


# Data management

## Session 01

Adina S. Wagner

 @AdinaWagner

Michael Hanke

 @eknahm

---

Psychoinformatics lab, Jülich Research Center

# AGENDA

- **Prerequisites & technicalities**
  - Setting up a Git identity
  - Gitlab/Github accounts
  - Howto: Issues on Gitlab, The handbook's repository, Readthedocs
- **Session 01**
  - What is a DataLad dataset?
  - YODA principles for dataset organization
- **Hands-on 01: Data in a DataLad dataset**

# GIT IDENTITY

- Git: free & open source version control software
- Git is already installed on the cluster
- Git identity: The name and e-mail address associated with what you "save".
- Configuration with `git config` command in a terminal:

```
$ git config --global user.name "Adina Wagner"  
$ git config --global user.email adina.wagner@example.com
```

- **Hands-on:** Log into `brainbfast` and configure your Git identity, if it isn't set up yet.

# GITHUB & GITLAB

- Two different web-based Git repository managers with similar features. Both are used to host Git repositories and to ease collaboration.



- **Github.com**: Most popular, proprietary, extensive functionality on free plans
- Core concepts: Repositories, organizations.

## GitLab

- Similar to GitHub, but is open source. The FZJ hosts many different GitLab instances.
- JuGit (**jugit.fz-juelich.de**): The GitLab instance we recommend.
- Core concepts: Groups, subgroups, projects

# HANDBOOK REPOSITORIES



- The DataLad Handbook: User-oriented, introductory course on DataLad and the basis for the data management course.
- Source code on
  - Github ([github.com/datalad-handbook/book](https://github.com/datalad-handbook/book)) and
  - Gitlab ([jugit.fz-jeulich.de/inm7/training/datalad-handbook](https://jugit.fz-jeulich.de/inm7/training/datalad-handbook))
- **File issues** if you have questions or requests!
- Contribute by **pull requesting** changes, additions, and fixes!

# FILING ISSUES ON GITLAB



- File issues if you have DataLad-related or course-related questions in the repository hosted on **Gitlab**
- **Hands-on:** File an issue right now!
  - go to [jugit.fz-juelich.de](https://jugit.fz-juelich.de)
  - find the handbook project
  - file an issue with any content



- The book is rendered with [Sphinx](#) and hosted on [Readthedocs.org](#)
- Readthedocs supports HTML, eReader, and PDF formats
- Rendered version: [handbook.datalad.org](#)
- There is an additional **INM-7 specific** version with additional sections on internal workflows
- **Hands-on:** Access public and INM-7-specific versions of the handbook in HTML and PDF format

# DATALAD DATASETS

DataLad datasets are DataLad's core data structure. Datasets have many features:

## **Version controlled content, regardless of size**

Relying on the tools Git and Git-annex working in the background.

## **Provenance tracking**

Record and find out how data came into existence (including the software environment), and reproduce entire analyses.

## **Easy collaboration**

Install others' datasets, share datasets, publish datasets with third-party services.

## **Staying up to date**

Datasets can know their copies or origins. This allows to **update** datasets from their sources with a single command.

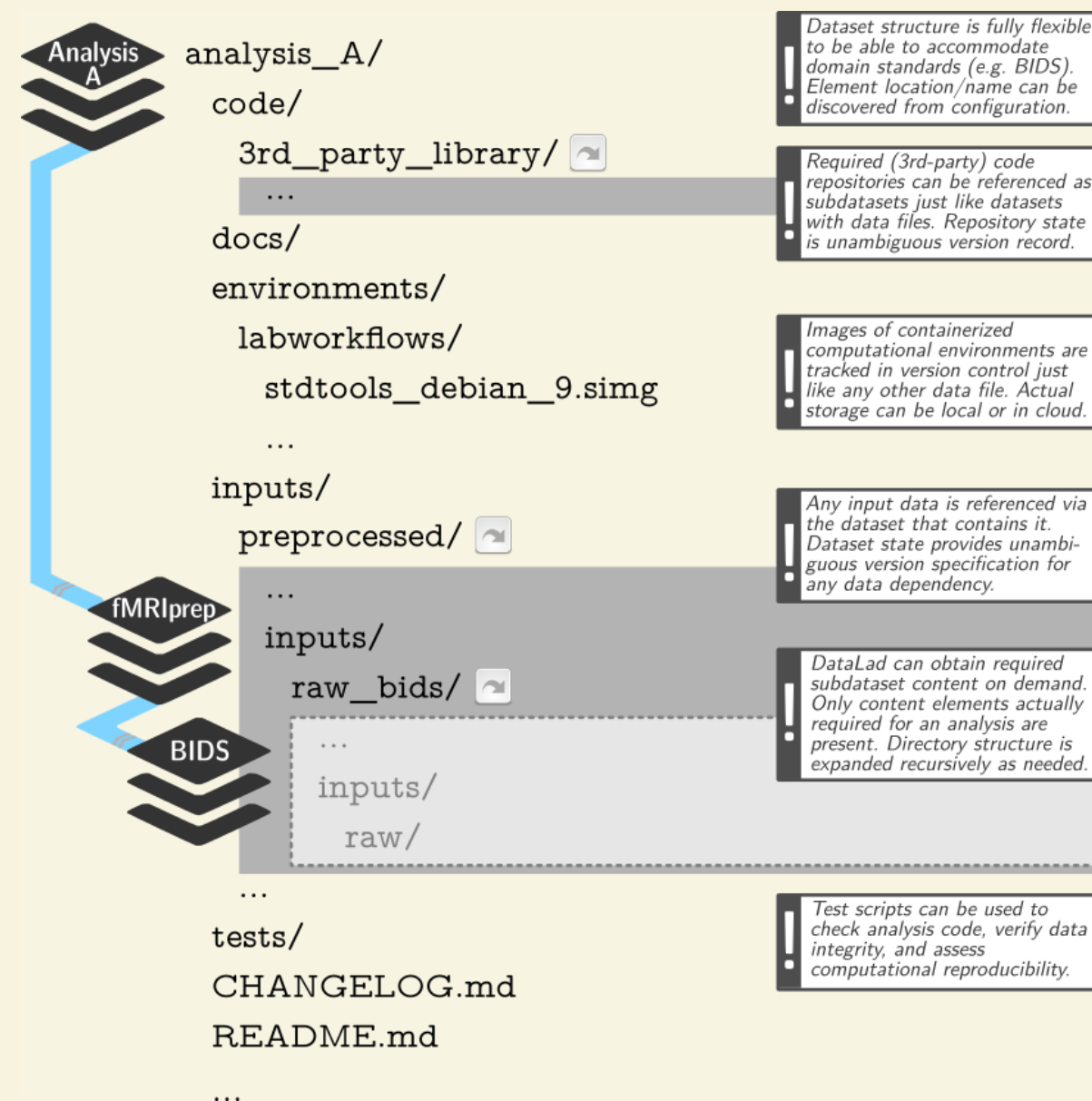
## **Modularity & Nesting**

Individual datasets are independent, versioned components that can be *nested* as *subdatasets* in *superdatasets*. Subdatasets have a stand-alone version history, and their *version state* is recorded in the superdataset.



# DATALAD DATASETS

DataLad datasets look like any other directory on your computer, and subdatasets look like subdirectories. DataLad, Git-annex, and Git work in the background (e.g., `.datalad/`, `.git/`)



You can **create & populate** a dataset from scratch, or **install** existing datasets from collaborators or open sources.



# DATALAD DATASETS FOR DATA ANALYSIS

- A DataLad dataset can have *any* structure, and use as many or few features of a dataset as required.
- However, for **data analyses** it is beneficial to make use of DataLad features and structure datasets according to the **YODA principles**:

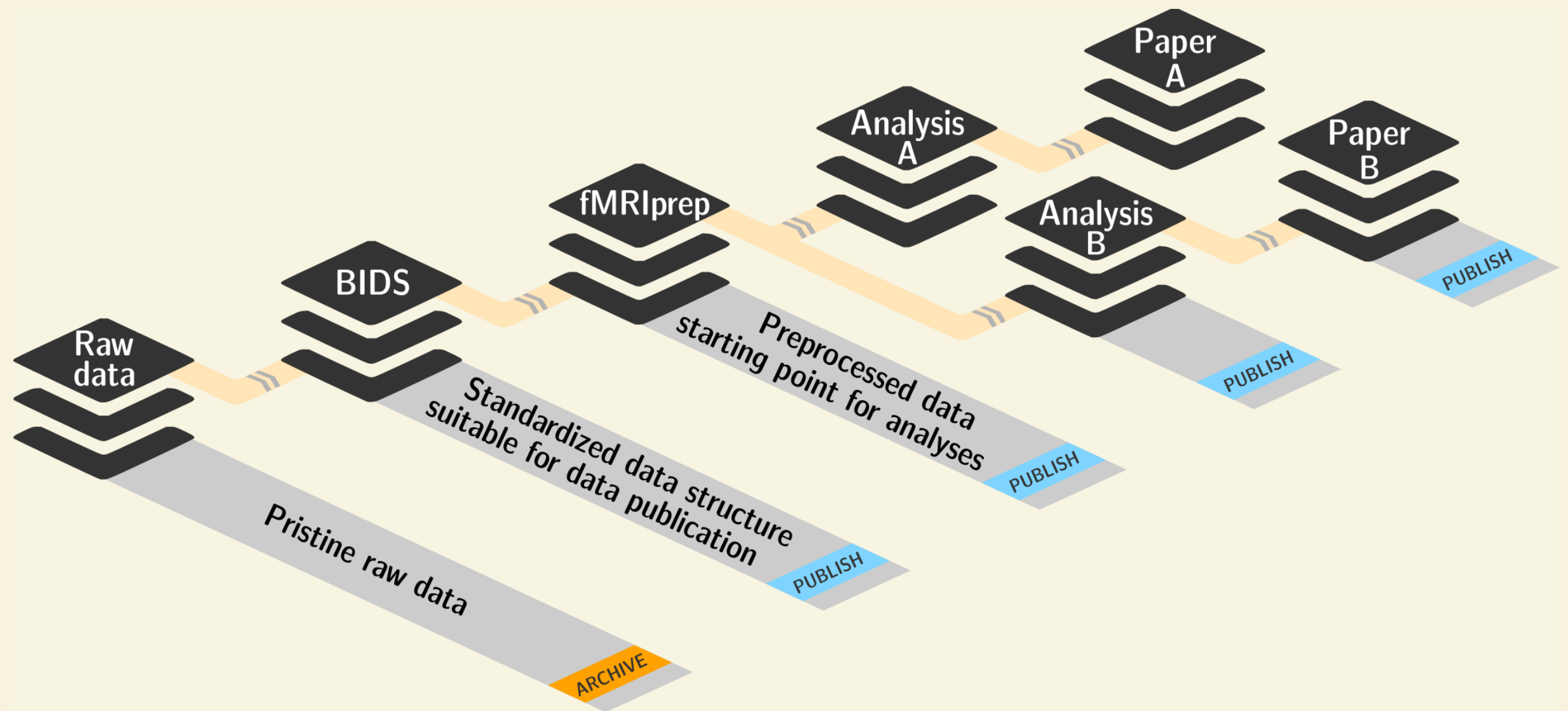


**P1: One thing, one dataset**

**P2: Record where you got it from, and where it is now**

**P3: Record what you did to it, and with what**

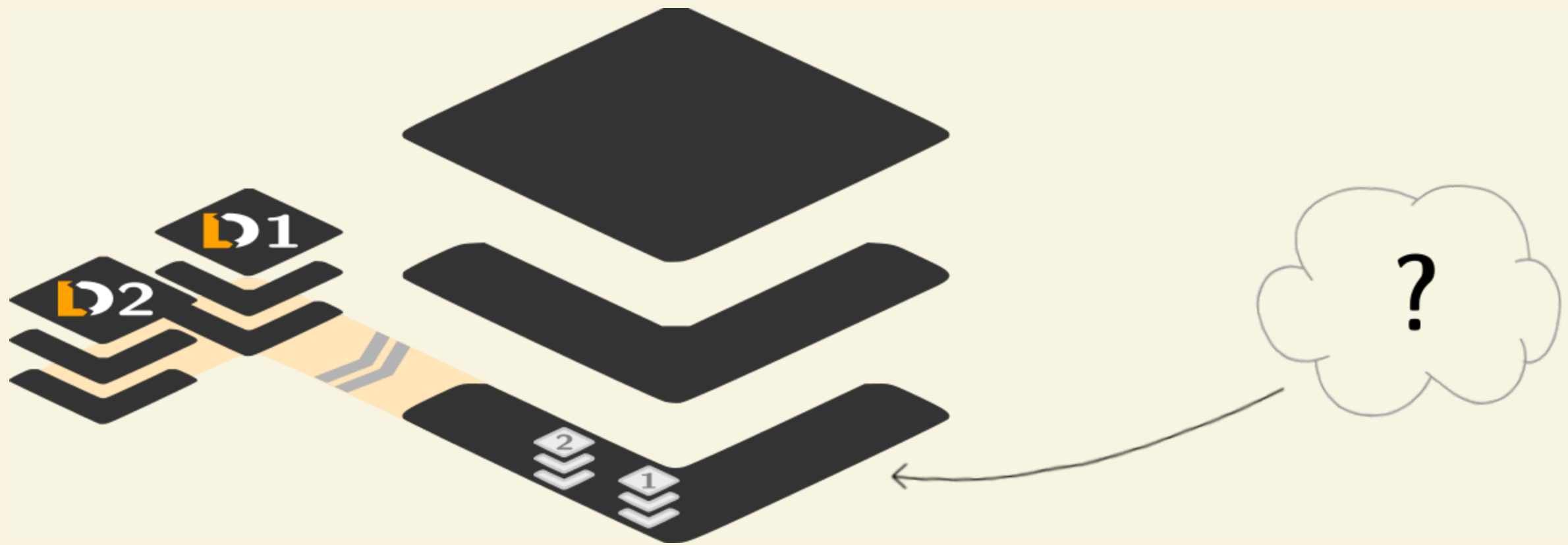
# P1: ONE THING, ONE DATASET



- Bundle all components of one analysis into one superdataset
- Whenever a particular collection of files could anyhow be useful in more than one context (e.g. data), put them in their own dataset, and install it as a subdataset.
- Keep everything clean and modular: Within an analysis, separate code, data, output, execution environments.

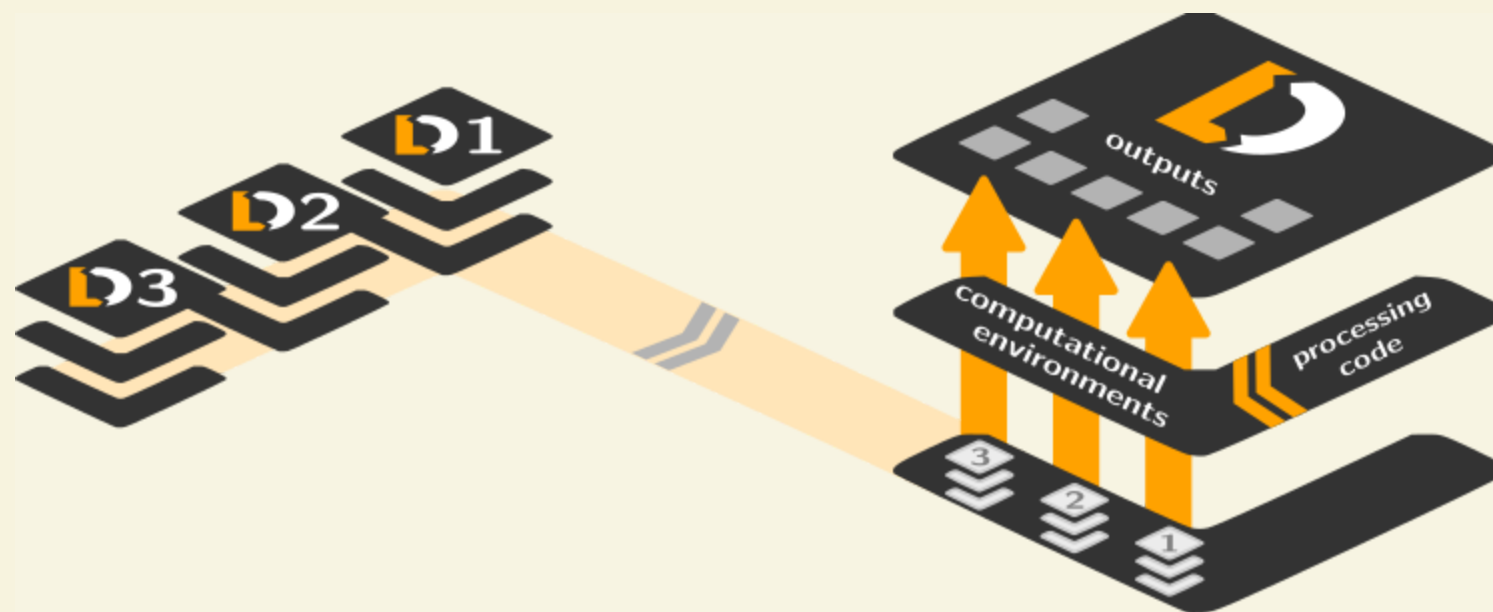


## P2: RECORD WHERE YOU GOT IT FROM, AND WE IT IS NOW



- Link individual datasets to declare data-dependencies (e.g. as subdatasets).
- Record data's origin with appropriate commands, for example to record access URLs for individual files obtained from (unstructured) sources "in the cloud".
- Keep a dataset self-contained with relative paths in scripts to subdatasets or files.
- Share and publish datasets to collaborate.

## P3: RECORD WHAT YOU DID TO IT, AND WITH WHAT



- Collect and store provenance of all contents of a dataset that you create (more on this in later sessions).

# HANDS-ON EXCERSISE

**Objective:** How would you get data into a dataset?

- Use [github.com/datalad/example-dicom-functional](https://github.com/datalad/example-dicom-functional) as test data. Download branch 1block as a **ZIP** archive.
- Log into `brainbfast`, get the data on `brainbfast`, and try to get this data into a DataLad dataset with a sensible structure suitable for data analysis.
- This excersise is meant for **exploration**:
  - use `data lad --help`, the handbook, or the documentation at [docs.datalad.org](https://docs.datalad.org) to find out about available commands to solve this task,
  - use tools of your choice to download/extract data,
  - try to set up an appropriate dataset structure.



# HANDS-ON SOLUTION

- transform the zip folder into a DataLad dataset:

```
$ cd example_dicom_functional_block
$ datalad create -f
[INFO   ] Creating a new annex repo at [...]/example-dicom-functional-1block
create(ok): [...]/example-dicom-functional-1block (dataset)

$ datalad save -m "add dicoms from functional acquisition"
add(ok): LICENSE (file)
add(ok): dicoms/MR.1.3.46.670589.11.38317.5.0.4476.2014042516045740754 (file) [...]
```

- create a dataset for a data analysis (independent from the data directory)

```
$ cd ../
$ datalad create -c yoda mydataset
[INFO   ] Creating a new annex repo at [...]/myanalysis
[INFO   ] Running procedure cfg_yoda
[INFO   ] == Command start (output follows) =====
[INFO   ] == Command exit (modification check follows) =====
create(ok): [...]/myanalysis (dataset)
```

- create a data directory and install the dicom dataset as a subdataset

```
$ cd myanalysis
$ mkdir data
$ datalad install -d . -s ../example_dicom_functional_1block data/dicoms
[INFO   ] Cloning ../example-dicom-functional-1block into '[...]/myanalysis/data/dicoms'
install(ok): data/dicoms (dataset)
action summary:
  add (ok: 2)
  install (ok: 1)
  save (ok: 1)
```

**Hands-on: Explore this dataset**



# FURTHER READING

You will find the topics of this session in more detail in the following chapters of the handbook:

- **The basics on datasets:**
  - Chapter [DataLad Datasets](#) in the handbook.
- **Best practices for data analyses in datasets (YODA):**
  - The section [YODA principles](#) in the handbook.
- **A preview into automatically reproducible analyses in datasets:**
  - Usecase "[An automatically reproducible neuroimaging analysis of public data](#)" in the handbook.

# OUTLINE: WHAT COMES NEXT?

- DataLad is installed on the cluster, try it out further, and ask questions on GitLab
- Sessions will start with open question time about a past exercise, and end with an exercise for the upcoming session.
- Upcoming topics: Reproducible analysis, collaboration, INM-7 specific workflows on data retrieval & JSC
- **Which date is suitable?** [Doodle poll](#)

**QUESTIONS?**