DATALAD BASICS

Code to follow along: handbook.datalad.org/r.html?MPIBerlin

PREREQUISITES: INSTALLATION AND CONFIGURATION

PREREQUISITES: INSTALLATION AND CONFIGURATION

- Your installed version of DataLad should be recent

```
datalad --version
0.13.5
```

PREREQUISITES: INSTALLATION AND CONFIGURATION

- Your installed version of DataLad should be recent

```
datalad --version
0.13.5
```

- You should have a configured Git identity

```
$ git config --list
user.name=Adina Wagner
user.email=adina.wagner@t-online.de
[...]
```

PREREQUISITES: INSTALLATION AND CONFIGURATION

- Your installed version of DataLad should be recent

```
datalad --version
0.13.5
```

- You should have a configured Git identity

```
$ git config --list
user.name=Adina Wagner
user.email=adina.wagner@t-online.de
[...]
```

Else, find installation and configuration instructions at handbook.datalad.org

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")
```

DATALAD DATASETS

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

DATALAD DATASETS

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

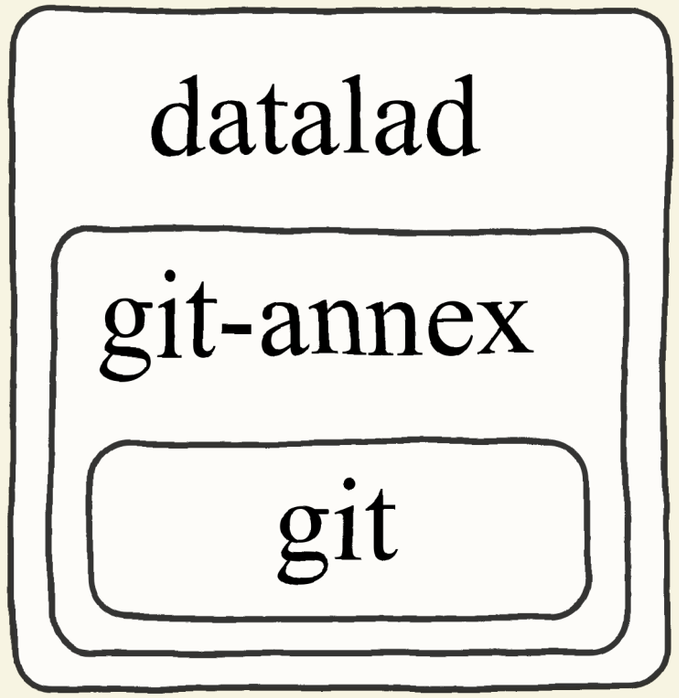
DATALAD DATASETS

- DataLad's core data structure
  - Dataset = A directory managed by DataLad
  - Any directory of your computer can be managed by DataLad.
  - Datasets can be *created* (from scratch) or *installed*
  - Datasets can be nested: *linked subdirectories*

DATALAD DATASETS

- DataLad's core data structure
  - Dataset = A directory managed by DataLad
  - Any directory of your computer can be managed by DataLad.
  - Datasets can be *created* (from scratch) or *installed*
  - Datasets can be nested: *linked subdirectories*
- Let's start by creating a dataset

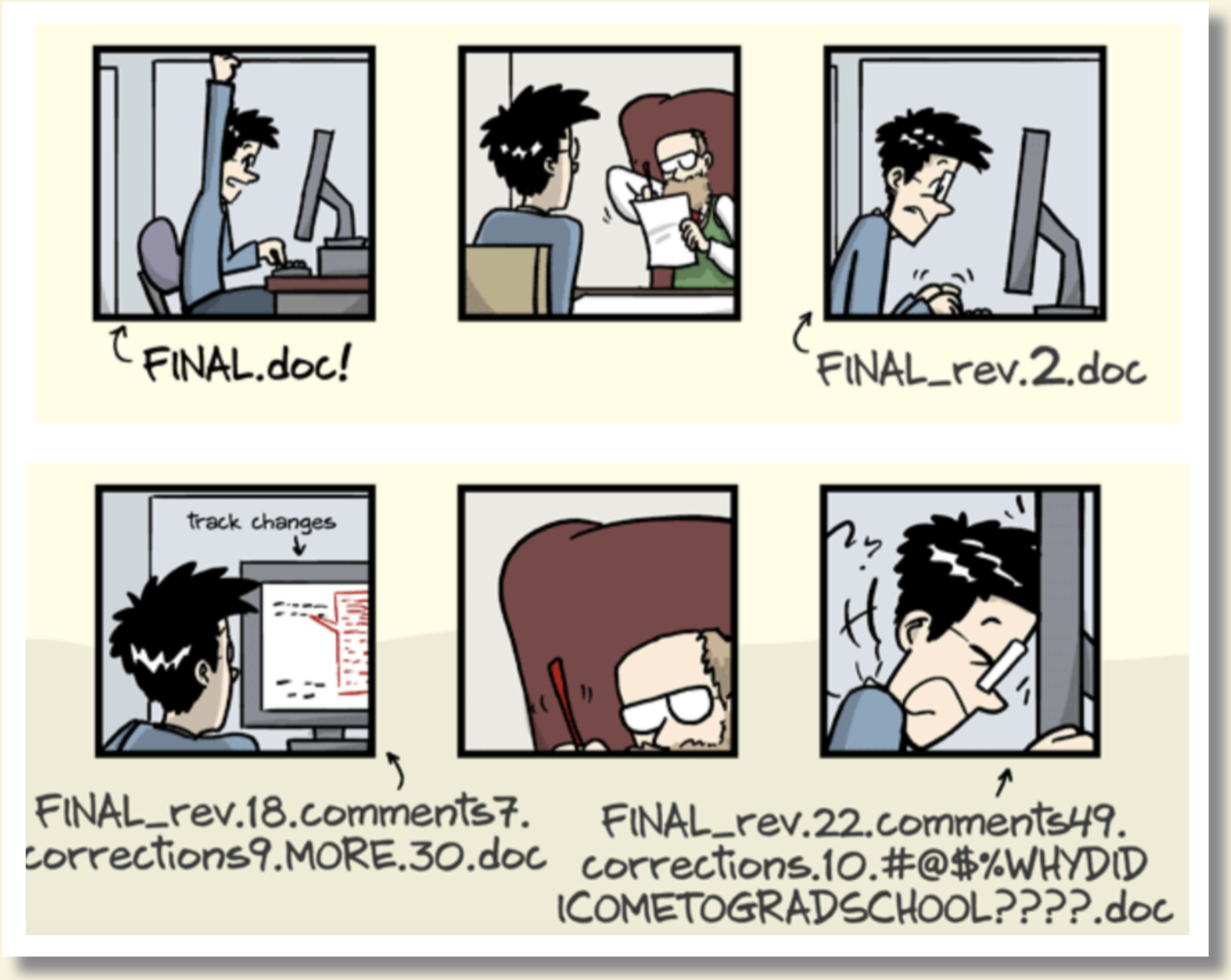
DATALAD DATASETS



A DataLad dataset is a joined Git + git-annex repository

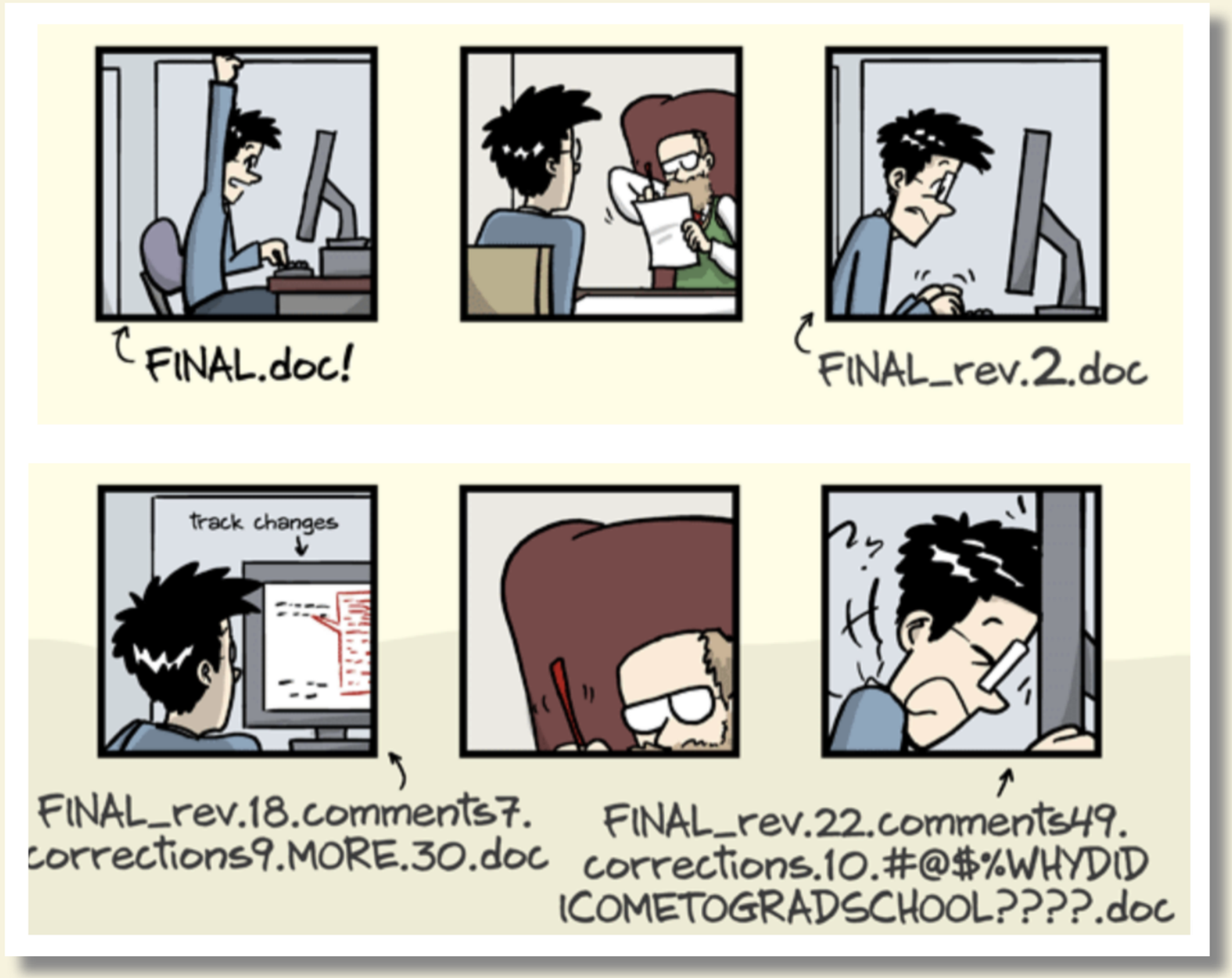
WHY VERSION CONTROL?

WHY VERSION CONTROL?



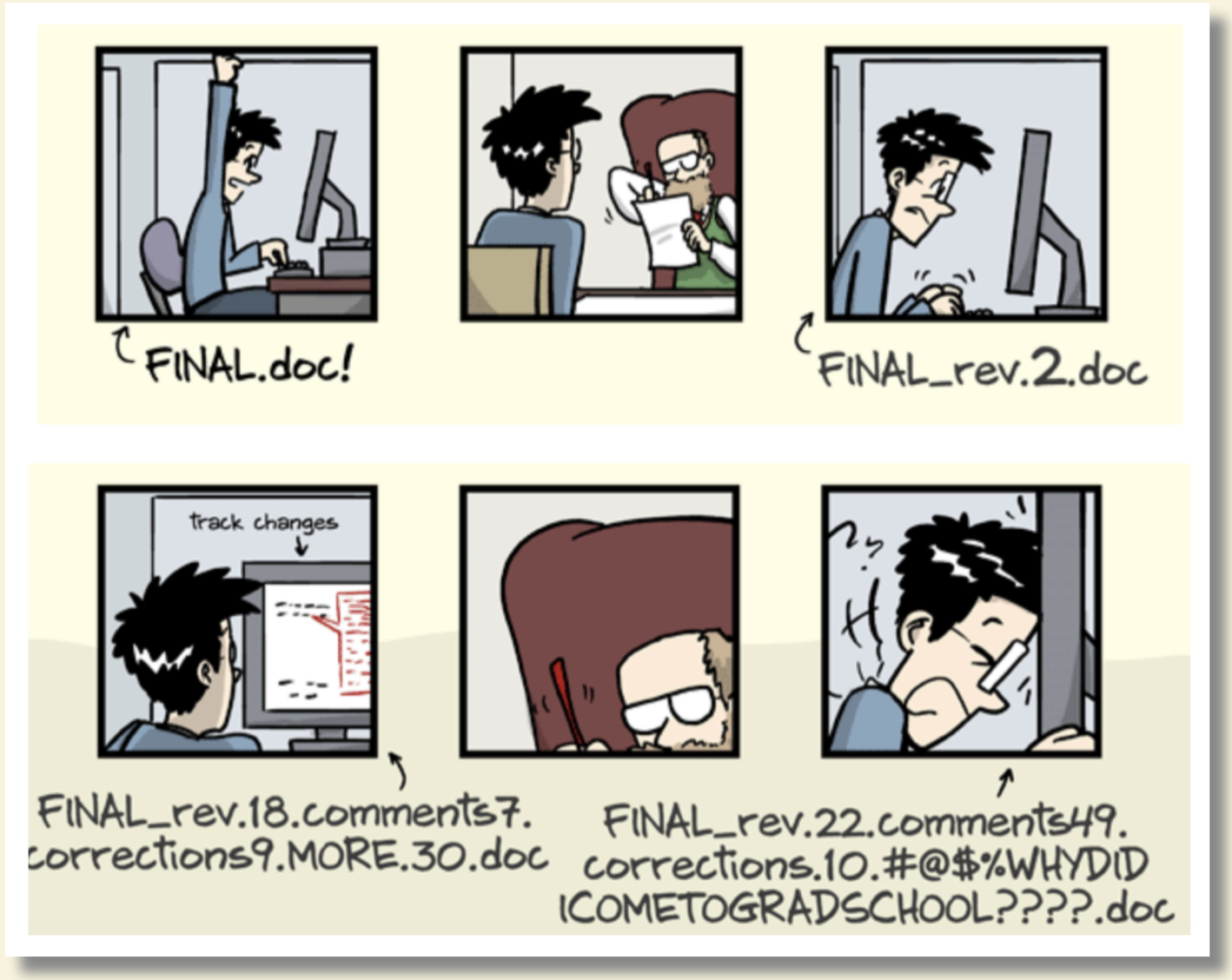
- keep things organized

WHY VERSION CONTROL?



- keep things organized
- keep track of changes

WHY VERSION CONTROL?



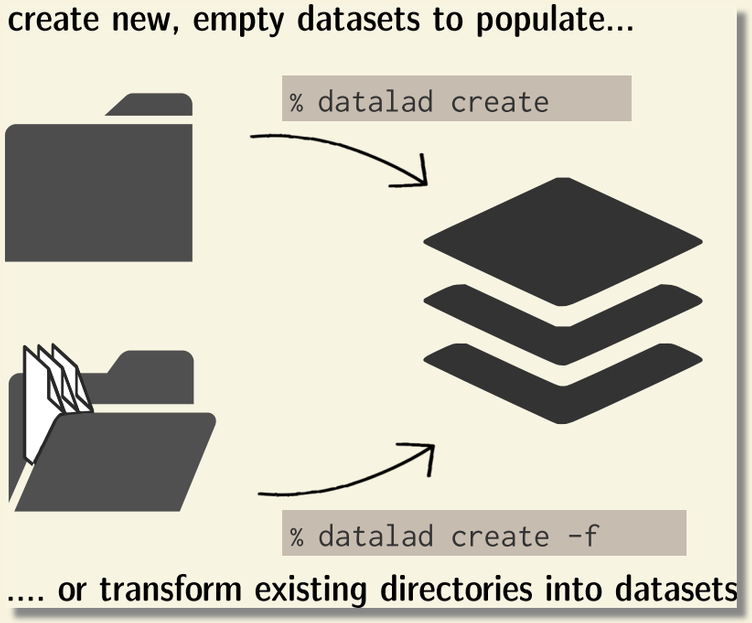
- keep things organized
- keep track of changes
- revert changes or go back to previous states

VERSION CONTROL

- DataLad knows two things: Datasets and files

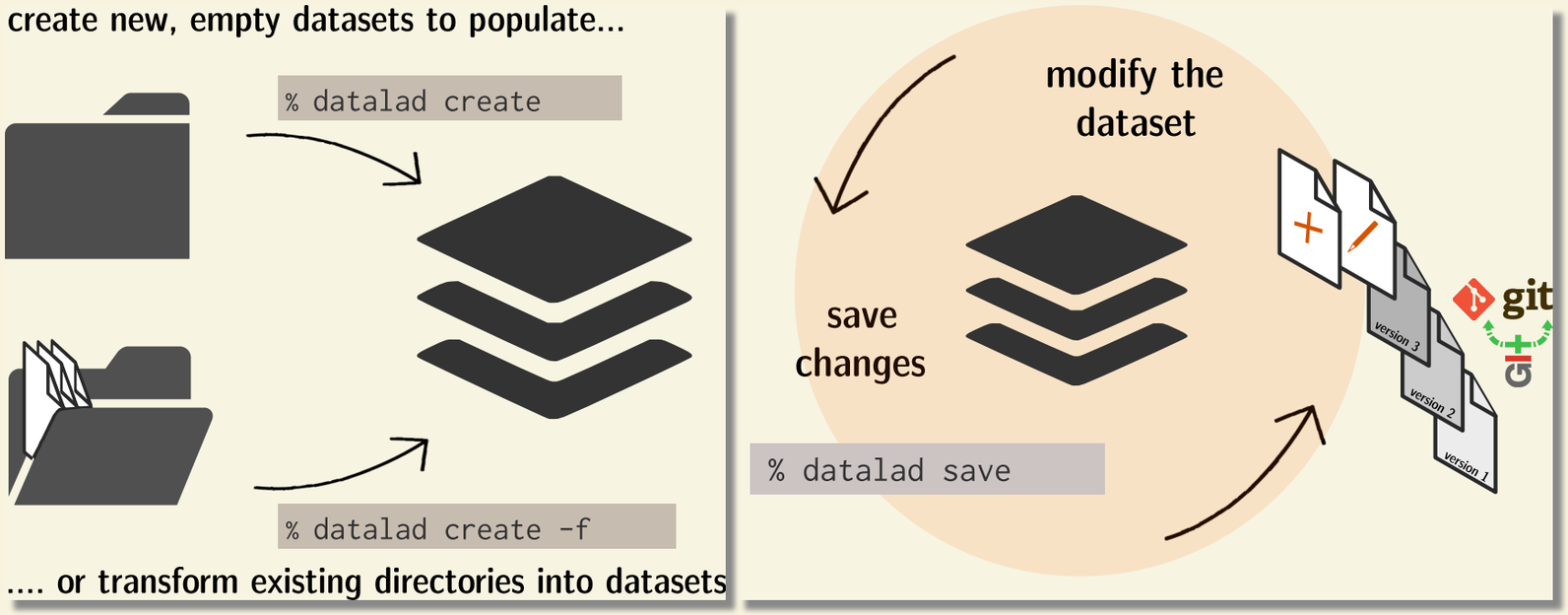
VERSION CONTROL

- DataLad knows two things: Datasets and files



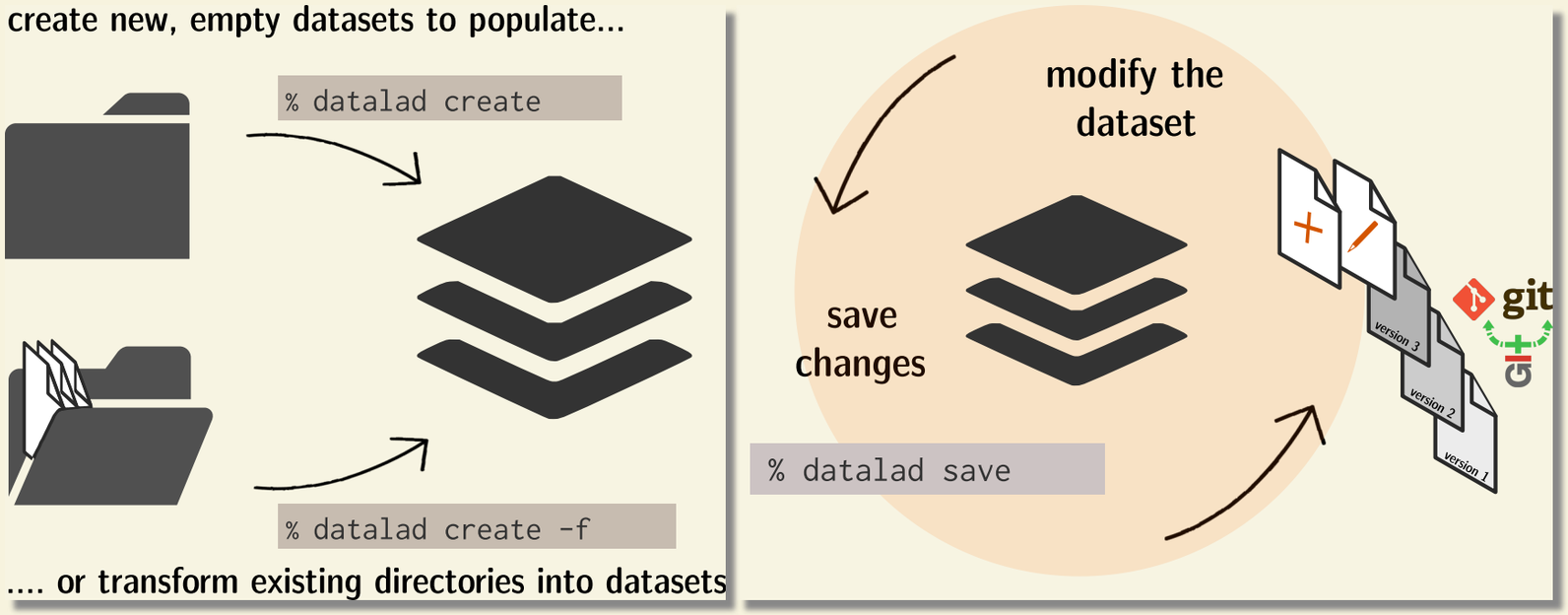
VERSION CONTROL

- DataLad knows two things: Datasets and files



VERSION CONTROL

- DataLad knows two things: Datasets and files



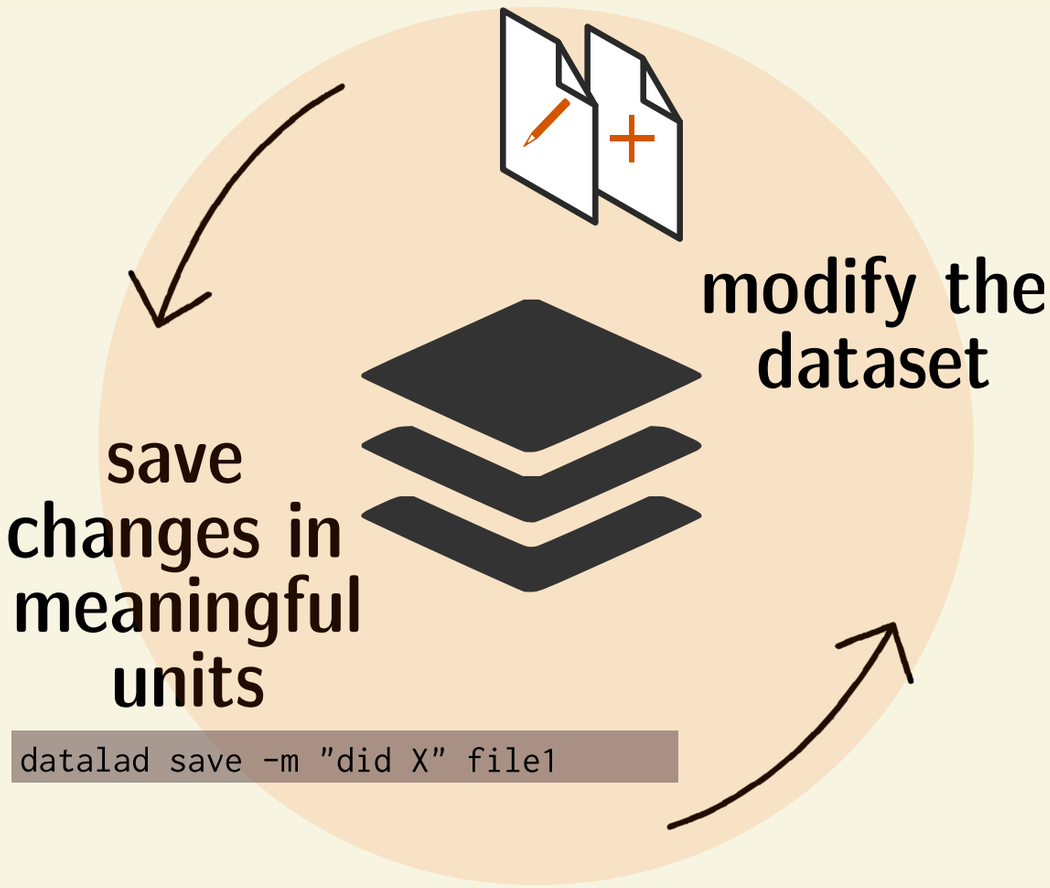
- Every file you put into a in a dataset can be easily version-controlled, regardless of size, with the same command.

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!

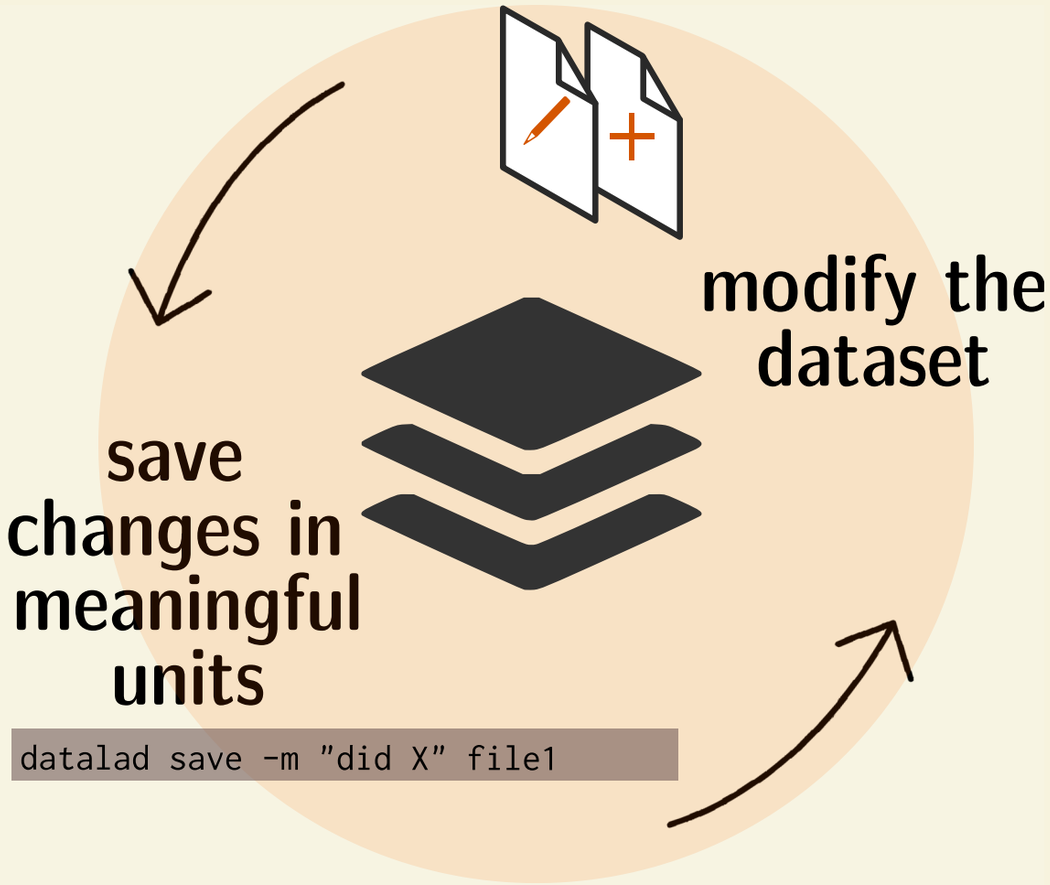
LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!



LOCAL VERSION CONTROL

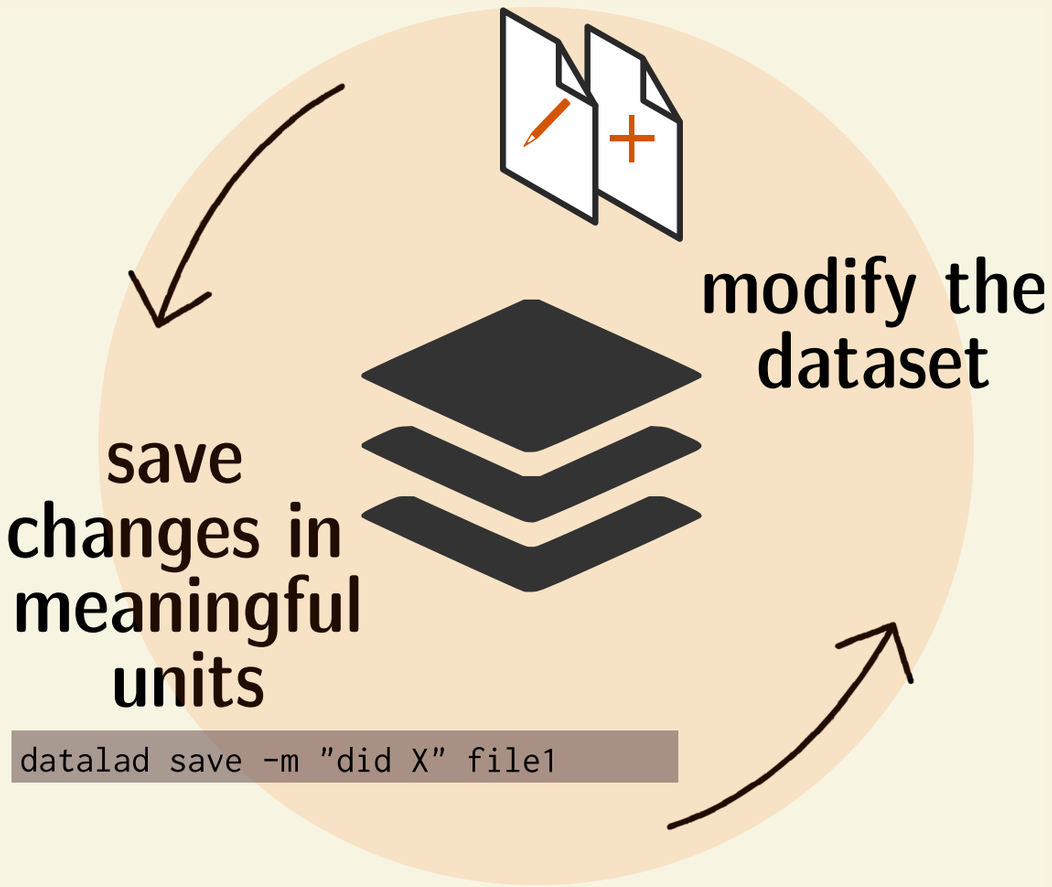
Procedurally, version control is easy with DataLad!



Advice:

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!

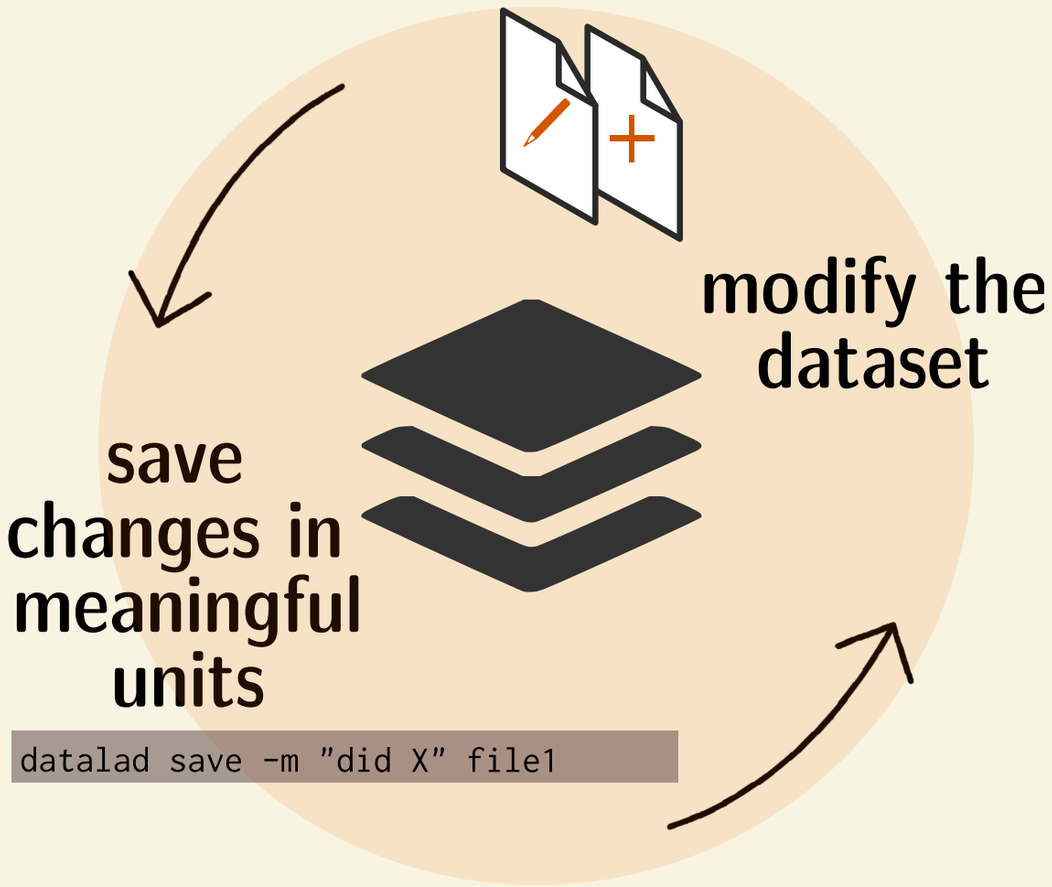


Advice:

- Save *meaningful* units of change

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!



Advice:

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

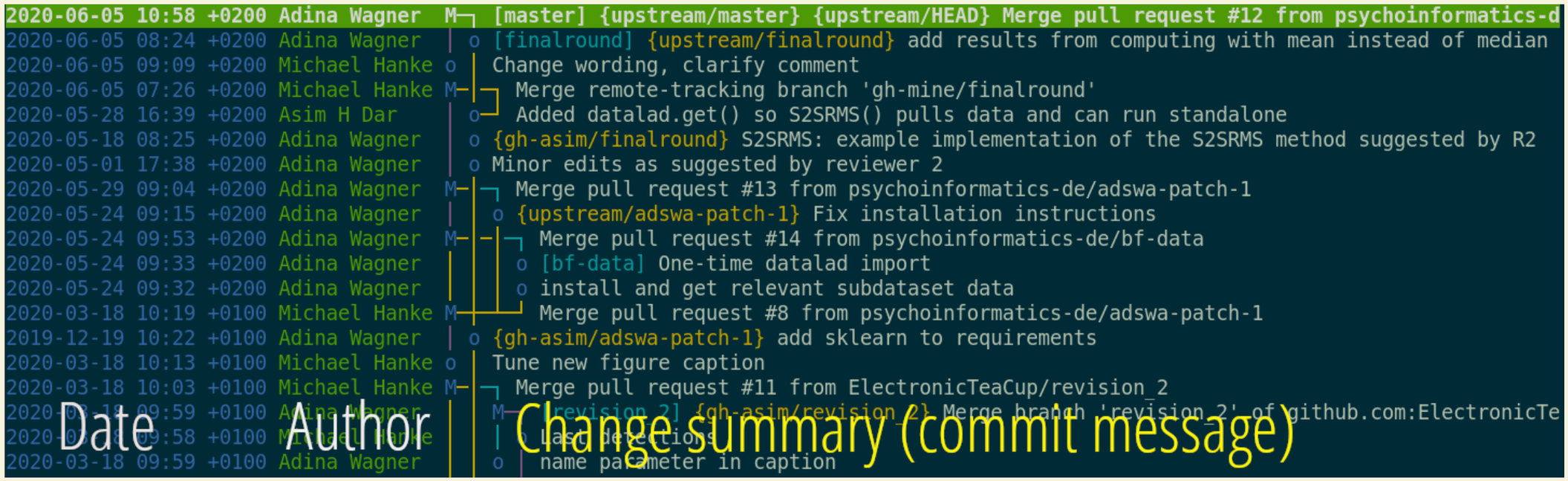
THIS MEANS: YOU CAN ALSO VERSION CONTROL DATA!

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)
```

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...
- ...

START TO RECORD PROVENANCE

- Have you ever saved a PDF to read later onto your computer, but forgot where you got it from?

START TO RECORD PROVENANCE

- Have you ever saved a PDF to read later onto your computer, but forgot where you got it from?
- Digital Provenance = *"The tools and processes used to create a digital file, the responsible entity, and when and where the process events occurred"*

START TO RECORD PROVENANCE

- Have you ever saved a PDF to read later onto your computer, but forgot where you got it from?
- Digital Provenance = *"The tools and processes used to create a digital file, the responsible entity, and when and where the process events occurred"*
- The history of a dataset already contains provenance, but there is more to record - for example: Where does a file come from? `datalad download-url` is helpful

SUMMARY - LOCAL VERSION CONTROL

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

Configurations (**-c yoda**, **-c text2git**) are useful (details soon).

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

Configurations (**-c yoda**, **-c text2git**) are useful (details soon).

A dataset has a *history* to track files and their modifications.

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

Configurations (**-c yoda**, **-c text2git**) are useful (details soon).

A dataset has a *history* to track files and their modifications.

Explore it with Git (**git log**) or external tools (e.g., **tig**).

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

Configurations (**-c yoda**, **-c text2git**) are useful (details soon).

A dataset has a *history* to track files and their modifications.

Explore it with Git (**git log**) or external tools (e.g., **tig**).

`datalad save` records the dataset or file state to the history.

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

Configurations (**-c yoda**, **-c text2git**) are useful (details soon).

A dataset has a *history* to track files and their modifications.

Explore it with Git (**git log**) or external tools (e.g., **tig**).

`datalad save` records the dataset or file state to the history.

Concise **commit messages** should summarize the change for future you and others.

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

Configurations (**-c yoda**, **-c text2git**) are useful (details soon).

A dataset has a *history* to track files and their modifications.

Explore it with Git (**git log**) or external tools (e.g., **tig**).

`datalad save` records the dataset or file state to the history.

Concise **commit messages** should summarize the change for future you and others.

`datalad download-url` obtains web content and records its origin.

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

Configurations (**-c yoda**, **-c text2git**) are useful (details soon).

A dataset has a *history* to track files and their modifications.

Explore it with Git (**git log**) or external tools (e.g., **tig**).

`datalad save` records the dataset or file state to the history.

Concise **commit messages** should summarize the change for future you and others.

`datalad download-url` obtains web content and records its origin.

It even takes care of saving the change.

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

Configurations (**-c yoda**, **-c text2git**) are useful (details soon).

A dataset has a *history* to track files and their modifications.

Explore it with Git (**git log**) or external tools (e.g., **tig**).

`datalad save` records the dataset or file state to the history.

Concise **commit messages** should summarize the change for future you and others.

`datalad download-url` obtains web content and records its origin.

It even takes care of saving the change.

`datalad status` reports the current state of the dataset.

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

Configurations (**-c yoda**, **-c text2git**) are useful (details soon).

A dataset has a *history* to track files and their modifications.

Explore it with Git (**git log**) or external tools (e.g., **tig**).

`datalad save` records the dataset or file state to the history.

Concise **commit messages** should summarize the change for future you and others.

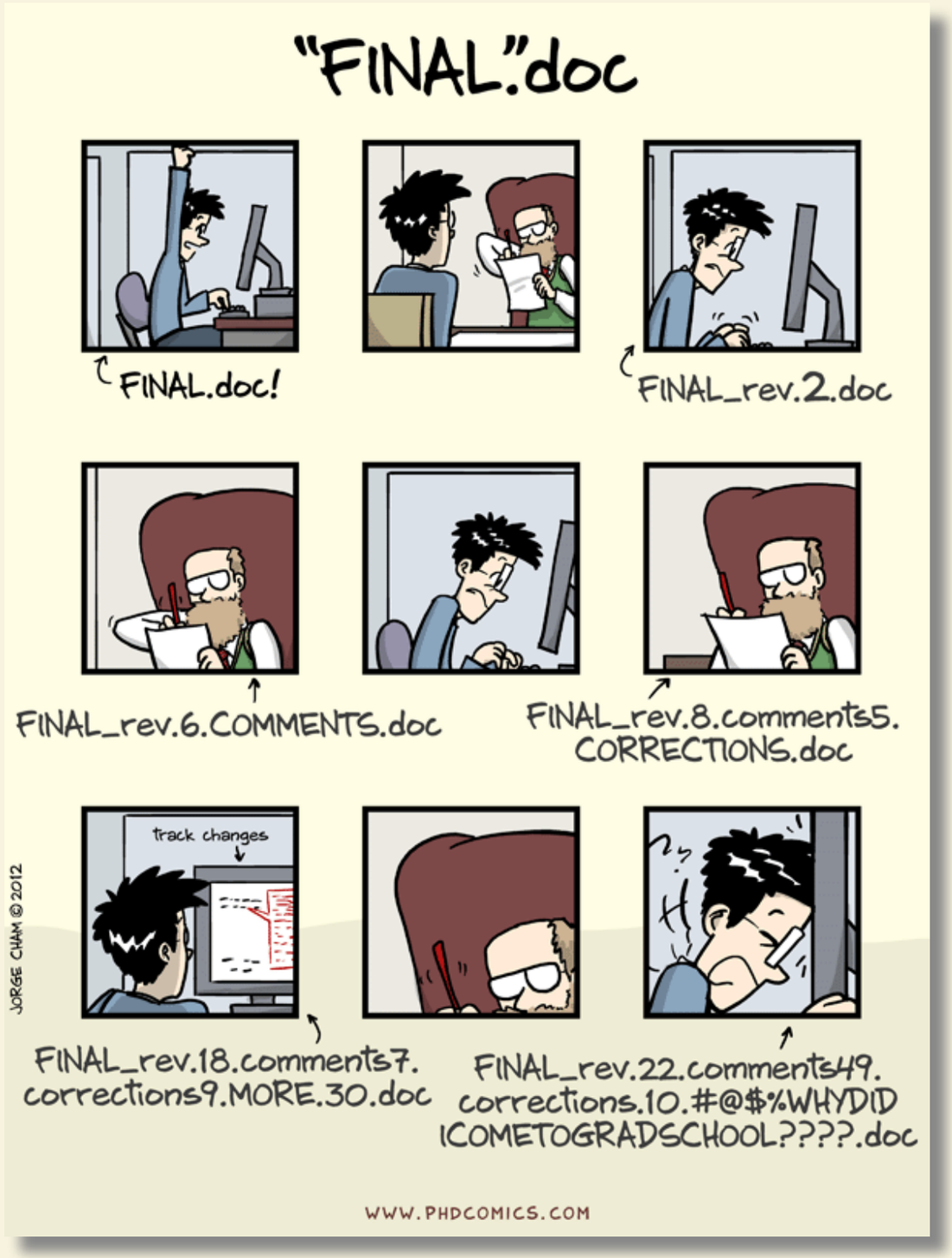
`datalad download-url` obtains web content and records its origin.

It even takes care of saving the change.

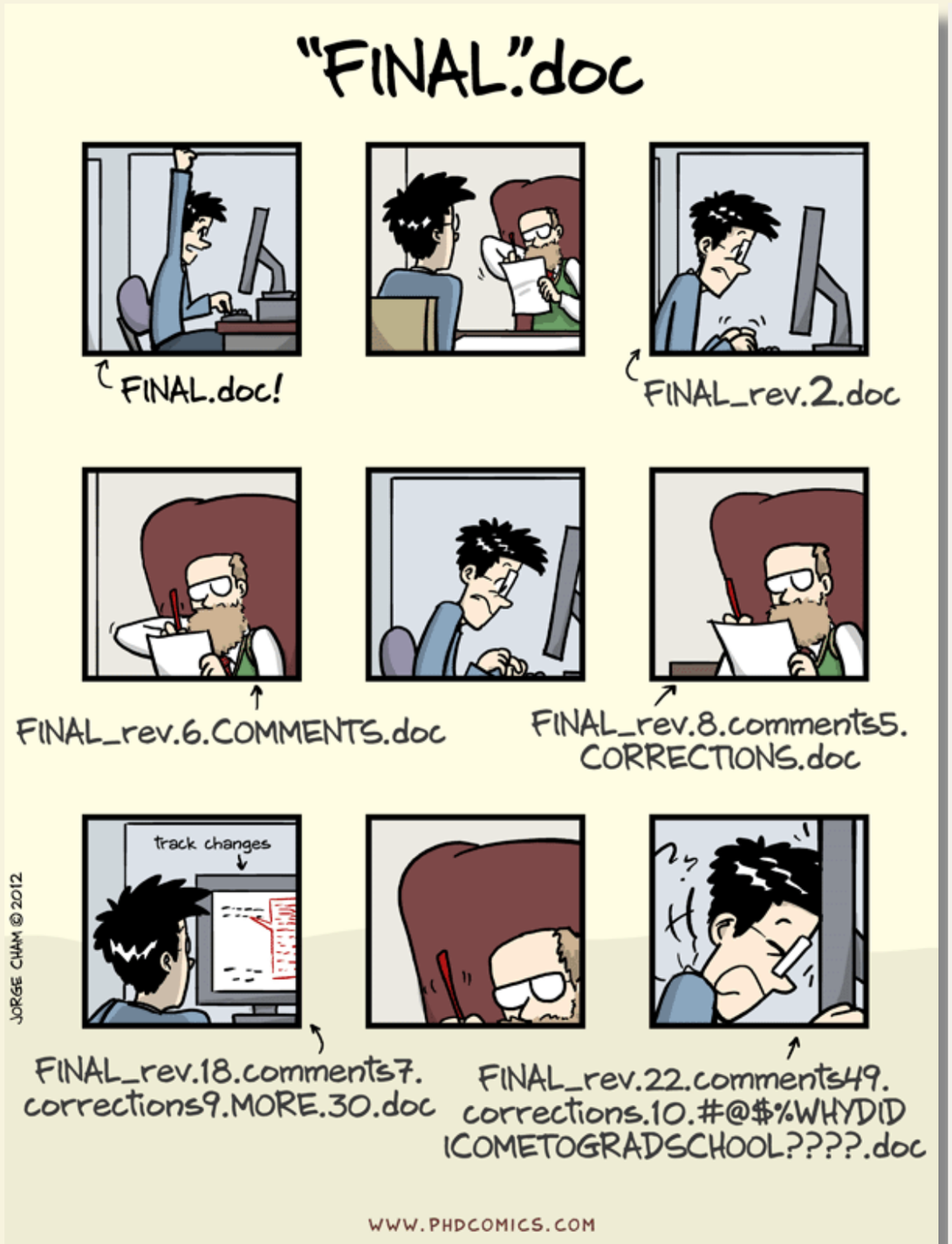
`datalad status` reports the current state of the dataset.

A clean dataset status (no modifications, not untracked files) is good practice.

FROM HERE

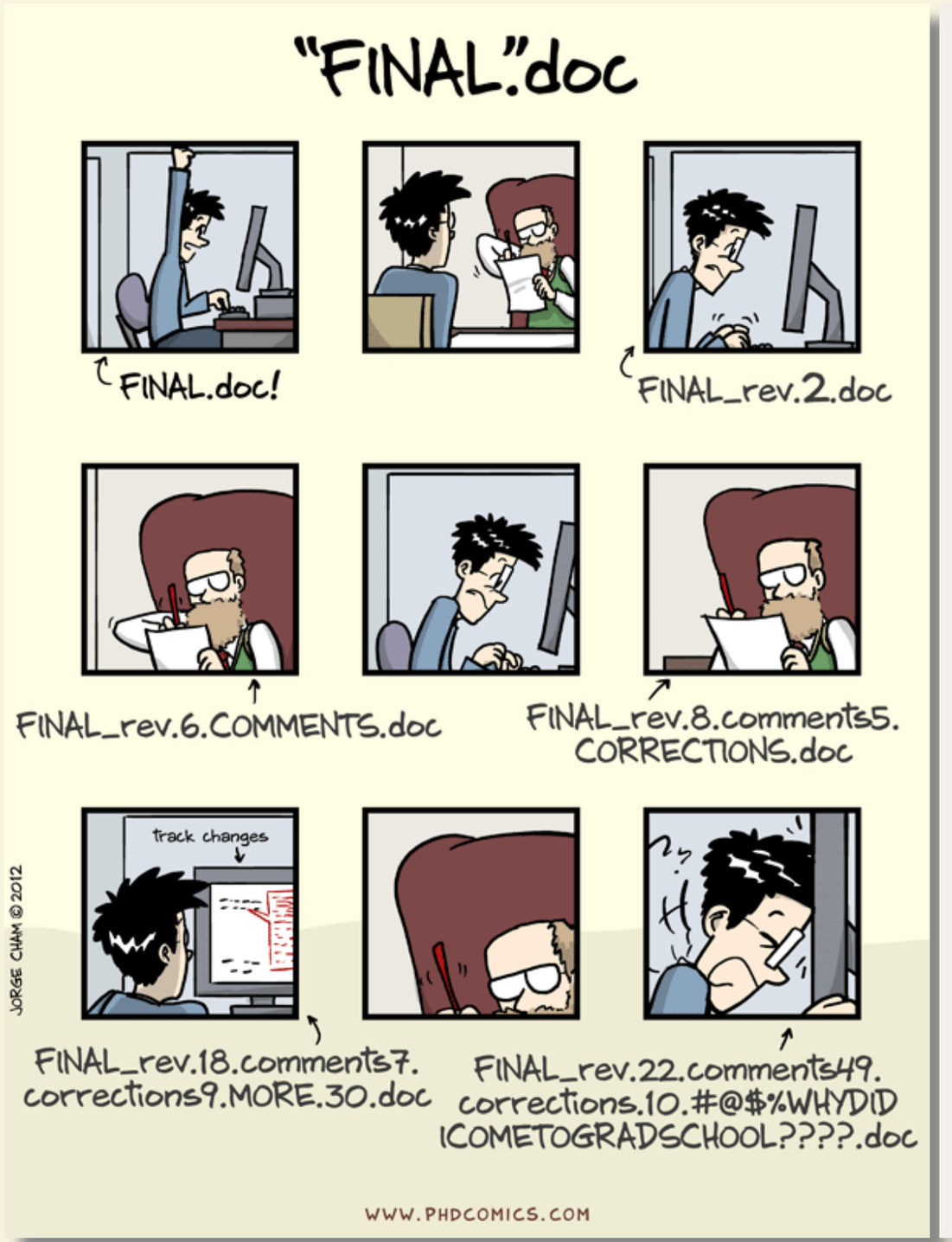

"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.
CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.
corrections9.MORE.30.doc
FINAL_rev.22.comments49.
corrections.10.#@$%WHYDID
ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

FROM HERE



TO THIS:

FROM HERE



TO THIS:

BUT: Version control is only one aspect of data management

QUESTIONS!

http://etc.ch/5xo7

CONSUMING DATASETS

- A dataset can be created from scratch/existing directories:

```
$ datalad create mydataset
[INFO
] Creating a new annex repo at /home/adina/mydataset
create(ok): /home/adina/mydataset (dataset)
```

- but datasets can also be installed from paths or from URLs:

```
$ datalad clone https://github.com/datalad-datasets/human-connectome-project-openac
install(ok): /tmp/HCP (dataset)
```

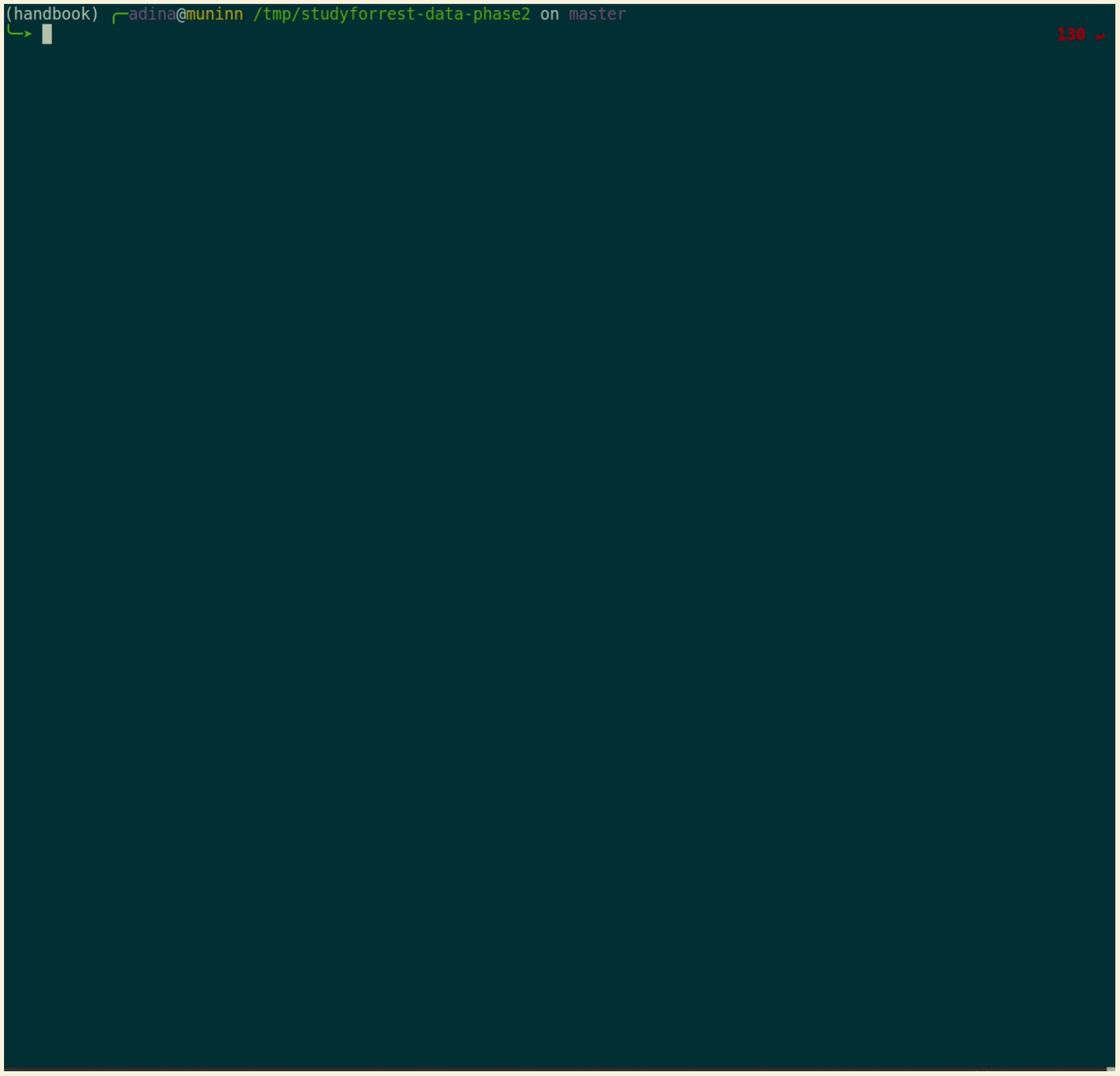
CONSUMING DATASETS

CONSUMING DATASETS

- Here's how a dataset looks after installation:

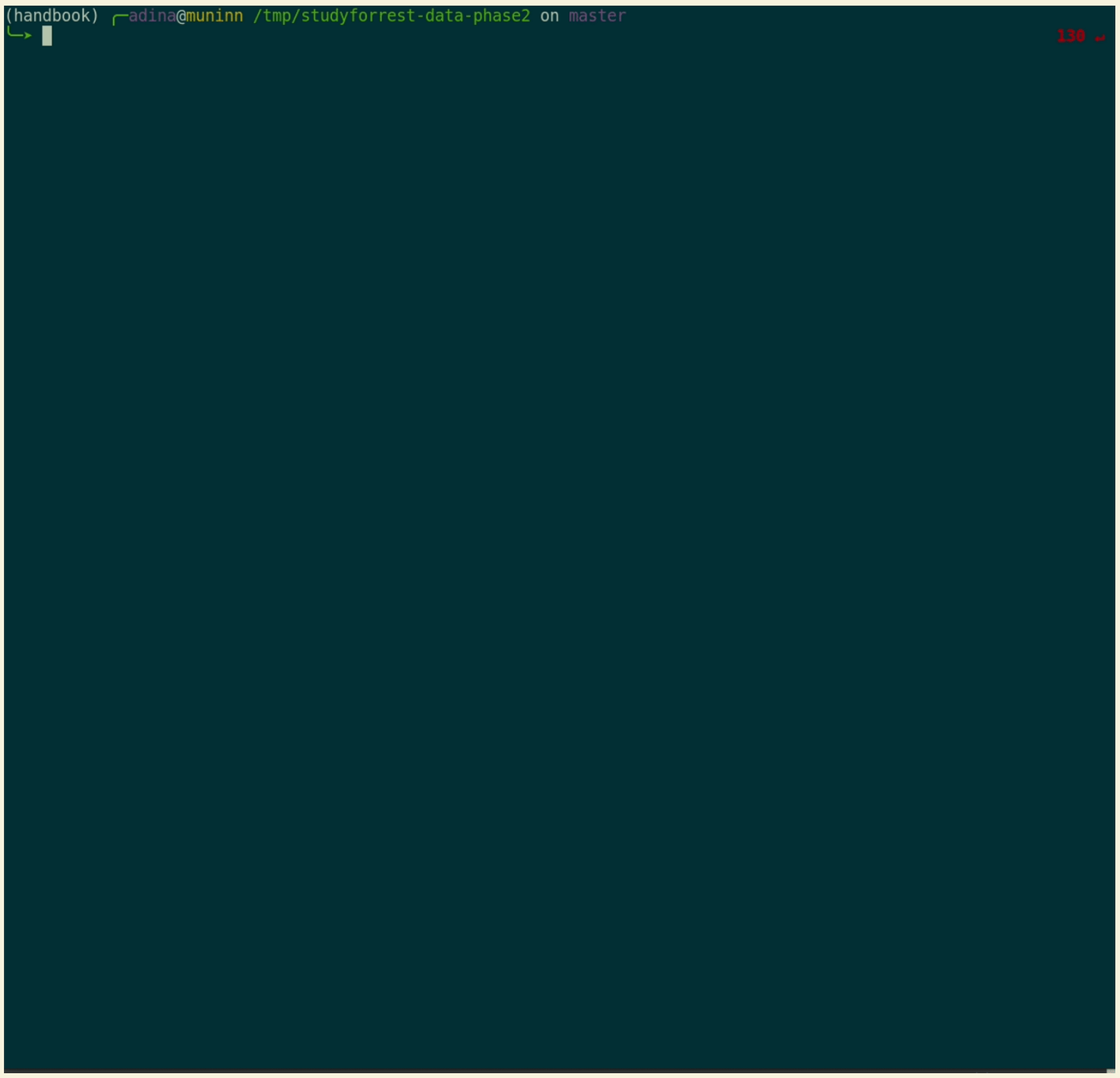
CONSUMING DATASETS

- Here's how a dataset looks after installation:



CONSUMING DATASETS

- Here's how a dataset looks after installation:



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

PLENTY OF DATA, BUT LITTLE DISK-USAGE

PLENTY OF DATA, BUT LITTLE DISK-USAGE

- Cloned datasets are lean. "Meta data" (file names, availability) are present, but **no file content**:

PLENTY OF DATA, BUT LITTLE DISK-USAGE

- Cloned datasets are lean. "Meta data" (file names, availability) are present, but **no file content**:

```
$ datalad clone git@github.com:psychoinformatics-de/studyforrest-data-phase2.git
  install(ok): /tmp/studyforrest-data-phase2 (dataset)
  $ cd studyforrest-data-phase2 && du -sh
  18M   .
```

PLENTY OF DATA, BUT LITTLE DISK-USAGE

- Cloned datasets are lean. "Meta data" (file names, availability) are present, but **no file content**:

```
$ datalad clone git@github.com:psychoinformatics-de/studyforrest-data-phase2.git
  install(ok): /tmp/studyforrest-data-phase2 (dataset)
  $ cd studyforrest-data-phase2 && du -sh
  18M   .
```

- file's contents can be retrieved on demand:

PLENTY OF DATA, BUT LITTLE DISK-USAGE

- Cloned datasets are lean. "Meta data" (file names, availability) are present, but **no file content**:

```
$ datalad clone git@github.com:psychoinformatics-de/studyforrest-data-phase2.git
  install(ok): /tmp/studyforrest-data-phase2 (dataset)
  $ cd studyforrest-data-phase2 && du -sh
  18M   .
```

- file's contents can be retrieved on demand:

```
$ 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.
```

PLENTY OF DATA, BUT LITTLE DISK-USAGE

- Cloned datasets are lean. "Meta data" (file names, availability) are present, but **no file content**:

```
$ datalad clone git@github.com:psychoinformatics-de/studyforrest-data-phase2.git
  install(ok): /tmp/studyforrest-data-phase2 (dataset)
  $ cd studyforrest-data-phase2 && du -sh
  18M   .
```

- file's contents can be retrieved on demand:

```
$ 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.
```

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

PLENTY OF DATA, BUT LITTLE DISK-USAGE

- Cloned datasets are lean. "Meta data" (file names, availability) are present, but **no file content**:

```
$ datalad clone git@github.com:psychoinformatics-de/studyforrest-data-phase2.git
  install(ok): /tmp/studyforrest-data-phase2 (dataset)
  $ cd studyforrest-data-phase2 && du -sh
  18M   .
```

- file's contents can be retrieved on demand:

```
$ 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.
```

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

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

PLENTY OF DATA, BUT LITTLE DISK-USAGE

PLENTY OF DATA, BUT LITTLE DISK-USAGE

Drop file content that is not needed:

PLENTY OF DATA, BUT LITTLE DISK-USAGE

Drop file content that is not needed:

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

PLENTY OF DATA, BUT LITTLE DISK-USAGE

Drop file content that is not needed:

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

When files are dropped, only "meta data" stays behind, and they can be re-obtained on demand. This allows disk-space aware computations:

PLENTY OF DATA, BUT LITTLE DISK-USAGE

Drop file content that is not needed:

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

When files are dropped, only "meta data" stays behind, and they can be re-obtained on demand. This allows disk-space aware computations:

Install your input data

PLENTY OF DATA, BUT LITTLE DISK-USAGE

Drop file content that is not needed:

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

When files are dropped, only "meta data" stays behind, and they can be re-obtained on demand. This allows disk-space aware computations:

Install your input data

➡ *get the data you need*

PLENTY OF DATA, BUT LITTLE DISK-USAGE

Drop file content that is not needed:

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

When files are dropped, only "meta data" stays behind, and they can be re-obtained on demand. This allows disk-space aware computations:

Install your input data

➡ *get the data you need*

➡ *compute your results*

PLENTY OF DATA, BUT LITTLE DISK-USAGE

Drop file content that is not needed:

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

When files are dropped, only "meta data" stays behind, and they can be re-obtained on demand. This allows disk-space aware computations:

Install your input data

➡ *get the data you need*

➡ *compute your results*

➡ *drop input data (and potentially all automatically re-computable results)*

PLENTY OF DATA, BUT LITTLE DISK-USAGE

Drop file content that is not needed:

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

When files are dropped, only "meta data" stays behind, and they can be re-obtained on demand. This allows disk-space aware computations:

Install your input data

➡ *get the data you need*

➡ *compute your results*

➡ *drop input data (and potentially all automatically re-computable results)*

```
dl.get('input/sub-01')

[really complex analysis]

dl.drop('input/sub-01')
```

GIT VERSUS GIT-ANNEX

Data in datasets is either stored in Git or git-annex

By default, everything is *annexed*, i.e., stored in a dataset annex by git-annex

Git	**git-annex**
handles **small** files well (text, code)	handles **all** types and sizes of files well
file contents are in the Git history and will be **shared** upon git/datalad push	file contents are in the annex. Not necessarily shared
Shared with every dataset clone	**Can be kept private** on a per-file level when sharing the dataset
Useful: Small, non-binary, frequently modified, need-to-be-accessible (DUA, README) files	Useful: Large files, private files

GIT VERSUS GIT-ANNEX

Data in datasets is either stored in Git or git-annex

By default, everything is *annexed*, i.e., stored in a dataset annex by git-annex

Git	**git-annex**
handles **small** files well (text, code)	handles **all** types and sizes of files well
file contents are in the Git history and will be **shared** upon git/datalad push	file contents are in the annex. Not necessarily shared
Shared with every dataset clone	**Can be kept private** on a per-file level when sharing the dataset
Useful: Small, non-binary, frequently modified, need-to-be-accessible (DUA, README) files	Useful: Large files, private files

- With annexed data, only content identity (hash) and location information is put into Git, rather than file content. The annex, and transport to and from it is managed with **git-annex**

GIT VERSUS GIT-ANNEX

Git

- dataset history (commit messages, run records)
- All files + content committed into Git (useful with code, text, ...)
- File identity information of all annexed files (file name, identity hash, storage locations where to retrieve it from)

git-annex

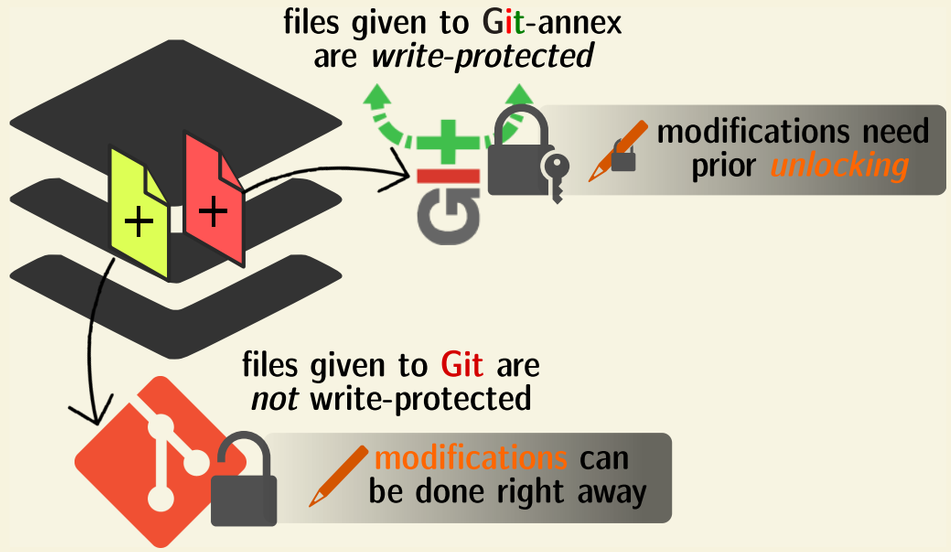
- contents of annexed files
- organized in the "annex" or "object tree" of the dataset

GIT VERSUS GIT-ANNEX

Useful background information for demo later. Read this handbook chapter for details

Git and Git-annex handle files differently: annexed files are stored in an annex. File content is hashed & only content-identity is committed to Git.

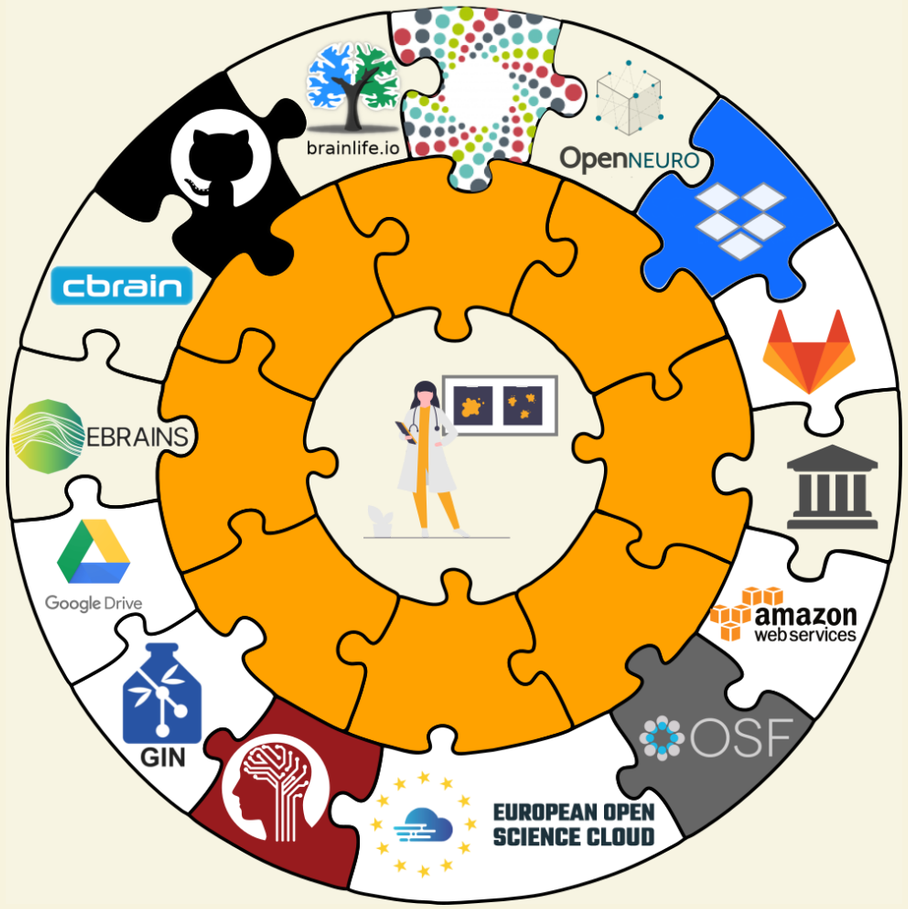
- Files stored in Git are modifiable, files stored in Git-annex are content-locked



- Annexed contents are not available right after cloning, only content identity and availability information (as they are stored in Git). Everything that is annexed needs to be retrieved with `datalad get` from whereever it is stored.

GIT VERSUS GIT-ANNEX

When sharing datasets with someone without access to the same computational infrastructure, annexed data is not necessarily stored together with the rest of the dataset (more in the **session on publishing**).



Transport logistics exist to interface with all major storage providers. If the one you use isn't supported, let us know!

GIT VERSUS GIT-ANNEX

Users can decide which files are annexed:

- **Pre-made run-procedures**, provided by DataLad (e.g., `text2git`, `yoda`) or created and shared by users (Tutorial)
- Self-made configurations in `.gitattributes` (e.g., based on file type, file/path name, size, ...; rules and examples)
- Per-command basis (e.g., via `datalad save --to-git`)

TRANSPORT LOGISTICS

TRANSPORT LOGISTICS

- Disk-space aware workflows: Cloned datasets are lean (only Git):

TRANSPORT LOGISTICS

- Disk-space aware workflows: Cloned datasets are lean (only Git):

```
$ datalad clone git@github.com:datalad-datasets/machinelearning-books.git
  install(ok): /tmp/machinelearning-books (dataset)
  $ cd machinelearning-books && du -sh
  348K  .
```

TRANSPORT LOGISTICS

- Disk-space aware workflows: Cloned datasets are lean (only Git):

```
$ datalad clone git@github.com:datalad-datasets/machinelearning-books.git
  install(ok): /tmp/machinelearning-books (dataset)
  $ cd machinelearning-books && du -sh
  348K  .
```

```
$ ls
  A.Shashua-Introduction_to_Machine_Learning.pdf
  B.Efron_T.Hastie-Computer_Age_Statistical_Inference.pdf
  C.E.Rasmussen_C.K.I.Williams-Gaussian_Processes_for_Machine_Learning.pdf
  D.Barber-Bayesian_Reasoning_and_Machine_Learning.pdf
  [...]
```

TRANSPORT LOGISTICS

- Disk-space aware workflows: Cloned datasets are lean (only Git):

```
$ datalad clone git@github.com:datalad-datasets/machinelearning-books.git
  install(ok): /tmp/machinelearning-books (dataset)
  $ cd machinelearning-books && du -sh
  348K  .
```

```
$ ls
  A.Shashua-Introduction_to_Machine_Learning.pdf
  B.Efron_T.Hastie-Computer_Age_Statistical_Inference.pdf
  C.E.Rasmussen_C.K.I.Williams-Gaussian_Processes_for_Machine_Learning.pdf
  D.Barber-Bayesian_Reasoning_and_Machine_Learning.pdf
  [...]
```

- annexed file's contents can be retrieved & dropped on demand:

TRANSPORT LOGISTICS

- Disk-space aware workflows: Cloned datasets are lean (only Git):

```
$ datalad clone git@github.com:datalad-datasets/machinelearning-books.git
  install(ok): /tmp/machinelearning-books (dataset)
  $ cd machinelearning-books && du -sh
  348K  .
```

```
$ ls
  A.Shashua-Introduction_to_Machine_Learning.pdf
  B.Efron_T.Hastie-Computer_Age_Statistical_Inference.pdf
  C.E.Rasmussen_C.K.I.Williams-Gaussian_Processes_for_Machine_Learning.pdf
  D.Barber-Bayesian_Reasoning_and_Machine_Learning.pdf
  [...]
```

- annexed file's contents can be retrieved & dropped on demand:

```
$ datalad get A.Shashua-Introduction_to_Machine_Learning.pdf
  get(ok): /tmp/machinelearning-books/A.Shashua-Introduction_to_Machine_Learning.pdf (file) [from web...]
```

TRANSPORT LOGISTICS

- Disk-space aware workflows: Cloned datasets are lean (only Git):

```
$ datalad clone git@github.com:datalad-datasets/machinelearning-books.git
  install(ok): /tmp/machinelearning-books (dataset)
  $ cd machinelearning-books && du -sh
  348K  .
```

```
$ ls
  A.Shashua-Introduction_to_Machine_Learning.pdf
  B.Efron_T.Hastie-Computer_Age_Statistical_Inference.pdf
  C.E.Rasmussen_C.K.I.Williams-Gaussian_Processes_for_Machine_Learning.pdf
  D.Barber-Bayesian_Reasoning_and_Machine_Learning.pdf
  [...]
```

- annexed file's contents can be retrieved & dropped on demand:

```
$ datalad get A.Shashua-Introduction_to_Machine_Learning.pdf
  get(ok): /tmp/machinelearning-books/A.Shashua-Introduction_to_Machine_Learning.pdf (file) [from web...]
```

```
$ datalad drop A.Shashua-Introduction_to_Machine_Learning.pdf
  drop(ok): /tmp/machinelearning-books/A.Shashua-Introduction_to_Machine_Learning.pdf (file) [checking htt
```

GIT-ANNEX PROTECTS YOUR FILES

- If git-annex does not know any other storage location for a file it will warn you and refuse to drop content (can be configured)

GIT-ANNEX PROTECTS YOUR FILES

- If git-annex does not know any other storage location for a file it will warn you and refuse to drop content (can be configured)
- Here is a file with a registered remote location (the web)

```
$ datalad drop .easteregg
drop(ok): /demo/myanalysis/.easteregg (file) [checking https://imgs.xkcd.com/comics/fuck_grapefruit.png...
```

GIT-ANNEX PROTECTS YOUR FILES

- If git-annex does not know any other storage location for a file it will warn you and refuse to drop content (can be configured)
- Here is a file with a registered remote location (the web)

```
$ datalad drop .easteregg
drop(ok): /demo/myanalysis/.easteregg (file) [checking https://imgs.xkcd.com/comics/fuck_grapefruit.png...
```

- Here is a file without a registered remote location (the web)

```
$ datalad drop compiling.png
[WARNING] Running drop resulted in stderr output: git-annex: drop: 1 failed
[ERROR  ] unsafe; Could only verify the existence of 0 out of 1 necessary copies; Rather than dropping thi
drop(error): /demo/myanalysis/compiling.png (file) [unsafe; Could only verify the existence of 0 out of 1 
```

GIT-ANNEX PROTECTS YOUR FILES

- If git-annex does not know any other storage location for a file it will warn you and refuse to drop content (can be configured)
- Here is a file with a registered remote location (the web)

```
$ datalad drop .easteregg
drop(ok): /demo/myanalysis/.easteregg (file) [checking https://imgs.xkcd.com/comics/fuck_grapefruit.png...
```

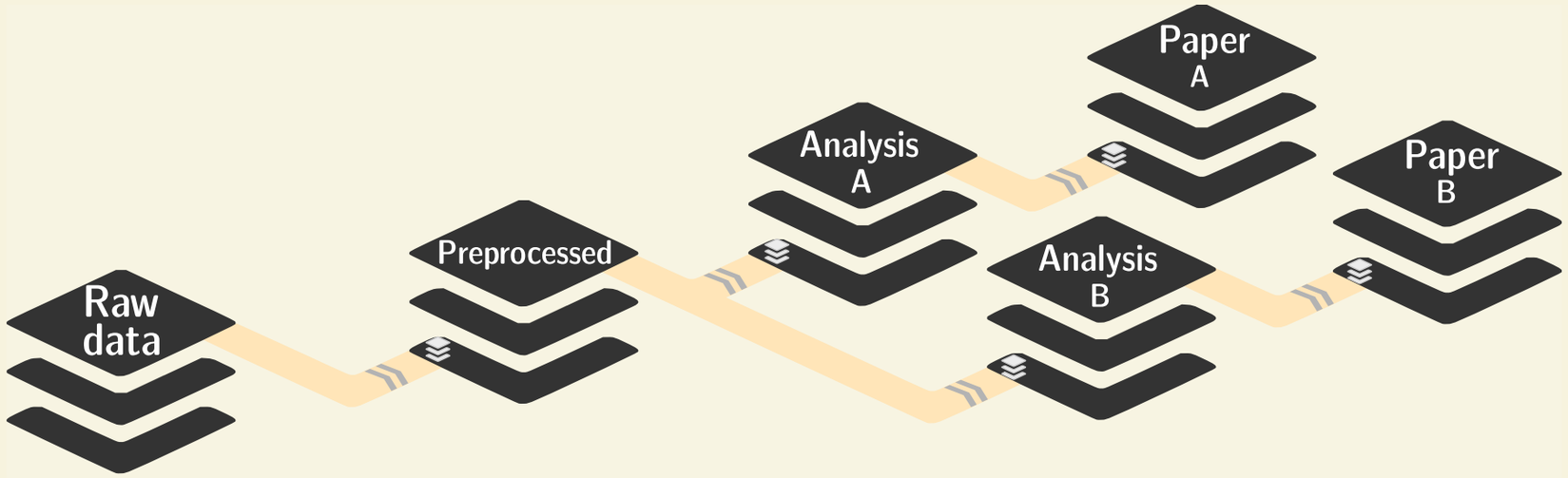
- Here is a file without a registered remote location (the web)

```
$ datalad drop compiling.png
[WARNING] Running drop resulted in stderr output: git-annex: drop: 1 failed
[ERROR  ] unsafe; Could only verify the existence of 0 out of 1 necessary copies; Rather than dropping thi
drop(error): /demo/myanalysis/compiling.png (file) [unsafe; Could only verify the existence of 0 out of 1 
```

- If a different location for file content is known, `datalad get` can retrieve file content after dropping

DATASET NESTING

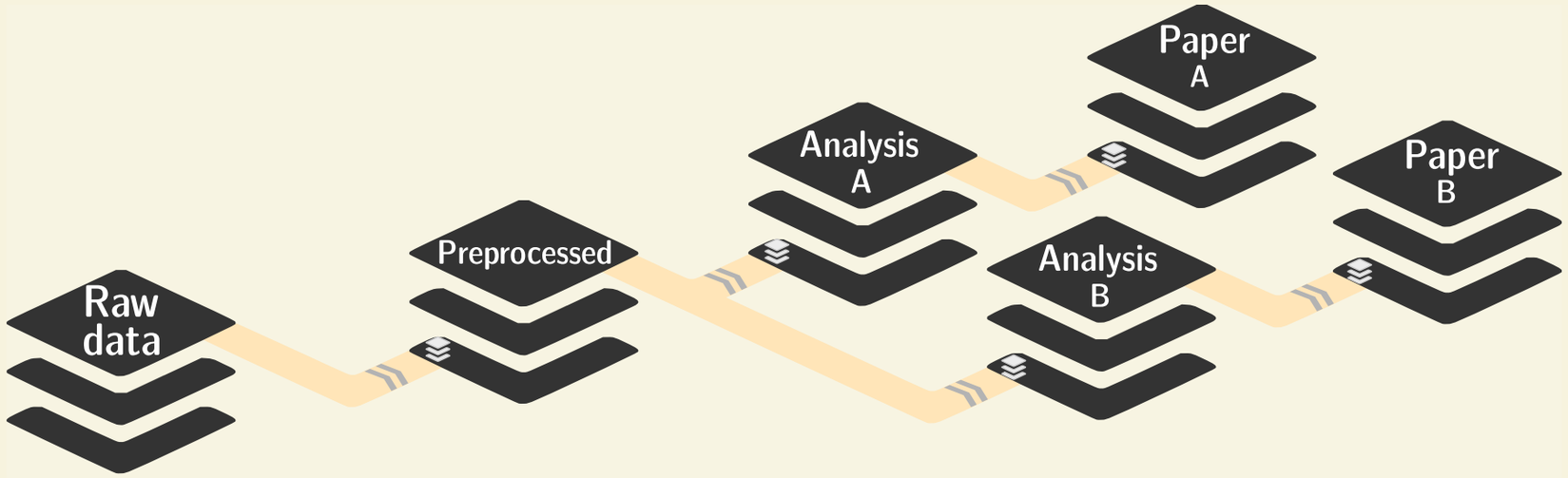
- Seamless nesting mechanisms:



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

DATASET NESTING

- Seamless nesting mechanisms:



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

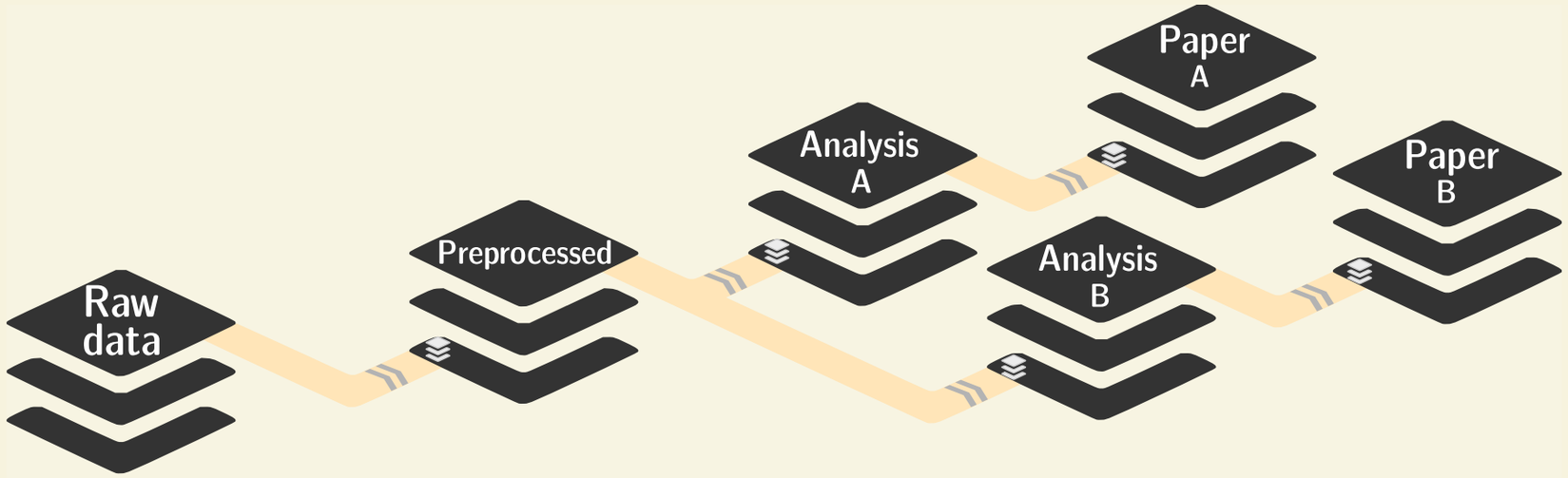
- Overcomes scaling issues with large amounts of files

```
adina@bulk1 in /ds/hcp/super on git:master❱ datalad status --annex -r
  15530572 annex'd files (77.9 TB recorded total size)
  nothing to save, working tree clean
```

(github.com/datalad-datasets/human-connectome-project-openaccess)

DATASET NESTING

- Seamless nesting mechanisms:



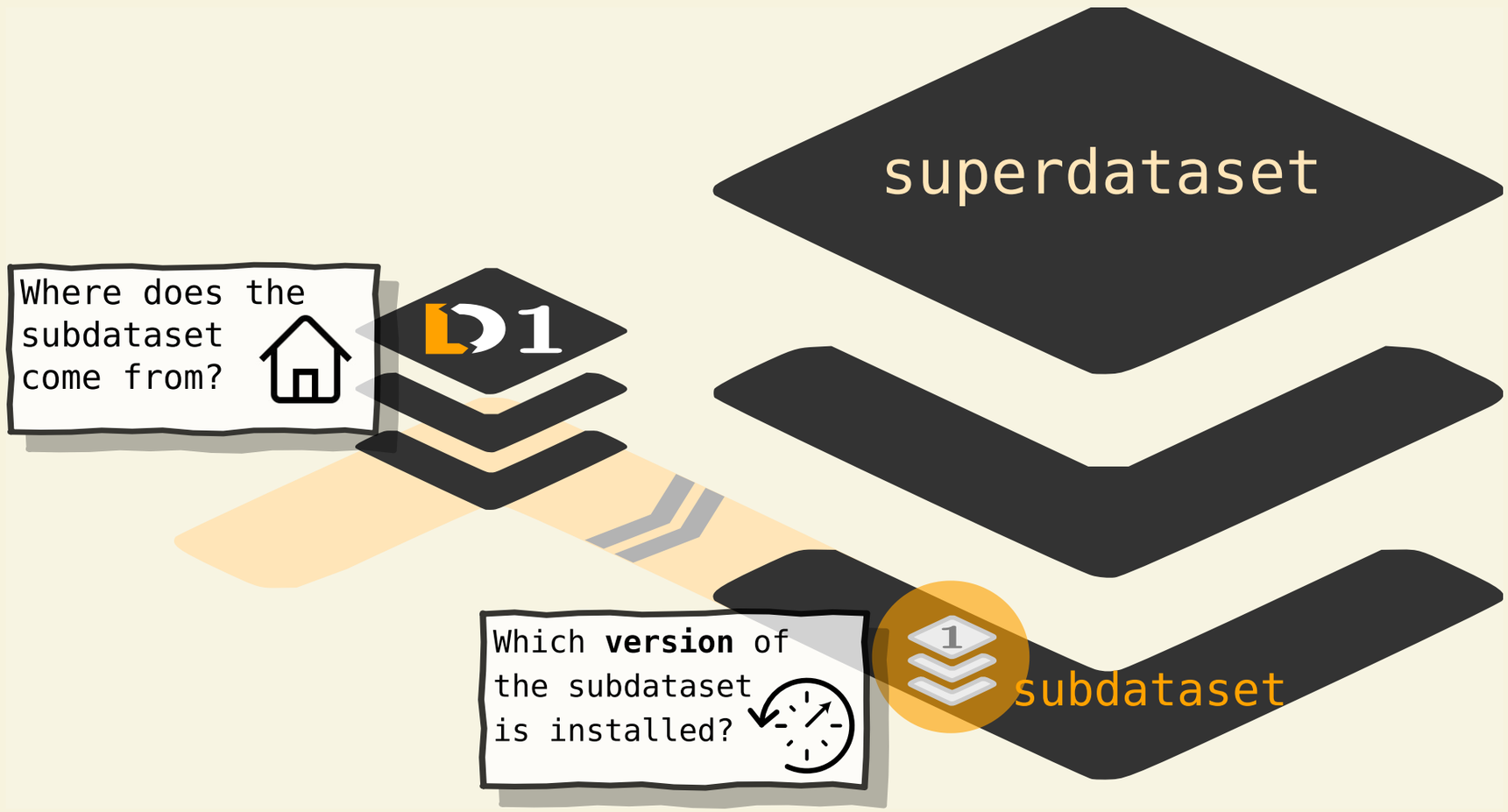
- Overcomes scaling issues with large amounts of files

```
adina@bulk1 in /ds/hcp/super on git:master❱ datalad status --annex -r
  15530572 annex'd files (77.9 TB recorded total size)
  nothing to save, working tree clean
```

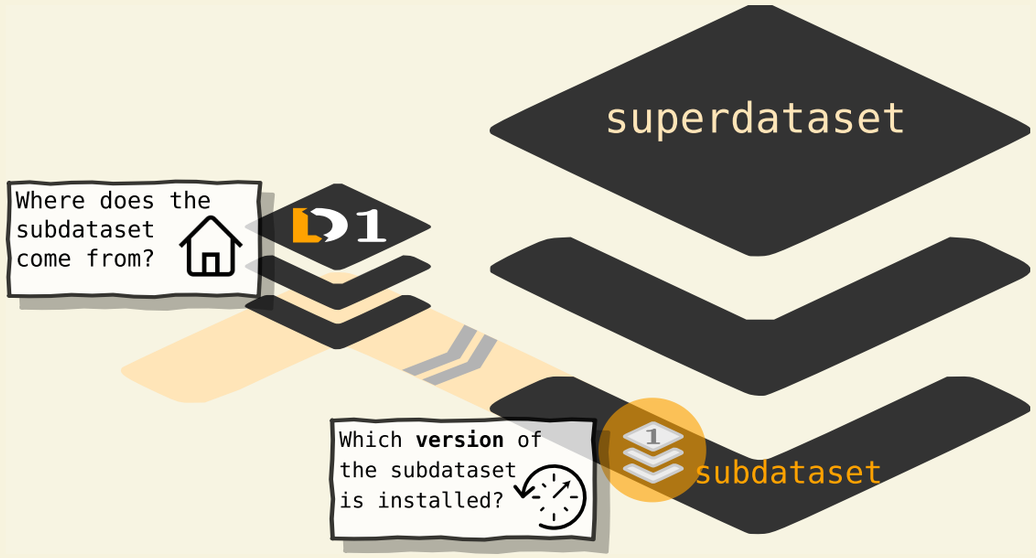
(github.com/datalad-datasets/human-connectome-project-openaccess)

- Modularizes research components for transparency, reuse, and access management (more on this in the **section on reproducible science**)

DATASET NESTING



DATALAD: DATASET LINKAGE




```
$ datalad clone --dataset . http://example.com/ds inputs/rawdata
```

```
$ git diff HEAD~1
diff --git a/.gitmodules b/.gitmodules
new file mode 100644
index 0000000..c3370ba
--- /dev/null
+++ b/.gitmodules
@@ -0,0 +1,3 @@
+[submodule "inputs/rawdata"]
+       path = inputs/rawdata
+       url = http://example.com/importantds
diff --git a/inputs/rawdata b/inputs/rawdata
new file mode 160000
index 0000000..fabf852
--- /dev/null
+++ b/inputs/rawdata
@@ -0,0 +1 @@
+Subproject commit fabf8521130a13986bd6493cb33a70e580ce8572
```

SUMMARY - DATASET CONSUMPTION & NESTING

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad clone` installs a dataset.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad clone` installs a dataset.

It can be installed “on its own”: Specify the source (url, path, ...) of the dataset, and an optional **path** for it to be installed to.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad clone` installs a dataset.

It can be installed “on its own”: Specify the source (url, path, ...) of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad clone` installs a dataset.

It can be installed “on its own”: Specify the source (url, path, ...) of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

The **--dataset/-d** option needs a path to the root of the superdataset.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad clone` installs a dataset.

It can be installed “on its own”: Specify the source (url, path, ...) of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

The **--dataset/-d** option needs a path to the root of the superdataset.

Only small files and metadata about file availability are present locally after an install.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad clone` installs a dataset.

It can be installed "on its own": Specify the source (url, path, ...) of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

The **--dataset/-d** option needs a path to the root of the superdataset.

Only small files and metadata about file availability are present locally after an install.

To retrieve actual file content of annexed files, `datalad get` downloads file content on demand.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad clone` installs a dataset.

It can be installed "on its own": Specify the source (url, path, ...) of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

The **--dataset/-d** option needs a path to the root of the superdataset.

Only small files and metadata about file availability are present locally after an install.

To retrieve actual file content of annexed files, `datalad get` downloads file content on demand.

Datasets preserve their history.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad clone` installs a dataset.

It can be installed "on its own": Specify the source (url, path, ...) of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

The **--dataset/-d** option needs a path to the root of the superdataset.

Only small files and metadata about file availability are present locally after an install.

To retrieve actual file content of annexed files, `datalad get` downloads file content on demand.

Datasets preserve their history.

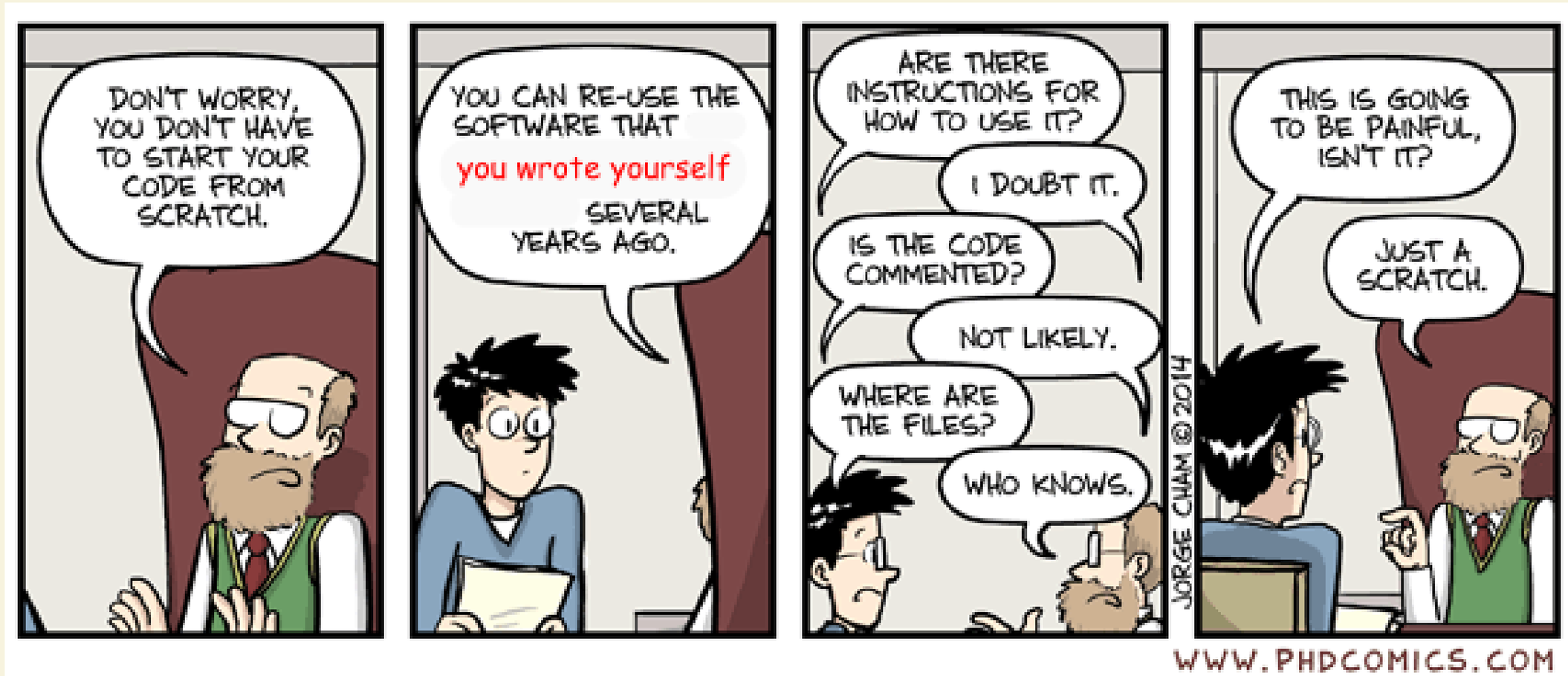
The superdataset records only the *version state* of the subdataset.

QUESTIONS!

http://etc.ch/5xo7

REPRODUCIBLE DATA ANALYSIS

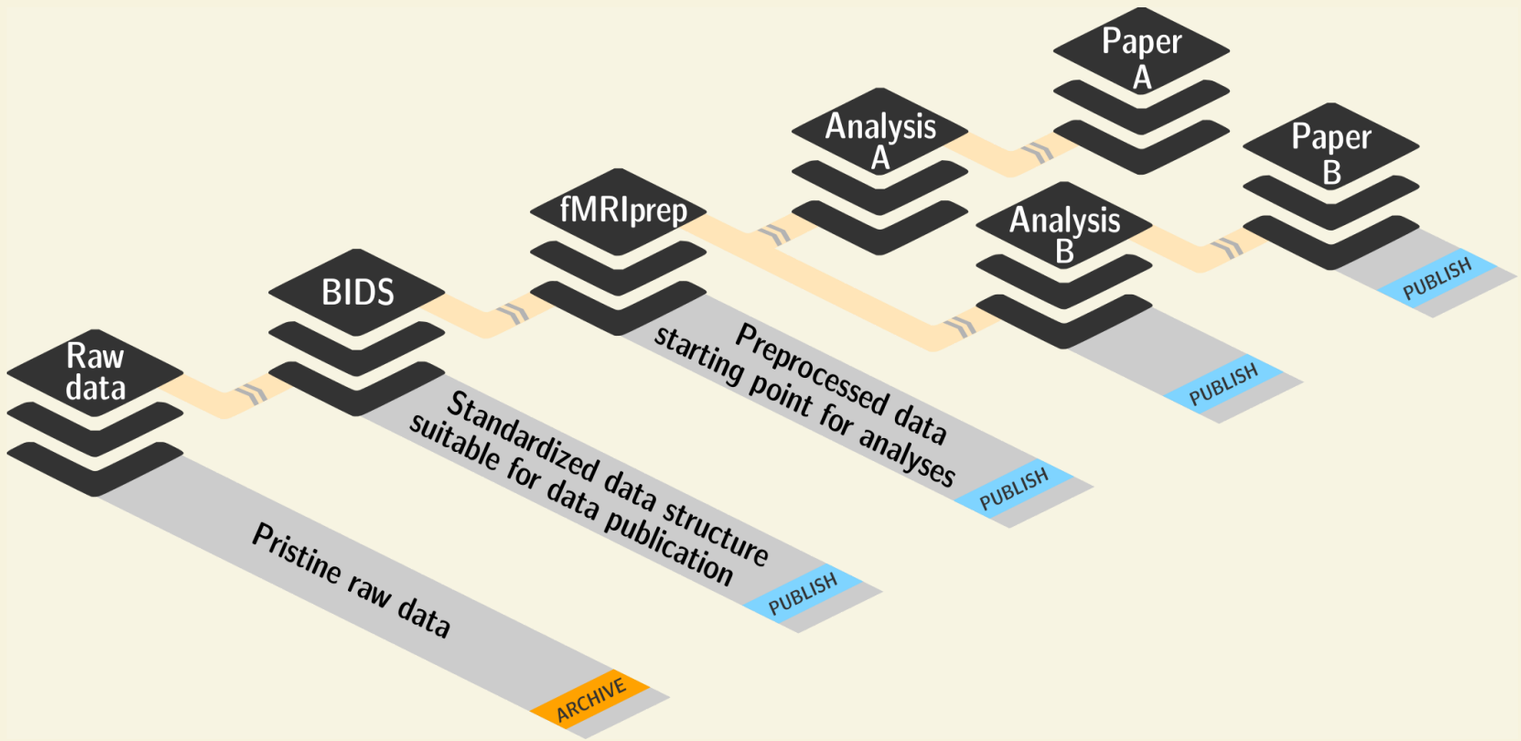
Your past self is the worst collaborator:

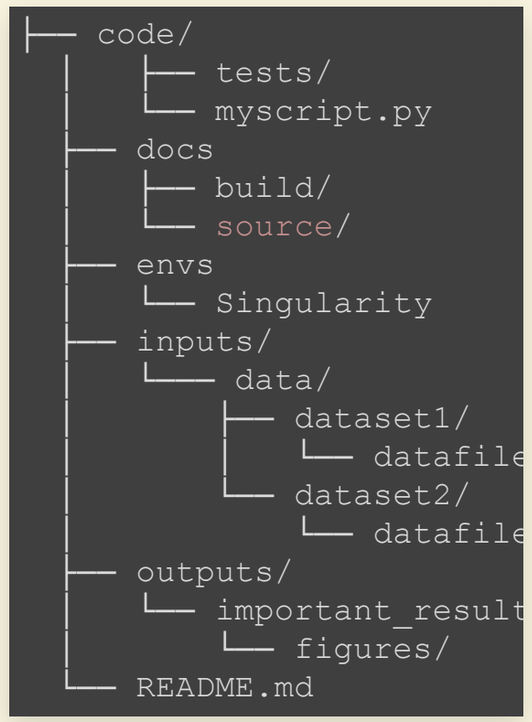


BASIC ORGANIZATIONAL PRINCIPLES FOR DATASETS

Keep everything clean and modular

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



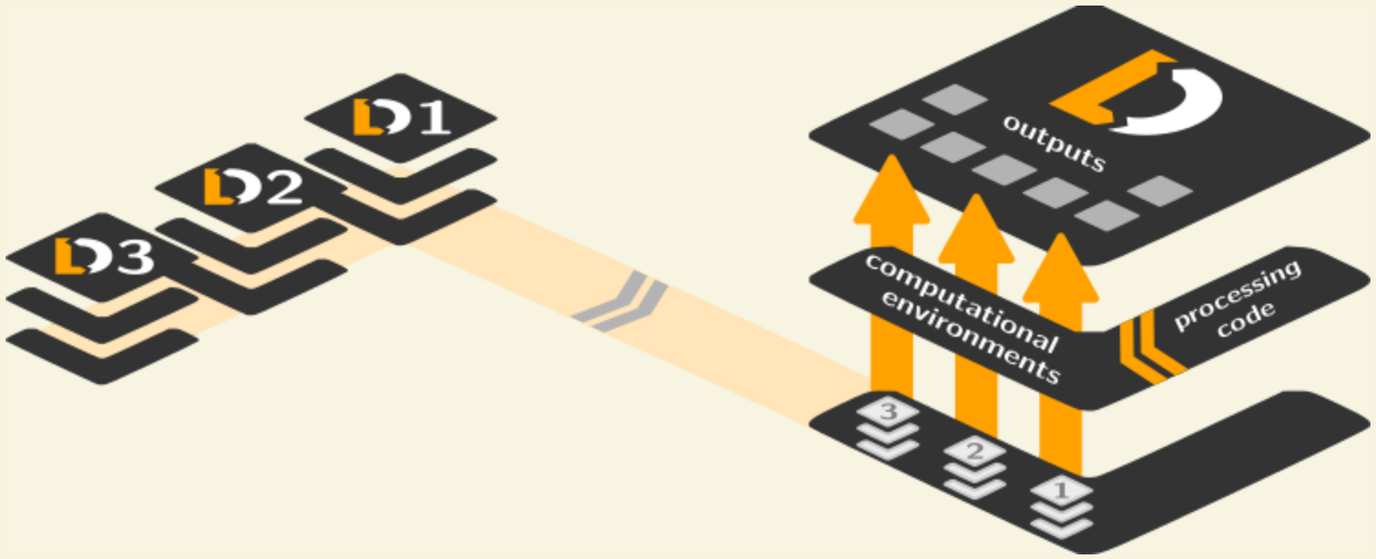


- do not touch/modify raw data: save any results/computations *outside* of input datasets
- Keep a superdataset self-contained: Scripts reference subdatasets or files with *relative paths*

BASIC ORGANIZATIONAL PRINCIPLES FOR DATASETS

Record where you got it from, where it is now, and what you do to it

- Link datasets (as subdatasets), record data origin
- Collect and store provenance of all contents of a dataset that you create

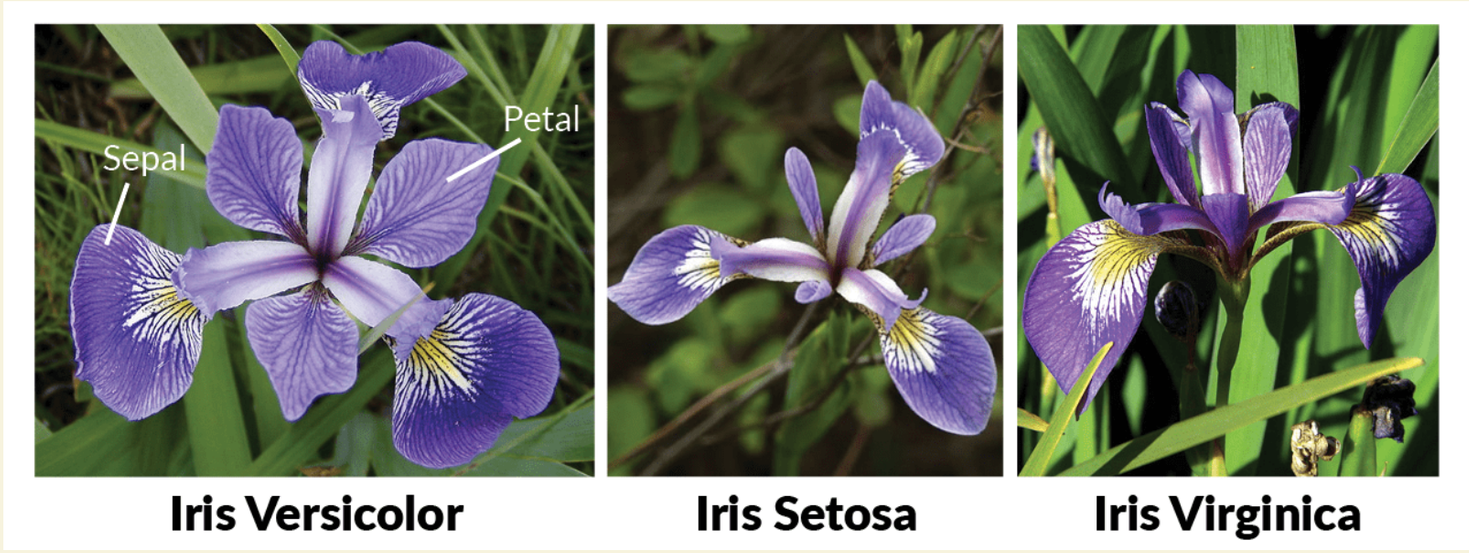


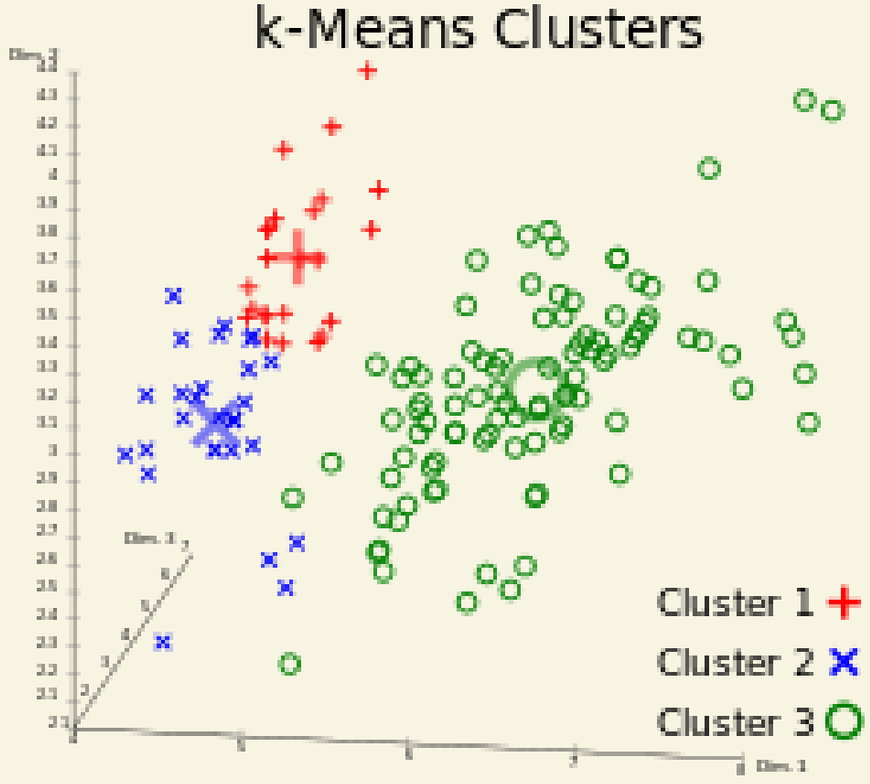
Document everything:

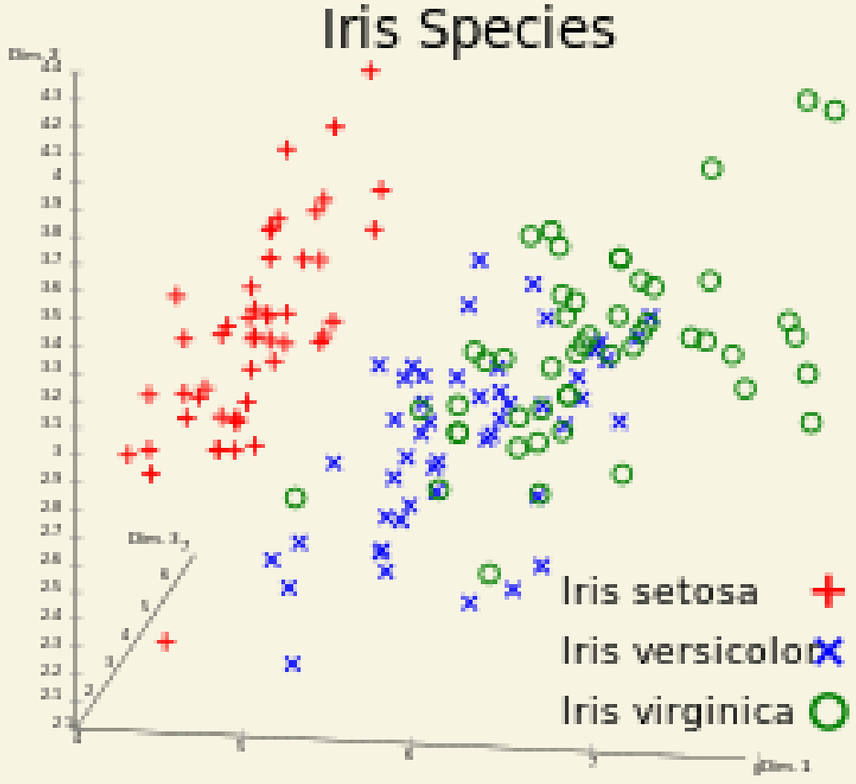
- Which script produced which output? From which data? In which software environment? …

Find out more about organizational principles in the YODA principles!

A CLASSIFICATION ANALYSIS ON THE IRIS FLOWER DATASET





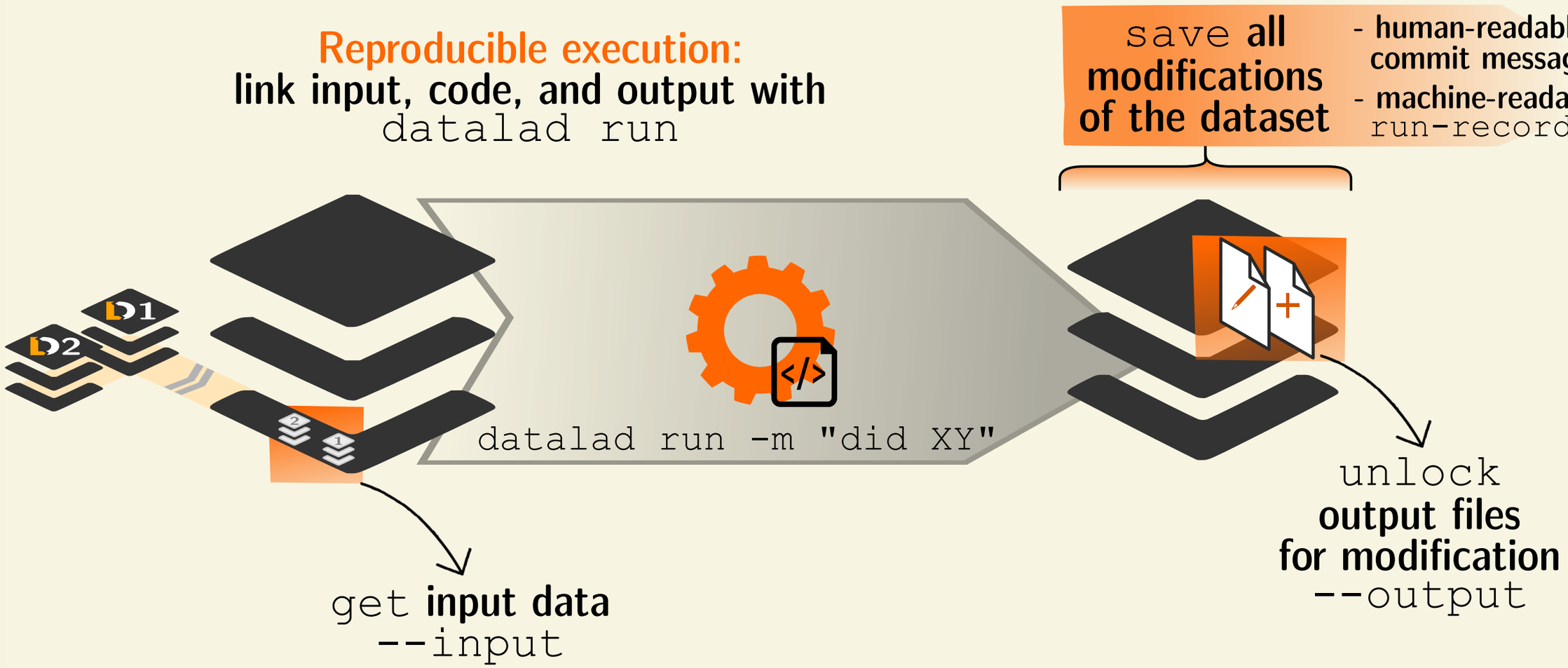


REPRODUCIBLE EXECUTION & PROVENANCE CAPTURE

datalad run

REPRODUCIBLE EXECUTION & PROVENANCE CAPTURE

datalad 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

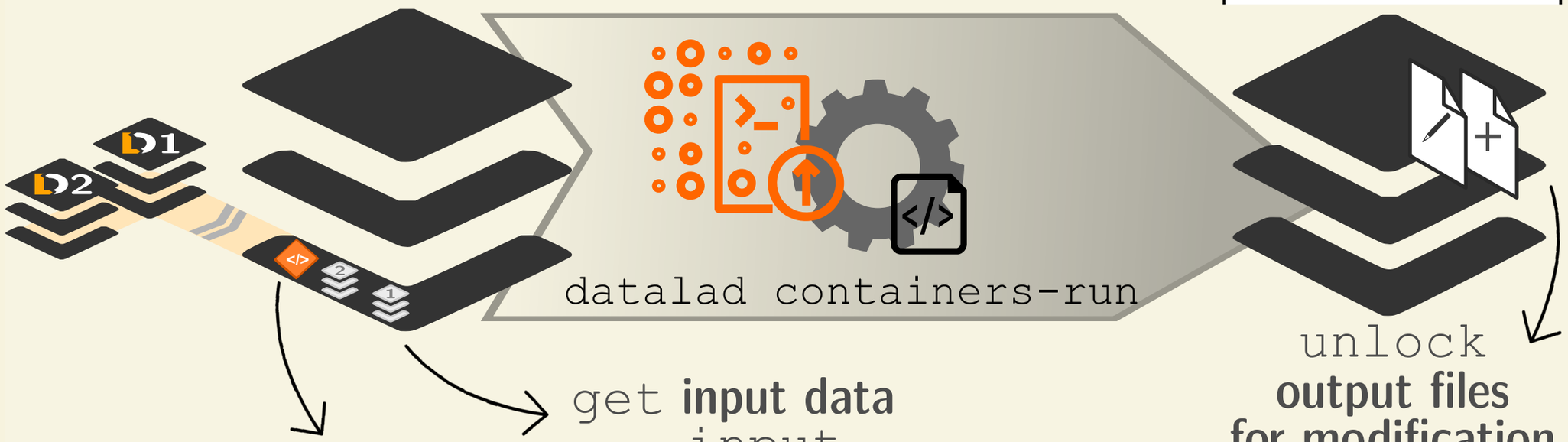
COMPUTATIONAL REPRODUCIBILITY

- 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

SUMMARY - REPRODUCIBLE EXECUTION

SUMMARY - REPRODUCIBLE EXECUTION

`datalad run` records a command and its impact on the dataset.

SUMMARY - REPRODUCIBLE EXECUTION

`datalad run` records a command and its impact on the dataset.

All dataset modifications are saved - use it in a clean dataset.

SUMMARY - REPRODUCIBLE EXECUTION

`datalad run` records a command and its impact on the dataset.

All dataset modifications are saved - use it in a clean dataset.

Data/directories specified as `--input` are retrieved prior to command execution.

SUMMARY - REPRODUCIBLE EXECUTION

`datalad run` records a command and its impact on the dataset.

All dataset modifications are saved - use it in a clean dataset.

Data/directories specified as `--input` are retrieved prior to command execution.

Use one flag per input.

SUMMARY - REPRODUCIBLE EXECUTION

`datalad run` records a command and its impact on the dataset.

All dataset modifications are saved - use it in a clean dataset.

Data/directories specified as `--input` are retrieved prior to command execution.

Use one flag per input.

Data/directories specified as `--output` will be unlocked for modifications prior to a rerun of the command.

SUMMARY - REPRODUCIBLE EXECUTION

`datalad run` records a command and its impact on the dataset.

All dataset modifications are saved - use it in a clean dataset.

Data/directories specified as `--input` are retrieved prior to command execution.

Use one flag per input.

Data/directories specified as `--output` will be unlocked for modifications prior to a rerun of the command.

Its optional to specify, but helpful for recomputations.

SUMMARY - REPRODUCIBLE EXECUTION

`datalad run` records a command and its impact on the dataset.

All dataset modifications are saved - use it in a clean dataset.

Data/directories specified as `--input` are retrieved prior to command execution.

Use one flag per input.

Data/directories specified as `--output` will be unlocked for modifications prior to a rerun of the command.

Its optional to specify, but helpful for recomputations.

`datalad containers-run` can be used to capture the software environment as provenance.

SUMMARY - REPRODUCIBLE EXECUTION

`datalad run` records a command and its impact on the dataset.

All dataset modifications are saved - use it in a clean dataset.

Data/directories specified as `--input` are retrieved prior to command execution.

Use one flag per input.

Data/directories specified as `--output` will be unlocked for modifications prior to a rerun of the command.

Its optional to specify, but helpful for recomputations.

`datalad containers-run` can be used to capture the software environment as provenance.

Its ensures computations are ran in the desired software set up. Supports Docker and Singularity containers

SUMMARY - REPRODUCIBLE EXECUTION

`datalad run` records a command and its impact on the dataset.

All dataset modifications are saved - use it in a clean dataset.

Data/directories specified as `--input` are retrieved prior to command execution.

Use one flag per input.

Data/directories specified as `--output` will be unlocked for modifications prior to a rerun of the command.

Its optional to specify, but helpful for recomputations.

`datalad containers-run` can be used to capture the software environment as provenance.

Its ensures computations are ran in the desired software set up. Supports Docker and Singularity containers

`datalad rerun` can automatically re-execute run-records later.

SUMMARY - REPRODUCIBLE EXECUTION

`datalad run` records a command and its impact on the dataset.

All dataset modifications are saved - use it in a clean dataset.

Data/directories specified as `--input` are retrieved prior to command execution.

Use one flag per input.

Data/directories specified as `--output` will be unlocked for modifications prior to a rerun of the command.

Its optional to specify, but helpful for recomputations.

`datalad containers-run` can be used to capture the software environment as provenance.

Its ensures computations are ran in the desired software set up. Supports Docker and Singularity containers

`datalad rerun` can automatically re-execute run-records later.

They can be identified with any commit-ish (hash, tag, range, ...)

DATALAD RERUN

- `datalad rerun` is helpful to spare others and yourself the short- or long-term memory task, or the forensic skills to figure out how you performed an analysis
- But it is also a digital and machine-reable provenance record
- Important: The better the run command is specified, the better the provenance record
- Note: run and rerun only create an entry in the history if the command execution leads to a change.

QUESTIONS!

http://etc.ch/5xo7

UNLOCKING THINGS

- `datalad run` "unlocks" everything specified as `--output`

UNLOCKING THINGS

- `datalad run` "unlocks" everything specified as `--output`
- Outside of `datalad run`, you can use `datalad unlock`
- This makes annex'ed files *writeable*:

```
$ ls -l myfile
lrwxrwxrwx 1 adina adina  108 Nov 17 07:08 myfile -> .git/annex/objects/22/Gw/MD5E-s7--f447b20a7fcbf53a5d5

# unlocking
$ datalad unlock myfile
unlock(ok): myfile (file)
$ ls -l myfile
-rw-r--r-- 1 adina adina    7 Nov 17 07:08 myfile  # not a symlink anymore!
```

UNLOCKING THINGS

- `datalad run` "unlocks" everything specified as `--output`
- Outside of `datalad run`, you can use `datalad unlock`
- This makes annex'ed files *writeable*:

```
$ ls -l myfile
lrwxrwxrwx 1 adina adina  108 Nov 17 07:08 myfile -> .git/annex/objects/22/Gw/MD5E-s7--f447b20a7fcbf53a5d5

# unlocking
$ datalad unlock myfile
unlock(ok): myfile (file)
$ ls -l myfile
-rw-r--r-- 1 adina adina    7 Nov 17 07:08 myfile  # not a symlink anymore!
```

- `datalad save` "locks" the file again

```
$ datalad save
add(ok): myfile (file)
action summary:
  add (ok: 1)
  save (notneeded: 1)

$ ls -l myfile
lrwxrwxrwx 1 adina adina 108 Nov 17 07:08 myfile -> .git/annex/objects/22/Gw/MD5E-s7--f447b20a7fcbf53a5d5b
```

UNLOCKING THINGS

- `datalad run` "unlocks" everything specified as `--output`
- Outside of `datalad run`, you can use `datalad unlock`
- This makes annex'ed files *writeable*:

```
$ ls -l myfile
lrwxrwxrwx 1 adina adina  108 Nov 17 07:08 myfile -> .git/annex/objects/22/Gw/MD5E-s7--f447b20a7fcbf53a5d5

# unlocking
$ datalad unlock myfile
unlock(ok): myfile (file)
$ ls -l myfile
-rw-r--r-- 1 adina adina    7 Nov 17 07:08 myfile  # not a symlink anymore!
```

- `datalad save` "locks" the file again

```
$ datalad save
add(ok): myfile (file)
action summary:
  add (ok: 1)
  save (notneeded: 1)

$ ls -l myfile
lrwxrwxrwx 1 adina adina 108 Nov 17 07:08 myfile -> .git/annex/objects/22/Gw/MD5E-s7--f447b20a7fcbf53a5d5b
```

Some tools (e.g., MatLab) don't like symlinks. Unlocking or running matlab with "datalad run" helps!

REMOVING DATASETS

- As mentioned before, annexed data is write-protected. So when you try to `rm -rf` a dataset, this happens:

REMOVING DATASETS

- As mentioned before, annexed data is write-protected. So when you try to `rm -rf` a dataset, this happens:

```
$ rm -rf mydataset
rm: cannot remove 'mydataset/.git/annex/objects/70/GM/MD5E-s27246--8b7ea027f6db1cda7af496e97d4eb7c9.png/MD
rm: cannot remove 'mydataset/.git/annex/objects/70/GM/MD5E-s35756--af496e97d4eb7c98b7ea027f6db1cda7.png/MD
[...]
```

REMOVING DATASETS

- As mentioned before, annexed data is write-protected. So when you try to `rm -rf` a dataset, this happens:

```
$ rm -rf mydataset
rm: cannot remove 'mydataset/.git/annex/objects/70/GM/MD5E-s27246--8b7ea027f6db1cda7af496e97d4eb7c9.png/MD
rm: cannot remove 'mydataset/.git/annex/objects/70/GM/MD5E-s35756--af496e97d4eb7c98b7ea027f6db1cda7.png/MD
[...]
```

- (If you accidentally ever do this, you need to apply write permissions recursively to all files)

```
$ chmod -R +w mydataset
$ rm -rf mydataset                # success!
```

REMOVING DATASETS

- The correct way to remove a dataset is using `datalad remove`:

```
$ datalad remove -d ds001241
remove(ok): . (dataset)
action summary:
  drop (notneeded: 1)
  remove (ok: 1)
```

REMOVING DATASETS

- The correct way to remove a dataset is using `datalad remove`:

```
$ datalad remove -d ds001241
remove(ok): . (dataset)
action summary:
  drop (notneeded: 1)
  remove (ok: 1)
```

- If a dataset contains file for which no other remote copy is known, you'll get a warning:

```
$ datalad remove -d mydataset
[WARNING] Running drop resulted in stderr output: git-annex: drop: 1 failed

[ERROR  ] unsafe; Could only verify the existence of 0 out of 1 necessary copies; Rather than dropping thi
drop(error): interdisciplinary.png (file) [unsafe; Could only verify the existence of 0 out of 1 necessary
[WARNING] could not drop some content in /tmp/mydataset ['/tmp/mydataset/interdisciplinary.png'] [drop(/tm
drop(impossible): . (directory) [could not drop some content in /tmp/mydataset ['/tmp/mydataset/interdisci
action summary:
  drop (error: 1, impossible: 1)
```

REMOVING DATASETS

- The correct way to remove a dataset is using `datalad remove`:

```
$ datalad remove -d ds001241
remove(ok): . (dataset)
action summary:
  drop (notneeded: 1)
  remove (ok: 1)
```

- If a dataset contains file for which no other remote copy is known, you'll get a warning:

```
$ datalad remove -d mydataset
[WARNING] Running drop resulted in stderr output: git-annex: drop: 1 failed

[ERROR  ] unsafe; Could only verify the existence of 0 out of 1 necessary copies; Rather than dropping thi
drop(error): interdisciplinary.png (file) [unsafe; Could only verify the existence of 0 out of 1 necessary
[WARNING] could not drop some content in /tmp/mydataset ['/tmp/mydataset/interdisciplinary.png'] [drop(/tm
drop(impossible): . (directory) [could not drop some content in /tmp/mydataset ['/tmp/mydataset/interdisci
action summary:
  drop (error: 1, impossible: 1)
```

- In that case, use `--nocheck` to force removal:

```
$ datalad remove -d mydataset --nocheck                                    1 !
remove(ok): . (dataset)
```

REMOVING DATASETS

- If a dataset contains subdatasets, `datalad remove` will also error:

REMOVING DATASETS

- If a dataset contains subdatasets, `datalad remove` will also error:

```
$ datalad remove -d myds
drop(ok): README.md (file) [locking gin...]
drop(ok): . (directory)
[ERROR  ] to be uninstalled dataset Dataset(/tmp/myds) has present subdatasets, forgot --recursive? [remov
remove(error): . (dataset) [to be uninstalled dataset Dataset(/tmp/myds) has present subdatasets, forgot -
action summary:
  drop (ok: 3)
  remove (error: 1)
```

REMOVING DATASETS

- If a dataset contains subdatasets, `datalad remove` will also error:

```
$ datalad remove -d myds
drop(ok): README.md (file) [locking gin...]
drop(ok): . (directory)
[ERROR  ] to be uninstalled dataset Dataset(/tmp/myds) has present subdatasets, forgot --recursive? [remov
remove(error): . (dataset) [to be uninstalled dataset Dataset(/tmp/myds) has present subdatasets, forgot -
action summary:
  drop (ok: 3)
  remove (error: 1)
```

- In that case, use `--recursive` to remove all subdatasets, too:

```
$ datalad remove -d myds --recursive
uninstall(ok): input (dataset)
remove(ok): . (dataset)
action summary:
  drop (notneeded: 2)
  remove (ok: 1)
  uninstall (ok: 1)
```

REMOVING DATASETS

- If a dataset contains subdatasets, `datalad remove` will also error:

```
$ datalad remove -d myds
drop(ok): README.md (file) [locking gin...]
drop(ok): . (directory)
[ERROR  ] to be uninstalled dataset Dataset(/tmp/myds) has present subdatasets, forgot --recursive? [remov
remove(error): . (dataset) [to be uninstalled dataset Dataset(/tmp/myds) has present subdatasets, forgot -
action summary:
  drop (ok: 3)
  remove (error: 1)
```

- In that case, use `--recursive` to remove all subdatasets, too:

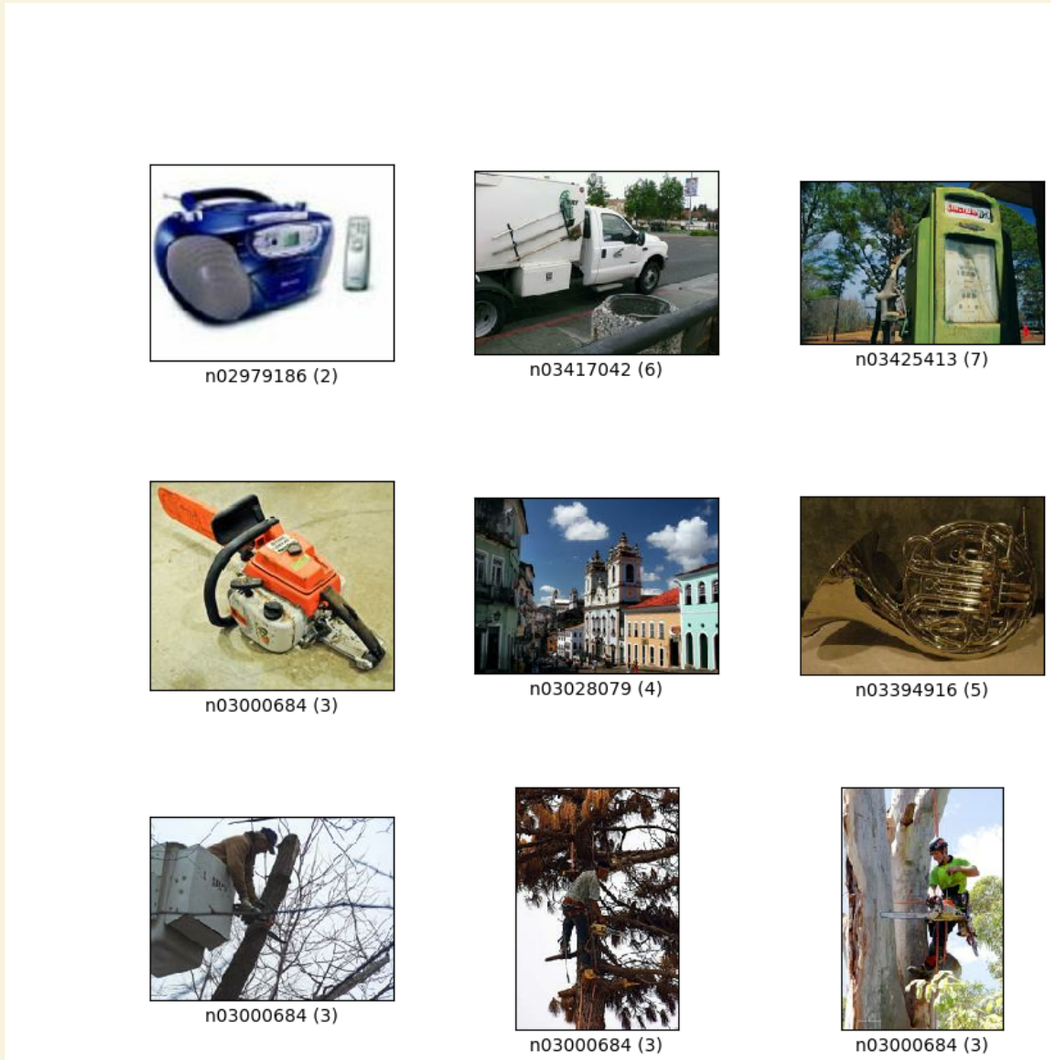
```
$ datalad remove -d myds --recursive
uninstall(ok): input (dataset)
remove(ok): . (dataset)
action summary:
  drop (notneeded: 2)
  remove (ok: 1)
  uninstall (ok: 1)
```

- A complete overview of file system operations is in handbook.datalad.org/en/latest/basics/101-136-filesystem.html

A MACHINE-LEARNING EXAMPLE

ANALYSIS LAYOUT

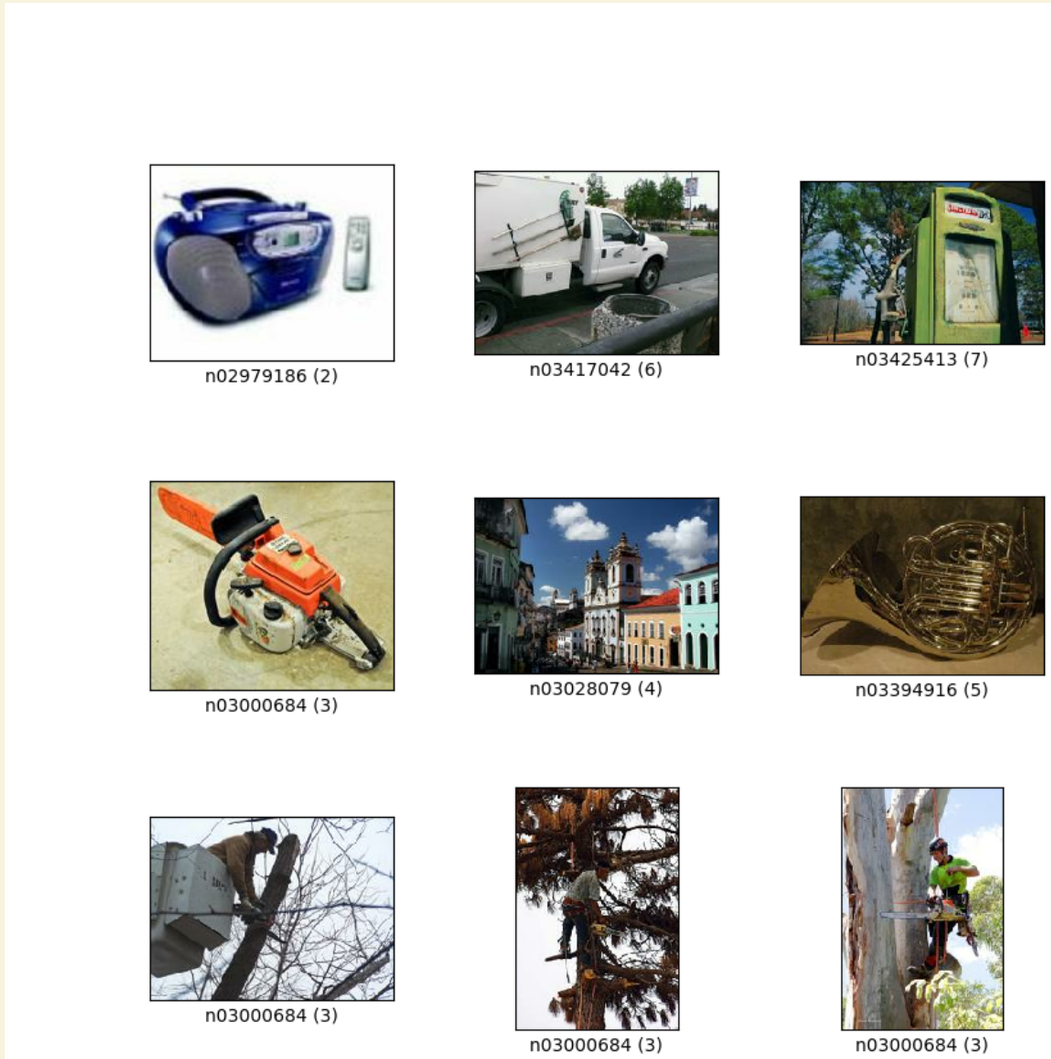
- Prepare an input data set



Imagenette dataset

ANALYSIS LAYOUT

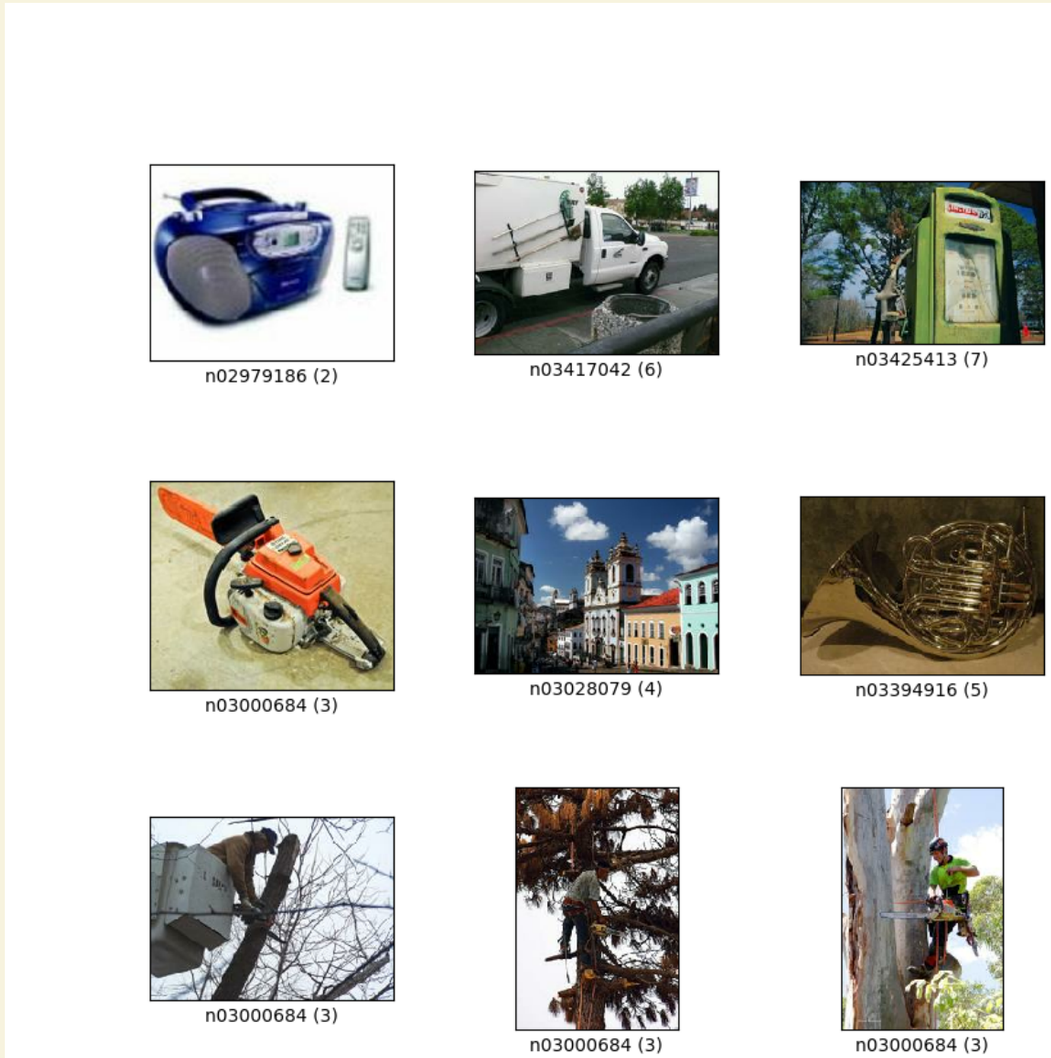
- Prepare an input data set
- Configure and setup an analysis dataset



Imagenette dataset

ANALYSIS LAYOUT

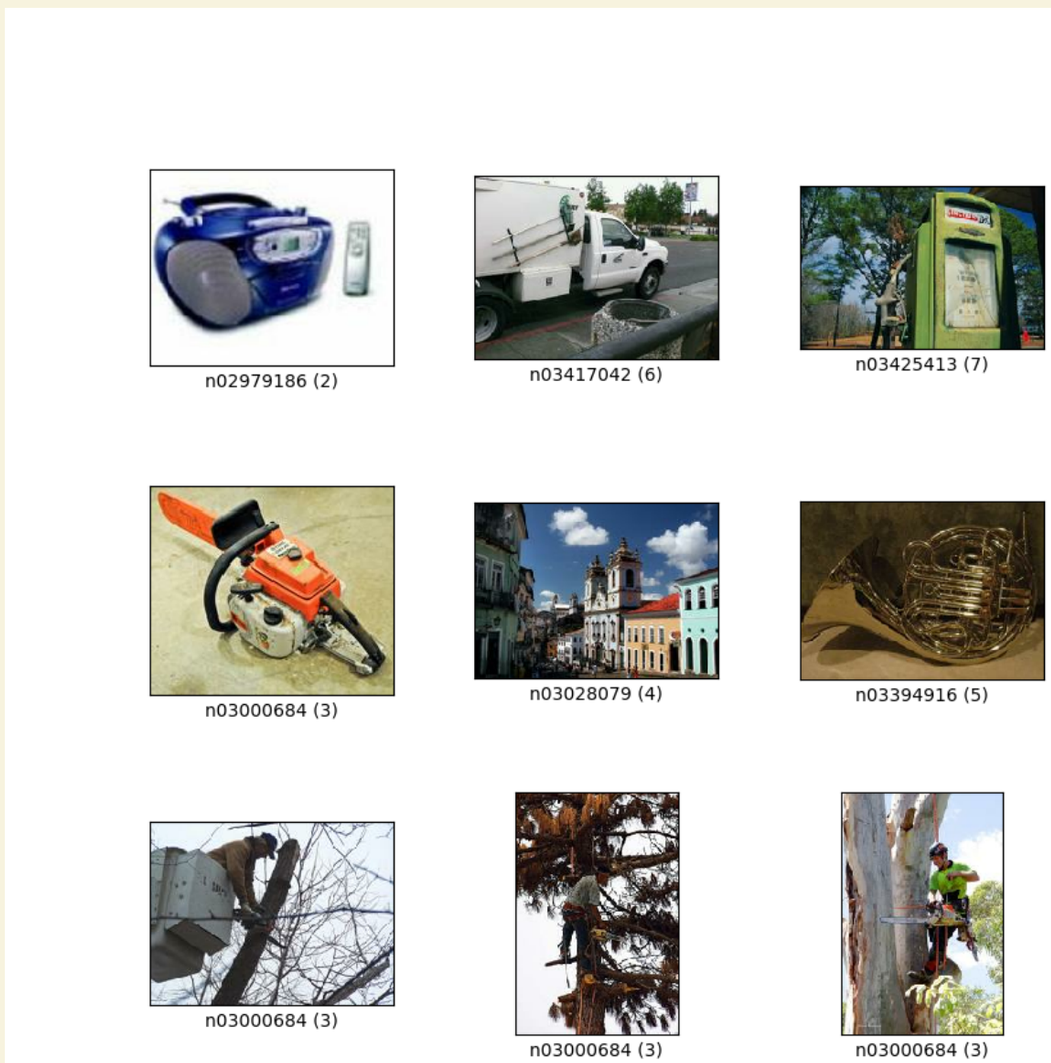
- Prepare an input data set
- Configure and setup an analysis dataset
- Prepare data



Imagenette dataset

ANALYSIS LAYOUT

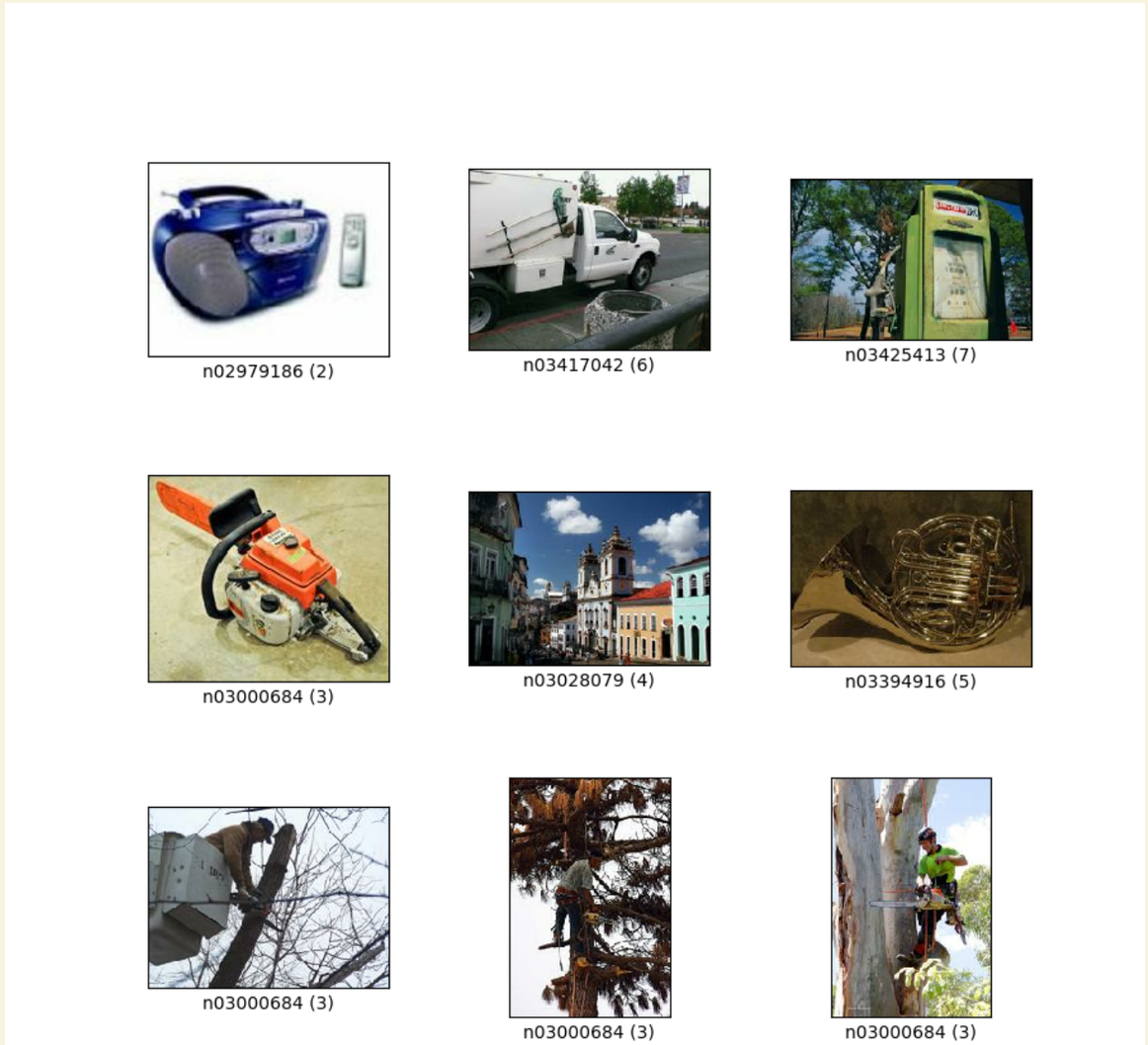
- Prepare an input data set
- Configure and setup an analysis dataset
- Prepare data
- Train models and evaluate them



Imagenette dataset

ANALYSIS LAYOUT

- Prepare an input data set
- Configure and setup an analysis dataset
- Prepare data
- Train models and evaluate them
- Compare different models, repeat with updated data



Imagenette dataset

PREPARE AN INPUT DATASET

- Create a stand-alone input dataset
- Either add data and `datalad save` it, or use commands such as `datalad download-url` or `datalad add-urls` to retrieve it from web-sources

CONFIGURE AND SETUP AN ANALYSIS DATASET

- Given the purpose of an analysis dataset, configurations can make it easier to use:
  - `-c yoda` prepares a useful structure
  - `-c text2git` keeps text files such as scripts in Git
- The input dataset is installed as a subdataset
- Required software is containerized and added to the dataset

PREPARE DATA

- Add a script for data preparation (labels train and validation images)
- Execute it using `datalad containers-run`

TRAIN MODELS AND EVALUATE THEM

- Add scripts for training and evaluation. This dataset state can be tagged to identify it easily at a later point
- Execute the scripts using `datalad containers-run`
- By dumping a trained model as a joblib object the trained classifier stays reusable

TIPS AND TRICKS FOR ML APPLICATIONS

TIPS AND TRICKS FOR ML APPLICATIONS

Standalone input datasets keep input data extendable and reusable

TIPS AND TRICKS FOR ML APPLICATIONS

Standalone input datasets keep input data extendable and reusable

Subdatasets can be registered in precise versions, and updated to the newest state

TIPS AND TRICKS FOR ML APPLICATIONS

Standalone input datasets keep input data extendable and reusable

Subdatasets can be registered in precise versions, and updated to the newest state

Software containers aid greatly with reproducibility

TIPS AND TRICKS FOR ML APPLICATIONS

Standalone input datasets keep input data extendable and reusable

Subdatasets can be registered in precise versions, and updated to the newest state

Software containers aid greatly with reproducibility

The correct software environment is preserved and can be shared

TIPS AND TRICKS FOR ML APPLICATIONS

Standalone input datasets keep input data extendable and reusable

Subdatasets can be registered in precise versions, and updated to the newest state

Software containers aid greatly with reproducibility

The correct software environment is preserved and can be shared

Re-executable run-records can capture all provenance

TIPS AND TRICKS FOR ML APPLICATIONS

Standalone input datasets keep input data extendable and reusable

Subdatasets can be registered in precise versions, and updated to the newest state

Software containers aid greatly with reproducibility

The correct software environment is preserved and can be shared

Re-executable run-records can capture all provenance

This can also capture command-line parametrization

TIPS AND TRICKS FOR ML APPLICATIONS

Standalone input datasets keep input data extendable and reusable

Subdatasets can be registered in precise versions, and updated to the newest state

Software containers aid greatly with reproducibility

The correct software environment is preserved and can be shared

Re-executable run-records can capture all provenance

This can also capture command-line parametrization

Git workflows can be helpful elements in ML workflows

TIPS AND TRICKS FOR ML APPLICATIONS

Standalone input datasets keep input data extendable and reusable

Subdatasets can be registered in precise versions, and updated to the newest state

Software containers aid greatly with reproducibility

The correct software environment is preserved and can be shared

Re-executable run-records can capture all provenance

This can also capture command-line parametrization

Git workflows can be helpful elements in ML workflows

DataLad is no workflow manager, but by checking out out tags or branches one can switch easy and fast between results of different models

WHY USE DATALAD?

WHY USE DATALAD?

- Mistakes are not forever anymore: Easy version control, regardless of file size

WHY USE DATALAD?

- Mistakes are not forever anymore: Easy version control, regardless of file size
- Who needs short-term memory when you can have run-records?

WHY USE DATALAD?

- Mistakes are not forever anymore: Easy version control, regardless of file size
- Who needs short-term memory when you can have run-records?
- Disk-usage magic: Have access to more data than your hard drive has space

WHY USE DATALAD?

- Mistakes are not forever anymore: Easy version control, regardless of file size
- Who needs short-term memory when you can have run-records?
- Disk-usage magic: Have access to more data than your hard drive has space
- Collaboration and updating mechanisms: Alice shares her data with Bob. Alice fixes a mistake and pushes the fix. Bob says "datalad update" and gets her changes. And vice-versa.

WHY USE DATALAD?

- Mistakes are not forever anymore: Easy version control, regardless of file size
- Who needs short-term memory when you can have run-records?
- Disk-usage magic: Have access to more data than your hard drive has space
- Collaboration and updating mechanisms: Alice shares her data with Bob. Alice fixes a mistake and pushes the fix. Bob says "datalad update" and gets her changes. And vice-versa.
- Transparency: Shared datasets keep their history. No need to track down a former student, ask their project what was done.