Adina S. **Wagner**[*1],
Laura K. **Waite**[*1],
Małgorzata **Wierzba**[*1,2],
Felix **Hoffstaedter**[1,3],
Alexander Q. **Waite**[1],
Benjamin **Poldrack**[1],
Simon B. **Eickhoff**[1,3]
& Michael **Hanke**[1,3]

FAIRly big: A framework for reproducible processing of large-scale data

[1] Institute of Neuroscience and Medicine, Brain & Behaviour INM-7, Research Center Jülich, Germany
[2] Laboratory of Brain Imaging, Nencki Institute of Experimental Biology, Polish Academy of Sciences, Warsaw, Poland
[3] Institute of Systems Neuroscience, Medical Faculty, Heinrich Heine University Düsseldorf, Germany

Big datasets challenge compute infrastructure & reproducibility alike. It takes much to store and compute the largest datasets, it takes even more to do so in a way that aids transparency & reuse. We developed a framework for scalable and reproducible processing, connecting container technology, job scheduling, and version control tools, with a proof-of-principle analysis on UK Biobank imaging data.

The steps of the framework. 1-4 are **set up** by a bootstrap script based on user-input. Main features are version control, ephemeral work-spaces & computational provenance

1. Create a Dataset

```
$ datalad create ukb-vbm
```

2. Link input data

```
$ datalad clone -d . \
    ${datastore}#~ukb-bids
```

3. Link processing pipeline

```
$ datalad clone -d . \
    ${containerstore}#~cat code/cat
$ datalad containers-add \
    code/cat/... --name cat
```

```
# draft a datalad (containers-)run
# command for analysis granularity
# of your choice (e.g., subject) &
# create its ephemeral workspace.
# Pick job scheduler and resources
```

4. Develop compute job

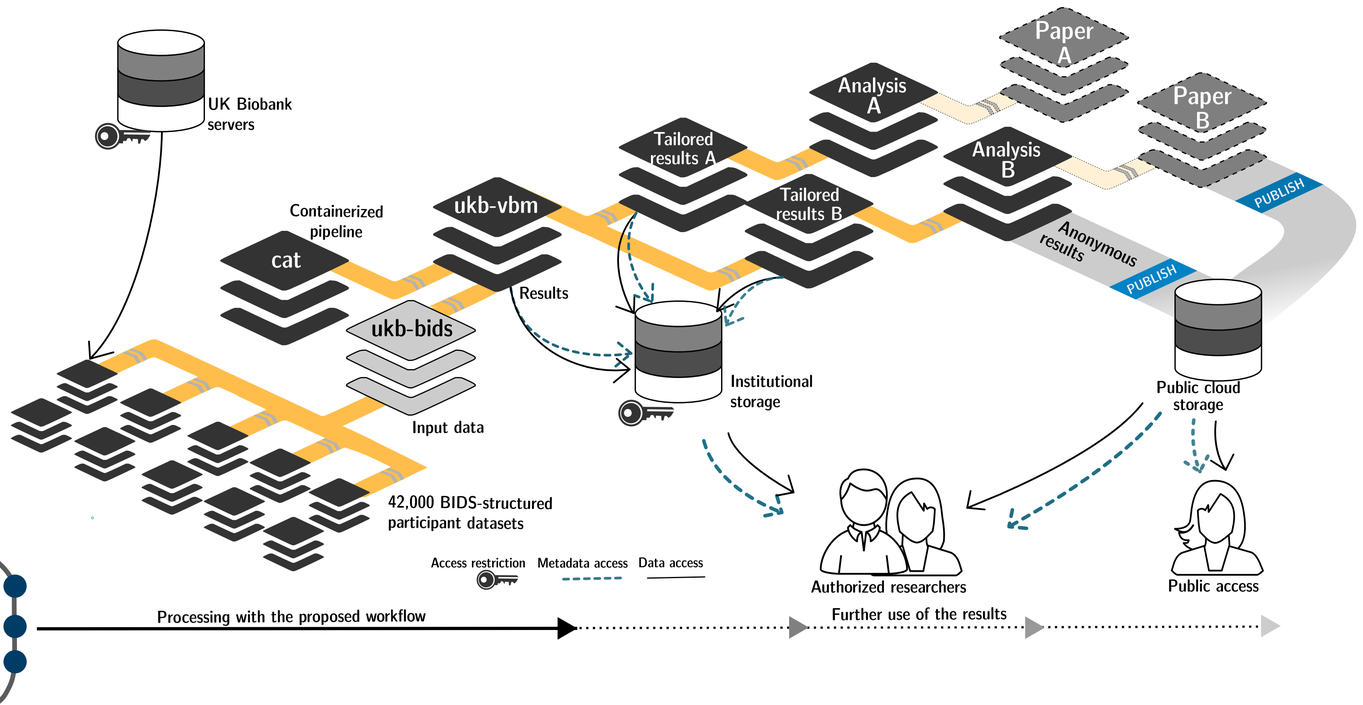
5. Parallel execution

```
# job scheduler
$ condor_submit ...
$ sbatch ...
```

Use software development routines for data analysis

6. Result consolidation

```
# octopus-merge all "job" branches
$ git merge -m "Merge results" \
  $(git branch -al | grep 'job')
```



DataLad[1] is a data management & data publication tool based on popular version control systems (Git, git-annex). Its datasets ease public or private data retrieval or distribution by streamlining data access for the desired target audience, and are central in the framework. UKB data was retrieved and structured to BIDS into one dataset per subject with datalad-ukbiobank[2].


```
#!/bin/bash
# fail on any issue, show commands
set -e -u -x
# name arguments for readability
dssource="$1"
pushgitremote="$2"
subid="$3"

# obtain the analysis dataset, which
# also tracks the required inputs
datalad clone "${dssource}" ds
cd ds

# register location for result
# deposition, separate from the input
# source for performance reasons only
git remote add outputstore
  ↪ "$pushgitremote"

# all job results will be put into
# a job-specific, dedicated branch
git checkout -b "job-$JOBID"

# START OF APPLICATION-SPECIFIC CODE
# pull down input data manually,
# only needed for wildcard-based file
# selection in the next command
datalad get -n "inputs/ukb/${subid}"
# datalad containers-run executes
# the "cat" computational pipeline.
# specified inputs are auto-obtained,
# specified outputs are saved with
# provenance record
datalad containers-run \
  -m "Compute subject ${subid}" \
  -n cat \
  --explicit \
  --output "${subid}" \
  --input
  ↪ "inputs/ukb/${subid}/*T1w.nii.gz"
  "<container invokation arguments>"
# END OF APPLICATION-SPECIFIC CODE

# push result file content to the
# configured "storage-remote"
datalad push --to storage-remote

# push branch with provenance records
# needs a global lock to prevent
# write conflicts
flock "$DSLOCKFILE" git push
  ↪ outputstore

# log entry to mark non-error exit
echo SUCCESS
```

The framework links different analysis components (data, processing pipeline) as a dataset hierarchy. A generic compute job defines an analysis subset (e.g., one subject), and, during parallel computing with a job scheduler (e.g., HTCondor, SLURM), bootstraps an ephemeral work-space by cloning the top-most dataset. It retrieves relevant input data and software containers from compressed storage, captures execution provenance on a branch, and pushes results back. The number of concurrent jobs are adjusted to fit available resources for scalability.

Compute jobs record actionable process provenance about any file's genesis with a datalad (containers-)run command. This provenance includes analysis input, output, software, and commands. Computing in ephemeral workspaces ensures that it is complete and thus portable. Beyond transparency, this allows consumers to automatically rerun individual jobs on their own laptops using the *datalad rerun* command, and inspect recomputations' computational reproducibility.

Check recomputations for binary identity

```
Author:     Jane Doe <j.doe@fz-juelich.de>
AuthorDate: Wed Feb 10 18:05:30 2021 +0100

{   "chain": [],
    "cmd": "singularity exec -B {pwd} --cleanenv
                code/pipeline[...] sh -e -u -x -c  [...]'
    "dsid": "8938de76-0302-45b5-9825-3c6ce3f3fffe",
    "exit": 0,
    "extra_inputs": ["code/pipeline/.datalad/environments/cat/image"],
    "inputs": ["inputs/ukb/sub-6.../ses-2/anat/sub-6..._T1w.nii.gz",
     "code/cat_standalone_batch.txt",
     "code/finalize_job_outputs.sh"],
    "outputs": ["sub-6025043/ses-2"],
    "pwd": "."    }
```

Recompute your colleagues' multi-TB analysis - on your laptop!

Scalability test: UK-Biobank

Key workflow metrics:

41.180 T1-weighted brain images from the UKB imaging dataset were analyzed with subject-wise compute jobs. Each compute job:

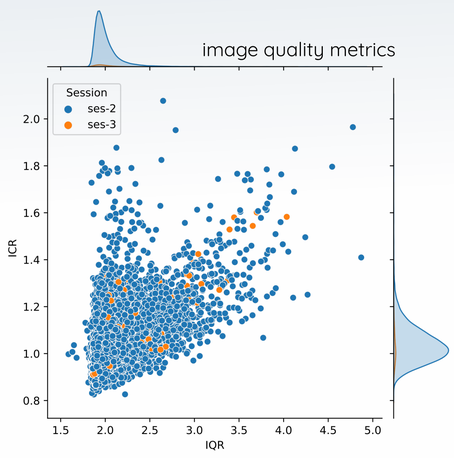
- processed one image for voxel-based morphometry[4] using a Singularity[5] container
- required one CPU hour
- needed 4 GB RAM
- created 4 tar archives, stripped from irrelevant sources of variance
- needed 5 GB disk space

Key computing metrics:

Low disk space **HTC**:
- up to 600 jobs at a time
- Processing time: 6 weeks

Inode-constrained **HPC**:
- 3125 jobs at a time
- Processing time: 10 hours

Recomputations yielded >50% binary identical results with the exception of cortical projections.



biobank[uk] application no. 41655

Code

Visualization

In the framework, datasets have a common representation (a directory tree) familiar to users, and an internal storage representation (a RIA store[3]) that can host a dataset of arbitrary size and number of files in fewer than 25 inodes, with minimal server-side requirements, & optional content compression and encryption.

Analyze datasets that exceed your computational resources

Storage representation (RIA store)

```
/data/store/
  893/8de76-0302-45b5-9825-3c6ce3f3fffe/
    branches/...
    config
    refs/...
    objects/...
    ...
  800/467ae-d521-11ea-8bf7-ac1f6bbc444e/
    archives/archive.7z
```

Common representation

```
sub-10..
ses-2/
  anat/ ...
  dwi/  ...
  func/ ...
  non-bids/ ...
    dMRI/ ...
    fieldmap/ ...
    fMRI/ ...
    SWI/ ...
    T1/ ...
    T2_FLAIR/ ...
  swi/ ...
ses-3/
  anat/ ...
  dwi/ ...
  func/ ...
  non-bids/ ...
...
```

```
# clone one participant dataset from the store
$ datalad clone ria+file:///store#800467ae-d521-11ea-8bf7-ac1f6bbc444e sub-10..
$ cd sub-10..
# retrieve all of the subjects data from the archives in
# the store
$ datalad get .
```

Funding

NSF 1912266
NSF 1229999

BMBF 01GQ1905
BMBF 01GQ1411

MOTOR SFB[FWF]

H2020-EU.3.1.5.3, 945539

H2020-EU.3.1.5.3, 826421

NATIONAL SCIENCE CENTRE POLAND

References

[1] Halchenko, Y. O. et al. (2021). doi: 10.21105/joss.03262
[2] Hanke et al. (2021) doi: 10.5281/zenodo.4773629
[3] Poldrack et al. (2021). doi: 10.7490/f1000research.1118494.1
[4] Gaser & Dahnke, http://www.neuro.uni-jena.de/cat
[5] Kurtzer et al. (2017). doi: 10.1371/journal.pone.0177459

Talk

Paper

Tutorial