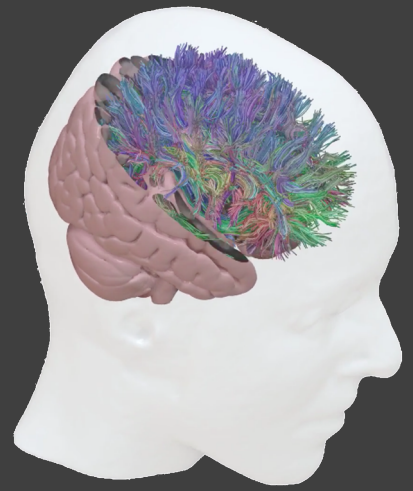
Provenance tracking with containerized pipelines for reproducible research



Michael Schirner
Brain Simulation Section (PI: Petra Ritter)
Charité—Universitätsmedizin Berlin
Berlin, July 13th, 2021

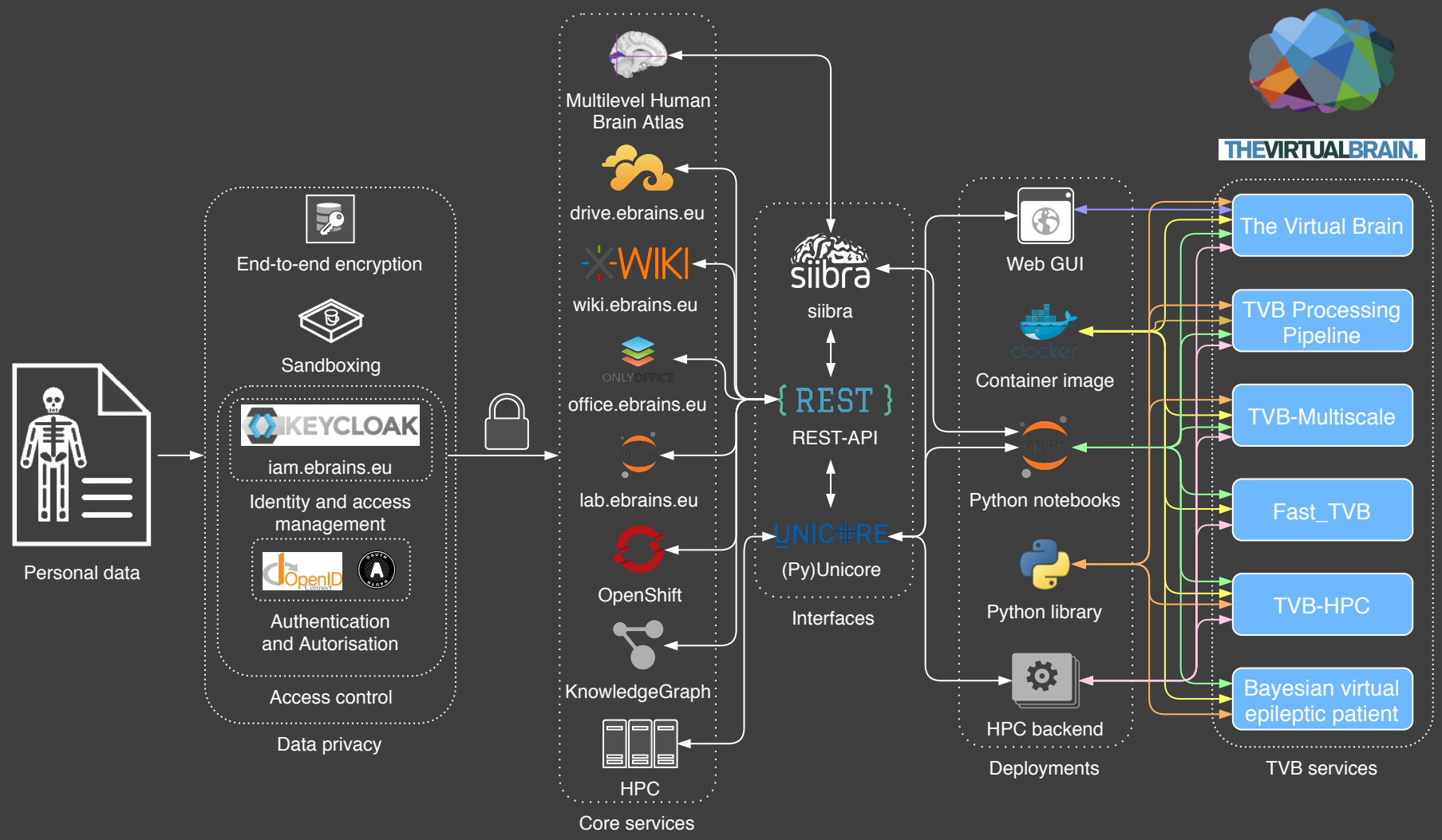
Outline

1. TVB on EBRAINS Image Processing Container Pipeline
2. Making the pipeline more reproducible and capturing more provenance with DataLad

Acknowledgments: much of the material shown here was created by the DataLad community. Special thanks to

- Michael Hanke
- Adina Wagner.

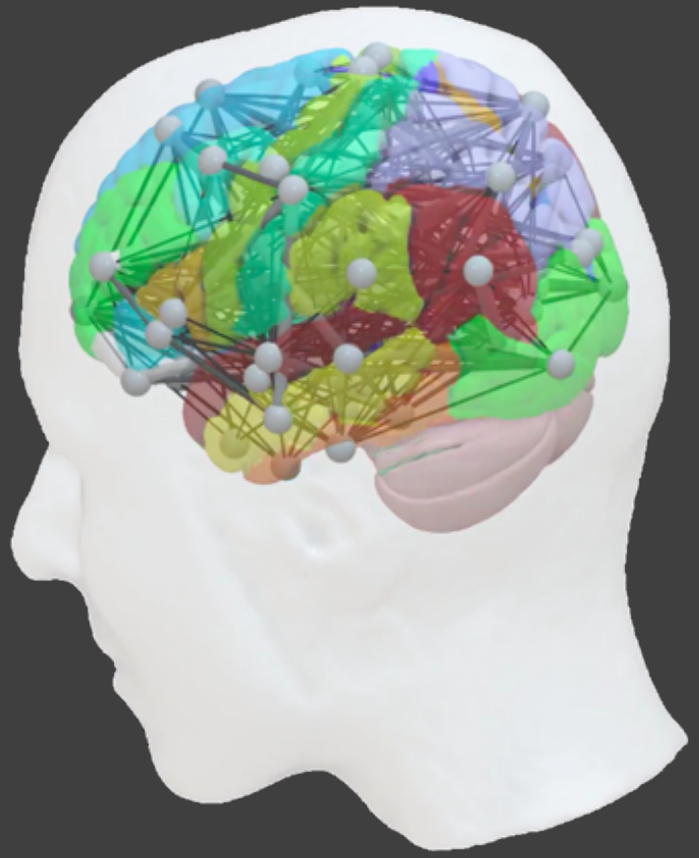
Thank You!



TVB-on-EBRAINS cloud infrastructure

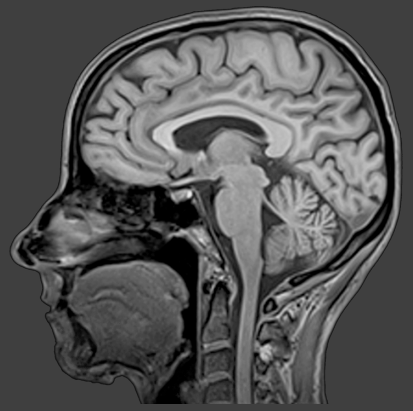
TVB Image Processing Pipeline

magnetic resonance imaging

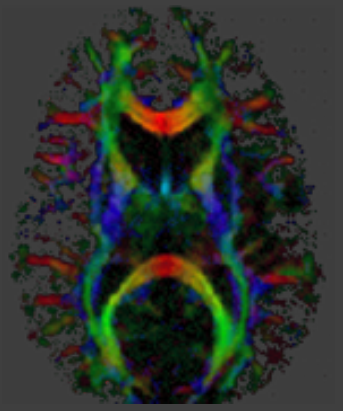


brain network model

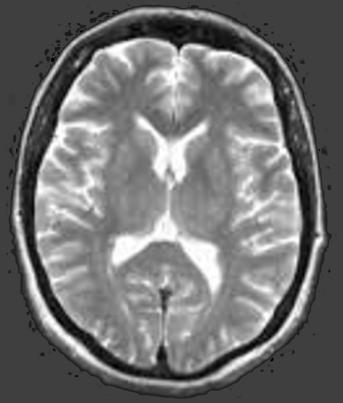
Pipeline input



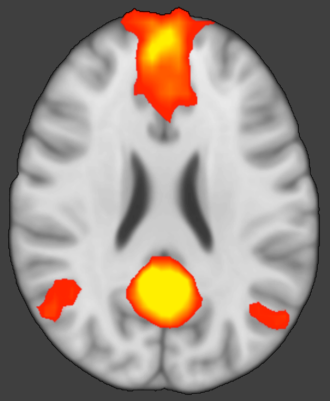
T1-weighted MRI



Diffusion-weighted MRI

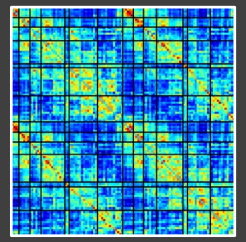


T2-weighted MRI

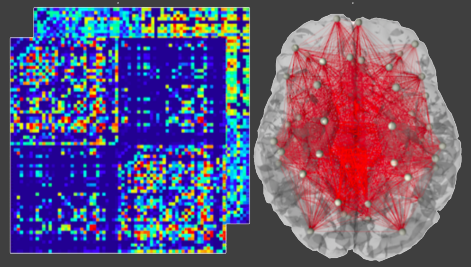


functional MRI

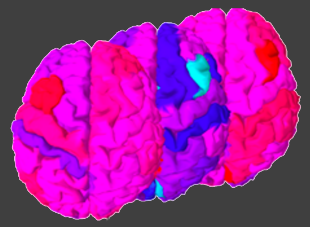
Pipeline output



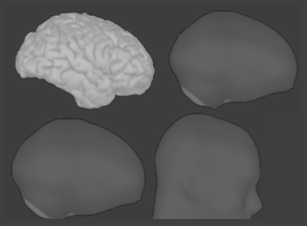
Functional connectivity



Structural connectivity

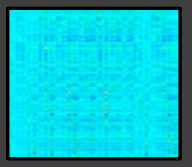


region-wise fMRI



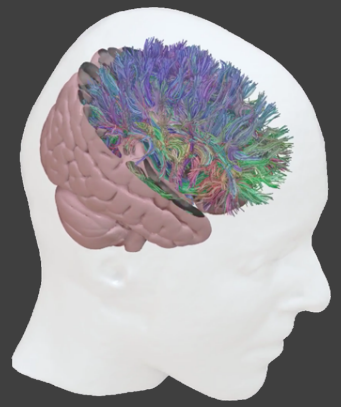
Surface triangulations

Brain maps

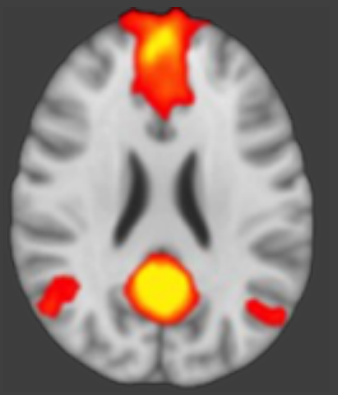


Projection matrices

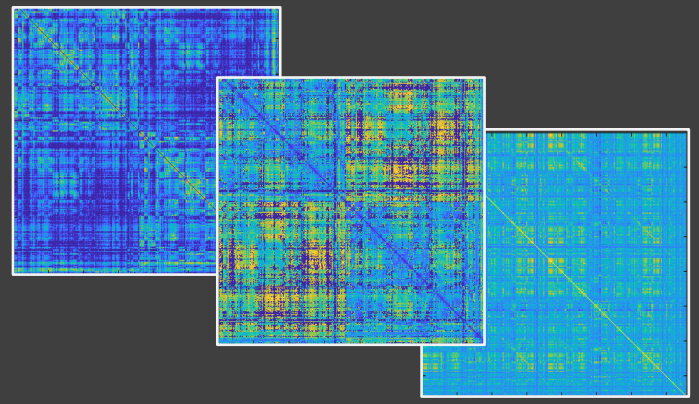
TVB Image Processing Pipeline



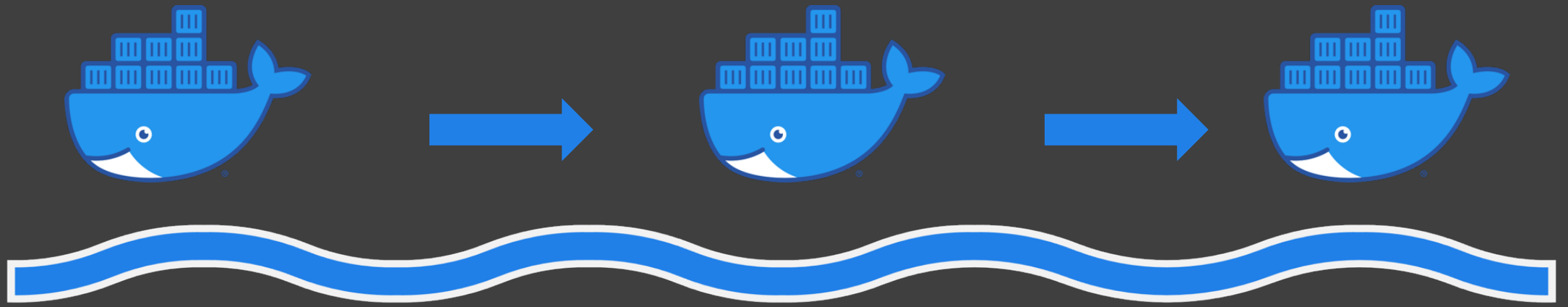
`bids/mrtrix3_connectome`



`nipreps/fmriprep`



`thevirtualbrain/tvb-pipeline-converter`



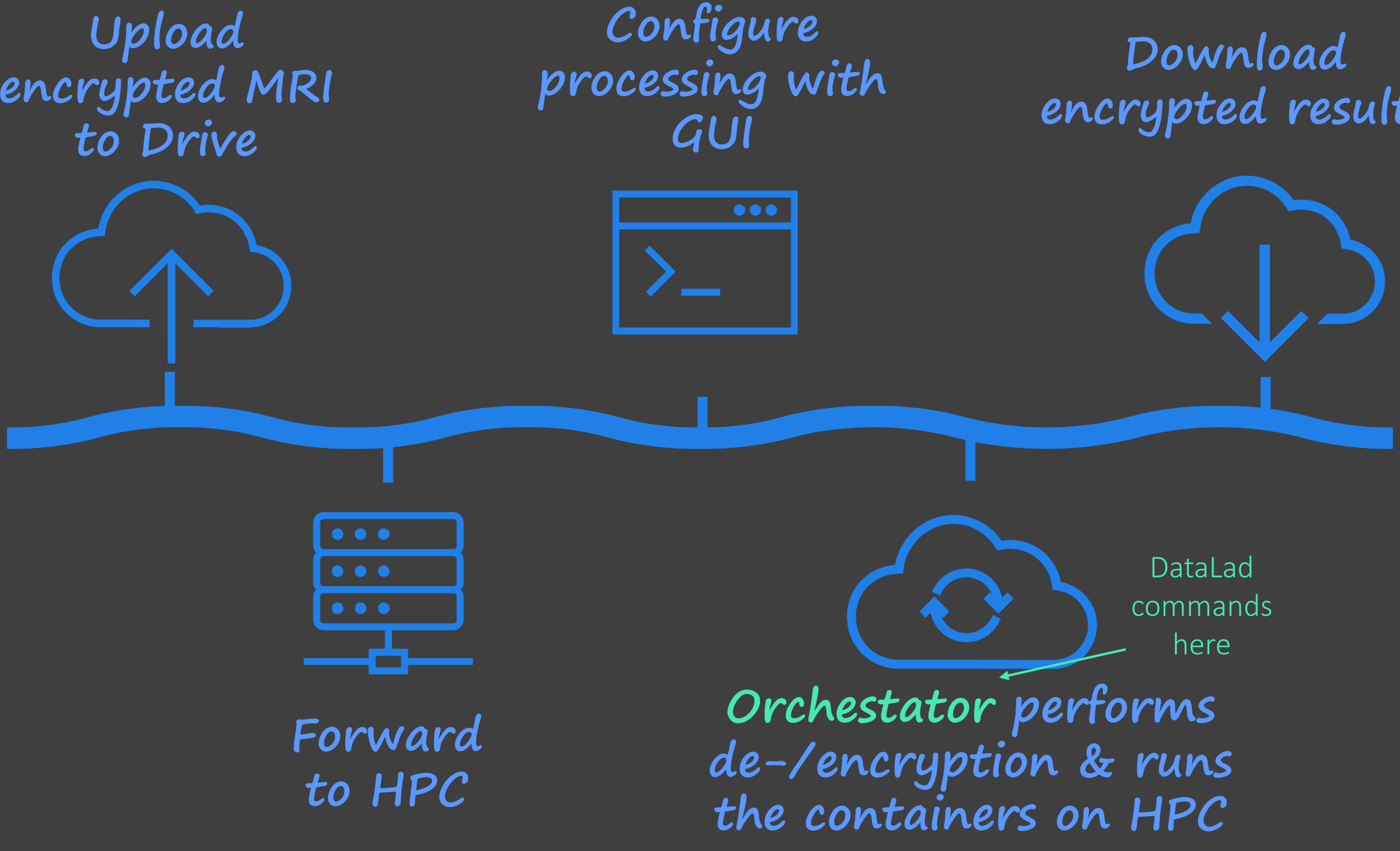
github.com/BIDS-Apps/MRtrix3_connectome/blob/master/Dockerfile

168 lines (155 sloc) | 8.52 KB

```
FROM ubuntu:18.04
MAINTAINER Robert E. Smith <robert.smith@florey.edu.au>

# Core system capabilities required
RUN apt-get update && DEBIAN_FRONTEND=noninteractive apt-get
    bc \
    build-essential \
    curl \
    dc \
    git \
    libegl1-mesa-dev \
    libopenblas-dev \
    nano \
    perl-modules-5.26 \
    python2.7 \
    python3 \
    tar \
    tcsh \
    tzdata \
    unzip \
    wget

# PPA for newer version of nodejs, which is required for bids
RUN curl -sL https://deb.nodesource.com/setup_12.x -o nodesou
    bash nodesource_setup.sh && \
    rm -f nodesource_setup.sh && \
    apt-get install -y nodejs

# NeuroDebian setup
```

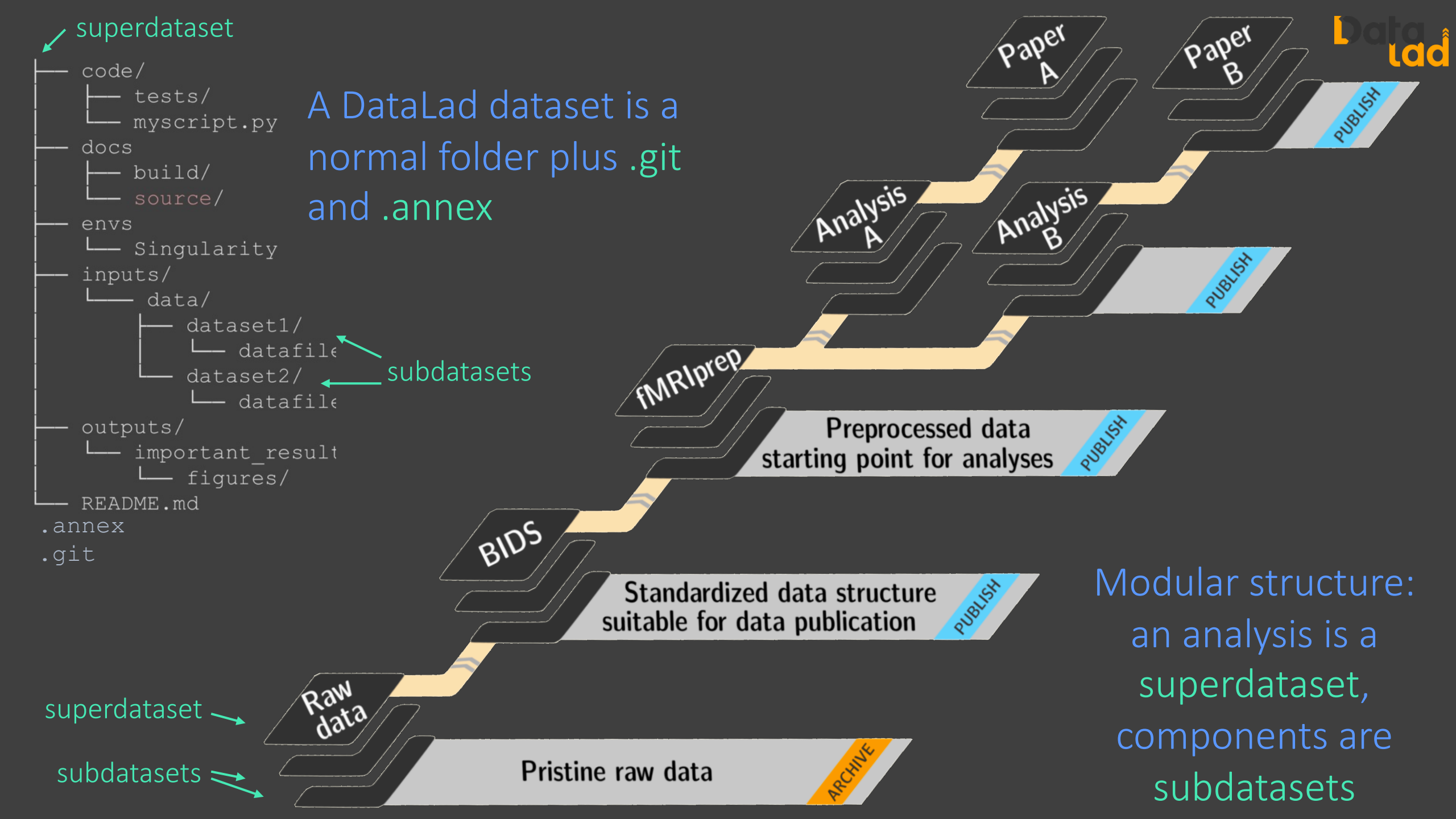
github.com/BIDS-Apps/MRtrix3_connectome/blob/master/Dockerfile

```
    SUBJECTS_DIR=/opt/freesurfer/subjects \
    FSLDIR=/opt/fsl \
    FSLOUTPUTTYPE=NIFTI \
    FSLMULTIFILEQUIT=TRUE \
    FSLTCLSH=/opt/fsl/bin/fsltclsh \
    FSLWISH=/opt/fsl/bin/fslwish \
    LD_LIBRARY_PATH=/opt/fsl/lib:$LD_LIBRARY_PATH \
    PATH=/opt/mrtrix3/bin:/usr/lib/ants:/opt/freesurfer/bin:/opt/freesurfer/mn
    PYTHONPATH=/opt/mrtrix3/lib:$PYTHONPATH

# MRtrix3 setup
# Commitish is 3.0.2 plus relevant hotfix
RUN git clone https://github.com/MRtrix3/mrtrix3.git /opt/mrtrix3 && \
    cd /opt/mrtrix3 && \
    git checkout 4ab54489f40997f7da1e1915c2adde3373cf6039 && \
    python3 configure -nogui && \
    python3 build -persistent -nopaginate && \
    git describe --tags > /mrtrix3_version && \
    rm -rf .git/ cmd/ core/ src/ testing/ tmp/ && \
    cd /

# Acquire extra MRtrix3 data
RUN wget -q "https://osf.io/v8n5g/download" -O /opt/mrtrix3/share/mrtrix3/labe
    wget -q "https://osf.io/ug2ef/download" -O /opt/mrtrix3/share/mrtrix3/labe

# Acquire script to be executed
COPY mrtrix3_connectome.py /mrtrix3_connectome.py
RUN chmod 775 /mrtrix3_connectome.py

COPY version /version

ENTRYPOINT ["/mrtrix3_connectome.py"]
```

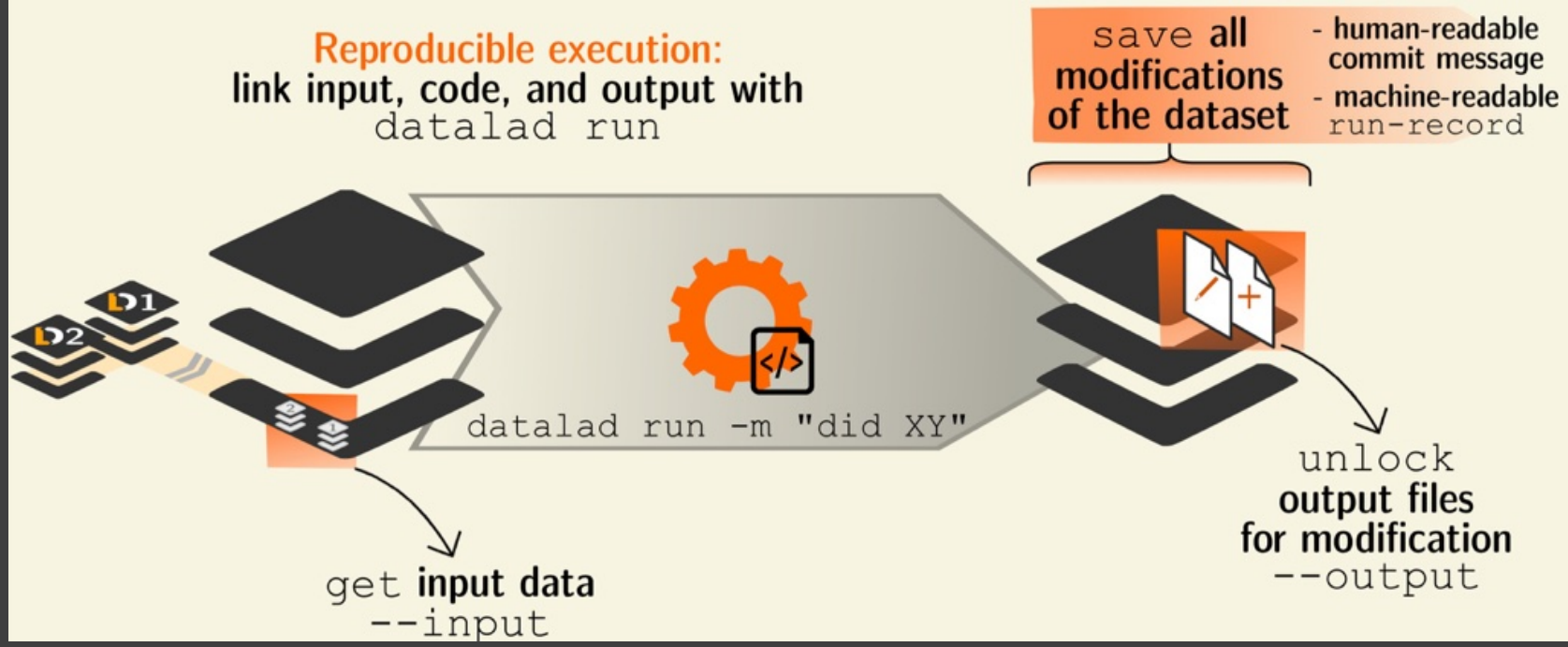

Upload encrypted MRI to Drive
Configure processing with GUI
Download encrypted results
Forward to HPC
DataLad commands here
Orchestator performs de-/encryption & runs the containers on HPC

Our situation

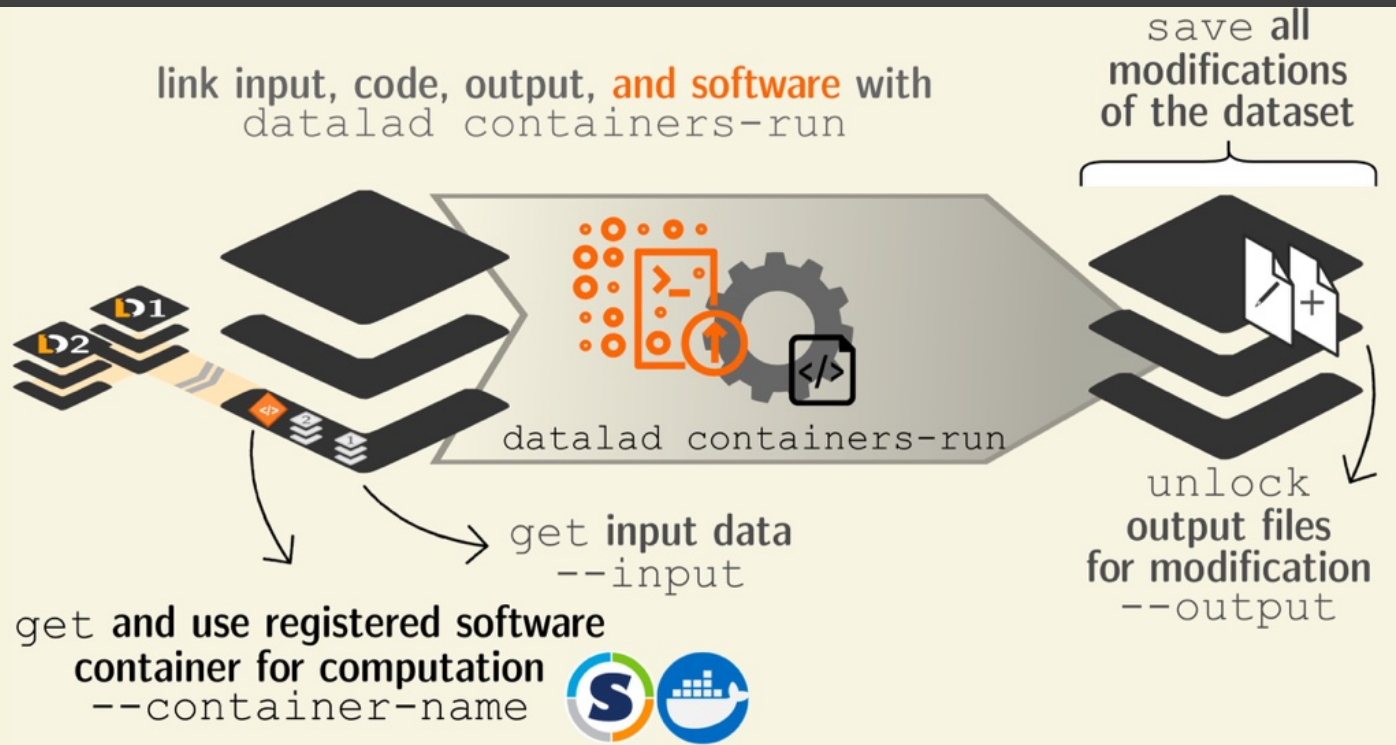
- created protected container pipelines
- manuscript submitted
- reviewer (paraphrased): „Your container pipeline involves user decisions (e.g. fmriprep parameters). iS tHIs rEaLLy rePrOdUcIbLe???“
- DataLad is a thorough solution for this problem
- First steps:
  - DataLad Handbook
  - exemplary UK Biobank / fmriprep workflows


superdataset
code/
tests/
myscript.py
docs
build/
source/
envs
Singularity
inputs/
data/
dataset1/
datafile
dataset2/
datafile
subdatasets
outputs/
important_result
figures/
README.md
.annex
.git
A DataLad dataset is a normal folder plus .git and .annex
Paper A
Paper B
PUBLISH
Analysis A
Analysis B
PUBLISH
fMRIprep
Preprocessed data starting point for analyses
PUBLISH
BIDS
Standardized data structure suitable for data publication
PUBLISH
Raw data
Pristine raw data
ARCHIVE
superdataset
subdatasets
Modular structure: an analysis is a superdataset, components are subdatasets

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



Containerization captures software and environment as provenance



DataLad

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

retrieved on demand

only specified output files get unlocked

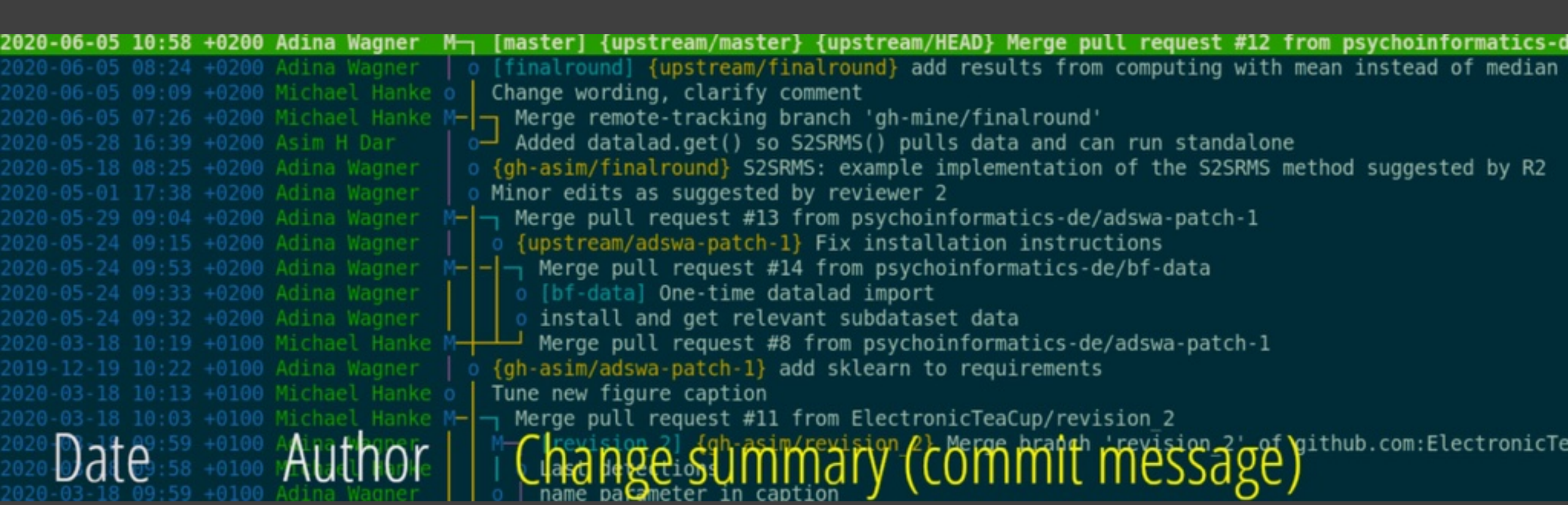
Execute the analysis script in a computationally reproducible manner

```
$ datalad containers-run -n neuroimaging-container \
  --input 'mri/*_bold.nii --output 'sub-*/LC_timeseries_run-*.csv' \
  "bash -c 'for sub in sub-*; do for run in run-1 ... run-8;
     do python3 code/extract_lc_timeseries.py \$sub \$run; done; done'"

-- Git commit -- Michael Hanke < ... @gmail.com>; Fri Jul 6 11:02:28 2019
    [DATALAD RUNCMD] singularity exec --bind {pwd} .datalad/e...
    === Do not change lines below ===
    {
     "cmd": "singularity exec --bind {pwd} .datalad/environments/nilearn.simg bash..",
     "dsid": "92ea1faa-632a-11e8-af29-a0369f7c647e",
     "inputs": [
      "mri/*.bold.nii.gz",
      ".datalad/environments/nilearn.simg"
     ],
     "outputs": ["sub-*/LC_timeseries_run-*.csv"],
     ...
    }
    ^^^ Do not change lines above ^^^
---
 sub-01/LC_timeseries_run-1.csv | 1 +
...
```

Track computations performed with containers

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

    [DATALAD RUNCMD] analyze iris data with classification analysis

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

`datalad (containers)-run` produces a machine-readable provenance record, identified by a Git commit hash



- the dataset logs everything that was done, by whom & when
- reset dataset to historic state
- add, remove or revisit individual steps

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

```
 $ 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)
```

- rerun recorded computations via commit hash
- helps to reproduce and verify a result (via content hash)
- granularity can be freely decided
- recompute: not everything needs to be stored or transported

Setting up a reproducible, version-controlled, provenance-tracked workflow

Install DataLad

With Miniconda DataLad can be installed with just three commands

```
ssh daint # or the supercomputer you are working on
module load daint-mc # supercomputer-specific module
module load cray-python/3.8.5.0 # load Python environment

# The easiest way to install DataLad with all
# dependencies on a supercomputer without root
# permissions is by using conda
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh
# acknowledge license, initialize Miniconda3, close and
# re-open shell
conda install -c conda-forge datalad
```

Install DataLad extensions

Two DataLad extensions are required:

datalad-container and

datalad-neuroimaging

container support

Metadata extraction

Install them with pip like this:

```
pip install datalad-neuroimaging datalad-container
```

Configure DataLad

Setup your git identity. Specify your git username and email, which will be used to track changes in your DataLad data set: changes you make are associated with your name and email address.

```
cd ~
git config --global --add user.name "MichaelSchirner"
git config --global --add user.email m.schirner@fu-berlin.de
```

Create an input dataset

If your data is not yet a DataLad dataset, you can transform it into one with the following commands.

```
# create a dataset in an existing directory
$ datalad create -f .
# save its contents
$ datalad save . -m "Import all data"
```

For large datasets (e.g. HCP, UK Biobank),
we create subdataset for each subject

Create a new DataLad dataset

First we create a new empty dataset and then add existing containers to it.

```
$ datalad create pipeline
$ cd pipeline
```

Add container(s)

Add a software-container to the dataset using *datalad containers-add* from the datalad-container extension. The *--url* parameter can be a local path to your container image, or a URL to a container hub such as Dockerhub or Singularity Hub.

```
$ datalad containers-add cat --url<path/or/url/to/image> \
--call-fmt "singularity run -B {{pwd}} --cleanenv {img} {cmd}"
```

Re-usable container pipeline dataset

Create empty dataset and then clone and add the container as subdataset

Here we create an empty dataset and then clone and register our needed container(s) as a subdataset (documentation). Arguments like *--bind* that are intended for singularity rather than the underlying command should be specified with *--call-fmt* when calling *containers-add*. It's also fine to edit the cmdexec value in *.datalad/config* after the fact.

```
# Create a source dataset for all analysis components
source_ds="pipeline"
datalad create $source_ds
cd $source_ds

# clone the container-dataset as a subdataset.
containername='tvbpipe-fmriprep'
containerstore="/scratch/snx3000/bp000225/tvbpipe-fmriprep"
datalad clone -d . ${containerstore} code/pipeline

# Register the container in the top-level dataset.
# If necessary, configure your own container call in the
# --call-fmt argument. If your container does not need a
# custom call format, remove the --call-fmt flag and its
# options below.
datalad containers-add \
  --call-fmt 'singularity run -B {{pwd}} --cleanenv {img} {cmd}' \
  -i /scratch/snx3000/bp000225/${source_ds}/code/pipeline/.datalad/environments/${containe
  $containername

# amend the previous commit with a nicer commit message
git commit --amend -m 'Register pipeline dataset'
```

Analysis superdataset to specify all dependencies of an analysis

Import custom code

Here we can copy custom code into the dataset. For example, FreeSurfer (part of fmriprep, MRtrix3_connectome) needs a license file (free but must be individually obtained), which is now copied from the home folder into the data set.

```
# committing changes after copying license file
cp ~/license.txt code/license.txt
datalad save -m "Add Freesurfer license file"
```

Quickly test-run the container on the login node

Before we create SLURM batch scripts we will give the container a test run to make sure we set up everything correctly so far.

```
# load Singularity module again in case there was an
# environment change
module load singularity

# the subject we test
subid="sub-CON03"

# create workdir for fmriprep inside the dataset to
# simplify singularity call PWD will be available in the
# container
mkdir -p .git/tmp/wdir

# Replace the command with an fmriprep parametrization
# that fits your analysis
datalad containers-run \
  -m "Compute ${subid}" \
  -n tvbpipe-fmriprep \
  --explicit \
  -o fmriprep/${subid} \
  -i inputs/data/${subid}/ses-postop/ \
  -i code/license.txt \
  "inputs/data/${subid} . participant --participant-label $subid \
    --anat-only -w .git/tmp/wdir --fs-no-reconall --skip-bids-validation \
    --fs-license-file code/license.txt"
```

Limitations

- not every feature of DataLad may work „out of the box“
- sometimes customization is required

⤴ Find-out-more: Fine-tuning: Enable re-running

If you want to make sure that your dataset is set up in a way that you have the ability to rerun a computation quickly, the following fMRIprep-specific consideration is important: If fMRIprep finds preexisting results, it will fail to run. Therefore, all outputs of a job need to be removed before the job is started[3]. We can simply add an attempt to do this in the script (it wouldn't do any harm if there is nothing to be removed):

```
(cd fmriprep && rm -rf logs "$subid" "$subid.html" dataset_descriptic
(cd freesurfer && rm -rf fsaverage "$subid")
```

```
# pybids (inside fmriprep) gets angry when it sees dangling symlir
# of .json files -- wipe them out, spare only those that belong tc
# the participant we want to process in this job
find sourcedata -mindepth 2 -name '*.json' -a ! -wholename "$1"'*'
```

meanwhile fixed in PyBIDS thanks to community interaction https://github.com/bids-standard/pybids/issues/631

Summary / take home messages

- The presented workflow is quite complex
- Nevertheless, it boils down to a handful of DataLad commands
- It‘s not difficult to re-combine the commands for different use cases
- Containers capture all software and environment
- DataLad captures how containers are executed including all varying parts (e.g. ad-hoc configuration, live user input, etc.)
- Enables everyone (e.g. reviewers, trainees) to re-run individual steps of the workflow on a fine-grained level on another machine with a simple command


THANK
YOU

Questions to the audience

Q1: Was the presentation understandable? What was not clear?

Q2: Is this type of large-scale processing relevant for your project?

Q3: Who in the audience considers using such a technology?

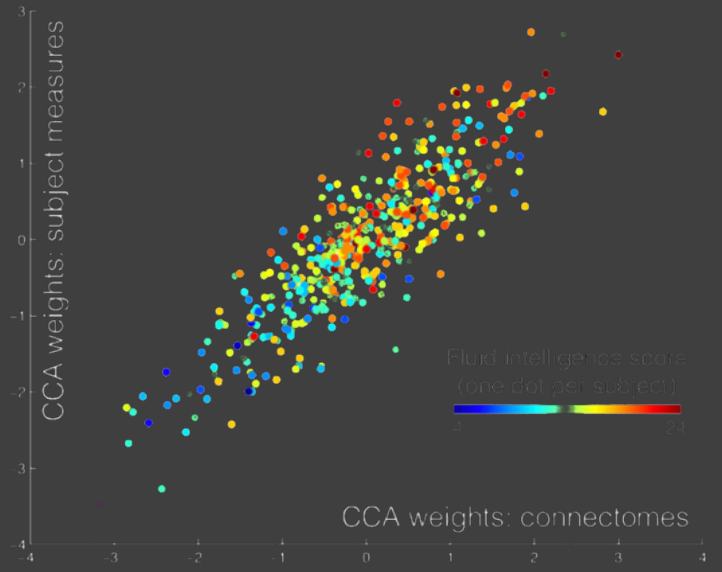
Q4: Provenance tracking: on a scale from hand-written lab book to DataLad – where do you locate yourself?

fMRI contains highly personal information

A positive-negative mode of population covariation links brain connectivity, demographics and behavior

Stephen M Smith[1], Thomas E Nichols[2], Diego Vidaurre[3], Anderson M Winkler[1], Timothy E J Behrens[1], Matthew F Glasser[4], Kamil Ugurbil[5], Deanna M Barch[4], David C Van Essen[4] & Karla L Miller[1]

We investigated the relationship between individual subjects' functional connectomes and 280 behavioral and demographic measures in a single holistic multivariate analysis relating imaging to non-imaging data from 461 subjects in the Human Connectome Project. We identified one strong mode of population co-variation: subjects were predominantly spread along a single 'positive-negative' axis linking lifestyle, demographic and psychometric measures to each other and to a specific pattern of brain connectivity.



picture vocabulary test
fluid intelligence (number of correct responses)
delay discounting (area under the curve for discounting of $200)

life satisfaction
list sorting working memory test
oral reading recognition test

sustained attention continuous performance test (specificity)
delay discounting (area under the curve for discounting of $40000)
picture sequence memory test

peg-board dexterity test (time taken)
visual acuity (ratio)
no history of psychiatric or neurologic disorders - father
pattern comparison processing speed
2-minute walking endurance test

Included in CCA

Variance explained:
2%
17%

thought problems score (self-report)
still smoking
perceived stress score
regional taste intensity score
rule breaking behavior score (self-report)
anger-physical aggression score
times used any tobacco today
Pittsburgh sleep quality index (higher is worse)
drug or alcohol problems - father
total weekdays with any tobacco in last week

positive test for THC (cannabis)

Smith et al Nat Neurosci. 2015

§

Art. 24 GDPR

Responsibility of the controller

"…shall implement appropriate technical and organisational measures to ensure and…demonstrate that processing is performed in accordance with this Regulation"

Controller determines **alone or jointly with others** the purposes and means of processing

Responsible for protecting the personal data!

Data privacy risks

Intercept during transit

Break-in on device

User impersonation

Problem 1: health data shall be secured by default

***Solution**: health data shall be always encrypted except during processing*

Problem 2: how to secure unencrypted data during processing?

***Solution**: Sandboxes, temporary mount file systems, authenticated data containers*

Exposed (anonymized)

Tracked

VERSION HISTORY

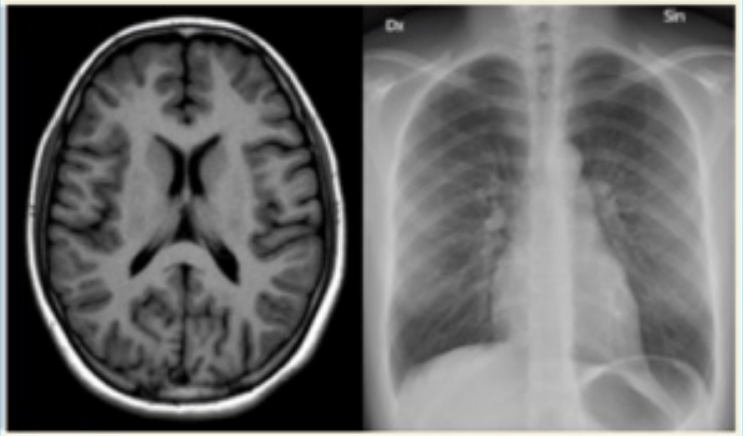
FILE NAMES

SALTED FILE CONTENT HASHES

```
patient_record/
MD5E-s3016749--64364d53b5f41fbe5bccfe5e.jpg  photo.jpg
MD5E-s3016749--8ccc1989f2ab223874428092.json  insurance.json
visit/
MD5E-s932874--ad9e369d8c908feb6d3fe84e.json  1.json
MD5E-s762273--1ec751b8cb432d16f6a39e64.json  2.json
imaging/
MD5E-s30983443--30dcd89ecfd5dd546f8ccc19.img  brain/20031.img
MD5E-s73826423--9432d12b1581f220c8b75953.img  thorax/20201.img
```

PRISTINE INFORMATION

```
{"account": "293924",
 "id": "a399b30033d"...}
{"date": "2020-04-21",
 "diagnosis": "...",
 "notes": "..." ...}
```



Authorized/local access only

METADATA

ANON IDS

```
{"type": "patient", "id": "122bf-342-422",
 "imaging": ["brain", "thorax"], "sex": "male",
 "diagnosis": ["F51", "G12", "G52"], "age": 52...}
```

Why?

- „Returning to work…later…we had a question: which were the final versions, the ones which should go in the article?...It is possible…that we…put the wrong version of some figures in the final copy”
- “When he went back to the old software library to try and do it, he couldn’t remember how the software worked…he abandoned the project”
- “Unfortunately, from the paper itself he couldn’t figure out what … [parameters] … were being used. He spoke the author of the paper, who replied: “…we forgot which parameters gave the nice picture you see in the published article…and thought it best to leave things vague””

From Buckheit & Donoho, 1995

Why??

- „Data available upon reasonable request“
- „When I run it on my machine it gives me the following error: …“
- "What is the difference between `data_version1_edit.csv` and `data_version8_new_final.csv`?
- „Which value did you set when it asks for input at step 9?“
- „But the code worked when I ran it last month!“
- „Where did this file come from?“

Why? To make error correction easier.

Problem: a method does not work

- is the approach itself incorrect?
- is the approach ok, but the implementation incorrect?
- is the implementation ok, but are the parameters incorrect?
- are the parameters ok, but the interpretation of the results incorrect?

From Buckheit & Donoho, 1995

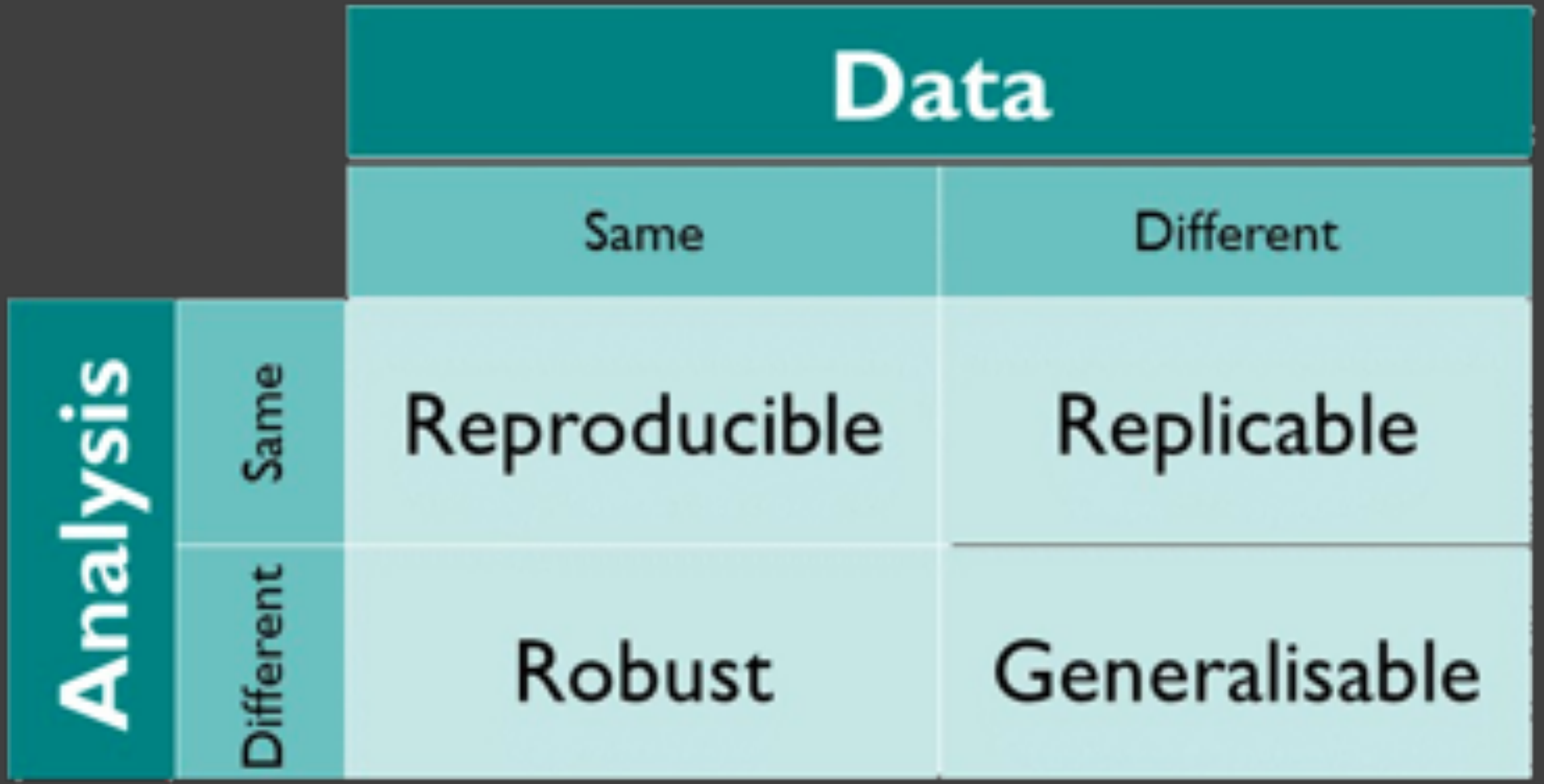
"An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures."

"When we publish articles containing figures which were generated by computer, we also publish the complete software environment which generates the figures.

From Buckheit & Donoho, 1995

Research data management

- to create, use, publish, share, archive, re-use, destroy
- digital objects
  - code / software
  - raw data
  - processed data
  - results
  - figures, manuscripts
- in a way that makes these digital assets
  - findable
  - accessible
  - interoperable
  - re-usable



https://the-turing-way.netlify.app/reproducible-research/overview/overview-definitions.html

Open Scholarship: what science can learn from (open source) software development

- transparency & accessability: everybody can engage
- version control: unique persistent identifiers for digital objects
- containerization (dependency management): code, data and environment in one robust package
- attribution: “commits” for fine-grained tracking of contributions
- issue tracking: peer-review by everyone and all the time
- unit testing: provenance; from raw data to figure
- continuous integration: consistent deployment of high-quality artefacts at all stages of development

Reproducible workflows

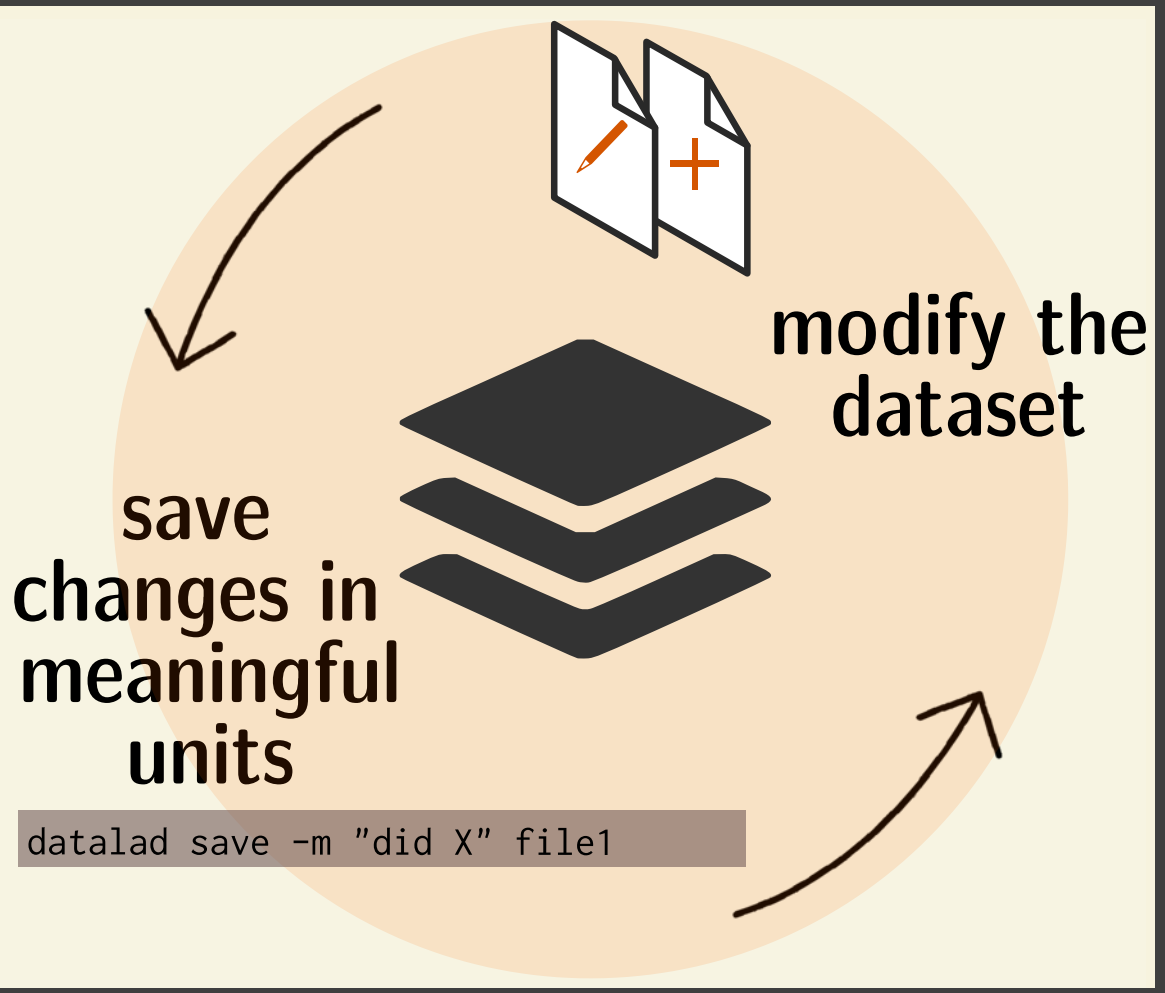
- version control datasets
- data provenance: history of digital objects
- nest datasets in hierarchy of modular units
- organize data with widely-used standards (e.g. BIDS)
- annotate and describe data with metadata and documentation
- automatically update products using continuous integration pipelines
- make the data accessible via web interfaces
- make code and computational environment reproducible with software containers (Docker, Singularity) and virtual environments

DataLad

- “Git for data”
- based on **Git** and **git-annex**
- command-line tool for Linux, OS X, Windows
- version control arbitrarily large files
- provenance tracking
- share and obtain data
- reproducible workflows
- domain-agnostic

- DataLad datasets
  - are directories managed by DataLad
  - are Git repositories
  - can be nested: linked subdirectories
  - can be created or installed
  - can branch and merge

Local version control



datalad create	Create empty dataset
datalad save	record state of dataset to history incl. commit message
datalad download-url	obtain web content and record origin
datalad status	report tracking state