



Open Science Tools from the Center for Open Neuroscience

Yaroslav O. Halchenko
@yarikoptic @centeropenneuro
Dartmouth College, Hanover, NH, USA

August 2019



<http://datalad.org>



<http://www.pymvpa.org>



<http://Neuro.Debian.net>



A Center for Reproducible
Neuroimaging Computation

Slides



[http://neuro.debian.net/_files/
mind2019-contools-halchenko.pdf](http://neuro.debian.net/_files/mind2019-contools-halchenko.pdf)

It is **YOU** not the tools who can make Science Open!

· Jul 6, 2018



Replies to [@yarkoptic](#)

Oh my all this unsolicited advise here.. why? Apparently you seem to know things better?

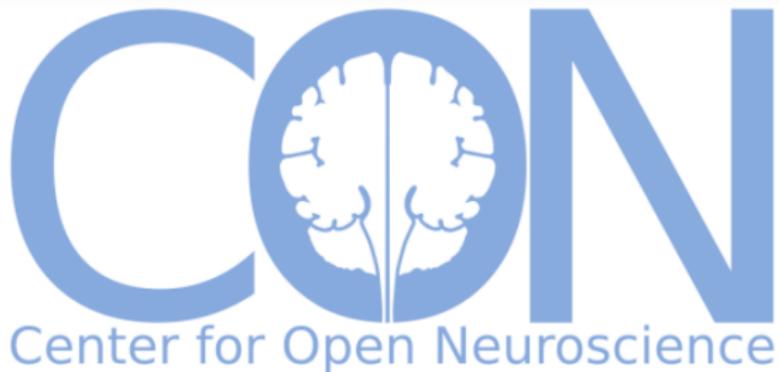




**Overarching goal:
To make neuroscience a better science**

Center for Open Neuroscience

PROJECTS WHO WE ARE SUPPORT CONTACT



provides open software frameworks, platforms, data and methodologies for neuroscience and beyond

Our principles

Open Source

Open source is not only the most efficient paradigm for scalability and collaboration, it facilitates verification and reproducibility.

We do not only advocate open source principles, but actively consult scientific developers on open source licensing and development practices.

Quality Assurance

Research developers are often not programmers by training. As a result, scientific tools are often insufficiently tested and inherently broken.

We work on exercising existing and establishing new unit-, regression, integration and validity testing to guarantee robust and correct operation. Reproducible bogus results are useless!

Re-Use & Integration

Scientific community is blooming with bright minds doing great research and sharing powerful software solutions and data collections.

Instead of (re)inventing a new "better" wheel we instead develop products to harness, improve, and integrate existing solutions.

Convenience

Never was a buzz word (yet), but it is one of the critical drivers summarizing reliability, accessibility etc. Only when software and data are conveniently available we can talk about wide-scale adoption and reproducibility.

We work to make software and data available to the community, so that everyone could harness available resources regardless of their original complexity or size.

Dissemination

Scientific software is developed by enthusiasts, who neither have facilities nor funds to support large-scale promotion or reliable distribution.

We help to promote and distribute software, data, and related documentation while paying special attention to such aspects as intellectual property and due acknowledgment.

Community

Neuroscience community is vast, but scientific projects often pursued in isolation and without interaction with other scientific communities.

We participate in many initiatives and projects that cross borders among different communities and disciplines, and in return make bridges among previously disconnected research groups.

Who we are/Acknowledgments

① centerforopenneuroscience.org/whowear#michael_hanke_

Center for Open Neuroscience

Michael Hanke

Centroids

Collaborators

Michael Hanke
Nikolaus N. Oosterhof
Matthew Brett
Joey Hess
Benjamin Poldrack

Emeritus

Collaborating projects

Partners



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.

Students/Interns

Benjamin Kyle



Jason

Alex



Chris



Adina



Sam



Matteo



Debanjum



Gergana



Oliver



Who we are/Common suspects

C H [centerforopenneuroscience.org/whowear#affiliated_fa...](https://centerforopenneuroscience.org/whowear#affiliated_faculty) 🔍 ⭐

 Center for Open Neuroscience

PROJECTS WHO WE ARE ENGAGE SUPPORT

Affiliated Faculty

Centroids

Collaborators

- Michael Hanke
- Nikolaas N. Oosterhof
- Matthew Brett
- Joey Hess
- Benjamin Poldrack

Affiliated Faculty

Collaborators in training

Emeritus

Collaborating projects

Partners

Luke Chang Jeremy Rothman Manning Matthijs (Matt) van der Meer



Who we are/That was only a tip of an iceberg

centerforopenneuroscience.org/whowear...@collaborating_projects...

Center for Open Neuroscience

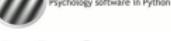
PROJECTS WHO WE ARE ENGAGE SUPPORT

Centroids

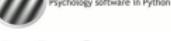
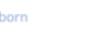
Yaroslav O. Halchenko
James V. Haxby
Matteo Visconti di Oleggio Castello
Samuel Nastase

Collaborators

Michael Hanke
Nikolaas N. Oosterhof
Matthew Brett
Joey Hess
Benjamin Poldrack

Collaborating projects

Partners

SPONSORED BY THE  Federal Ministry of Education and Research

 International Neuroinformatics Coordinating Facility

 The source for neuroinformatics tools & resources

 Neuroimaging data repository

 Cloud computing environment

Methodology: “Open by Design”

The screenshot shows a web browser window with the URL centerforopenneuroscience.org. The page title is "Center for Open Neuroscience". The navigation menu includes links for "PROJECTS", "WHO WE ARE", "SUPPORT", and "CONTACT". The main content area is titled "References" and lists two academic papers:

- Halchenko, Y. O. and Hanke, M. (2015). Four aspects to make science open “by design” and not as an after-thought. *GigaScience*, 4. [\[PDF\]](#) DOI: 10.1186/s13742-015-0072-7
- Halchenko, Y. O. & Hanke, M. (2012). Open is not enough. Let's take the next step: An integrated, community-driven computing platform for neuroscience. *Frontiers in Neuroinformatics*, 6:22. [\[PDF\]](#) DOI: 10.3389/fninf.2012.00022

A blue vertical bar on the left side contains the following text:

3. Innovation:
The effort here matches, if it does not exceed, Friston's brilliancy many years ago in envisioning SPM as a cross-platform language for communication of research results in a standard format.

— *Anonymous reviewer of the (not funded) NIH grant submission for the NeuroDebian project.*

Center for Open Neuroscience

Together we can make neuroscience a better science!

Center

Contact

Who we are

Stay in touch

Center For Open Neuroscience is not directly affiliated with COS (Center For Open Science).

Website content is copyright of respective authors and released under CC BY 3.0 license. Website design is based on buddycloud.com, released under Apache 2.0 license.

“Open by Design” 101.1

- Prepare to share even if you might not!
 - **Use** instead of **Disregard** legal mechanisms
(Copyright, License, Participants Consent)
 - Your work will stay “Open” to future yourself, your colleagues, students, etc.

"Open by Design" 101.2

- **Invest** your time at the **beginning** to do it "right", to save 10-100x in the (near) future.
 - Keep learning and using new tools, frameworks, approaches, etc. to make yourself more efficient and keep a good record of actions
 - E.g. learn your shell <http://www.repronim.org/module-reproducible-basics/01-shell-basics/>
 - Automate (and script it) whenever possible

"Open by Design" 101.2

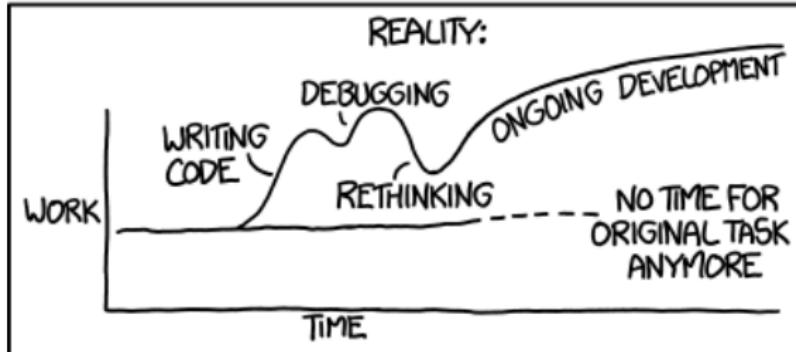
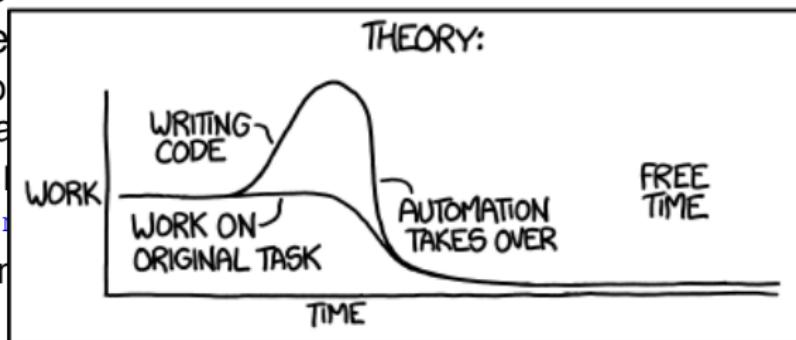
- Invest yo
in the (ne
■ Keep
to ma
■
■ Autom

"I SPEND A LOT OF TIME ON THIS TASK.
I SHOULD WRITE A PROGRAM AUTOMATING IT!"

ave 10-100x

baches, etc.
d of actions

cs /



“Open by Design” 101.2

- **Invest** your time at the **beginning** to do it “right”, to save 10-100x in the (near) future.
 - Keep learning and using new tools, frameworks, approaches, etc. to make yourself more efficient and keep a good record of actions
 - E.g. learn your shell <http://www.repronim.org/module-reproducible-basics/01-shell-basics/>
 - Automate (and script it) whenever possible
- **Do not reinvent the wheel – use and contribute to existing projects**
 - Learn better coding and testing practices
 - (re)Use already established development, release, and deployment infrastructure
 - Share/offload “user-support”
 - Get credit faster (Zenodo, duecredit)

“Open by Design” 101.3

- See something, say something!
 - **Report** instead of **Ignore** errors
 - Try to make your bug reports “reproducible”:
[http://www.repronim.org/
module-reproducible-basics/05-misc/](http://www.repronim.org/module-reproducible-basics/05-misc/)

5 Steps to More Reproducible Neuroimaging Research

Going whole-hog into completely reproducible neuroimaging is hard. But, there are lots of little things you can do today to increase the reproducibility of your current studies.

Study Design

[repronim.org/5Steps/design](https://www.repronim.org/5Steps/design)

- Build data shareability into the consent process - see [Open Brain Consent](#), for example.
- Plan your study with a RoadMap - see, for example, [BrainVerse](#).
- Pre-register your study - at, for example, [Open Science Foundation](#).

Data Collection

[repronim.org/5Steps/acquisition](https://www.repronim.org/5Steps/acquisition)

- Adopt standards-based data representation from the get go.
- Use [Reproin](#) to automate conversion of your imaging data to [BIDS](#).
- Annotate your metadata (all experimental details) as you collect it - see, for example, [BrainVerse](#).
- Use a version control system for all files (data, code, environments, etc.) - i.e. [DataLad](#).

Data Processing

[repronim.org/5Steps/processing](https://www.repronim.org/5Steps/processing)

- Document your software and computational environment - for example [NeuroDocker](#) and [NiceMan](#).
- Use containerization/virtualization to encapsulate and share analysis workflow ([Docker](#), [Singularity](#), [Virtual Machines](#), etc.).
- Annotate your results - using NeuroImaging Data Model ([NIDM](#)), for example.

Statistical Analysis

[repronim.org/5Steps/stats](https://www.repronim.org/5Steps/stats)

- Most studies are underpowered, combat this by sharing your data and reusing available data to enable creating larger data sets, thereby increasing study power. Understand:
 - Effect size, confidence intervals, power, positive predictive value and significance testing - some training materials can be found [here](#).

Publication

[repronim.org/5Steps/publication](https://www.repronim.org/5Steps/publication)

- Include the anonymized raw data and workflow used - using resources such as [zenodo](#), etc.
- Include complete processed results
- Make it Findable, Accessible, Interoperable and Reusable ([FAIR](#))

Problem for open sharing awaits from the beginning

Neuroimaging data must not be shared without explicit consent from the participants!

Why should I care (an example)?

① crcns.org/forum/using-datasets/219722707

150%

..

Vim-1 dataset availability

[^ Up to Using data sets](#)

Posted by [Furkan Ozcelik](#) at January 17, 2018

Hi,



I have an intention to use vim-1 dataset in my undergraduate project but I see currently it is not available for downloading. Admin's last response to situation is approximately 3 months ago and there he has said that it will be available in a few weeks. Will vim-1 dataset be available soon or is there any other option to download dataset currently?

Thanks, Have a good day

Posted by [admin](#) at January 22, 2018



The reason this data is not available is that questions were raised about whether sharing the data complied with United States government rules. It's been taking much longer than expected to resolve this. An update will be posted here when the data is made available again (which could be soon, but might not be). There is not another option to download the data.

Open Brain Consent

YOUR LOGO CONTRIBUTION HERE

open-brain-consent.readthedocs.io

Consents: Federally regulated but locally “managed”

The screenshot shows a web browser window with the URL <https://open-brain-consent.readthedocs.io/en/stable/>. The page title is "Open Brain Consent stable". The main content is titled "Make open data sharing a no-brainer for ethics committees." It includes a section titled "Statement of the problem" and discusses the challenges of managing neuroimaging data under federal regulations.

Docs » Make open data sharing a no-brainer for ethics committees. [Edit on GitHub](#)

Make open data sharing a no-brainer for ethics committees.

Statement of the problem

The ideology of open and reproducible science makes its ways into various fields of science. Neuroimaging is a driving force today behind many fields of brain sciences. Despite possibly terabytes of neuroimaging data collected for research daily, just a small fraction becomes publicly available. Partially it is because management of neuroimaging data requires to conform to established legal norms, i.e. addressing the aspect of subjects privacy. Those norms are usually established by institutional review boards (IRB, or otherwise called ethics committees), which are in turn "governed" by the federal regulations, such as [45 Code of Federal Regulations Part 46](#) in US.

Flexibility in interpretation of original regulations established in the past century, decentralization of those committees, and lack of a "community" influence over them created the problem: for neuroimaging studies there is no commonly accepted version of a Consent form template which would allow for collected imaging data to be shared as openly as possible while providing adequate guarantees for subjects' privacy. In majority of the cases, used Consent forms simply do not include any provision for public sharing of the data to get a "speedy" IRB approval for a study. Situation is particularly tricky because major granting agencies (e.g. NIH, NSF) nowadays require public data sharing, but do not provide explicit instructions on how.

Open Brain Consent: Sample consent forms

The screenshot shows a web browser window with the URL <https://open-brain-consent.readthedocs.io/en/latest/sample/>. The page title is "Sample consent forms". On the left, there's a sidebar with links to "Recommendations", "Discussions", "Ultimate consent form", "Anonymization tools", "Contribute", and "Contact information". The main content area contains a heading "Sample consent forms" and a paragraph explaining that the "samples/" directory contains samples of consent forms found online, noting that explicit permission for re-distribution is not granted. It also mentions that the repository is a git annex repository. Below this, it says "And here you can find a list of those files contained under samples/:" followed by a bulleted list of file names.

Docs » Sample consent forms [Edit on GitHub](#)

Sample consent forms

[samples/](#) directory of our [git repository](#) contains samples of the consent forms found online. Because there is no explicit permission allowing their re-distribution we are not including them in this repository/site, but rather link to them as they are available on the web. Our [git repository](#) is also a [git annex](#) repository so you should be to [git annex get](#) any file of interest, if it is still available online.

And here you can find a list of those files contained under [samples/](#):

- [Arizona_consent.pdf](#)
- [CMU_fmri-consent-v-april-201011.doc](#)
- [Dartmouth-fMRI-Consent-Template.doc](#)
- [GIN_consent-fr.pdf](#)
- [NMR_MGH_samplefMRIconsent.html](#)
- [UCB_SpatialRep_MRI.pdf](#)
- [UCLA_sample_consent.html](#)
- [UK_cf_CUBRIC_InfoConsentDebrief_fMRIonly.doc.html](#)
- [UK_gla_fmri_study_consent_form_0820110.doc](#)
- [USC_Informed-Consent-Template-3-29-13-FMRI.doc](#)
- [psychLMU_ConsentForm_Template_Dyads_German.pdf](#)
- [psychLMU_ConsentForm_Template_NonDyads_German.pdf](#)
- [psychLMU_ConsentForm_Template_easy_German.pdf](#)

Open Brain Consent: Ultimate consent form

The screenshot shows a browser window displaying the "Ultimate consent form" page from the "Open Brain Consent" documentation. The sidebar on the left contains navigation links such as "Search docs", "Sample consent forms", "Recommendations", "Discussions", and "Ultimate consent form". Under "Ultimate consent form", there are two sections: "Single access type version (all data shared publicly; recommended)" and "Two access types version (some data shared publicly, more data shared to approved researchers)". The main content area displays the "Ultimate consent form" itself, which is described as a merged version of existing forms and consulting with experts in research ethics. It includes sections for "Single access type version" (with translations in English, German, French, Chinese, Spanish, and Italian) and "Two access types version". The "Single access type version" section states that data and samples might be used for future research projects and shared publicly via the Internet and a database. The "Two access types version" section states that data will be shared with the general public using a code number, and no personal information will be included. The main content also includes a note about changing consent and withdrawing data.

Ultimate consent form

The following consent form has been put together, by merging best parts of existing consent forms and consulting with experts in research ethics.

Single access type version (all data shared publicly; recommended)

English

The data and samples from this study might be used for other, future research projects in addition to the study you are currently participating in. Those future projects can focus on any topic that might be unrelated to the goals of this study. We will give access to the data we are collecting, including the imaging data, to the general public via the Internet and a fully open database.

The data we share with the general public will not have your name on it, only a code number, so people will not know your name or which data are yours. In addition, we will not share any other information that we think might help people who know you guess which data are yours.

If you change your mind and withdraw your consent to participate in this study (you can call <PI name> at <phone number> to do this), we will not collect any additional data about you. We will delete your data if you withdraw before it was deposited in the database. However, any data and research results already shared with other investigators or the general public cannot be destroyed, withdrawn or recalled.

Open Brain Consent: Tools

The screenshot shows a web browser displaying the "Anonymization tools" page from the "Open Brain Consent" documentation. The sidebar on the left includes links for "Sample consent forms", "Recommendations", "Discussions", "Ultimate consent form", and sections for "Anonymization tools" (selected), "Sanitization of headers/filenames" (selected), "Elimination of facial (and dental) features" (selected), "Skull stripping", "Faces/dental stripping", "Rendering faces unrecognizable", "Contribute", and "Contact information". The main content area has a heading "Anonymization tools" and a sub-section "Sanitization of headers/filenames" with a bullet list. It also contains sections for "Elimination of facial (and dental) features", "Skull stripping", and "Faces/dental stripping".

Anonymization tools

Sanitization of headers/filenames

- see http://www.researchgate.net/post/Best_free_tool_for_DICOM_data_anonymization discussion on sanitization of DICOM headers
- [DeID \(see paper\)](#), which provides an interactive tool for inspection and sanitization of Analyze and NIfTI images
- [PyDICOM's deid](#), the "best effort anonymization for medical images using python" assists in filtering out DICOM fields and also masking out actual image data

Elimination of facial (and dental) features

Skull stripping

One of the approaches is perform complete skull stripping, e.g. using

- [BET of FSL](#)
- [3dSkullStrip of AFNI](#)
- [FreeSurfer](#)

Some dedicated anonymization tools work on this principle, e.g. [DeID](#)

Faces/dental stripping

More "gentle" approach is to strip out only the areas of face/mouth leaving skull, which might be important for some types of analysis. Usually achieved through alignment of pre-crafted mask to the subject anatomy and removing of the masked out regions.

Get involved!

CONTRIBUTE

- Use it!
- Contribute a Logo!
- It is fully present on GitHub:
<https://github.com/datalad/open-brain-consent>,
(look for **good-for-hackathon** labeled issues)
- Submit fixes and/or translations
- Recommend tools and sample consent forms
- Spread the word

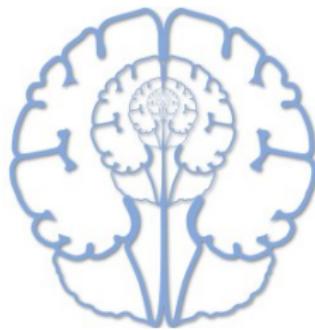
Oh come on, not yet another “data problem”!

BIDS (Brain Imaging Data Structure)
is the next great thing after NIfTI
but

I have no time to prepare those “BIDS datasets”!

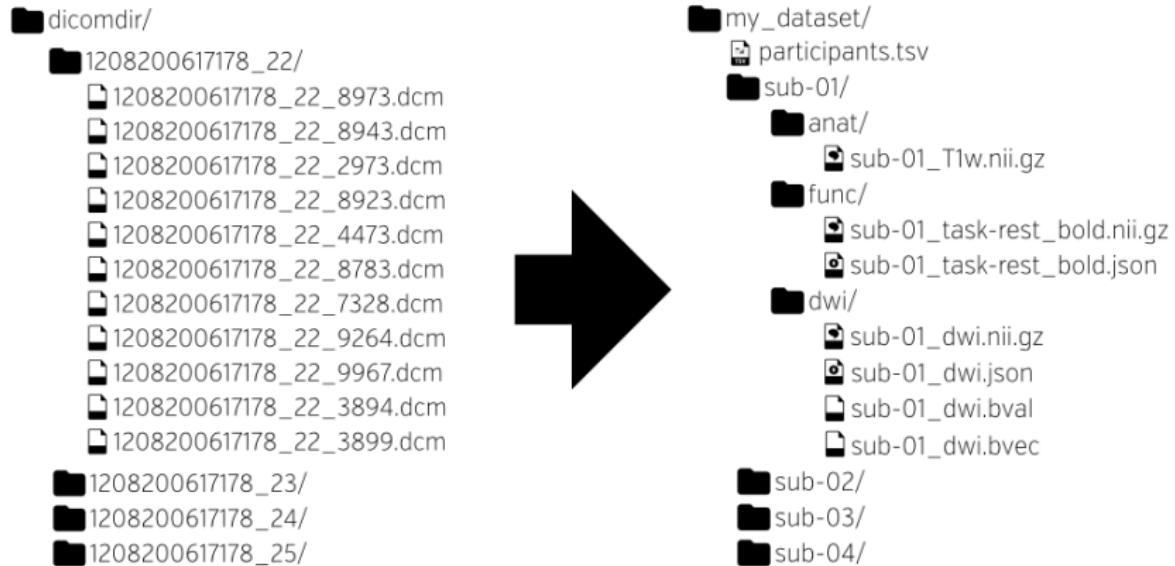
Gorgolewski, K. J., Auer, T., Calhoun, V. D., Craddock, R. C., Das, S., Duff, E. P., Flandin, G., Ghosh, S. S., Glatard, T., Halchenko, Y. O., Handwerker, D. A., Hanke, M., Keator, D., Li, X., Michael, Z., Maumet, C., Nichols, B. N., Nichols, T. E., Pellman, J., Poline, J.-B., Rokem, A., Schaefer, G., Sochat, V., Triplett, W., Turner, J. A., Varoquaux, G., and Poldrack, R. A. (2016). The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. *Scientific Data*, 3:160044

ReproIn (Reproducible Input)



reproin.repronim.org

BIDS: A directory/files structure for neuroimaging



BIDS Benefits

You have seen one BIDS dataset – you have seen them all!

- BIDS is both human- and machine- friendly
- BIDS compliance could be automatically verified using bids-validator
- PyBIDS, matlab-bids, etc. can assist scripting use of BIDS datasets
- BIDS-apps (such as mriqc, fmriprep, etc) provide a turnkey solution for BIDS datasets

github.com/INCF/bids-validator

github.com/INCF/pybids, github.com/bids-standard/bids-matlab

bids-apps.neuroimaging.io - main catalog, Docker images

github.com/ReproNim/containers - DataLad dataset with Singularity images

ReproIN: “BIDS” at the scanner console

Scanner

The Scanner interface shows a sidebar with a search bar and a list of scan sessions:

- Gobbini
- ▼ Gobbini_Matteo
 - ▶ 1002_face-angles
 - ▼ 1017_famface-angles
 - ses-famfirst
 - ses-strfirst
 - ▶ 1037_budapest
 - ▶ 1038_hyperface
- ▶ Gobbini_Vassiki

... ➤ ses-strfirst	
anat-scout_ses-strfirst	00:14
	AutoAlign Scout
anat_T2w	03:23
	...
fmap_acq-2.5mm	02:12
	...
func_run-01_task-str1back	06:00
	...
func_run-02_task-fam1back	06:00
	...
func_run-03_task-str1back	06:00
	...
func_run-04_task-fam1back	06:00
	...
func_run-05_task-str1back	06:00
	...
func_run-06_task-localizer	06:56
	...

BIDS

anat
sub-sid000005_ses-strfirst_T2w.json

DICOM

001-anat-scout_ses-strfirst
005-anat_T2w
000001.dcm
000002.dcm
...
006-fmap_acq-2.5mm
007-fmap_acq-2.5mm
008-func_run-01_task-str1back
011-func_run-01_task-str1back
018-func_run-02_task-fam1back
025-func_run-03_task-str1back
032-func_run-04_task-fam1back
039-func_run-05_task-str1back
046-func_run-06_task-localizer

\$ heudiconv
Data
ladd

\$ heudiconv


func_run-05_task-str1back	06:00
func_run-06_task-localizer	06:56

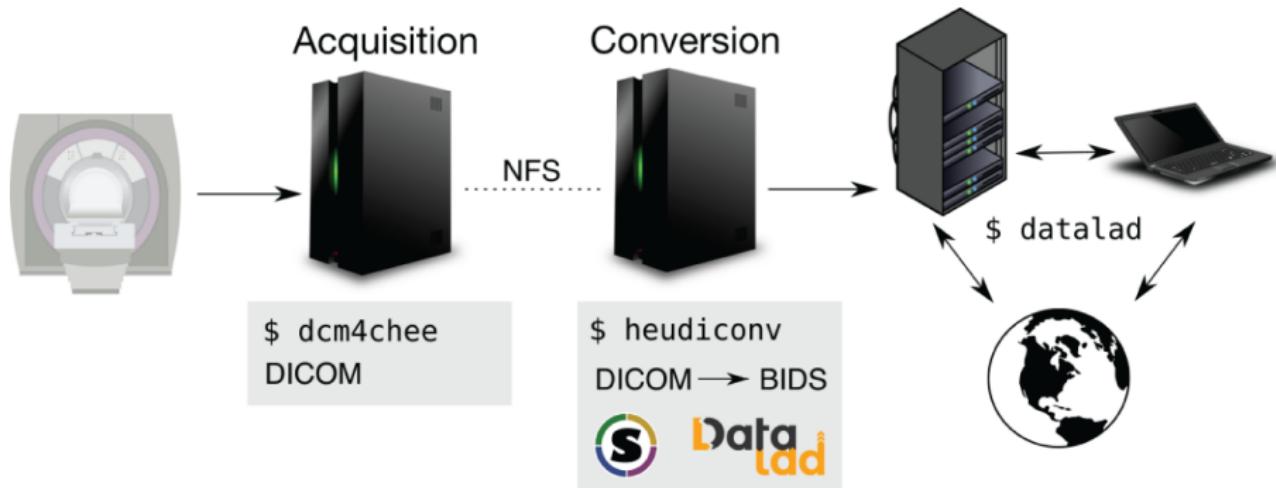
BIDS

```
anat
  sub-sid000005_ses-strffirst_T2W.json
  sub-sid000005_ses-strffirst_T2W.nii.gz
fmap
  sub-sid000005_ses-strffirst_acq-25mm_magnitude1.json
  sub-sid000005_ses-strffirst_acq-25mm_magnitude1.nii.gz
  ...
func
  sub-sid000005_ses-strffirst_task-fam1back_run-02_bold.json
  sub-sid000005_ses-strffirst_task-fam1back_run-02_bold.nii.gz
  sub-sid000005_ses-strffirst_task-fam1back_run-02_events.tsv
  ...
  sub-sid000005_ses-strffirst_scans.tsv
```

```
$ git grep TODO
```

```
CHANGES: TODOs:
README: TODO: Provide description for the dataset ...
dataset_description.json: "Acknowledgements": "TODO...",
dataset_description.json: "TODO:",
dataset_description.json: "DatasetDOI": "TODO: ..."
task-fam1back_bold.json: "CogAtlasID": "TODO",
task-fam1back_bold.json: "TaskName": "TODO: full task name",
task-localizer_bold.json: "CogAtlasID": "TODO",
task-localizer_bold.json: "TaskName": "TODO: full task name",
task-str1back_bold.json: "CogAtlasID": "TODO",
task-str1back_bold.json: "TaskName": "TODO: full task name",
  ...
```

ReproIN: Pile of DICOMs → BIDS



ReproIN: Benefits

- Minimal single time investment of adhering to sequence naming convention
 - convert at will
 - catch problems with acquisition early (`bids-validator`)
- All datasets within center organized into a hierarchy reflecting hierarchy at the scanner console
(no more “*where that RA buried original data?*”)
- Sidecar .json files in BIDS contain “useful” DICOM fields
(no more of “*I’ve lost the notes about slice order*”)
- DICOM files are retained under `sourcedata/`
(easy to re-convert if needed)
- All data (optionally) are maintained under distributed version control system (DataLad, more on that one later) to facilitate
 - incremental updates
 - collaboration
 - orchestration of data flow across computing infrastructure

Get involved!

CONTRIBUTE

- Collect DICOMs, not NIfTIs or PAR/RECs
- Use ReproIN:
 - for new studies adhere to the naming convention at the scanner
 - use Heudiconv with provided ReproIN heuristic
- Use Heudiconv!
 - for old studies come up with your own heuristic to map to BIDS, or
 - let's work together and map from your naming into ReproIN naming
- Fully present on GitHub:
 - <https://github.com/nipy/heudiconv>,
 - <https://github.com/repronim/reproin>
 - Submit fixes and/or translations
 - Look at [good-for-hackathon](#) labeled issues
- Spread the word



Houston, we've got a problem (in 2007 and beyond)

There is no *standard* software for data-driven analysis
of neural data

Secret sauce: no code == no reproducibility



PyMVPA
<http://www.pymvpa.org>

Princeton, Anno 2007



PyMVPA features

- User-centered *programmability*
 - ⇒ Concise scripting interface in Python and command line
- Thoroughly documented
- Extensible
- Portable
- Reliable
- Free and open source software

Hanke, M., Halchenko, Y. O., Sederberg, P. B., Hanson, S. J., Haxby, J. V., and Pollmann, S. (2009a). PyMVPA: A Python toolbox for multivariate pattern analysis of fMRI data. *Neuroinformatics*, 7(1):37–53. PMC2664559

PyMVPA features

- User-centered *programmability*
 - ⇒ Concise scripting interface in Python and command line
- Thoroughly documented
 - ⇒ Tutorial, user manual, and examples (IPython notebooks)
- Extensible

- Portable

- Reliable

- Free and open source software

Hanke, M., Halchenko, Y. O., Sederberg, P. B., Hanson, S. J., Haxby, J. V., and Pollmann, S. (2009a). PyMVPA: A Python toolbox for multivariate pattern analysis of fMRI data. *Neuroinformatics*, 7(1):37–53. PMC2664559

PyMVPA features

- User-centered *programmability*
 - ⇒ Concise scripting interface in Python and command line
- Thoroughly documented
 - ⇒ Tutorial, user manual, and examples (IPython notebooks)
- Extensible
 - ⇒ Modular architecture connecting extensions
in multiple languages
- Portable
- Reliable
- Free and open source software

Hanke, M., Halchenko, Y. O., Sederberg, P. B., Hanson, S. J., Haxby, J. V., and Pollmann, S. (2009a). PyMVPA: A Python toolbox for multivariate pattern analysis of fMRI data. *Neuroinformatics*, 7(1):37–53. PMC2664559

PyMVPA features

- User-centered *programmability*
 - ⇒ Concise scripting interface in Python and command line
- Thoroughly documented
 - ⇒ Tutorial, user manual, and examples (IPython notebooks)
- Extensible
 - ⇒ Modular architecture connecting extensions
in multiple languages
- Portable
 - ⇒ Runs on anything from mainframes to cell phones
- Reliable
- Free and open source software

Hanke, M., Halchenko, Y. O., Sederberg, P. B., Hanson, S. J., Haxby, J. V., and Pollmann, S. (2009a). PyMVPA: A Python toolbox for multivariate pattern analysis of fMRI data. *Neuroinformatics*, 7(1):37–53. PMC2664559

PyMVPA features

- User-centered
 - Thoroughly tested
 - Extensible
 - Portable
 - Reliable
 - Free and open source software
- 
- command line
notebooks)
extensions
languages
cell phones

Trautmann, E., Ray, L., and Lever, J. (2009). Development of an autonomous robot for ground penetrating radar surveys of polar ice. In *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 1685–1690

PyMVPA features

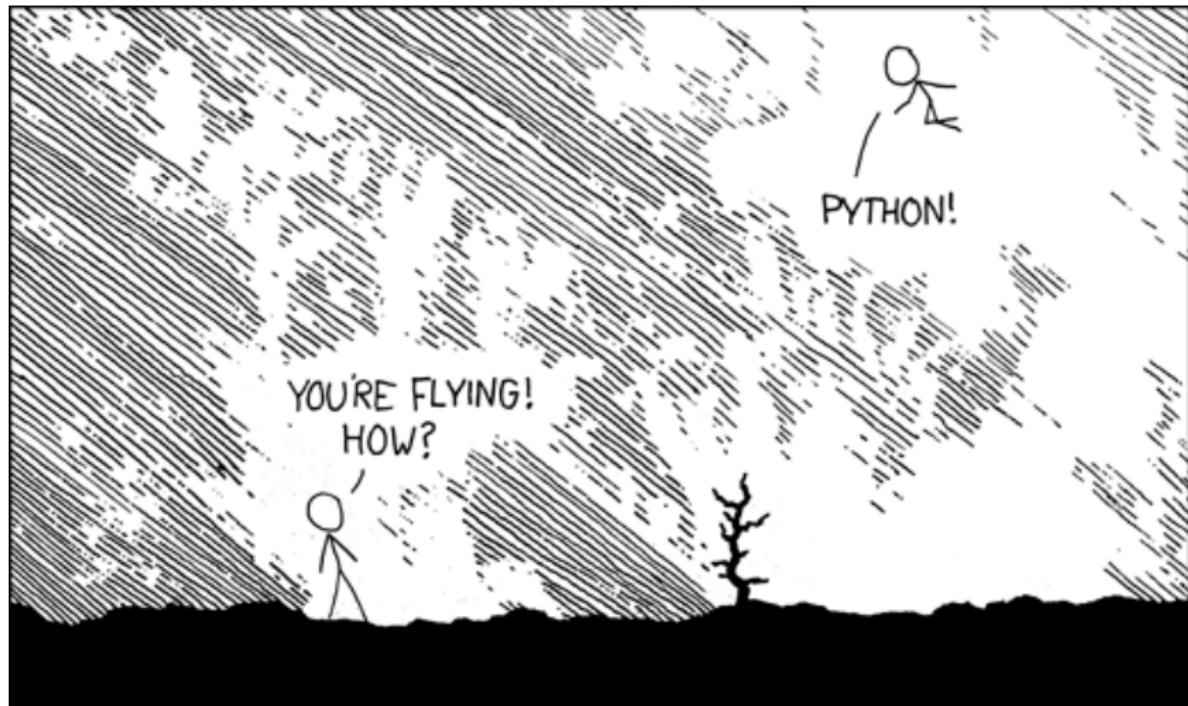
- User-centered *programmability*
 - ⇒ Concise scripting interface in Python and command line
- Thoroughly documented
 - ⇒ Tutorial, user manual, and examples (IPython notebooks)
- Extensible
 - ⇒ Modular architecture connecting extensions
in multiple languages
- Portable
 - ⇒ Runs on anything from mainframes to cell phones
- Reliable
 - ⇒ Unit-, doc-, example- tests
- Free and open source software

PyMVPA features

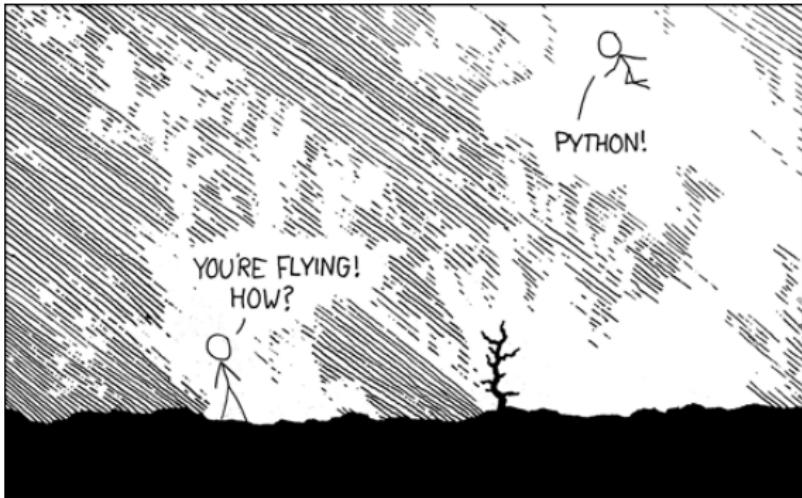
- User-centered *programmability*
 - ⇒ Concise scripting interface in Python and command line
- Thoroughly documented
 - ⇒ Tutorial, user manual, and examples (IPython notebooks)
- Extensible
 - ⇒ Modular architecture connecting extensions in multiple languages
- Portable
 - ⇒ Runs on anything from mainframes to cell phones
- Reliable
 - ⇒ Unit-, doc-, example- tests
- Free and open source software
 - ⇒ MIT-licensed

Hanke, M., Halchenko, Y. O., Sederberg, P. B., Hanson, S. J., Haxby, J. V., and Pollmann, S. (2009a). PyMVPA: A Python toolbox for multivariate pattern analysis of fMRI data. *Neuroinformatics*, 7(1):37–53. PMC2664559

Python



Python



I LEARNED IT LAST
NIGHT! EVERYTHING
IS SO SIMPLE!
/ HELLO WORLD IS JUST
print "Hello, world!"

I DUNNO...
DYNAMIC TYPING?
WHITESPACE?
/ COME JOIN US!
PROGRAMMING
IS FUN AGAIN!
IT'S A WHOLE
NEW WORLD
UP HERE!
BUT HOW ARE
YOU FLYING?

I JUST TYPED
import antigravity
THAT'S IT? /
/ ... I ALSO SAMPLED
EVERYTHING IN THE
MEDICINE CABINET
FOR COMPARISON.
/ BUT I THINK THIS
IS THE PYTHON.

Analysis example: “Import antigravity”

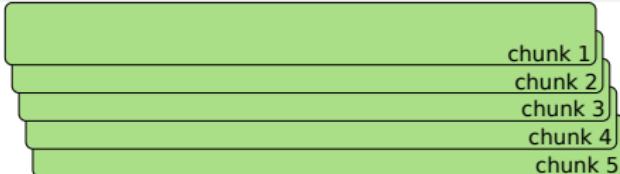
```
from mvpa2.suite import *
```

Analysis example : Datasets

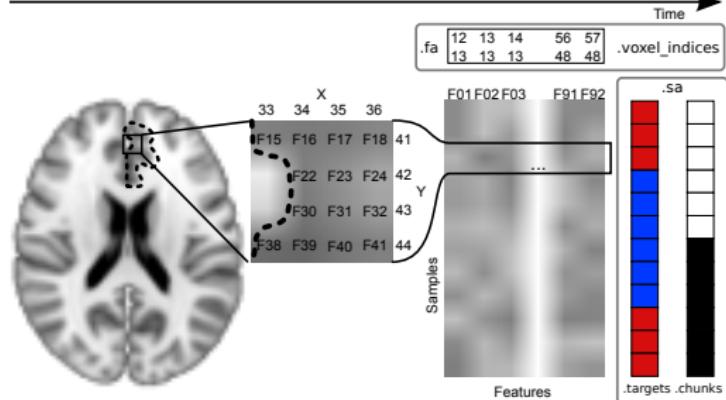
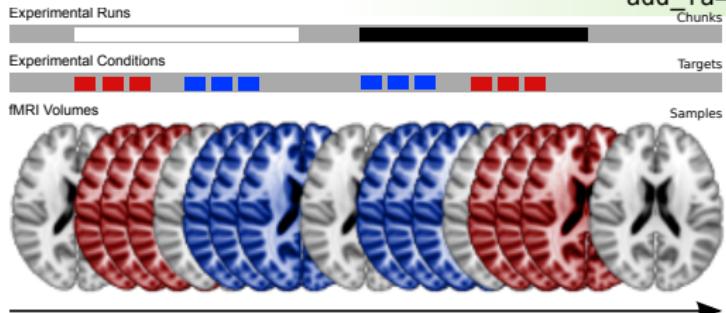
```
chunk 1  
chunk 2  
chunk 3  
chunk 4  
chunk 5
```

```
attr = SampleAttributes('attributes.txt')  
ds = fmri_dataset(samples='bold.nii.gz',  
                   targets=attr.targets,  
                   chunks=attr.chunks,  
                   mask='mask_brain.nii.gz',  
                   add_fa={'vt':'vt.nii.gz'})
```

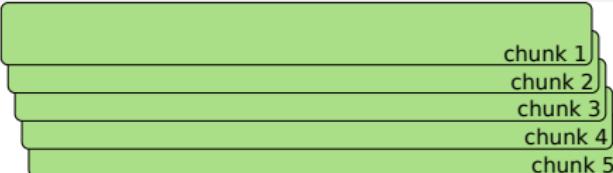
Analysis example : Datasets



```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                   targets=attr.targets,
                   chunks=attr.chunks,
                   mask='mask_brain.nii.gz',
                   add_fa={'vt':'vt.nii.gz'})
```



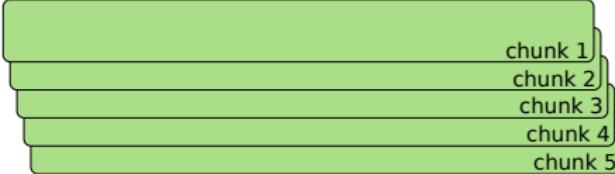
Analysis example : Datasets



```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                   targets=attr.targets,
                   chunks=attr.chunks,
                   mask='mask_brain.nii.gz',
                   add_fa={'vt':'vt.nii.gz'})

ds = ds[:, ds.fa.vt == 1]
```

Analysis example : Datasets



chunk 1
chunk 2
chunk 3
chunk 4
chunk 5

```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                   targets=attr.targets,
                   chunks=attr.chunks,
                   mask='mask_brain.nii.gz',
                   add_fa={'vt':'vt.nii.gz'})

print dataset.summary()
```

Analysis example : Datasets

chunk 1
chunk 2
chunk 3
chunk 4
chunk 5

```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                   targets=attr.targets,
                   chunks=attr.chunks,
                   mask='mask_brain.nii.gz',
                   add_fa={'vt':'vt.nii.gz'})
```

```
with dataset_summary()
```

```
Dataset / int16 81 x 577
uniq: 3 chunks 3 labels
stats: mean=1670.84 std=344.597 var=118747 min=430 max=2707
```

```
Counts of labels in each chunk:
chunks\labels bottle cat chair
    ---  ---  ---
0.0      9      9      9
1.0      9      9      9
2.0      9      9      9
```

```
Summary per label across chunks
label mean std min max #chunks
bottle  9   0   9   9   3
cat     9   0   9   9   3
chair   9   0   9   9   3
```

```
Summary per chunk across labels
chunk mean std min max #labels
0      9   0   9   9   3
1      9   0   9   9   3
2      9   0   9   9   3
```

Analysis example: Classification

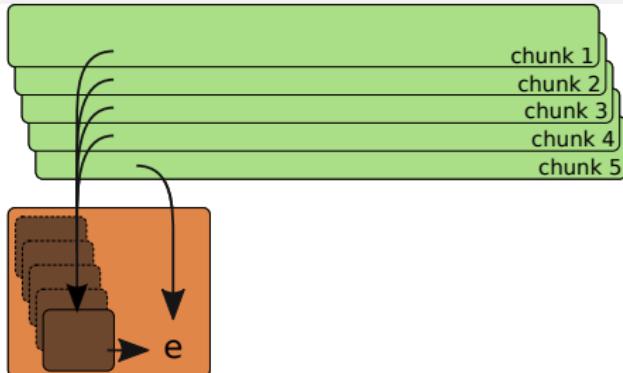
chunk 1
chunk 2
chunk 3
chunk 4
chunk 5

```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                   targets=attr.targets,
                   chunks=attr.chunks,
                   mask='mask_brain.nii.gz',
                   add_fa={'vt':'vt.nii.gz'})
```

```
clf = LinearCSVMC()
```

Clf

Analysis example: Classification



```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                   targets=attr.targets,
                   chunks=attr.chunks,
                   mask='mask_brain.nii.gz',
                   add_fa={'vt':'vt.nii.gz'})
```

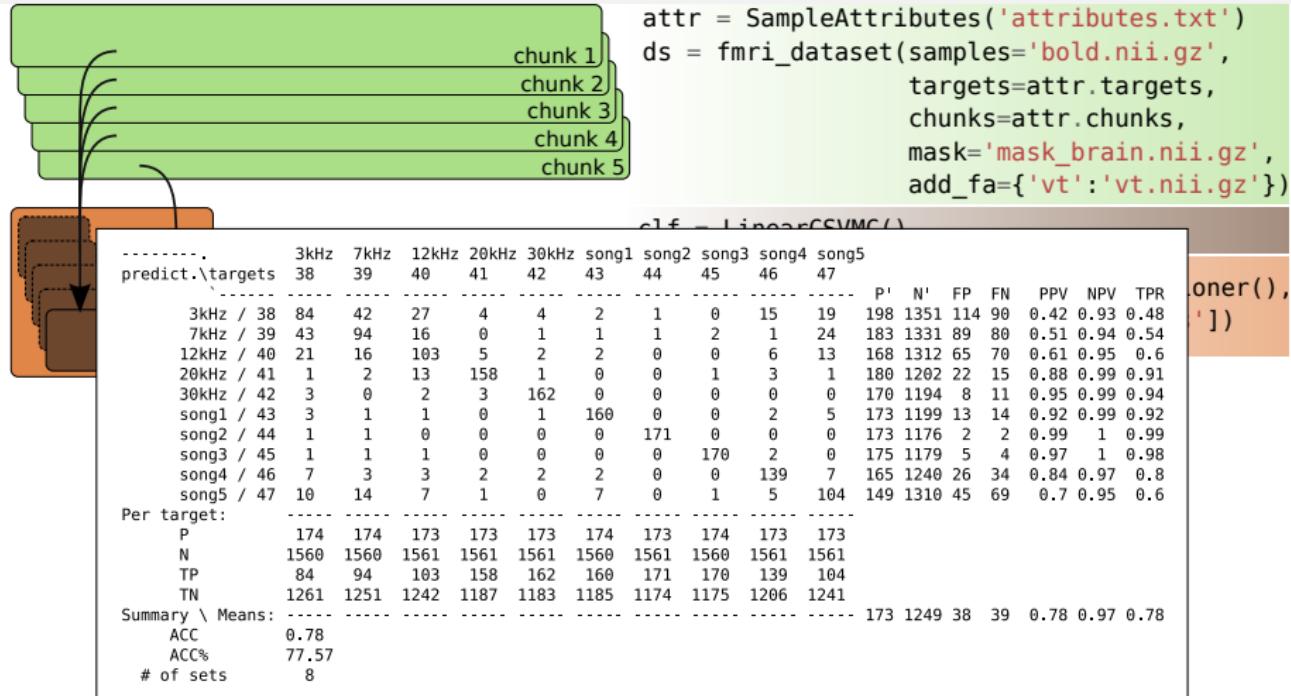
```
clf = LinearCSVMC()
```

```
cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])
```

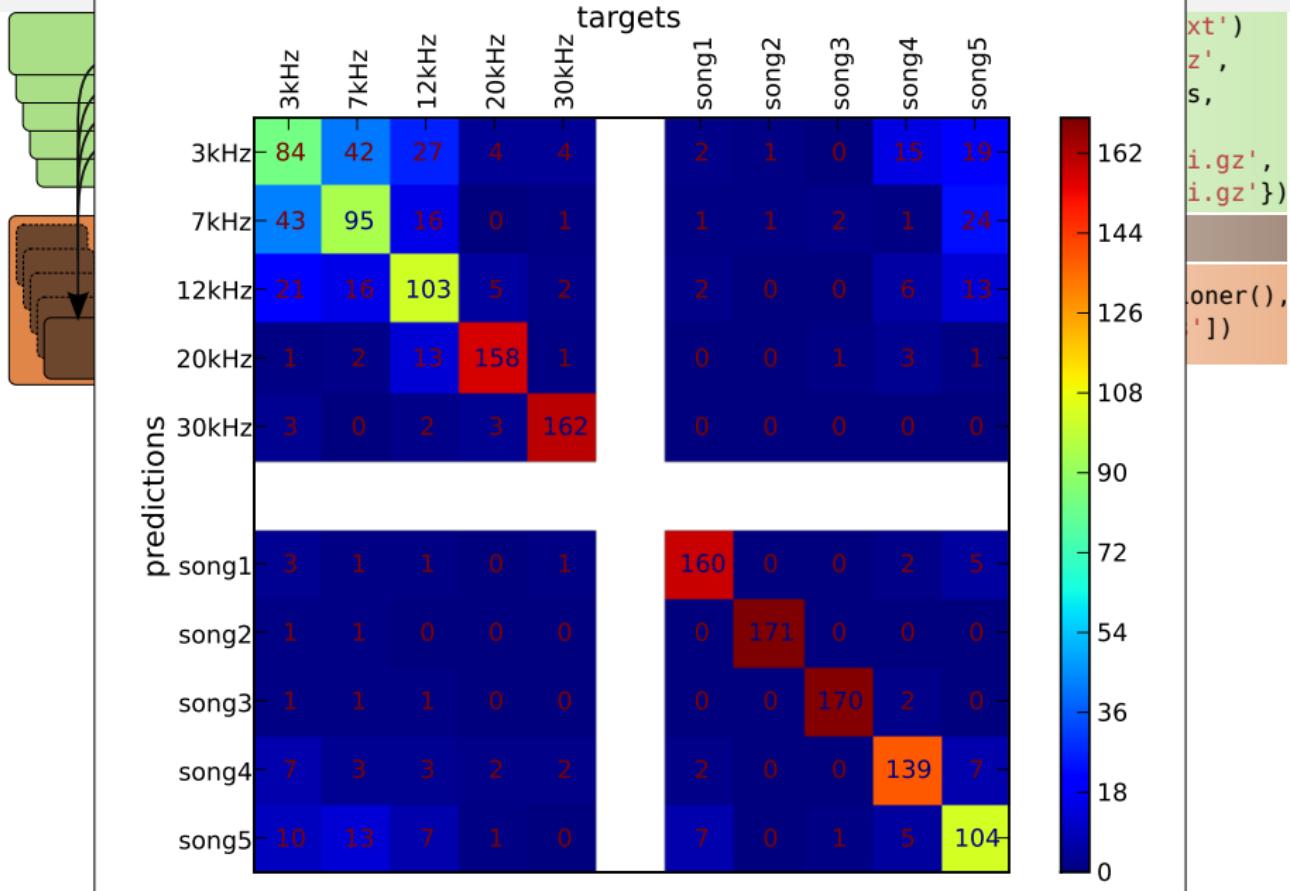
```
errors = cv(ds)
```

```
print cv.ca.stats
cv.ca.stats.plot()
```

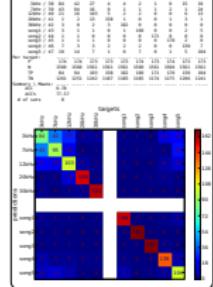
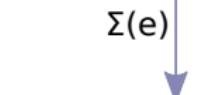
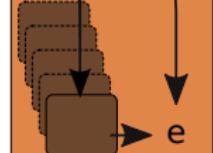
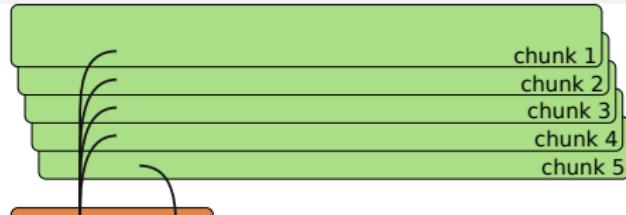
Analysis example: Classification (Everyone matters)



Analysis example: Classification



Analysis example: Classification



```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                   targets=attr.targets,
                   chunks=attr.chunks,
                   mask='mask_brain.nii.gz',
                   add_fa={'vt':'vt.nii.gz'})
```

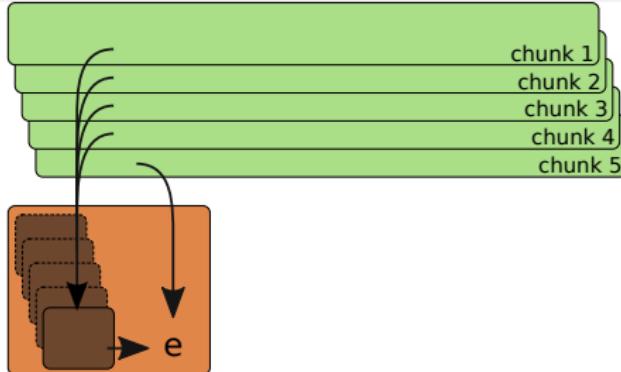
```
clf = LinearCSVMC()
```

```
cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])
```

```
errors = cv(ds)
```

```
print cv.ca.stats
cv.ca.stats.plot()
```

Analysis example: Searchlights

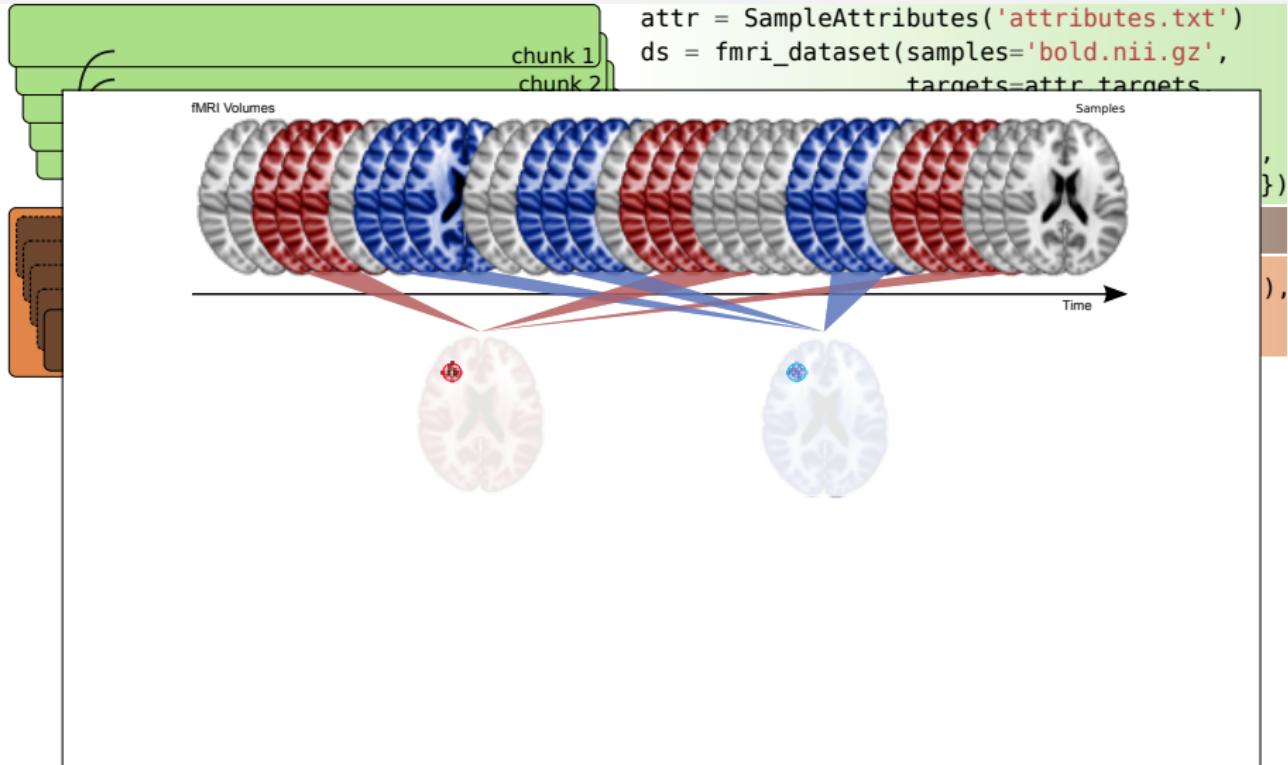


```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                   targets=attr.targets,
                   chunks=attr.chunks,
                   mask='mask_brain.nii.gz',
                   add_fa={'vt':'vt.nii.gz'})

clf = LinearCSVMC()

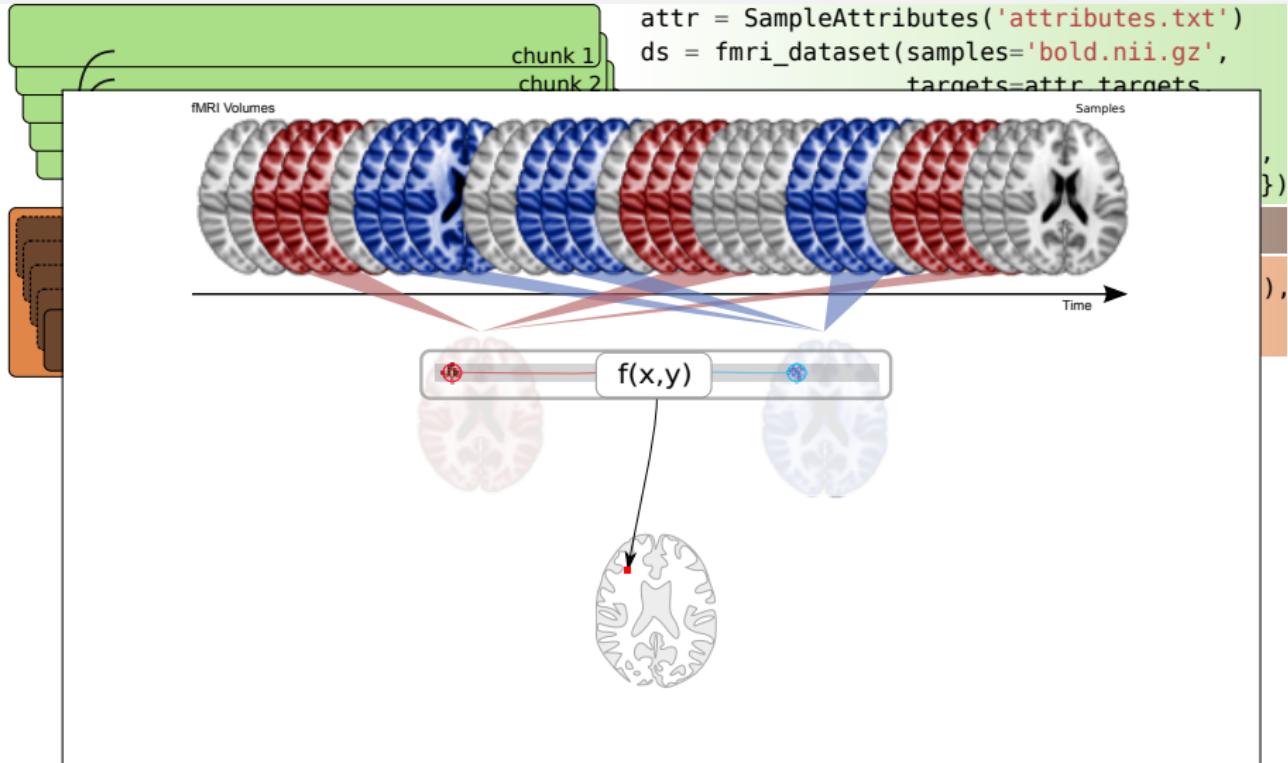
cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])
```

Analysis example: Searchlights



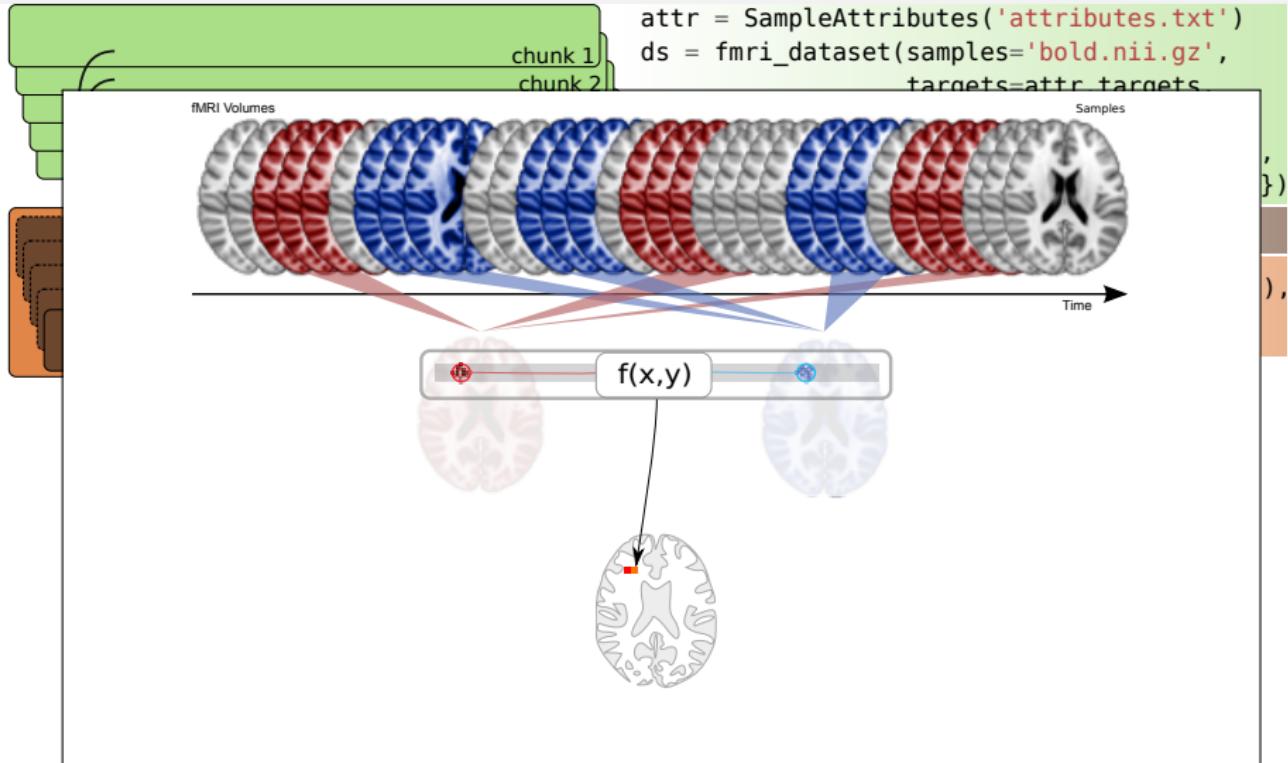
Kriegeskorte, N., Goebel, R., and Bandettini, P. (2006). Information-based functional brain mapping. *Proceedings of the National Academy of Sciences of the USA*, 103:3863–3868

Analysis example: Searchlights



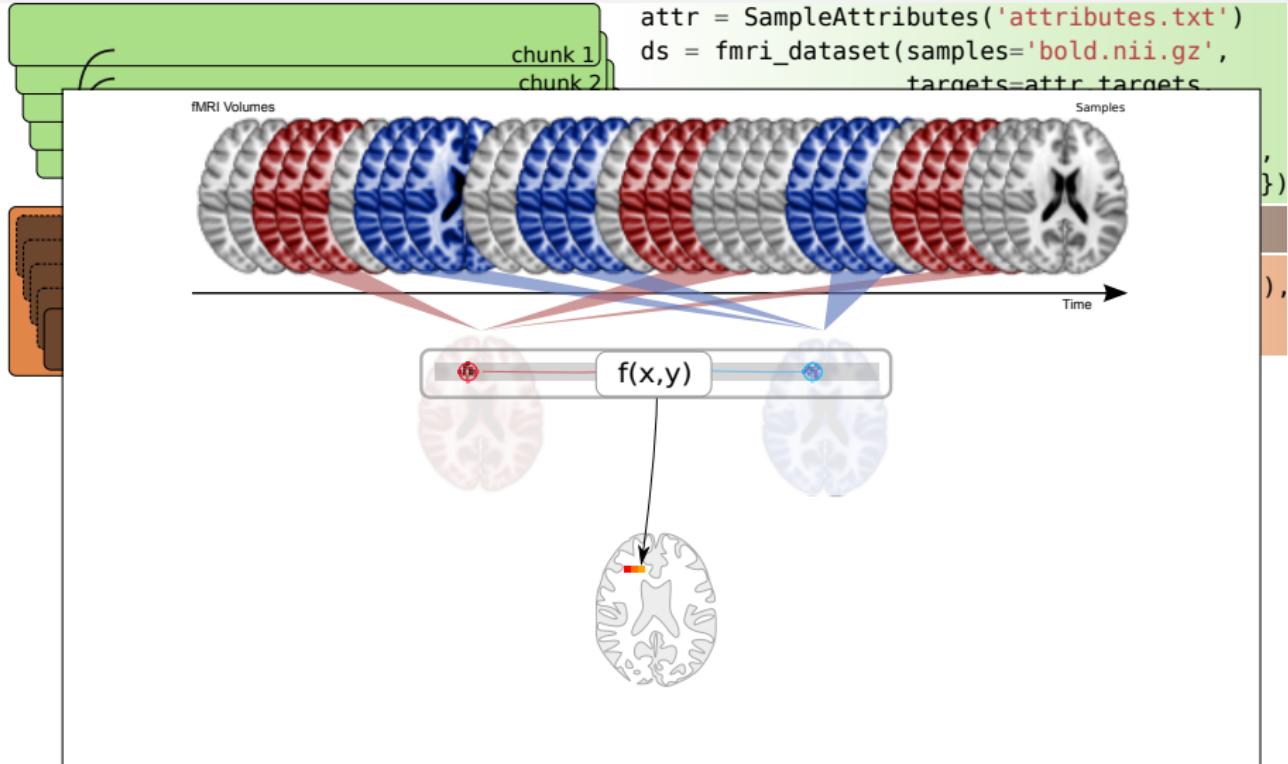
Kriegeskorte, N., Goebel, R., and Bandettini, P. (2006). Information-based functional brain mapping. *Proceedings of the National Academy of Sciences of the USA*, 103:3863–3868

Analysis example: Searchlights



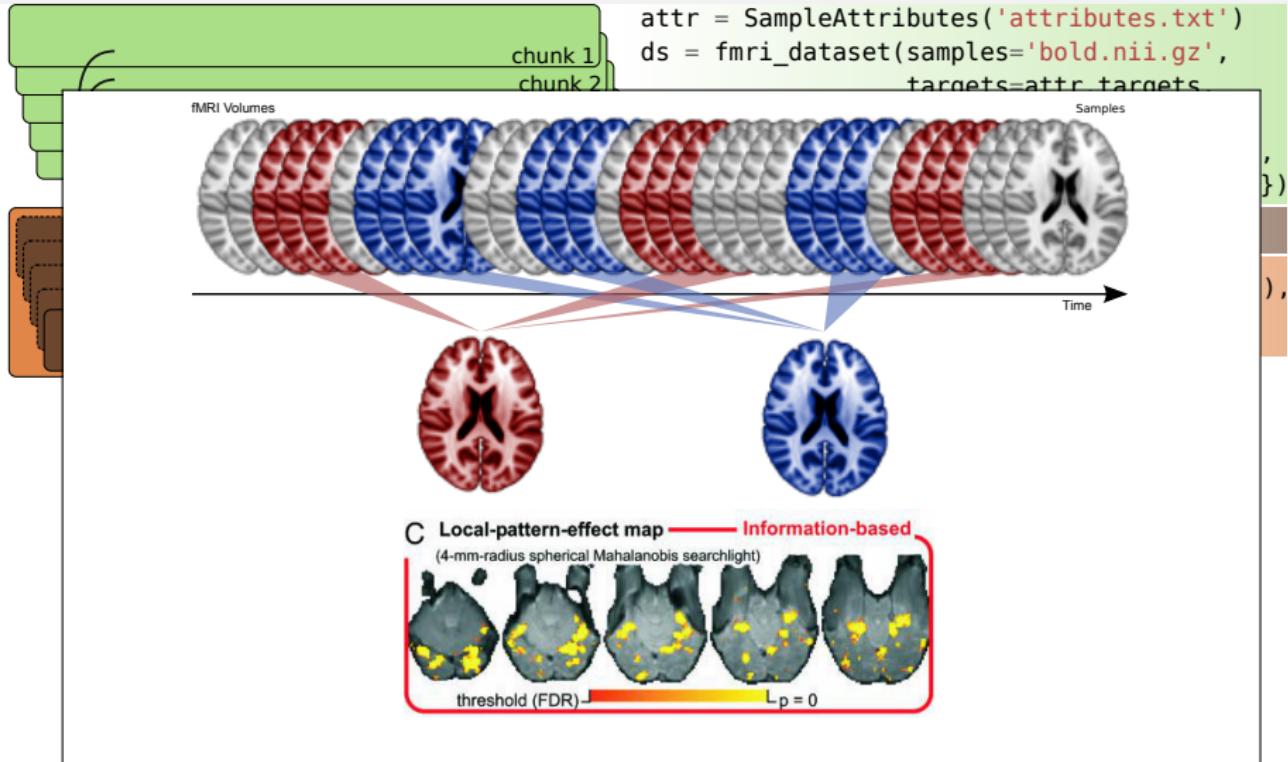
Kriegeskorte, N., Goebel, R., and Bandettini, P. (2006). Information-based functional brain mapping. *Proceedings of the National Academy of Sciences of the USA*, 103:3863–3868

Analysis example: Searchlights



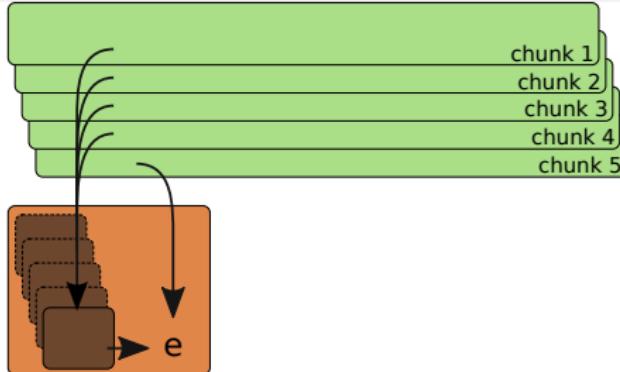
Kriegeskorte, N., Goebel, R., and Bandettini, P. (2006). Information-based functional brain mapping. *Proceedings of the National Academy of Sciences of the USA*, 103:3863–3868

Analysis example: Searchlights



Kriegeskorte, N., Goebel, R., and Bandettini, P. (2006). Information-based functional brain mapping. *Proceedings of the National Academy of Sciences of the USA*, 103:3863–3868

Analysis example: Searchlights

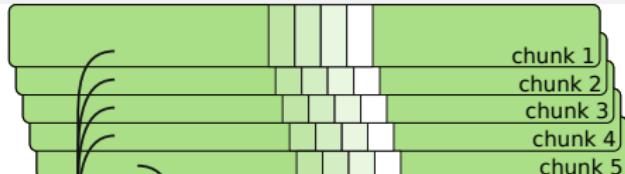


```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                   targets=attr.targets,
                   chunks=attr.chunks,
                   mask='mask_brain.nii.gz',
                   add_fa={'vt':'vt.nii.gz'})

clf = LinearCSVMC()

cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])
```

Analysis example: Searchlights



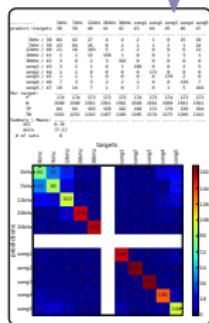
```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                   targets=attr.targets,
                   chunks=attr.chunks,
                   mask='mask_brain.nii.gz',
                   add_fa={'vt':'vt.nii.gz'})
```

```
clf = LinearCSVMC()
```

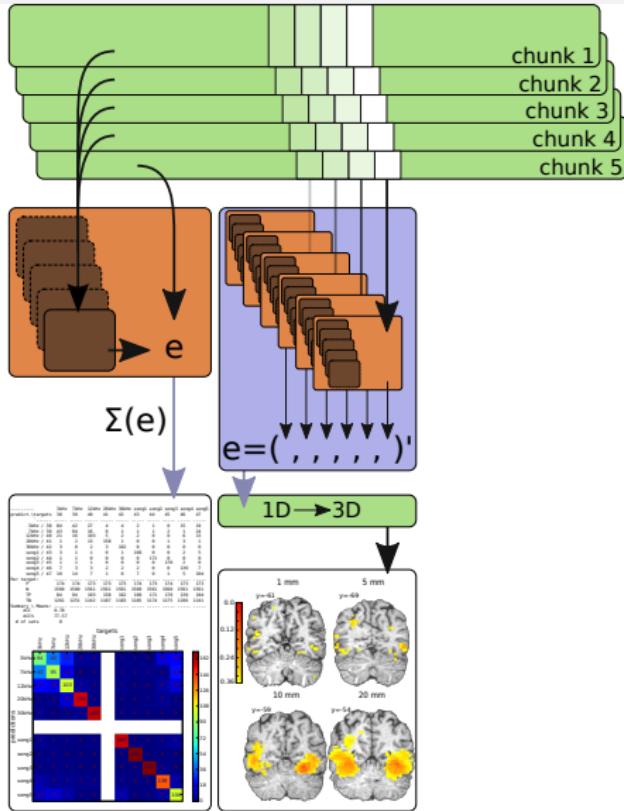
```
cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])
```

```
fwm = sphere_searchlight(cv, radius=3)
errors = cv(ds)
fwm_map = fwm(ds)
```

```
print cv.ca.stats
cv.ca.stats.plot()
```



Complete analysis example: Searchlights



```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                   targets=attr.targets,
                   chunks=attr.chunks,
                   mask='mask_brain.nii.gz',
                   add_fa={'vt':'vt.nii.gz'})

clf = LinearCSVMC()

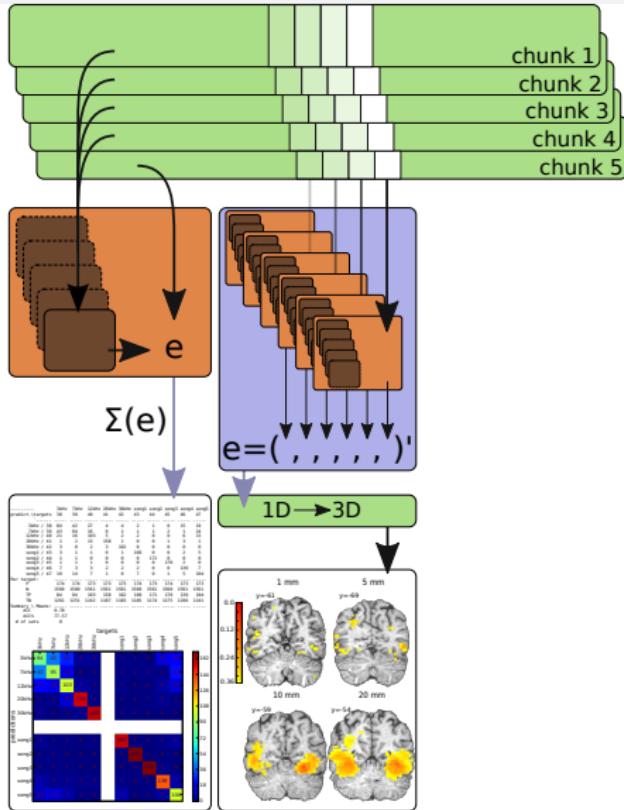
cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])

fwm = sphere_searchlight(cv, radius=3)

fwm_map = fwm(ds)

map2nifti(ds, fwm_map).to_filename(
    'out.nii.gz')
```

Complete analysis example: Searchlights



```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                   targets=attr.targets,
                   chunks=attr.chunks,
                   mask='mask_brain.nii.gz',
                   add_fa={'vt':'vt.nii.gz'})

clf = LinearCSVMC()

cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])

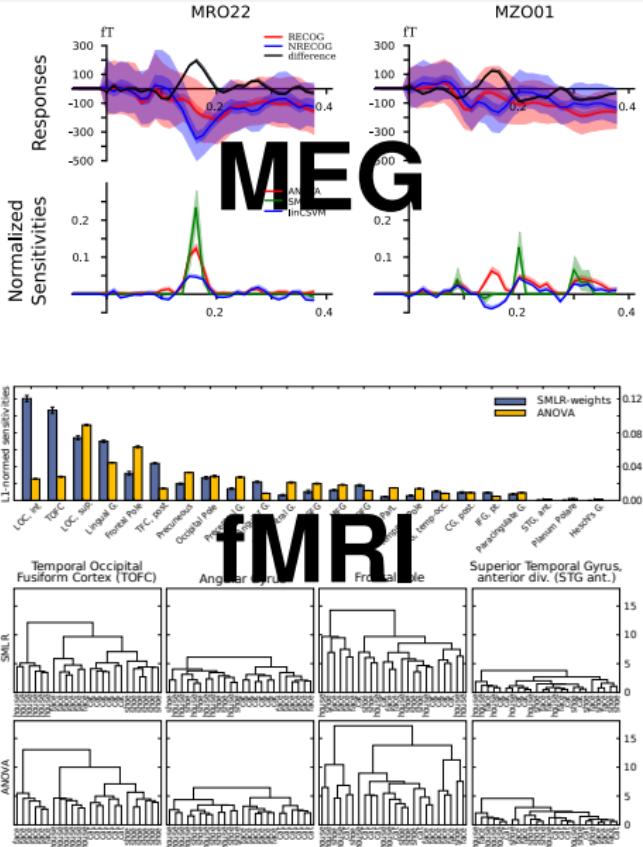
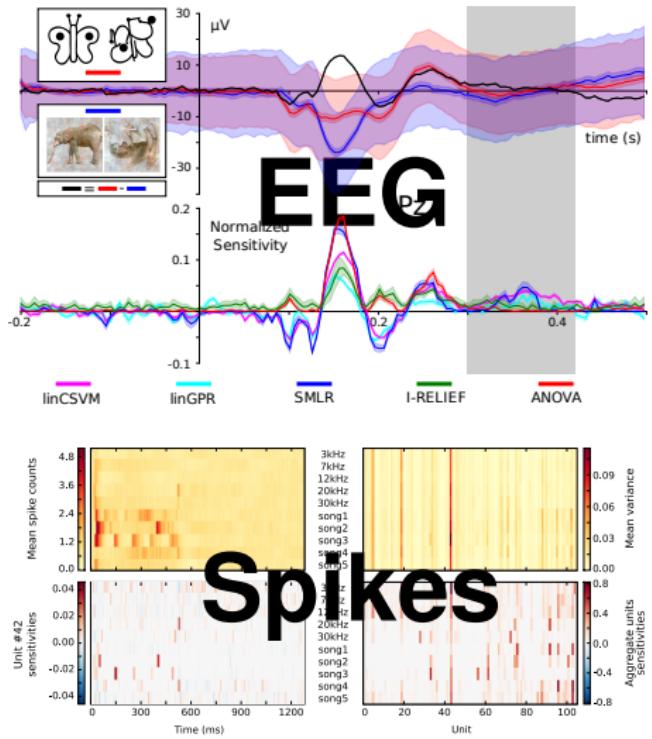
fwm = sphere_searchlight(cv, radius=3)

fwm_map = fwm(ds)

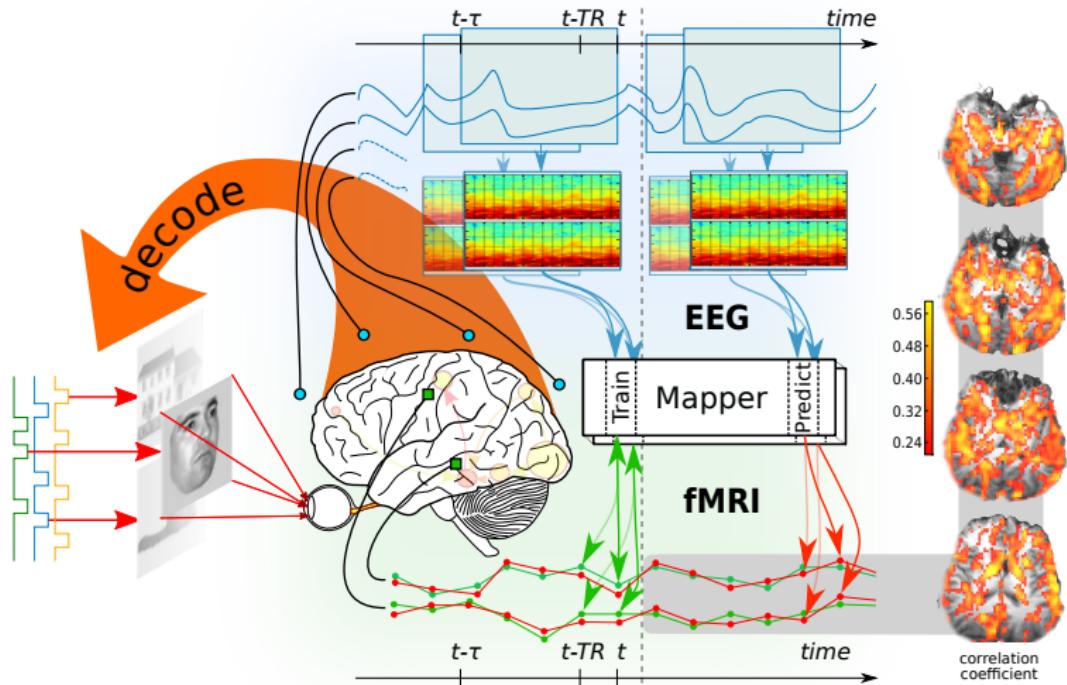
map2nifti(ds, fwm_map).to_filename(
    'out.nii.gz')

h5save('out.hdf5', fwm_map)
```

And with that we could do:



And with that we could do: TRANSfusion



Halchenko, Y. O., Hanke, M., Haxby, J. V., Hanson, S. J., and Herrmann, C. S. (2013). Transmodal analysis of neural signals. *ArXiv e-prints*

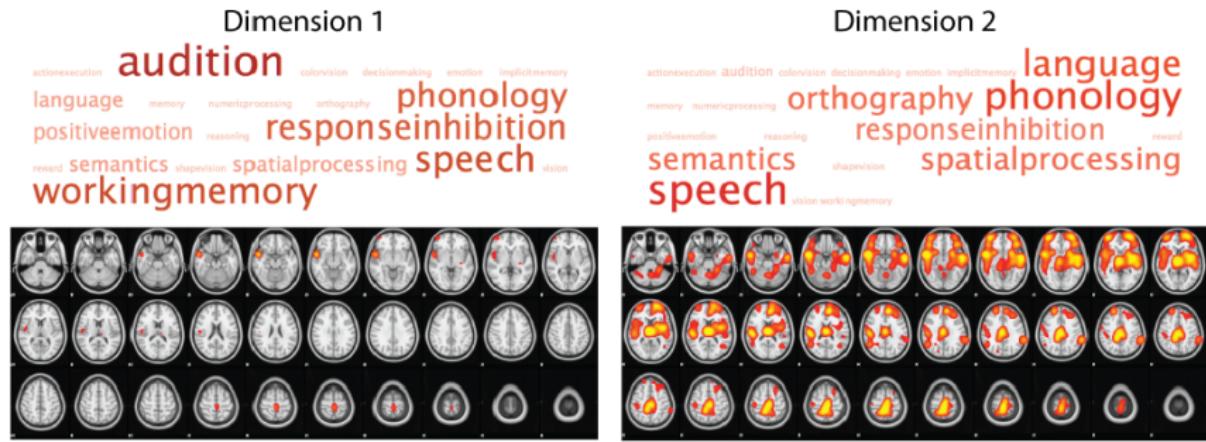
And with that could have done: Classification of states across studies

TABLE 2
Confusion Matrix for the Support Vector Machine Analyses

True task	Predicted task								
	Risk taking	Classification	Rhyme judgments	Working memory	Gambling decisions	Semantic judgments	Reading aloud	Response inhibition	Accuracy
Risk taking	14	0	0	0	1	0	1	0	87.50%
Classification	0	18	0	0	0	0	1	1	90.00%
Rhyme judgments	1	2	8	0	1	1	0	0	61.54%
Working memory	0	0	0	14	0	0	0	3	82.35%
Gambling decisions	0	3	0	0	11	2	0	0	68.75%
Semantic judgments	0	2	0	0	1	11	0	0	78.57%
Reading aloud	0	1	0	0	0	0	17	1	89.47%
Response inhibition	0	0	0	1	0	0	3	11	73.33%

Note. The table presents results obtained using leave-one-out cross-validation. The numbers in each row indicate how many subjects were classified as performing each task when they were actually performing the task indicated by the row label.

And with that could have done: Classifier sensitivity & concepts



Poldrack, R. A., Halchenko, Y. O., and Hanson, S. J. (2009). Decoding the large-scale structure of brain function by classifying mental states across individuals. *Psychological Science*, 20(11):1364–1372. PMC2935493

And with that **others** could do:

~~CONTINUE~~

Studies employing PyMVPA

2016

- Watson et al., *NeuroImage* (2016). Spatial properties of objects predict patterns of neural response in the ventral visual pathway
- Watson et al., *NeuroImage* (2016). Patterns of neural response in scene-selective regions of the human brain are modulated by level manipulations of spatial frequency

2015

- Floren et al., *Frontiers in Human Neuroscience* (2015). Accurately decoding visual information from fMRI data recorded in a realistic virtual environment
- Merkel et al., *NeuroImage* (2015). Neural correlates of multiple object tracking strategies
- Pogoda, et al., *Brain and Cognition* (2015). Multivariate representation of food preferences in the human brain
- Emmerling et al., *NeuroImage* (2015). Decoding the direction of imagined visual motion using 7 T ultra-high field fMRI
- Danelli et al., *Frontiers in Psychology* (2015). Framing effects reveal discrete lexical-semantic and sublexical processing during reading: an fMRI study
- Schlegel et al., *NeuroImage* (2015). The artist emerges: Visual art learning alters neural structure and function
- Maass et al., *ELife* (2015). Functional subregions of the human entorhinal cortex
- Sha et al., *Journal of Cognitive Neuroscience* (2015). The Animacy Continuum in the Human Ventral Vision Pathway
- Greisel et al., *arXiv* (2015). Photometric redshifts and model spectral energy distributions of galaxies from the SDSS DR10 data
- McNamee et al., *Journal of Neuroscience* (2015). Characterizing the associative content of brain structures involved in action and goal-directed actions in humans: a multivariate fMRI study
- Cole et al., *Cerebral Cortex* (2015). The behavioral relevance of task information in human prefrontal cortex
- Guo and Meng, *NeuroImage* (2015). The encoding of category-specific versus nonspecific information in human inferior temporal cortex

And with that **you** could do: Extended tutorial

Tutorial Introduction to PyMVPA

This chapter offers a tutorial introduction into PyMVPA. In the tutorial we are going to take a look at all major parts of PyMVPA, introduce the most important concepts, and explore particular functionality in real-life analysis examples.

- Tutorial Prerequisites
 - What Do I Need To Get Python Running
 - Recommended Reading and Viewing
- Part 1: A Gentle Start
 - Getting the data
 - Dealing With A Classifier
 - Cross-validation
 - References
- Part 2: Dataset Basics and Concepts
 - Attributes
 - Slicing, resampling, feature selection
 - Loading fMRI data
 - Storage
- Part 3: Mappers – The Swiss Army Knife
 - Doing `get_haxby2001_data()` From Scratch
 - There and back again – a Mapper's tale
- Part 4: Classifiers – All Alike, Yet Different
 - We Need To Take A Closer Look
 - Meta-Classifiers To Make Complex Stuff Simple
- Part 5: Searchlite
 - Measures
 - Searching, searching, searching, ...
 - For real!
- Part 6: Looking Without Searching – Sensitivity Analysis
 - It's A Kind Of Magic

Or maybe this would motivate **you**?

“... The PyMVPA manual has a picture of a **dude performing pattern-classification on fMRI data with his freaking cellphone.** **Awesome.** If you can do it on a cellphone, then I’m set.”



Who is PyMVPA for?

You want to ...

- use existing data-driven methodologies (such as MVPA, RSA, hyperalignment, etc) on **your data**
- apply **your new methods** to neural data
- conveniently compare **your methods** to ones available from other toolkits
- benefit from **established testing and distribution** infrastructure
- offer **students** a solid machine learning library

Get involved!

- Use it
- Find and evaluate existing methods
- Report bugs, send patches
- Support: Mailing list, GitHub
(<http://github.com/PyMVPA/PyMVPA>, also look for
good-for-hackathon labeled issues)
- Contribute your own developments
- Send us tests
- Spread the word

WE WELCOME
CONTRIBUTIONS!

DUE CREDITED
CONTRIBUTE



Motivation

“... The PyMVPA manual has a picture of a **dude performing pattern-classification on fMRI data with his freaking cellphone.** **Awesome.** If you can do it on a cellphone, then I’m set. ...”



Motivation vs pain

“... The PyMVPA manual has a picture of a **dude performing pattern-classification on fMRI data with his freaking cellphone.**

Awesome. If you can do it on a cellphone, then I’m set. I have a computer (MAC).

Hours later, I’m wrestling with MAC OS (Leopard) ...

PyMVPA follow up: 12.5 hours to happy time

Python installation and the PyMVPA toolbox passed, 12 1/2 hours after I started. Whew! ”

Houston, we've got a problem... AGAIN!

We use dozens of scientific software projects

Reliable deployment of (past or current) scientific heterogeneous software environments is not trivial



<http://Neuro.Debian.net>

Beware: No software is written by God



Unknown artist/origin, borrowed from

<http://blogs.quovantis.com/god-programmer/>

Beware: All software has bugs!



Unknown artist/origin, borrowed from

<http://blogs.quovantis.com/god-programmer/>

Beware: Even data can have bugs!



Unknown artist/origin, borrowed from

<http://blogs.quovantis.com/god-programmer/>

Most research software is not *rock solid*

- Too few users, on too many platforms
- Bug reporting is heterogeneous, time-consuming, and painful
- Lack of professional programming training/experience
- Insufficient or inappropriate testing and quality assurance
- Death-by-Ph.D. phenomenon
- Opaque development procedures
 - No public version control system
 - No public bug tracker

Most research software is not *rock solid*

- Too few users, on too many platforms
- Bug reporting is heterogeneous, time-consuming, and painful
- Lack of professional programming training/experience
- Insufficient or inappropriate testing and quality assurance
- Death-by-Ph.D. phenomenon
- Opaque development procedures
 - No public version control system
 - No public bug tracker

Broken by design?

- Impossible to obtain funding for software development and maintenance (alone)
- Development of software tools often not considered scientific progress

Vision

Why don't we all use the same platform...

- **that we can freely share with anyone**
- that works on all devices, operating systems, ...
- that is guaranteed to be available for as long as we want, wherever we want
- that makes manual maintenance trivial, or superfluous
- so all software is available in a single environment
- so we can share our experience with colleagues
- so we can share data processing workflows easily
- so developers can focus their scarce resources

Deployment resolution: **Neuro****Debian**

Role model: **debian**

- Vast archive of maintained software
(≈50,000 binary/30,000 source packages)
- Self-governed, “do-ocracy”, no need to earn money, going strong for over 20 years
- Origin of most active software distributions



Deployment resolution: **NeuroDebian**

Role model: debian



Deployment resolution: **Neuro****Debian**

Role model: **debian**

- Vast archive of maintained software
(≈50,000 binary/30,000 source packages)
- Self-governed, “do-ocracy”, no need to earn money, going strong for over 20 years
- Origin of most active software distributions



We could ...

- Adopt technology and procedures
- Participate in the Debian project and integrate all research software
- Benefit from the work of thousands of *additional* developers
- Call it **Neuro****Debian**, add fancy logo



More detail on why and how

www.frontiersin.org/Neuroinformatics/10.3389/fninf.2012.00022/full



frontiers IN NEUROINFORMATICS

OPINION ARTICLE

Share 2 Like 8 Comment 0 [f](#) [in](#) [t](#) [Q+1](#) 24 Share Altmetric 10 4,511 Views

Front. Neuroinform., 29 June 2012 | doi: 10.3389/fninf.2012.00022

Open is not enough. Let's take the next step: an integrated, community-driven computing platform for neuroscience

Yaroslav O. Halchenko^{1,2,3,†*} and Michael Hanke^{4,5,3,†*}

¹ Center for Cognitive Neuroscience, Dartmouth College, Hanover, NH, USA
² Department of Psychological and Brain Sciences, Dartmouth College, Hanover, NH, USA
³ Debian Project, <http://www.debian.org>
⁴ Department of Experimental Psychology, Otto-von-Guericke University, Magdeburg, Germany
⁵ Center for Behavioral Brain Sciences, Magdeburg, Germany

The last 5 years have seen dramatic improvements in the collaborative research infrastructure. A need for open research tools has been identified

Article Type: All

Publication Date: From

Halchenko, Y. O. and Hanke, M. (2012). Open is not enough. Let's take the next step: An integrated, community-driven computing platform for neuroscience. *Frontiers in Neuroinformatics*, 6(00022). PMC3458431

NeuroDebian from a researcher's perspective

Install simple editor and complex MRI analysis package

```
apt-get install gedit fsl
```

Install software collection for psychophysics

```
apt-get install science-psychophysics
```

Keep the whole system up-to-date

```
apt-get dist-upgrade
```

Get support

neurodebian-users@alioth-lists.debian.net, #neurodebian IRC,
neurostars.org

Reproducibility (*Can you replicate your environment?*)

<http://archive.debian.org>

Reproduce state of a research box with Python 1.5.2 as in 2003:

```
debootstrap --include=emacs20,python-base \
    potato /tmp/potato http://archive.debian.org/debian
```

schroot into it

```
> cat /etc/schroot/chroot.d/potato
[potato]
description=Antique Debian from 2003
directory=/tmp/potato
users=YOURLOGIN

> schroot -c potato
> python -c "import sys; print sys.version"
1.5.2 (#0, Dec 27 2000, 13:59:38) [GCC 2.95.2 20000220 ...
```

NeuroDebian: Reproducible containers

nd_freeze freezes APT repositories specification to a specific date.

Just say to which date:

```
reproin > head -n 20 Singularity
# Generated by Neurodocker version 0.4.3-2-g01cdd22
#
# Thank you for using Neurodocker...
```

```
Bootstrap: docker
From: neurodebian:stretch
```

```
%post
...
nd_freeze 20190513
...
```

NeuroDebian friend: Neurodocker

Creator of containerized environments for neuroimaging.

Features:

- Generates custom and reproducible analysis environments
- Contains recipes for installing many common neuroimaging tools
- Supports generating
 - Dockerfiles
 - Singularity recipes
- Helps minimize the size of existing containers using ReproZip

Tools supported (8 May 2018):

- AFNI
- ANTs
- Convert3D
- dcm2niix
- FreeSurfer
- FSL
- MATLAB Compiler Runtime
- MINC
- Miniconda
- MRtrix3
- NeuroDebian**
- PETPVC
- SPM12



Containers aren't a panacea, but a useful technology, with a range of tools to assist with

Fantastic containers and how to tame them

Yaroslav O. Halchenko¹, Kyle Meyer¹, Matt Travers², Dorota Jarecka³, Satrajit Ghosh³, Jakub Kaczmarzyk³, Michael Hanke^{4,5}

Contemporary neuroimaging research is computationally intensive and heavily relies on analysis software. Results of an analysis might depend not only on the version of the software the scientist is using, but also on the overall computational environment (including the operating system, preinstalled software, environment variables, and data). Containerized computing environments, using [Docker](https://www.docker.com) (<https://www.docker.com>) and [Singularity](https://www.sylabs.io/singularity) (<https://www.sylabs.io/singularity>), gained popularity as a means to maximize portability of complete environments, minimize operational variance across available infrastructure (laptops/HPC/cloud), and share complete environments. While containers are a very powerful technology, **additional care and tooling are necessary to use them efficiently and reproducibly**. We explore containers-oriented computing across the container life cycle—creation, validation (QA), storage (dissemination), and usage—along with approaches and tooling to assist at each step. This overview will focus on the reproducibility aspects, including the reproducibility of the containers themselves, and highlight recent technological developments.

Containers ARE NOT the ...

- reproducibility panacea
- solution for software sustainability or integration
- replacement for software distributions
(e.g., GNU/Linux distributions, Conda, etc)

Containers are great because they ...

- are portable and can be used on most operating systems
- provide easy access to broad range of existing environments from container hubs
- can be used for “quick&dirty” bundling of computing elements
- are isolated and easy to install/remove
- simplify Continuous Integration



Visit <https://github.com/myyoda> for more info

Track all input data, code, and computational environments needed to produce analysis outputs in version controlled datasets — and reproducibility you will achieve!

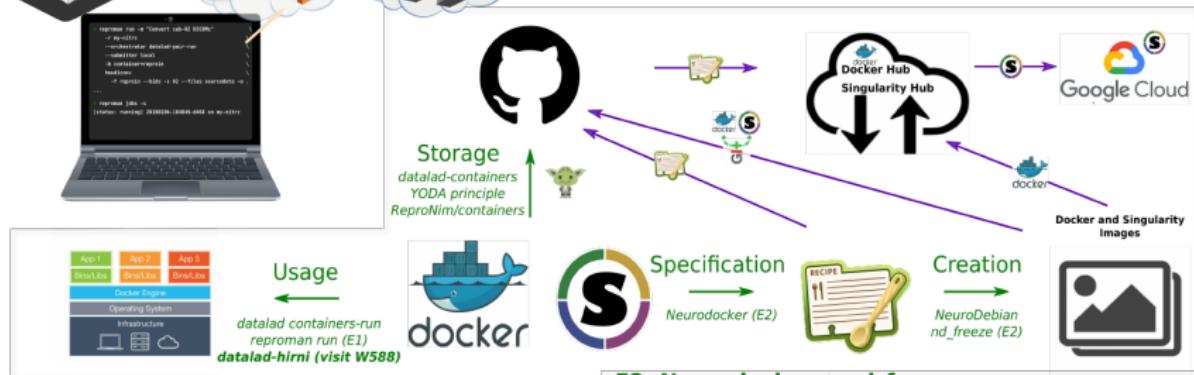
neuro.debian.net/_files/ohbm2019-containers.pdf

E1: repreman run for remote execution



(see e.g. <https://lwn.net/Articles/786066/>):

- security and legal implications
- difficult to inspect/compare different containers on how they differ
(use <https://github.com/GoogleContainerTools/container-diff>,



Verification

Automated testing of containers should be performed!

- **Boutique** specification that can contain **test definitions** to validate containerized pipelines [Glatard et al., 2018], <https://github.com/boutiques/boutiques/#test-your-tool>
- **Flywheel Gears** generated by the **gearificator** (<https://github.com/yarikoptic/gearificator>) carry regression tests, see <https://github.com/yarikoptic/gearificated-nipype>
- **Nifflows** (<https://github.com/nifflows>) use Testkraken

E2: Neurodocker + nd freeze

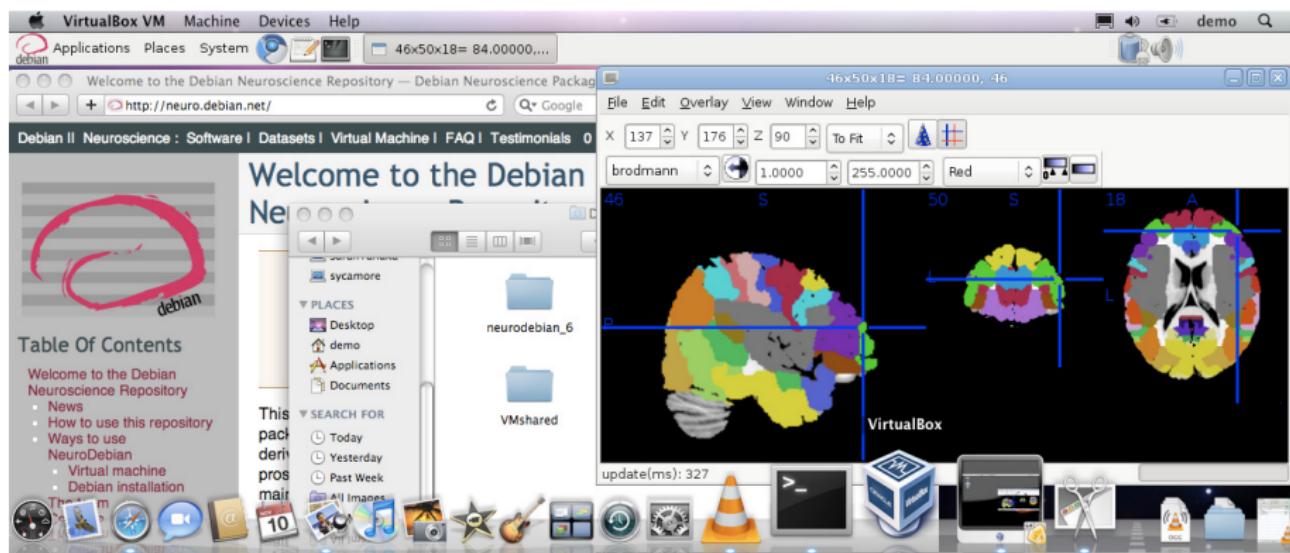
Reproin reproducible container recipe generation

A terminal window displays the command: "neurodocker generate 'docker:singularity'". The output shows a Dockerfile being generated with various commands, including apt-get updates, installations of curl, nodejs, npm, and boids-validator@0.1.1, and a user named "neuro". The Dockerfile ends with an entrypoint for "neuro".

Below the terminal, a "Singularity" section shows a snippet of a Singularity bootstrap file. It includes commands for apt-get update, installing curl, nodejs, and npm, and setting environment variables like DEBIAN_FRONTEND and environment. It also exports an environment variable and installs certificates.

NeuroDebian: Availability

- apt-get install neurodebian on any Debian-based OS
- docker pull neurodebian (a base for many custom images)
- singularity pull shub://neurodebian/neurodebian
- Virtual machine (great for teaching, workshops etc):



After X years and the contributions of many people:



Who is **NeuroDebian** for?

You want to ...

- use a **rock-solid** operating system
- have **readily usable and latest** software at your fingertips
- **try something new**, without investing much time
- offer **students** a fully functional “take-away” research environment
- **efficiently collaborate** with other researchers
- **waste less time** maintaining computers&containers
- have **your own software** easily available for other's to use
- **develop software** without worrying about dependencies

Get involved!

CONTRIBUTE

- Use it!
- Report bugs, send patches
- Support: Mailing list, IRC
(<http://neuro.debian.net/#contacts>)
- Help to (co-)maintain a package
- Accompany your tools with reliable build infrastructure and good test batteries
- Package your own software
- Spread the word
- Contribute to the coffee art collection
(<http://neuro.debian.net/coffeeart.html>)

Houston, we've got a problem... not again, please!

Data is a 2nd-class citizen within software platforms



<http://datalad.org>

Why?

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

derived and/or curated data does change!

Why?

Filename	Date Modified	Size	Type
data_2010.05.28_test.dat	3:37 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_re-test.dat	4:29 PM 5/28/2010	421 KB	DAT file
data_2010.05.28_re-re-test.dat	5:43 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_calibrate.dat	7:17 PM 5/28/2010	1,256 KB	DAT file
data_2010.05.28_huh???.dat	7:20 PM 5/28/2010	30 KB	DAT file
data_2010.05.28_WTF.dat	9:58 PM 5/28/2010	30 KB	DAT file
data_2010.05.29_aaarrgh.dat	12:37 AM 5/29/2010	30 KB	DAT file
data_2010.05.29_#\$@*&!!..dat	2:40 AM 5/29/2010	0 KB	DAT file
data_2010.05.29_crap.dat	3:22 AM 5/29/2010	437 KB	DAT file
data_2010.05.29_notbad.dat	4:16 AM 5/29/2010	670 KB	DAT file
data_2010.05.29_woohoo!.dat	4:47 AM 5/29/2010	1,349 KB	DAT file
data_2010.05.29_USETHISONE.dat	5:08 AM 5/29/2010	2,894 KB	DAT file
analysis_graphs.xls	7:13 AM 5/29/2010	455 KB	XLS file
ThesisOutline.doc	7:26 AM 5/29/2010	38 KB	DOC file
Notes_Meeting_with_ProfSmith.txt	11:38 AM 5/29/2010	1,673 KB	TXT file
JUNK...	2:45 PM 5/29/2010		Folder
data_2010.05.30_startingover.dat	8:37 AM 5/30/2010	420 KB	DAT file

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

The screenshot shows the homepage of the DataLad website. At the top, there is a browser header with a back/forward button, refresh, home, and search icons. The URL bar shows 'datalad.org'. Below the header is the DataLad logo, followed by a navigation menu with links for About, Get DataLad, Features, Datasets, Development, and Docs. The main content area has a dark background with a hexagonal grid pattern. It features a large white text block that reads: "Providing a data portal and a versioning system for everyone, DataLad lets you have your data and control it too." Below this is a yellow button with the text "Get DataLad". The page is divided into two main sections: "Discover Data" on the left and "Consume Data" on the right, each with its own icon and descriptive text.

Discover Data

DataLad has built-in support for **metadata** extraction and **search**. With only a few steps, you can search through a large collection of readily available datasets and immediately download them.

[See more...](#)

Consume Data

DataLad offers direct **access to individual files** — great when you only need a few files from some large datasets for an analysis. Files in a dataset can be distributed across multiple download sources with tailored permissions to match your **data privacy needs**. [See more...](#)

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

How: Foundation #2 – Git-annex

- is **built on top of Git**
- provides **access to data content** from variety of sources:
HTTP, FTP, Webdav, bittorrent, RSYNC, Amazon S3, etc.
- allows for **custom extensions** to get access to offload data:
Dropbox, Google Drive, Box.com (will demo later) 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, bittorrent, RSYNC, Amazon S3, etc.
- allows for **custom extensions** to get access to offload data:
Dropbox, Google Drive, Box.com (will demo later) 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, bittorrent, RSYNC, Amazon S3, etc.
- allows for **custom extensions** to get access to offload data:
Dropbox, Google Drive, Box.com (will demo later) 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

DataLad in one figure

DataLad

Install any dataset
datalad install <url/path>
Install from open-data collection
datalad install ///

From scratch

Create and add arbitrary content
datalad create
datalad add ...
DataLad-enabled software

Web resources

Customizable (repeated) crawling
of data portals or websites
datalad crawl-init ...
datalad crawl

Dataset

joint management of code, data,
and metadata in one repository

.git/
annex/
Git
version
control

.datalad/
metadata/

largefile.dat
dir/doc.pdf

...

Git-annex data management

metadata extraction

Dataset

link datasets for automatically
resolvable data-dependencies

.git/
.gitmodules

.datalad/
metadata/

rawdata
code/fig1.py

...

arbitrary nesting

DataLad

Create publication target(s)
datalad create-sibling ...
(Repeated) publication
datalad publish --to ...

Data export

Snapshot TAR/ZIP archive
datalad export-archive ...
Data sharing portal
datalad export-to-figshare

Metadata export

Full metadata dump (JSON-LD)
datalad metadata ...
Search for files or datasets
datalad search

DataLad in one figure

DataLad

Install any dataset
datalad install <url/path>
Install from open-data collection
datalad install ///

From scratch

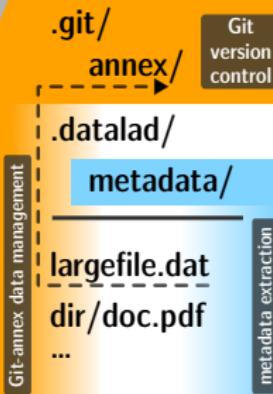
Create and add arbitrary content
datalad create
datalad add ...
DataLad-enabled software

Web resources

Customizable (repeated) crawling
of data portals or websites
datalad crawl-init ...
datalad crawl

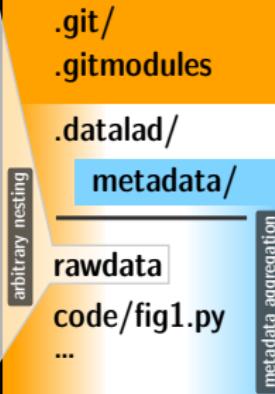
Dataset

Joint management of code, data,
and metadata in one repository



Dataset

link datasets for automatically
resolvable data-dependencies



DataLad

Create publication target(s)
datalad create-sibling ...
(Repeated) publication
datalad publish --to ...

Data export

Snapshot TAR/ZIP archive
datalad export-archive ...
Data sharing portal
datalad export-to-fgshare

Metadata export

Full metadata dump (JSON-LD)
datalad metadata ...
Search for files or datasets
datalad search

DataLad in one figure

DataLad

Install any dataset
datalad install <url/path>
Install from open-data collection
datalad install ///

From scratch

Create and add arbitrary content
datalad create
datalad add ...
DataLad-enabled software

Web resources

Customizable (repeated) crawling of data portals or websites
datalad crawl-init ...
datalad crawl

Dataset

joint management of code, data, and metadata in one repository

.git/
annex/

Git
version
control

.datalad/
metadata/

Git-annex data management

metadata extraction

largefile.dat
dir/doc.pdf
...

Dataset

link datasets for automatically resolvable data-dependencies

.git/
.gitmodules

.datalad/
metadata/

arbitrary nesting

rawdata
code/fig1.py
...

metadata aggregation

DataLad

Create publication target(s)
datalad create-sibling ...
(Repeated) publication
datalad publish --to ...

Data export

Snapshot TAR/ZIP archive
datalad export-archive ...
Data sharing portal
datalad export-to-fgshare

Metadata export

Full metadata dump (JSON-LD)
datalad metadata ...
Search for files or datasets
datalad search

DataLad in one figure

DataLad

Install any dataset
datalad install <url/path>
Install from open-data collection
datalad install ///

From scratch

Create and add arbitrary content
datalad create
datalad add ...
DataLad-enabled software

Web resources

Customizable (repeated) crawling
of data portals or websites
datalad crawl-init ...
datalad crawl

Dataset

joint management of code, data,
and metadata in one repository

.git/
annex/

Git
version
control

.datalad/
metadata/

arbitrary data management

largefile.dat
dir/doc.pdf
...

metadata extraction

Dataset

link datasets for automatically
resolvable data-dependencies

.git/
.gitmodules

.datalad/
metadata/

arbitrary nesting

rawdata
code/fig1.py
...

metadata aggregation

DataLad

Create publication target(s)
datalad create-sibling ...
(Repeated) publication
datalad publish --to ...

Data export

Snapshot TAR/ZIP archive
datalad export-archive ...
Data sharing portal
datalad export-to-fgshare

Metadata export

Full metadata dump (JSON-LD)
datalad metadata ...
Search for files or datasets
datalad search

DataLad in one figure

DataLad

Install any dataset
datalad install <url/path>
Install from open-data collection
datalad install ///

From scratch

Create and add arbitrary content
datalad create
datalad add ...
DataLad-enabled software

Web resources

Customizable (repeated) crawling
of data portals or websites
datalad crawl-init ...
datalad crawl

Dataset

joint management of code, data,
and metadata in one repository

.git/
annex/

Git
version
control

.datalad/
metadata/

Git-annex data management

metadata extraction

largefile.dat
dir/doc.pdf

...

Dataset

link datasets for automatically
resolvable data-dependencies

.git/
.gitmodules

.datalad/
metadata/

arbitrary nesting

rawdata
code/fig1.py

...

metadata aggregation

DataLad

Create publication target(s)
datalad create-sibling ...
(Repeated) publication
datalad publish --to ...

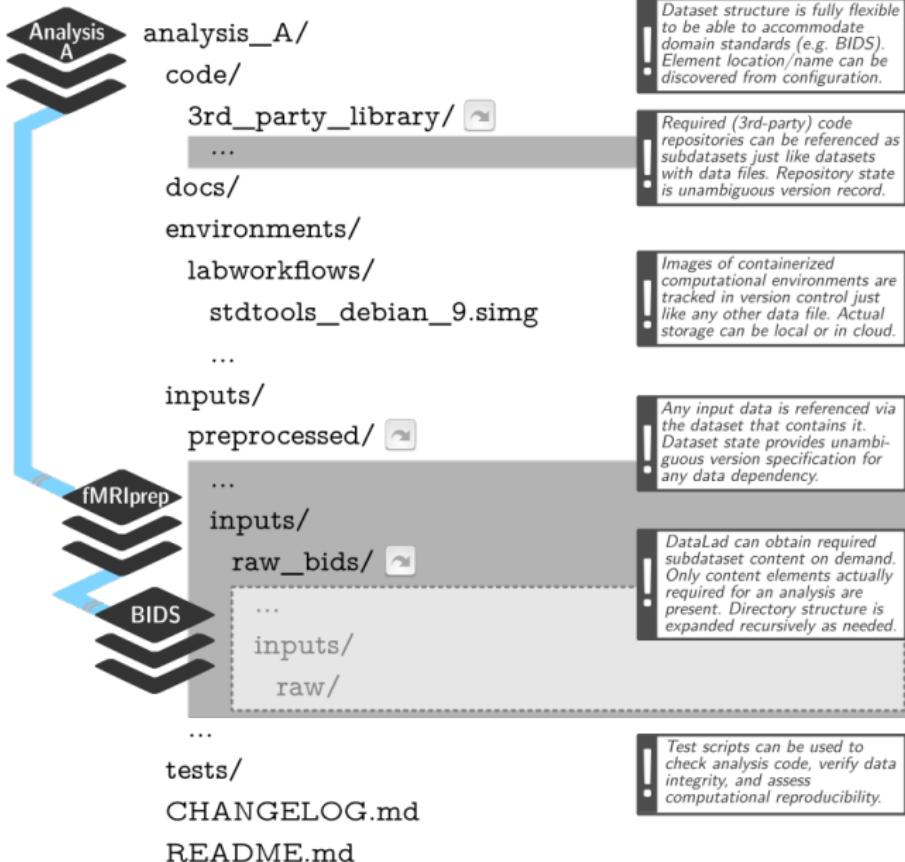
Data export

Snapshot TAR/ZIP archive
datalad export-archive ...
Data sharing portal
datalad export-to-fgshare

Metadata export

Full metadata dump (JSON-LD)
datalad metadata ...
Search for files or datasets
datalad search

DataLad in one figure



DataLad in one figure

DataLad

Install any dataset
datalad install <url/path>
Install from open-data collection
datalad install ///

From scratch

Create and add arbitrary content
datalad create
datalad add ...
DataLad-enabled software

Web resources

Customizable (repeated) crawling
of data portals or websites
datalad crawl-init ...
datalad crawl

Dataset

joint management of code, data,
and metadata in one repository

.git/
annex/

Git
version
control

.datalad/
metadata/

management

Dataset

link datasets for automatically
resolvable data-dependencies

.git/
.gitmodules

.datalad/
metadata/

nesting

rawdata
code/fig1.py
...

DataLad

Create publication target(s)
datalad create-sibling ...
(Repeated) publication
datalad publish --to ...

Data export

Snapshot TAR/ZIP archive
datalad export-archive ...
Data sharing portal
datalad export-to-fgshare

Metadata export

Full metadata dump (JSON-LD)
datalad metadata ...
Search for files or datasets
datalad search

DataLad in one figure

DataLad

Install any dataset
datalad install <url/path>
Install from open-data collection
datalad install ///

From scratch

Create and add arbitrary content
datalad create
datalad add ...
DataLad-enabled software

Web resources

Customizable (repeated) crawling
of data portals or websites
datalad crawl-init ...
datalad crawl

Dataset

joint management of code, data,
and metadata in one repository

.git/
annex/

Git
version
control

.datalad/

metadata/

largefile.dat

dir/doc.pdf

...

Git-annex data management

metadata extraction

Dataset

link datasets for automatically
resolvable data-dependencies

.git/
.gitmodules

.datalad/

metadata/

rawdata

code/fig1.py

...

arbitrary nesting

DataLad

Create publication target(s)
datalad create-sibling ...
(Repeated) publication
datalad publish --to ...

Data export

Snapshot TAR/ZIP archive
datalad export-archive ...
Data sharing portal
datalad export-to-fgshare

Metadata export

Full metadata dump (JSON-LD)
datalad metadata ...
Search for files or datasets
datalad search

DataLad in one figure

DataLad

Install any dataset
datalad install <url/path>
Install from open-data collection
datalad install ///

From scratch

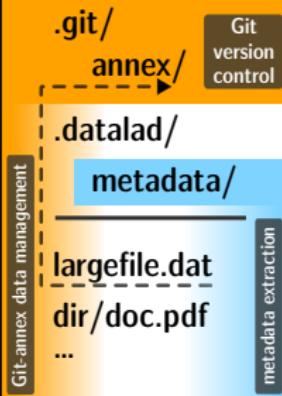
Create and add arbitrary content
datalad create
datalad add ...
DataLad-enabled software

Web resources

Customizable (repeated) crawling
of data portals or websites
datalad crawl-init ...
datalad crawl

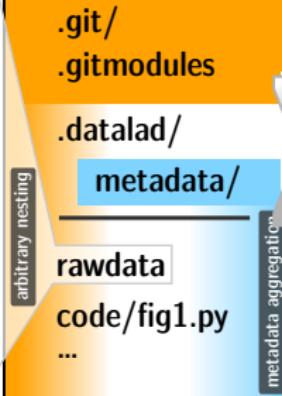
Dataset

joint management of code, data,
and metadata in one repository



Dataset

link datasets for automatically
resolvable data-dependencies



DataLad

Create publication target(s)
datalad create-sibling ...
(Repeated) publication
datalad publish --to ...

Data export

Snapshot TAR/ZIP archive
datalad export-archive ...
Data sharing portal
datalad export-to-figshare

Metadata export

Full metadata dump (JSON-LD)
datalad metadata ...
Search for files or datasets
datalad search

DataLad in one figure

DataLad

Install any dataset
datalad install <url/path>
Install from open-data collection
datalad install ///

From scratch

Create and add arbitrary content
datalad create
datalad add ...
DataLad-enabled software

Web resources

Customizable (repeated) crawling
of data portals or websites
datalad crawl-init ...
datalad crawl

Dataset

joint management of code, data,
and metadata in one repository

.git/
annex/

Git
version
control

.datalad/
metadata/

Git-annex data management

metadata extraction

largefile.dat
dir/doc.pdf

...

Dataset

link datasets for automatically
resolvable data-dependencies

.git/
.gitmodules

.datalad/
metadata/

arbitrary nesting

rawdata
code/fig1.py

...

DataLad

Create publication target(s)
datalad create-sibling ...
(Repeated) publication
datalad publish --to ...

Data export

Snapshot TAR/ZIP archive
datalad export-archive ...
Data sharing portal
datalad export-to-fgshare

Metadata export

Full metadata dump (JSON-LD)
datalad metadata ...
Search for files or datasets
datalad search

Open Data Collection (AKA ///)

Growing data “distribution” (>60TB, hosting locally only 342GB)

- <http://datasets.datalad.org>
- Get it “all”: datalad install [-r] [-g] ///

Covered :

- Neuroimaging archives: <openfmri.org>, <crcns.org>, etc.
- Other neuro-data (from kaggle, INDI, FC etc.)
- Even some music (podcast radio):
<github.com/datalad/ratholeradio-archive>

Coming :

- More neuroimaging data (HCP, XNAT-support, etc)

Integrations :

- ReproIN, OpenNeuro, WiP: OSF, Zenodo

Metadata :

- Various extractors (BIDS, DICOM, XMP, ...)



DataLad: File-level metadata search

Find T1w anatomicals for females of age 24

```
> datalad \
  -c datalad.search.index-egrep-documenttype=files \
  search \
    bids.subject.sex:female \
    bids.type:t1 \
    bids.subject.age:24

.../labs/haxby/attention/sub-rid000012/anat/sub-rid000012_T1w.nii.gz (file)
.../labs/haxby/attention/sub-rid000012/anat/sub-rid000012_rec-ehalfhalf_T1w.nii.gz (file)
.../labs/haxby/attention/sub-rid000024/anat/sub-rid000024_T1w.nii.gz (file)
.../labs/haxby/attention/sub-rid000024/anat/sub-rid000024_rec-ehalfhalf_T1w.nii.gz (file)
.../labs/haxby/attention/sub-rid000032/anat/sub-rid000032_T1w.nii.gz (file)
.../labs/haxby/attention/sub-rid000032/anat/sub-rid000032_rec-ehalfhalf_T1w.nii.gz (file)
.../openfmri/ds000001/sub-11/anat/sub-11_T1w.nii.gz (file)
.../openfmri/ds000001/sub-15/anat/sub-15_T1w.nii.gz (file)
.../openfmri/ds000002/sub-02/anat/sub-02_T1w.nii.gz (file)
.../openfmri/ds000002/sub-05/anat/sub-05_T1w.nii.gz (file)
.../openfmri/ds000005/sub-07/anat/sub-07_T1w.nii.gz (file)
.../openfmri/ds000006/sub-14/ses-retest/anat/sub-14_ses-retest_T1w.nii.gz (file)

...
```

DataLad: Rich metadata

Various extractors (XMP, DICOM, BIDS, etc)

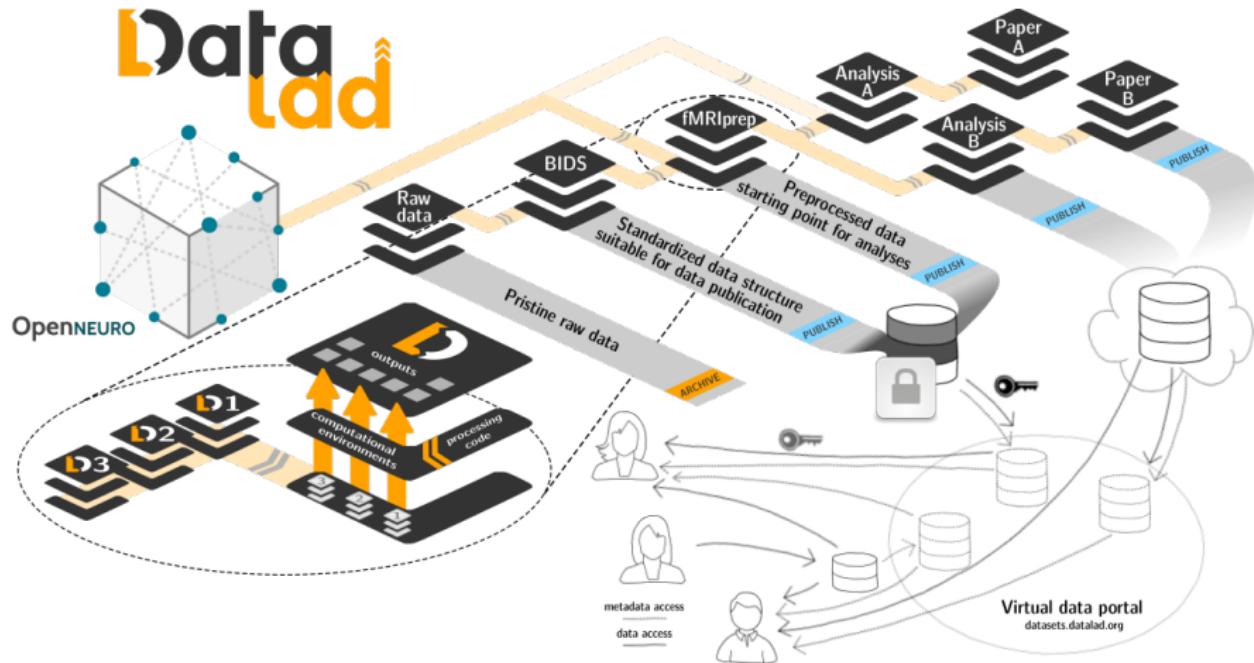
```
> datalad search --show-keys name | grep -5 bids\.Cog  
bids.BodyPartExamined  
bids.Bolus  
bids.CalibrationImage  
bids.CalibrationImageTR  
bids.CardiacTriggerDelayTimes  
bids.CogAtlasID  
bids.CogPOID  
bids.CoilCombinationMethod  
bids.Columns  
bids.ConsistencyInfo  
bids.ContactEmail  
bids.ContentDescription
```

DataLad: Search for the tasks of interest

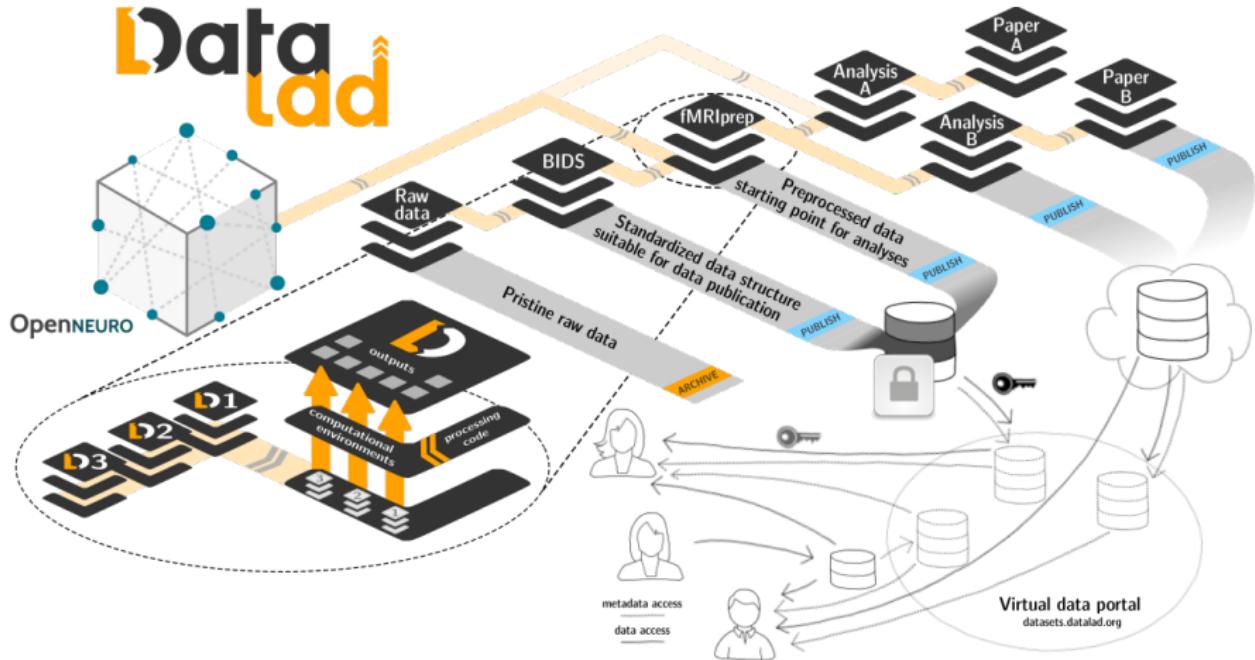
Fields to find files for a task of interest

```
> datalad search --show-keys full | grep -A3 bids.CogAtlas  
bids.CogAtlasID  
in 52 datasets  
has 62 unique values:  
'https://www.cognitiveatlas.org/task/id/trm_4c899211a965'  
'http://www.cognitiveatlas.org/task/id/trm_565a31fa6f444'  
...  
  
> datalad search --show-keys full | grep -A3 bids.CogPOID  
bids.CogPOID  
in 3 datasets  
has 8 unique values: '',  
'http://wiki.cogpo.org/index.php?title=Breath-Holding_Par'  
'http://wiki.cogpo.org/index.php?title=Flashing_Checkerbo'  
'http://www.cogpo.org/ontologies/Film_Clip', 'COGPO_00086'  
'COGPO_00078', 'COGPO_00107', 'COGPO_00136'
```

DataLad: Summary in one figure



DataLad: Summary in one figure



Managing data can be as simple to managing code and software

Later just go to datalad.org/features.html

All together, DataLad can automatically create an extensive provenance record, with all the information necessary to rerun an analysis from beginning to end and demonstrate reproducibility.

Here is a quick demo of how DataLad captures the results derived from input data. Scroll further down for a more in-depth demo of a full analysis.

```
commit d5e47ddaa6311f233cb0630a7a4c11e453a3ed7e
Author: DataLad Demo <demo@datalad.org>
Date:   Wed Aug 23 13:41:01 2017 +0200

[DATALAD RUNCMD] convert -extract 1522x1522+1470+1470 sou...
    === Do not change lines below ===
    {
        "pwd": ".",
        "cmd": [
            "convert",
            "-extract",
            "1522x1522+1470+1470",
            "sources/flowers.jpg",
            "flower2.jpg"
        ],
        "exit": 0
    }
    ^^^ Do not change lines above ^^^

flower2.jpg | 1 +
1 file changed, 1 insertion(+)
/demo % #
```

Get the script for this demo

DANDI



github.com/dandi

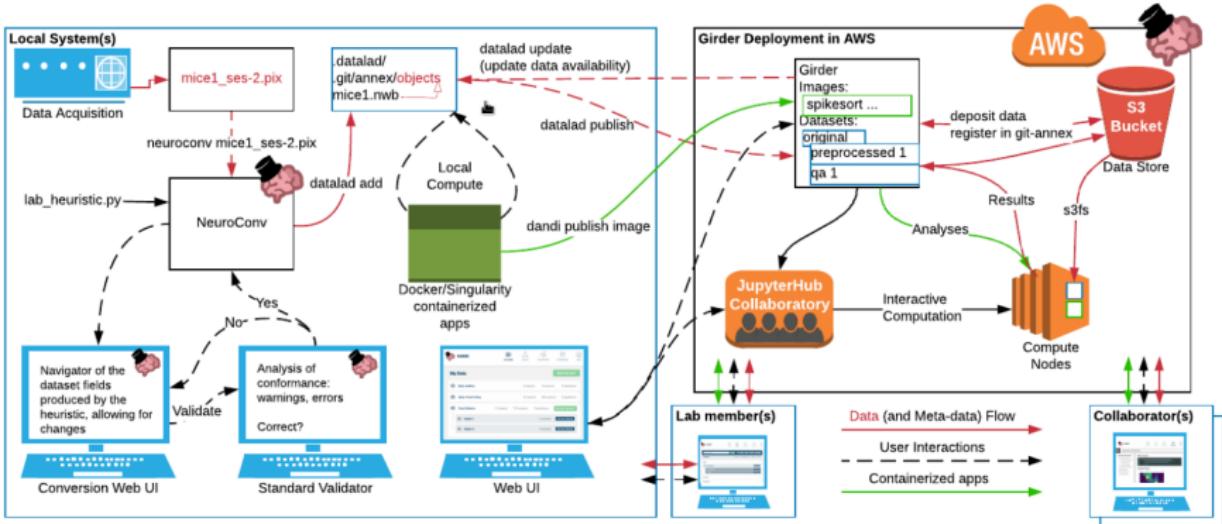
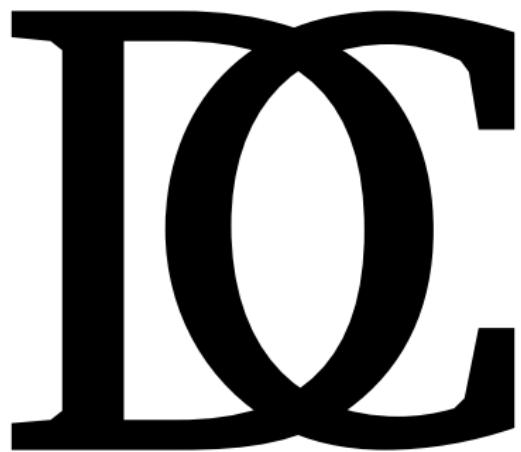


Figure 3. DANDI architecture diagram with interactions between components. The local system overview shows an example workflow of data conversion using NeuroConv to create and validate a DataLad dataset, which can be pushed to AWS or computed on locally. Apps and data can be shared between lab members and collaborators, and other users of the archive. Components not shown: Local girder clusters, IPFS network.

Only one more problem, I promise!

Scientific methods and software are not adequately cited...
since often we do not even know what we use!



<http://duecredit.org>

Why?

- Our (methods, software, datasets authors) **work is not cited adequately**
- Even if cited, **version information is often omitted**
- Absent visibility of contributions to existing projects fosters **prima ballerina projects**
- It is **tedious to collect/format references** for publications using our products

DueCredit's approach

Make it **VERY EASY** to

- **collect references** for methods, software, and data **actually used** by the analysis
- **accumulate reference** information through the entirety of the research project
- **collect references** for methods implemented in the software and its users
- **request references** to cite using desired format: LaTeX + BibTeX, rendered citations in variety of styles, etc

Example: examples/example_scipy.py

```
# A tiny analysis script to demonstrate duecredit
#
# Import of duecredit is not necessary if you just run this script with
# python -m duecredit
# import duecredit # Just to enable duecredit
from scipy.cluster.hierarchy import linkage
from scipy.spatial.distance import pdist
from sklearn.datasets import make_blobs

print("I: Simulating 4 blobs")
data, true_label = make_blobs(centers=4)

dist = pdist(data, metric='euclidean')

Z = linkage(dist, method='single')
print("I: Done clustering 4 blobs")
```

What is it for?

```
> python -m duecredit examples/example_scipy.py  
I: Simulating 4 blobs  
I: Done clustering 4 blobs
```

DueCredit Report:

```
- scipy (v 0.14.1) [1]  
  - scipy.cluster.hierarchy:linkage (v 0.14.1) [2]  
- sklearn (v 0.16.1) [3]
```

2 packages cited

0 modules cited

1 functions cited

References

- [1] Jones, E. et al., 2001. SciPy: Open source scientific tools for ...
- [2] Sibson, R., 1973. SLINK: an optimally efficient algorithm for ...
- [3] Pedregosa, F. et al., 2011. Scikit-learn: Machine learning in Python.

A bit bigger example

```
> python -m duecredit /usr/bin/nosetests mvpa2.tests.test_transerror
.....
DueCredit Report:
- libsvm (v None) [1]
- mvpa2 (v 2.4) [2]
  - mvpa2.clfs.SVM (v 2.4) [3]
  - mvpa2.clfs.smlr:SMLR (v 2.4) [4]
  - mvpa2.clfs.transerror:_call (v 2.4) [5]
  - mvpa2.featsel.rfe:_train (v 2.4) [6]
  - mvpa2.measures.searchlight:_call (v 2.4) [7]
- scipy (v 0.14.1) [8]
- sklearn (v 0.16.1) [9]

4 packages cited
1 modules cited
4 functions cited
```

References

- [1] Chang, C.-C. & Lin, C.-J., 2011. LIBSVM. ACM Trans. Intell. Syst. ...
- [2] Hanke, M. et al., 2009. PyMVPA: a Python Toolbox for Multivariate ...
- [3] Vapnik, V., 1995. The Nature of Statistical Learning Theory, New ...
- ...

A bit bigger example: alternative output

```
> duecredit summary --format=bibtex
@article{Hanke_2009, title={PyMVPA: a unifying approach to the analysis of
@article{pedregosa2011scikit,
    title={Scikit-learn: Machine learning in Python},
    author={Pedregosa, Fabian and Varoquaux, Ga{"e}l and Gramfort,
    Alexandre and Michel, Vincent and Thirion, Bertrand and Grisel,
    Olivier and Blondel, Mathieu and Prettenhofer, Peter and Weiss,
    Ron and Dubourg, Vincent and others},
    journal={The Journal of Machine Learning Research},
    volume={12},
    pages={2825--2830},
    year={2011},
    publisher={JMLR. org}
}
@article{Kriegeskorte_2006, title=...
...
...
```

Contribute HOWTO 101.1: Use in your software

- 1 copy duecredit/stub.py to your codebase, e.g.

```
wget -q -O /path/tomodule/yourmodule/due.py \
https://raw.githubusercontent.com/
duecredit/duecredit/master/duecredit/stub.py
```

- 2 Then use duecredit import due and necessary entries in your code as

```
from .due import due, Doi
```

to provide reference for the entire module just use e.g.

```
due.cite(Doi("1.2.3/x.y.z"), description="Solves all your problems",
          path="magicpy")
```

To provide a reference for a function or a method, use dcite decorator

```
@due.dcite(Doi("1.2.3/x.y.z"), description="Resolves constipation ...
def pushit():

    ..."
```

Contribute HOWTO 101.2: Inject for other modules

Example: duecredit/injections/mod_scipy.py

```
from ..entries import Doi, BibTeX, Url
def inject(injector):
    injector.add('scipy', None, BibTeX("""
        @Misc{JOP+01,
            ...
        }"""),
        description="Scientific tools library",
        tags=['implementation'])

    ...
    injector.add('scipy.cluster.hierarchy', 'linkage', BibTeX("""
        @article{ward1963hierarchical,
            ...
        }"""),
        conditions={(1, 'method'): {'ward'}},
        description="Ward hierarchical clustering",
        min_version='0.4.3',
        tags=['reference'])
    ...
    ...
```

Get involved!

CONTRIBUTE

- Use in your software and to collect references for your analysis scripts
- Report bugs, send pull requests/patches
- Provide support for other languages/environments

Matlab/Octave <https://github.com/duecredit/duecredit/issues/20>

Java, R, C/C++, ... **You?**

- Spread the word

WE NEED HELP!



Lessons learned or
How to make any analysis open, re-executable and
results reproducible?

HOWTO

- Make sure you could (potentially) share your data openly
- Establish efficient code and data management to retain full history of changes to be shared (some time) publicly
- Test (unit-, regression-) your analysis and assumptions
- (Re)use public datasets (and publish your own!)
- Collect information about your computation environment and analysis
- Create your own shareable (legally!) virtualized/containerized computational environments from unambiguous specification
- Automate your analysis as much as possible

HOWTO

- Make sure you could (potentially) share your data openly
- Establish efficient code and data management to retain full history of changes to be shared (some time) publicly
- Test (unit, regression) your analysis and assumptions
- **If originally might have sounded Utopian?**
- You saw now that Neuroimaging community is blessed with great initiatives and tools to make it actually possible
- Create your own standardized/containerized computational environments from unambiguous specification
- Automate your analysis as much as possible

HOWTO

- Make sure you could (potentially) share your data openly
 - [Open Brain Consent](#)
- Establish efficient code and data management to retain full history of changes to be shared (some time) publicly
 - [BIDS](#), [ReproIn](#), [DataLad](#) (git, git-annex), [GitHub](#), ...
- Test (unit-, regression-) your analysis and assumptions
 - [PyTest](#), [MOxUnit](#), [Bats](#) (for bash), [CTest](#), [Travis-CI](#), [CircleCI](#), ...
- (Re)use public datasets (and publish your own!)
 - [datasets.datalad.org](#), [OpenfMRI/OpenNeuro](#), [NITRC-IR](#), [INDI](#), ...
- Collect information about your computation environment and analysis
 - [ReproZip](#), [NICEMAN](#), [DueCredit](#), ...
- Create your own shareable (legally!) virtualized/containerized computational environments from unambiguous specification
 - [NeuroDebian](#), [NITRC-CE](#), [NeuroDocker](#), ...
- Automate your analysis as much as possible
 - [PyMVPA](#), [nipype](#), [datalad run](#), [rerun](#) and [containers-run](#), ...

Open, Reproducible, and **Correct** Science is in reach!

Free and Open Source Software,
Data sharing,

Good code and data management practices
are helping to make it happen

Brain Download:



iz compltes.

References

- Gorgolewski, K. J., Auer, T., Calhoun, V. D., Craddock, R. C., Das, S., Duff, E. P., Flandin, G., Ghosh, S. S., Glatard, T., Halchenko, Y. O., Handwerker, D. A., Hanke, M., Keator, D., Li, X., Michael, Z., Maumet, C., Nichols, B. N., Nichols, T. E., Pellman, J., Poline, J.-B., Rokem, A., Schaefer, G., Sochat, V., Triplett, W., Turner, J. A., Varoquaux, G., and Poldrack, R. A. (2016). The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. *Scientific Data*, 3:160044.
- Halchenko, Y. O. (2012). Incorrect probabilities in Harvard-Oxford-sub left hemisphere. [Retrieved 11-Mar-2013].
- Halchenko, Y. O. and Hanke, M. (2012). Open is not enough. Let's take the next step: An integrated, community-driven computing platform for neuroscience. *Frontiers in Neuroinformatics*, 6(00022). PMC3458431.
- Halchenko, Y. O. and Hanke, M. (2015). Four aspects to make science open "by design" and not as an after-thought. *GigaScience*, 4(31).
- Halchenko, Y. O., Hanke, M., Haxby, J. V., Hanson, S. J., and Herrmann, C. S. (2013). Transmodal analysis of neural signals. *ArXiv e-prints*.
- Hanke, M. and Halchenko, Y. O. (2011). Neuroscience runs on GNU/Linux. *Front. Neuroinform.*, 5:8. PMC3133852.
- Hanke, M., Halchenko, Y. O., Sederberg, P. B., Hanson, S. J., Haxby, J. V., and Pollmann, S. (2009a). PyMVPA: A Python toolbox for multivariate pattern analysis of fMRI data. *Neuroinformatics*, 7(1):37–53. PMC2664559.
- Hanke, M., Halchenko, Y. O., Sederberg, P. B., Olivetti, E., Fründ, I., Rieger, J. W., Herrmann, C. S., Haxby, J. V., Hanson, S. J., and Pollmann, S. (2009b). PyMVPA: A unifying approach to the analysis of neuroscientific data. *Frontiers in Neuroinformatics*, 3(3). PMC2638552.
- Kohler, P. J., Fogelson, S. V., Reavis, E. A., Meng, M., Guntupalli, J. S., Hanke, M., Halchenko, Y. O., Connolly, A. C., Haxby, J. V., and Tse, P. U. (2013). Pattern classification precedes region-average hemodynamic response in early visual cortex. *Neuroimage*, 78C:249–260. PMID: 23587693.
- Kriegeskorte, N., Goebel, R., and Bandettini, P. (2006). Information-based functional brain mapping. *Proceedings of the National Academy of Sciences of the USA*, 103:3863–3868.
- Poldrack, R. A., Halchenko, Y. O., and Hanson, S. J. (2009). Decoding the large-scale structure of brain function by classifying mental states across individuals. *Psychological Science*, 20(11):1364–1372. PMC2935493.
- Rohlfing, T. (2013). Incorrect icbm-dti-81 atlas orientation and white matter labels. *Frontiers in Neuroscience*, 7(4).
- Trautmann, E., Ray, L., and Lever, J. (2009). Development of an autonomous robot for ground penetrating radar surveys of polar ice. In *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 1685–1690.