

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

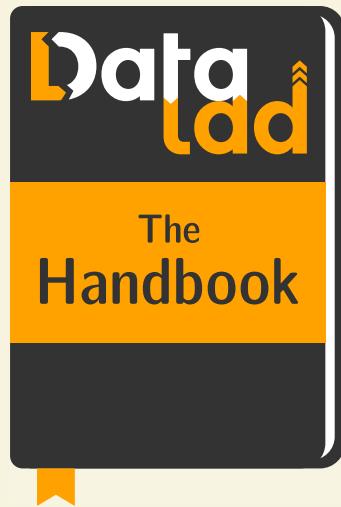
A FEW THINGS THAT DATALAD CAN HELP WITH

- 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*



NSF 1429999

Funders



germany-usa



EUROPEAN UNION
European Regional Development Fund



cbbs
center for behavioral
brain sciences



BMBF 01GQ1411



Collaborators



OpenNEURO



brainlife.io

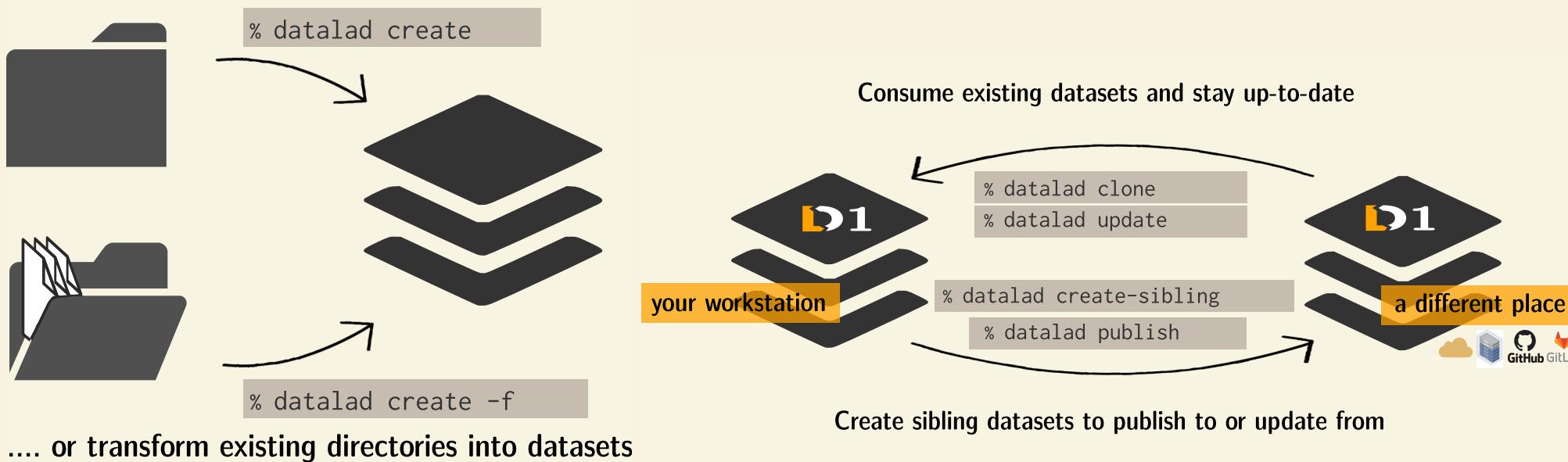
EVERYTHING HAPPENS IN DATALAD DATASETS

- 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*

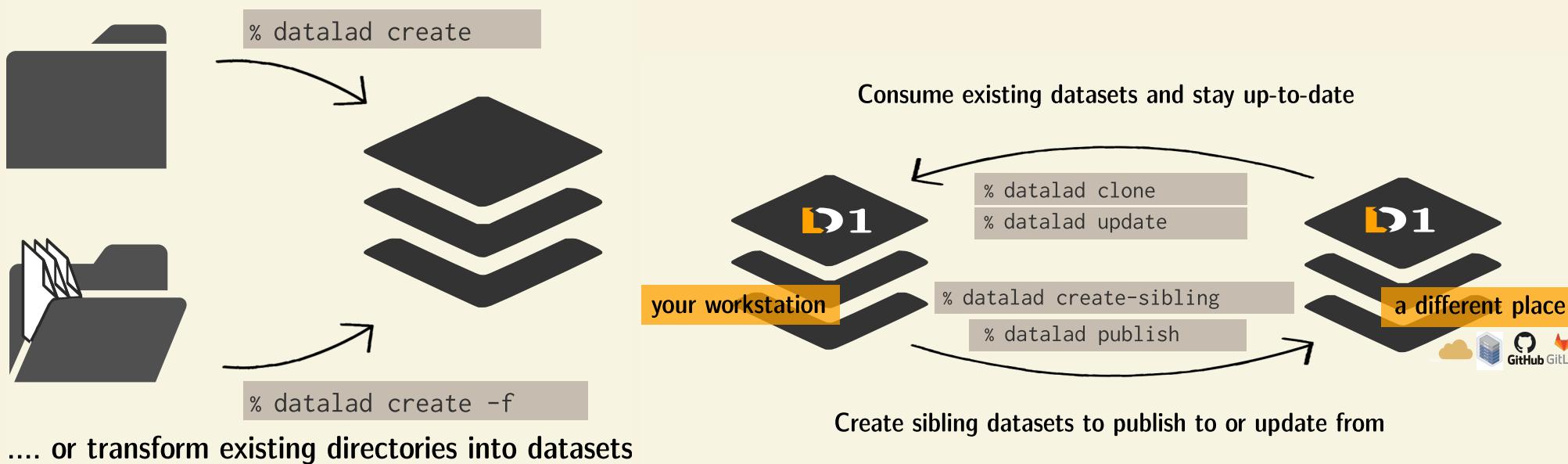
create new, empty datasets to populate...



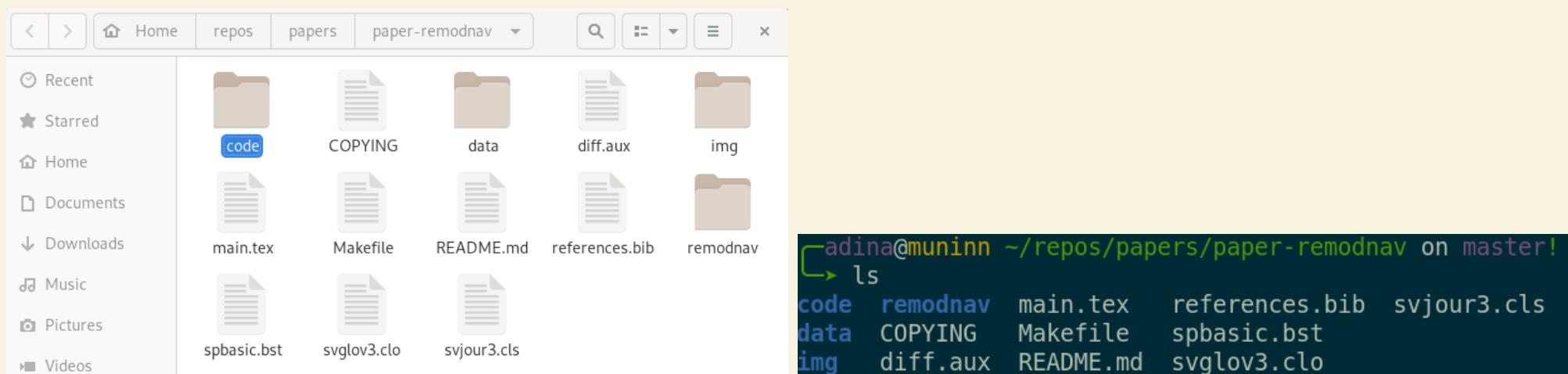
EVERYTHING HAPPENS IN DATALAD DATASETS

- 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...



File viewer and terminal view of a DataLad dataset



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

The screenshot shows a GitHub repository page for 'psychoinformatics-de / studyforrest-data-phase2'. The repository has 1 issue, 1 pull request, and 8 forks. The 'Code' tab is selected, showing the master branch with 2 branches and 1 tag. The commit history lists 77 commits by user 'mih' from May 7, 2016. The commits are mostly related to respiratory trace imports and BIDS compatibility. The right sidebar includes sections for About, Releases (1), Packages, Contributors (3), and Languages.

About

studyforrest.org: Phase2 data (movie, eyetracking, remapping, visual localizers) [BIDS]

[studyforrest.org](#)

[Readme](#) [View license](#)

Releases

First public release Latest
on Mar 26, 2016

Packages

No packages published

Contributors

mih Michael Hanke
dakot Daniel Kottke
adswa Adina Wagner

Languages

A horizontal bar chart showing the languages used in the codebase, with the most prominent being Python (blue).

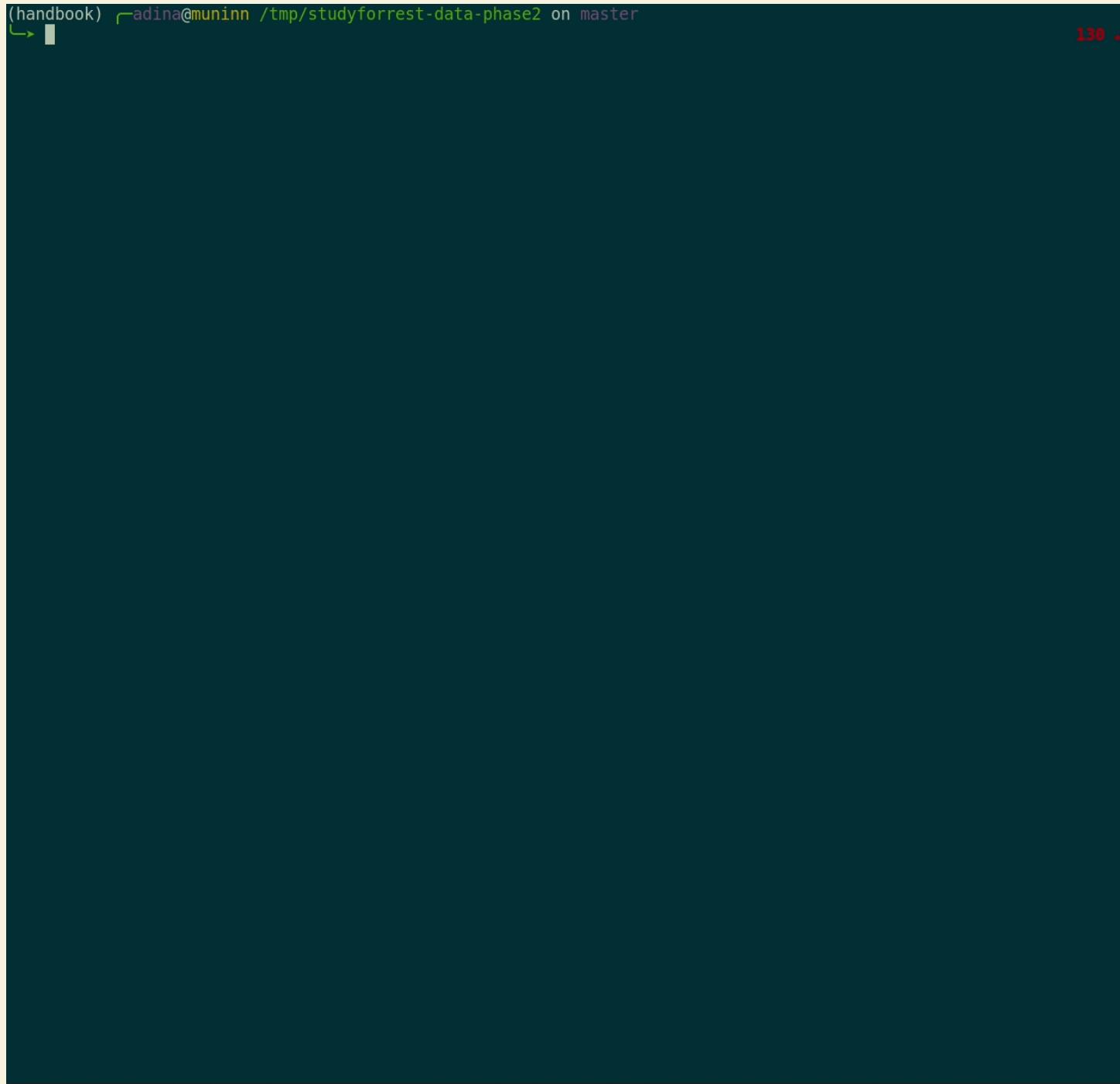
Commit	Message	Date
mih	Merge pull request #15 from adswa/ENH/README	b5306e2 on May 7
.datalad	[DATALAD] dataset aggregate metadata update	2 years ago
code	Fix type in physio log converter (fixes gh-11)	3 years ago
src	Recover lost segment from eyetracker (closes gh-3)	5 years ago
stimuli	Add BIDS-compatible stimuli/ directory (with symlinks)	4 years ago
sub-01	BF: Re-import respiratory trace after bug fix in converte...	3 years ago
sub-02	BF: Re-import respiratory trace after bug fix in converte...	3 years ago
sub-03	BF: Re-import respiratory trace after bug fix in converte...	3 years ago
sub-04	BF: Re-import respiratory trace after bug fix in converte...	3 years ago
sub-05	BF: Re-import respiratory trace after bug fix in converte...	3 years ago
sub-06	BF: Re-import respiratory trace after bug fix in converte...	3 years ago
sub-09	BF: Re-import respiratory trace after bug fix in converte...	3 years ago
sub-10	BF: Re-import respiratory trace after bug fix in converte...	3 years ago
sub-14	BF: Re-import respiratory trace after bug fix in converte...	3 years ago
sub-15	BF: Re-import respiratory trace after bug fix in converte...	3 years ago
sub-16	BF: Re-import respiratory trace after bug fix in converte...	3 years ago
sub-17	BF: Re-import respiratory trace after bug fix in converte...	3 years ago
sub-18	BF: Re-import respiratory trace after bug fix in converte...	3 years ago
sub-19	BF: Re-import respiratory trace after bug fix in converte...	3 years ago
sub-20	BF: Re-import respiratory trace after bua fix in converte...	3 years ago

- 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**.

```
(handbook) radina@muninn /tmp/studyforrest-data-phase2 on master
└─▶ 130 ..
```

- 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
```

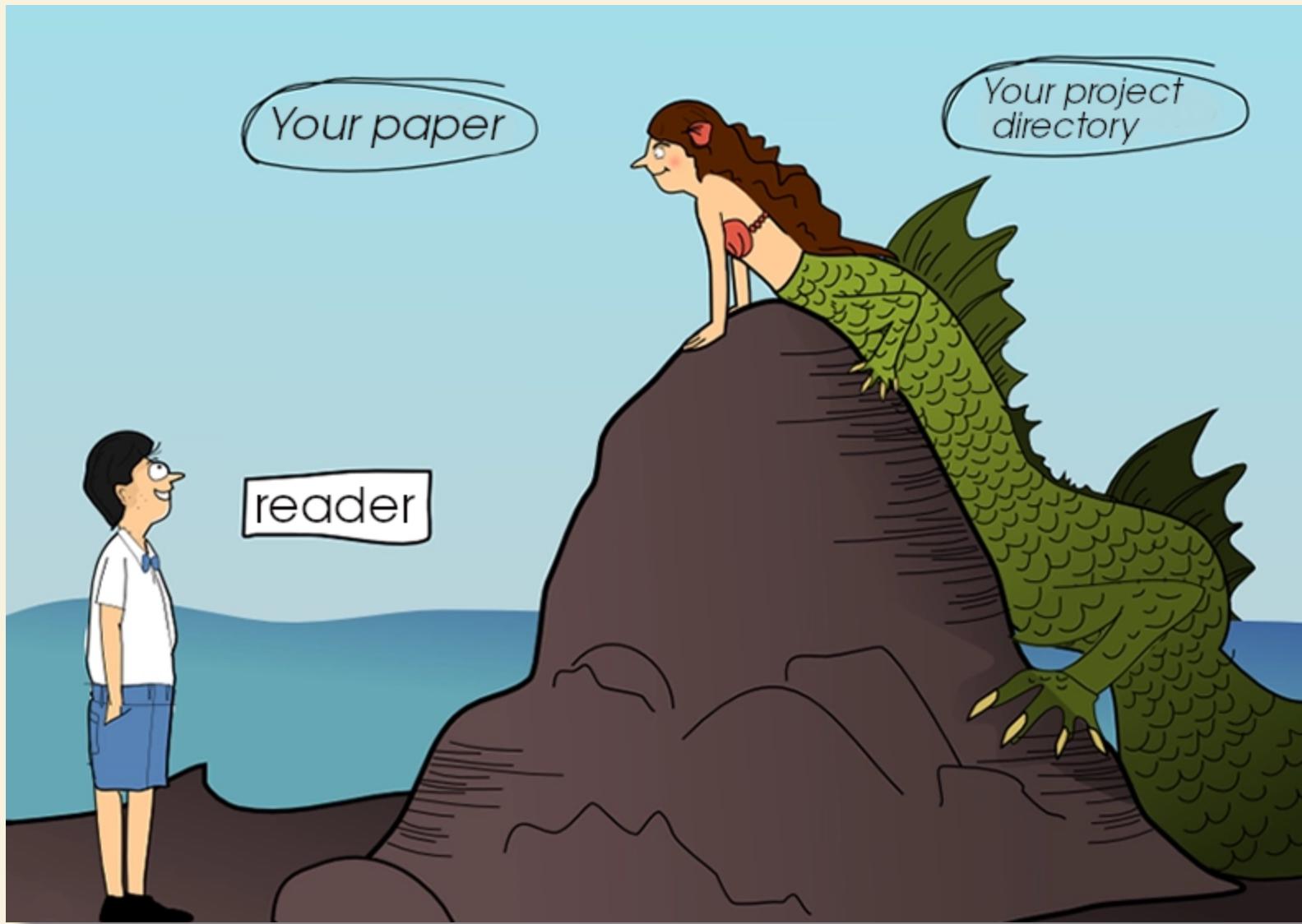
- The human connectome project data (full, and in subsets): github.com/datalad-datasets/human-connectome-project-openaccess

```
$ 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 ///
```

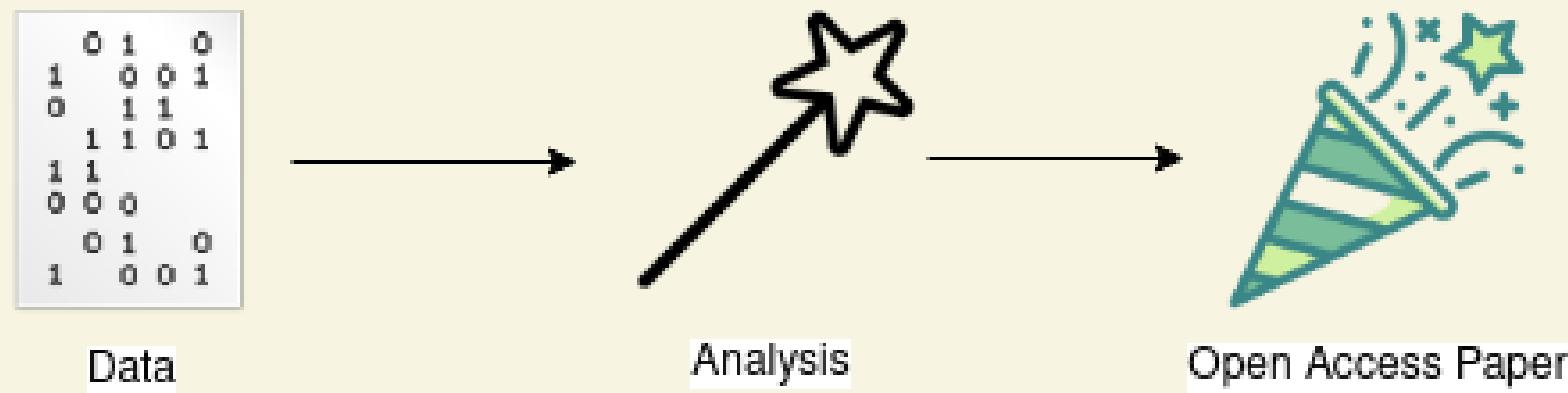
KEEPING A PROJECT CLEAN AND ORDERLY



This a metaphor for most projects after publication

KEEPING A PROJECT CLEAN AND ORDERLY

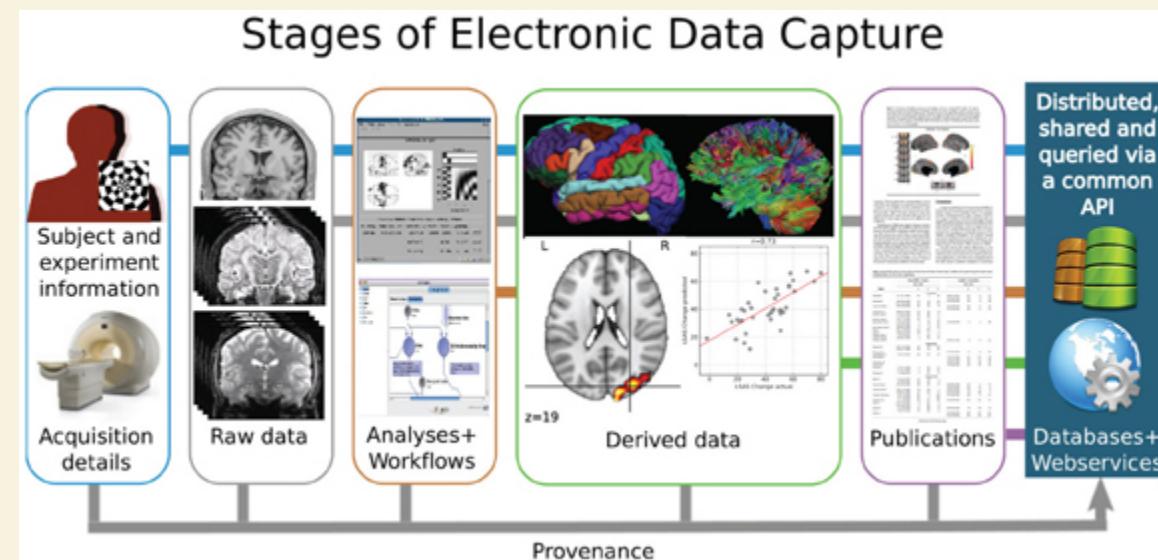
KEEPING A PROJECT CLEAN AND ORDERLY



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



COMPLEX ANALYSIS → CHAOTIC PROJECTS

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

Image credit: CC-BY Scriberia and The Turing Way



KEEPING A PROJECT CLEAN AND ORDERLY

KEEPING A PROJECT CLEAN AND ORDERLY

TRACK PROJECT HISTORY

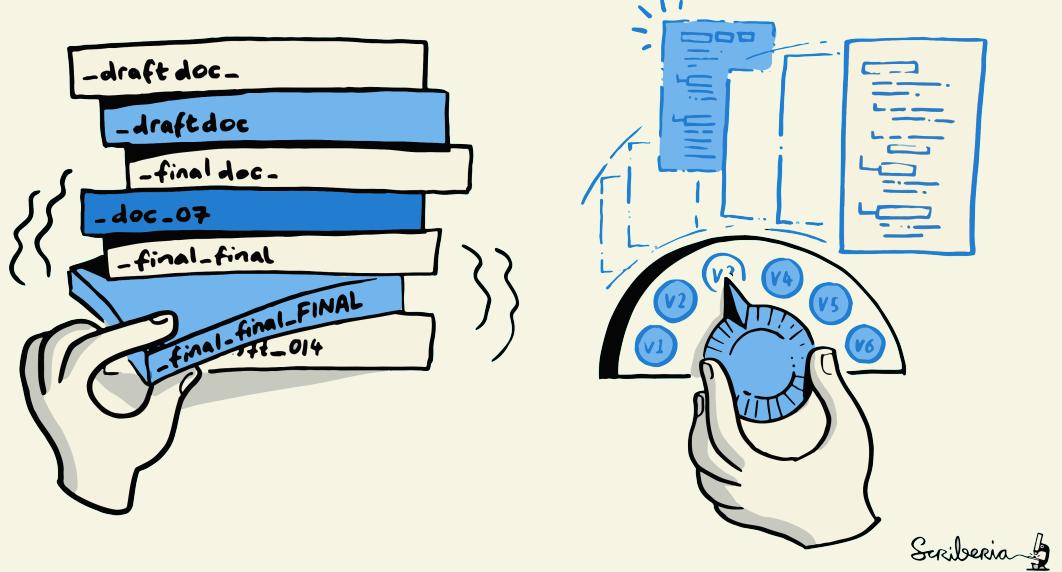


Image credit: CC-BY Scriberia & The Turing Way

Version control

KEEPING A PROJECT CLEAN AND ORDERLY

TRACK PROJECT HISTORY

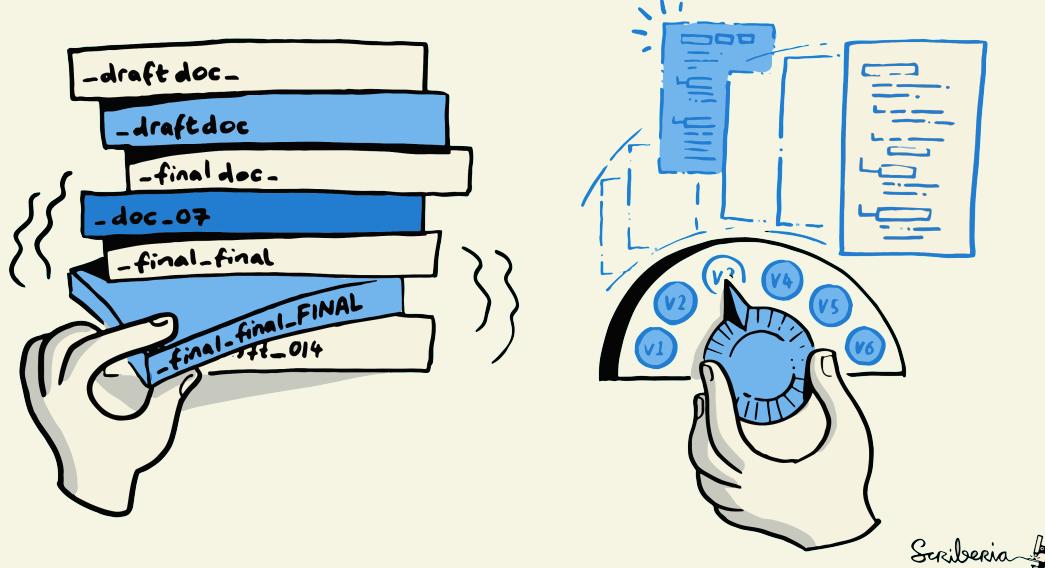


Image credit: CC-BY Scriberia & The Turing Way

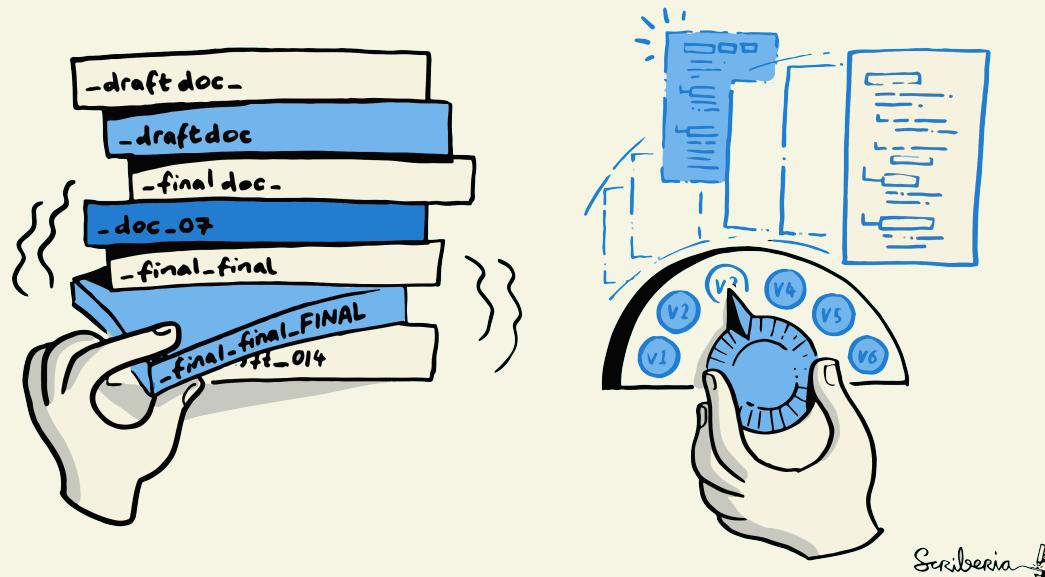
Version control

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

KEEPING A PROJECT CLEAN AND ORDERLY

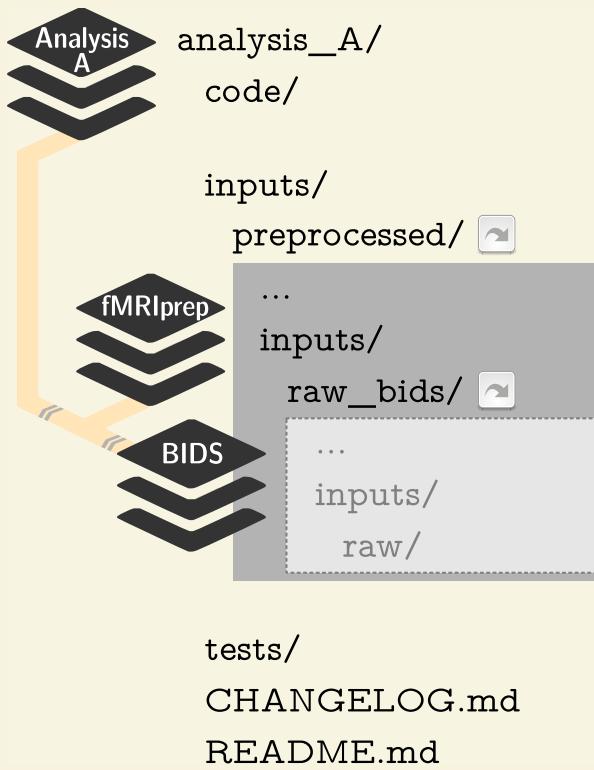
TRACK PROJECT HISTORY

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

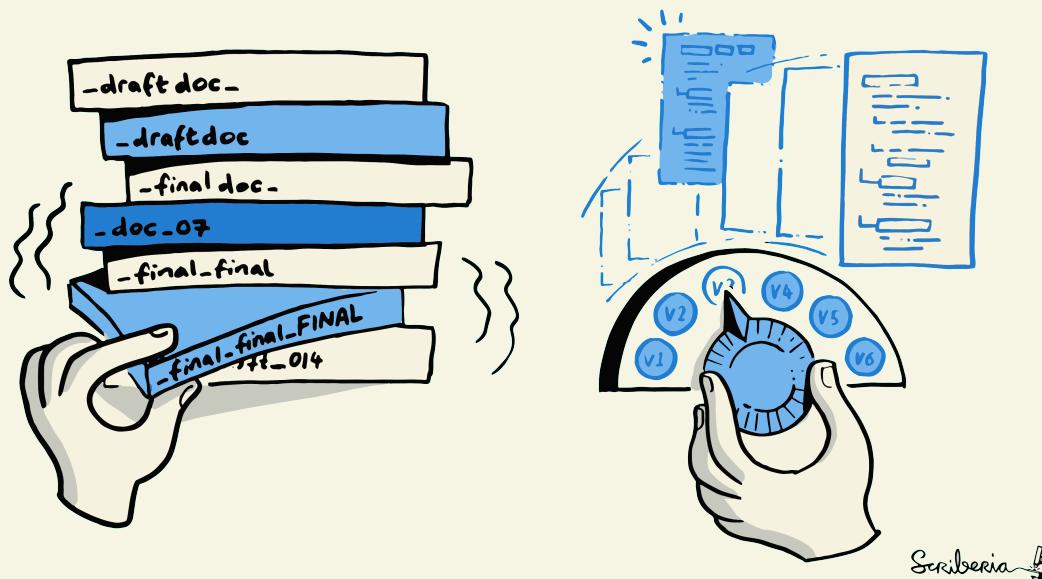


Intuitive structure

KEEPING A PROJECT CLEAN AND ORDERLY

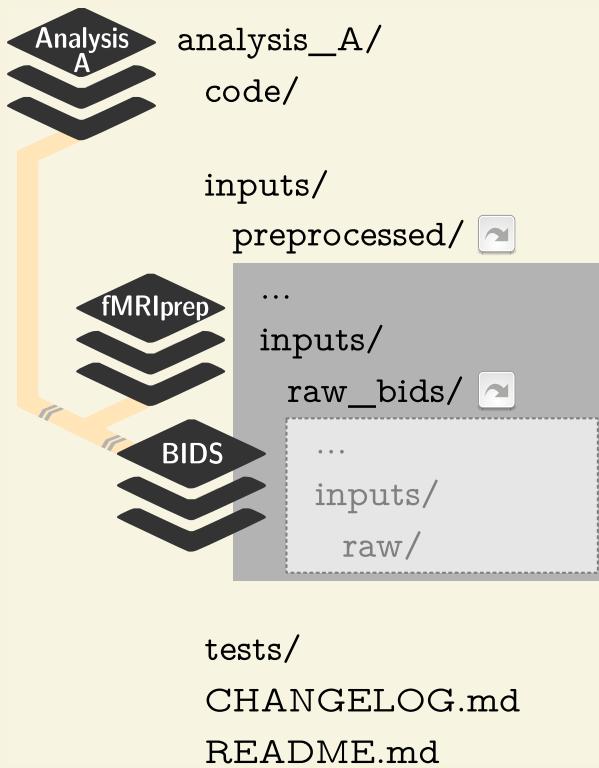
TRACK PROJECT HISTORY

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
```

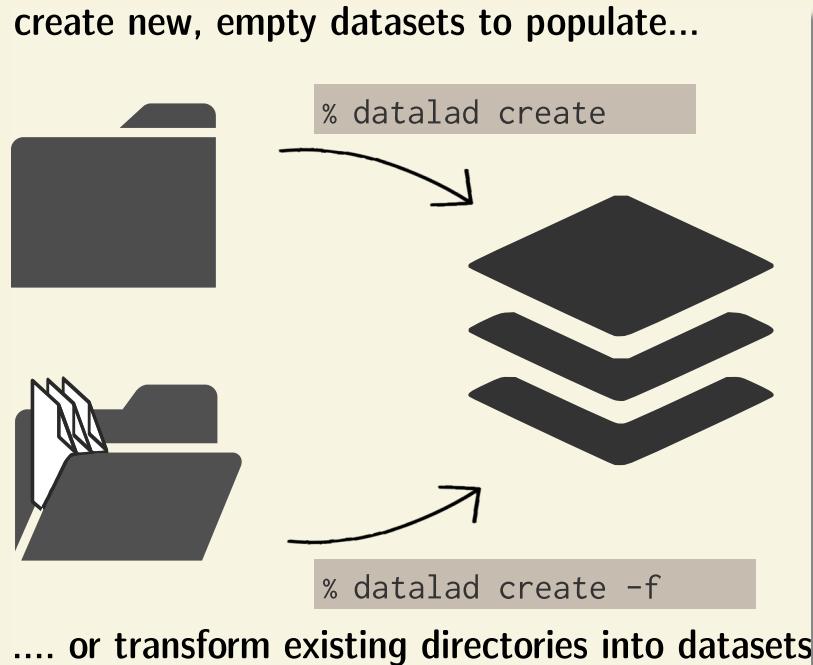
VERSION CONTROL

- DataLad knows two things: Datasets and files

VERSION CONTROL

- DataLad knows two things: Datasets and files

create new, empty datasets to populate...

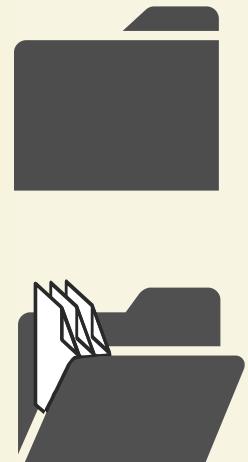


.... or transform existing directories into datasets

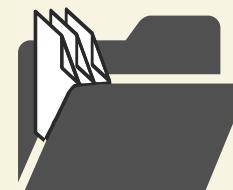
VERSION CONTROL

- DataLad knows two things: Datasets and files

create new, empty datasets to populate...

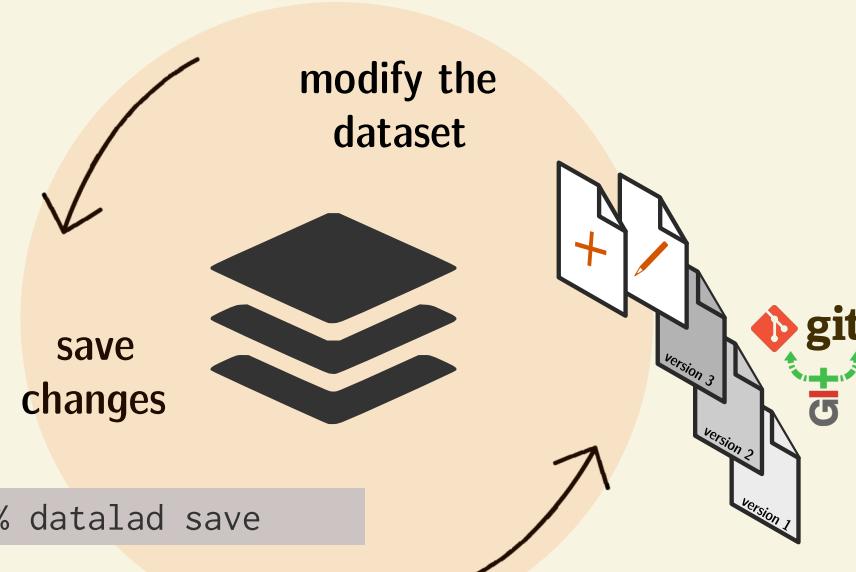


% datalad create



% datalad create -f

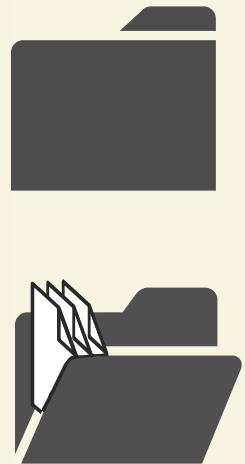
.... or transform existing directories into datasets



VERSION CONTROL

- DataLad knows two things: Datasets and files

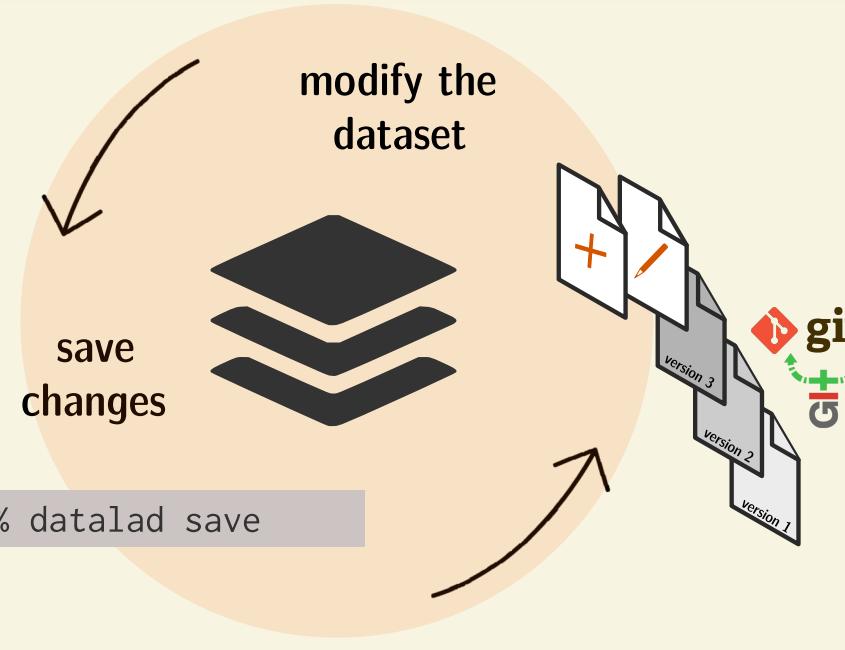
create new, empty datasets to populate...



% datalad create

% datalad create -f

.... or transform existing directories into datasets



- 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

```
1 # create a data analysis script
2 $ datalad status
3 untracked: code/script.py (file)
4 $ git status
5 On branch master
6 Untracked files:
7   (use "git add file..." to include in what will be committed)
8     code/script.py
9
10 nothing added to commit but untracked files present (use "git add" to track)
```

VERSION CONTROL

- Example: Add a new file into a dataset

```
1 # create a data analysis script
2 $ datalad status
3 untracked: code/script.py (file)
4 $ git status
5 On branch master
6 Untracked files:
7   (use "git add file..." to include in what will be committed)
8     code/script.py
9
10 nothing added to commit but untracked files present (use "git add" to track)
```

VERSION CONTROL

- Example: Add a new file into a dataset

```
1 # create a data analysis script
2 $ datalad status
3 untracked: code/script.py (file)
4 $ git status
5 On branch master
6 Untracked files:
7   (use "git add file..." to include in what will be committed)
8     code/script.py
9
10 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)
```

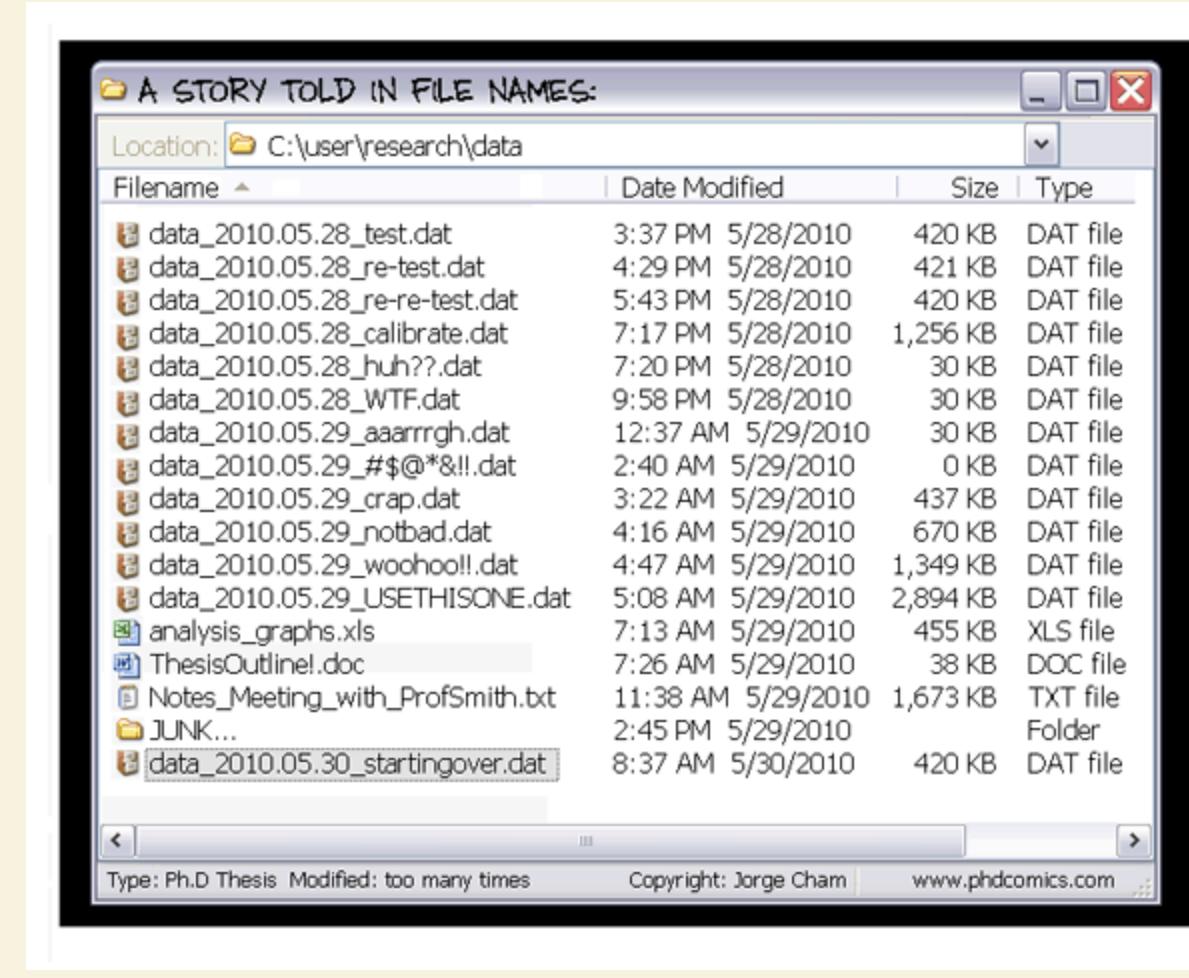
Why should you version control more than just your code?

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



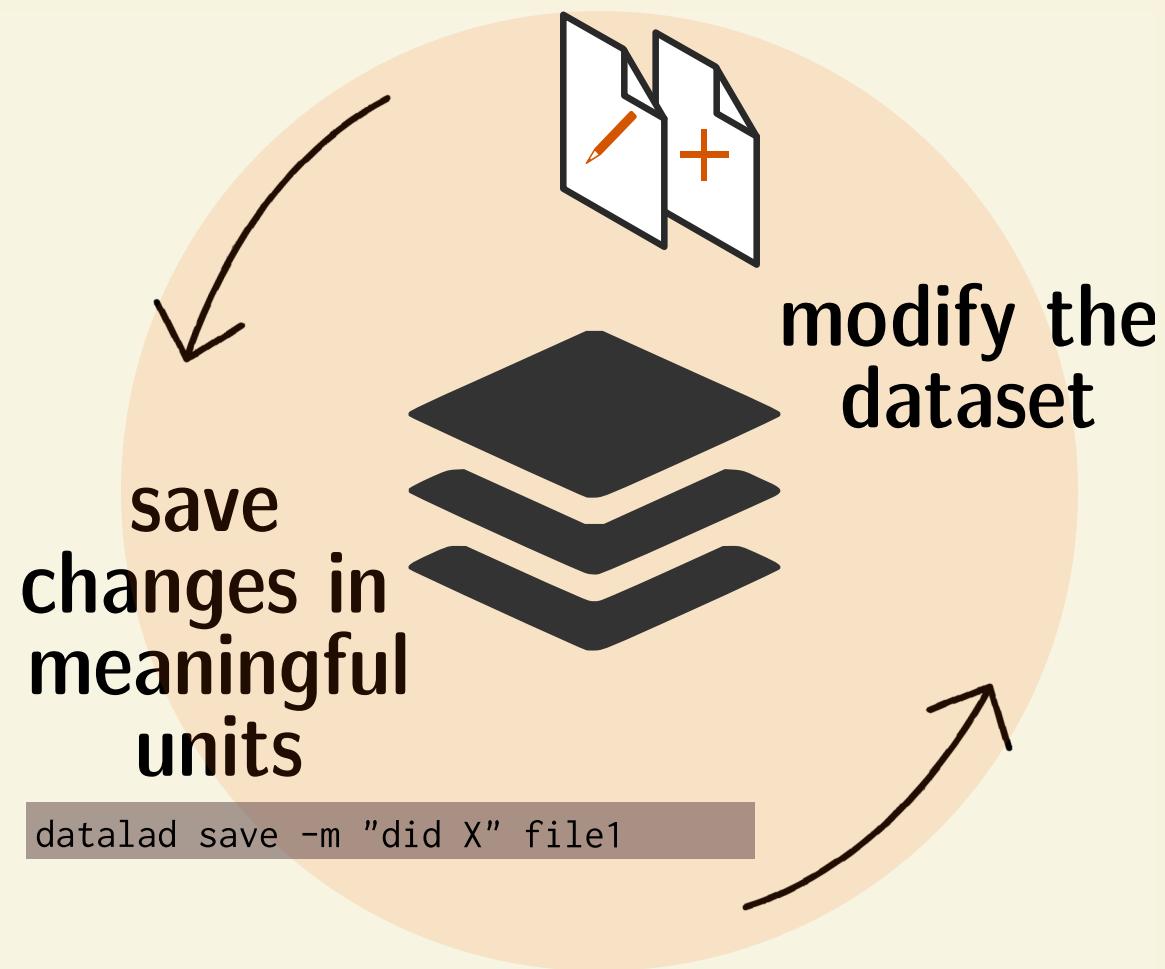
Version controlling data allows to track data changes and uniquely identify precise versions that were used in your analysis

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!

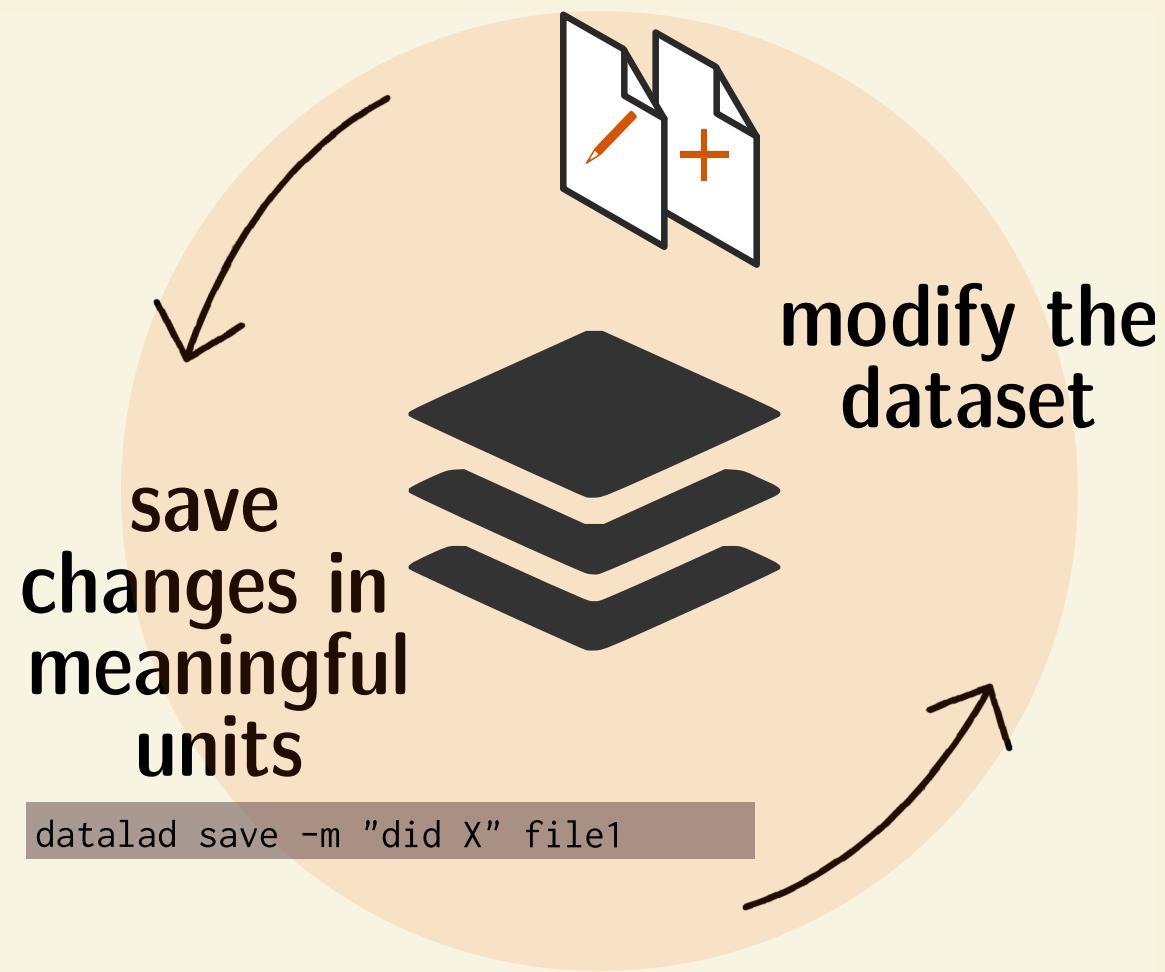
LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!



LOCAL VERSION CONTROL

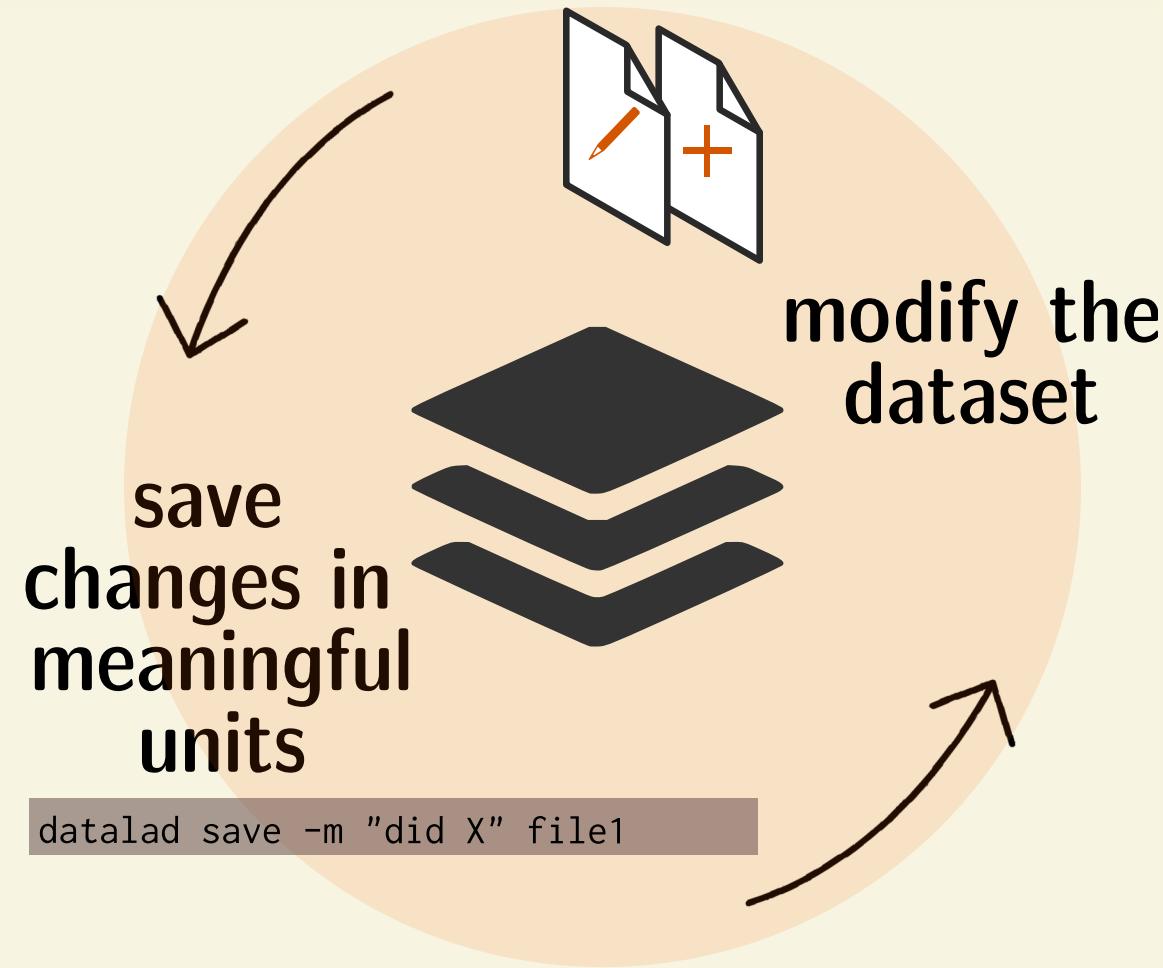
Procedurally, version control is easy with DataLad!



Stay flexible:

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!

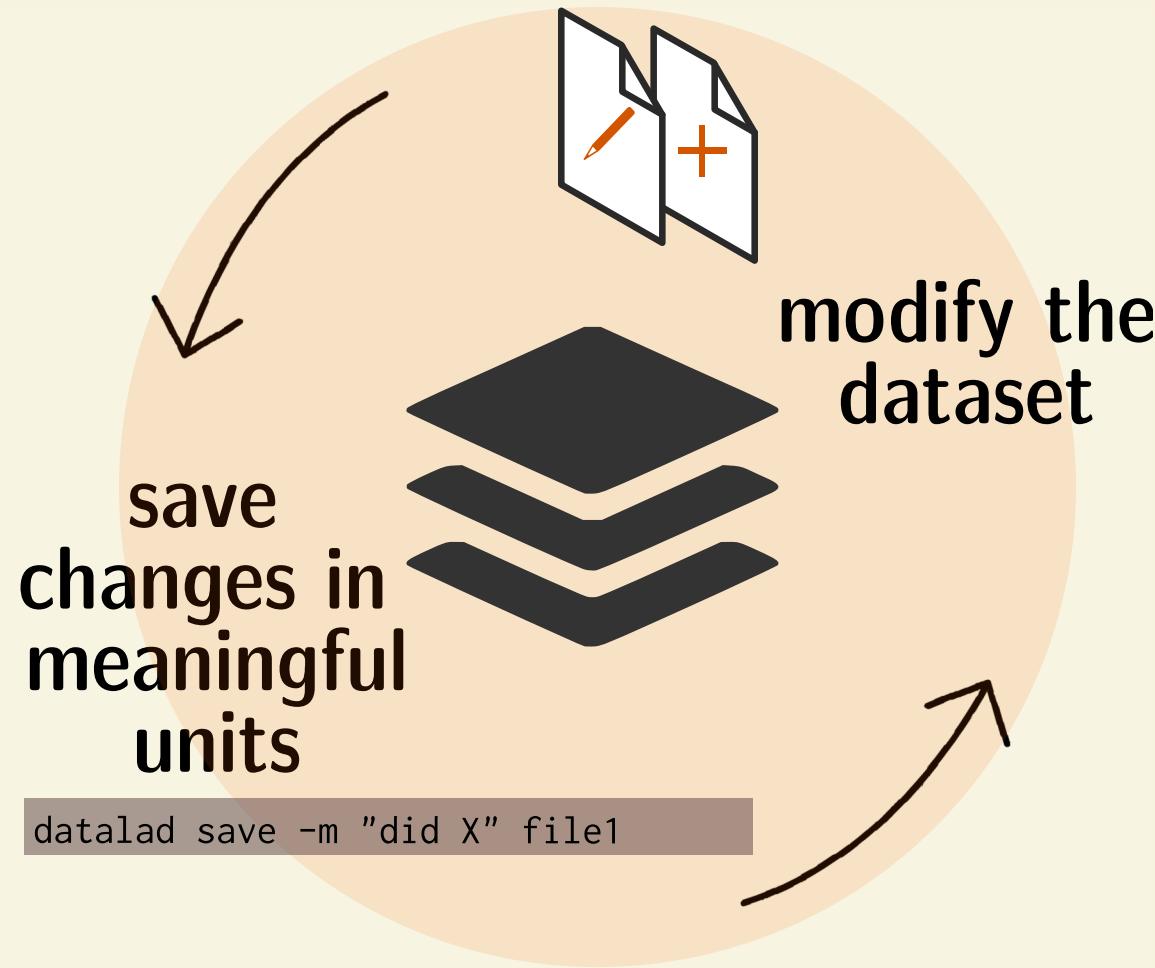


Stay flexible:

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

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!

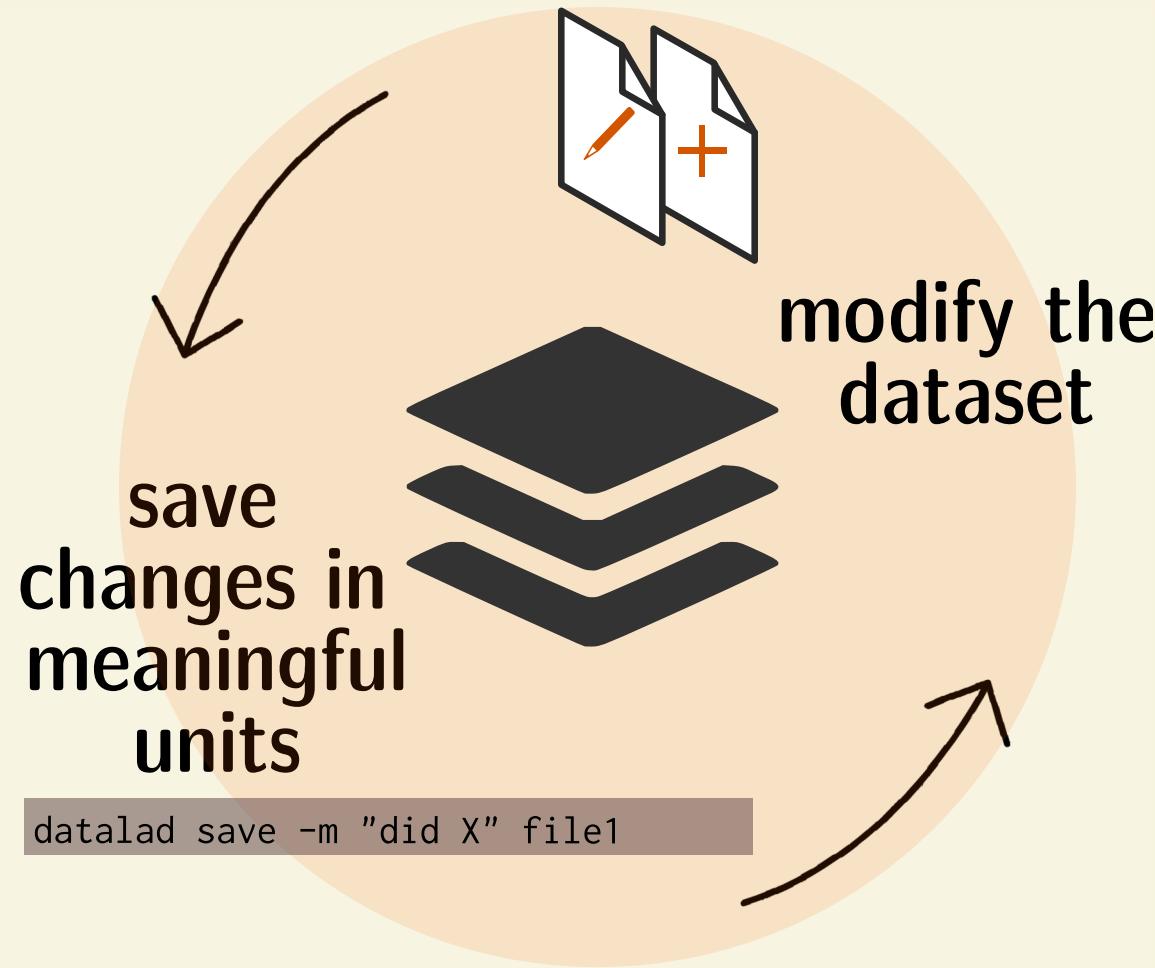


Stay flexible:

- 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!



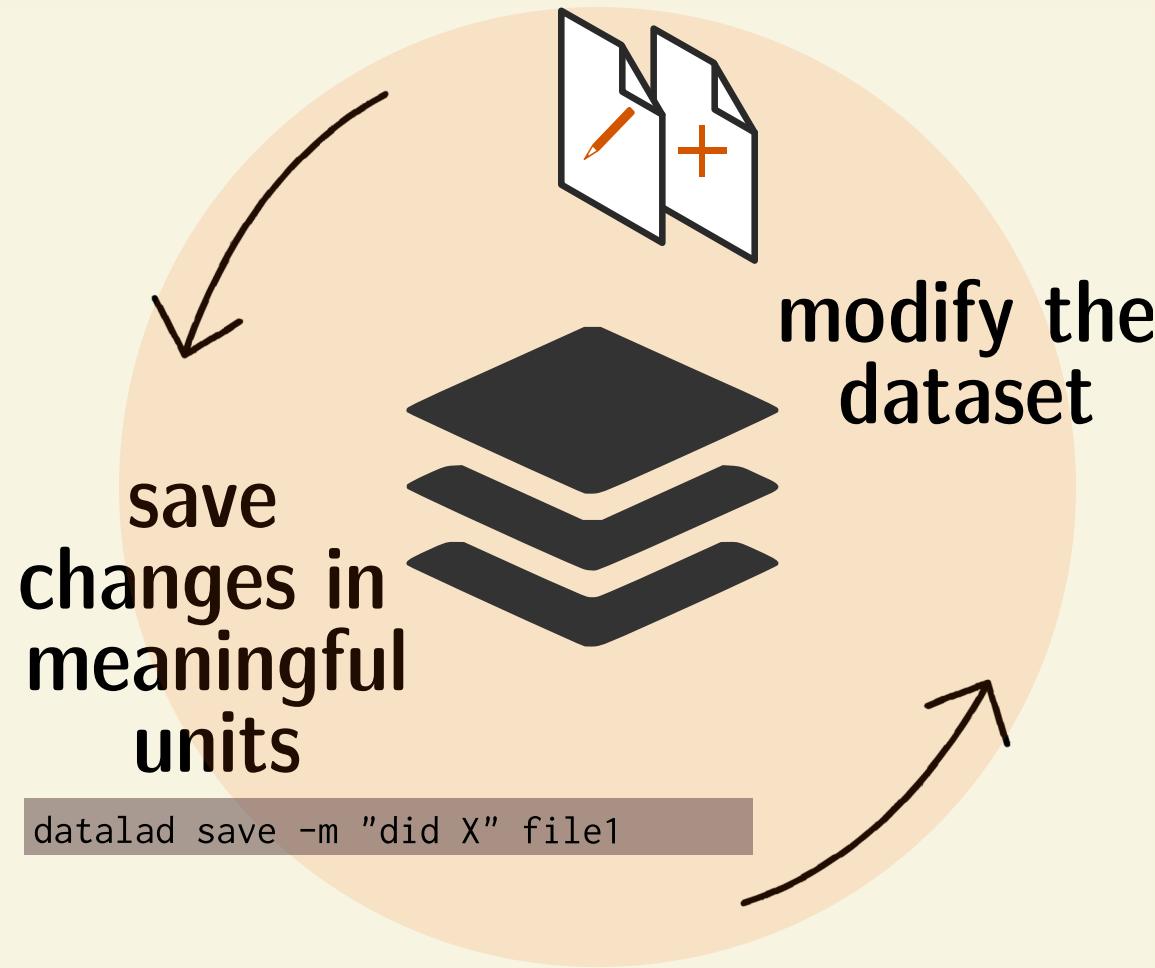
Stay flexible:

- 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:

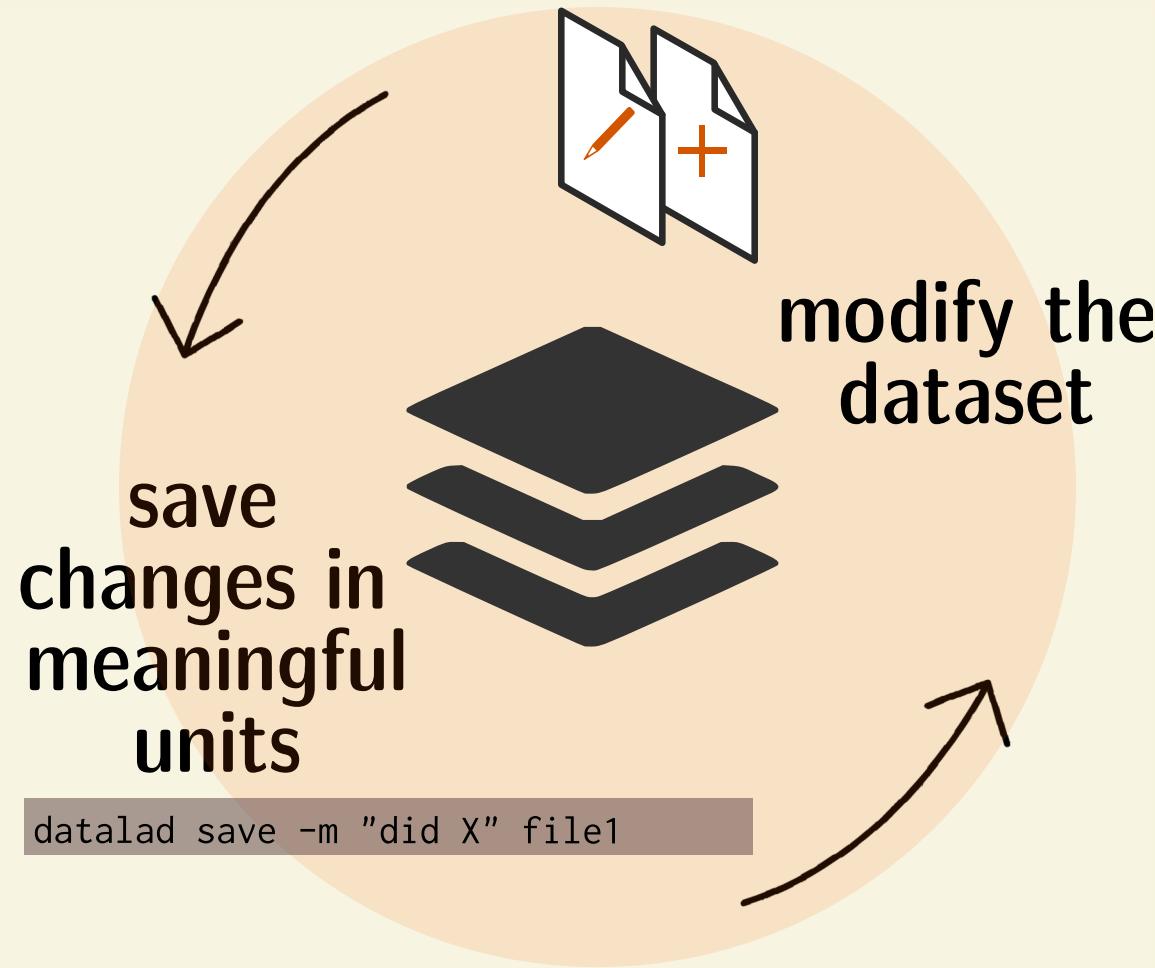
- Non-complex DataLad core API (easier than Git)
- Pure Git or git-annex commands (for regular Git or git-annex users, or to use specific functionality)

- 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)

- Advice:**
- Save *meaningful* units of change
 - Attach helpful commit messages

VERSION CONTROL

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

Date	Author	Change summary (commit message)
2020-06-05 10:58 +0200	Adina Wagner	M [master] {upstream/master} {upstream/HEAD} Merge pull request #12 from psychoinformatics-d
2020-06-05 08:24 +0200	Adina Wagner	o [finalround] {upstream/finalround} add results from computing with mean instead of median
2020-06-05 09:09 +0200	Michael Hanke	o Change wording, clarify comment
2020-06-05 07:26 +0200	Michael Hanke	M Merge remote-tracking branch 'gh-mine/finalround'
2020-05-28 16:39 +0200	Asim H Dar	o Added datalad.get() so S2SRMS() pulls data and can run standalone
2020-05-18 08:25 +0200	Adina Wagner	o {gh-asim/finalround} S2SRMS: example implementation of the S2SRMS method suggested by R2
2020-05-01 17:38 +0200	Adina Wagner	o Minor edits as suggested by reviewer 2
2020-05-29 09:04 +0200	Adina Wagner	M Merge pull request #13 from psychoinformatics-de/adswa-patch-1
2020-05-24 09:15 +0200	Adina Wagner	o {upstream/adswa-patch-1} Fix installation instructions
2020-05-24 09:53 +0200	Adina Wagner	M Merge pull request #14 from psychoinformatics-de/bf-data
2020-05-24 09:33 +0200	Adina Wagner	o [bf-data] One-time datalad import
2020-05-24 09:32 +0200	Adina Wagner	o install and get relevant subdataset data
2020-03-18 10:19 +0100	Michael Hanke	M Merge pull request #8 from psychoinformatics-de/adswa-patch-1
2019-12-19 10:22 +0100	Adina Wagner	o {gh-asim/adswa-patch-1} add sklearn to requirements
2020-03-18 10:13 +0100	Michael Hanke	o Tune new figure caption
2020-03-18 10:03 +0100	Michael Hanke	M Merge pull request #11 from ElectronicTeaCup/revision_2
2020-03-18 09:59 +0100	Adina Wagner	M [revision_2] {gh-asim/revision_2} Merge branch 'revision_2' of github.com:ElectronicTe
2020-03-18 09:58 +0100	Michael Hanke	o Last detections
2020-03-18 09:59 +0100	Adina Wagner	o name parameter in caption

- 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

"FINAL".doc



FINAL.doc!



FINAL_rev.2.doc



FINAL_rev.6.COMMENTS.doc



FINAL_rev.8.comments5.
CORRECTIONS.doc



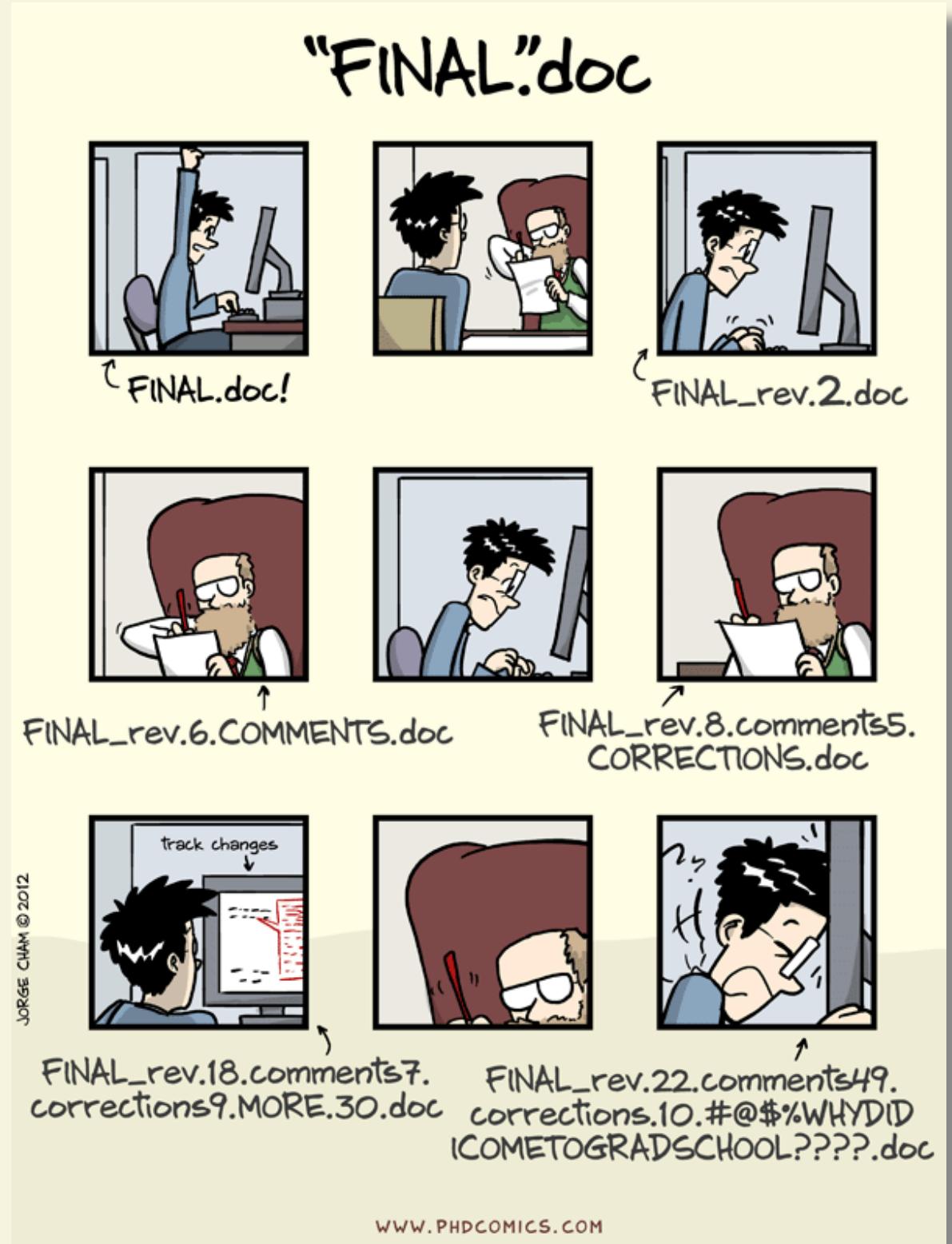
JORGE CHAM © 2012

FINAL_rev.18.comments7.
corrections9.MORE.30.doc FINAL_rev.22.comments49.
corrections.10.#@\$%WHYDID
ICOMETOGRAD SCHOOL????.doc

FROM HERE TO THIS:



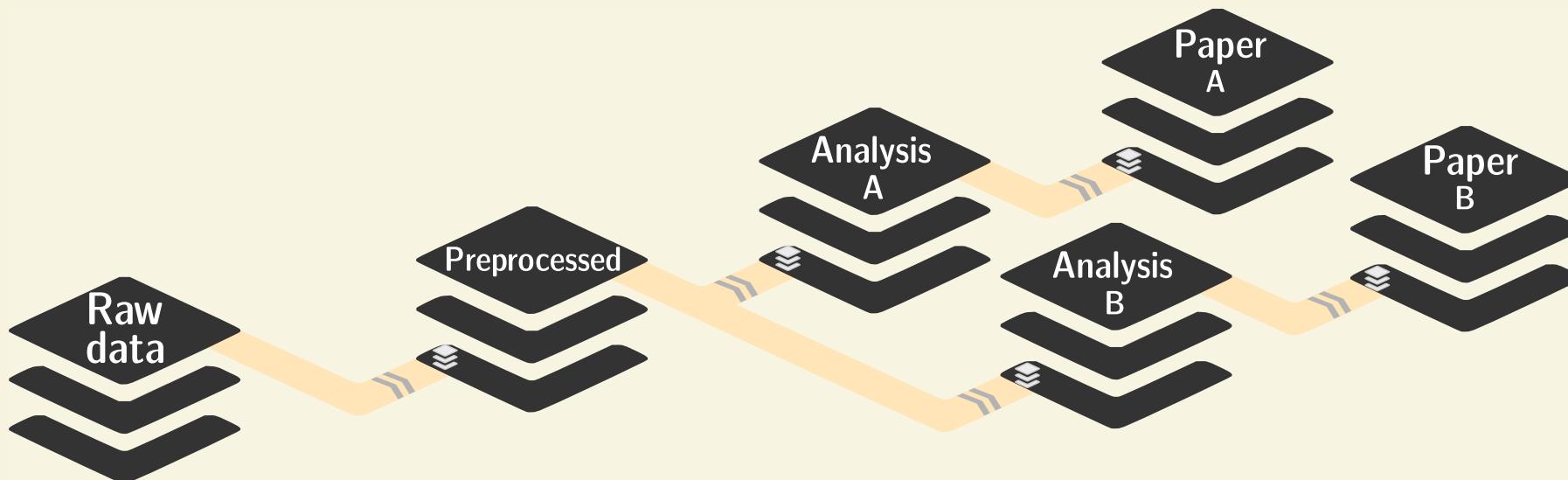
FROM HERE TO THIS:



BUT: Version control is only one aspect of data management

INTUITIVE DATA ANALYSIS STRUCTURE

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

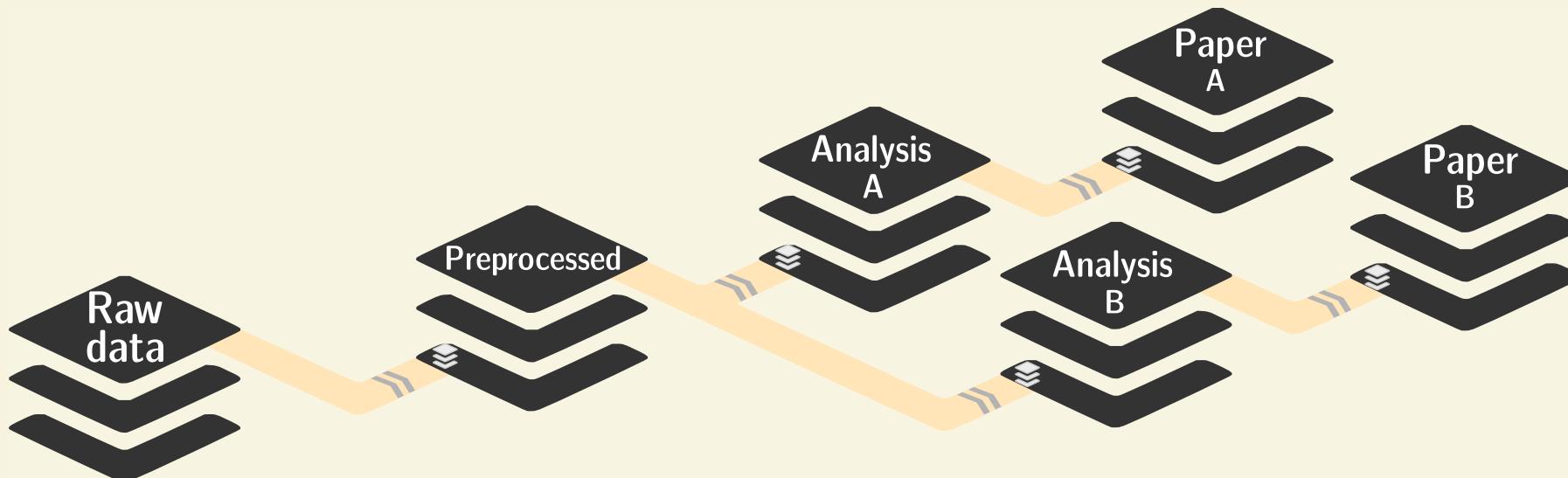


Nest modular datasets to create a linked hierarchy of datasets,
and enable recursive operations throughout the hierarchy

```
1 $ cd myanalysis
2 # we can install analysis input data as a subdataset to the dataset
3 $ datalad clone -d . https://github.com/datalad-handbook/iris_data.git input/
4 [INFO    ] Scanning for unlocked files (this may take some time)
5 [INFO    ] Remote origin not usable by git-annex; setting annex-ignore
6 install(ok): input (dataset)
7 add(ok): input (file)
8 add(ok): .gitmodules (file)
9 save(ok): . (dataset)
10 action summary:
11   add (ok: 2)
12   install (ok: 1)
13   save (ok: 1)
```

INTUITIVE DATA ANALYSIS STRUCTURE

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



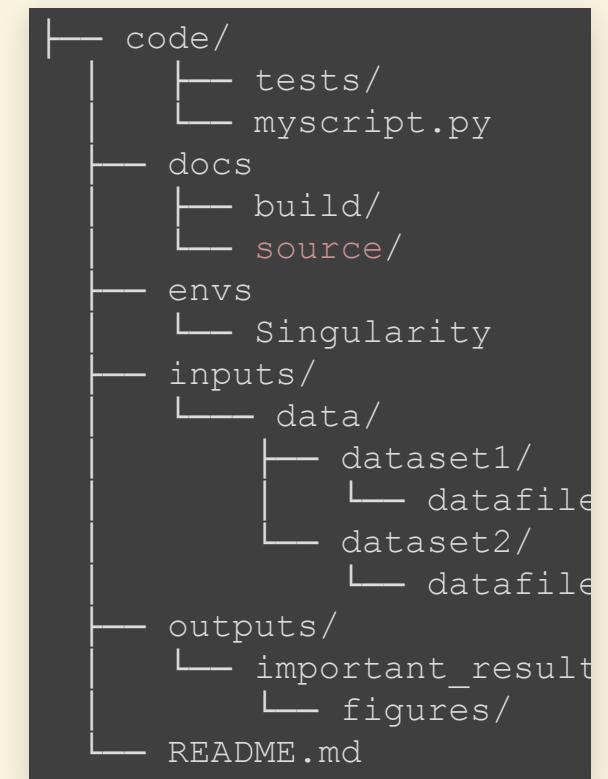
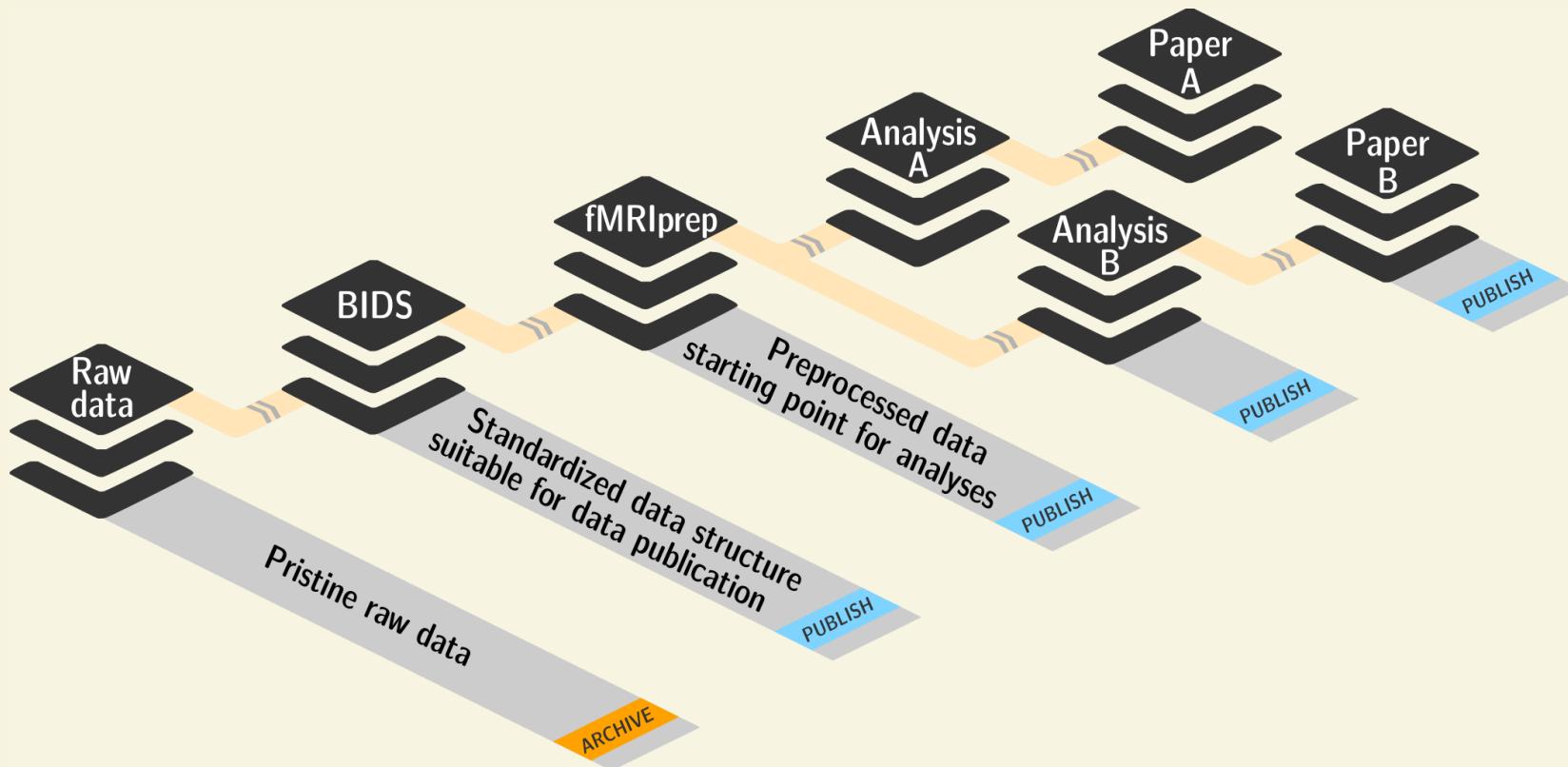
Nest modular datasets to create a linked hierarchy of datasets,
and enable recursive operations throughout the hierarchy

```
$ 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!](#)

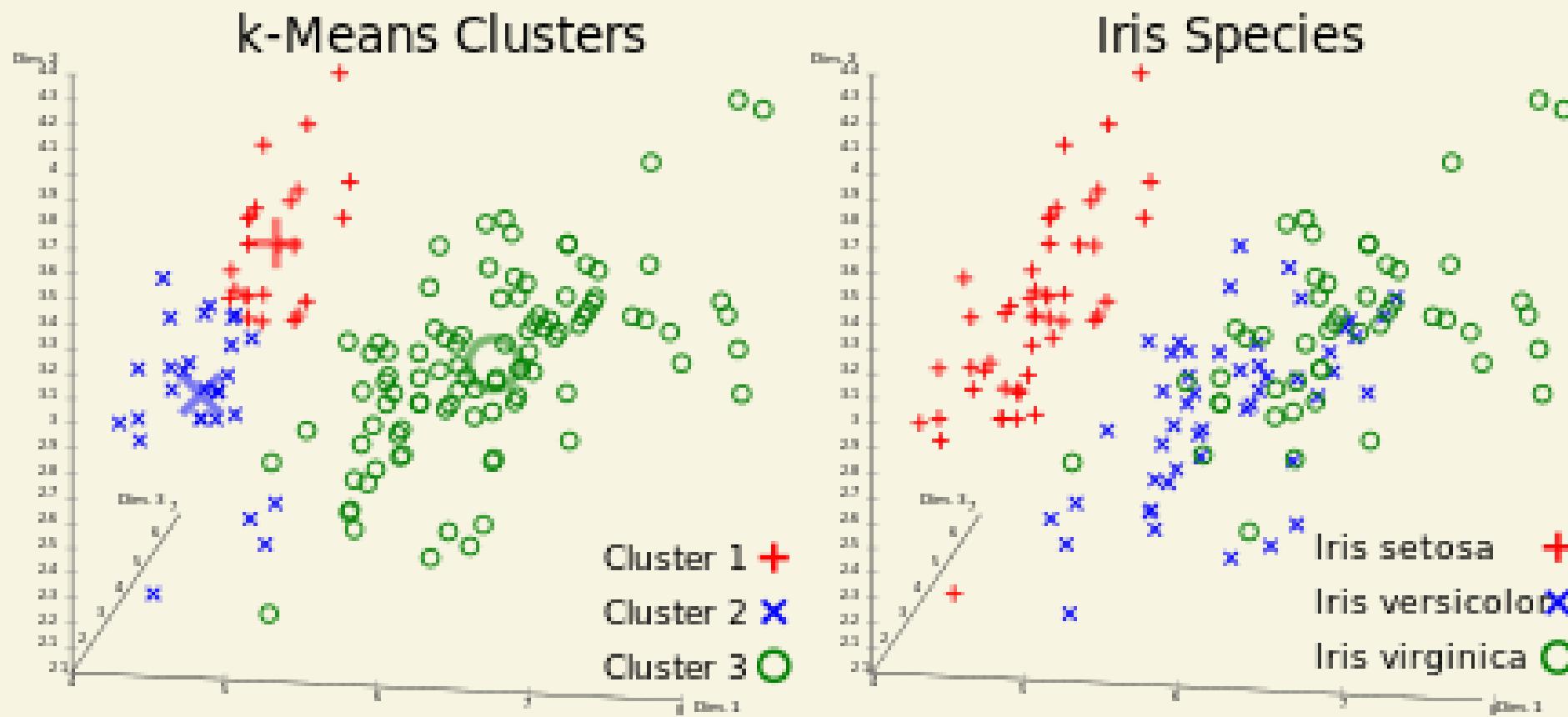
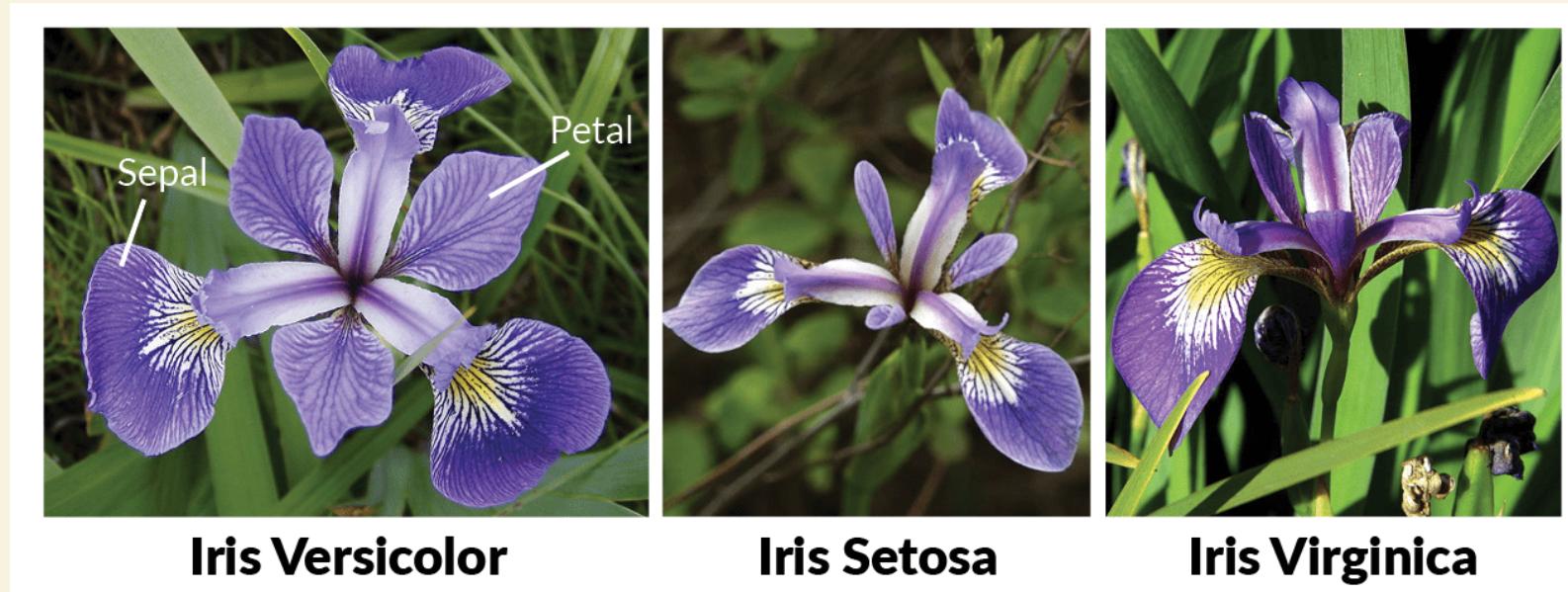
COMPUTATIONALLY REPRODUCIBLE DATA ANALYSIS

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

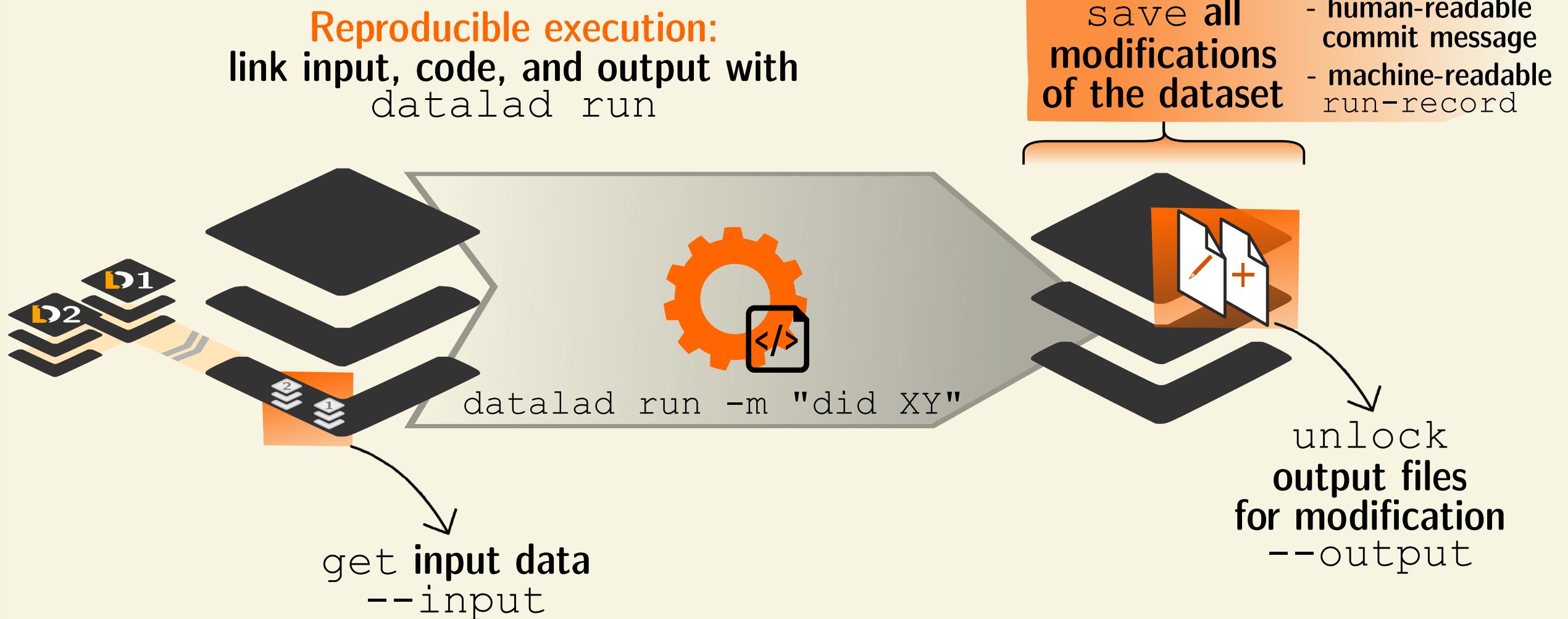


REPRODUCIBLE EXECUTION & PROVENANCE CAPTURE

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?

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

COMPUTATIONAL REPRODUCIBILITY

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

```
1 $ datalad run -m "analyze iris data with classification analysis" \
2   --input "input/iris.csv" \
3   --output "prediction_report.csv" \
4   --output "pairwise_relationships.png" \
5   "python3 code/script.py"
6 [INFO    ] Making sure inputs are available (this may take some time)
7 get(ok): input/iris.csv (file) [from web...]
8 [INFO    ] == Command start (output follows) =====
9 [INFO    ] == Command exit (modification check follows) =====
10 add(ok): pairwise_relationships.png (file)
11 add(ok): prediction_report.csv (file)
12 save(ok): . (dataset)
13 action summary:
14   add (ok: 2)
15   get (notneeded: 2, ok: 1)
16   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": "9ffdbfcf-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`

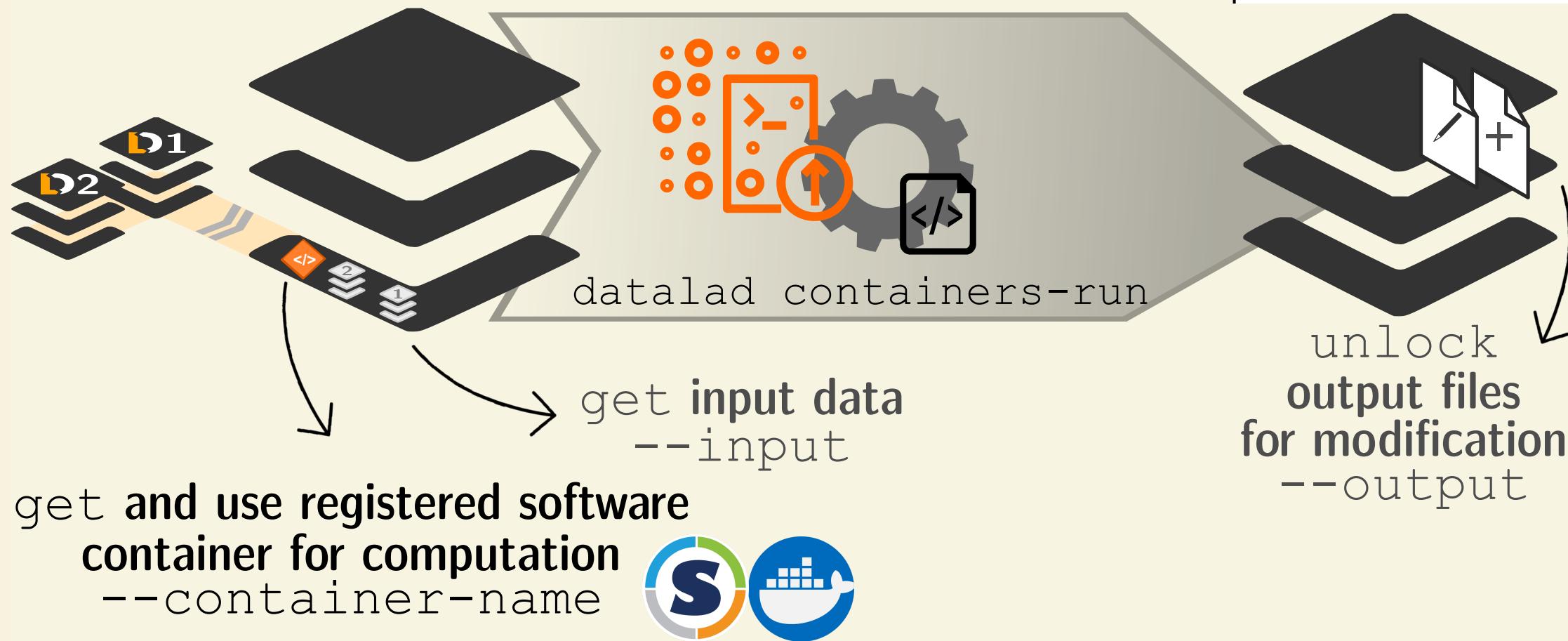
`datalad-containers run`

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**



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

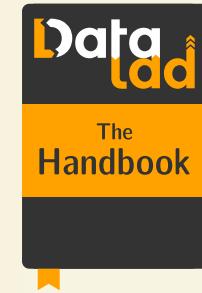
IS THERE MORE?

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