DataLad – decentralized data distribution for consumption and sharing of scientific datasets

Yaroslav O. Halchenko¹, Michael Hanke²
¹ Dartmouth College, Hanover, NH, USA
² Otto-von-Guericke University, Magdeburg, Germany

CRCNS 2016, Paris, France 2016


Visit DataLad booth #4113 at SfN 2016, San Diego
Acknowledgments

Center for Open Neuroscience

Michael Hanke
University of Magdeburg, Germany
Formerly a visiting post-doctoral research at Dr. Haxby’s lab, now a J-Prof., one of the first Psychoinformaticians, official Debian developer, member of INCF neuroimaging task force -- he is an old-time collaborator and a lead of PyMVPA, NeuroDebian, DataLad and other projects.

Joey Hess
Independent Guru
joey’s own introduction "I'm joey Hess and I write programs" conceals his paramount role in establishing the core of the Debian distribution (debhelper, debian-installer, debconf, pristine-tar, etc.) and his work on variety of other software projects, such as git-annex which we rely upon in the DataLad project.

Benjamin Poldrack
University of Magdeburg, Germany
Works on the DataLad project.

Interns (Dartmouth)
Debanjum
Gergana
Houston, we’ve got a problem...

Data is a 2nd-class citizen within software platforms
Why?

- tarballs are **inefficient** distribution format
- **absent versioning** of data

*derived and/or curated data does change!*
Why?

- Tarballs are an inefficient distribution format absent versioning of data derived and/or curated data does change!
- Code version control systems are inadequate for data duplication, monolithic storage, etc. absent generic data distributions
- No efficient ways to install and upgrade cacophony of authorization schemes, interfaces, protocols
- Absent data testing data can and does have bugs (see e.g. Halchenko, 2012; Rohlfing, 2013)
- Difficulty to share new or derivative data shareable? Some is not! Where to host? How to "link" back?
Why?

- tarballs are **inefficient** distribution format
- **absent versioning** of data
  
  *derived and/or curated data does change!*
- code version control systems are **inadequate** for data
  
  *duplication, monolithic storage, etc.*
- **absent generic data distributions**
  
  *no efficient ways to install and upgrade*
- **cacophony** of authorization schemes, interfaces, protocols
- **absent data testing**
  
  *data can and does have bugs (see e.g. Halchenko, 2012; Rohlfing, 2013)*
- **difficulty to share** new or derivative data
  
  *shareable? some is not! where to host? how to “link” back?*
DataLad’s goal
Managing data should be as easy as managing code and software
Welcome datalad.org

Using Git …

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

DataLad aims to provide access to scientific data available from various sources (e.g. lab or consortium web-sites such as Human connectome; data sharing portals such as OpenFMRI and CRCNS) through a single convenient interface and integrated with your software package managers (such as APT in Debian). Although initially targeting neuroimaging and neuroscience data in general, it will not be limited by the domain and we would welcome a wide range of contributions.
DataLad demo1 - from search to get

Fri 21 Oct 2016 09:57:10 PM EDT:
$> 2 14868..........................:Fri 21 Oct 2016 09:57:10 PM EDT:
(git)hopa:~/.datalad[master]
$>  $
DataLad demo1 - from search to get

$> datalad search Haxby
labs/haxby
labs/haxby/raiders
openfmri/ds000105

$>
DataLad demo1 - from search to get

2 14868.................................:Fri 21 Oct 2016 09:57:10 PM EDT:
(git)hopa:~/datalad[master]
$> datalad search Haxby
labs/haxby
labs/haxby/raiders
openfmri/ds000105

2 14871.................................:Fri 21 Oct 2016 09:57:54 PM EDT:
(git)hopa:~/datalad[master]
$> datalad search Haxby | xargs datalad install
3 installed items are available at
<Dataset path=/home/yoh/datalad/labs/haxby>
<Dataset path=/home/yoh/datalad/labs/haxby/raiders>
<Dataset path=/home/yoh/datalad/openfmri/ds000105>

2 14872.................................:Fri 21 Oct 2016 09:58:14 PM EDT:
(git)hopa:~/datalad[master]
$>
DataLad demo1 - from search to get

File Edit View Search Terminal Help

(git)hopa:~/datalad[master]

$> datalad search Haxby
labs/haxby
labs/haxby/raiders
openfmri/ds000105

Fri 21 Oct 2016 09:57:54 PM EDT:

(git)hopa:~/datalad[master]

$> datalad search Haxby | xargs datalad install
3 installed items are available at
<Dataset path=/home/yoh/datalad/labs/haxby>
<Dataset path=/home/yoh/datalad/labs/haxby/raiders>
<Dataset path=/home/yoh/datalad/openfmri/ds000105>

Fri 21 Oct 2016 09:58:14 PM EDT:

(git)hopa:~/datalad[master]

$> cd /home/yoh/datalad/labs/haxby/raiders

Fri 21 Oct 2016 09:58:30 PM EDT:

hopa:~/datalad/labs/haxby/raiders

$>
datalad demo1 - from search to get

Fri 21 Oct 2016 10:00:20 PM EDT:
hopa:~/datalad/labs/haxby/raiders
$ ls sub001/anatomy/
highres004.par@  highres008.par@  scout001.par@
highres004.rec@  highres008.rec@  scout001.rec@
highres004.nii.gz@  highres008.nii.gz@  scout005.par@
highres004_defaced.nii.gz@  highres008_defaced.nii.gz@  scout005.rec@
highres004_defacemask.nii.gz@  highres008_defacemask.nii.gz@

Fri 21 Oct 2016 10:00:29 PM EDT:
hopa:~/datalad/labs/haxby/raiders
$
DataLad demo1 - from search to get

Fri 21 Oct 2016 10:00:20 PM EDT:

hopa:~/datalad/labs/haxby/raid
er

```bash
$ ls sub001/anatomy/
highres004.PAR
highres004.REC
highres004.nii.gz
highres004_defaced.nii.gz
highres004_defacemask.nii.gz

Fri 21 Oct 2016 10:00:29 PM EDT:

hopa:~/datalad/labs/haxby/raid
er

$ datalad get -J4 sub*/anatomy/*nii.gz

2016-10-21 22:00:46,030 [INFO ] Getting 58 items of dataset <Dataset path=/home/yoh/datalad/labs/haxby/raid
er> ...

Total (15 ok, 12 failed out of 58) 23% | 1 57.0M/253M [00:17<01:04, 3.06MB/s]
sub005/anat .. aced.nii.gz: 98% | 7.33M/7.50M [00:08<00:00, 726KB/s]
sub005/anat .. aced.nii.gz: 71% | 5.08M/7.12M [00:05<00:03, 624KB/s]
sub006/anat .. aced.nii.gz: 75% | 4.24M/5.66M [00:04<00:01, 850KB/s]
sub007/anat .. aced.nii.gz: 21% | 1.49M/6.99M [00:00<00:00, 6.18MB/s]
```
DataLad demo1 - from search to get

Fri 21 Oct 2016 10:01:25 PM EDT:

hopa:~/datalad/labs/haxby/raiderns

$ ls sub001/anatomy/

highres004.PAR@g highres008.PAR@g scout001.PAR@g
highres004.REC@g highres008.REC@g scout001.REC@g
highres004.nii.gz@ highres008.nii.gz@ scout005.PAR@g
highres004_defaced.nii.gz@ highres008_defaced.nii.gz@ scout005.REC@g
highres004_defacemask.nii.gz@ highres008_defacemask.nii.gz@

Fri 21 Oct 2016 10:01:44 PM EDT:

hopa:~/datalad/labs/haxby/raiderns

$
How: Foundation #1 – Git is

- a **version control system** initially developed to manage Linux project code
- **distributed** - content is available across all copies of the repository while allowing for aggregation of individual differences
- a backbone of **GitHub** and other **social coding** portals
- **very efficient** for managing textual information (code, text, configuration, etc.)
- **inefficient** for storing data
is built on top of Git

provides **access to data content** from variety of sources: HTTP, FTP, Webdav, bittorent, RSYNC, Amazon S3, *etc.*

allows for **custom extensions** to get access to offload data: Dropbox, Google drive, *etc.*

features optional Dropbox-like **synchronization** facility via *git-annex assistant*
How: Foundation #2 – Git-annex

- Is built on top of Git
- Provides access to data content from variety of sources: HTTP, FTP, Webdav, bitorrent, RSYNC, Amazon S3, etc.
- Allows for custom extensions to get access to offload data: Dropbox, Google drive, etc.
- Features optional Dropbox-like synchronization facility via git-annex assistant

Both Git and git-annex largely work on a single repository level
How: Foundation #2 – Git-annex

- **is built on top of Git**
- provides **access to data content** from variety of sources: HTTP, FTP, Webdav, bittorent, RSYNC, Amazon S3, etc.
- allows for **custom extensions** to get access to offload data: Dropbox, Google drive, etc.
- features optional Dropbox-like **synchronization** facility via **git-annex assistant**

Both Git and git-annex largely work on a single repository level. TBs of scientific data are out there in separate custom portals.
How #1+2=#3: DataLad

- comes with **command line and Python** interfaces
- supports both **git** and **git/annex** repositories
- manages **multiple repositories** organized into “super-datasets” using standard git sub-modules mechanism
- is **scalable** since data stays with original data providers
- unifies access to data regardless of its origin (custom portals with authentication, S3, *etc.*) or serialization (*e.g.*, tarballs)
- aggregates datasets’ **meta-data** and allows for quick **search**
- can **publish** original or derived datasets publicly (a web server, WiP: github) or for internal use (*e.g.* via ssh), while possibly keeping data available from elsewhere
- can **export** datasets (tarballs, WiP: ISA-TAB)
- can **crawl** external online data sources, and update git/annex repositories upon changes
Revision: 2.0.0  Date Set: May 24, 2016, 7:26 p.m.

Notes:

- Converted to BIDS standard.

Data Associated with Revision:

- Raw data on AWS

Revision: 1.1.0  Date Set: Feb. 18, 2016, 8:28 p.m.

Notes:

Updated orientation information in NIFTI headers for better left-right determination.

Data Associated with Revision:

- Raw data checksums
- Raw data on AWS

Revision: 1.0.0  Date Set: July 10, 2012, 8:28 p.m.

Data Associated with Revision:
To install this dataset in your current directory use

datalad install //openfmri/

To install with all subdatasets and all data

datalad install -r -g //openfmri/

For more information about DataLad and installation instructions visit datalad.org

<table>
<thead>
<tr>
<th>Name</th>
<th>Last Modified</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>./</td>
<td>2016-10-11 00:27:05</td>
<td>717.8 kB/1.3 TB</td>
<td>OpenfMRI (<a href="http://openfmri.org">http://openfmri.org</a>)</td>
</tr>
<tr>
<td>../</td>
<td>2016-09-22 15:31:00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ds000001/@@2.0.1</td>
<td>2016-09-19 19:18:54</td>
<td>2.8 kB/2.4 GB</td>
<td>Balloon Analog Risk Task</td>
</tr>
<tr>
<td>ds000002/@@2.0.1</td>
<td>2016-10-07 22:43:10</td>
<td>2.4 kB/2.9 GB</td>
<td>Classification learning</td>
</tr>
<tr>
<td>ds000003/@@2.0.1</td>
<td>2016-10-07 22:43:13</td>
<td>2.1 kB/413.5 MB</td>
<td>Rhyme judgment</td>
</tr>
<tr>
<td>ds000005/@2.0.0+3-2-gbed0245</td>
<td>2016-09-07 16:26:20</td>
<td>2.4 kB/1.9 GB</td>
<td>Mixed-gambles task</td>
</tr>
<tr>
<td>ds000006/@2.0.0+2-2-g8b1d5a</td>
<td>2016-09-07 16:26:20</td>
<td>2.4 kB/4.8 GB</td>
<td>Living-nonliving decision with plain or mirror-reversed text</td>
</tr>
</tbody>
</table>
DataLad WiP

Our growing “distribution”:

- http://datasets.datalad.org

Covered:

- http://openfmri.org (S3)
- http://crcns.org
- http://studyforrest.org etc.

Coming:

- http://humanconnectome.org (S3, XNAT)
- http://nitrc.org/ir (INDI, FCP, etc.)

Straight from the oven: MRI DICOM → DataLad BIDS

https://github.com/nipy/heudiconv/pull/32

Integration: NeuroDebian

apt-get install openfmri-ds000113
apt-get install openfmri
Summary: DataLad ...

- helps to manage and share available and your own data via a simple (command line or Python) interface
- helps with
  - authentication
  - crawling of websites with data resources
  - getting data from archives
  - publishing your new or derived data
- uses pure git/git-annex repositories under – power users can stay in power, and everything is version controlled
- makes meta-data *useful* to normal humans
- is ready for you to start using it, documentation is growing: www.datalad.org
Summary: DataLad ...

- helps to manage and share available and your own data via a simple (command line or Python) interface
- helps with
  - authentication
  - crawling of websites with data resources
  - getting data from archives
  - publishing your new or derived data
- uses pure git/git-annex repositories under – power users can stay in power, and everything is version controlled
- makes meta-data *useful* to normal humans
- is ready for you to start using it, documentation is growing:  
  www.datalad.org

Managing data can be similar to managing code and software
Brain Download:

ęż completes.
Thank you!

For more information visit

Website: datalad.org
Github: github.com/datalad
Twitter: @datalad (I am @yarikoptic, Michael is @eknahm)
SfN 2016: Booth #4113

References

DataLad data distribution: Data life cycle

Data hosting

- S3openfmri
- HTTP
- XNAT
- OpenfMRI
- ds112:
- ds113:
- ds113: sub001:
- ds113: BOLD:
- ds113: task001_run001:
- ds112:
- ds113_raw_6.tgz

Server or cloud storage

- ds113:
- ds112:
- HTTP
- S3
- sub001/
- BOLD/
- task001_run001:
- bold.nii.gz

Data content

Git content

Content reference

DataLad

Data user

Local workstation
DataLad data distribution: Data life cycle

Data user

Local workstation

Data distribution

DataLad

Git repository hosting (GitHub)

Data content

Content reference

Git content

Data content

Data hosting

Server or cloud storage

OpenfMRI

ds112:

ds113:

ds113_raw_6.tgz

HTTP

XNAT

XNAT

http

HTTP

Data content

Content reference

DataLad
DataLad data distribution: Data life cycle
DataLad data distribution: Data life cycle

Dataset handles
Content reference
Data content
Git content
DataLad
DataLad data distribution: Data life cycle

1. **Data hosting**
   - OpenfMRI
   - XNAT

2. **Data distribution**
   - datasets.datalad.org
   - Git repository hosting (GitHub)

3. **Data user**
   - Local workstation

4. **Data content**
   - $HOME

- **Dataset handles**
- **Content reference**
- **Data content**
- **Git content**
- **DataLad**

**Commands**
- `datalad add bold_filt.nii.gz`
- `datalad install -d . -g //openfmri/ds000113`
- `datalad create`
- `datalad crawl`

**Data**
- `studyX/ds000113/
  .git/
  .datalad/crawl/
  README
  sub001/
  BOLD/
  task001_run001/
  bold.nii.gz`
- `ds000113:
  ds112:
  ds113:
  ds113_raw_6.tgz`
- `ds113:
  sub001:
  BOLD:
  task001_run001:
  bold.nii.gz`
- `ds112:
  ...`
- `studyX/ds000113/
  .git/
  .gitmodules
  ds000112/
  ds113:
  ds112:
  HTTP
  S3
  bold.nii.gz`
- `studyX/ds000113/
  .git/
  .gitmodules
  ds113:
  ds113_raw_6.tgz
  ds113:
  ds112:
  ...`
- `HTTP`
- `S3`
- `datasets.datalad.org/
  .git/
  .gitmodules`
- `studyX/ds000113/
  .git/
  .gitmodules
  ds000113/
  ds113:
  ds112:
  ... other_ds/`
DataLad data distribution: Data life cycle
DataLad data distribution: Data life cycle

- **Data producer**: OpenfMRI
- **Data hosting**
  - S3
  - XNAT
  - Other cloud storage
- **Data distribution**
  - datasets.datalad.org
  - github.com/user1
- **Data user**
  - $HOME
- **Git repository hosting (GitHub)**
  - studyX
  - studyX--other_ds
- **Local workstation**
  - studyX
  - studyX/ds000113
  - studyX--ds000113
  - studyX--other_ds

**Data content**
- bold.nii.gz
- bold_filt.nii.gz

**Git content**
- .git
- .gitmodules
- .git/annex
- .git/crawl/
- README

**Content reference**
- studyX/ds000113
- studyX--ds000113
- studyX--other_ds
- datasets.datalad.org
- github.com/user1

**Dataset handles**
- ds112:
- ds113:
- ds113_raw_6.tgz
- sub001:
- BOLD:
- task001_run001:
- bold.nii.gz
- ...
AutomagicIO: automatically fetch necessary files

Given Python code which accesses files within annex repository (example from PyMVPA):

```python
verbose(1, "Loading data...")
filepath = os.path.join(cfg.get('location', 'tutorial data'),
                       'hyperalignment_tutorial_data.hdf5.gz')
ds_all = h5load(filepath)
# zscore all datasets individually
_ = [zscore(ds) for ds in ds_all]
# inject the subject ID into all datasets
for i,sd in enumerate(ds_all):
    sd.sa['subject'] = np.repeat(i, len(sd))
# number of subjects
nsubjs = len(ds_all)
# number of categories
ncats = len(ds_all[0].UT)
# number of run
nruns = len(ds_all[0].UC)
verbose(2, "%d subjects" % len(ds_all))
verbose(2, "Per-subject dataset: %i samples with %i features" % ds_all[0].shape)
verbose(2, "Stimulus categories: %s" % ','.join(ds_all[0].UT))
```
AutomagicIO: automatically fetch necessary files

DataLad can automatically fetch necessary load whenever specific file is requested:

```
2 5329................................................:Thu 23 Jun 2016 12:39:11 PM CEST:..
(git)hopa:/tmp/PyMVPA[master]
$> datalad install -s http://data.pymvpa.org/datasets/tutorial_data /tmp/tutorial_data
1 installed item is available at
<Dataset path=/tmp/tutorial_data>
2 5329................................................:Thu 23 Jun 2016 12:39:13 PM CEST:..
(git)hopa:/tmp/PyMVPA[master]
$> MVPA_LOCATION_TUTORIAL_DATA=/tmp/tutorial_data python -m datalad doc/examples/hyperalignment.py
Loading data...
/tmp/tutorial_data/.git 100%[====================================] 15.04M --.-KB/s in 0.02s
10 subjects
Per-subject dataset: 56 samples with 3509 features
Stimulus categories: Chair, DogFace, FemaleFace, House, MaleFace, MonkeyFace, Shoe
Performing classification analyses...
within-subject... done in 1.2 seconds
between-subject (anatomically aligned)... done in 0.6 seconds
between-subject (hyperaligned)... done in 3.3 seconds
Average classification accuracies:
within-subject: 0.57 +/-0.053
between-subject (anatomically aligned): 0.42 +/-0.035
between-subject (hyperaligned): 0.62 +/-0.050
```
DataLad’s testing

All is well — 9 successful checks

- datalad-pr-virtualbox-dl-wln7-64 — DEV build done.
- datalad-pr-docker-dl-nd80 — DEV build done.
- datalad-pr-docker-dl-nd14_10 — DEV build done.
- datalad-pr-docker-dl-nd70 — DEV build done.
- datalad-pr-docker-dl-nd14_04 — DEV build done.
- datalad-pr-docker-dl-nd90 — DEV build done.
- continuous-integration/travis-ci/pr — The Travis CI build passed
- coverage/coveralls — Coverage increased (+0.18%) to 83.88%
- datalad-pr-dl-osx-64 — DEV build done.

This pull request can be automatically merged.
You can also merge branches on the command line.