BRAINHACK GLOBAL 2020 ANKARA

🧠💻

AN INTRODUCTION TO DATALAD

Adina Wagner
@AdinaKrik

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

DataLad can help with small or large-scale data management

DataLad can help with small or large-scale data management

Free,
open source,
command line tool & Python API

SOME DataLad BASICS

- A command-line tool, available for all major operating systems (Linux, macOS/OSX, Windows), MIT-licensed
- Build on top of Git and Git-annex
- **Allows...**

  ... version-controlling arbitrarily large content

  version control data and software alongside to code!

  ... transport mechanisms for sharing and obtaining data

  consume and collaborate on data (analyses) like software

  ... (computationally) reproducible data analysis

  Track and share provenance of all digital objects

  ... and *much* more
- Completely domain-agnostic

A FEW THINGS THAT DATALAD CAN HELP WITH

A FEW THINGS THAT DATALAD CAN HELP WITH

- Getting data
- Keeping a project clean and orderly
- Computationally reproducible data analysis

A FEW THINGS THAT DATALAD CAN HELP WITH

- Getting data
- Keeping a project clean and orderly
- Computationally reproducible data analysis

There is much more, and you can read about it in
The DataLad Handbook (handbook.datalad.org)

ACKNOWLEDGEMENTS

Software

- Michael Hanke
- Yaroslav Halchenko
- Joey Hess (git-annex)
- Kyle Meyer
- Benjamin Poldrack
- *26 additional contributors*

Documentation project

- Michael Hanke
- Laura Waite
- *28 additional contributors*

Funders

NSF 1429999

BMBF 01GQ1411

Collaborators

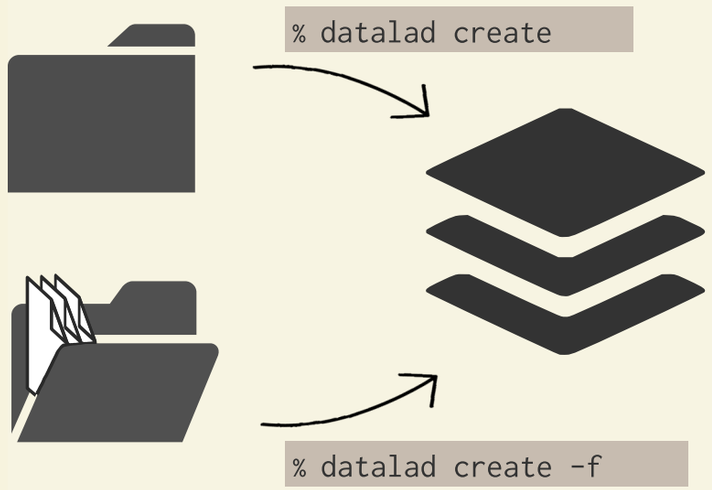
EVERYTHING HAPPENS IN DATALAD DATASETS

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

EVERYTHING HAPPENS IN DATALAD DATASETS

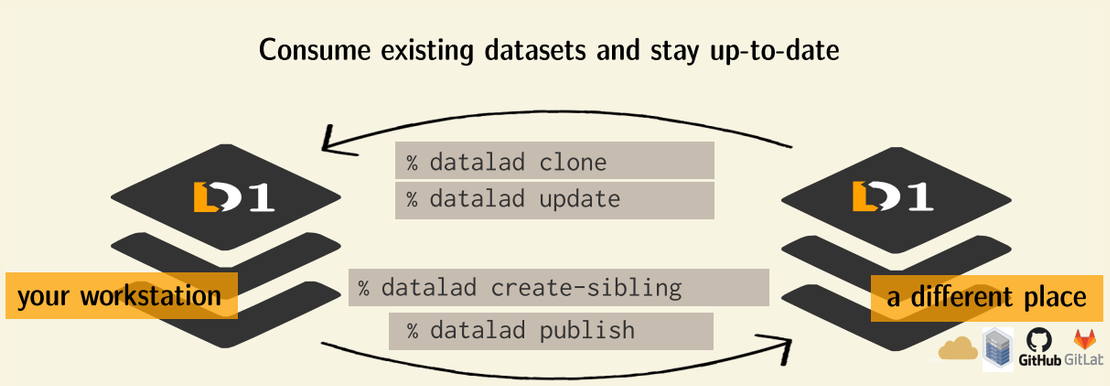
- DataLad's core data structure
  - Dataset = A directory managed by DataLad
  - A Git/git-annex repository
  - Any directory of your computer can be managed by DataLad.
  - Datasets can be *created* (from scratch) or *installed*

create new, empty datasets to populate...



.... or transform existing directories into datasets

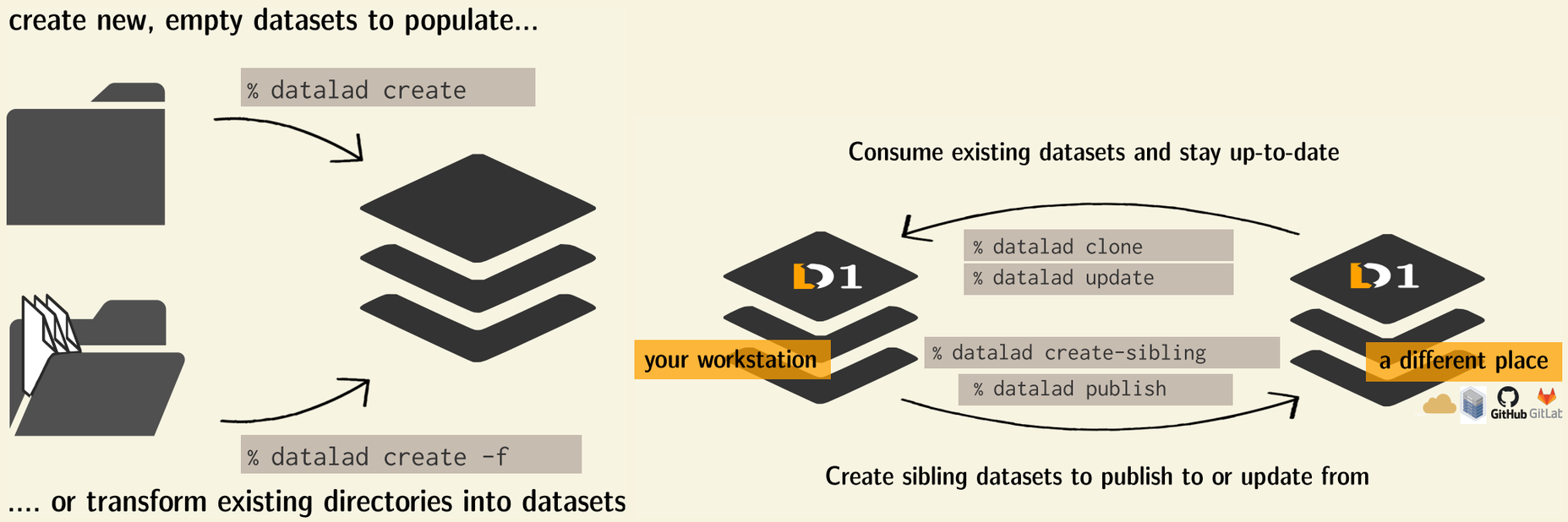
Consume existing datasets and stay up-to-date



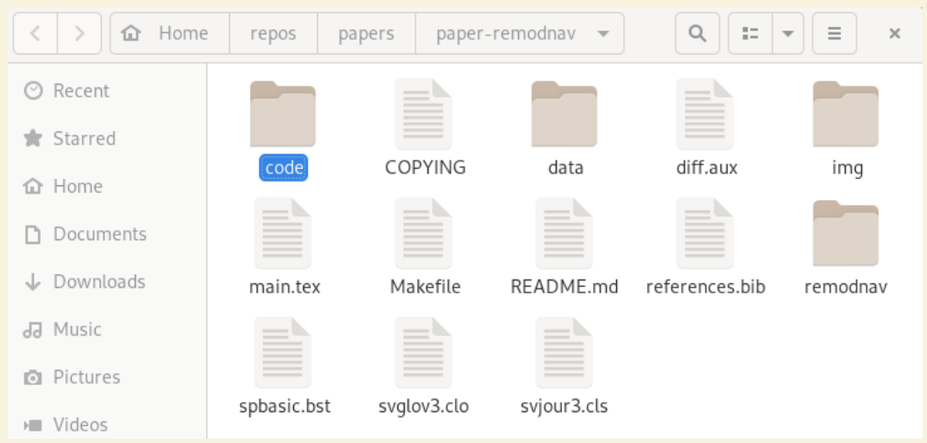
Create sibling datasets to publish to or update from

EVERYTHING HAPPENS IN DATALAD DATASETS

- DataLad's core data structure
  - Dataset = A directory managed by DataLad
  - A Git/git-annex repository
  - Any directory of your computer can be managed by DataLad.
  - Datasets can be *created* (from scratch) or *installed*



File viewer and terminal view of a DataLad dataset




```
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
```

USING DATALAD

- DataLad can be used from the command line

```
datalad create mydataset
```

- ... or with its Python API

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

USING DATALAD

- DataLad can be used from the command line

```
datalad create mydataset
```

- ... or with its Python API

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

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

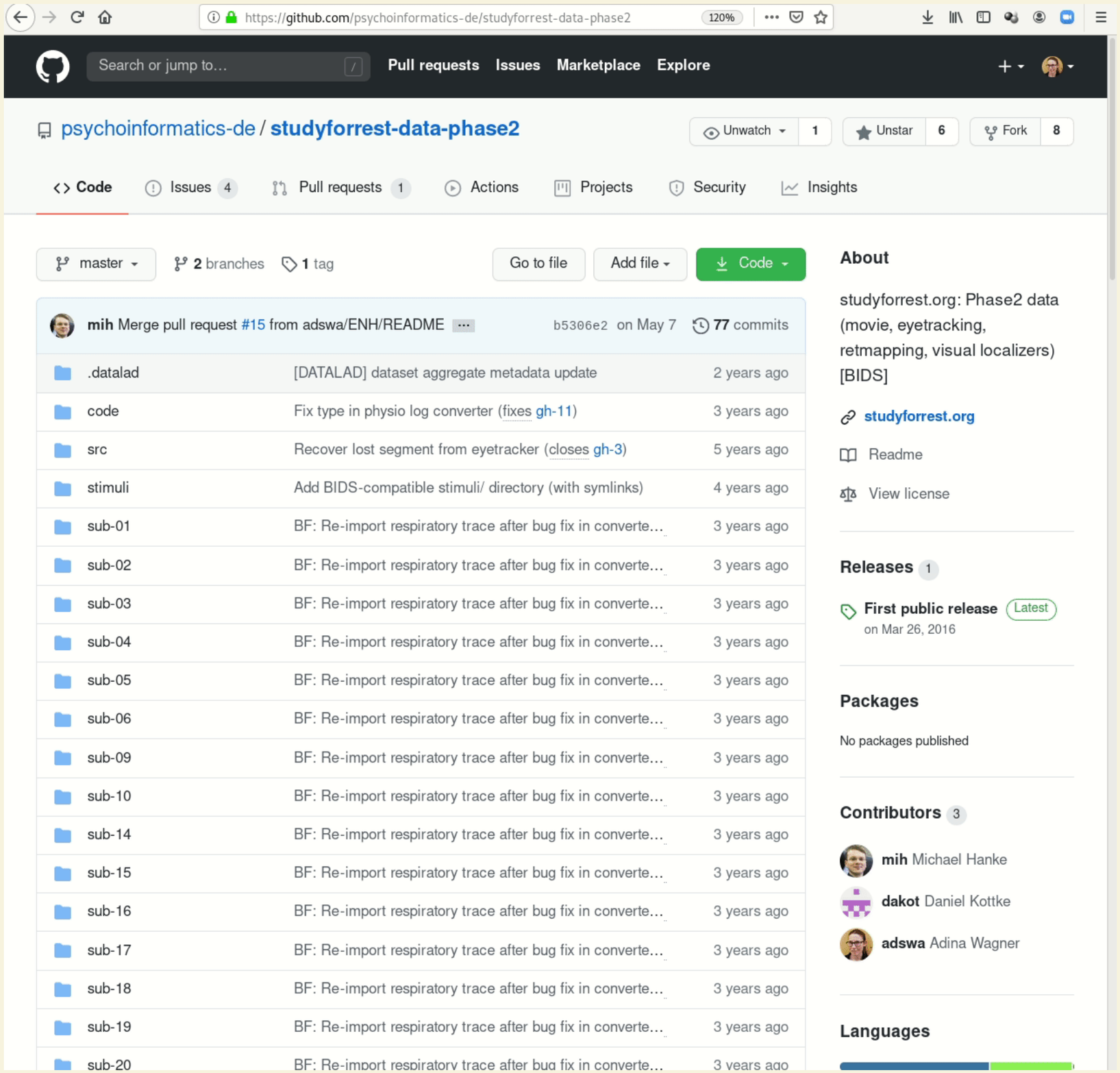
```
# in R
> system("datalad create mydataset")
```

GETTING DATA

- Datasets can be used to distribute data
- You can `clone` a dataset from a public or private place and get access to the data it tracks

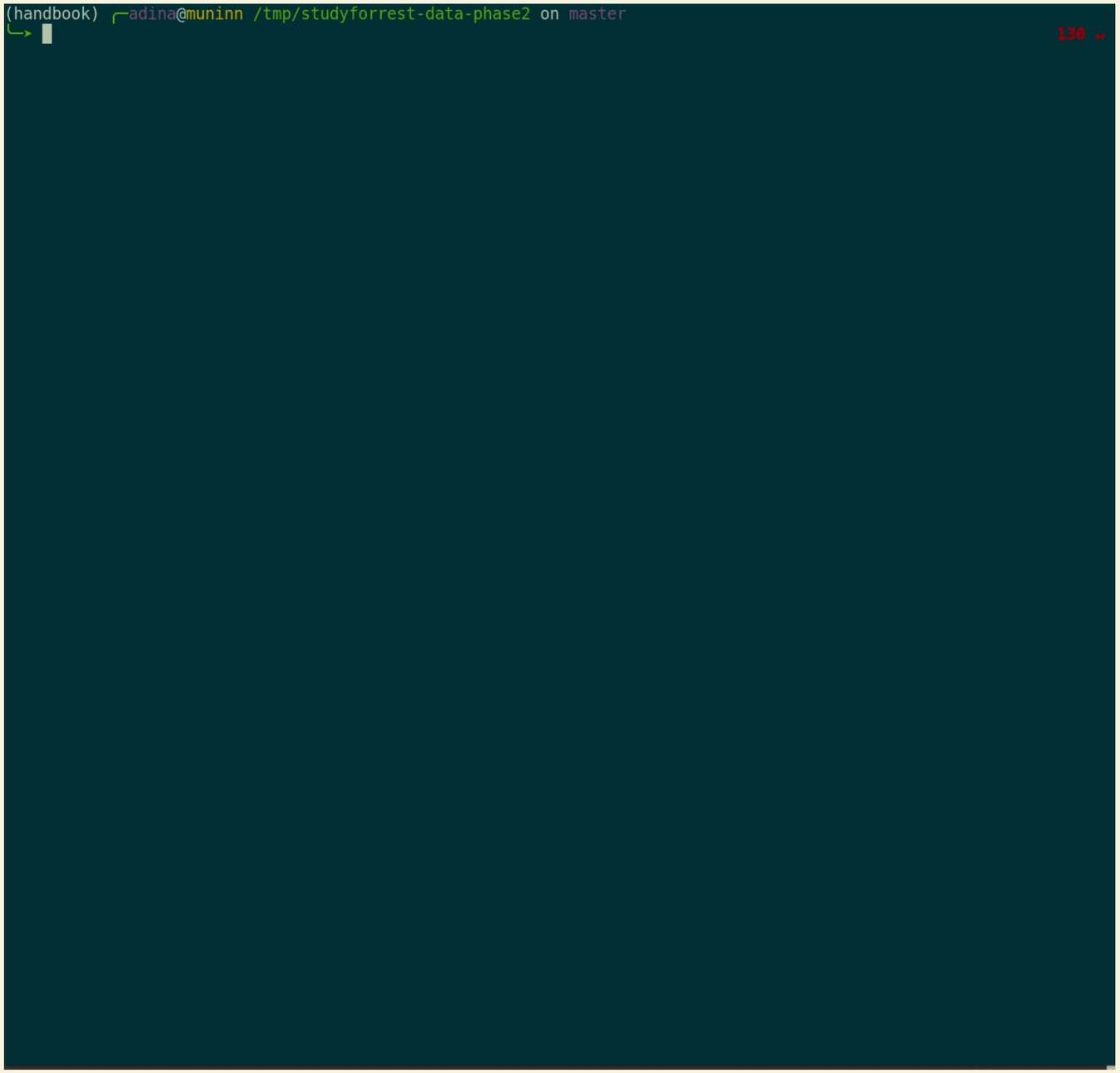
GETTING DATA

- Datasets can be used to distribute data
- You can `clone` a dataset from a public or private place and get access to the data it tracks

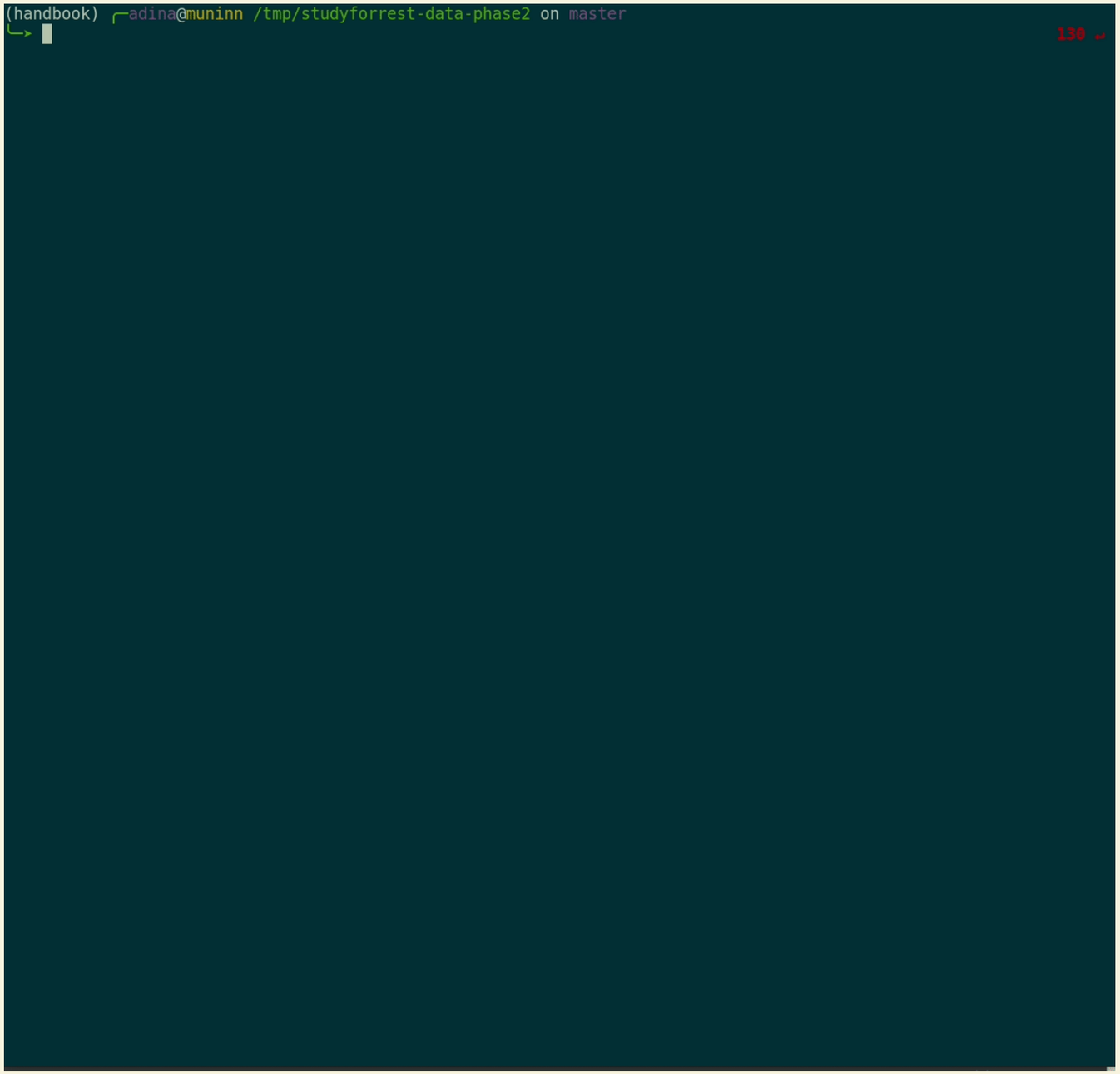


- Datasets are light-weight: Upon installation, only small files and meta data about file availability are retrieved, but **no file content**.

- Datasets are light-weight: Upon installation, only small files and meta data about file availability are retrieved, but **no file content**.



- Datasets are light-weight: Upon installation, only small files and meta data about file availability are retrieved, 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   .          # its tiny!
```

GETTING DATA

- A cloned dataset gets you access to plenty of data, but has only little disk-usage

GETTING DATA

- A cloned dataset gets you access to plenty of data, but has only little disk-usage
- Specific file contents can be retrieved on demand via `datalad get`:

GETTING DATA

- A cloned dataset gets you access to plenty of data, but has only little disk-usage
- Specific file contents can be retrieved on demand via `datalad get`:

```
$ 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_bold.nii.
```

GETTING DATA

- A cloned dataset gets you access to plenty of data, but has only little disk-usage
- Specific file contents can be retrieved on demand via `datalad get`:

```
$ 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_bold.nii.
```

- You can also drop file content if you don't need it anymore with `datalad drop`:

GETTING DATA

- A cloned dataset gets you access to plenty of data, but has only little disk-usage
- Specific file contents can be retrieved on demand via `datalad get`:

```
$ 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_bold.nii.
```

- You can also drop file content if you don't need it anymore with `datalad drop`:

```
$ 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_bold.nii
```

GETTING DATA

- A cloned dataset gets you access to plenty of data, but has only little disk-usage
- Specific file contents can be retrieved on demand via `datalad get`:

```
$ 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_bold.nii.
```

- You can also drop file content if you don't need it anymore with `datalad drop`:

```
$ 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_bold.nii
```

- Feature: Have access to more data than your computer has disk-space!

GETTING DATA

- A cloned dataset gets you access to plenty of data, but has only little disk-usage
- Specific file contents can be retrieved on demand via `datalad get`:

```
$ 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_bold.nii.
```

- You can also drop file content if you don't need it anymore with `datalad drop`:

```
$ 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_bold.nii
```

- Feature: Have access to more data than your computer has disk-space!

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

GETTING DATA

- You can get more than 200TB of public data with DataLad, for example...

GETTING DATA

- You can get more than 200TB of public data with DataLad, for example...
  - All OpenNeuro datasets: github.com/OpenNeuroDatasets

```
$ datalad clone https://github.com/OpenNeuroDatasets/ds003171.git
```

  - The human connectome project data (full, and in subsets): github.com/datalad-datasets/human-connectome-project-openaccess

```
$ datalad clone https://github.com/datalad-datasets/human-connectome-project-openaccess.git
```

  - ABIDE (I-II), INDI, ADH200, CORR, Healthy Brain Network SSI, and many more in the DataLad superdataset (datasets.datalad.org)

```
$ datalad clone ///
```

KEEPING A PROJECT CLEAN AND ORDERLY

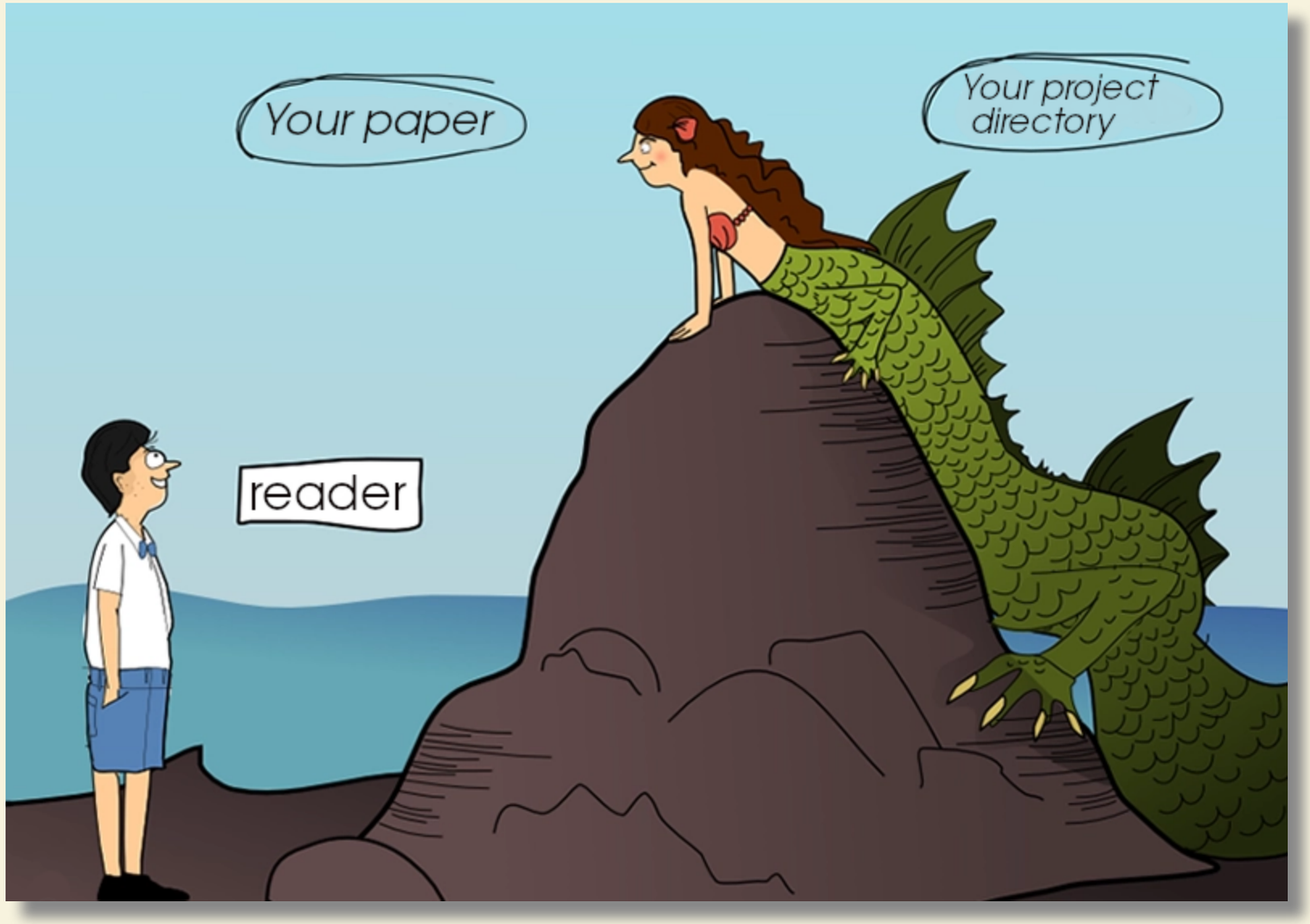


Image credit: adapted from https://dribbble.com/shots/3090048-Front-end-vs-Back-end

⬆

This a metaphor for most projects after publication

KEEPING A PROJECT CLEAN AND ORDERLY

Image credit: Poline et al., 2011

KEEPING A PROJECT CLEAN AND ORDERLY

Image credit: Poline et al., 2011

KEEPING A PROJECT CLEAN AND ORDERLY

- Much of neuroscientific research is computationally intensive, with complex workflows from raw data to result, and plenty of researchers degrees of freedom

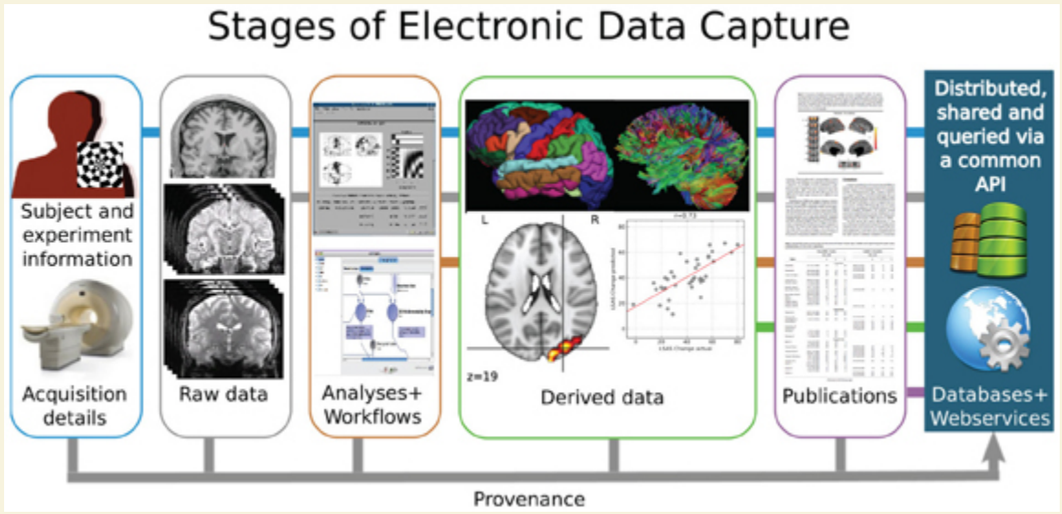


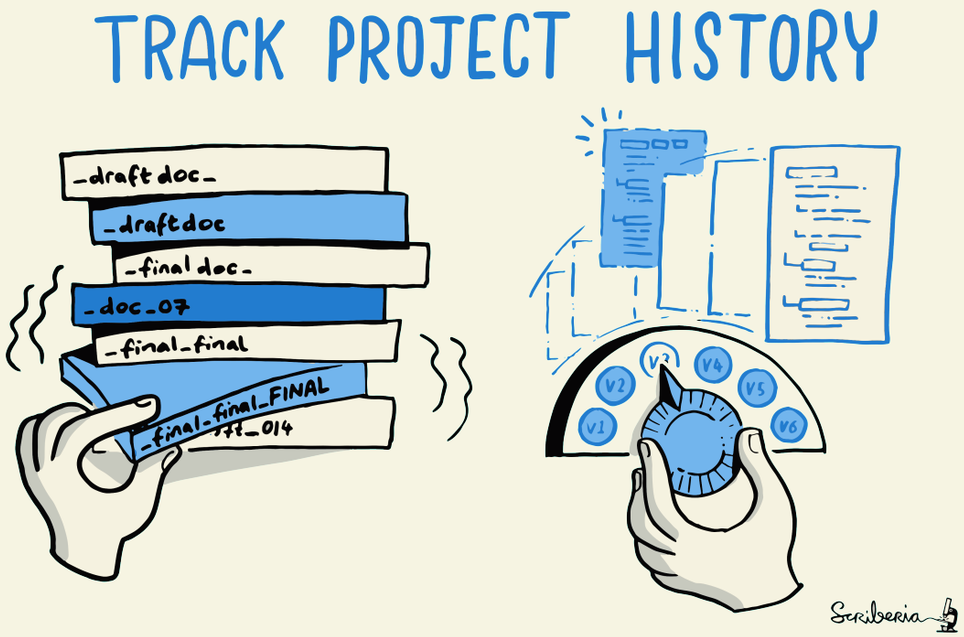
Image credit: Poline et al., 2011

COMPLEX ANALYSIS → CHAOTIC PROJECTS

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

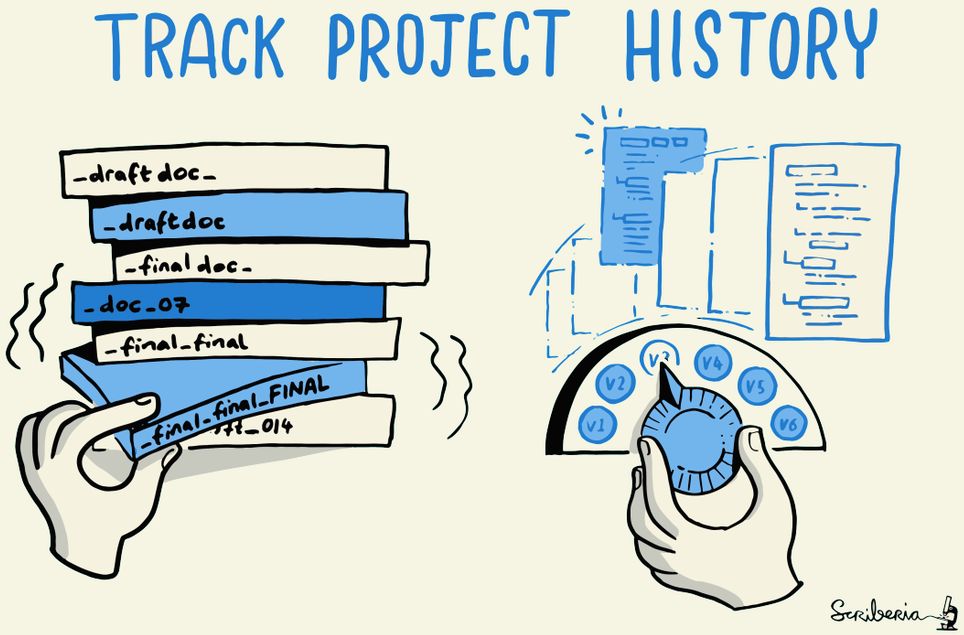
KEEPING A PROJECT CLEAN AND ORDERLY

KEEPING A PROJECT CLEAN AND ORDERLY



Version control

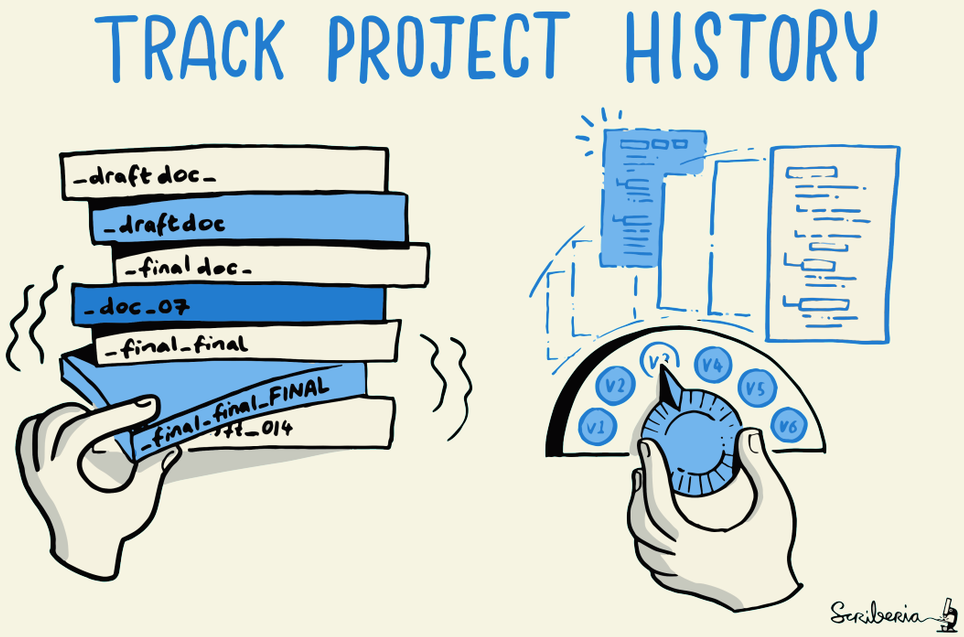
KEEPING A PROJECT CLEAN AND ORDERLY



Version control

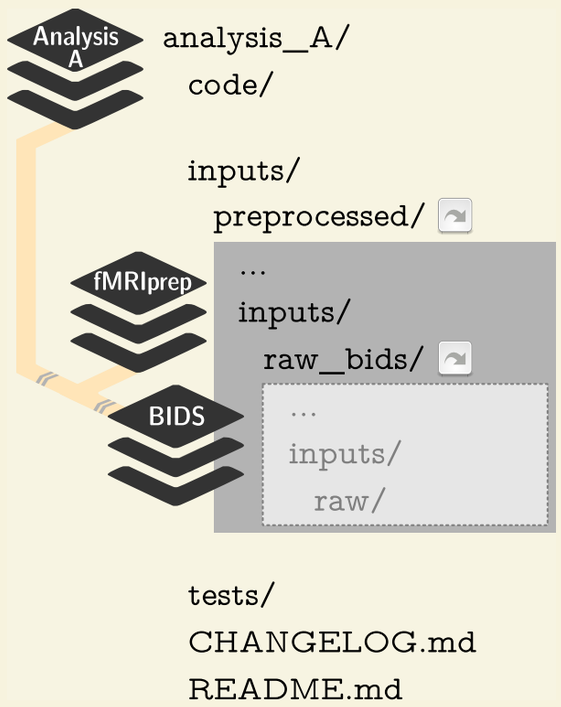
- keep things organized
- keep track of changes
- revert changes or go back to previous states

KEEPING A PROJECT CLEAN AND ORDERLY



Version control

- keep things organized
- keep track of changes
- revert changes or go back to previous states



Intuitive structure

KEEPING A PROJECT CLEAN AND ORDERLY

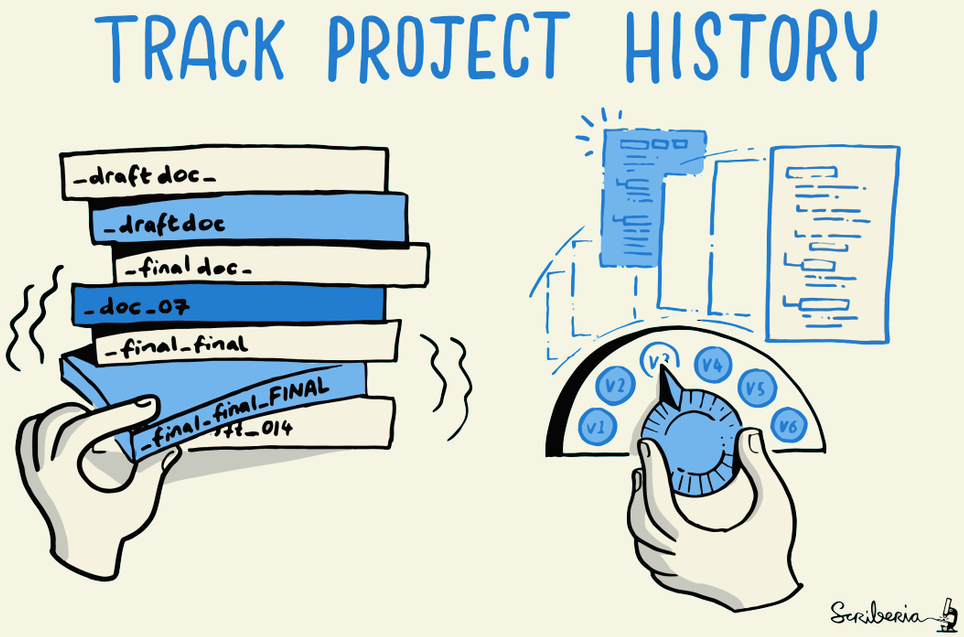
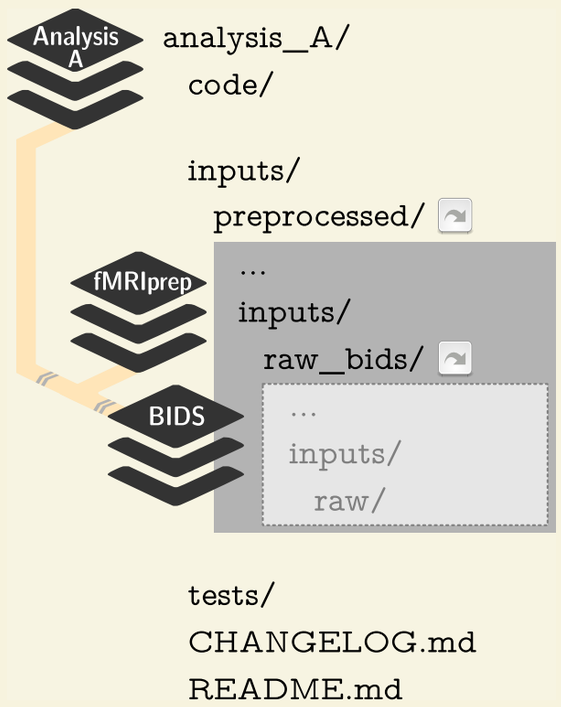


Image credit: CC-BY Scriberia & The Turing Way

Version control

- keep things organized
- keep track of changes
- revert changes or go back to previous states



Intuitive structure

- Follow the YODA principles

KEEPING A PROJECT CLEAN AND ORDERLY

First, let's create a new data analysis dataset with `datalad create`

```
$ 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` applies useful pre-structuring and configurations:

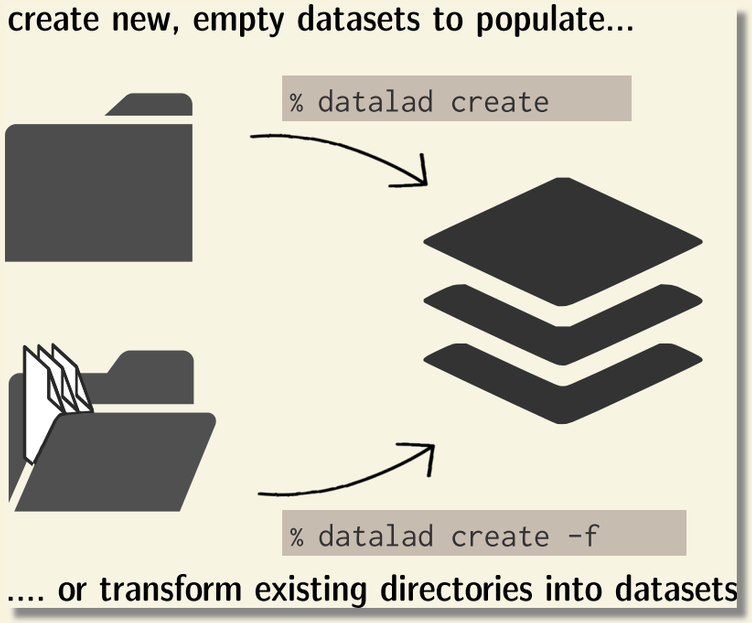
```
$ tree
.
├── CHANGELOG.md
├── code
│   └── README.md
└── README.md
```

VERSION CONTROL

- DataLad knows two things: Datasets and files

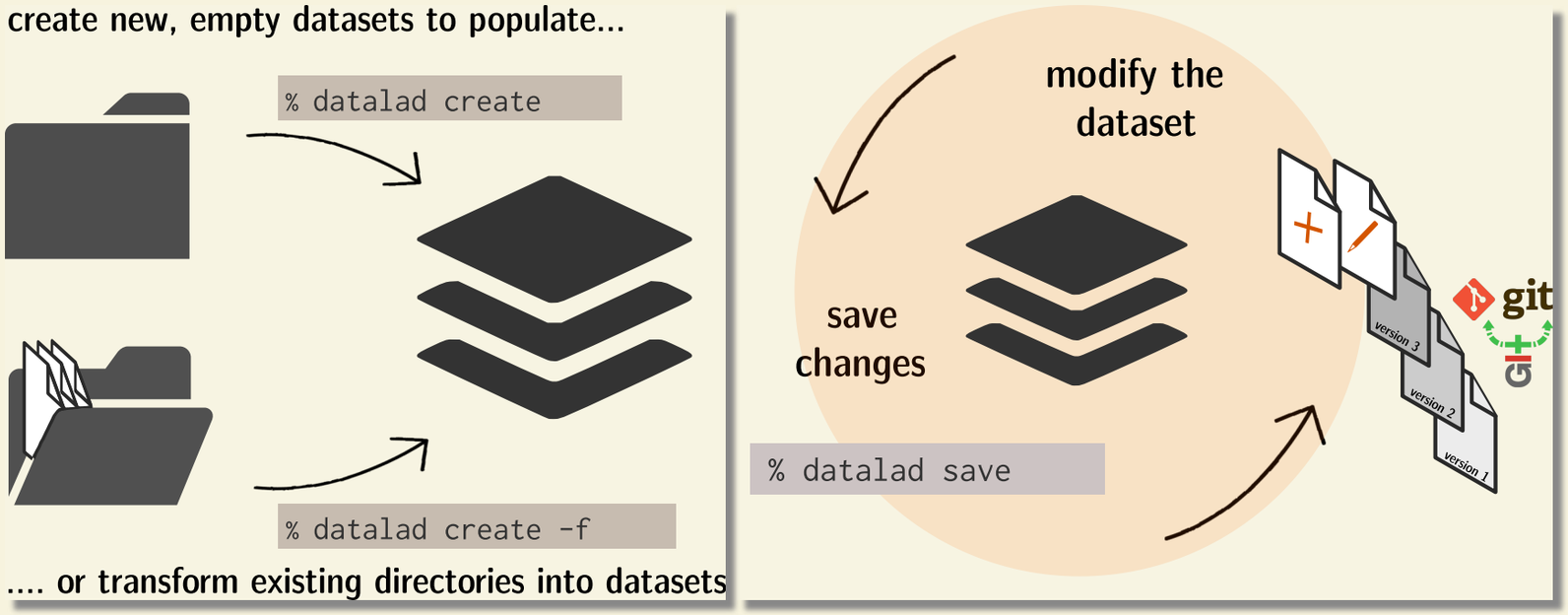
VERSION CONTROL

- DataLad knows two things: Datasets and files



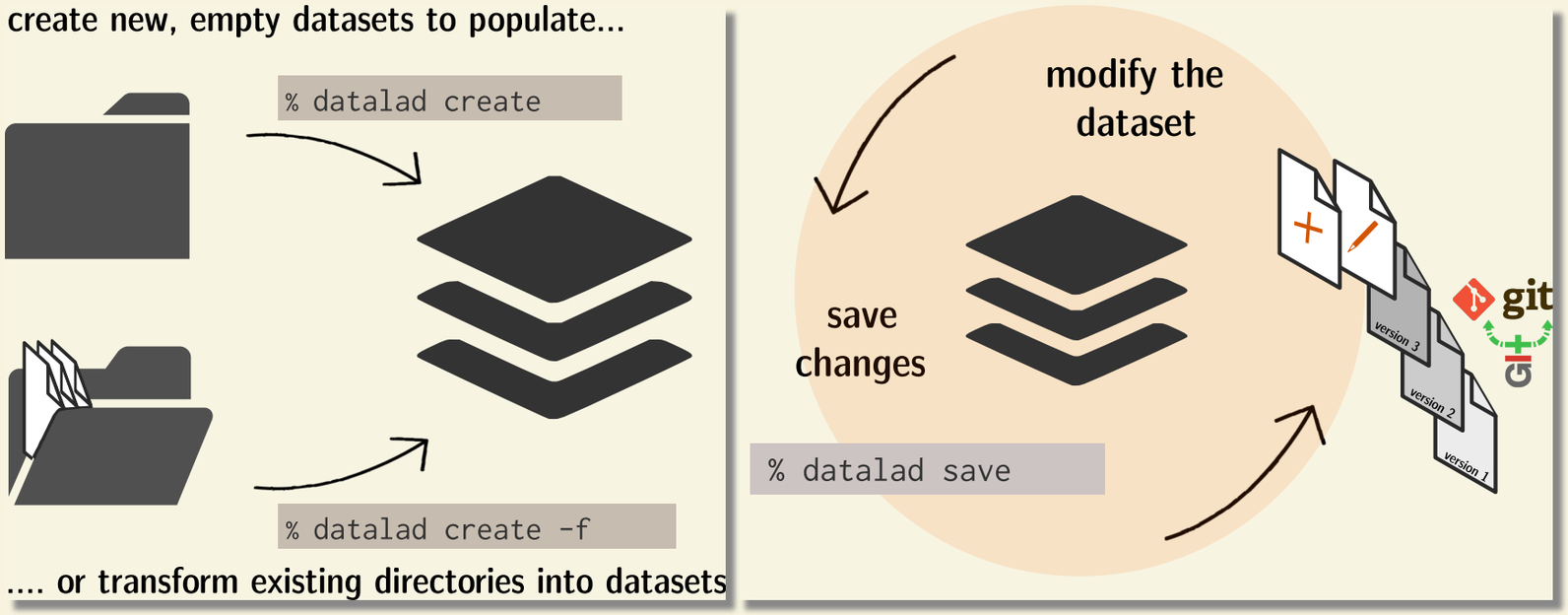
VERSION CONTROL

- DataLad knows two things: Datasets and files



VERSION CONTROL

- DataLad knows two things: Datasets and files



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

VERSION CONTROL

- Example: Add a new file into a dataset

```
# create a data analysis script
$ datalad status
untracked: code/script.py (file)
$ git status
On branch master
Untracked files:
  (use "git add file..." to include in what will be committed)
        code/script.py

nothing added to commit but untracked files present (use "git add" to track)
```

VERSION CONTROL

- Example: Add a new file into a dataset

```
# create a data analysis script
$ datalad status
untracked: code/script.py (file)
$ git status
On branch master
Untracked files:
  (use "git add file..." to include in what will be committed)
        code/script.py

nothing added to commit but untracked files present (use "git add" to track)
```

VERSION CONTROL

- Example: Add a new file into a dataset

```
# create a data analysis script
$ datalad status
untracked: code/script.py (file)
$ git status
On branch master
Untracked files:
  (use "git add file..." to include in what will be committed)
        code/script.py

nothing added to commit but untracked files present (use "git add" to track)
```

- Save the dataset modification

```
$ datalad save -m "Add a k-nearest-neighbour clustering analysis" code/script.py
```

THIS MEANS: YOU CAN ALSO VERSION CONTROL DATA!

THIS MEANS: YOU CAN ALSO VERSION CONTROL DATA!

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

THIS MEANS: YOU CAN ALSO VERSION CONTROL DATA!

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

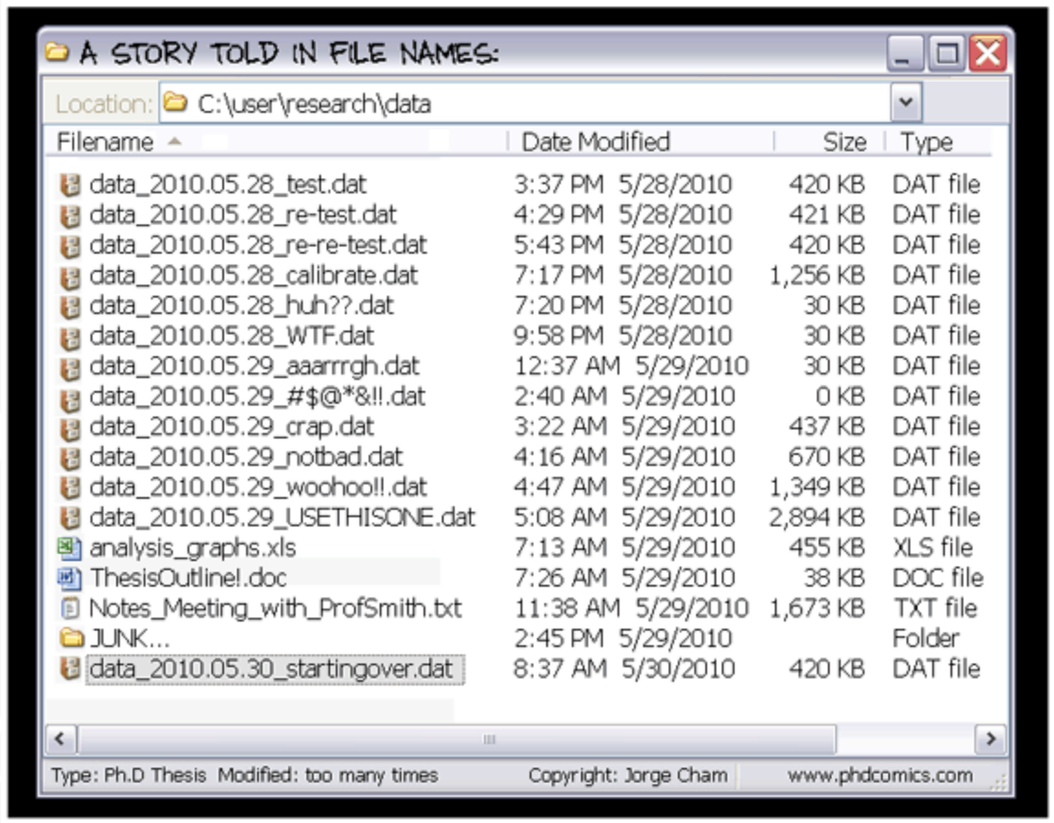
Why should you version control more than just your code?

THIS MEANS: YOU CAN ALSO VERSION CONTROL DATA!

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

Why should you version control more than just your code?

Because all building blocks of your analysis evolve



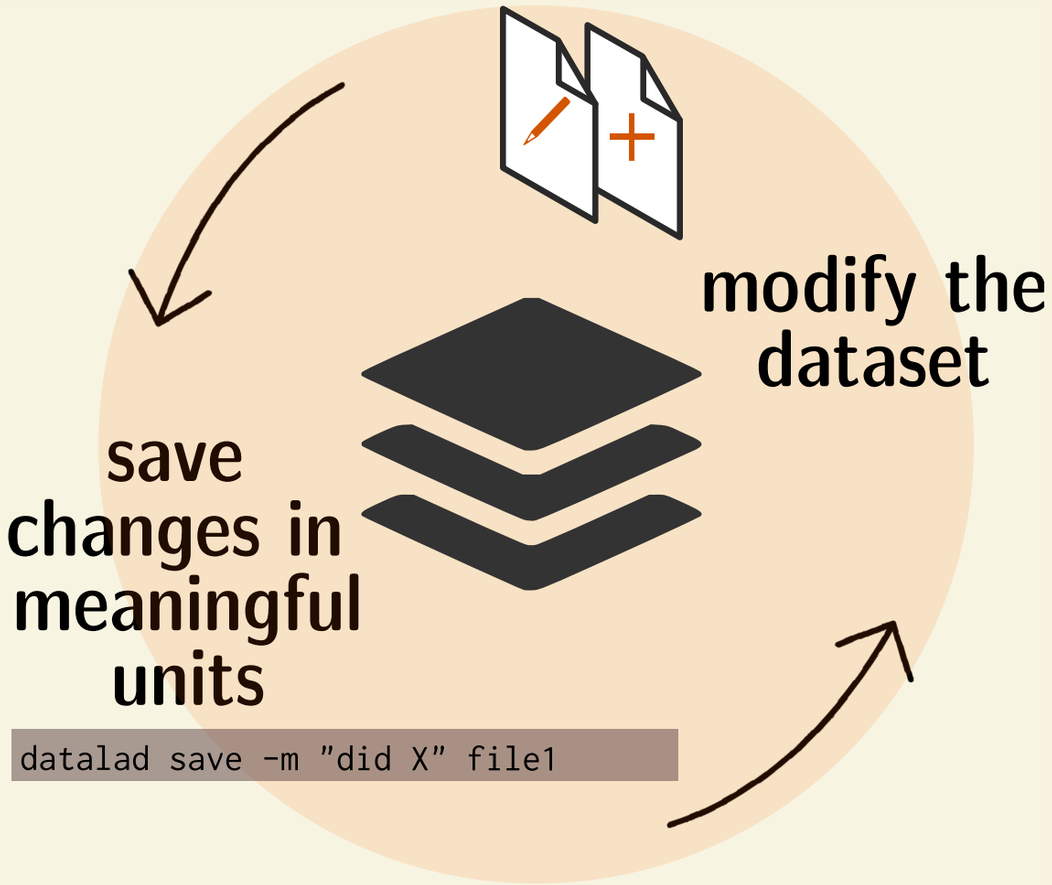
Version controlling data allows to track data changes and uniquely identify precise versions that were used in your analysis

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!

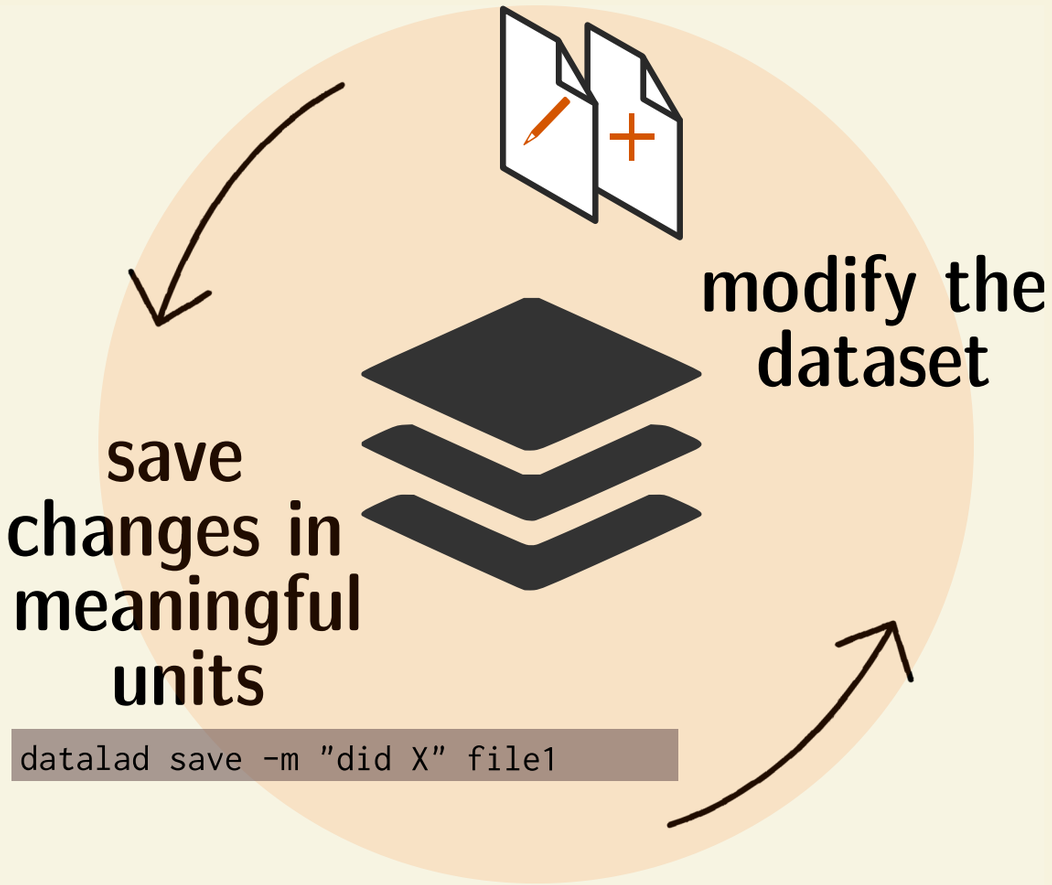
LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!



LOCAL VERSION CONTROL

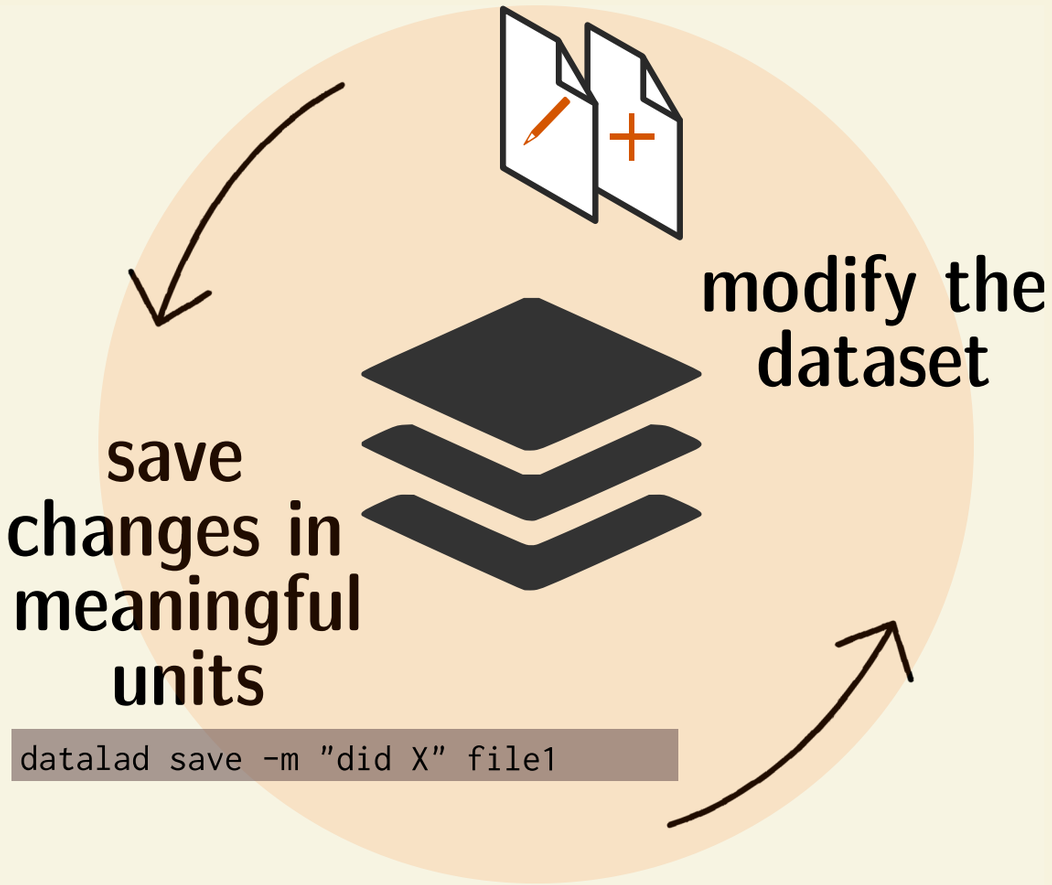
Procedurally, version control is easy with DataLad!



Stay flexible:

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!

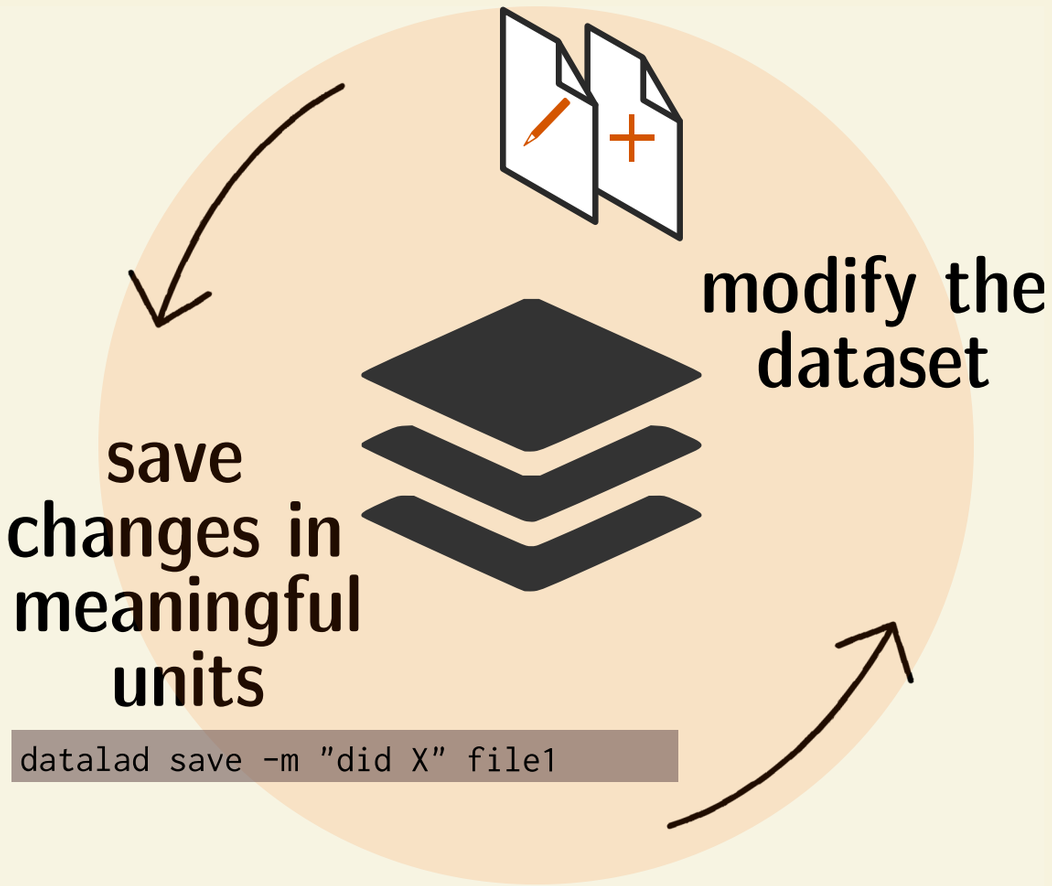


Stay flexible:

- Non-complex DataLad core API (easier than Git)

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!

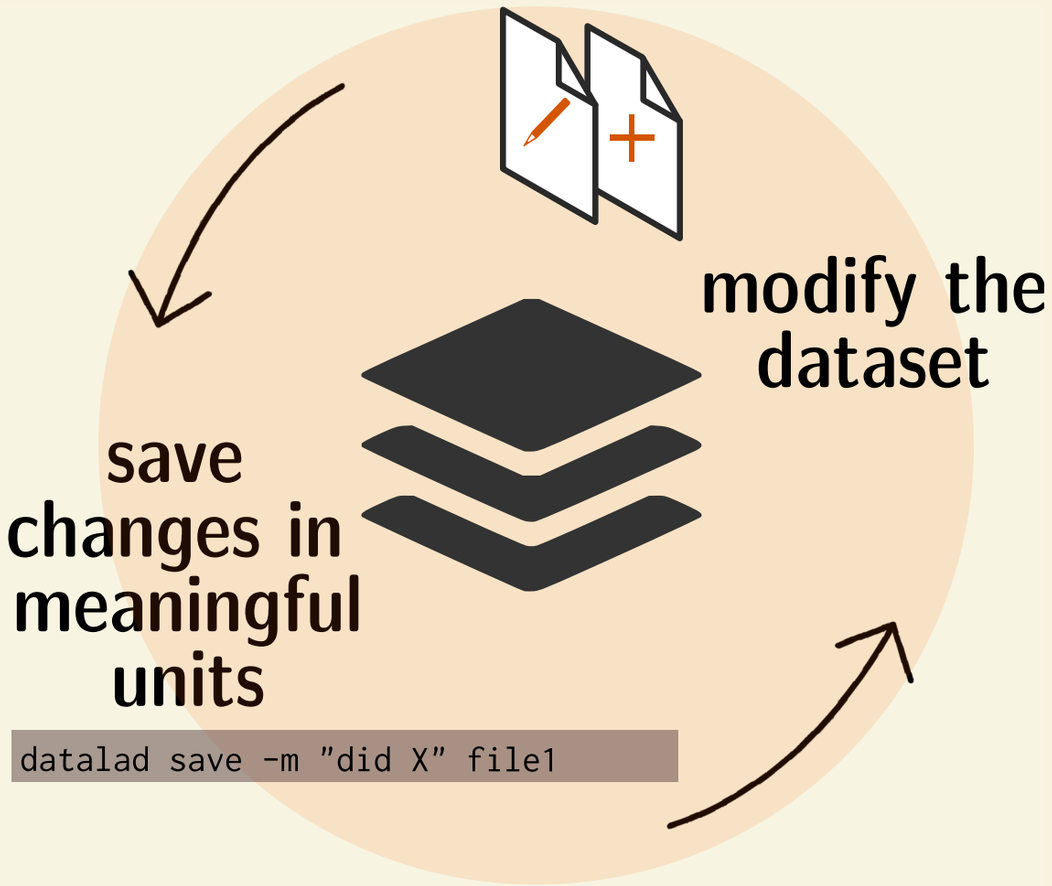


Stay flexible:

- Non-complex DataLad core API (easier than Git)
- Pure Git or git-annex commands (for regular Git or git-annex users, or to use specific functionality)

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!



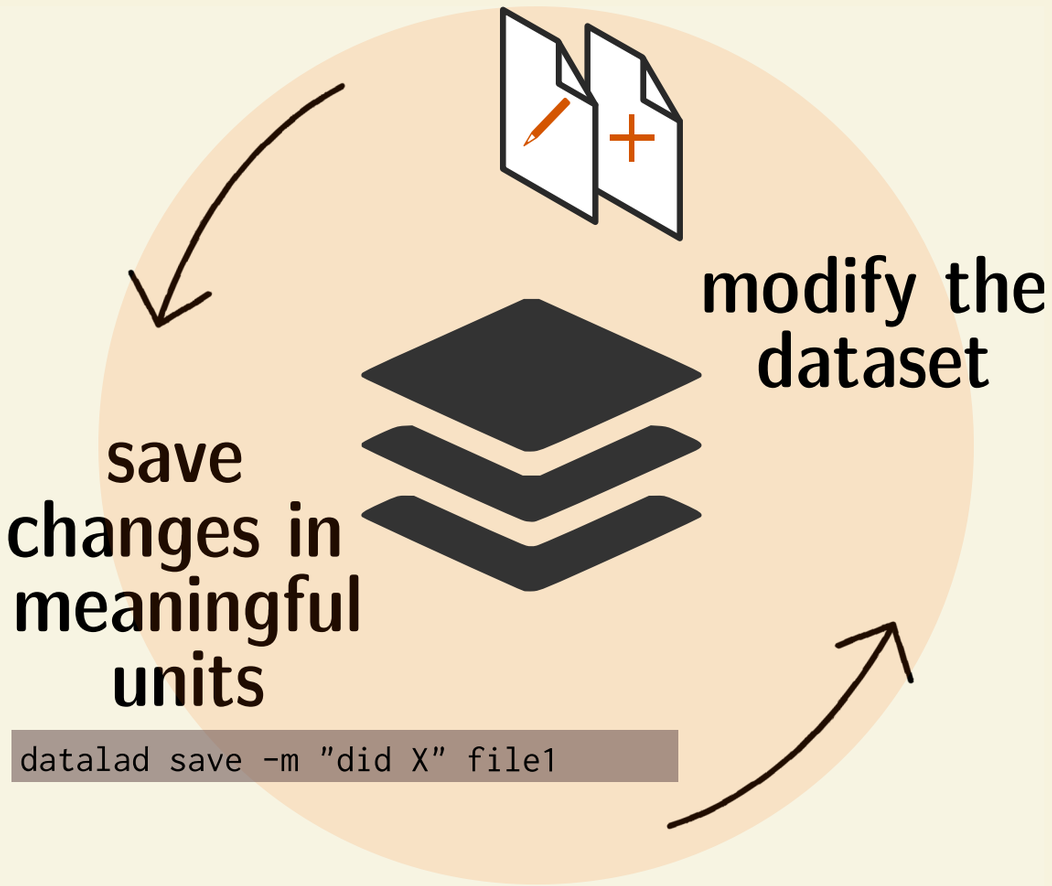
Stay flexible:

- Non-complex DataLad core API (easier than Git)
- Pure Git or git-annex commands (for regular Git or git-annex users, or to use specific functionality)

Advice:

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!



Stay flexible:

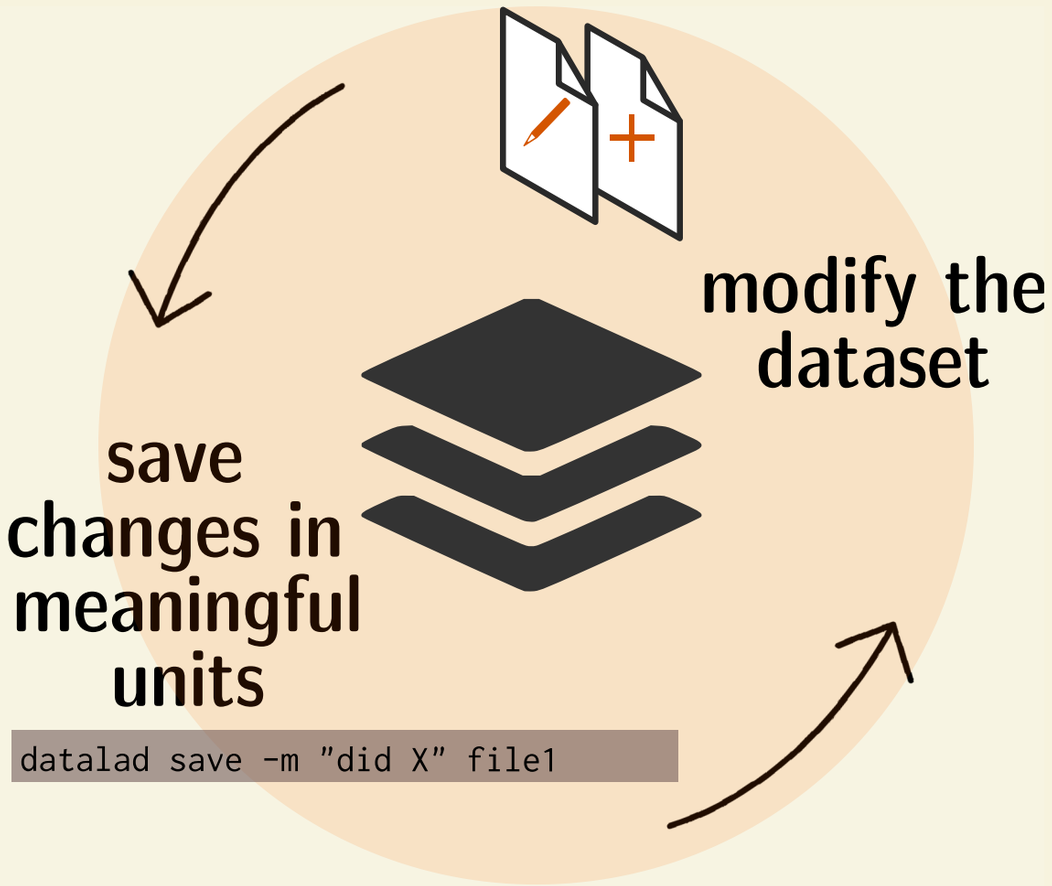
- Non-complex DataLad core API (easier than Git)
- Pure Git or git-annex commands (for regular Git or git-annex users, or to use specific functionality)

- Save *meaningful* units of change

Advice:

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!



Stay flexible:

- Non-complex DataLad core API (easier than Git)
- Pure Git or git-annex commands (for regular Git or git-annex users, or to use specific functionality)

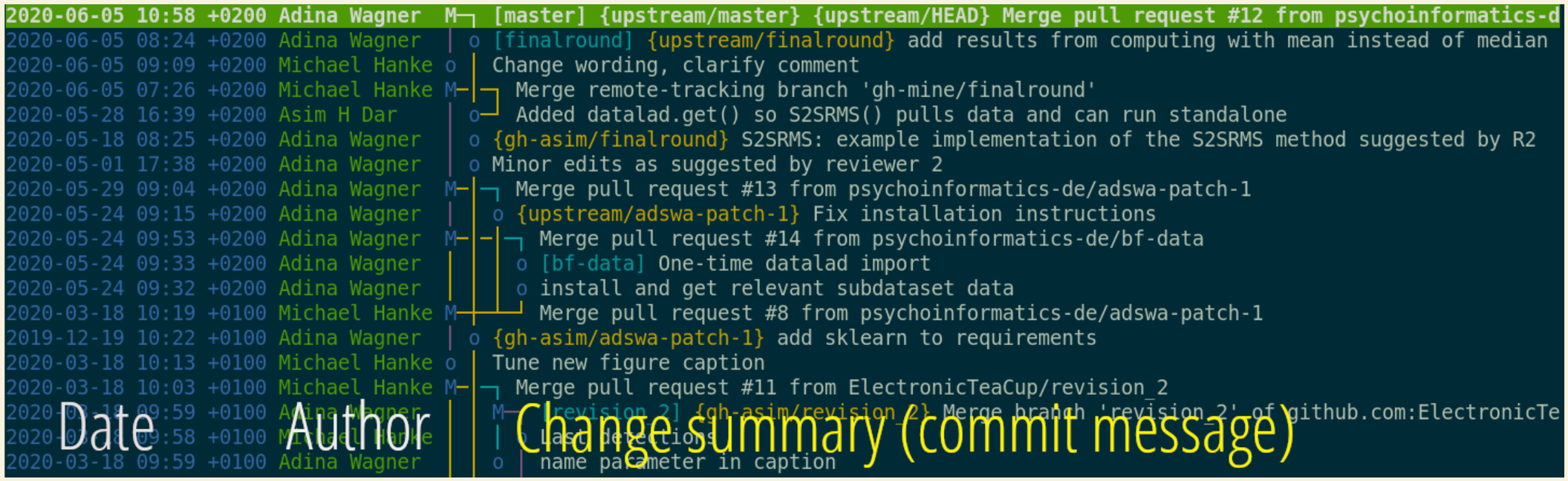
- Save *meaningful* units of change

Advice:

- Attach helpful commit messages

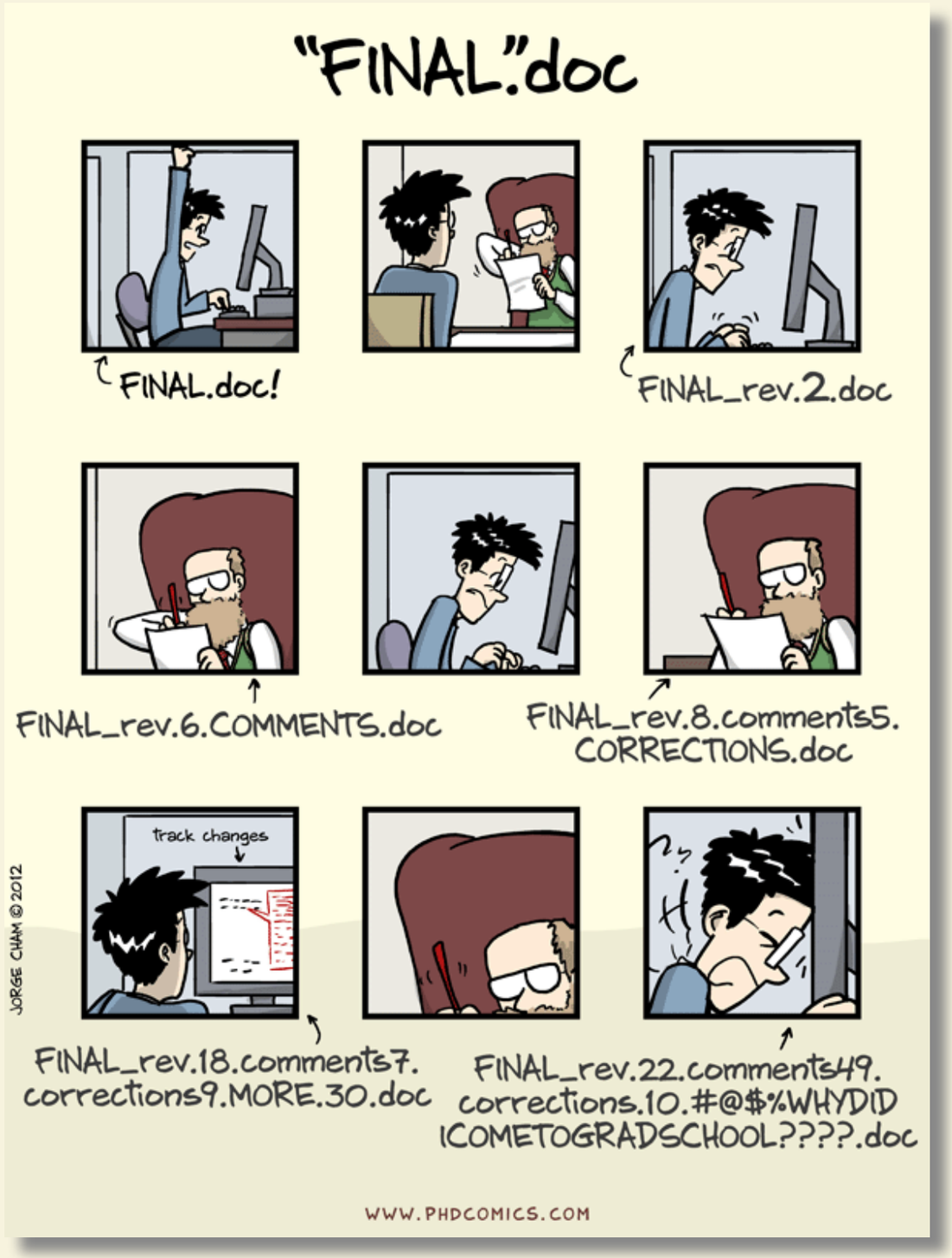
VERSION CONTROL

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



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

FROM HERE



FROM HERE

TO THIS:

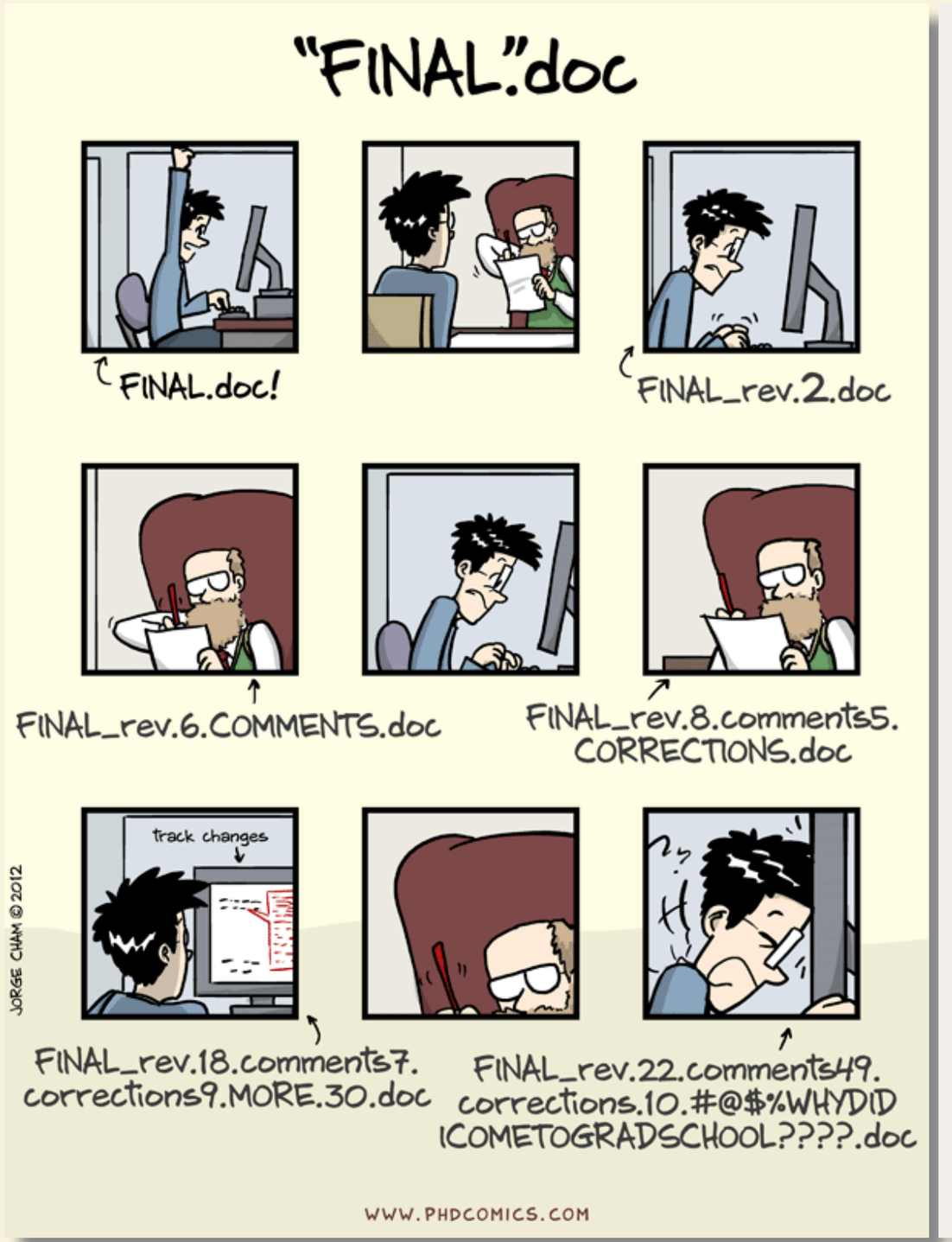


Image credit: www.phdcomics.com www.linode.com

FROM HERE

TO THIS:

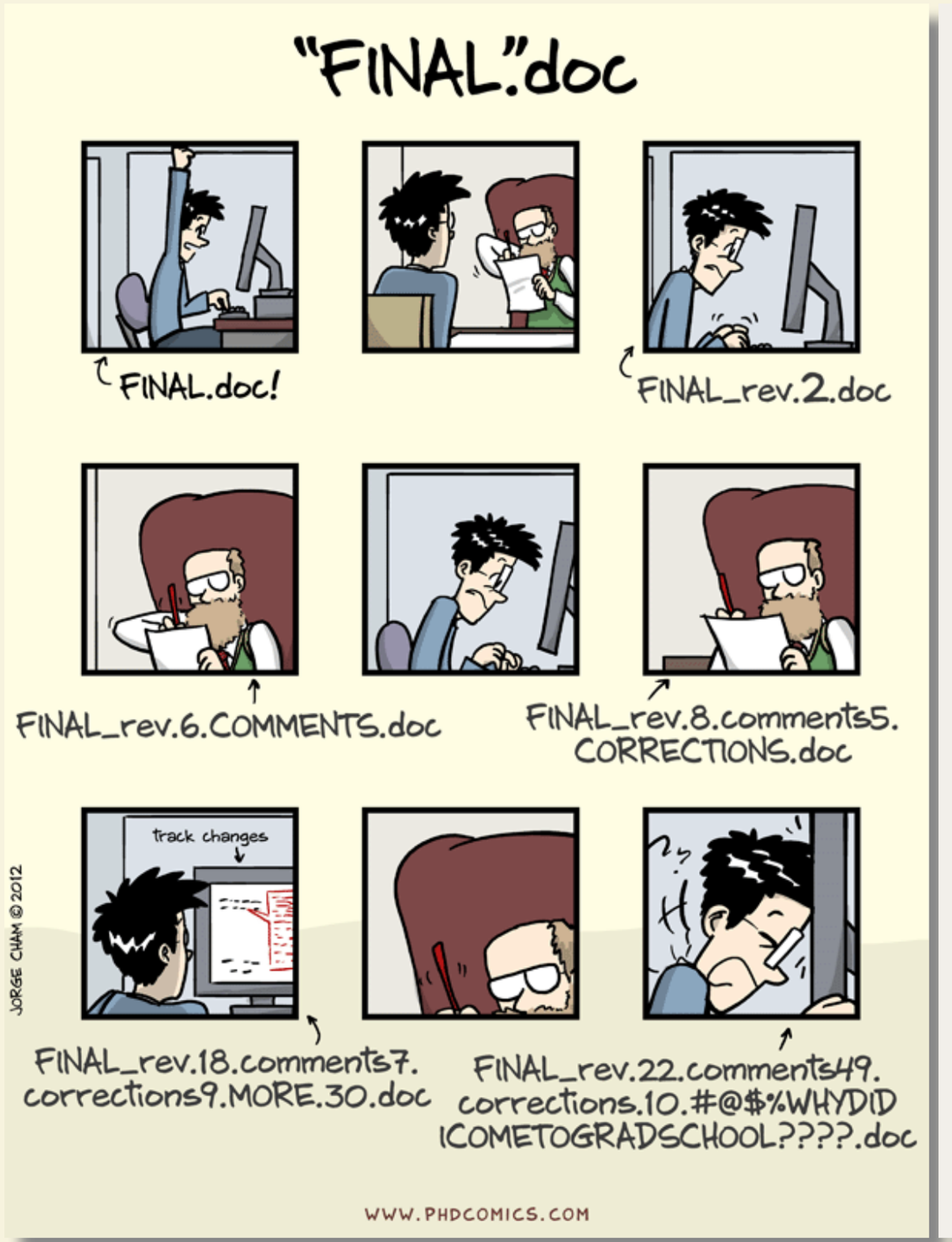
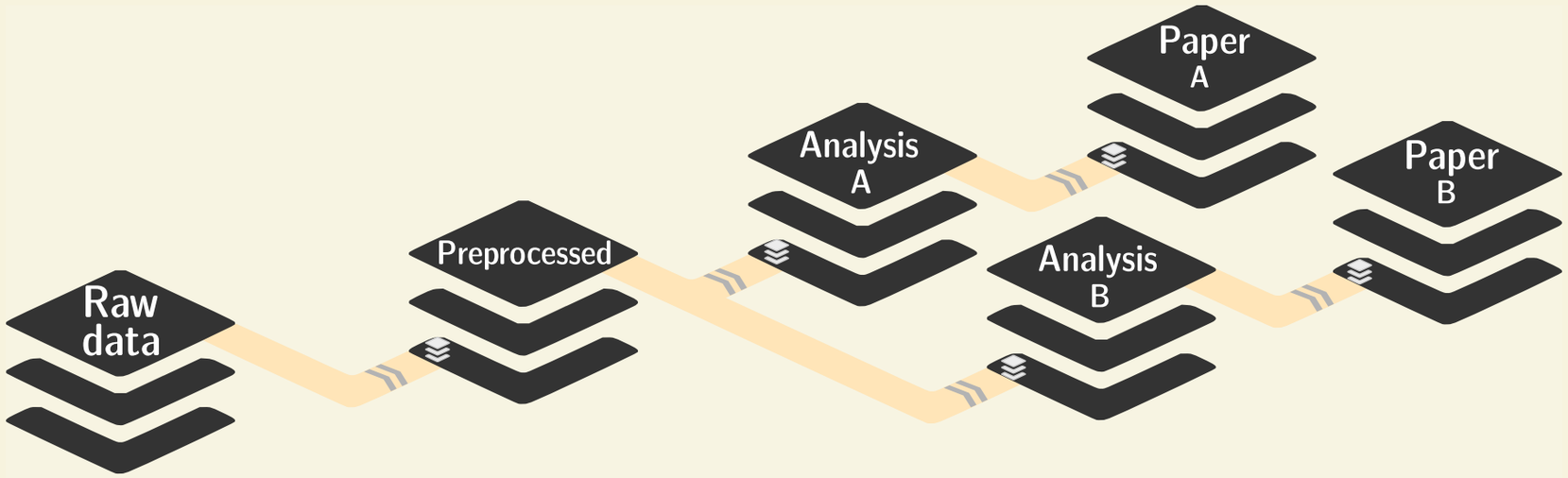


Image credit: www.phdcomics.com www.linode.com

BUT: Version control is only one aspect of data management

INTUITIVE DATA ANALYSIS STRUCTURE

- You can link datasets together in superdataset-subdataset hierarchies:

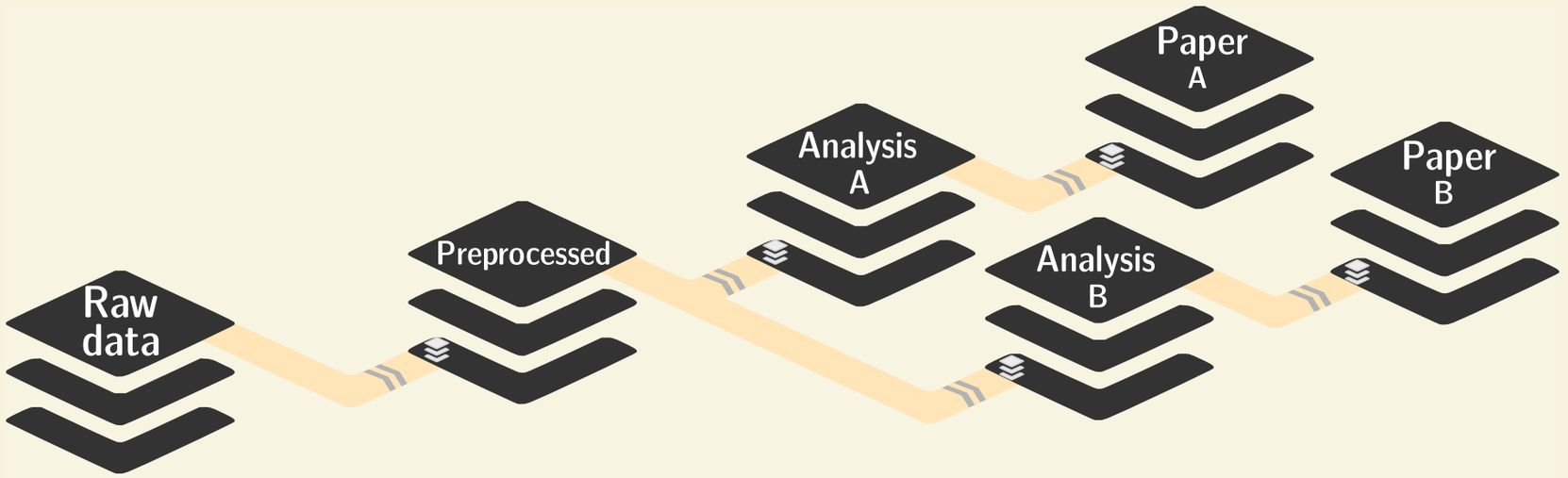


Nest modular datasets to create a linked hierarchy of datasets, and enable recursive operations throughout the hierarchy

```
$ cd myanalysis
# we can install analysis input data as a subdataset to the dataset
$ datalad clone -d . https://github.com/datalad-handbook/iris_data.git input/
[INFO   ] Scanning for unlocked files (this may take some time)
[INFO   ] Remote origin not usable by git-annex; setting annex-ignore
install(ok): input (dataset)
add(ok): input (file)
add(ok): .gitmodules (file)
save(ok): . (dataset)
action summary:
  add (ok: 2)
  install (ok: 1)
  save (ok: 1)
```

INTUITIVE DATA ANALYSIS STRUCTURE

- You can link datasets together in superdataset-subdataset hierarchies:



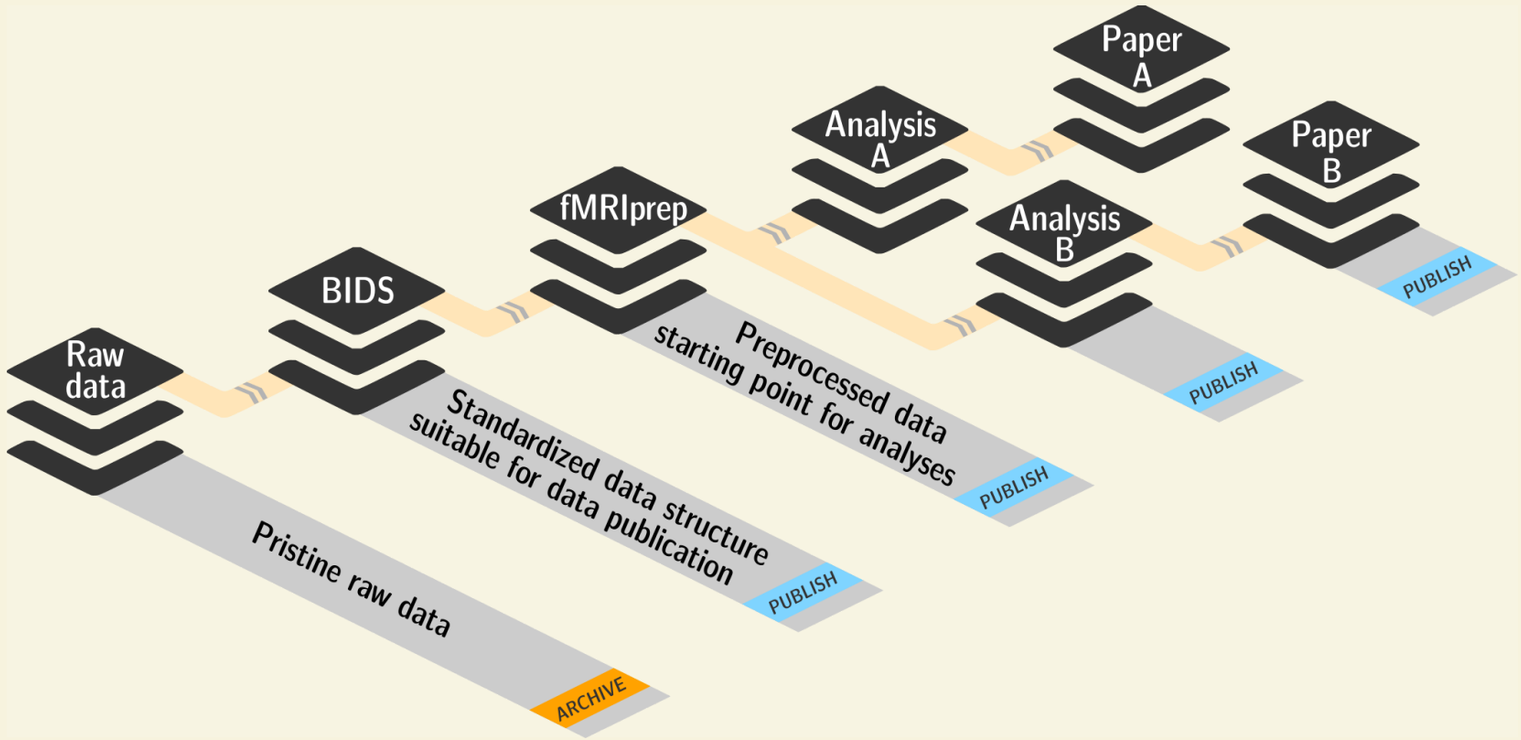
Nest modular datasets to create a linked hierarchy of datasets, and enable recursive operations throughout the hierarchy

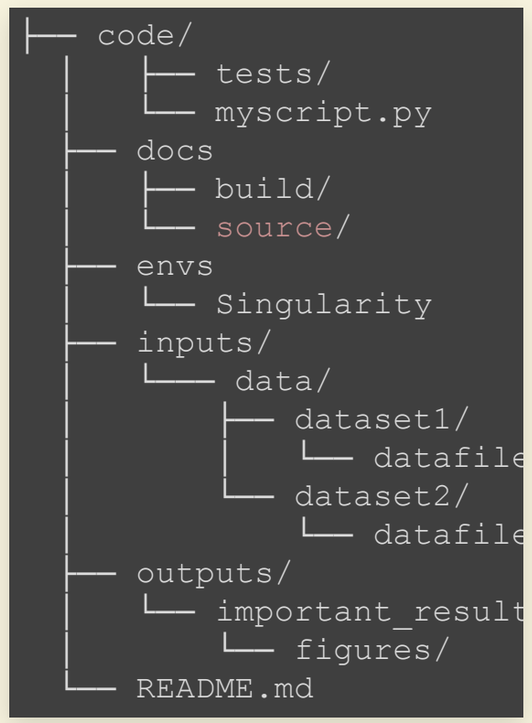
```
$ tree
.
├── CHANGELOG.md
├── code
│   ├── README.md
│   └── script.py
└── input
    └── iris.csv
```

BASIC ORGANIZATIONAL PRINCIPLES FOR DATASETS

Keep everything clean and modular

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





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

Find out more about organizational principles in the YODA principles!

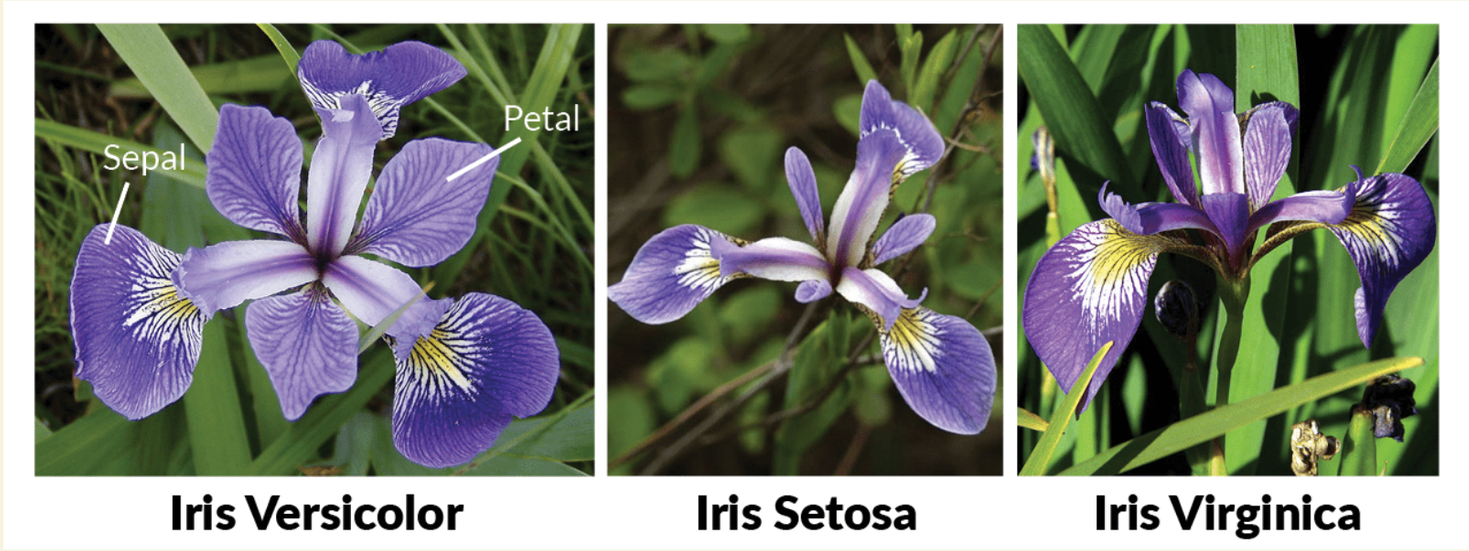
COMPUTATIONALLY REPRODUCIBLE DATA ANALYSIS

This a metaphor for reproducing (your own) research
a few months after publication

⬇

Write-up: handbook.datalad.org/en/latest/basics/101-130-yodaproject.html

A CLASSIFICATION ANALYSIS ON THE IRIS FLOWER DATASET



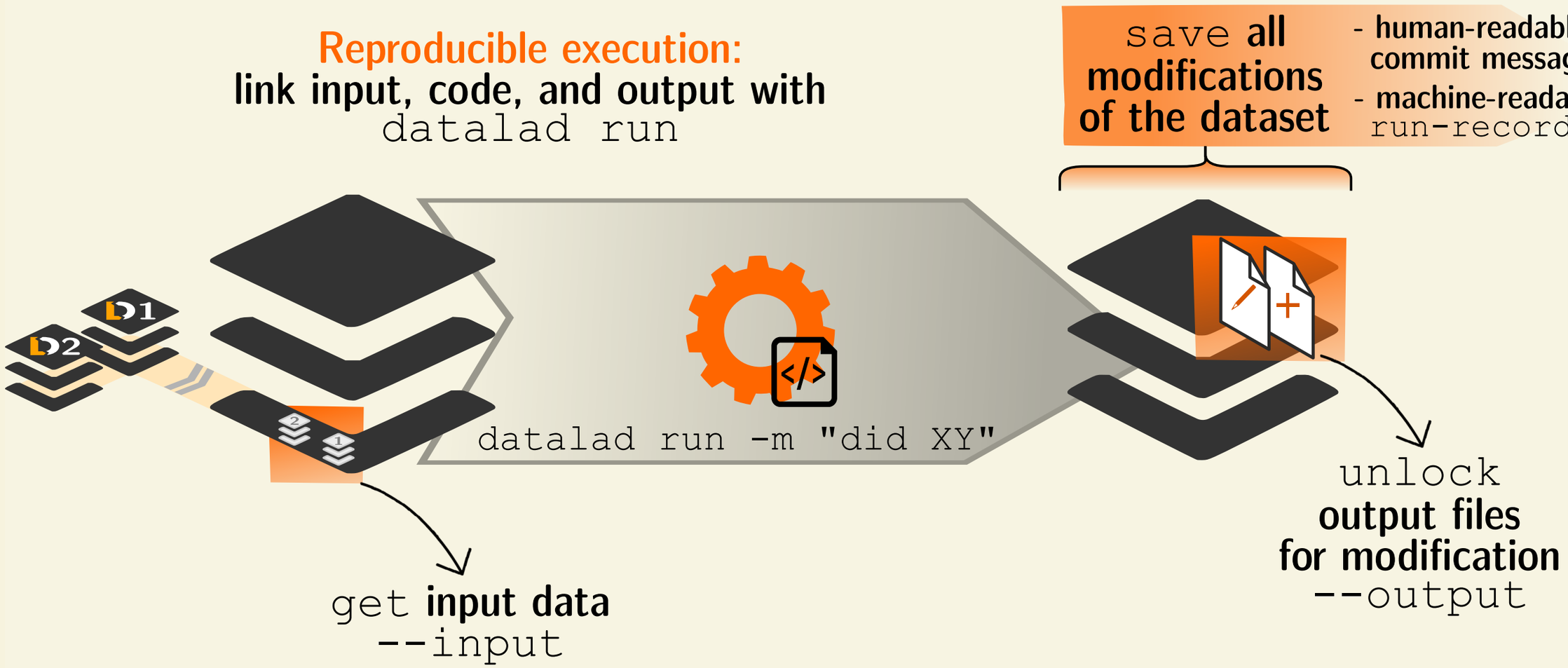
Write-up: handbook.datalad.org/en/latest/basics/101-130-yodaproject.html

REPRODUCIBLE EXECUTION & PROVENANCE CAPTURE

datalad run

REPRODUCIBLE EXECUTION & PROVENANCE CAPTURE

datalad run



COMPUTATIONAL REPRODUCIBILITY

How can I execute the analysis script on my input data in a computationally reproducible manner?

```
$ datalad run -m "analyze iris data with classification analysis" \
  --input "input/iris.csv" \
  --output "prediction_report.csv" \
  --output "pairwise_relationships.png" \
  "python3 code/script.py"
[INFO   ] Making sure inputs are available (this may take some time)
get(ok): input/iris.csv (file) [from web...]
[INFO   ] == Command start (output follows) =====
[INFO   ] == Command exit (modification check follows) =====
add(ok): pairwise_relationships.png (file)
add(ok): prediction_report.csv (file)
save(ok): . (dataset)
action summary:
  add (ok: 2)
  get (notneeded: 2, ok: 1)
  save (notneeded: 1, ok: 1)
```

COMPUTATIONAL REPRODUCIBILITY

How can I execute the analysis script on my input data in a computationally reproducible manner?

```
$ datalad run -m "analyze iris data with classification analysis" \
  --input "input/iris.csv" \
  --output "prediction_report.csv" \
  --output "pairwise_relationships.png" \
  "python3 code/script.py"
[INFO   ] Making sure inputs are available (this may take some time)
get(ok): input/iris.csv (file) [from web...]
[INFO   ] == Command start (output follows) =====
[INFO   ] == Command exit (modification check follows) =====
add(ok): pairwise_relationships.png (file)
add(ok): prediction_report.csv (file)
save(ok): . (dataset)
action summary:
  add (ok: 2)
  get (notneeded: 2, ok: 1)
  save (notneeded: 1, ok: 1)
```

COMPUTATIONAL REPRODUCIBILITY

- A datalad run command produces a machine-readable record, identifiable via commit hash

```
$ git log
commit df2dae9b5af184a0c463708acf8356b877c511a8 (HEAD -> master)
Author: Adina Wagner adina.wagner@t-online.de
Date:   Tue Dec 1 11:58:18 2020 +0100

    [DATALAD RUNCMD] analyze iris data with classification analysis

    === Do not change lines below ===
    {
     "chain": [],
     "cmd": "python3 code/script.py",
     "dsid": "9ffdbfcd-f4af-429a-b64a-0c81b48b7f62",
     "exit": 0,
     "extra_inputs": [],
     "inputs": [
      "input/iris.csv"
     ],
     "outputs": [
      "prediction_report.csv",
      "pairwise_relationships.png"
     ],
     "pwd": "."
    }
    ^^^ Do not change lines above ^^^
```

COMPUTATIONAL REPRODUCIBILITY

- A datalad run command produces a machine-readable record, identifiable via commit hash

```
$ git log
commit df2dae9b5af184a0c463708acf8356b877c511a8 (HEAD -> master)
Author: Adina Wagner adina.wagner@t-online.de
Date:   Tue Dec 1 11:58:18 2020 +0100

    [DATALAD RUNCMD] analyze iris data with classification analysis

    [...]
```

- You can `rerun` this hash to repeat the analysis:

```
 $ datalad rerun df2dae9b5af1
datalad rerun df2dae9b5af18
[INFO   ] run commit df2dae9; (analyze iris data...)
[INFO   ] Making sure inputs are available (this may take some time)
unlock(ok): pairwise_relationships.png (file)
unlock(ok): prediction_report.csv (file)
[INFO   ] == Command start (output follows) =====
[INFO   ] == Command exit (modification check follows) =====
add(ok): pairwise_relationships.png (file)
add(ok): prediction_report.csv (file)
action summary:
  add (ok: 2)
  get (notneeded: 3)
  save (notneeded: 2)
  unlock (ok: 2)
```

COMPUTATIONAL REPRODUCIBILITY

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

datalad-containers run

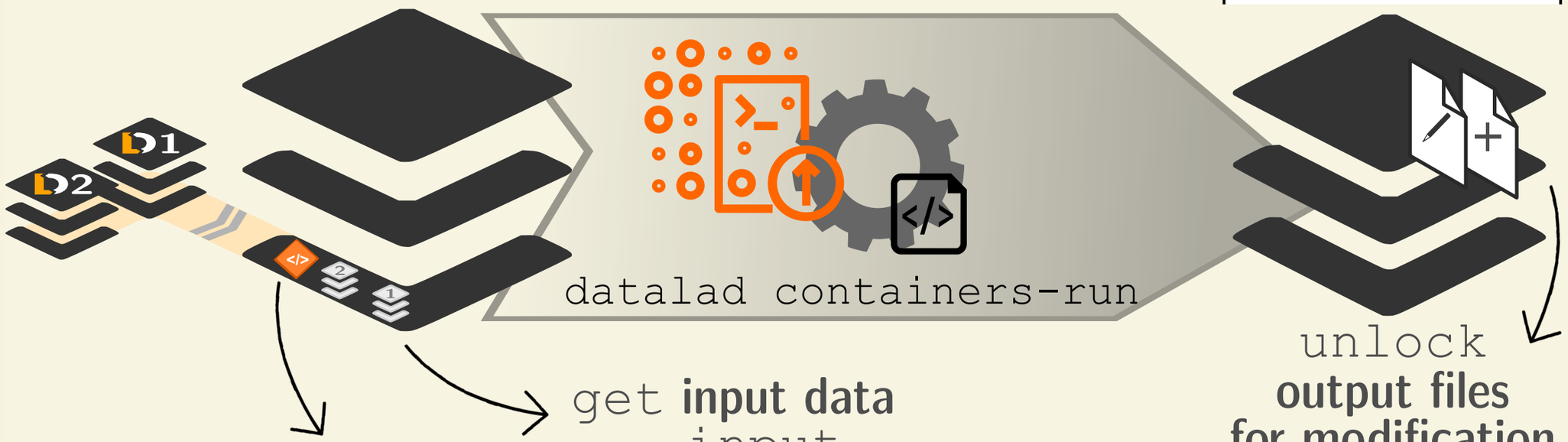
COMPUTATIONAL REPRODUCIBILITY

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

datalad-containers run

link input, code, output, and software with
datalad containers-run

save all
modifications
of the dataset



get input data
--input

unlock
output files
for modification
--output

get and use registered software
container for computation
--container-name

COMPUTATIONAL REPRODUCIBILITY

- You can add (any amount of) software containers to your dataset to link a software environment to your analysis

```
$ datalad containers-add software --url shub://adswa/resources:2
[INFO   ] Initiating special remote datalad
add(ok): .datalad/config (file)
save(ok): . (dataset)
containers_add(ok): /tmp/myanalysis/.datalad/environments/software/image (file)
action summary:
  add (ok: 1)
  containers_add (ok: 1)
  save (ok: 1)
```

Write-up: http://handbook.datalad.org/en/latest/basics/101-133-containersrun.html

COMPUTATIONAL REPRODUCIBILITY

- `datalad containers-run` will execute the command in the specified software environment

```
$ datalad containers-run -m "rerun analysis in container" \
  --container-name midterm-software \
  --input "input/iris.csv" \
  --output "prediction_report.csv" \
  --output "pairwise_relationships.png" \
  "python3 code/script.py"
[INFO] Making sure inputs are available (this may take some time)
[INFO] == Command start (output follows) =====
[INFO] == Command exit (modification check follows) =====
unlock(ok): pairwise_relationships.png (file)
unlock(ok): prediction_report.csv (file)
add(ok): pairwise_relationships.png (file)
add(ok): prediction_report.csv (file)
save(ok): . (dataset)
action summary:
  add (ok: 2)
  get (notneeded: 4)
  save (notneeded: 1, ok: 1)
  unlock (ok: 2)
```

- ... And a `datalad rerun` will repeat the analysis in the specified software environment

A QUICK SUMMARY OF THIS SNEAK PEEK

- Getting data

A QUICK SUMMARY OF THIS SNEAK PEEK

- Getting data
  - You can retrieve DataLad datasets with "datalad clone url/path"
  - A dataset allows you to retrieve data on demand via "datalad get"
  - You can drop unused data to free up disk space with "datalad drop"

A QUICK SUMMARY OF THIS SNEAK PEEK

- Getting data
    - You can retrieve DataLad datasets with "datalad clone url/path"
    - A dataset allows you to retrieve data on demand via "datalad get"
    - You can drop unused data to free up disk space with "datalad drop"
- Keeping projects clean

A QUICK SUMMARY OF THIS SNEAK PEEK

- Getting data
  - You can retrieve DataLad datasets with "datalad clone url/path"
  - A dataset allows you to retrieve data on demand via "datalad get"
  - You can drop unused data to free up disk space with "datalad drop"
- Keeping projects clean
  - Create a dataset for data analysis using "datalad create -c yoda mydatasetname"
  - In this dataset, DataLad can version control data of any size with "datalad save"
  - You can link individual datasets as reusable and intuitive modular components, for example your input data to your analysis, with "datalad clone -d . url"

A QUICK SUMMARY OF THIS SNEAK PEEK

- Getting data
    - You can retrieve DataLad datasets with "datalad clone url/path"
    - A dataset allows you to retrieve data on demand via "datalad get"
    - You can drop unused data to free up disk space with "datalad drop"
- Keeping projects clean
    - Create a dataset for data analysis using "datalad create -c yoda mydatasetname"
    - In this dataset, DataLad can version control data of any size with "datalad save"
    - You can link individual datasets as reusable and intuitive modular components, for example your input data to your analysis, with "datalad clone -d . url"
- Computational reproducibility

A QUICK SUMMARY OF THIS SNEAK PEEK

- Getting data
  - You can retrieve DataLad datasets with "datalad clone url/path"
  - A dataset allows you to retrieve data on demand via "datalad get"
  - You can drop unused data to free up disk space with "datalad drop"
- Keeping projects clean
  - Create a dataset for data analysis using "datalad create -c yoda mydatasetname"
  - In this dataset, DataLad can version control data of any size with "datalad save"
  - You can link individual datasets as reusable and intuitive modular components, for example your input data to your analysis, with "datalad clone -d . url"
- Computational reproducibility
  - "datalad run" can create a digital, machine-readable, and re-executable record of how you did your data analysis
  - You or others can redo the analysis automatically with "datalad rerun"
  - You can even link software environments to your analysis with the "datalad-container" extension, and run analysis with "datalad containers-run"

IS THERE MORE?

Yes, a lot!

IS THERE MORE?

Yes, a lot!

- For example: Collaborative data analysis workflows

IS THERE MORE?

Yes, a lot!

- For example: Collaborative data analysis workflows
- Publishing data

IS THERE MORE?

Yes, a lot!

- For example: Collaborative data analysis workflows
- Publishing data
- Writing reproducible papers

IS THERE MORE?

Yes, a lot!

- For example: Collaborative data analysis workflows
- Publishing data
- Writing reproducible papers
- computationally reproducible machine learning pipelines

IS THERE MORE?

Yes, a lot!

- For example: Collaborative data analysis workflows
- Publishing data
- Writing reproducible papers
- computationally reproducible machine learning pipelines
- ...

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