Make sure data/metadata are self-contained
to facilitate/enable transition to another service

SECURITY AND RELIABILITY - FOR DATA

Decentral version control for data integrates with a variety of services to let you store data in different places - creating a resilient network for data



"In defense of decentralized Research Data Management", doi.org/10.1515/nf-2020-0037

COLLABORATION

Teamscience on more than code:

Q: Adjustment to tiny labeling mistake (ref Zemblys, 2018)? #2

Closed **adswa** opened this issue on Mar 8, 2019 · 1 comment

adswa commented on Mar 8, 2019 — Contributor

Zemblys et al., 2018, report a minor labeling mistake in one image file here (or a pay-wall free version here) in Appendix 2:

> We found an obvious labeling mistake in the one of the validation trials (file UH29_img_Europe_labelled_MN. We fixed this error by reassigning 75 samples, [3197,3272) (zero-based index), from the saccade to the fixation class.

I checked the data file in question and it appears to still contain the erroneous saccade labels. Just to reconfirm: this labeling error has not been fixed in the data file in this repository, correct?

If you wish, I can PR a fixed file, the issue at hand is intended to just reconfirm my assumption.

Thanks in advance!

adswa added a commit to adswa/remodnav that referenced this issue on Mar 8, 2019

`ENH/FIX: use file with fixed labels.` ··· `1b2b162`

```
2019-03-08 12:38 +0100 Richard Andersson M─┐  [master] {origin/master} {origin/HEAD} Merge pull request #3 from AdinaWagner/datafix
2019-03-08 11:35 +0100 Adina Wagner        │ o ENH/FIX: relabel erroneous saccades to fixations, closes #2.
2018-12-05 15:27 +0100 Richard Andersson o─┘ Uploaded a folder consting only of the data used in the original article
2017-08-22 19:40 +0200 Richard Andersson o Code added
2016-12-14 10:35 +0100 richardandersson  o Added currently shared data and stimuli.
2016-12-14 10:29 +0100 richardandersson  o Deleted -- to be reuploaded
2016-12-14 10:08 +0100 Richard Andersson o Add files via upload
2016-12-14 10:07 +0100 Richard Andersson o Add files via upload
2016-12-14 10:04 +0100 Richard Andersson I Initial commit
```

Fixed data

Original data

EXHAUSTIVE TRACKING OF RESEARCH COMPONENTS

text document
source code
binary file
regular file
file reference: identity and availability

Well-structured datasets (using community standards), and portable computational environments — and their evolution — are the precondition for reproducibility

```
# turn any directory into a dataset
# with version control


% datalad create <directory>
```

```
# save a new state of a dataset with
# file content of any size


% datalad save
```

CAPTURE COMPUTATIONAL PROVENANCE

Which data was needed at which version, as input into which code, running with what parameterization in which computional environment, to generate an outcome?

```
# execute any command and capture its output
# while recording all input versions too


% datalad run --input ... --output ... <command>
```

EXHAUSTIVE CAPTURE ENABLES PORTABILITY

Precise identification of data and computational environments combined with provenance records form a comprehensive and portable data structure, capturing all aspects of an investigation.

```
# transfer data and metadata to other sites and services
# with fine-grained access control for dataset components


% datalad push --to <site-or-service>
```

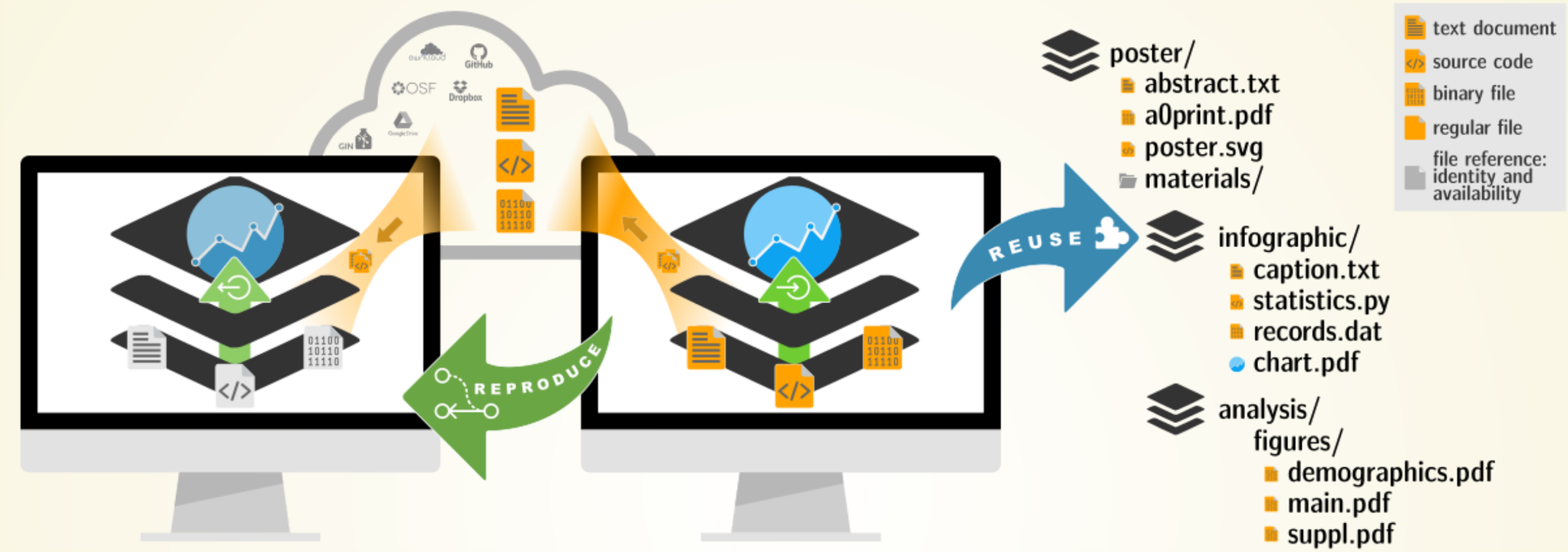
REPRODUCIBILITY STRENGTHENS TRUST

Outcomes of computational transformations can be validated by authorized 3rd-parties. This enables audits, promotes accountability, and streamlines automated "upgrades" of outputs

```
# obtain dataset (initially only identity,
# availability, and provenance metadata)


% datalad clone <url>
```

```
# immediately actionable provenance records
# full abstraction of input data retrieval


% datalad rerun <commit|tag|range>
```

ULTIMATE GOAL: (RE-)USABILITY



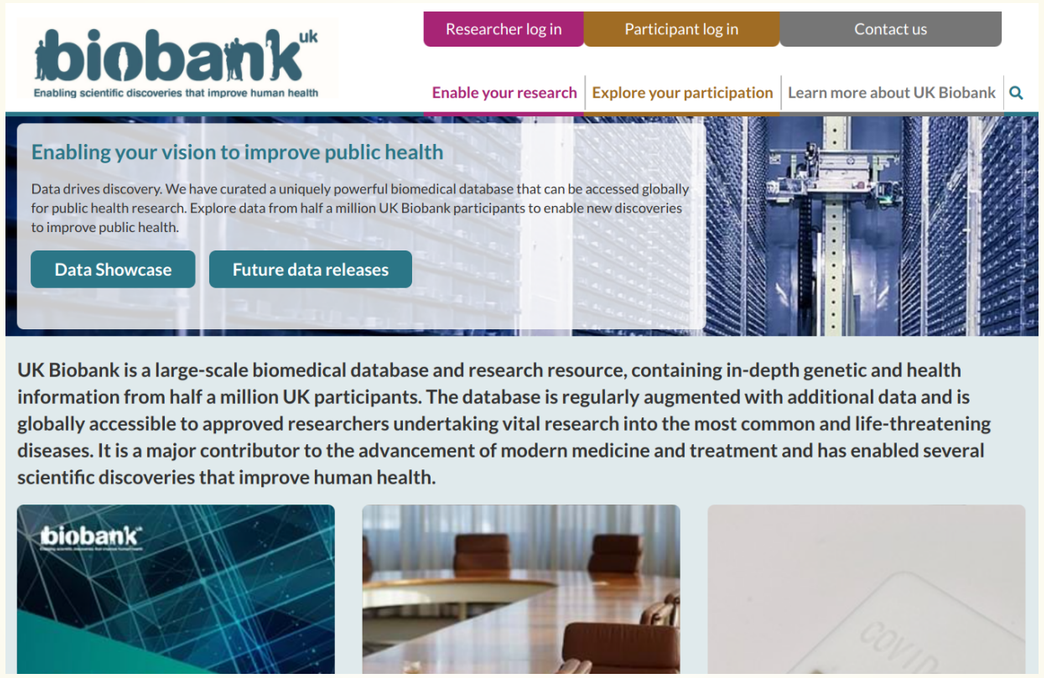
Verifiable, portable, self-contained data structures that track all aspects of an investigation exhaustively can be (re-)used as modular components in larger contexts — propagating their traits

```
# declare a dependency on another dataset and
# re-use it a particular state in a new context


% datalad clone -d <superdataset> <url> <path-in-dataset>
```

BIG DATA

FAIRLY BIG: SCALING UP

Objective: Process the UK Biobank (imaging data)

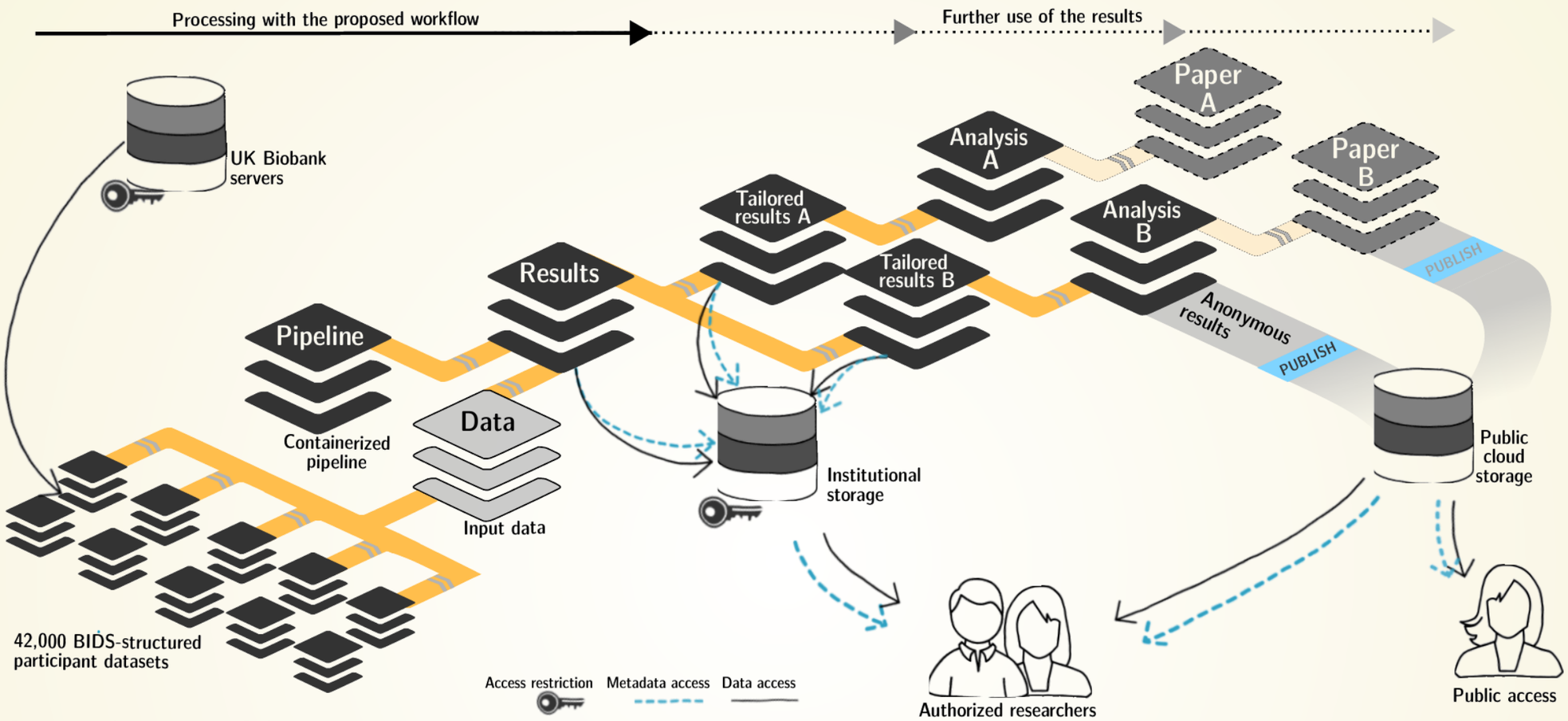


- 76 TB in 43 million files in total
- 42,715 participants contributed personal health data
- Strict DUA
- Custom binary-only downloader
- Most data records offered as (unversioned) ZIP files

CHALLENGES

- Process data such that
  - Results are computationally reproducible (without the original compute infrastructure)
  - There is complete linkage from results to an individual data record download
  - It scales with the amount of available compute resources
- Data processing pipeline
  - Compiled MATLAB blob
  - 1h processing time per image, with 41k images to process
  - 1.2 M output files (30 output files per input file)
  - 1.2 TB total size of outputs

FAIRLY BIG SETUP

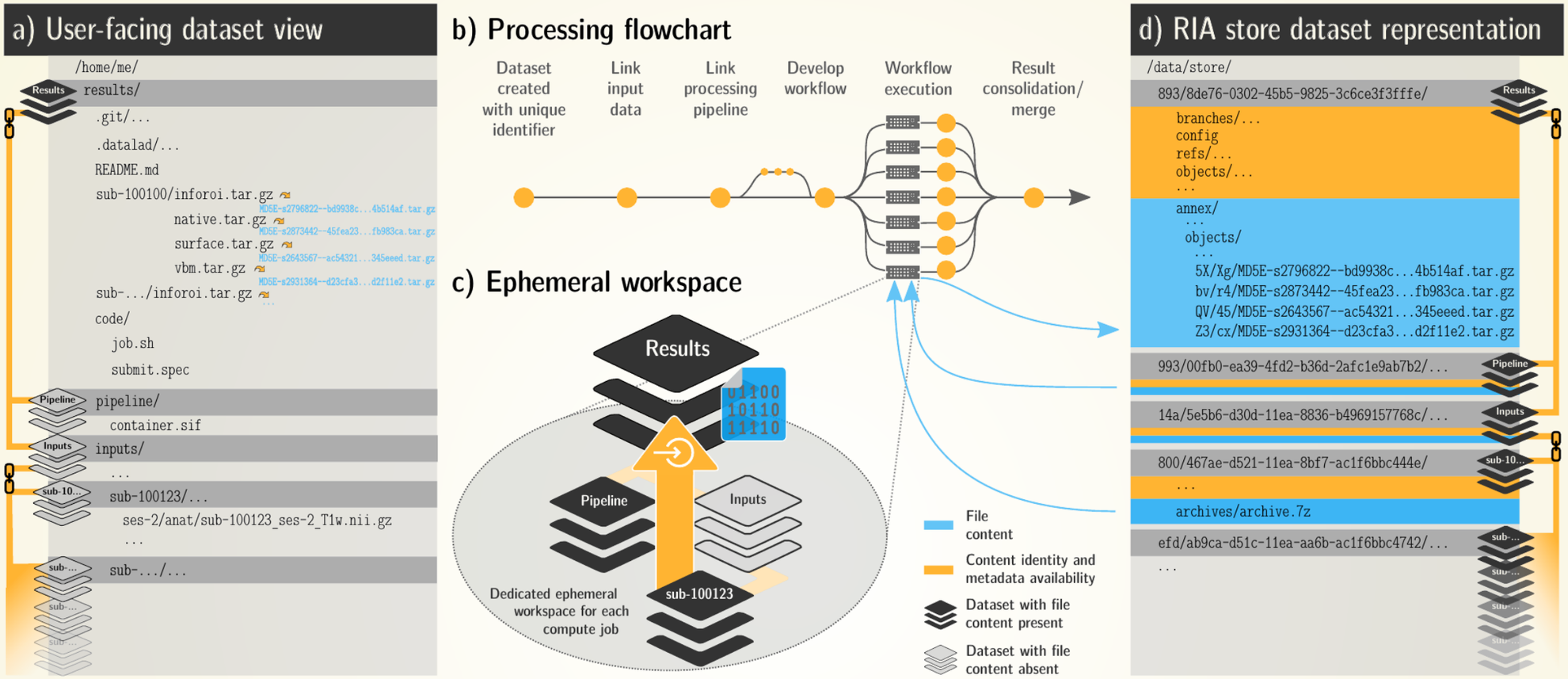


Exhaustive tracking

- datalad-ukbiobank extension downloads, transforms & track the evolution of the complete data release in DataLad datasets
- Native and BIDSified data layout (at no additional disk space usage)
- Structured in 42k individual datasets, combined to one superdataset
- Containerized pipeline in a software container
- Link input data & computational pipeline as dependencies

Wagner, Waite, Wierzba et al. (2021). FAIRly big: A framework for computationally reproducible processing of large-scale data.

FAIRLY BIG WORKFLOW



portability

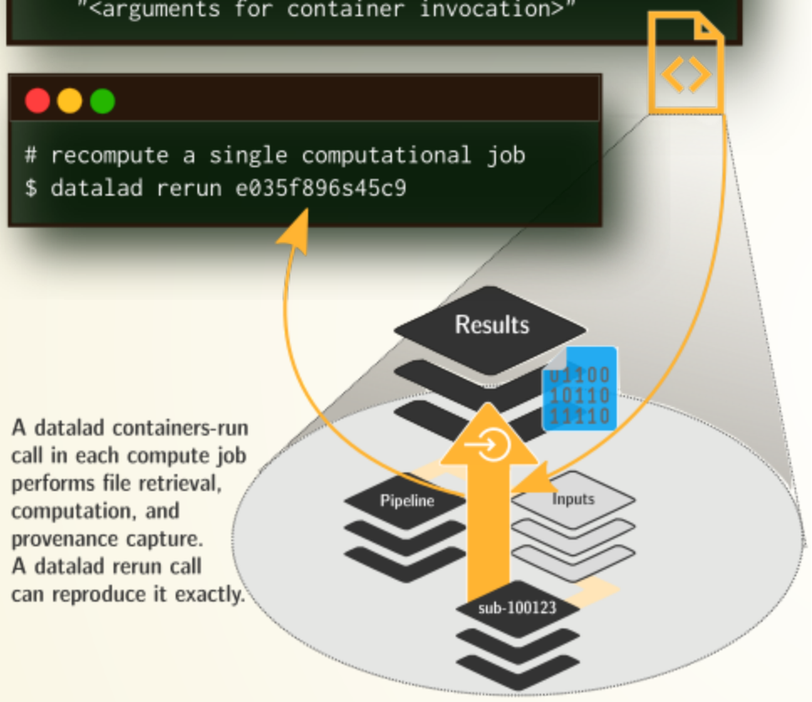
- Parallel processing: 1 job = 1 subject (number of concurrent jobs capped at the capacity of the compute cluster)
- Each job is computed in a ephemeral (short-lived) dataset clone, results are pushed back: Ensure exhaustive tracking & portability during computation
- Content-agnostic persistent (encrypted) storage (minimizing storage and inodes)
- Common data representation in secure environments

Wagner, Waite, Wierzba et al. (2021). FAIRly big: A framework for computationally reproducible processing of large-scale data.

FAIRLY BIG PROVENANCE CAPTURE

```
# perform and capture a computational execution
$ datalad containers-run \
    -m "Compute subject ${subid}" \
    -n cat \
    --input "inputs/${subid}/*T1w.nii.gz" \
    --output "${subid}" \
    "<arguments for container invocation>"
```

```
# recompute a single computational job
$ datalad rerun e035f896s45c9
```



A datalad containers-run call in each compute job performs file retrieval, computation, and provenance capture. A datalad rerun call can reproduce it exactly.

```
commit e035f896s45c9fac70cn7cc4dbd0dad43907755p
Author:     Jane Doe <j.doe@fz-juelich.de>
AuthorDate: Wed Feb 10 18:05:30 2021 +0100
Commit:     Jane Doe <j.doe@fz-juelich.de>
CommitDate: Wed Feb 10 18:05:30 2021 +0100

[DATALAD RUNCMD] Compute sub-6025043/ses-2

  === Do not change lines below ===
  {
   "chain": [],
   "cmd": "singularity exec -B {pwd} --cleanenv code/pipeline/.datalad/
           environments/cat/image sh -e -u -x -c  [...]'
   "dsid": "8938de76-0302-45b5-9825-3c6ce3f3fffe",
   "exit": 0,
   "extra_inputs": [
    "code/pipeline/.datalad/environments/cat/image"
   ],
   "inputs": [
    "inputs/ukb/sub-6025043/ses-2/anat/sub-6025043_ses-2_T1w.nii.gz",
    "code/cat_standalone_batch.txt",
    "code/finalize_job_outputs.sh"
   ],
   "outputs": [
    "sub-6025043/ses-2"
   ],
   "pwd": "."
  }
  ^^^ Do not change lines above ^^^
---
 sub-6025043/ses-2/inforoi.tar.gz | 1 +
 sub-6025043/ses-2/native.tar.gz  | 1 +
 sub-6025043/ses-2/surface.tar.gz | 1 +
 sub-6025043/ses-2/vbm.tar.gz     | 1 +
 4 files changed, 4 insertions(+)
```

Basic commit metadata

Author, Agent, Date, Time, and Commit Message

Transformations

Command call/
Container parametrization

Software container image

Origin: http://containers.ds.inm7.de/..
Version: dfa6d975ea8888ed33bf714c67

Input data

Origin: http://ukb.ds.inm7.de/.../bids
Version: 0c7f0b45140dde1d7291b1572

Expected output data/folder

Captured output data

Path, Content hash

Provenance

- Every single pipeline execution is tracked
- Execution in ephemeral workspaces ensures results individually reproducible without HPC access

Wagner, Waite, Wierzba et al. (2021). FAIRly big: A framework for computationally reproducible processing of large-scale data.

FAIRLY BIG MOVIE

Reproducible processing of 41,180 brain images from the UK Biobank with DataLad

- Two computations on clusters of different scale (small cluster, supercomputer). Full video: https://youtube.com/datalad
- Two full (re-)computations, programmatically comparable, verifiable, reproducible -- on any system with data access

TAKE HOME MESSAGES

Data deserves version control

It changes and evolves just like code

Science, especially on big data, relies on good data management

But effort pays off: Increased transparency, better reproducibility, easier accessibility, efficiency through automation and collaboration, streamlined procedures for synchronizing and updating your work, ...

DataLad can help with some things

Have access to more data than you have disk space

Who needs short-term memory when you can have automatic provenance capture?

Link versioned data to your analysis at no disk-space cost

...

HELP?!

If you have a question, you can reach out for help any time:

Reach out to to the DataLad team via

- Matrix (free, decentralized communication app, no app needed). We run a weekly Zoom office hour (Thursday, 4pm Berlin time) from this room as well.
- the development repository on GitHub (github.com/datalad/datalad)

Reach out to the user community with

- A question on neurostars.org with a `datalad` tag

Find more user tutorials or workshop recordings

- On DataLad's YouTube channel (www.youtube.com/channel/datalad)
- In the DataLad Handbook (handbook.datalad.org)
- In the DataLad RDM course (psychoinformatics-de.github.io/rdm-course)
- In the Official API documentation (docs.datalad.org)

ACKNOWLEDGEMENTS

Software

- Joey Hess (git-annex)
- The DataLad team & contributors

Illustrations

- The Turing Way project & Scriberia

Science

- Psychoinformatics Lab & INM-7
- Countless open scientists

Funders

EUROPEAN UNION
European Regional Development Fund

cbbs center for behavioral brain sciences

SACHSEN-ANHALT
Ministerium für Wissenschaft und Wirtschaft

Collaborators

Human Brain Project

CONP PCNO

VirtualBrainCloud

OpenNEURO

cbrain

brainlife.io

LET'S CLEAN UP

- Removing files from a version control system can be unintuitive and harder than expected
- Let's clean up!

DROP & REMOVE

- **datalad drop** removes annexed file contents from a local dataset annex and frees up disk space. It is the antagonist of **get** (which can get files and subdatasets).

```
$ datalad drop inputs/sub-02
drop(ok): input/sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz (file)
drop(ok): input/sub-02 (directory)
action summary:
  drop (ok: 2)
```

- But: Default safety checks require that dropped files can be re-obtained to prevent accidental data loss. **git annex whereis** reports all registered locations of a file's content
- **drop** does not only operate on individual annexed files, but also directories, or globs, and it can uninstall subdatasets:

```
$ datalad drop --what all input
uninstall(ok): input (dataset)
```

DROP & REMOVE

- **datalad remove** removes complete dataset or dataset hierarchies and leaves no trace of them. It is the antagonist to **clone**.

```
# The command operates outside of the to-be-removed dataset!
$ datalad remove inputs
uninstall(ok):  inputs (dataset)
```

- But: Default safety checks require that it could be re-cloned in its most recent version from other places, i.e., that there is a *sibling* that has all revisions that exist locally **datalad siblings** reports all registered siblings of a dataset.

DROP & REMOVE

- **datalad drop** refuses to remove annexed file contents if it can't verify that **datalad get** could re-retrieve it

```
$ datalad drop figures/sub-02_mean-epi.png
drop(error): figures/sub-02_mean-epi.png (file) [unsafe; Could only verify the existence of 0 out of 1 necessa
                                                 copy; (Use --reckless availability to override this check, or
                                                 adjust numcopies.)]
```

- Adding **--reckless availability** overrides this check

```
$ datalad drop figures/sub-02_mean-epi.png --reckless availability
```

- Be mindful that **drop** will only operate on the most recent version of a file - past versions may still exist afterwards unless you drop them specifically. **git annex unused** can identify all files that are left behind

DROP & REMOVE

- **datalad remove** refuses to remove datasets without an up-to-date *sibling*

```
$ datalad remove -d my-analysis
uninstall(error): . (dataset) [to-be-dropped dataset has revisions that are not available at a
                  sibling. Use `datalad push --to ...` to push these before dropping the loca
                  or ignore via `--reckless availability`. Unique revisions: ['main']]
```

- Adding **--reckless availability** overrides this check

```
$ datalad remove -d my-analysis --reckless availability
```

REMOVING WRONGLY

- Removing datasets the wrong way causes chaos and leaves an usuable dataset corpse behind:

```
$ rm -rf local-dataset
rm: cannot remove 'local-dataset/.git/annex/objects/Kj/44/MD5E-s42--8f008874ab52d0ff02a5bbd0174ac95e.txt/
MD5E-s42--8f008874ab52d0ff02a5bbd0174ac95e.txt': Permission denied
```

- The dataset can't be fixed, but to remove the corpse **chmod** (change file mode bits) it (i.e., make it writable)

```
$ chmod +w -R local-dataset
$ rm -rf local-dataset
```