Reproducible Research Data Management with DataLad

6th RDM-Workshop 2024 on Research Data Management in the Max Planck Society

Slides | Source

 DOI 10.5281/zenodo.10844705

Dr. Lennart Wittkuhn
wittkuhn@mpib-berlin.mpg.de
Max Planck Research Group NeuroCode, Max Planck Institute for Human Development, Berlin, Germany
Max Planck UCL Centre for Computational Psychiatry and Ageing Research, Berlin, Germany
Institute of Psychology, University of Hamburg, Germany

March 20, 2024

About

Dr. Lennart Wittkuhn

wittkuhn@mpib-berlin.mpg.de
https://lennartwittkuhn.com/
Mastodon GitHub LinkedIn

About me

I am a **Postdoctoral Research Data Scientist** in Cognitive Neuroscience at the Institute of Psychology at the University of Hamburg (PI: Nicolas Schuck)

BSc Psychology & **MSc Cognitive Neuroscience** (TU Dresden), **PhD Cognitive Neuroscience** (Max Planck Institute for Human Development)

I study **the role of fast neural memory reactivation** in the human brain, applying **machine learning** and **computational modeling** to **fMRI** data

I am passionate about **computational reproducibility**, **research data management**, **open science** and tools that improve the scientific workflow

Find out more about my work on my website, Google Scholar and ORCiD

About this presentation

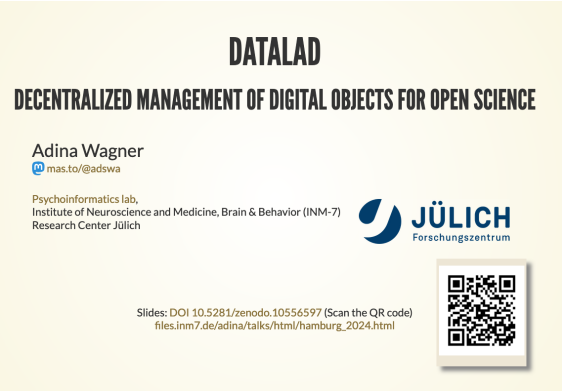
Slides: https://lennartwittkuhn.com/talk-mps-fdm-2024

Source: https://github.com/lnnrtwttkhn/talk-mps-fdm-2024

Software: Reproducible slides built with Quarto and deployed to GitHub Pages using GitHub Actions for continuous integration & deployment

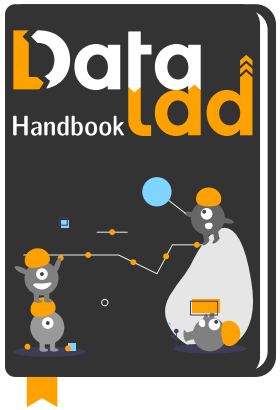
Contact: Feedback or suggestions via email or GitHub issues. Thank you!

Acknowledgements and further reading



Slides by Wagner (2024) (CC BY-SA 4.0)

Slides and presentations by Dr. Adina Wagner and the DataLad team, e.g., "DataLad - Decentralized Management of Digital Objects for Open Science" (Wagner 2024)



DataLad Handbook (CC BY-SA 4.0)

The **DataLad Handbook** by Wagner et al. (2022) is a comprehensive educational resource for data management with DataLad.

Papers

- Wilson et al. (2014). Best practices for scientific computing. *PLOS Biology*.
- Wilson et al. (2017). Good enough practices in scientific computing. *PLOS Computational Biology*.
- Lowndes et al. (2017). Our path to better science in less time using open data science tools. *Nature Ecology Evolution*.

Talks

- Richard McElreath (2020). Science as amateur software development. YouTube
- Russ Poldrack (2020). Toward a Culture of Computational Reproducibility. YouTube

… and many more!

Agenda

1. Scientific workflows with DataLad

1.1 Version Control

1.2 Modularity & Linking

1.3 Provenance

1.4 Collaboration & Interoperability

2. Integrating DataLad with MPS infrastructure (and beyond)

2.1 Keeper

2.2 Edmond

2.3 ownCloud / nextCloud

2.4 GIN

2.5 GitLab

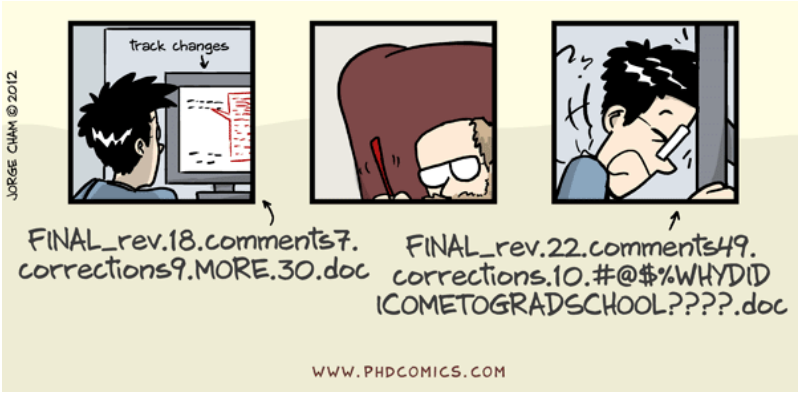
3. Summary & Discussion

Scientific building blocks are not static

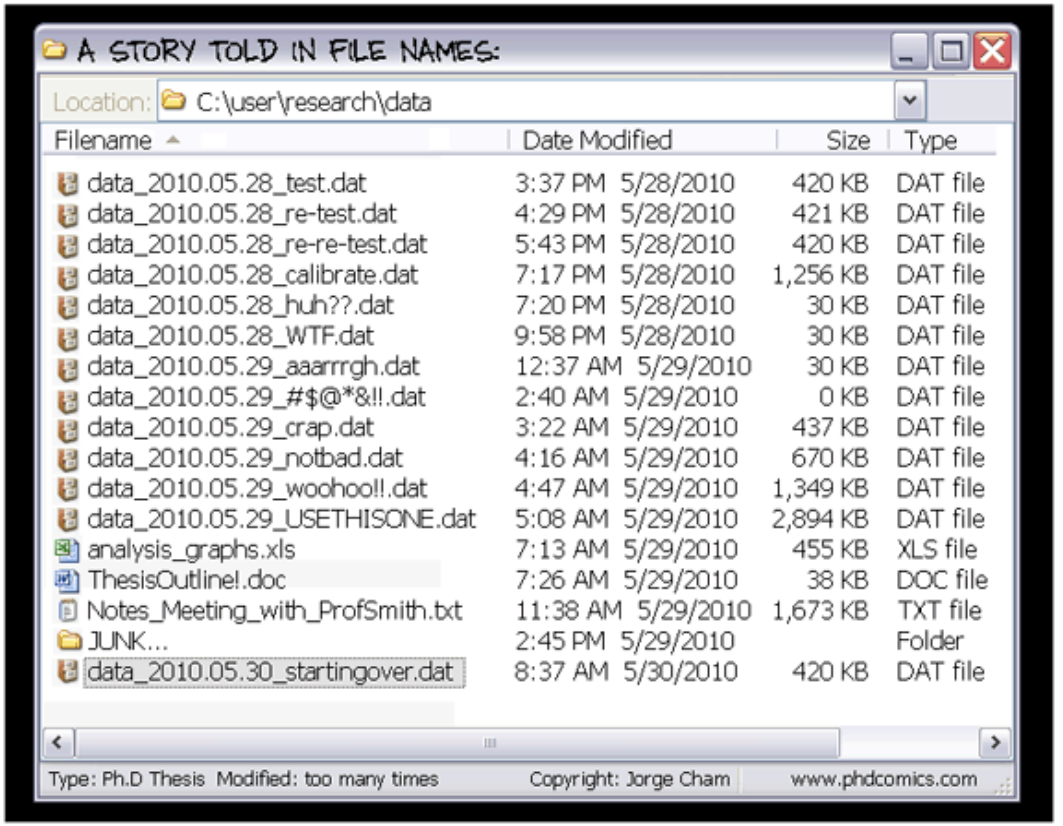
We need version control

Why we need version control

… for **code** (text files)



… for **data** (binary files)



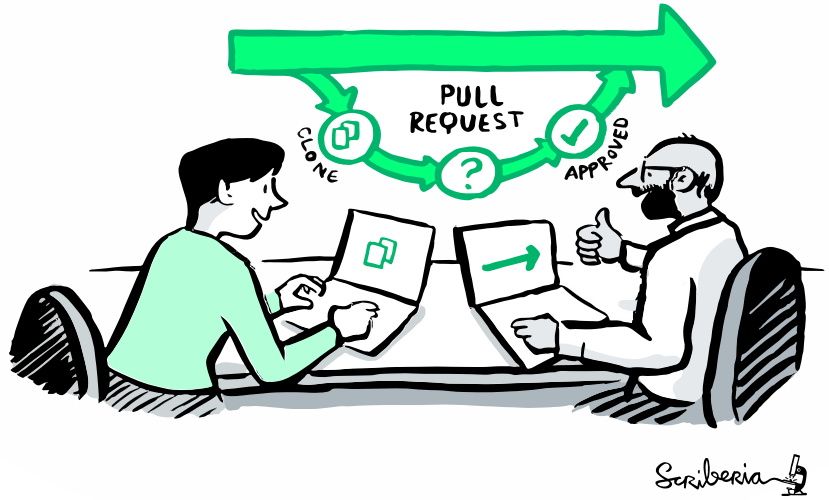
If everything is relevant, track everything.

What is version control?

"Version control is a systematic approach to record changes made in a […] set of files, over time. This allows you and your collaborators to track the history, see what changed, and recall specific versions later […]" (Turing Way)

- keep track of changes in a directory (a "repository")
- take snapshots ("commits") of your repo at any time
- know the history: what was changed when by whom
- compare commits and go back to any previous state
- work on parallel "branches" & flexibly "merge" them
- "push" your repo to a "remote" location & share it
- share repos on platforms like GitHub or GitLab
- work together on the same files at the same time
- others can read, copy, edit and suggest changes
- make your repo public and openly share your work

by The Turing Way Community and Scriberia (2024) (CC BY 4.0)



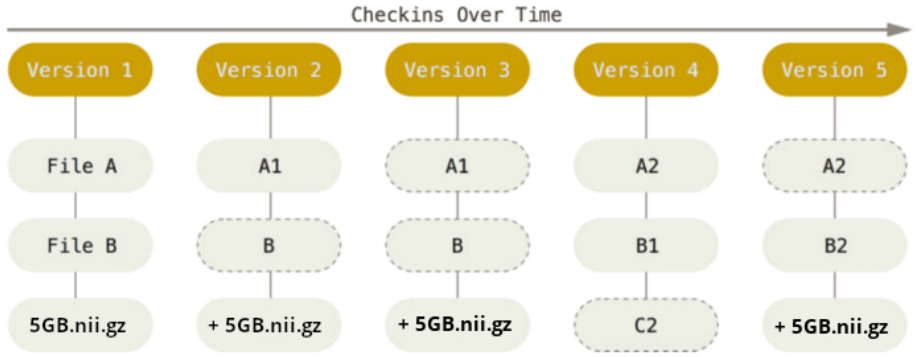
by The Turing Way Community and Scriberia (2024) (CC BY 4.0)

What are Git and DataLad?

git-scm.com (by Jason Long; CC BY 3.0 Unported)

- most popular version control system
- free, open-source command-line tool
- graphical user interfaces exist, e.g., GitKraken
- standard tool in the software industry
- 100 million GitHub users [1]

Sadly, Git does not handle large files well.

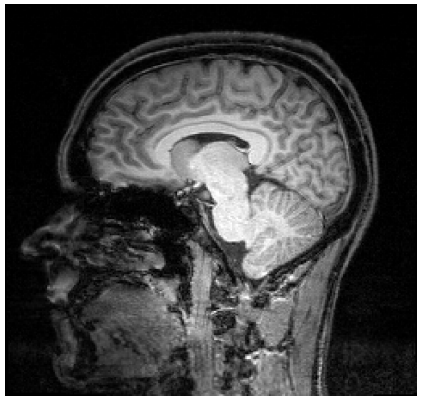


datalad.org (from the DataLad Handbook by Wagner et al. (2022); CC BY-SA 4.0)

- "Git for (large) data"
- free, open-source command-line tool
- builds on top of Git and git-annex
- **allows to version control arbitrarily large datasets** [2]
- DataLad Python API: Use DataLad in your Python code
- graphical user interface exists: DataLad Gooey

Example Dataset: Brain Imaging Data

Single subject epoch (block) auditory fMRI activation data



```
dicomdir/
  1208200617178_22/
    1208200617178_22_8973.dcm
    1208200617178_22_8943.dcm
    1208200617178_22_2973.dcm
    1208200617178_22_8923.dcm
    1208200617178_22_4473.dcm
    1208200617178_22_8783.dcm
    1208200617178_22_7328.dcm
    1208200617178_22_9264.dcm
    1208200617178_22_9967.dcm
    1208200617178_22_3894.dcm
    1208200617178_22_3899.dcm
  1208200617178_23/
  1208200617178_24/
  1208200617178_25/
```

```
my_dataset/
  participants.tsv
  sub-01/
    anat/
      sub-01_T1w.nii.gz
    func/
      sub-01_task-rest_bold.nii.gz
      sub-01_task-rest_bold.json
    dwi/
      sub-01_dwi.nii.gz
      sub-01_dwi.json
      sub-01_dwi.bval
      sub-01_dwi.bvec
  sub-02/
  sub-03/
  sub-04/
```

Brain Imaging Data Structure (BIDS) Gorgolewski et al. (2016) (CC BY 4.0)

```
mkdir neuro-data
wget https://www.fil.ion.ucl.ac.uk/spm/download/data/MoAEpilot/MoAEp
-O neuro-data.zip
unzip neuro-data.zip -d neuro-data
rm neuro-data.zip
cd neuro-data
mv MoAEpilot/* .
rm -R MoAEpilot
```

BIDS
BRAIN IMAGING DATA STRUCTURE

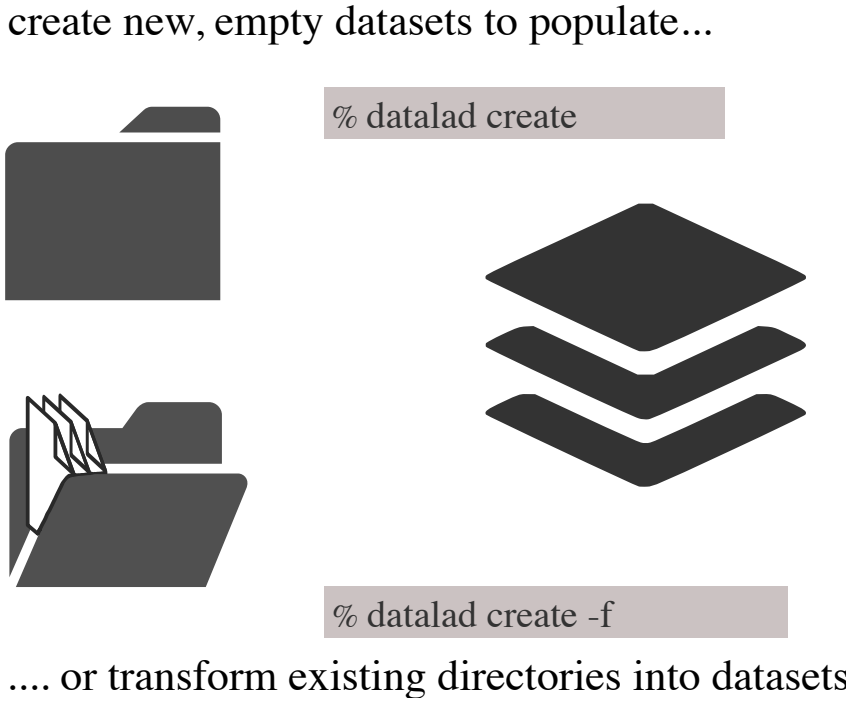
Brain Imaging Data Structure (BIDS) Gorgolewski et al. (2016) (CC BY 4.0)

```
tree
.
├── CHANGES
├── README
├── dataset_description.json
├── sub-01
│   ├── anat
│   │   └── sub-01_T1w.nii
│   └── func
│       ├── sub-01_task-auditory_bold.nii
│       └── sub-01_task-auditory_events.tsv
└── task-auditory_bold.json

4 directories, 7 files
```

Dataset from Functional Imaging Laboratory, UCL Queen Square Institute of Neurology, London, UK (Source)

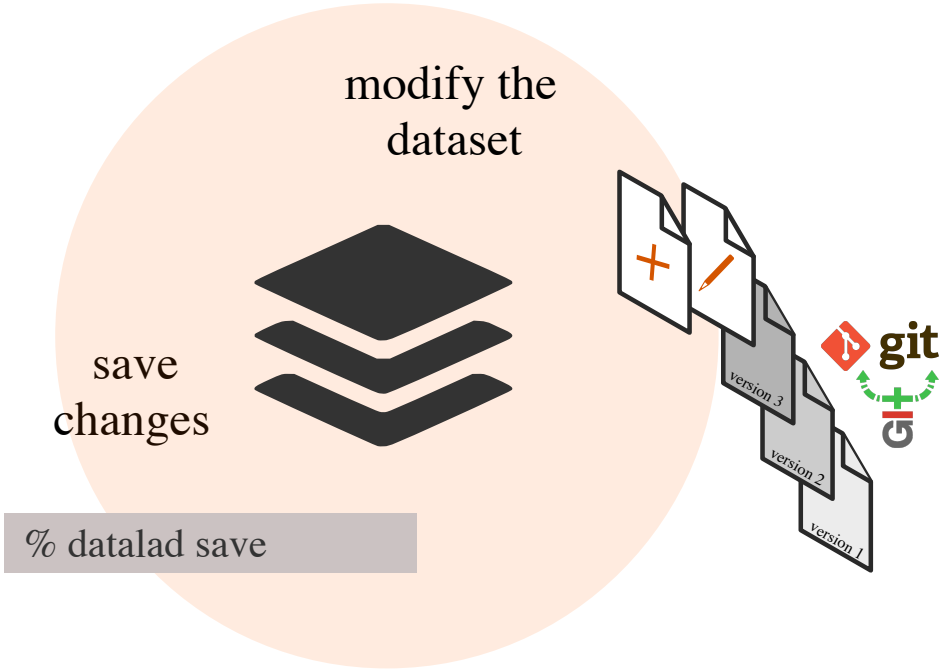
Version Control with DataLad



from the DataLad Handbook by Wagner et al. (2022) (CC BY-SA 4.0)

```
1  datalad create neuro-data
```

▶ View output

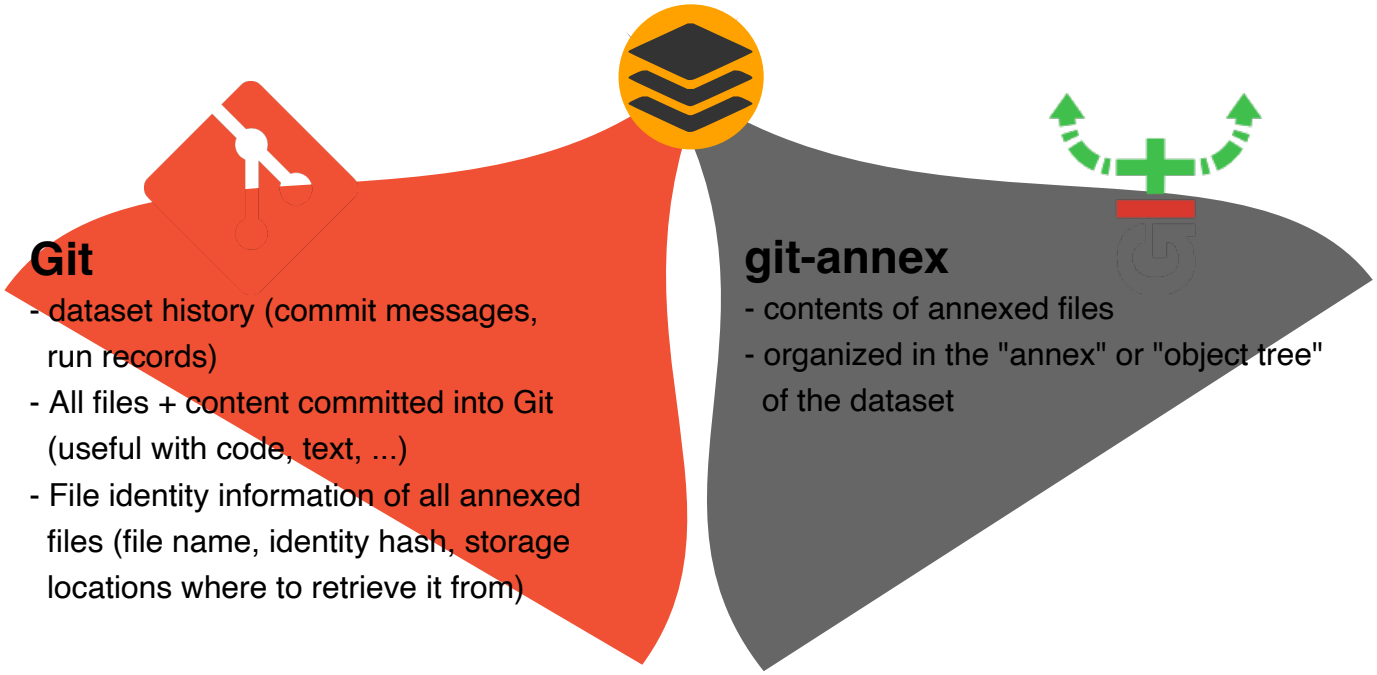


from the DataLad Handbook by Wagner et al. (2022) (CC BY-SA 4.0)

```
1  datalad save -m "save neuro data"
```

▶ View output

Data in DataLad datasets are either stored in Git or git-annex



from the DataLad Handbook by Wagner et al. (2022) (CC BY-SA 4.0)

Git

- handles **small files** well (text, code)
- file contents are in Git history and will be **shared**
- Shared with every dataset clone
- Useful: small, non-binary, frequently modified files

git-annex

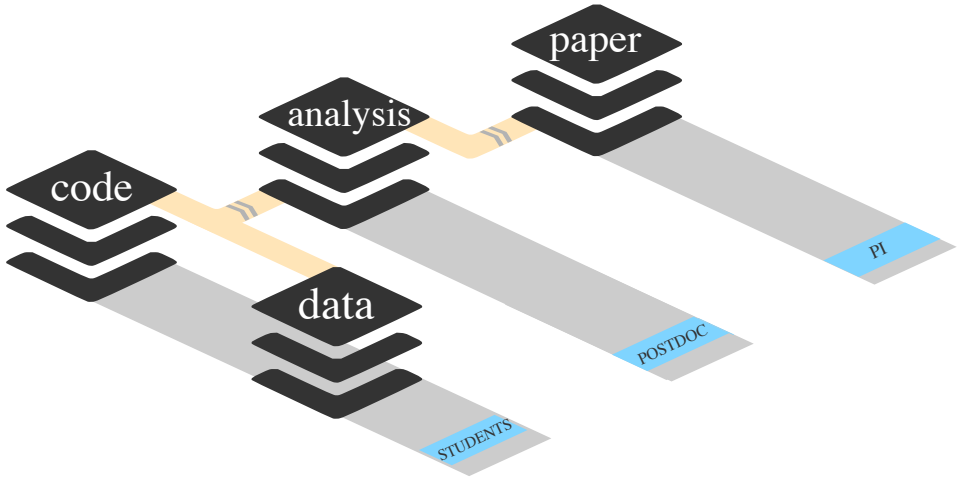
- handles **all** types and sizes of files well
- file contents are in the annex, not necessarily shared
- **Can be kept private** on a per-file level
- Useful: Large files, private files

Science is build from modular units

We need modularity and linking

Version control beyond single repositories

Research as a sequence

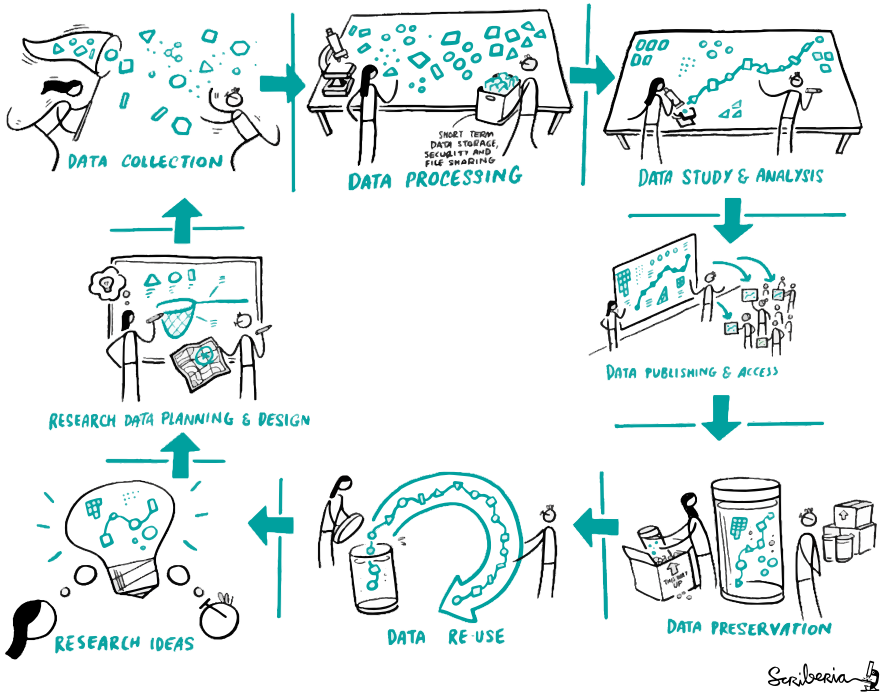


from the DataLad Handbook by Wagner et al. (2022) (CC BY-SA 4.0)

- Prior works (code development, empirical data, etc.) are combined to produce results with goal of a publication
- Aggregation across time and contributors
- Aiming for (but often failing) to be reproducible
- Often, there is one big project folder

A single repository is not enough!

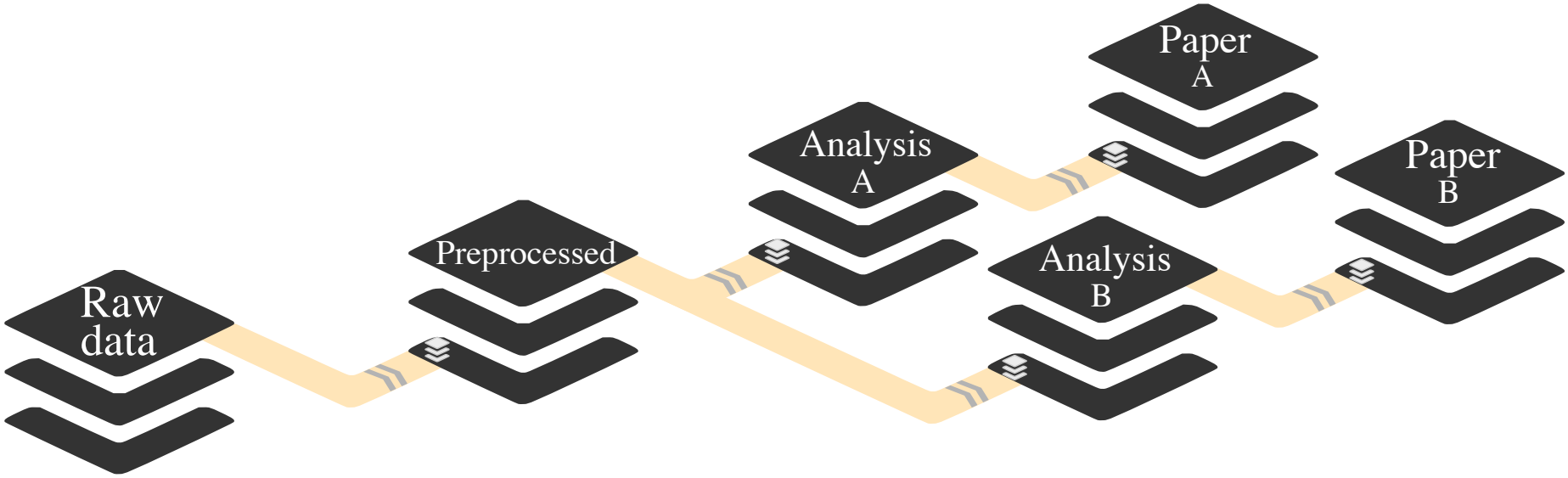
Research as a cycle



by The Turing Way Community and Scriberia (2024) (CC BY 4.0)

- Develop scientific outputs as modular but linked units
- Independently update and develop data sources
- Manage access to public / private datasets

Nesting of modular DataLad datasets

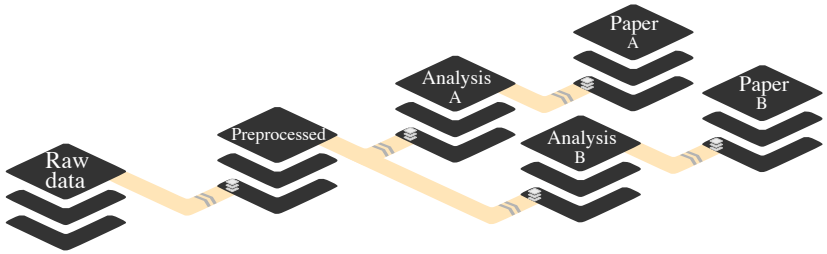


Nest modular datasets to create a linked hierarchy of datasets, and enable recursive operations throughout the hierarchy

from the DataLad Handbook by Wagner et al. (2022) (CC BY-SA 4.0)

- seamless nesting of modular datasets in hierarchical super-/sub-dataset relationships
- based in Git submodules, but mono-repo feel thanks to recursive operations
- overcomes scaling issues with large amounts of files (Example: Human Connectome Project)
- modularizes research components for transparency, reuse and access management

Example: Intuitive data analysis structure



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

from the DataLad Handbook by Wagner et al. (2022) (CC BY-SA 4.0)

First, let's create a new data analysis dataset:

```
datalad create -c yoda myanalysis
[INFO   ] Creating a new annex repo at /tmp/myanalysis
[INFO   ] Scanning for unlocked files (this may take some time)
[INFO   ] Running procedure cfg_yoda
[INFO   ] == Command start (output follows) =====
[INFO   ] == Command exit (modification check follows) =====
create(ok): /tmp/myanalysis (dataset)
```

`-c yoda` initializes useful structure (details here):

```
tree
.
├── CHANGELOG.md
├── README.md
└── code
    └── README.md
2 directories, 3 files
```

We install analysis input data as a subdataset to the dataset:

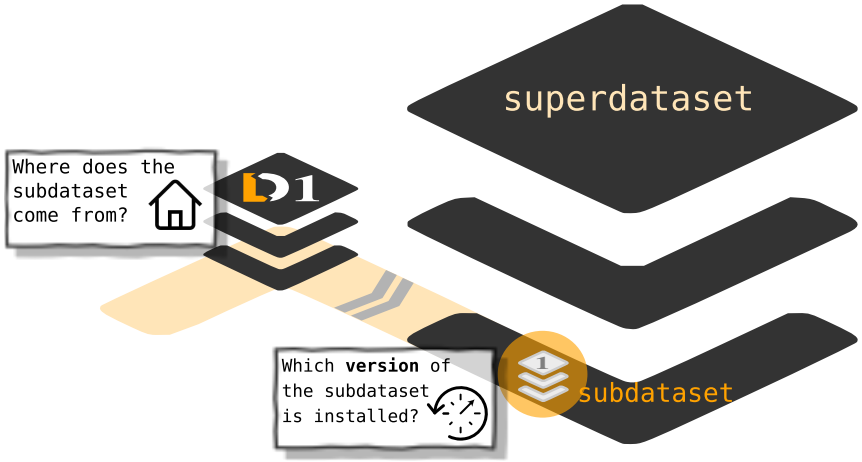
```
datalad clone -d . https://github.com/datalad-handbook/iris_data.git input/
[INFO   ] Remote origin not usable by git-annex; setting annex-ignore
install(ok): input (dataset)
add(ok): input (dataset)
add(ok): .gitmodules (file)
save(ok): . (dataset)
action summary:
  add (ok: 2)
  install (ok: 1)
  save (ok: 1)
```

`input` is a regular folder inside `myanalysis`

```
tree
.
├── CHANGELOG.md
├── README.md
├── code
│   └── README.md
└── input
    └── iris.csv
3 directories, 4 files
```

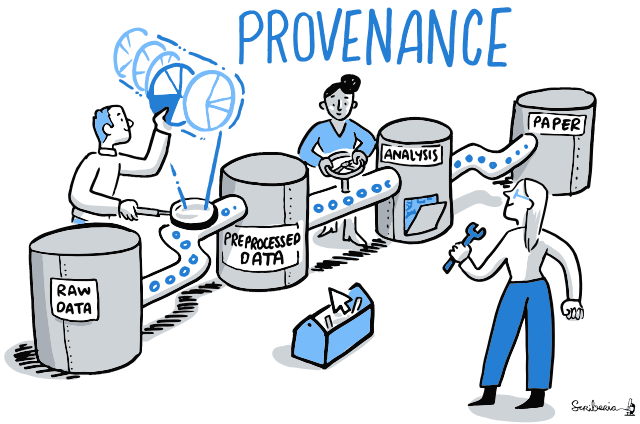
Modular units with clear provenance

```
git diff HEAD~1
diff --git a/.gitmodules b/.gitmodules
new file mode 100644
index 0000000..fc69c84
--- /dev/null
+++ b/.gitmodules
@@ -0,0 +1,5 @@
+[submodule "input"]
+       path = input
+       url = https://github.com/datalad-handbook/iris_data.git
+       datalad-id = 5800e71c-09f9-11ea-98f1-e86a64c8054c
+       datalad-url = https://github.com/datalad-handbook/iris_data.git
diff --git a/input b/input
new file mode 160000
index 0000000..b9eb768
--- /dev/null
+++ b/input
@@ -0,0 +1 @@
+Subproject commit b9eb768c145e4a253d619d2c8285e540869d2021
```



from the DataLad Handbook by Wagner et al. (2022) (CC BY-SA 4.0)

- We know *exactly* where the subdataset comes from
- We know *exactly* which version of the subdataset is installed
- We can develop and update each subdataset independently

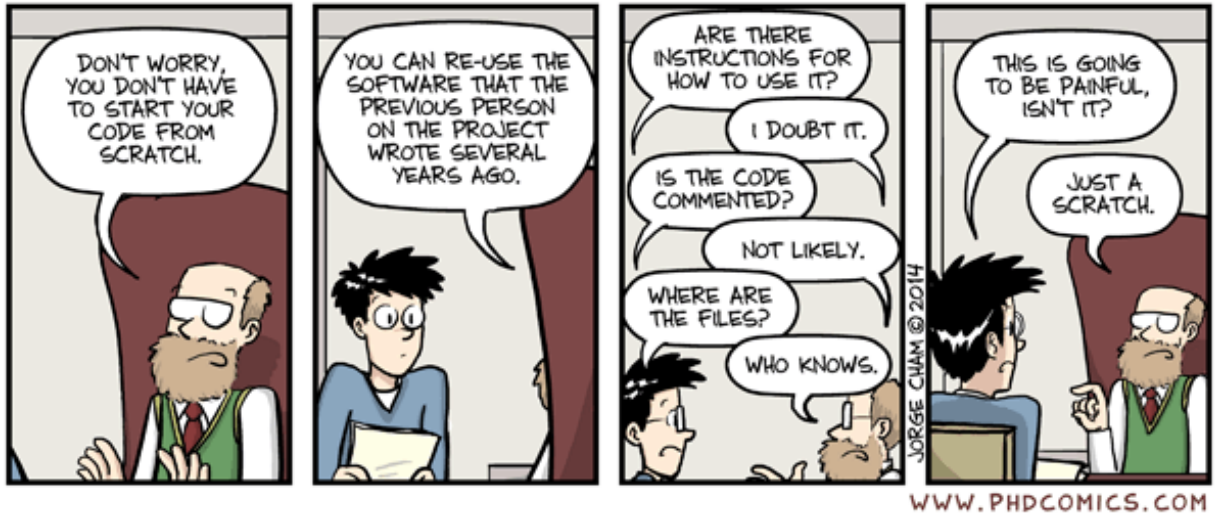


by The Turing Way Community and Scriberia (2024) (CC BY 4.0)

Science is exploratory and iterative

We need provenance

Reusing previous work is hard



© Jorge Cham (phdcomics.com)

by The Turing Way Community and Scriberia (2024)

"Your number one collaborator is yourself from 6 months ago and they don't answer emails."

by The Turing Way Community and Scriberia (2024)

"Which version of which script produced these outputs from which version of which data?"

Establishing provenance with DataLad

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

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

datalad containers-run executes command line calls inside a tracked software container and saves the dataset modifications resulting from the execution.

from the DataLad Handbook by Wagner et al. (2022) (CC BY-SA 4.0)

```
datalad containers-run \
  --message "Time series extraction from Locus Coeruleus"
  --container-name nilearn \
  --input 'mri/*_bold.nii' \
  --output 'sub-*/LC_timeseries_run-*.csv' \
  "python3 code/extract_lc_timeseries.py"

-- Git commit --
  commit 5a7565a640ff6de67e07292a26bf272f1ee4b00e
  Author:     Adina Wagner adina.wagner@t-online.de
  AuthorDate: Mon Nov 11 16:15:08 2019 +0100

  [DATALAD RUNCMD] Time series extraction from Locus Coeruleus
  === Do not change lines below ===
  {
   "cmd": "singularity exec --bind {pwd} .datalad/environments/nilear
   "dsid": "92ea1faa-632a-11e8-af29-a0369f7c647e",
   "inputs": [
    "mri/*.bold.nii.gz",
    ".datalad/environments/nilearn.simg"
   ],
   "outputs": ["sub-*/LC_timeseries_run-*.csv"],
   ...
  }
  ^^^ Do not change lines above ^^^
```

- Enshrine the analysis in a script and record code execution together with input data, output files and software environment in the execution command
- Result: Machine readable record about which data, code and software produced a result how, when and why
- Use the unique identifier (hash) of the execution record to have a machine recompute and verify past work

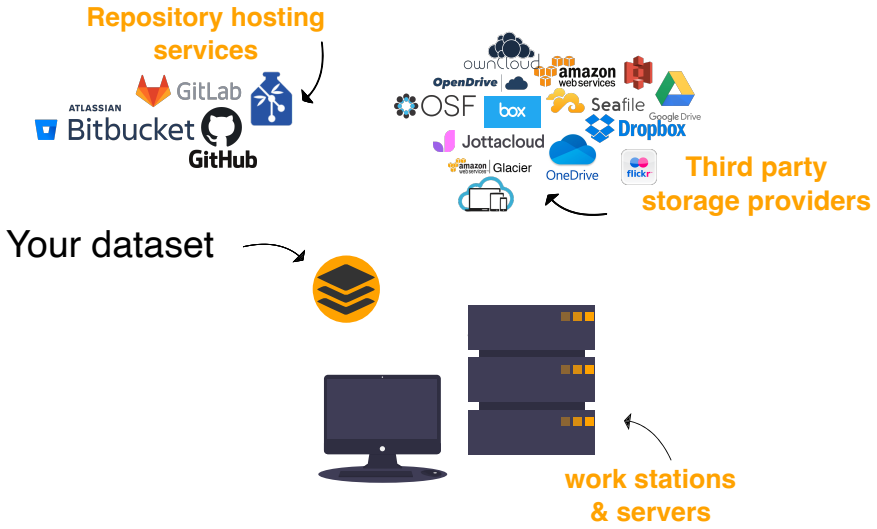
```
datalad rerun 5a7565a640ff6de67
[INFO   ] run commit 5a7565a640ff6de67; (Time series extraction from
[INFO   ] Making sure inputs are available (this may take some time)
get(ok): mri/sub-01_bold.nii (file)
        [...]
[INFO   ] == Command start (output follows) =====
[INFO   ] == Command exit (modification check follows) =====
add(ok): sub-01/LC_timeseries_run-*.csv(file)
```

Science is collaborative & distributed

We need interoperability & transport logistics

Data sharing and collaboration with DataLad

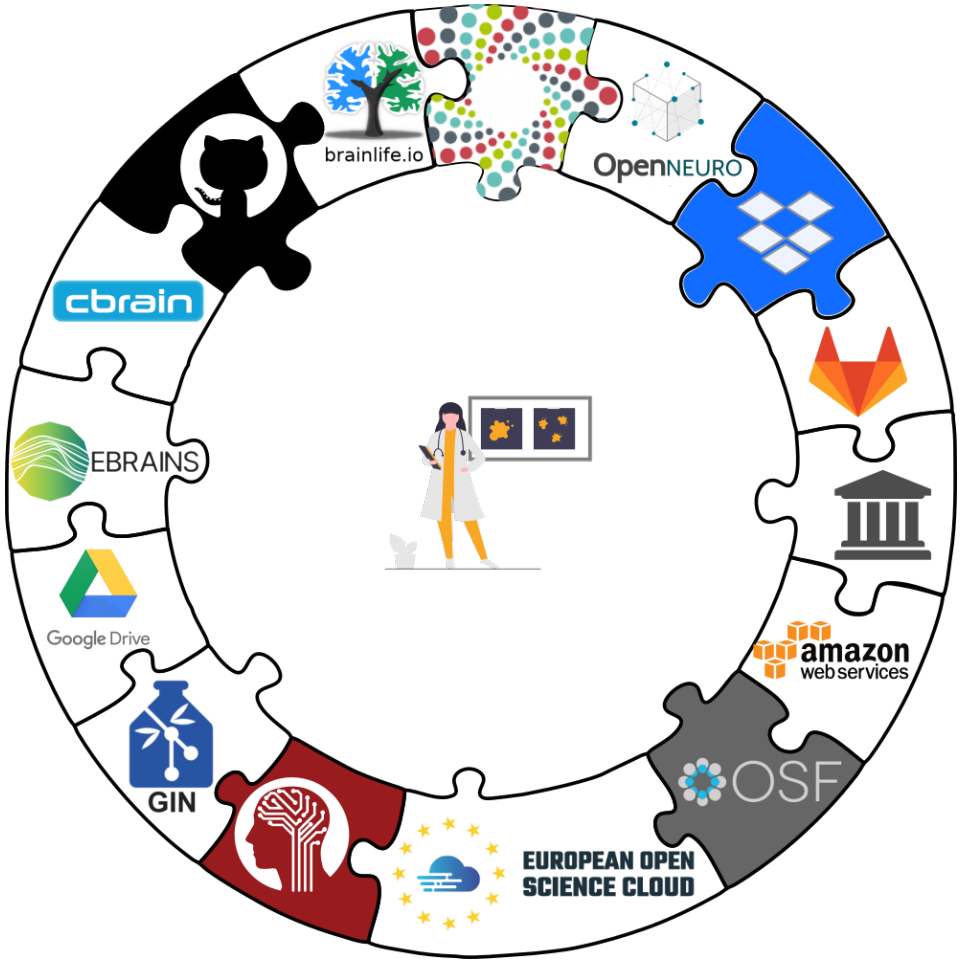
"*I have a dataset on my computer.*
How can I share it or collaborate on it?"




```
tree
.
├── CHANGES
├── README
├── dataset_description.json
├── sub-01
│   ├── anat
│   │   └── sub-01_T1w.nii
│   └── func
│       ├── sub-01_task-auditory_bold.nii
│       └── sub-01_task-auditory_events.tsv
└── task-auditory_bold.json

4 directories, 7 files
```

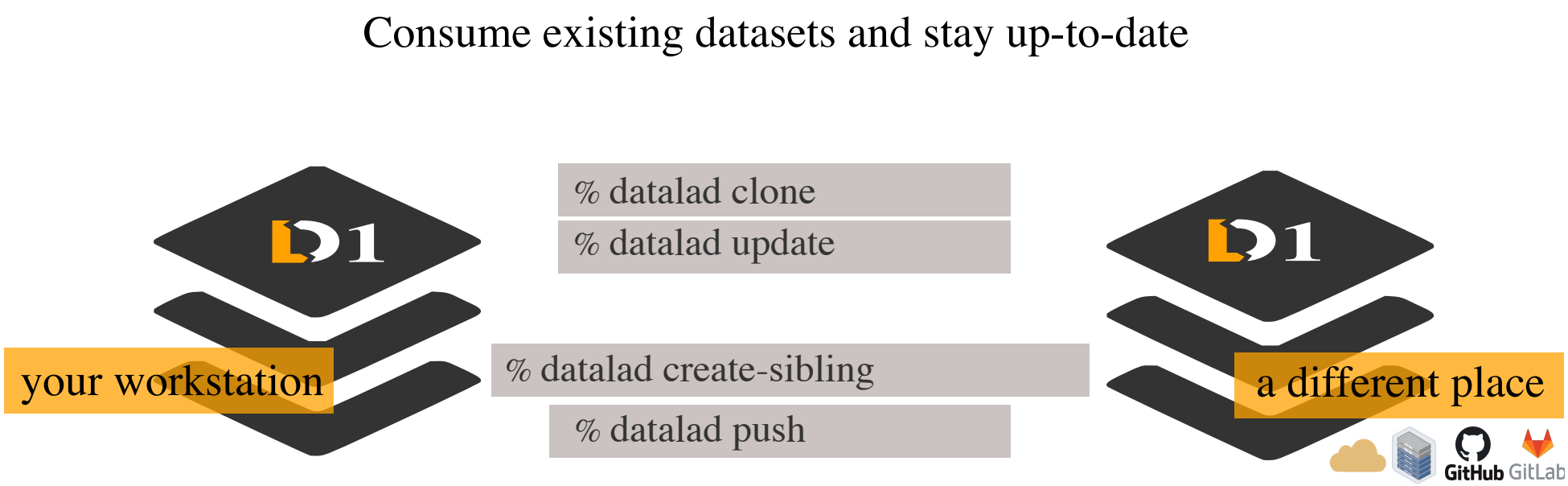
Challenge: Scientific workflows are idiosyncratic across institutions / departments / labs / any two scientists



from the DataLad Handbook by Wagner et al. (2022) (CC BY-SA 4.0)

Share data like code

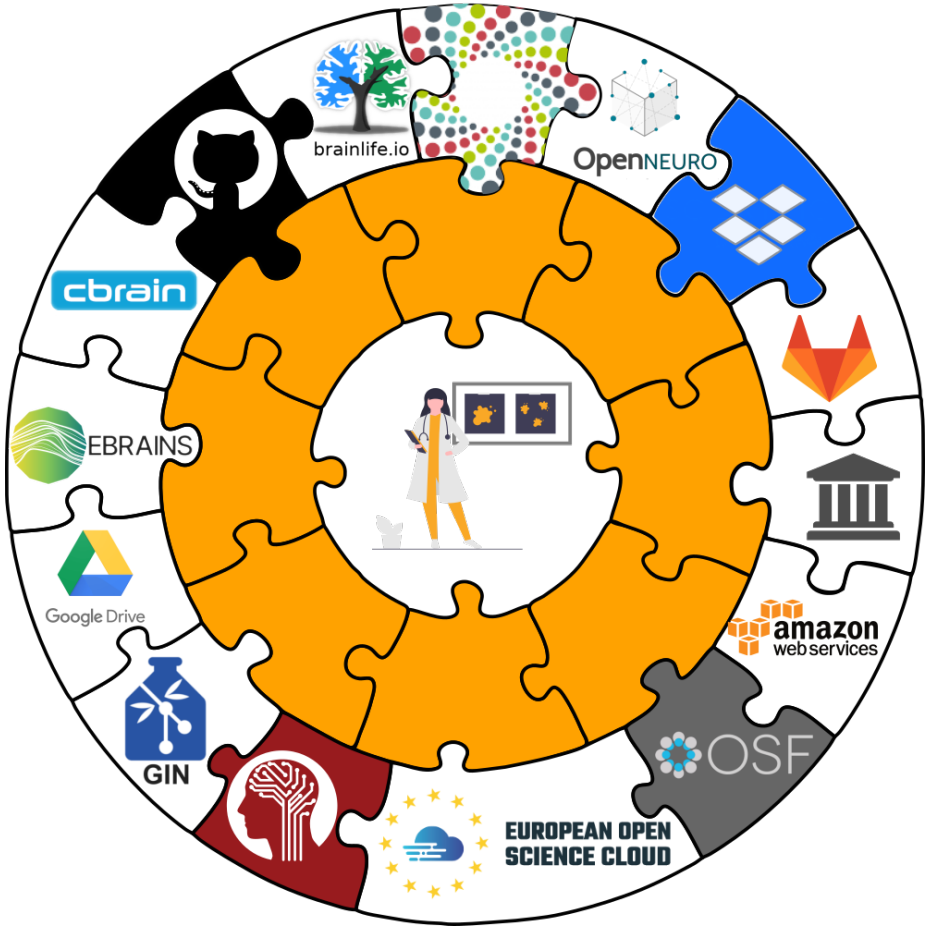
- With DataLad, you can **share data like you share code**: As version-controlled datasets via repository hosting services
- DataLad datasets can be cloned, pushed and updated from and to a wide range of remote hosting services



from the DataLad Handbook by Wagner et al. (2022) (CC BY-SA 4.0)

Interoperability with a range of hosting services

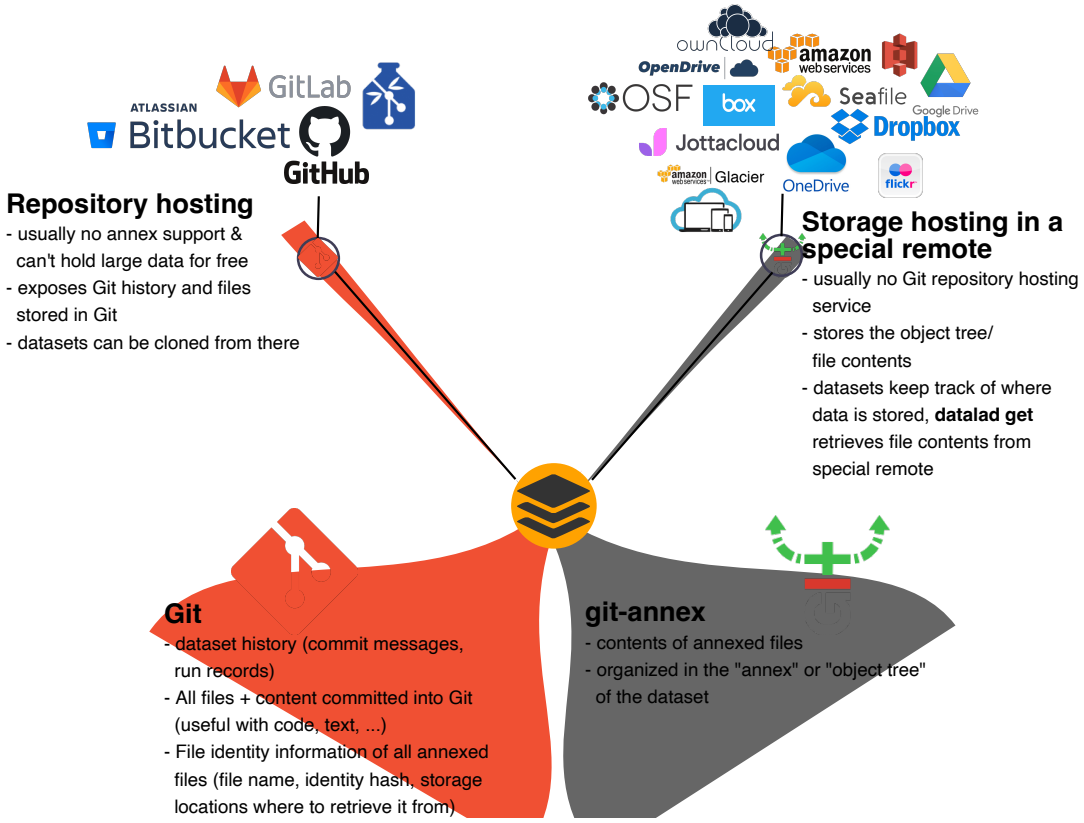
DataLad is built to **maximize interoperability** and **streamline routines across hosting services** and storage technology



see DataLad Handbook: "Beyond shared infrastructure"

Separate content in Git vs. git-annex behind the scenes

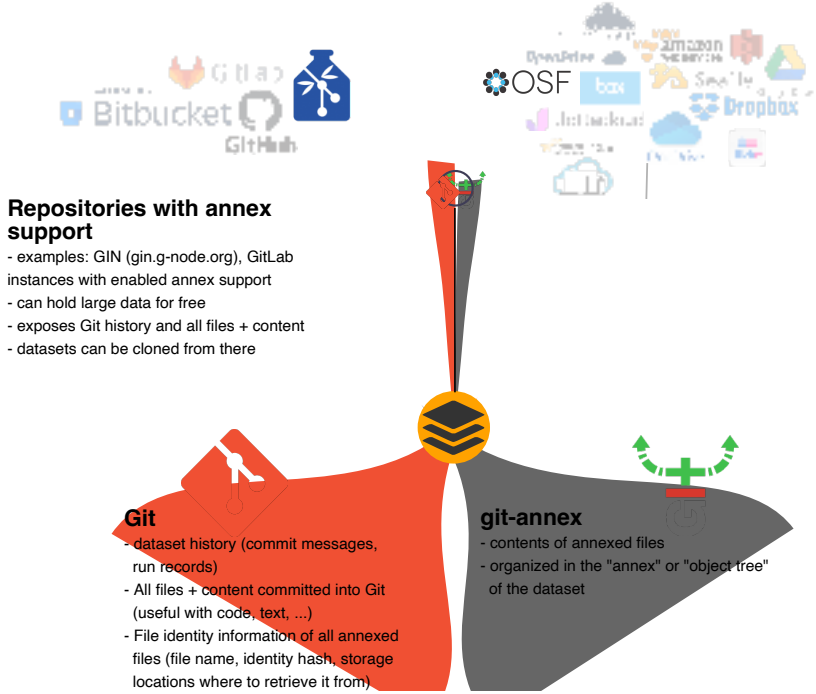
- DataLad datasets are exposed via private or public repositories on a repository hosting service (e.g., GitLab or GitHub)
- Data can't be stored in the repository hosting service but can be kept in almost any third party storage
- Publication dependencies automate interactions between both paces



from the DataLad Handbook by Wagner et al. (2022) (CC BY-SA 4.0)

Special cases

Repositories with annex support

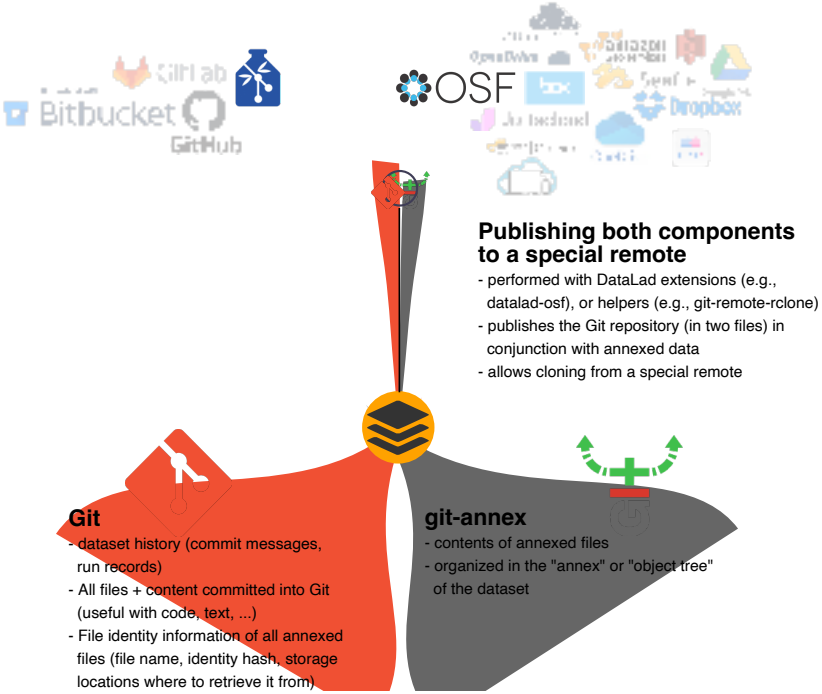


from the DataLad Handbook by Wagner et al. (2022) (CC BY-SA 4.0)

- Easy: Only one remote repository
- Examples: GIN, GitLab with annex support

gin.g-node.org

Special remotes with repositories



from the DataLad Handbook by Wagner et al. (2022) (CC BY-SA 4.0)

- Flexible: Full history or single snapshot
- Examples: DataLad-OSF

DataLad-OSF

Have access to more data than you have disk-space

Cloned datasets are lean.

```
datalad clone git@gin.g-node.org:/lnnrtwttkhn/neuro-data.git
install(ok): /tmp/neuro-data (dataset)
cd neuro-data && du -sh
212K
```

“Metadata” (file names, availability) are present …

```
tree
.
├── CHANGES
├── README
├── dataset_description.json
├── sub-01
│   ├── anat
│   │   └── sub-01_T1w.nii
│   └── func
│       ├── sub-01_task-auditory_bold.nii
│       └── sub-01_task-auditory_events.tsv
└── task-auditory_bold.json

4 directories, 7 files
```

… but no file content:

```
open README
The file /tmp/README does not exist.
```

File contents can be retrieved on demand:

```
datalad get .
get(ok): CHANGES (file) [from origin...]
get(ok): README (file) [from origin...]
get(ok): dataset_description.json (file) [from origin...]
get(ok): sub-01/anat/sub-01_T1w.nii (file) [from origin...]
get(ok): sub-01/func/sub-01_task-auditory_bold.nii (file) [from origin...]
get(ok): sub-01/func/sub-01_task-auditory_events.tsv (file) [from origin...]
action summary:
  get (ok: 6)
```

Let’s check the dataset size again:

```
du -sh
49M
```

Drop file content that is not needed:

```
datalad drop .
drop(ok): CHANGES (file) [locking origin...]
drop(ok): README (file) [locking origin...]
drop(ok): dataset_description.json (file) [locking origin...]
drop(ok): sub-01/anat/sub-01_T1w.nii (file) [locking origin...]
drop(ok): sub-01/func/sub-01_task-auditory_bold.nii (file) [locking origin..
drop(ok): sub-01/func/sub-01_task-auditory_events.tsv (file) [locking origin
drop(ok): . (directory)
action summary:
  drop (ok: 7)
```

When files are dropped, only “metadata” stays behind, and files can be re-obtained on demand.

Data sharing using DataLad and data infrastructure of the Max Planck Society

Sharing DataLad datasets via Keeper

KEEPER

keeper.mpdl.mpg.de

*"A free service for all Max Planck employees and project partners with **more than 1TB of storage per user** for your researchdata."*

✨Features of Keeper ✨

- > 1 TB per Max Planck employee (and expandable):
- based on cloud-sharing service Seafile
- data hosted on MPS servers
- configurable as a DataLad special remote

1. Configure rclone

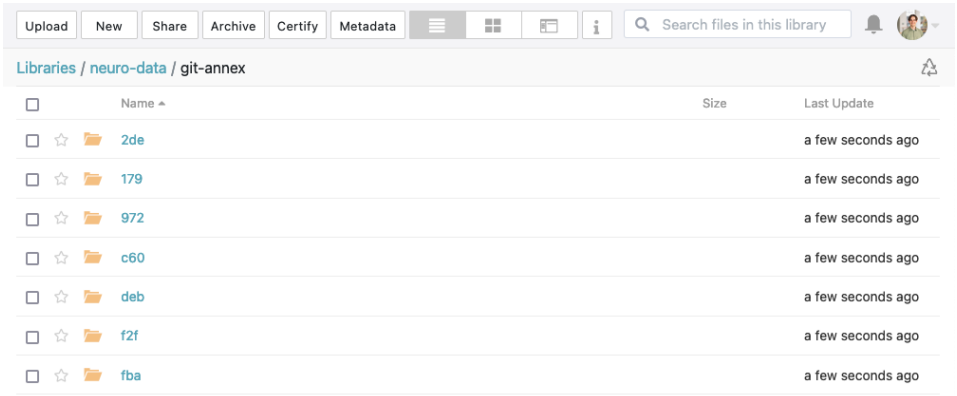
```
rclone config create neuro-data seafile \
url https://keeper.mpdl.mpg.de/ user wittkuhn@mpib-berlin.mpg.de \
library neuro-data pass supersafepassword
```

2. Create a library on Keeper and a Keeper sibling

```
git annex initremote keeper type=external externaltype=rclone \
chunk=50MiB encryption=none target=neuro-data
```

3. Push dataset to Keeper

```
datalad push --to keeper
```



Sharing DataLad datasets via Edmond

edmond.mpg.de

"Edmond is a research data repository for Max Planck researchers. It is the place to store completed datasets of research data with open access."

✨Features of Edmond ✨

- based on Dataverse, hosted on MPS servers
- use is free of charge
- no storage limitation (on datasets or individual files)
- flexible licensing

Two modes:

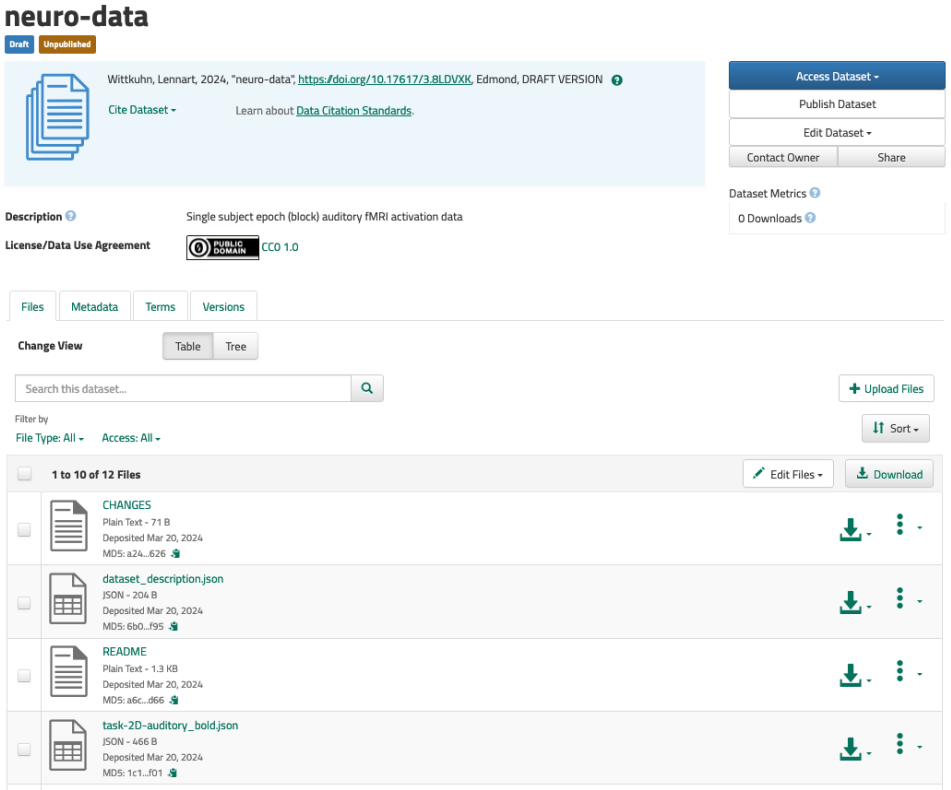
1. **annex mode** (default): non-human readable representation of the dataset that includes Git history and annexed data
2. **filetree mode**: human readable single snapshot of your dataset "as it currently is" that does not include history of annexed files (but Git history)

1. Create a Dataverse sibling for Edmond:

```
datalad add-sibling-dataverse https://edmond.mpg.de/ \
doi:10.17617/3.8LDVXK --mode filetree
```

2. Push dataset to Edmond / Dataverse

```
datalad push --to dataverse
```



Sharing DataLad datasets via ownCloud / Nextcloud

owncloud.gwdg.de

nextcloud.com

DataLad NEXT extension allows to push / clone DataLad datasets to / from ownCloud & Nextcloud (via WebDAV)

> **ownCloud GWDG:** "*50 GByte default storage space per user; flexible increase possible upon request*"

✨Features of ownCloud and Nextcloud ✨

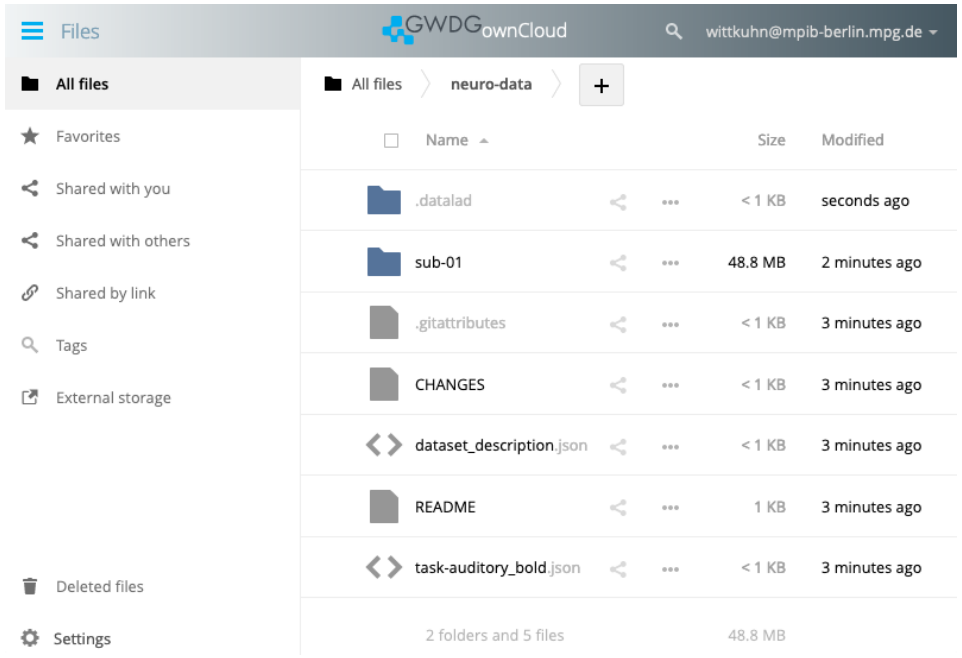
- data privacy compliant alternative to Google Drive, Dropbox, etc. (usually hosted on-site)
- provided by your institution, so free to use
- supports private and public repositories
- can be used together with external collaborators
- expose datasets for regular download without DataLad

1. Create a WebDAV sibling:

```
datalad create-sibling-webdav --dataset . \
  --name owncloud-gwdg --mode filetree \
  'https://owncloud.gwdg.de/remote.php/nonshib-webdav/neuro-data'
```

2. Push dataset to ownCloud

```
datalad push --to owncloud-gwdg
```



Access the dataset on owncloud.gwdg.de

Sharing DataLad datasets via GIN

"GIN is […] a web-accessible repository store of your data ***based on git and git-annex*** *that you can access securely anywhere you desire while keeping your data in sync, backed up and easily accessible […]"*

gin.g-node.org

✨Features of GIN ✨

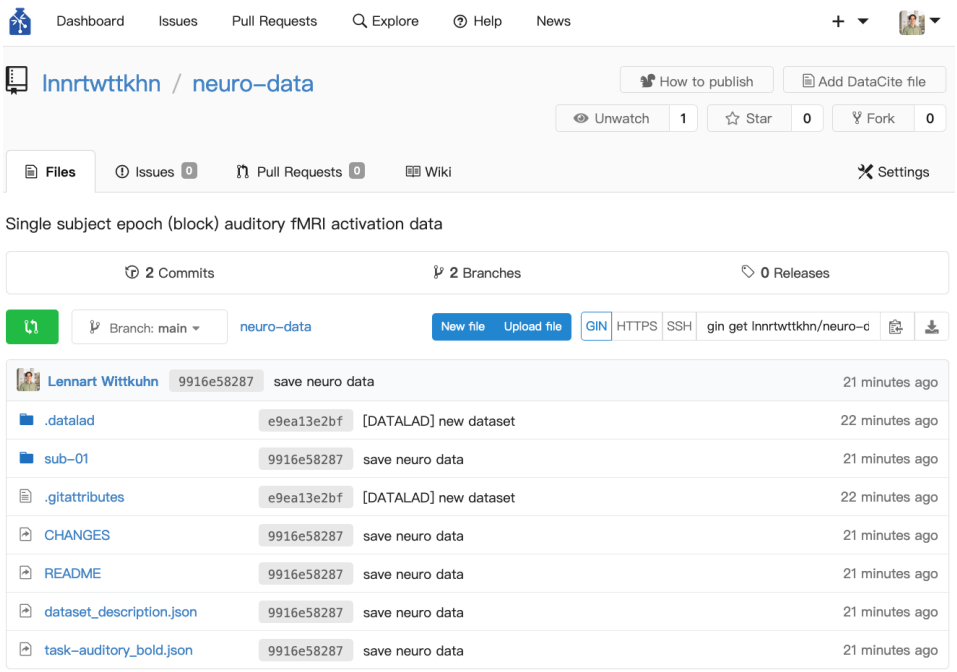
- free to use and open-source (could be hosted within your institution; for more details, see here)
- currently unlimited storage capacity and no restrictions on individual file size
- supports private and public repositories
- publicly funded by the Federal Ministry of Education and Research (BMBF; details here)
- servers on German land (Munich, Germany; cf. GDPR)
- provides Digital Object Identifiers (DOIs) (details here) and allows free licensing (details here)

1. Create a GIN sibling

```
datalad siblings add --dataset . \
--name gin --url git@gin.g-node.org:/lnnrtwttkhn/neuro-data.git
```

2. Push dataset to GIN

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



Access the dataset on GIN

Sharing DataLad datasets via GitLab

"GitLab is open source software to collaborate on code. Manage git repositories with fine-grained access controls that keep your code secure."

gitlab.com

GitLab for Max Planck employees

- hosted by GWDG: gitlab.gwdg.de
- hosted by your institute, e.g., git.mpib-berlin.mpg.de

✨Features of GitLab ✨

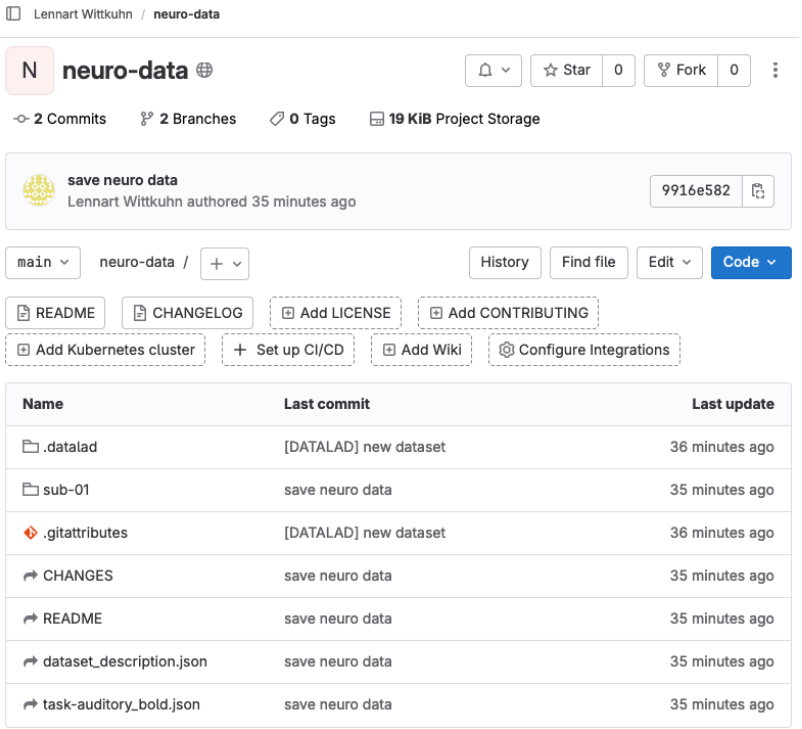
- free to use and open-source
- several MPS instances available (see above)
- supports private and public repositories
- use project management infrastructure (merge requests, issue boards, etc.) for your dataset projects

1. Create a GitLab sibling

```
datalad siblings add --dataset . --name gitlab \
--url git@git.mpib-berlin.mpg.de:wittkuhn/neuro-data.git
```

2. Push dataset *metadata* to GitLab

```
datalad push --to gitlab
```



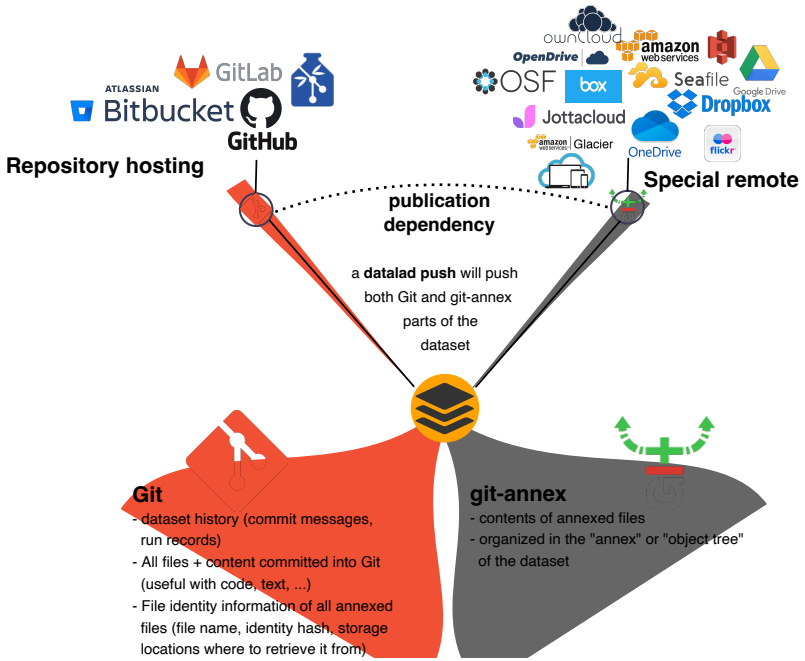
Access the dataset on GitLab

Publish and consume datasets like source code

Datasets can *comfortably* live in multiple locations:

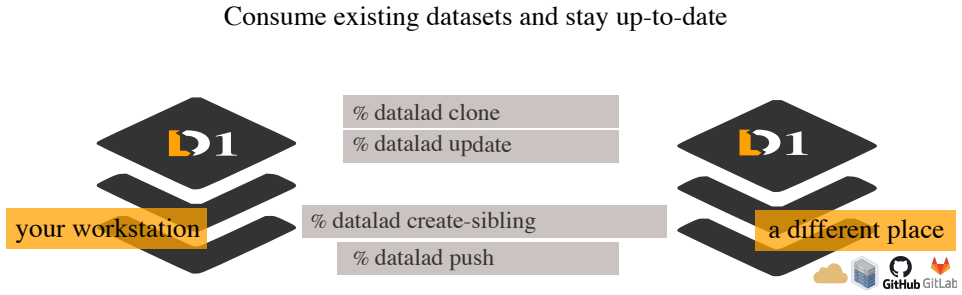
```
datalad siblings
.: here(+) [git]
.: owncloud-gwdg(+) [git]
.: dataverse(+) [dataverse]
.: gin(+) [https://gin.g-node.org/lnnrtwttkhn/neuro-data (git)]
.: keeper(+) [rclone]
.: gitlab(-) [git@git.mpib-berlin.mpg.de:wittkuhn/neuro-data.git (git
```

Publication dependencies automate update in all places:




```
datalad siblings configure --name gitlab --publish-depends SIBLING
```

Redundancy: DataLad gets data from available sources



from the DataLad Handbook by Wagner et al. (2022) (CC BY-SA 4.0)

Clone the dataset from GitLab:

```
datalad clone https://git.mpib-berlin.mpg.de/wittkuhn/neuro-data
```

Access to special remotes needs to be configured:

```
[INFO   ] access to 3 dataset siblings keeper, dataverse-storage,
owncloud-gwdg-storage not auto-enabled, enable with:
|       datalad siblings -d "/tmp/neuro-data" enable -s SIBLING
```

DataLad retrieves data from available sources (here, GIN):

```
datalad get .
get(ok): CHANGES (file) [from gin-src...]
get(ok): README (file) [from gin-src...]
[...]
```

Summary

Summary and discussion

Science is complex

- Scientific units are not static: We need version control
- Science is modular: We need to link modular datasets
- Science is iterative: We need to establish provenance
- Science is collaborative and distributed: We want to share our work and integrate with diverse infrastructure

DataLad: Decentralized management of digital objects for open science

- DataLad can version control arbitrary datasets
- DataLad links modular version-controlled datasets
- DataLad establishes provenance and transparency
- DataLad integrates with diverse infrastructure

Develop *everything* like source code

- Code and data *management* using **Git** and **DataLad** (free, open-source command-line tools)
- Code and data *sharing* via flexible repository hosting services (**GitLab, GitHub, GIN**, etc.)
- Code and data *storage* on various infrastructure (**GIN**, **OSF**, **S3**, **Keeper**, **Dataverse**, and many more!)
- Project-related communication (ideas, problems, discussions) via **issue boards** on GitLab / GitHub etc.
- Transparent contributions to code and data via **merge requests** on GitLab (i.e., pull requests on GitHub)
- **Reproducible procedures** using datalad run, rerun, and containers-run commands (also Make etc.)
- *Reproducible computational environments using software containers (e.g., Docker, Apptainer, etc.)*

✨**Towards science as distributed, open-source ~~software~~ *knowledge* development** ✨(cf. McElreath, 2020, 2023)

Overview of learning resources

Learn Git

- "Pro Git" by Scott Chacon & Ben Straub
- "Happy Git and GitHub for the useR" by Jenny Bryan, the STAT 545 TAs & Jim Hester
- "Version Control" by The Turing Way
- "Version Control with Git" by The Software Carpentries
- "Version control" (chapter 3 of "Neuroimaging and Data Science") by Ariel Rokem & Tal Yarkoni

Learn DataLad

- "Datalad Handbook" by the DataLad team / Wagner et al., 2022, *Zenodo*
- "Research Data Management with DataLad" | Recording of a full-day workshop on YouTube
- Datalad on YouTube | Recorded workshops, tutorials and talks on DataLad

Learn both (disclaimer: shameless plug 🙈)

Full-semester course on "Version control of code and data using Git and DataLad" (v2.0!) in summer semester 2024 at University of Hamburg (generously funded by the Digital and Data Literacy in Teaching Lab program) with many open educational resources (online guide, quizzes and exercises)

References

Gorgolewski, Krzysztof J., Tibor Auer, Vince D. Calhoun, R. Cameron Craddock, Samir Das, Eugene P. Duff, Guillaume Flandin, et al. 2016. “The Brain Imaging Data Structure, a Format for Organizing and Describing Outputs of Neuroimaging Experiments.” *Scientific Data* 3 (1). https://doi.org/10.1038/sdata.2016.44.

The Turing Way Community, and Scriberia. 2024. “Illustrations from the Turing Way: Shared Under CC-BY 4.0 for Reuse,” January. https://doi.org/10.5281/ZENODO.3332807.

Wagner, Adina S. 2024. “DataLad: Decrentalized Management of Digital Objects for Open Science.” *Zenodo*, January. https://doi.org/10.5281/ZENODO.10556597.

Wagner, Adina S., Laura K. Waite, Kyle Meyer, Marisa K. Heckner, Tobias Kadelka, Niels Reuter, Alexander Q. Waite, et al. 2022. “The DataLad Handbook,” April. https://doi.org/10.5281/ZENODO.6463273.

Thank you!

Dr. Lennart Wittkuhn

wittkuhn@mpib-berlin.mpg.de
https://lennartwittkuhn.com/
Mastodon GitHub LinkedIn

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MAX PLANCK INSTITUTE
FOR HUMAN DEVELOPMENT

Slides: https://lennartwittkuhn.com/talk-mps-fdm-2024

Source: https://github.com/lnnrtwttkhn/talk-mps-fdm-2024

Software: Reproducible slides built with Quarto and deployed to GitHub Pages using GitHub Actions for continuous integration & deployment

Contact: Feedback or suggestions via email or GitHub issues. Thank you!