

RESEARCH DATA MANAGEMENT



WITH DATALAD

Adina Wagner

 mas.to/@adswa

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

Slides: files.inm7.de/adina/talks/html/sfb-1280.html
Sources: <https://github.com/datalad-handbook/datalad-course>

WELCOME & LOGISTICS!

- A approximate schedule for today:
 - 1.00 pm: Introduction & Logistics
 - 1.30 pm: Overview of DataLad + break ☕
 - 2.00 pm: What's version control, and why should I care?
 - 2:45 pm: Reproducibility features + break
 - 3.30 pm: Data publication to the OSF + break ☕
 - 4.30 pm: Outlook and/or Your Questions and Usecases
- Collaborative notes & anonymous questions:
etherpad.wikimedia.org/p/Datalad@sfb1280.
- Slides are CC-BY and will be shared after the workshop. Additional workshop contents: psychoinformatics-de.github.io/rdm-course
- Some guidelines for the virtual workshop venue...
 - Please mute yourself when you don't speak
 - Ask questions anytime, but make use of the "Raise hand" feature
 - Drop out and re-join as you please

QUESTIONS/INTERACTION THROUGHOUT THE WORKSHOP

- There are no stupid questions :)
- Lively discussions are wonderful - unless its interrupting others, please feel encouraged to unmute/turn on your video to interact.
- There is room discuss specific or advanced use cases at the end. Please make a note about them in the [Etherpad](#).

QUESTIONS/INTERACTION AFTER THE WORKSHOP

If you have a question after the workshop, you can reach out for help:

Reach out to to the DataLad team via

- [Matrix](#) (free, decentralized communication app, no app needed). We run a weekly Zoom office hour (Tuesday, 4pm Berlin time) from this room as well.
- [the development repository on GitHub](#)

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](#)
- In the [DataLad Handbook](#)
- In the [DataLad RDM course](#)
- In the [Official API documentation](#)

RESOURCES AND FURTHER READING

Comprehensive user documentation in the DataLad Handbook (handbook.datalad.org)

- High-level function/command overviews, Installation, Configuration, Cheatsheet
-

- Narrative-based code-along course
 - Independent on background/skill level, suitable for data management novices
-

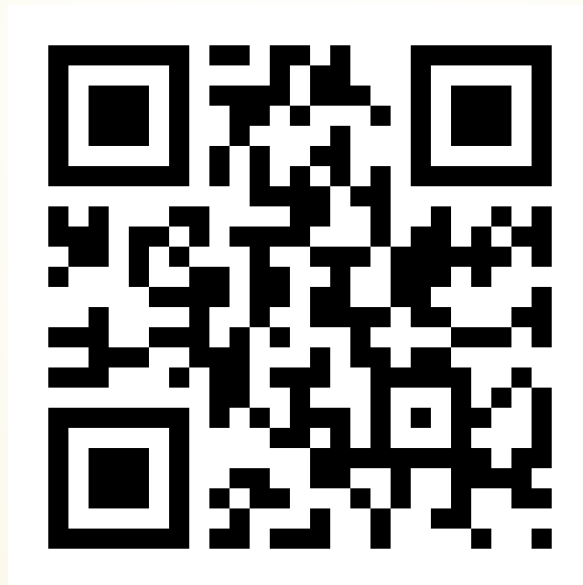
- Step-by-step solutions to common data management problems, like how to make a reproducible paper

Overview of most tutorials, talks, videos, ... at github.com/datalad/tutorials

LIVE POLLING SYSTEM

Please use your phone to scan to QR code, or open the link in a new browser window

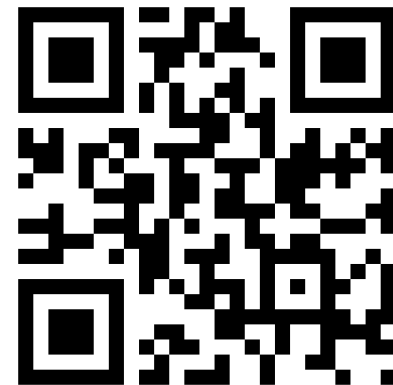
<http://etc.ch/yNtn>



WHAT'S YOUR MOOD TODAY?



<http://etc.ch>



PRACTICAL ASPECTS



- We'll work in the browser on a cloud server with JupyterHub
- Cloud-computing environment:
 - datalad-hub.inm7.de
- We have pre-installed DataLad and other requirements
- We will work via the terminal
- Your username is all lower-case and follows this pattern: Firstname + Lastname initial (Adina Wagner -> adinaw)
- Pick any password with at least 8 characters at first log-in (and remember it)

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.

API	<code>datalad [--GLOBAL-OPTION <opt. flag spec.>] COMMAND [ARGUMENTS] [--OPTION <opt. flag spec.>]</code>	Each datalad invocation can have two sets of options: general options are given first, command-specific ones go after the subcommand.
GLOBAL OPTIONS	<code>-c KEY=VALUE</code> Set config variables (overrides configurations in files) <code>-f/--output-format default json json_pp tailored</code> Specify the format for command result rendering <code>-l/--log-level critical error warning info debug</code> Set logging verbosity level	
COMMAND OPTIONS	<code>-d/--dataset</code> Dataset location: path to root, or ^ for superdataset <code>-D/--description</code> A location description (e.g., "my backup server") <code>-f/--force</code> Force execution of a command (Dangerzone!) <code>-m/--message</code> A description about a change made to the dataset <code>-r/--recursive</code> Perform an operation recursively across subdatasets <code>-R/--recursion-limit <n></code> Limit recursion to n subdataset levels	

- 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.
```

USING DATALAD IN THE TERMINAL

Check the installed version:

```
datalad --version
```

copy

For help on using DataLad from the command line (press q to exit):

```
datalad --help
```

copy

For extensive info about the installed package, its dependencies, and extensions, use `datalad wtf`. Let's find out what kind of system we're on:

```
datalad wtf -S system
```

copy

GIT IDENTITY

Check git identity:

```
git config --get user.name  
git config --get user.email
```

copy

Configure git identity:

```
git config --global user.name "Adina Wagner"  
git config --global user.email "adina.wagner@t-online.de"
```

copy

Use the latest datalad features:

```
git config --global --add datalad.extensions.load next
```

copy

USING DATALAD VIA ITS PYTHON API

Open a Python environment:

```
ipython
```

copy

Import and start using:

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

copy

Exit the Python environment:

```
exit
```

copy

DIFFERENT WAYS TO USE 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")
```

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

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

- ... or via a graphical user interface "DataLad Gooney"

ACKNOWLEDGEMENTS

Funders



Software

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

Illustrations

- The Turing Way project & Scriberia



Collaborators

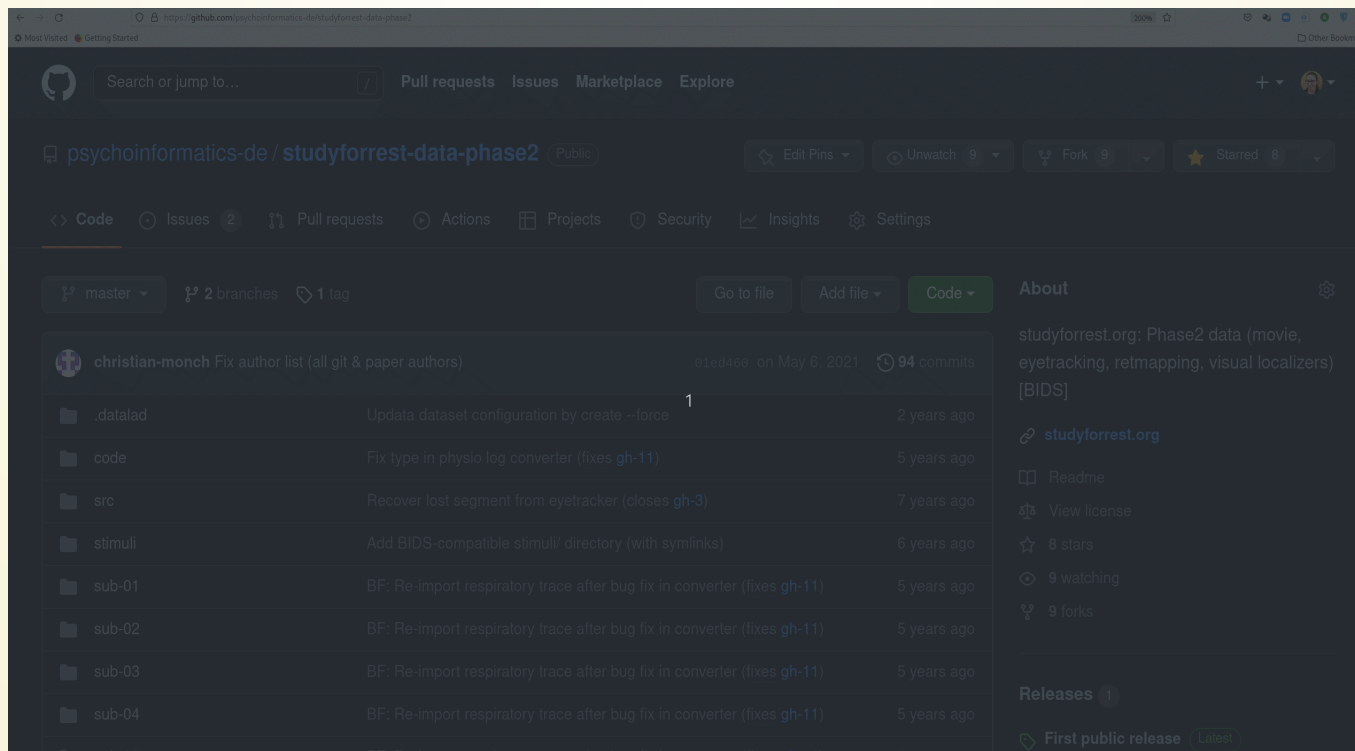


DataLad CORE FEATURES:

- **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, update, 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

EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

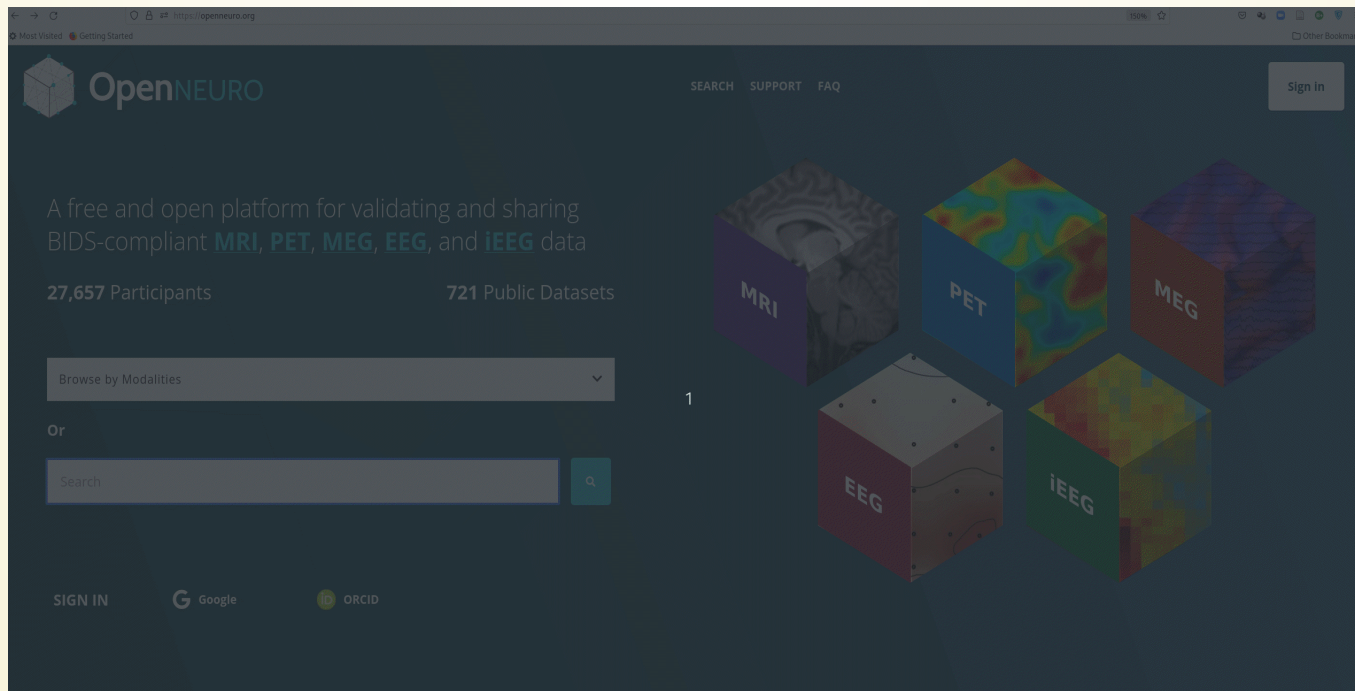
- Publish or consume datasets via GitHub, GitLab, OSF, the European Open Science Cloud, or similar services



The screenshot shows a GitHub repository page for 'psychoinformatics-de / studyforrest-data-phase2'. The repository is public and has 9 forks and 8 stars. The main content area displays a list of files and folders, including .datalad, code, src, stimuli, and sub-01 through sub-04. The .datalad folder is highlighted, showing its description: 'Update dataset configuration by create -force'. The code folder is described as 'Fix type in physio log converter (fixes gh-11)'. The src folder is described as 'Recover lost segment from eyetracker (closes gh-3)'. The stimuli folder is described as 'Add BIDS-compatible stimuli directory (with symlinks)'. The sub-01 through sub-04 folders are described as 'BF: Re-import respiratory trace after bug fix in converter (fixes gh-11)'. The right sidebar shows the repository's description: 'studyforrest.org: Phase2 data (movie, eyetracking, retmapping, visual localizers) [BIDS]'. It also includes links to the studyforrest.org website, a README file, and a license. The repository has 8 stars, 9 watchers, and 9 forks. The 'Releases' section shows the first public release, which is the latest.

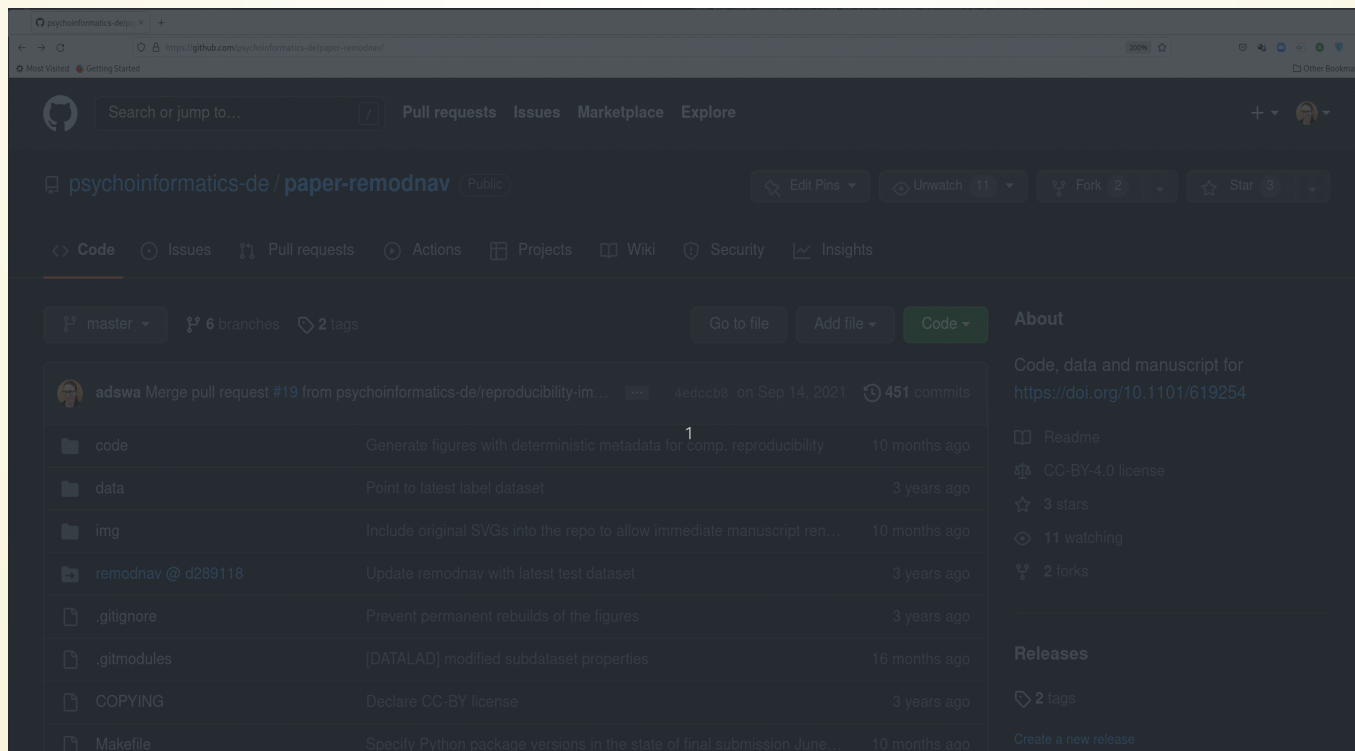
EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

- 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



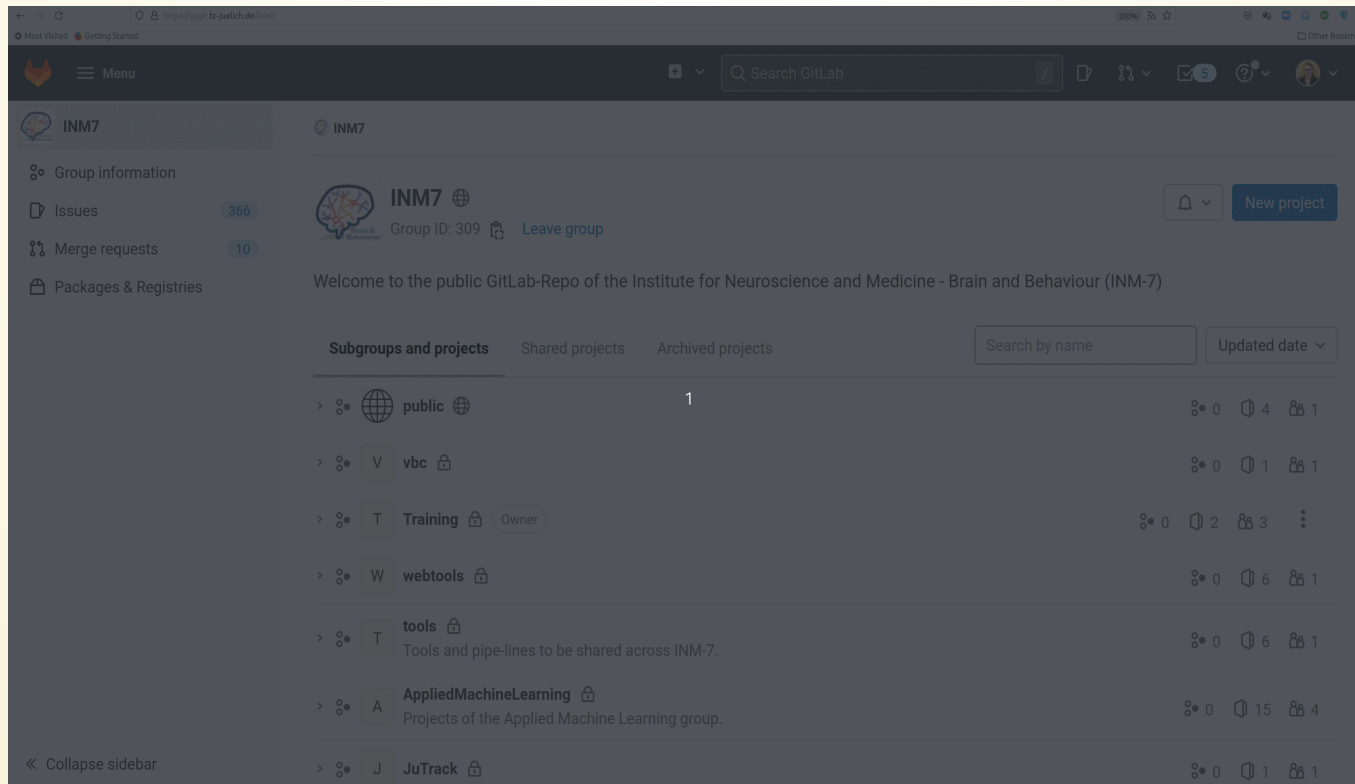
EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

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

The screenshot shows a Twitter thread from Lennart Wittkuhn (@lennartwittkuhn) dated 19. März. The main tweet discusses a new fMRI analysis method and includes a link to a paper on nature.com. A reply from the same user mentions sharing code and data via @gnode and #GitHub, and lists various data-related terms like #BIDSstandard, #fMRIPrep, and #MRIQC. The interface also shows engagement metrics (4 replies, 93 retweets, 270 likes) and a sidebar with 'Relevante Personen' including Lennart Wittkuhn, INCF G-Node, and DataLad, along with 'Trends für dich'.

EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

- Central data management and archival system



EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

- Scalable computing framework for reproducible science

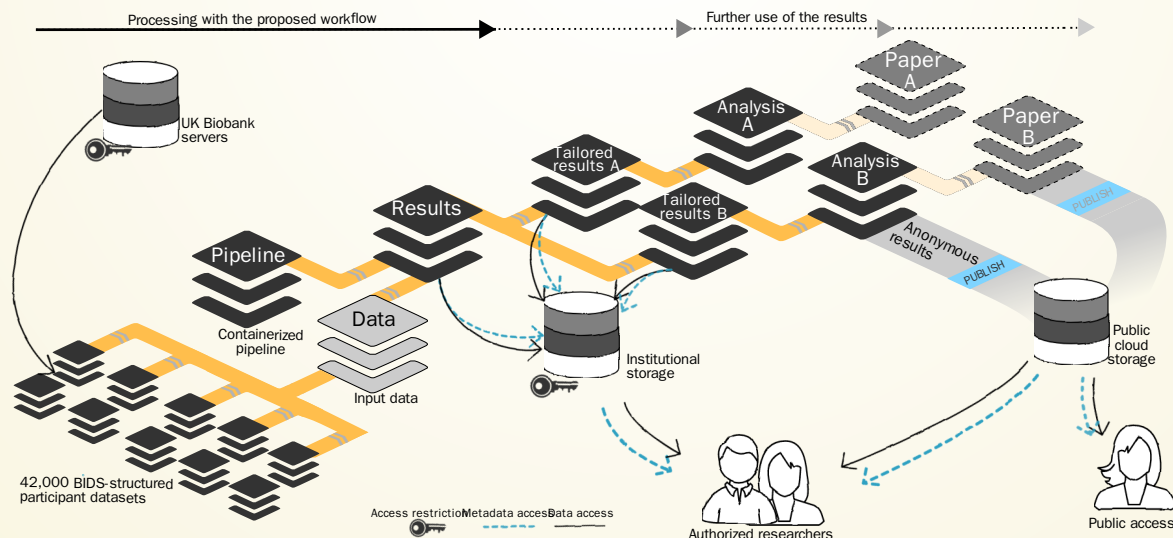
Article | [Open Access](#) | [Published: 11 March 2022](#)

FAIRly big: A framework for computationally reproducible processing of large-scale data

[Adina S. Wagner](#) , [Laura K. Waite](#), [Małgorzata Wierzba](#), [Felix Hoffstaedter](#), [Alexander Q. Waite](#), [Benjamin Poldrack](#), [Simon B. Eickhoff](#) & [Michael Hanke](#)

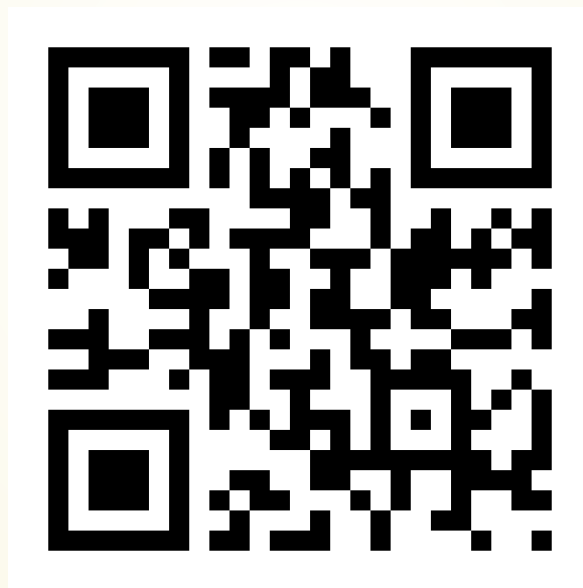
[Scientific Data](#) 9, Article number: 80 (2022) | [Cite this article](#)

813 Accesses | 23 Altmetric | [Metrics](#)

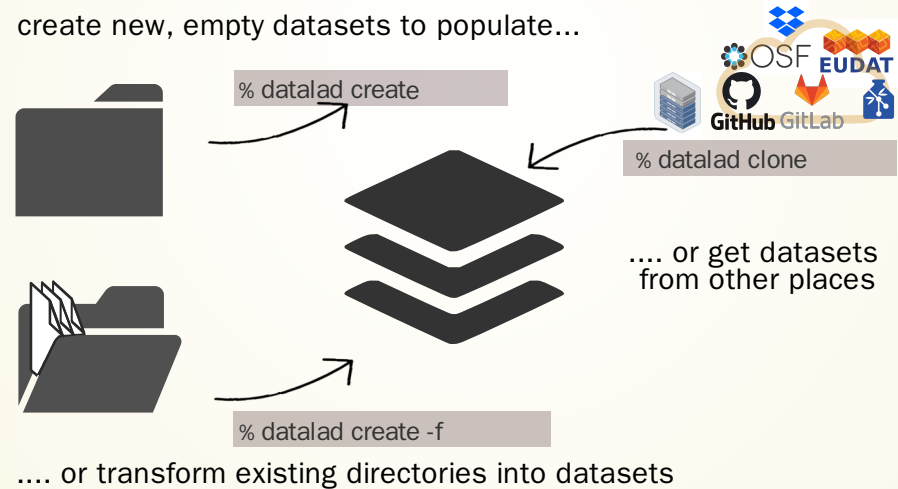


WHAT'S VERSION CONTROL, AND WHY SHOULD I CARE?

<http://etc.ch/yNtn>



EVERYTHING HAPPENS IN DATALAD DATASETS



File viewer

...DATALAD DATASETS

Create a dataset (here, with the `text2git` configuration, which adds a helpful configuration):

```
datalad create -c text2git my-analysis
```

copy

Let's have a look inside. Navigate using `cd` (change directory):

```
cd my-analysis
```

copy

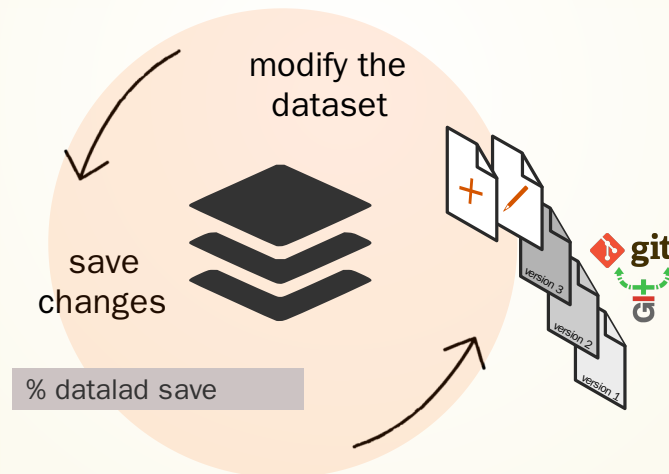
List the directory content, including hidden files, with `ls`:

```
ls -la .
```

copy

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)

...VERSION CONTROL

Let's build a dataset for an analysis by adding a README. The command below writes a simple header into a new file README.md:

```
echo "# My example DataLad dataset" > README.md
```

copy

Now we can check the `status` of the dataset:

```
datalad status
```

copy

We can save the state with `save`

```
datalad save -m "Create a short README"
```

copy

Further modifications:

```
echo "This dataset contains a toy data analysis" >> README.md
```

copy

You can also checkout what has changed:

```
git diff
```

copy

Save again:

```
datalad save -m "Add information on the dataset contents to the README"
```

copy

...VERSION CONTROL

Now, let's check the dataset history:

```
git log
```

copy

We can also make the history prettier:

```
tig
```

```
(navigate with arrow keys and enter, press "q" to go back and exit the program)
```

copy

EXHAUSTIVE TRACKING

The building blocks of a scientific result are rarely static

Analysis code evolves

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

Image credit: Based on Piled Higher and Deeper, 1999

EXHAUSTIVE TRACKING

Image credit: Piled Higher and Deeper 1323

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?"

Image credit: CC-BY Scriberia and The Turing Way



EXHAUSTIVE TRACKING

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.

```
2020-03-13 10:46 +0100 Adina Wagner o [DATALAD RUNCMD] add non-defaced
2020-03-13 10:29 +0100 Adina Wagner o [DATALAD RUNCMD] reconvert DICOM
2018-05-11 09:23 +0200 Michael Hanke o [master] {origin/HEAD} {origin/m
2018-05-11 09:19 +0200 Michael Hanke o Enable DataLad metadata extracto
2018-05-11 09:17 +0200 Michael Hanke o [DATALAD] new dataset
2018-05-11 09:17 +0200 Michael Hanke o [DATALAD] Set default backend fo
2018-01-19 14:19 +0100 Michael Hanke o <v1.5> Update changelog for 1.5
2018-01-19 14:09 +0100 Michael Hanke o BF: Re-import respiratory trace
2018-01-14 18:59 +0100 Michael Hanke o Fix type in physio log converter
2017-01-10 10:10 +0100 Michael Hanke o ENH: Report per-stimulus events
2016-12-10 20:18 +0100 Michael Hanke o Add BIDS-compatible stimuli/ dir
2016-11-15 07:04 +0100 Michael Hanke o Minor tweaks to gaze overlay scr
2016-10-30 11:03 +0100 Michael Hanke o Add "TaskName" meta data field f
2016-09-21 08:33 +0200 Michael Hanke o Add task-*_physio.json files
2016-09-21 08:23 +0200 Michael Hanke o BF: Fix task label in file names
2016-08-04 13:14 +0200 Michael Hanke o Update changelog
2016-08-03 22:22 +0200 Michael Hanke o Add cut position information to
2016-05-27 17:35 +0200 Michael Hanke o {origin/_} Mention openfmri as d
2016-04-04 09:31 +0200 Michael Hanke o Update publication links
2016-03-31 11:26 +0200 Michael Hanke o Disable invalid test
[main] 6da25fb6fee2c698d35f52066698b6f94850f4d2 - commit 10 of 79 27%
commit 6da25fb6fee2c698d35f52066698b6f94850f4d2
Refs: v1.0-19-g6da25fb6
Author: Michael Hanke <michael.hanke@gmail.com>
AuthorDate: Fri Jan 19 14:09:53 2018 +0100
Commit: Michael Hanke <michael.hanke@gmail.com>
CommitDate: Fri Jan 19 14:11:23 2018 +0100
BF: Re-import respiratory trace after bug fix in converter (fixes gh-
---
...er task-movielocalizer_run-1_recording-cardresp_physio.tsv.gz | 2 ++
..._task-objectcategories_run-1_recording-cardresp_physio.tsv.gz | 2 ++
..._task-objectcategories_run-2_recording-cardresp_physio.tsv.gz | 2 ++
..._task-objectcategories_run-3_recording-cardresp_physio.tsv.gz | 2 ++
..._task-objectcategories_run-4_recording-cardresp_physio.tsv.gz | 2 ++
...calizer_task-retmapccw_run-1_recording-cardresp_physio.tsv.gz | 2 ++
...calizer_task-retmapclw_run-1_recording-cardresp_physio.tsv.gz | 2 ++
...calizer_task-retmapcon_run-1_recording-cardresp_physio.tsv.gz | 2 ++
...calizer_task-retmapexp_run-1_recording-cardresp_physio.tsv.gz | 2 ++
..._2_ses-movie_task-movie_run-1_recording-cardresp_physio.tsv.gz | 2 ++
..._2_ses-movie_task-movie_run-2_recording-cardresp_physio.tsv.gz | 2 ++
[diff] 6da25fb6fee2c698d35f52066698b6f94850f4d2 - Line 1 of 2391 0%
```

EXHAUSTIVE TRACKING

With the `datalad-container` extension, we can not only add code or data, but also software containers to datasets and work with them. Let's add a software container with Python software for later:

```
datalad containers-add nilearn \  
  --url shub://adswa/nilearn-container:latest
```

copy

inspect the list of registered containers:

```
datalad containers-list
```

copy

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?

DIGITAL PROVENANCE

Imagine that you are getting a script from a colleague to perform your analysis, but they email it to you or upload it to a random place for to download:

```
wget -P code/ \
https://raw.githubusercontent.com/datalad-handbook/resources/master/get_brainmask.py
```

The `wget` command downloaded a script for extracting a brain mask:

```
datalad status
```

Save it into your dataset to have the script ready:

```
datalad save -m "Adding a nilearn-based script for brain masking"
```

Convenience functions make downloads easier. Let's add a nilearn tutorial, and also register the original location of this file as digital provenance:

```
datalad download-url -m "Add a tutorial on nilearn" \
-O code/nilearn-tutorial.pdf \
https://raw.githubusercontent.com/datalad-handbook/resources/master/nilearn-tutorial
```

Notice how its automatically saved:

```
datalad status
```

Check out the file's history:

```
git log code/nilearn-tutorial.pdf
```

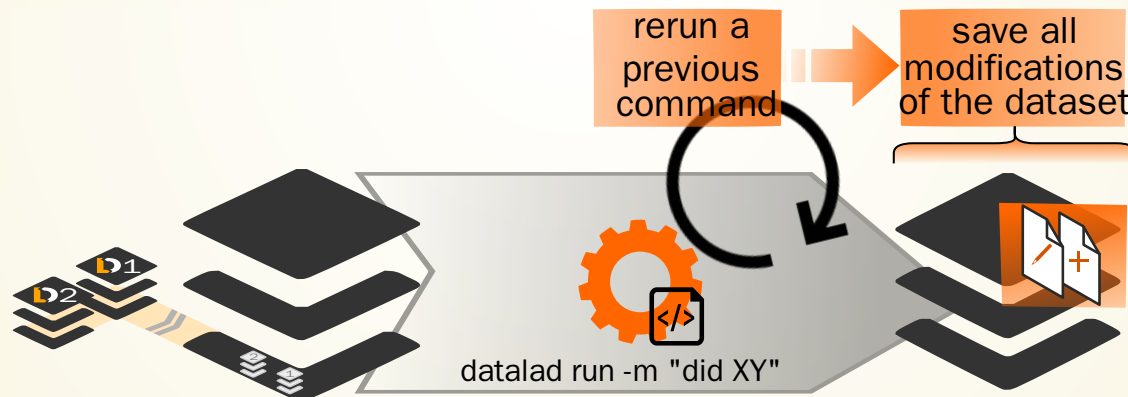
PROVENANCE AND REPRODUCIBILITY

`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.



... COMPUTATIONALLY REPRODUCIBLE EXECUTION I

A variety of processes can modify files. A simple example: Code formatting

```
black code/get_brainmask.py
```

copy

Version control makes changes transparent:

```
git diff
```

copy

But its useful to keep track beyond that. Let's discard the latest changes...

```
git restore code/get_brainmask.py
```

copy

... and record precisely what we did

```
datalad run -m "Reformat code with black" \  
"black code/get_brainmask.py"
```

copy

let's take a look (press q to exit):

```
git show
```

copy

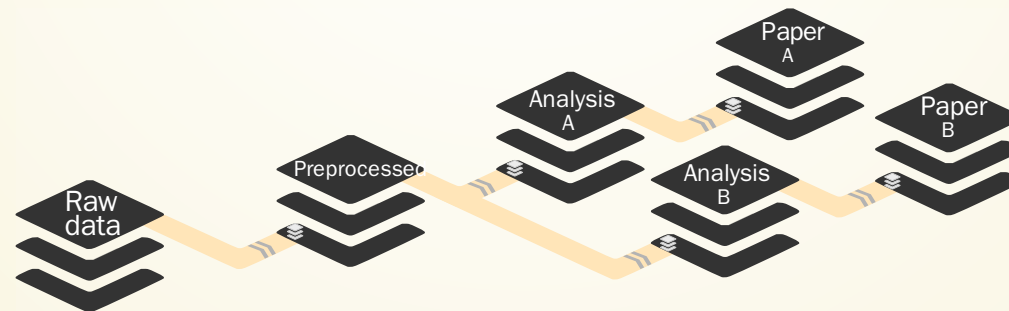
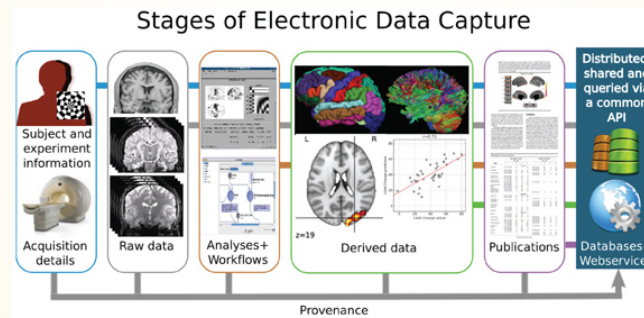
... and repeat!

```
datalad rerun
```

copy

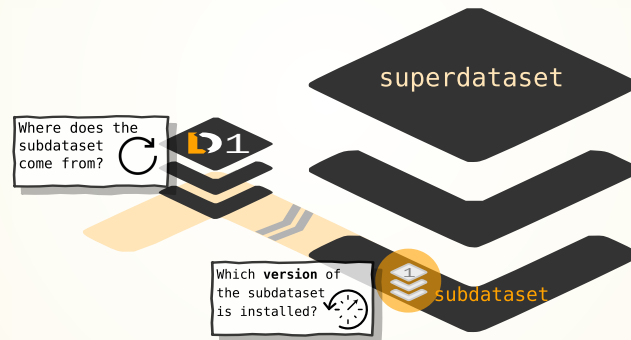
SEAMLESS DATASET NESTING & LINKAGE

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
```

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
+  datalad-id = 68bdb3f3-eafa-4a48-bddd-31e94e8b8242
+  datalad-url = http://example.com/importantds
diff --git a/inputs/rawdata b/inputs/rawdata
```

copy

...DATASET NESTING

Let's make a nest!

Clone a dataset with analysis data into a specific location ("input/") in the existing dataset, making it a *subdataset*:

```
datalad clone -d . \  
https://gin.g-node.org/adswa/bids-data \  
input
```

copy

Let's see what changed in the dataset, using the `subdatasets` command:

```
datalad subdatasets
```

copy

... and also `git show`:

```
git show
```

copy

We can now view the cloned dataset's file tree:

```
cd input  
ls
```

copy

...and also its history

```
tig
```

copy

Let's check the dataset size (with the `du` disk-usage command):

```
du -sh
```

copy

Let's check the *actual* dataset size:

```
datalad status --annex
```

copy

You can `get` or `drop` annexed file contents depending on your needs:

```
datalad get sub-02
```

copy

```
datalad drop sub-02
```

copy

...COMPUTATIONALLY REPRODUCIBLE EXECUTION...

Try to execute the downloaded analysis script. Does it work?

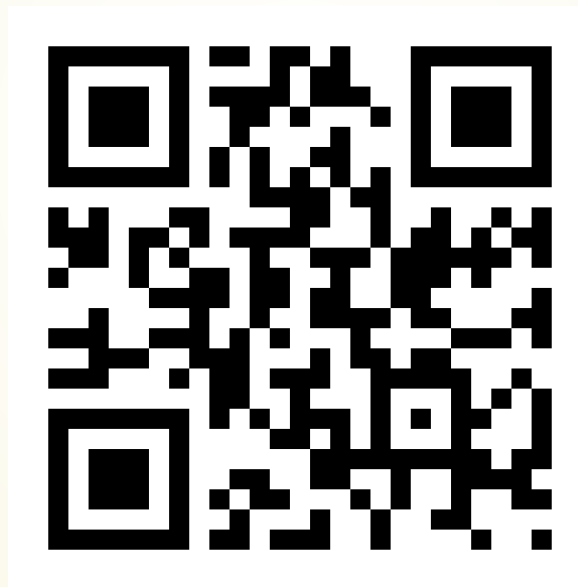
```
cd ..  
datalad run -m "Compute brain mask" \  
  --input input/sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz \  
  --output "figures/*" \  
  --output "sub-02*" \  
  "python code/get_brainmask.py"
```

copy

- Software can be difficult or impossible to install (e.g. conflicts with existing software, or on HPC) for you or your collaborators
- Different software versions/operating systems can produce different results: [Glatard et al., doi.org/10.3389/fninf.2015.00012](https://doi.org/10.3389/fninf.2015.00012)
- **Software containers** encapsulate a software environment and isolate it from a surrounding operating system. Two common solutions: Docker, Singularity

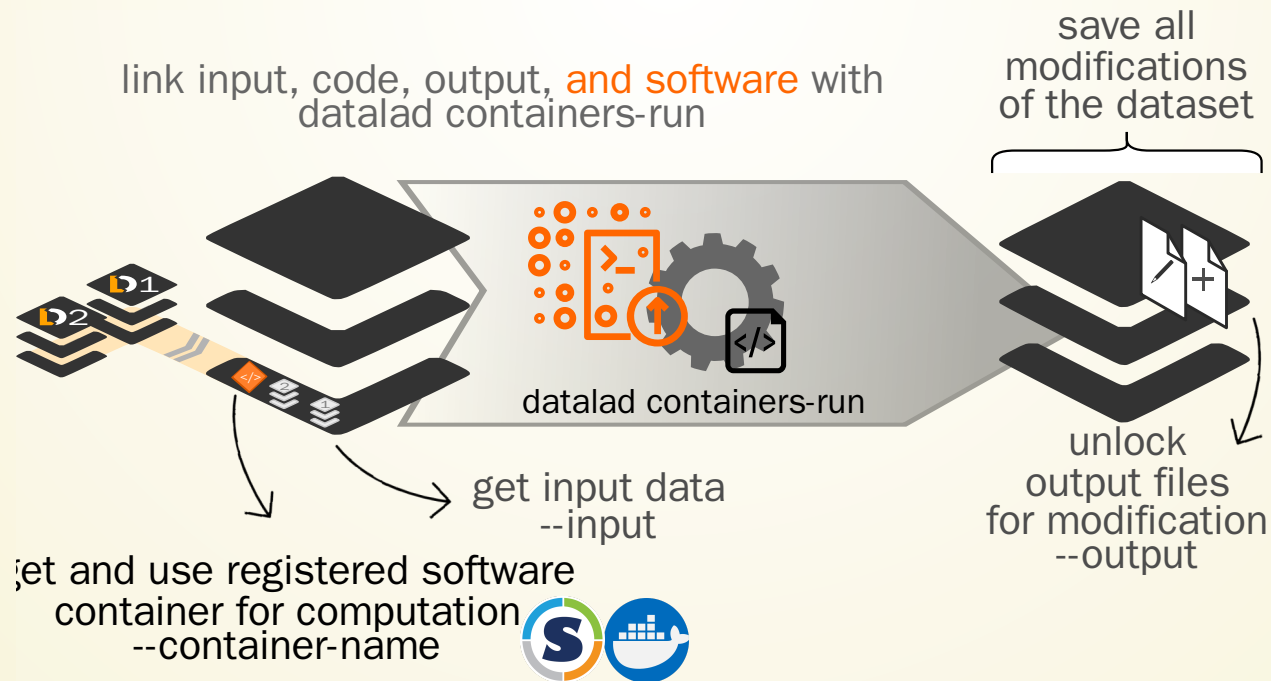
SOFTWARE CONTAINERS

<http://etc.ch/yNtn>



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



...COMPUTATIONALLY REPRODUCIBLE EXECUTION

Let's try out the `containers-run` command:

```
datalad containers-run -m "Compute brain mask" \  
-n nilearn \  
--input input/sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz \  
--output "figures/*" \  
--output "sub-02*" \  
"python code/get_brainmask.py" copy
```

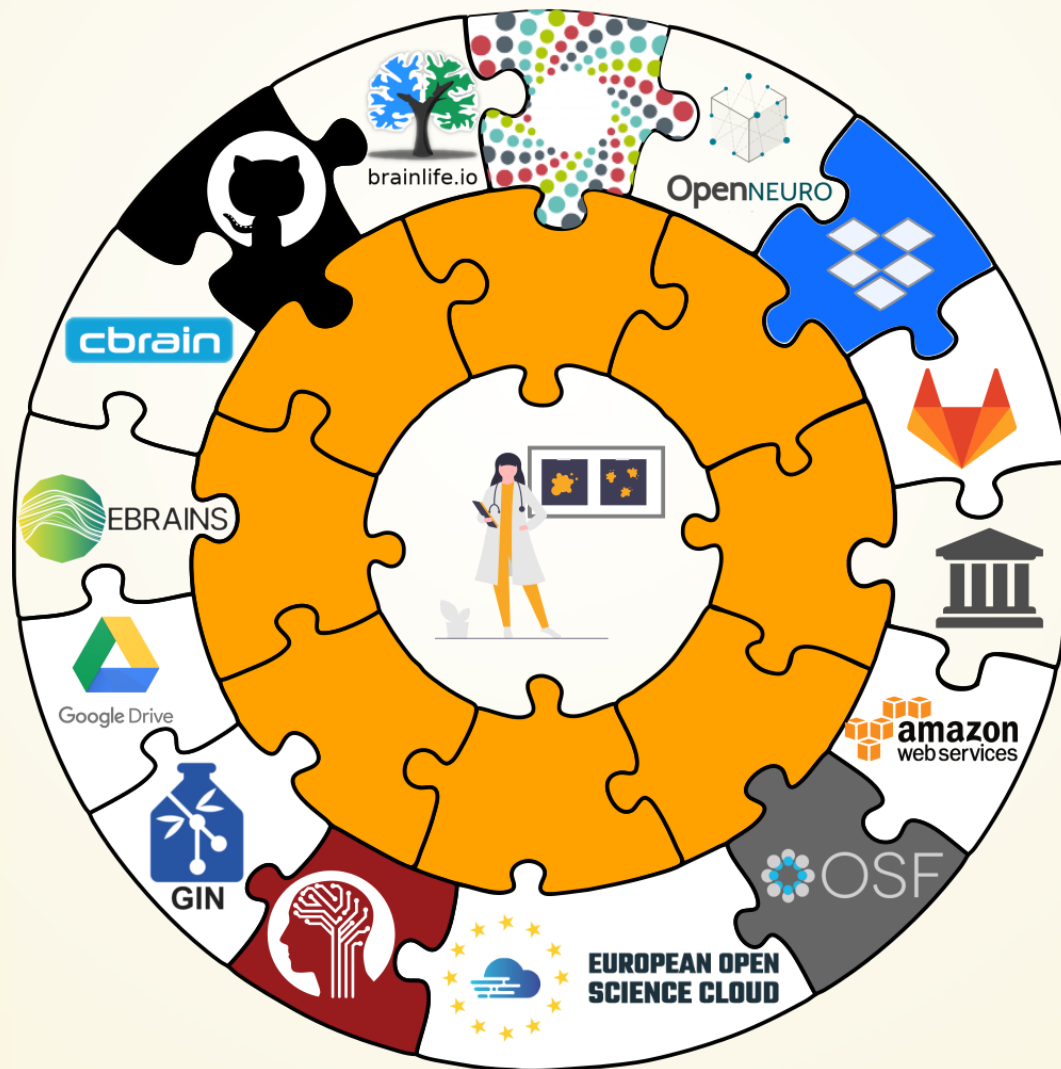
You can now query an individual file how it came to be...

```
git log sub-02_brain-mask.nii.gz copy
```

... and the computation can be redone automatically and checked for computational reproducibility based on the recorded provenance using `datalad rerun`:

```
datalad rerun copy
```

SHARING DATASETS

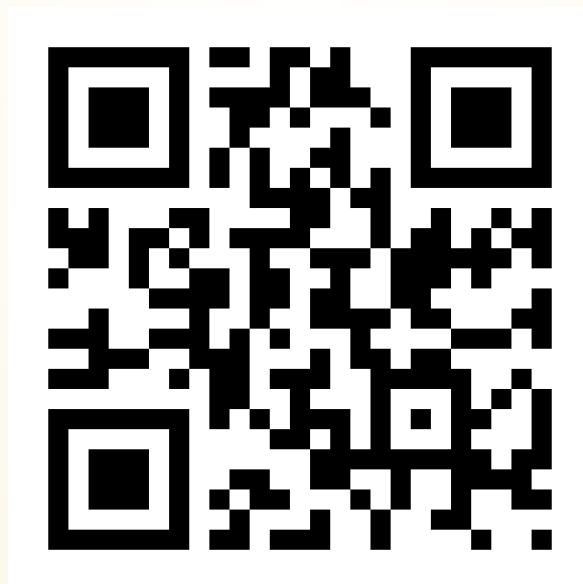


Apart from local computing infrastructure (from private laptops to computational clusters), datasets can be hosted in

SHARING DATASETS

There are lots of available services, but we will focus on the Open Science Framework.

<http://etc.ch/yNtn>



TRANSPORT LOGISTICS: LOTS OF DATA, 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 copy
get(ok): /tmp/studyforrest-data-phase2/sub-01/ses-movie/func/sub-01_ses-movie_task-mo
```

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

```
# eNKI dataset (1.5TB, 34k files): copy
$ 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:

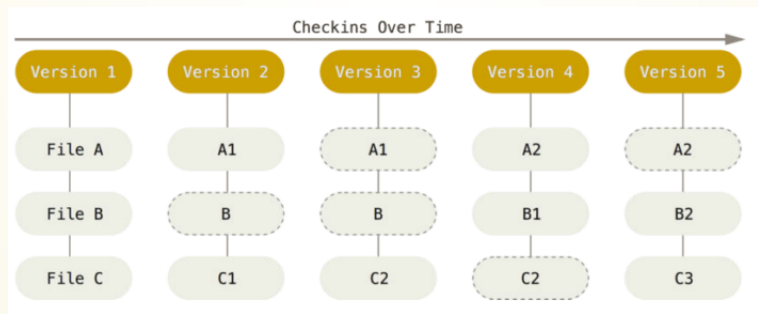
```
$ datalad drop sub-01/ses-movie/func/sub-01_ses-movie_task-movie_run-1_bold.nii.gz copy  
drop(ok): /tmp/studyforrest-data-phase2/sub-01/ses-movie/func/sub-01_ses-movie_task-mov
```

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

```
dl.get('input/sub-01') copy  
[really complex analysis]  
dl.drop('input/sub-01')
```

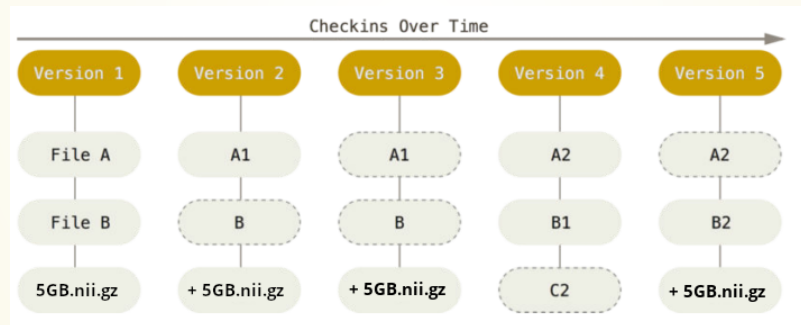
THERE ARE TWO VERSION CONTROL TOOLS AT WORK - WHY?

Git does not handle large files well.



THERE ARE TWO VERSION CONTROL TOOLS AT WORK - WHY?

Git does not handle large files well.




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-s2418
--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.p
```

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 wherever it is stored.
- Files stored in Git are modifiable, annexed files are protected against accidental modifications

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

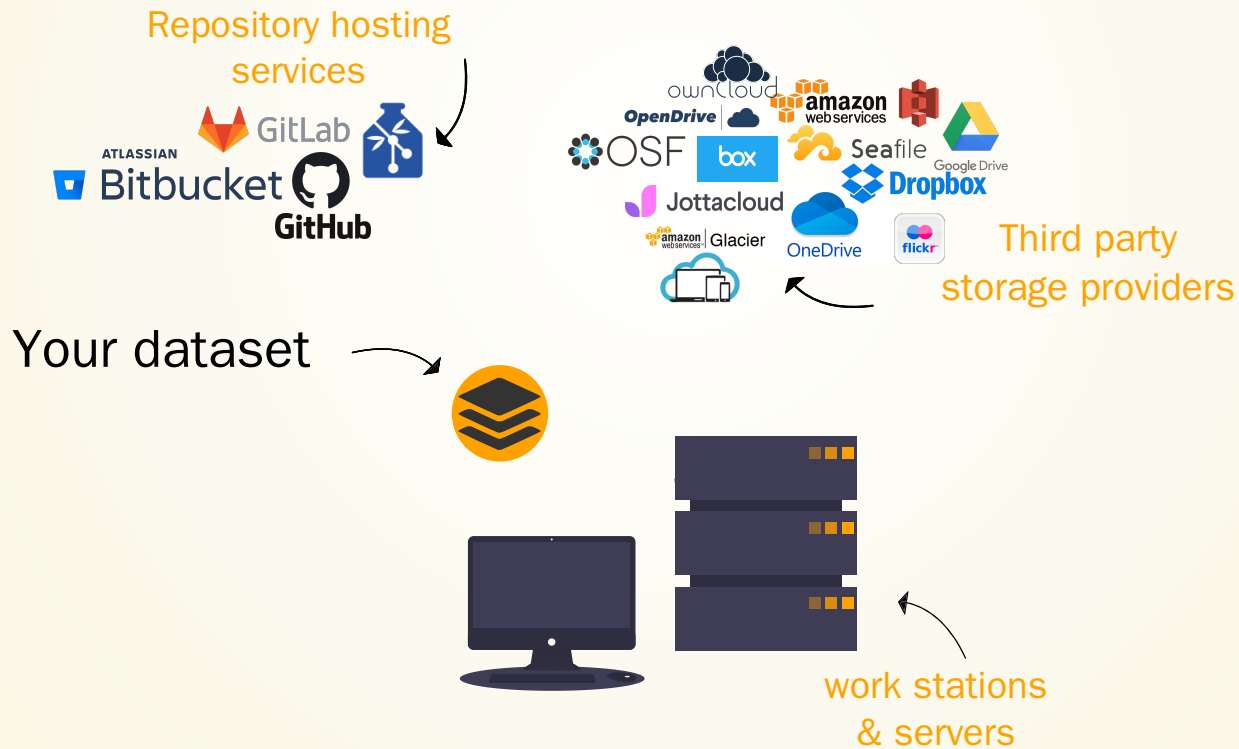
GIT VERSUS GIT-ANNEX

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`)

PUBLISHING DATASETS

I have a dataset on my computer. How can I share it, or collaborate on it?



GLOSSARY

Sibling (remote)

Linked clones of a dataset. You can usually update (from) siblings to keep all your siblings in sync (e.g., ongoing data acquisition stored on experiment compute and backed up on cluster and external hard-drive)

Repository hosting service

Webservices to host Git repositories, such as GitHub, GitLab, Bitbucket, Gin, ...

Third-party storage

Infrastructure (private/commercial/free/...) that can host data. A "special remote" protocol is used to publish or pull data to and from it

Publishing datasets

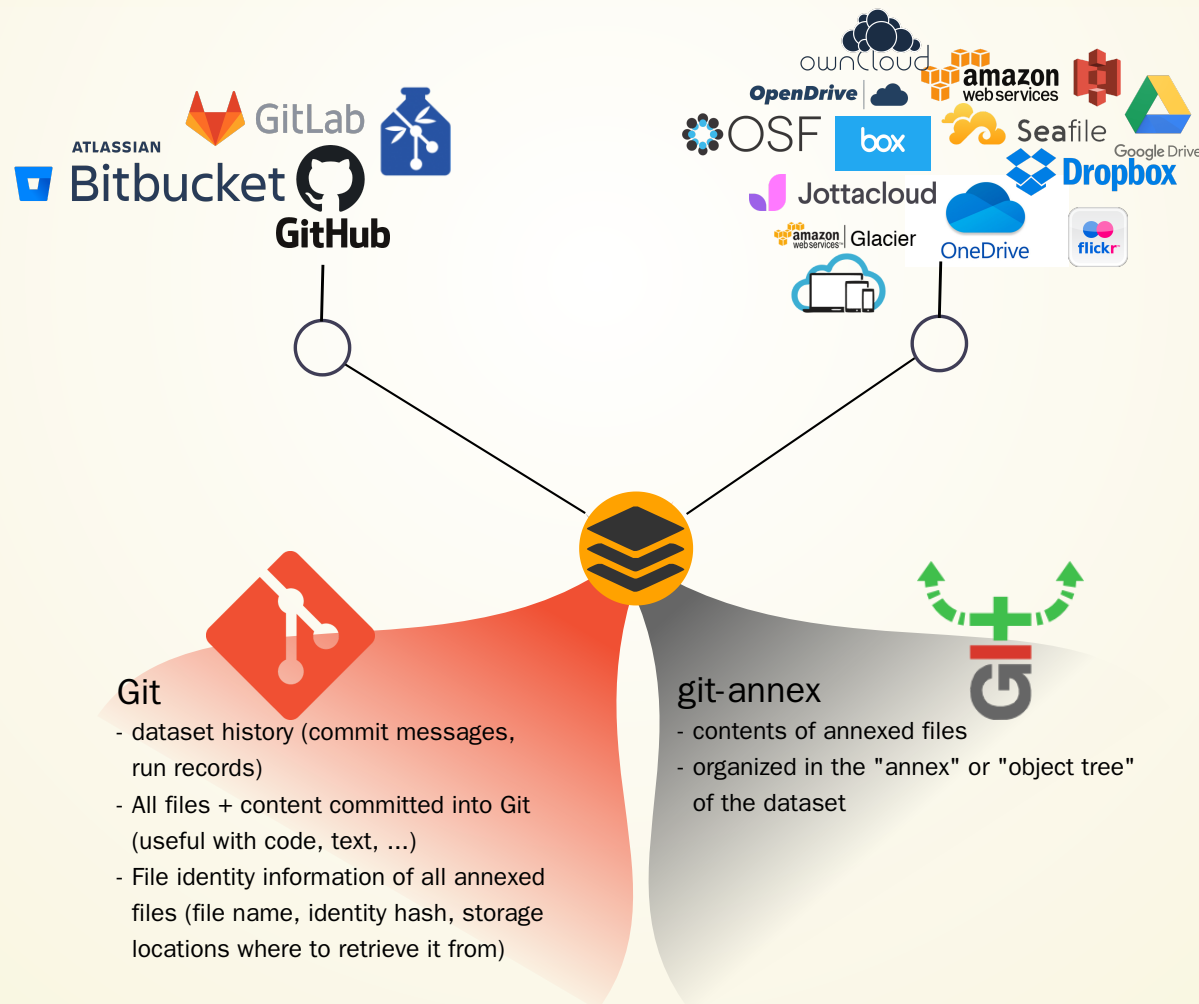
Pushing dataset contents (Git and/or annex) to a sibling using **datalad push**

Updating datasets

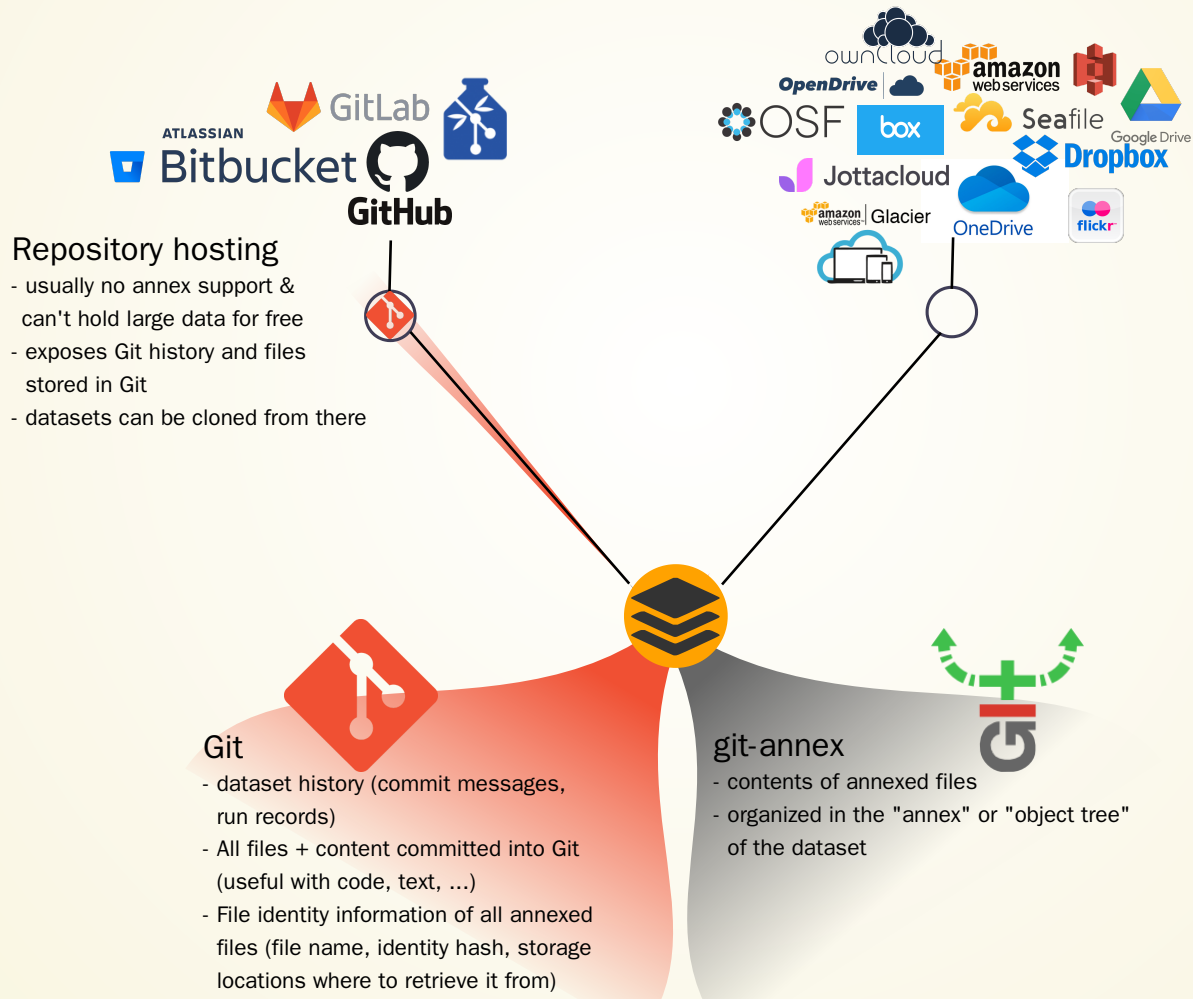
Pulling new changes from a sibling using **datalad update --merge**

PUBLISHING DATASETS

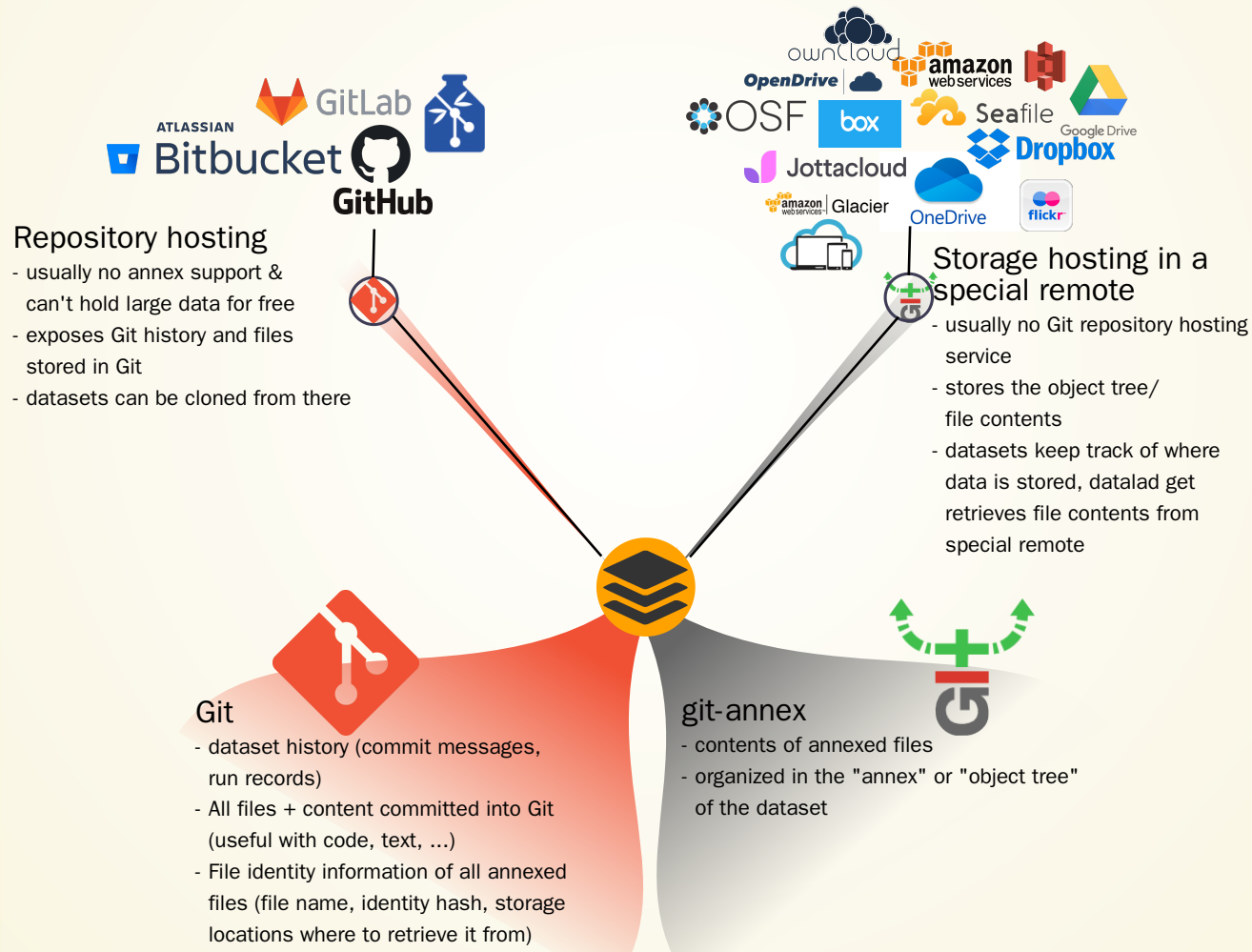
- Most public datasets separate content in Git versus git-annex behind the scenes



PUBLISHING DATASETS



PUBLISHING DATASETS



PUBLISHING DATASETS

Typical case:

- Datasets are exposed via a private or public repository on a repository hosting service
- Data can't be stored in the repository hosting service, but can be kept in almost any third party storage
- Publication dependencies automate pushing to the correct place, e.g.,

```
$ git config --local remote.github.datalad-publish-depends gdrive  
# or  
$ datalad siblings add --name origin --url git@git.jugit.fzj.de:adswa/experiment-data.git --publish-depends s3
```

PUBLISHING DATASETS

Special case 1: repositories with annex support



Repositories with annex support

- examples: GIN (gin.g-node.org), GitLab instances with enabled annex support
- can hold large data for free
- exposes Git history and all files + content
- datasets can be cloned from there

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

PUBLISHING DATASETS

Special case 2: Special remotes with repositories



Publishing to OSF

<https://osf.io/>



CREATE-SIBLING-OSF (docs)

Requires the DataLad extensions `datalad-osf` and `datalad-next`

Prerequisites:

1. Log into OSF
2. Create personal access token
3. Enter credentials using `datalad osf-credentials:`

```
datalad osf-credentials
```

copy

CREATE-SIBLING-OSF (docs)

Create the sibling in your dataset (different modes are possible):

```
datalad create-sibling-osf -d . -s my-osf-sibling \  
--title 'my-osf-project-title' --mode export --public
```

copy

Push to the sibling:

```
datalad push -d . --to my-osf-sibling
```

copy

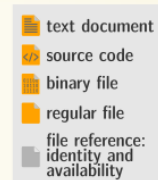
Clone from the sibling:

```
cd ..  
datalad clone osf://my-osf-project-id my-osf-clone
```

copy

SUMMARY AND TAKE-HOME MESSAGES

EXHAUSTIVE TRACKING OF RESEARCH COMPONENTS



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



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

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

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

% detailed_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
```

```
& detailed 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

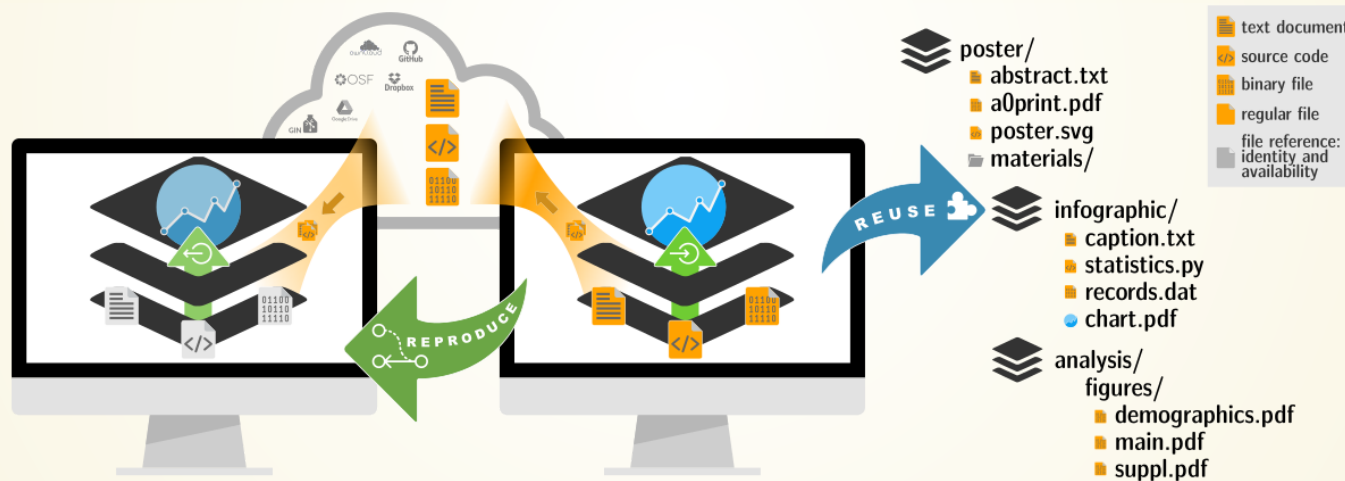
```
# obtain dataset (initially only identity,  
# availability, and provenance metadata)
```

```
% detailed_clone <url>
```

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

```
% detailed_run <commit/tag/branch>
```

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

% detailed clone_d <superdataset> <url> <path_in_dataset>
```

YOUR QUESTIONS AND USECASES

POST-WORKSHOP CONTACT

- Slides are CC-BY. They will stay online and will be made available as a PDF as well
- Contact the DataLad Team anytime via GitHub issue, Matrix chat message, or in our office hour video call
- Find more DataLad content and tutorials at handbook.datalad.org
- Join us at our first conference for distributed data management: distribits.live (April 2024, registration closes October 15th)

THANKS FOR YOUR ATTENTION!

LIST OF INSTALLED SOFTWARE ON JUPYTER

The JupyterHub runs on Ubuntu 22.04 via an AWS EC2 instance. The following packages were installed with different package managers:

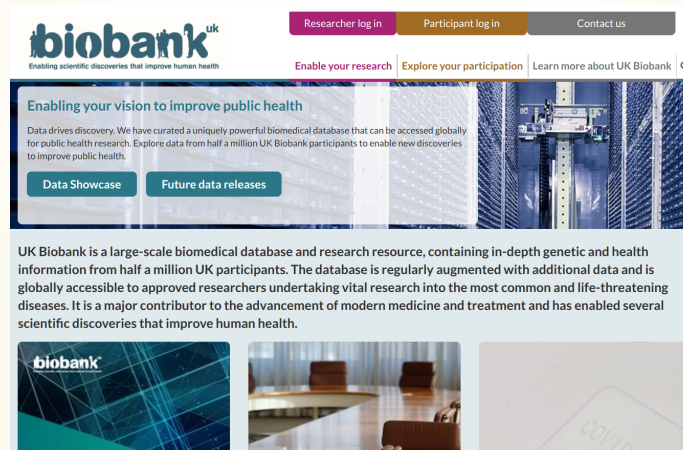
- apt: Git, git-annex, tree, tig, zsh, singularity
- pip: datalad, datalad-next, datalad-container, datalad-osf, black

Instructions to set up and configure your own JupyterHub are publicly available at psychoinformatics-de.github.io/rdm-course/for_instructors

OUTLOOK

FAIRLY BIG: SCALING UP

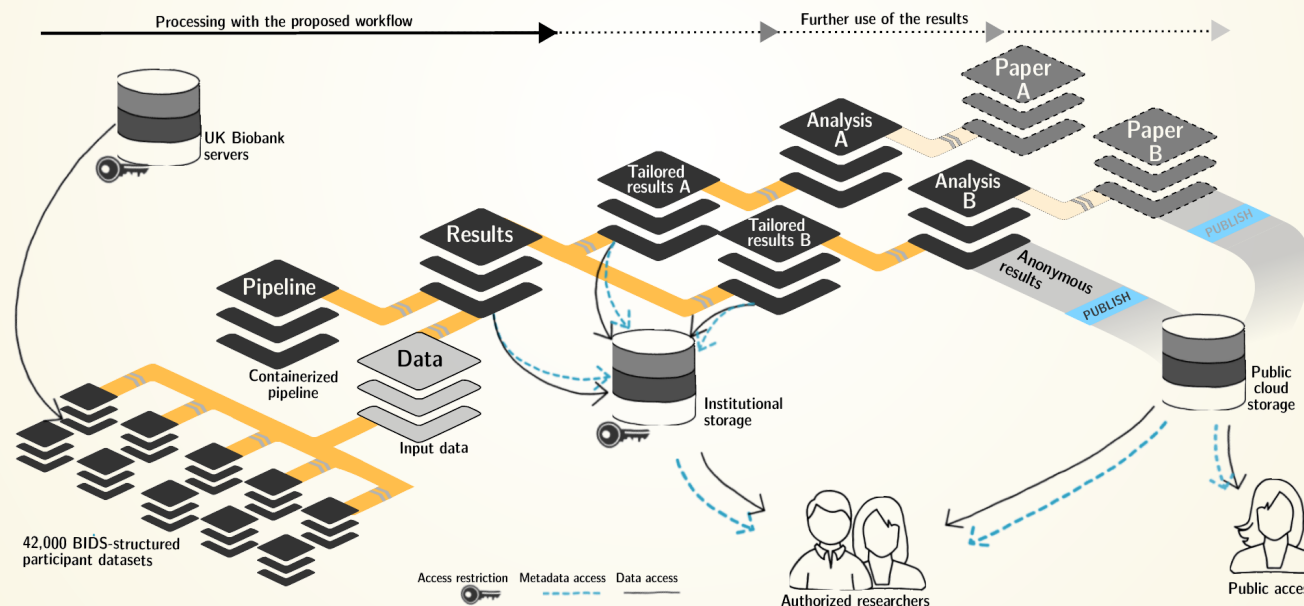
Objective: Process the UK Biobank (imaging data)



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

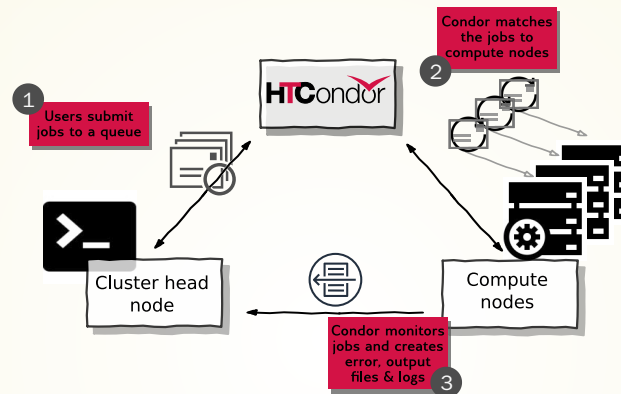
FAIRLY BIG SETUP



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

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

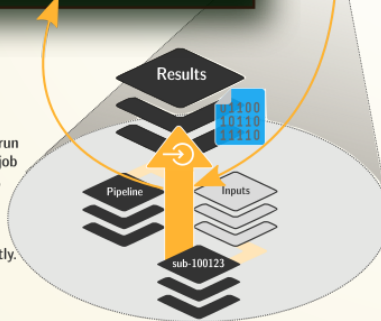
Wagner, Waite, Wierzba et al. (2021). FAIRly big: A framework for computationally reproducible processing of large-scale data.

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 ^^^
```

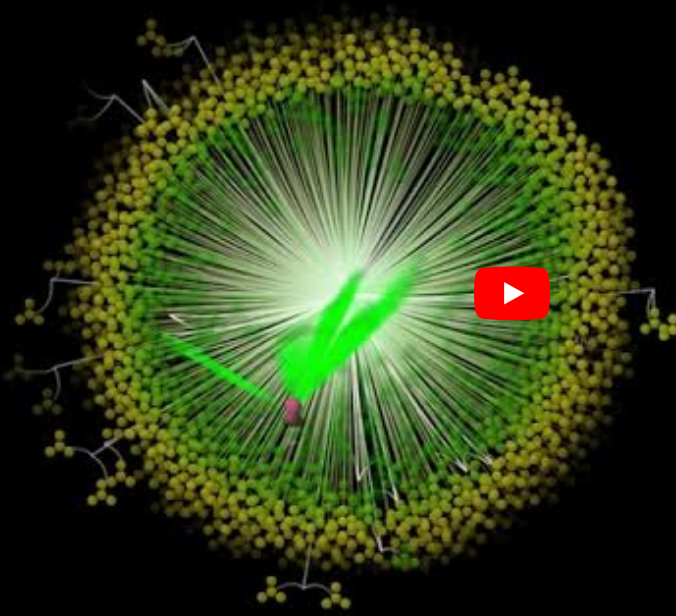
```
---
sub-6025043/ses-2/inforoi.tar.gz | 1 +
sub-6025043/ses-2/native.tar.gz | 1 +
sub-6025043/ses-2/surface.tar.gz | 1 +
sub-6025043/ses-2/vbm.tar.gz | 1 +
4 files changed, 4 insertions(+)
```

	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: dfa6d975ea888ed33bf714c67
	Input data
	Origin: http://ukb.ds.inm7.de/.../bids Version: 0c7f0b45140dde1d7291b1572
	Expected output data/folder
	Captured output data
	Path, Content hash

FAIRLY BIG MOVIE

2021-01-12 19:07

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



Each job

- processed one image
- required one CPU hour
- needed 4 GB RAM
- used 5 GB of disk space
- created four output files

- Two computations on clusters of different scale (small cluster, supercomputer). Full video: <https://youtube.com/datalad>
- Two full (re-)computations, programmatically comparable, verifiable.