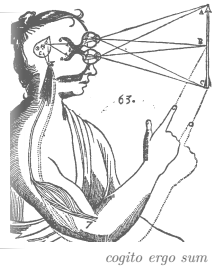
CCN Center for Cognitive Neuroscience at Dartmouth



CON Center for Open Neuroscience

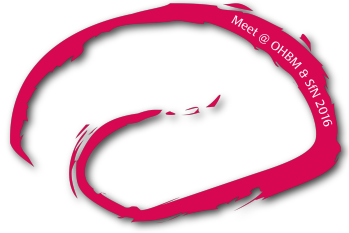
Tools from Center for Open Neuroscience: be Open by Design

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

August 2018


http://datalad.org

http://www.pymvpa.org



http://Neuro.Debian.net

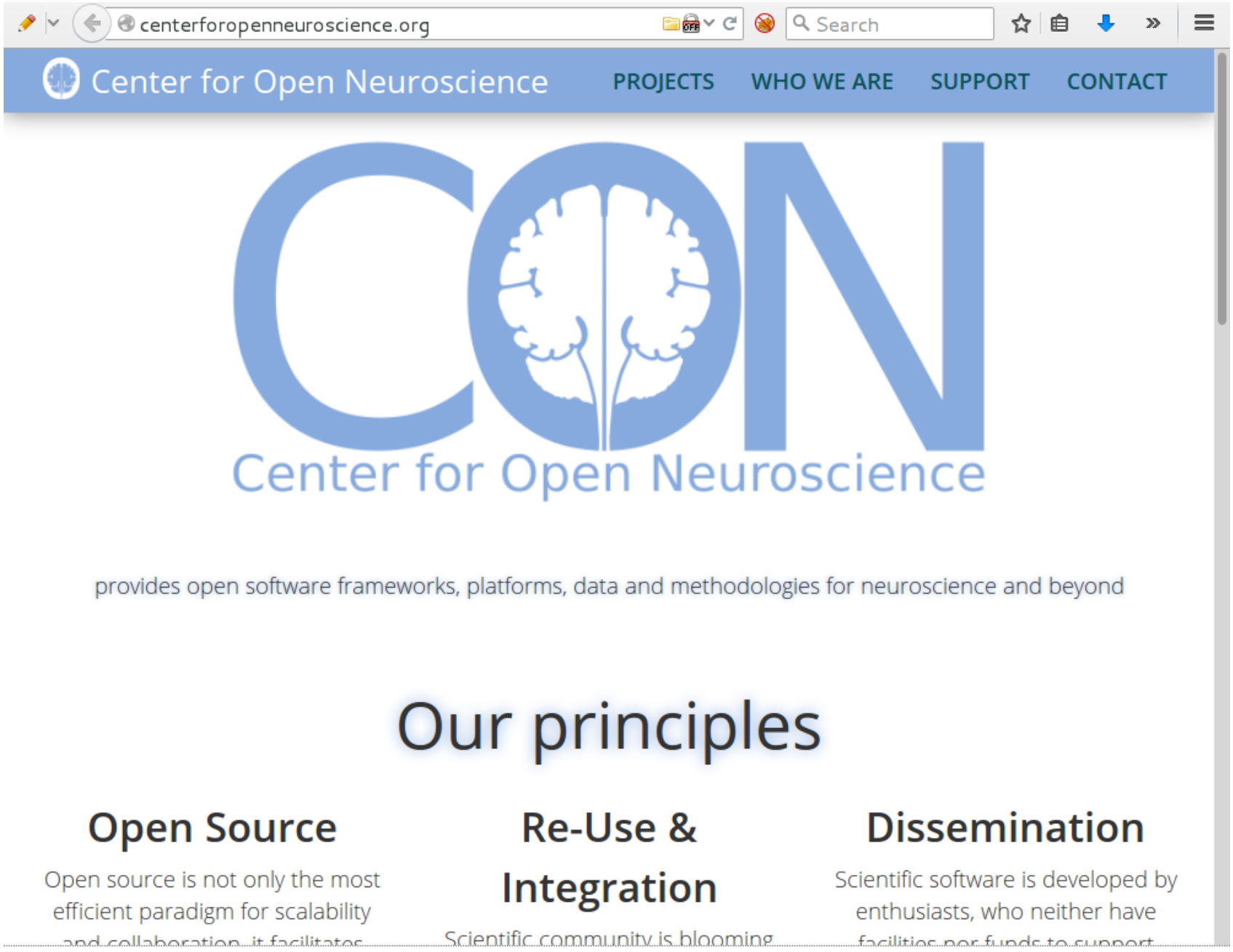
Slides

http://neuro.debian.net/_files/mind2018-contools-halchenko.pdf

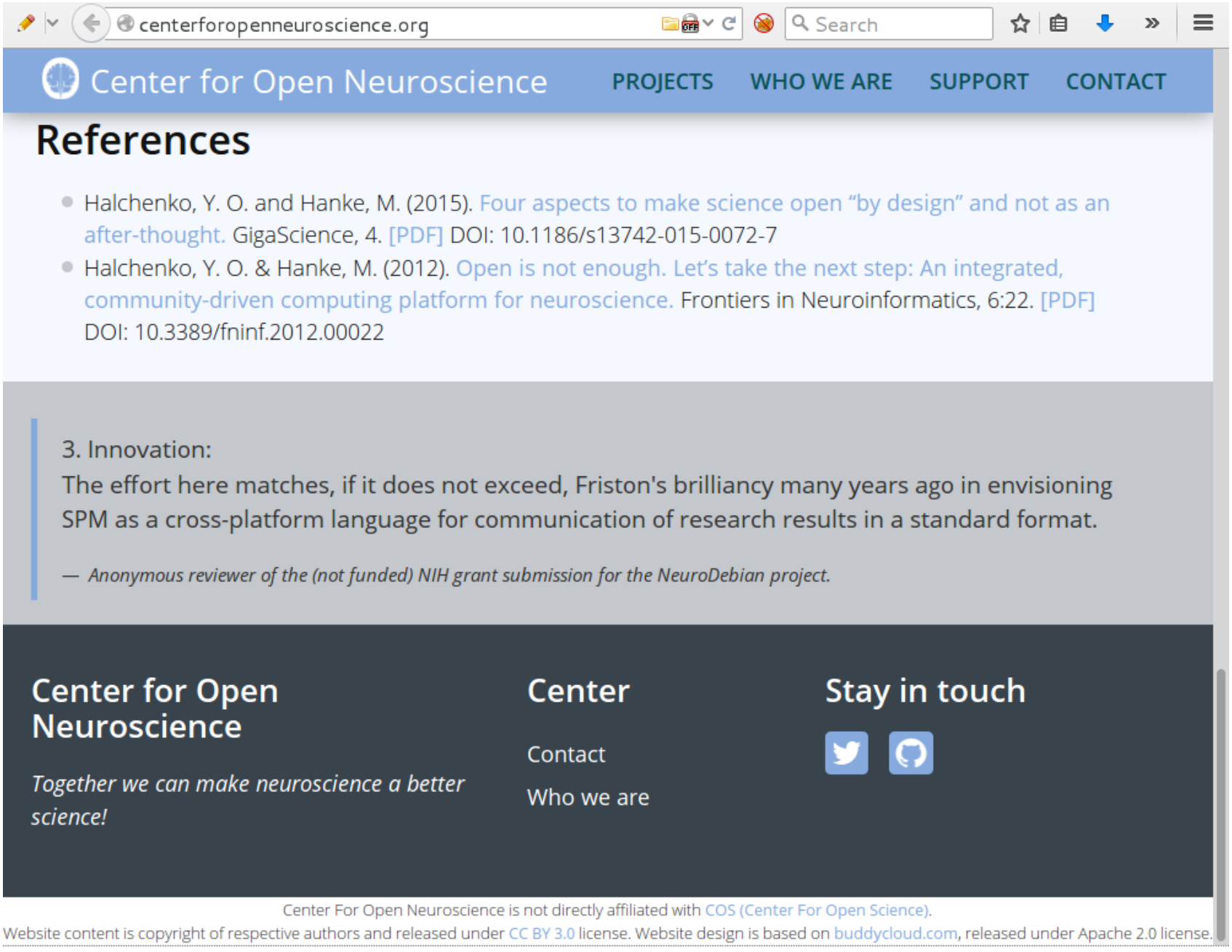

nutcase

Overarching goal:
To make neuroscience a better science

CenterForOpenNeuroscience.org



CenterForOpenNeuroscience.org



Who we are/Acknowledgments

centerforopenneuroscience.org/whoweare#michael_hanke_

Center for Open Neuroscience

Centroids

Collaborators

Michael Hanke

Nikolaas N. Oosterhof

Matthew Brett

Joey Hess

Benjamin Poldrack

Emeritus

Collaborating projects

Partners

Michael Hanke

University of Magdeburg, Germany

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

Joey Hess

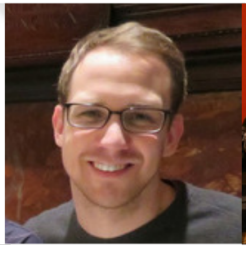
Independent Guru

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

Other Gurus

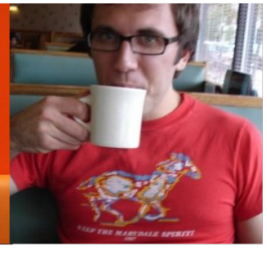
Benjamin

Kyle



Alex

Jason



Interns

Debanjum

Gergana

Who we are/Acknowledgments

centerforopenneuroscience.org/whoweare#affiliated_fa...

Center for Open Neuroscience

PROJECTS WHO WE ARE ENGAGE SUPPORT

Centroids

Collaborators

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

Affiliated Faculty

Collaborators in training

Emeritus

Collaborating projects

Partners

Affiliated Faculty

Luke Chang

Jeremy Rothman Manning

Matthijs (Matt) van der Meer

Who we are/Acknowledgments

“Open by Design” 101

- 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.
- **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
  - Do not reinvent the wheel – use and contribute to existing projects
  - Automate (and script it) whenever possible

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)?

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 undergraduation 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

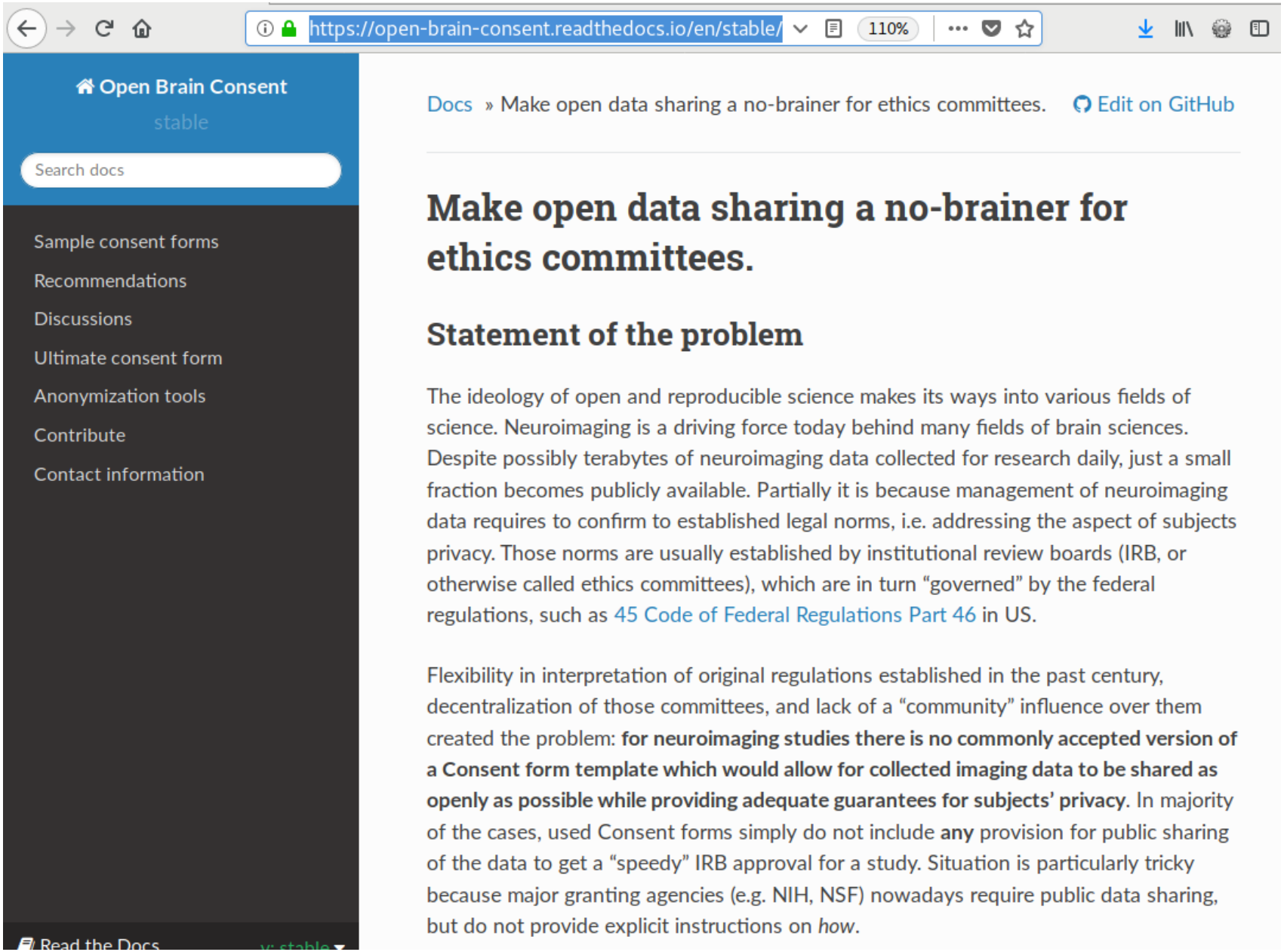
The reason this data is not available is that questions were raised about whether sharing the data complied with United States government rules. It's been taking much longer than expected to resolve this. An update will be posted here when the data is made available again (which could be soon, but might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

Open Brain Consent

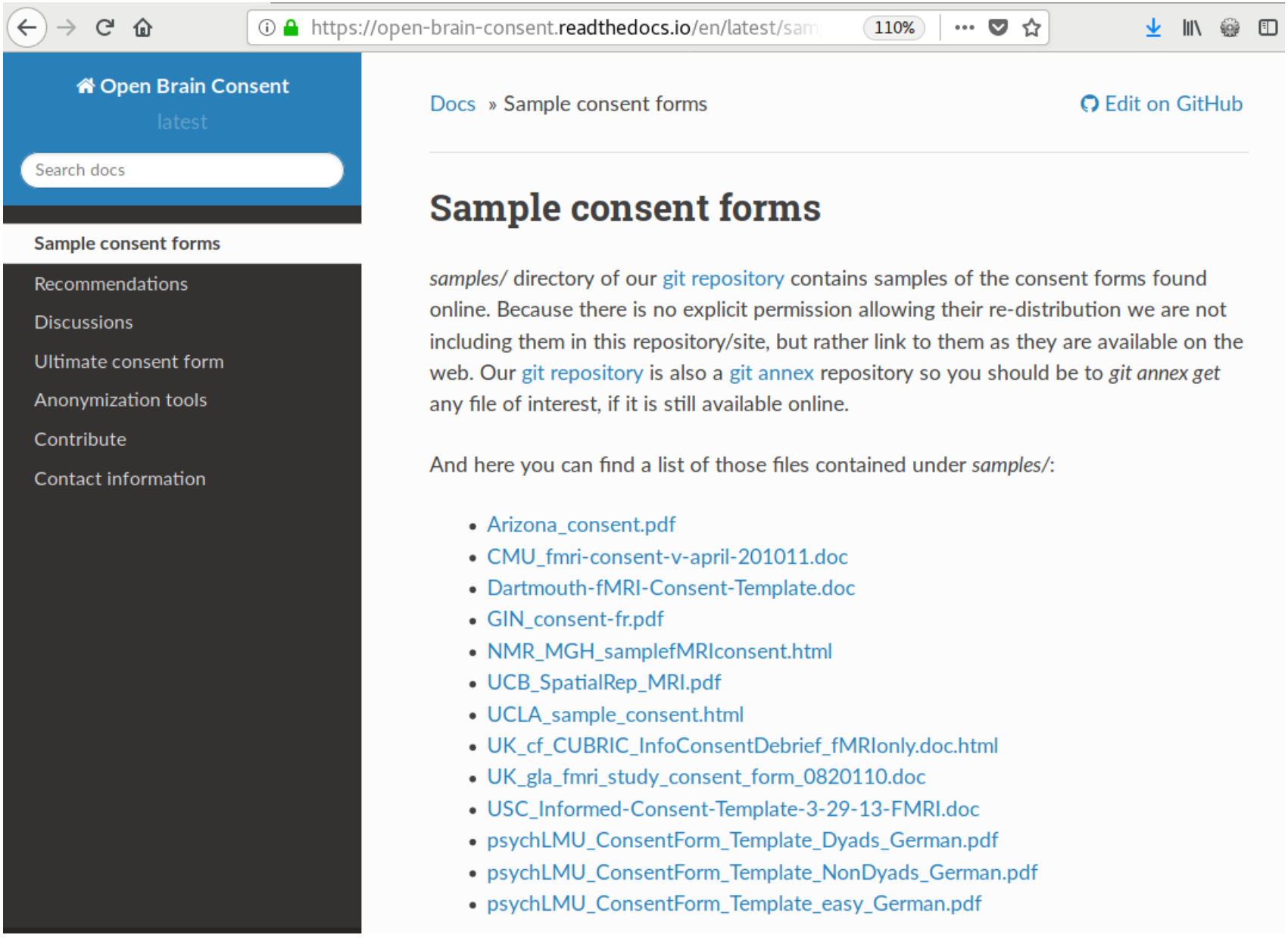
YOUR LOGO CONTRIBUTION HERE

open-brain-consent.readthedocs.io

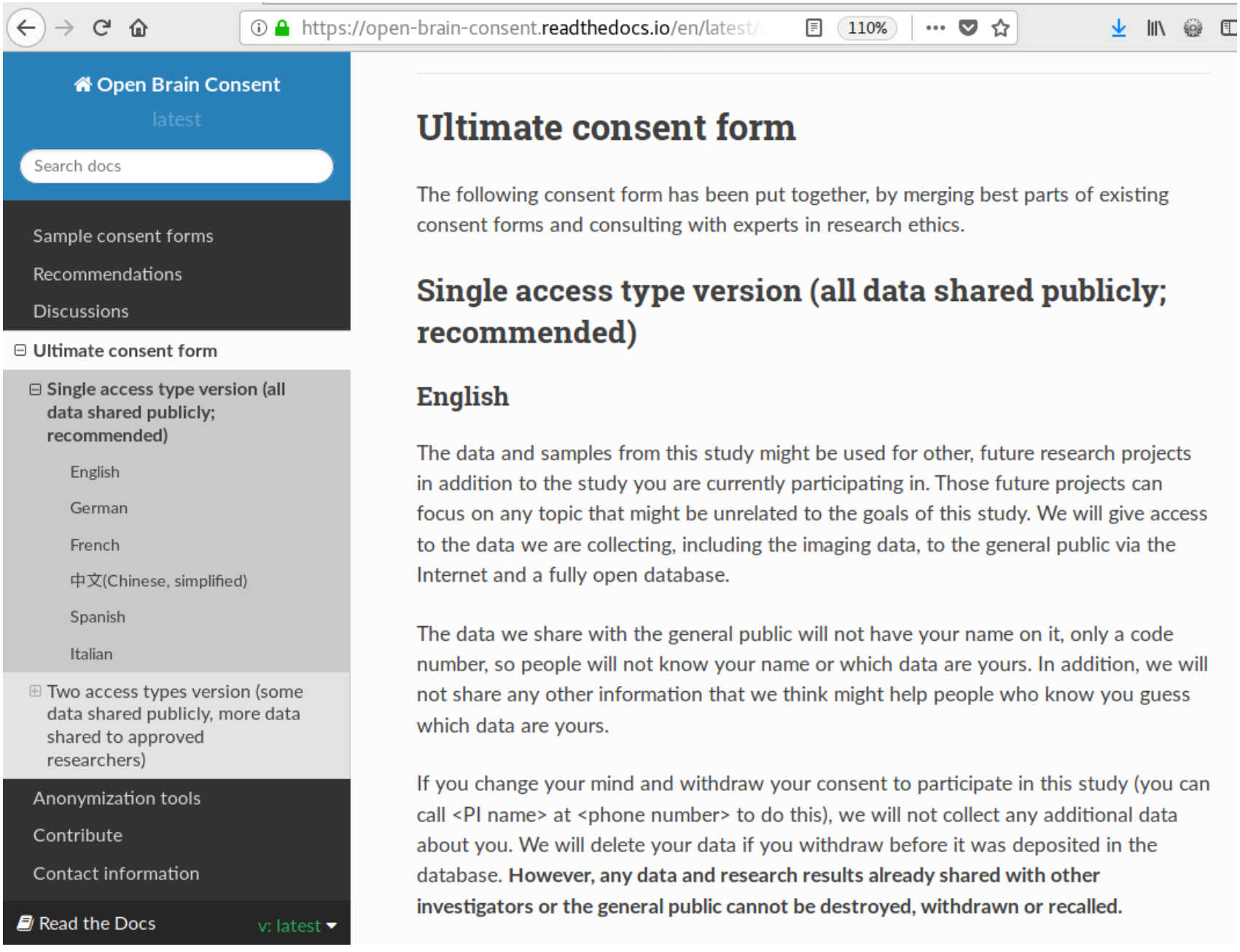
Consents: Federally regulated but locally “managed”



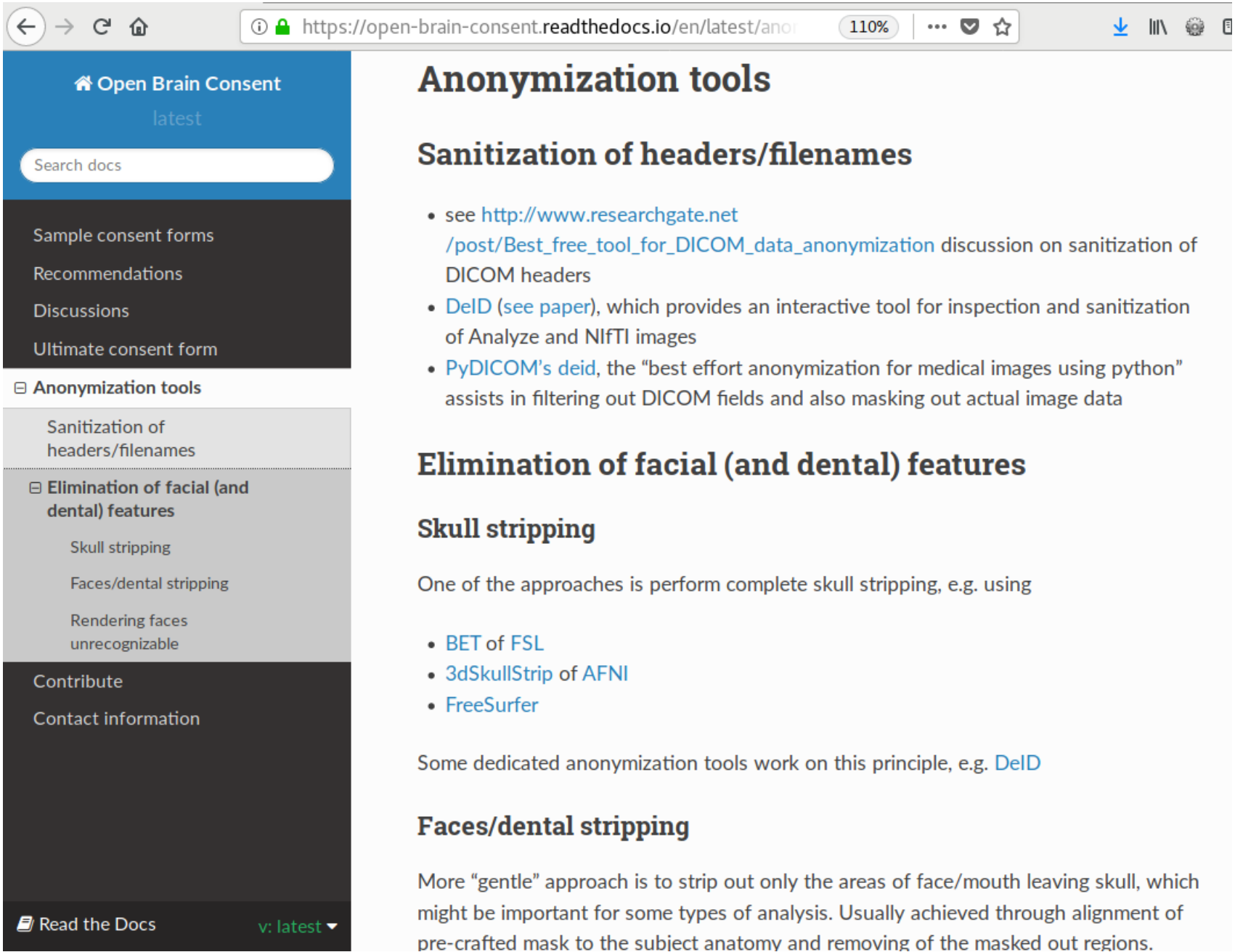
Open Brain Consent: Sample consent forms



Open Brain Consent: Ultimate consent form



Open Brain Consent: Tools



Get involved!

- 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”!

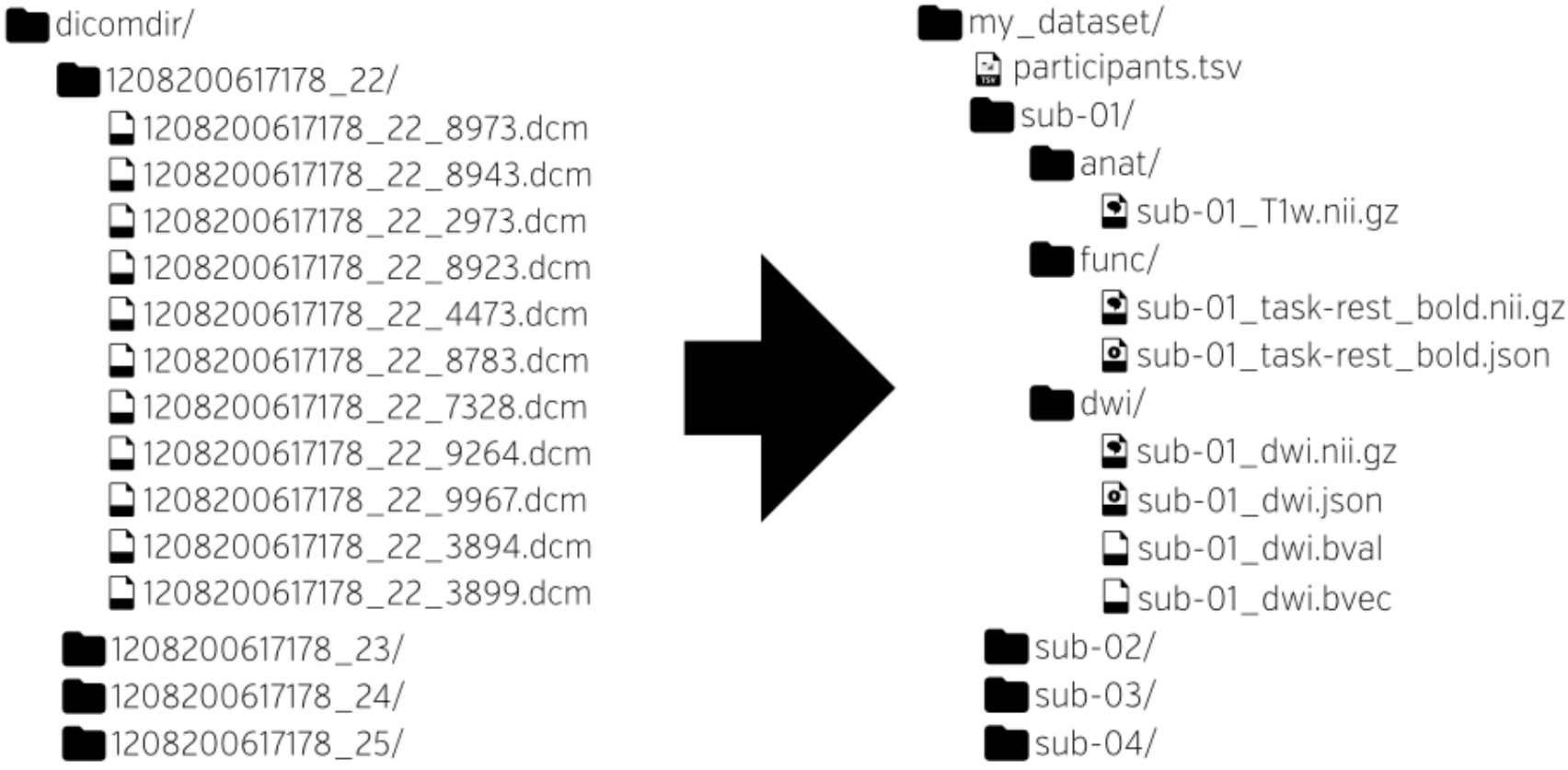
BIDS (Brain Imaging Data Structure)
is the next great thing after NIfTI
but
I have no time to prepare those “BIDS datasets”!

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!

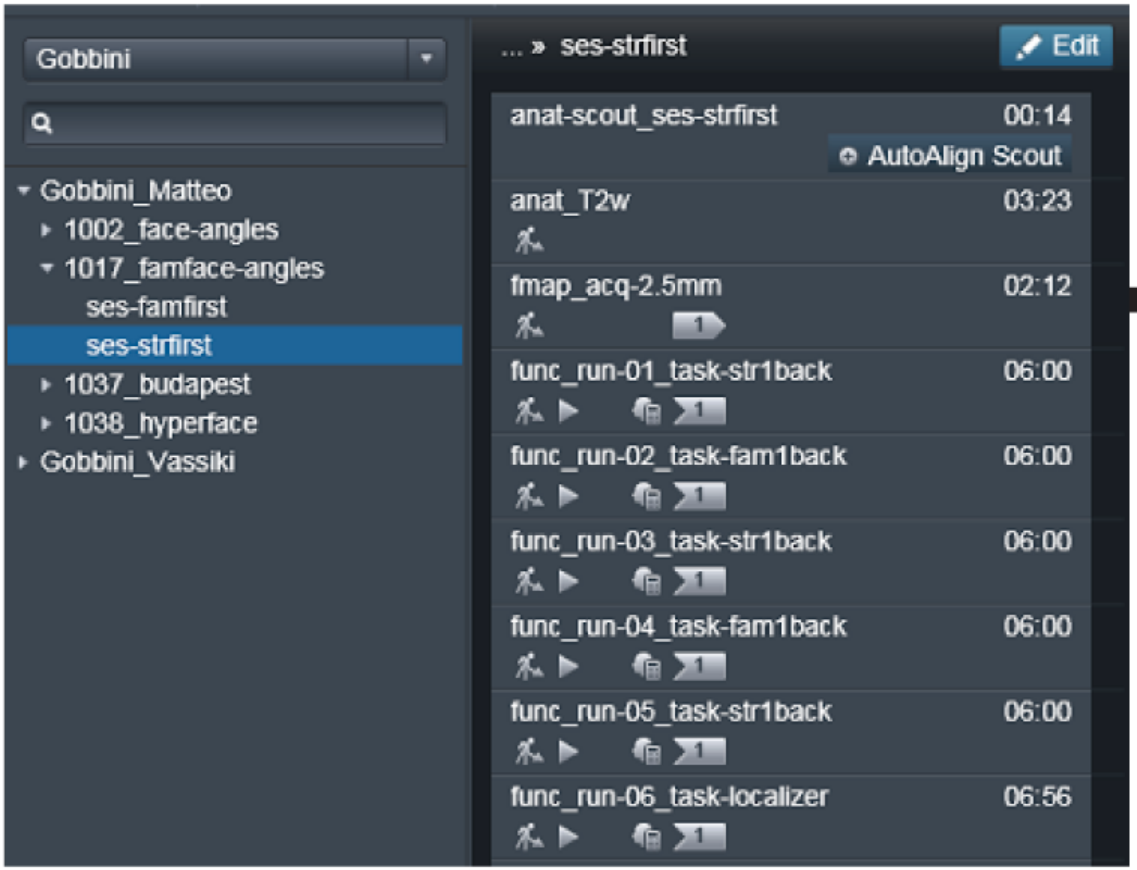
- BIDS is both human- and machine- friendly
- BIDS compliance could be automatically verified using bids-validator
- PyBIDS *etc.* can assist scripting use of BIDS datasets
- BIDS-apps (such as mriqc, fmriprep, etc) provide a turnkey solution for BIDS datasets

---

github.com/INCF/bids-validator
bids-apps.neuroimaging.io

ReproIn: “BIDS” at the scanner console

Scanner



DICOM

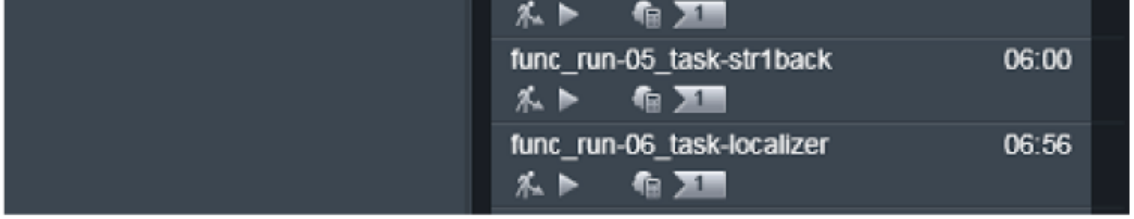
```
001-anat-scout_ses-strfirst
005-anat_T2w
  000001.dcm
  000002.dcm
  ...
006-fmap_acq-2.5mm
007-fmap_acq-2.5mm
008-func_run-01_task-str1back
011-func_run-01_task-str1back
018-func_run-02_task-fam1back
025-func_run-03_task-str1back
032-func_run-04_task-fam1back
039-func_run-05_task-str1back
046-func_run-06_task-localizer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

$ heudiconv

Datalad

BIDS

```
anat
  sub-sid000005_ses-strfirst_T2w.json
```



$ heudiconv

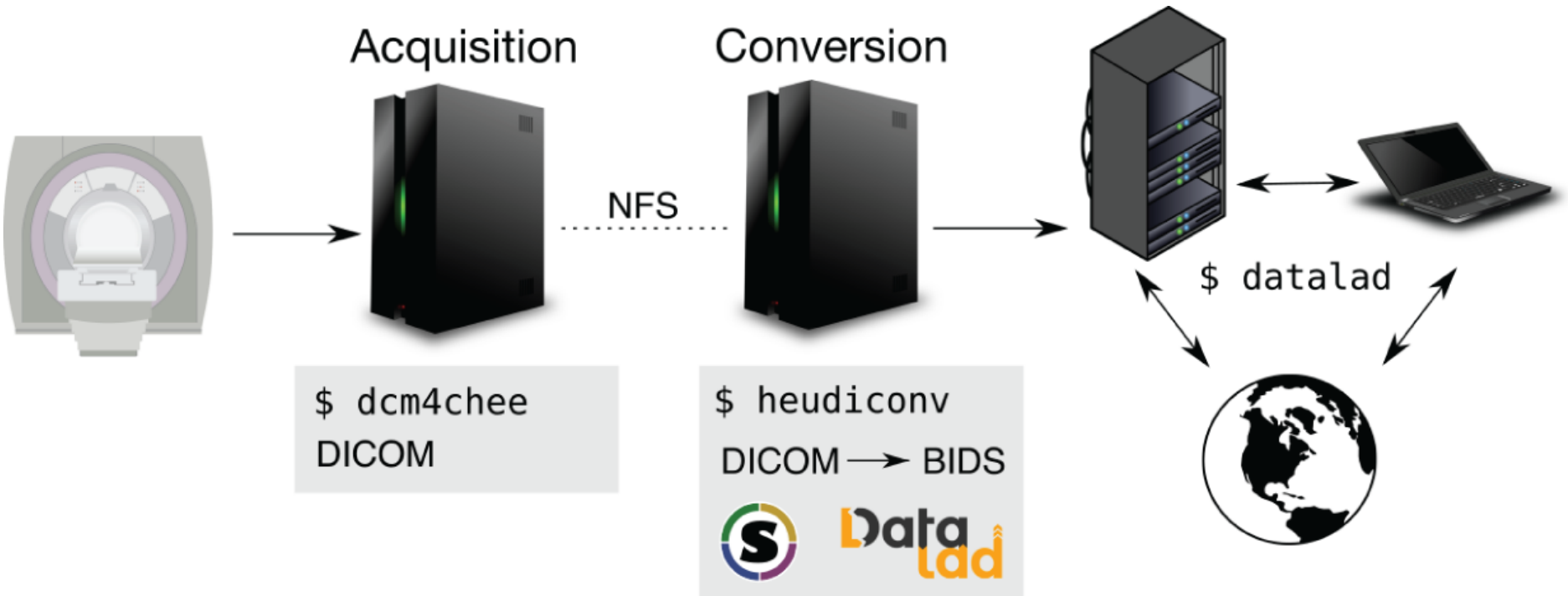
Datalad

BIDS

```
anat
  sub-sid000005_ses-strfirst_T2w.json
  sub-sid000005_ses-strfirst_T2w.nii.gz
fmap
  sub-sid000005_ses-strfirst_acq-25mm_magnitude1.json
  sub-sid000005_ses-strfirst_acq-25mm_magnitude1.nii.gz
  ...
func
  sub-sid000005_ses-strfirst_task-fam1back_run-02_bold.json
  sub-sid000005_ses-strfirst_task-fam1back_run-02_bold.nii.gz
  sub-sid000005_ses-strfirst_task-fam1back_run-02_events.tsv
  ...
sub-sid000005_ses-strfirst_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",
...
```

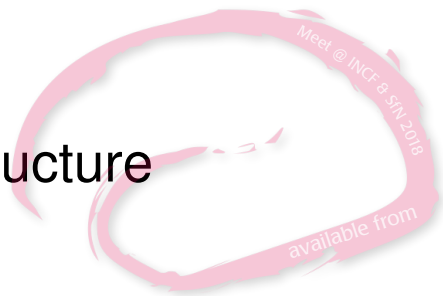
ReproIn: Pile of DICOMs → BIDS



ReproIn heuristic is part of github.com/nipy/heudiconv

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



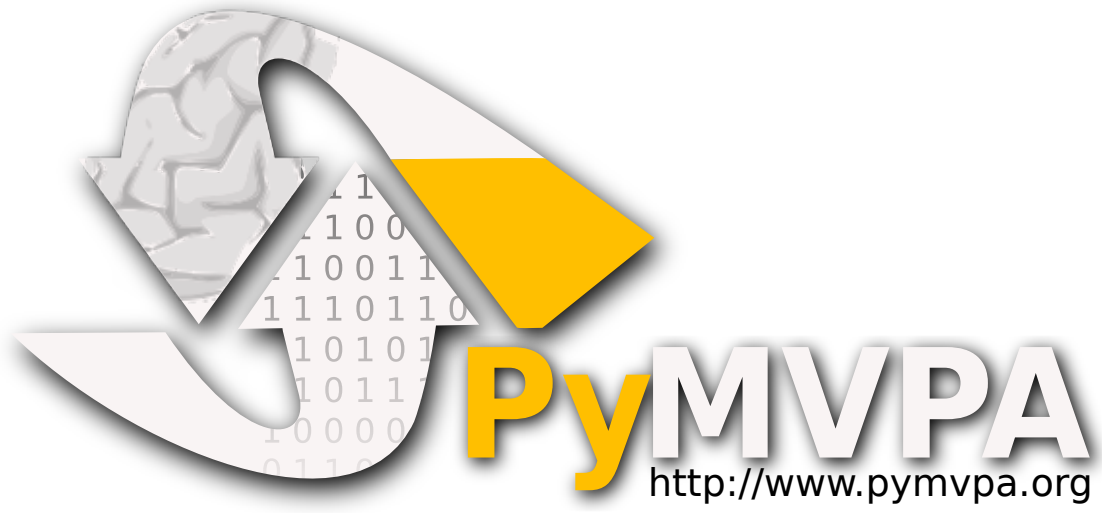
Get involved!

- 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*
  $\Rightarrow$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*
  $\Rightarrow$Concise scripting interface in Python and command line
- Thoroughly documented
  $\Rightarrow$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*
  $\Rightarrow$Concise scripting interface in Python and command line
- Thoroughly documented
  $\Rightarrow$Tutorial, user manual, and examples (IPython notebooks)
- Extensible
  $\Rightarrow$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*
  $\Rightarrow$Concise scripting interface in Python and command line
- Thoroughly documented
  $\Rightarrow$Tutorial, user manual, and examples (IPython notebooks)
- Extensible
  $\Rightarrow$Modular architecture connecting extensions in multiple languages
- Portable
  $\Rightarrow$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-ce
- Thorou
- Extensi
- Portabl
- Reliable
- Free ar

nmand line

notebooks)

extensions

languages

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

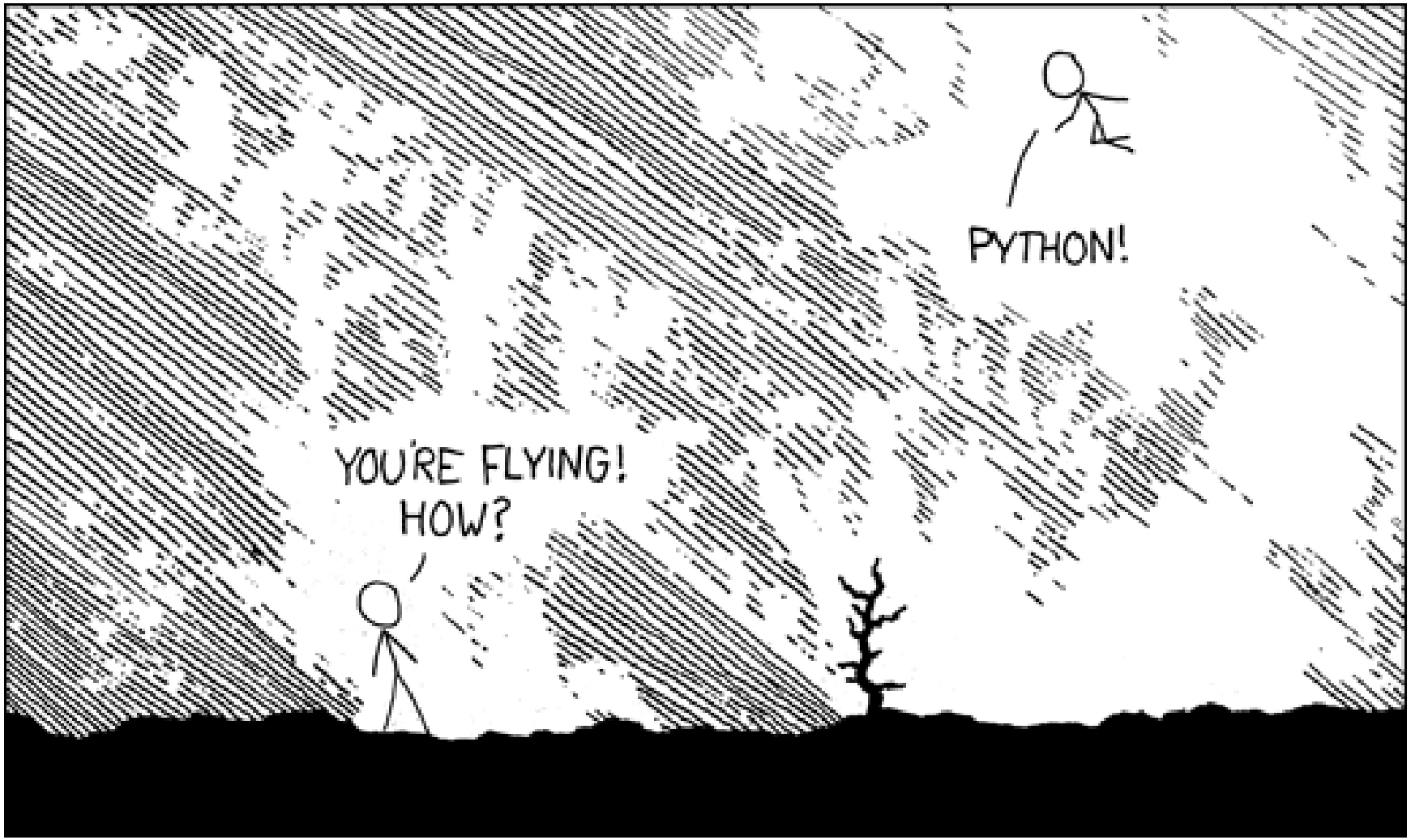
Hanke, M., Halchenko, Y. O., Sederberg, P. B., Hanson, S. J., Haxby, J. V., and Pollmann, S. (2009a). PyMVPA: A Python toolbox for multivariate pattern analysis of fMRI data. *Neuroinformatics*, 7(1):37–53. PMC2664559

PyMVPA features

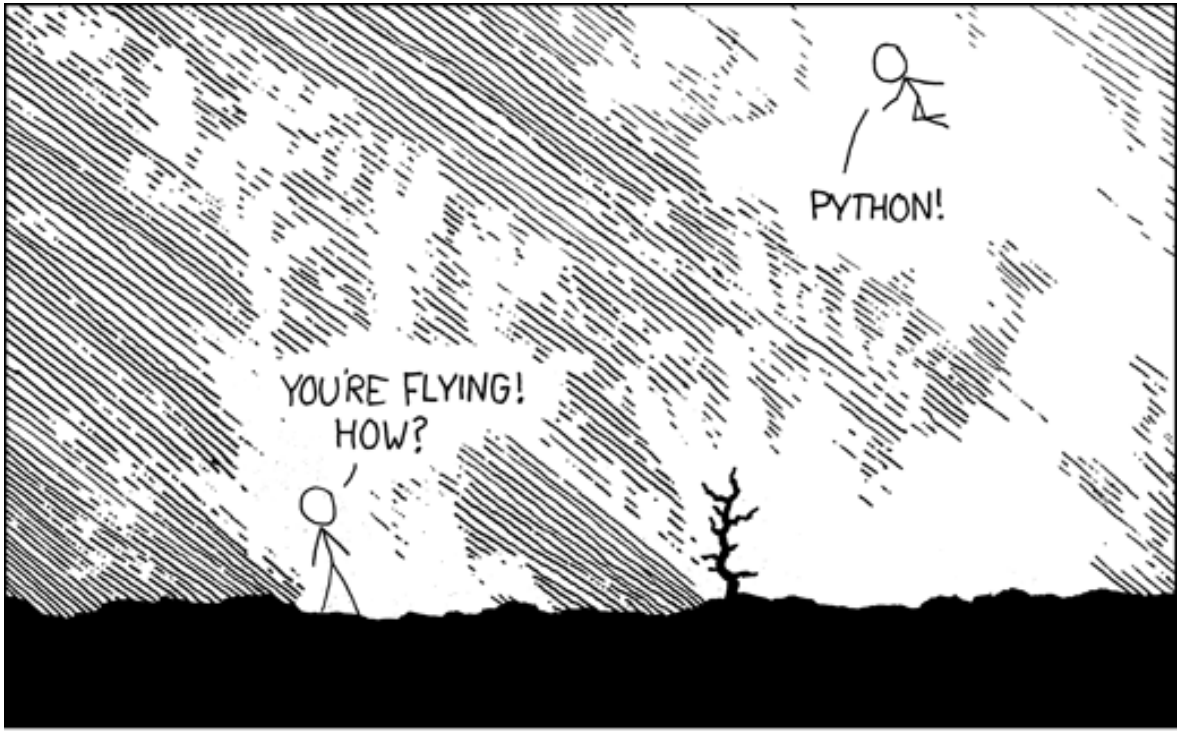
- User-centered *programmability*
  $\Rightarrow$Concise scripting interface in Python and command line
- Thoroughly documented
  $\Rightarrow$Tutorial, user manual, and examples (IPython notebooks)
- Extensible
  $\Rightarrow$Modular architecture connecting extensions in multiple languages
- Portable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

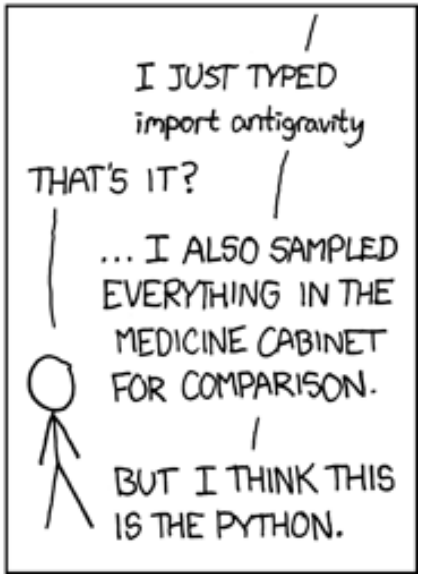


Python


PYTHON!
YOU'RE FLYING! HOW?


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

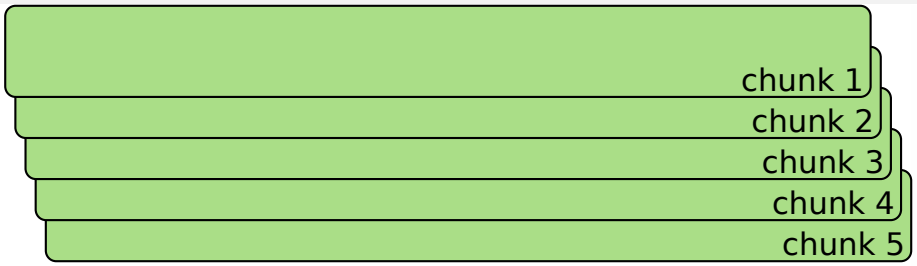

I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?


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”

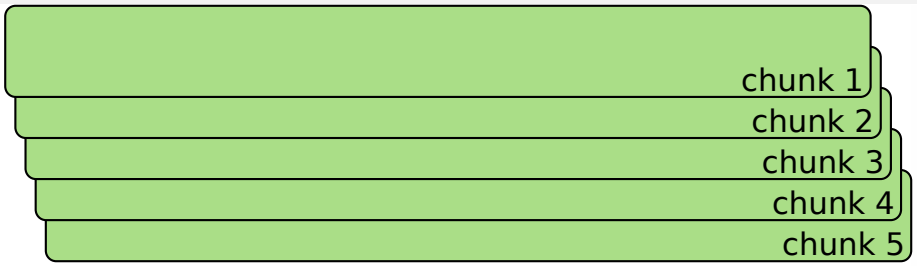
```
from mvpa2.suite import *
```

Analysis example : Datasets

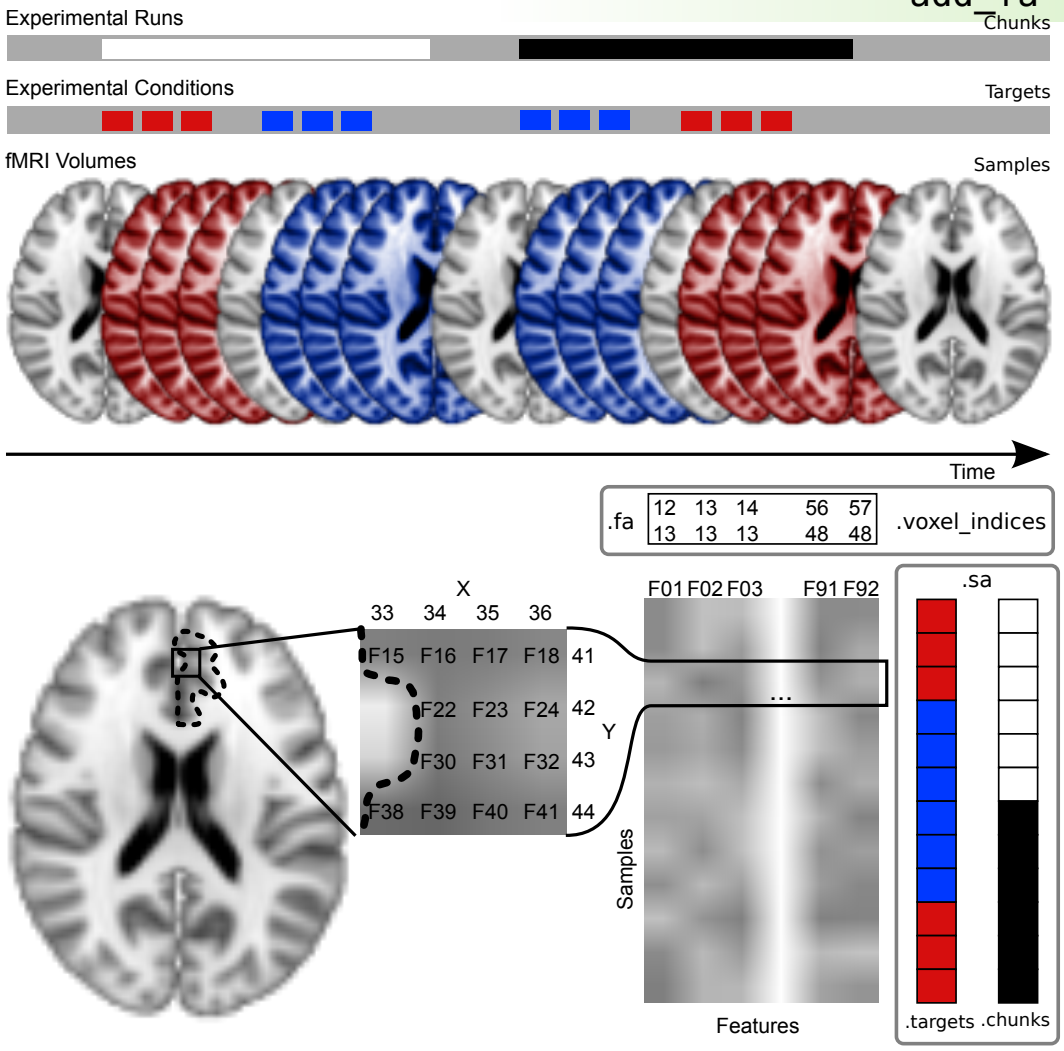



```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets,
                  chunks=attr.chunks,
                  mask='mask_brain.nii.gz',
                  add_fa={'vt':'vt.nii.gz'})
```

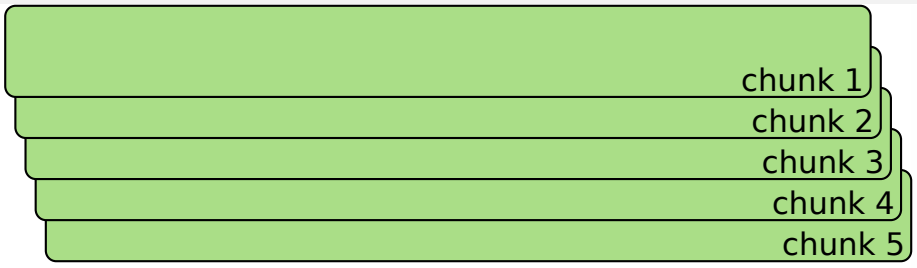
Analysis example : Datasets




```
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'})
```



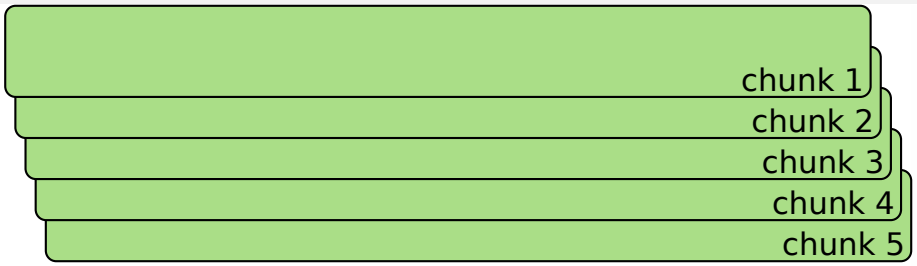
Analysis example : Datasets




```
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]
```

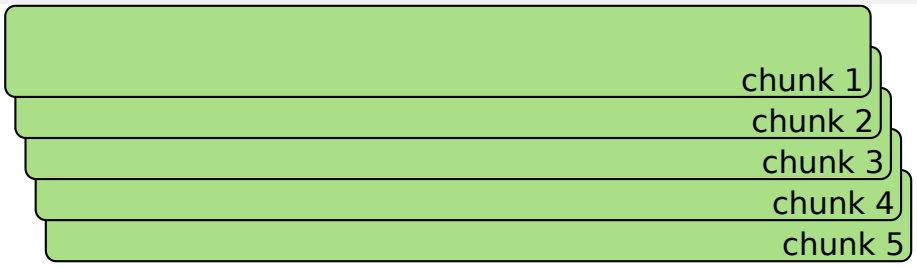
Analysis example : Datasets




```
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()
```

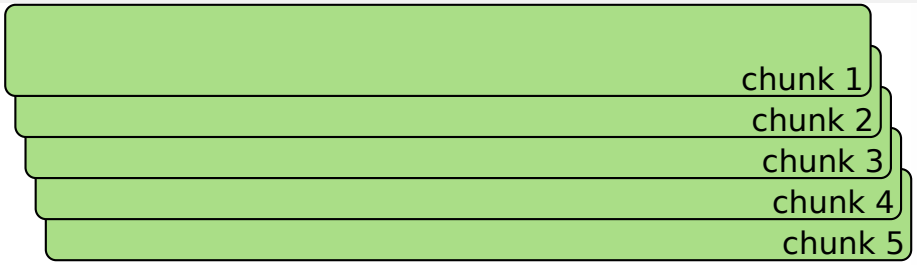
Analysis example : Datasets




```
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()
```

```
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
```

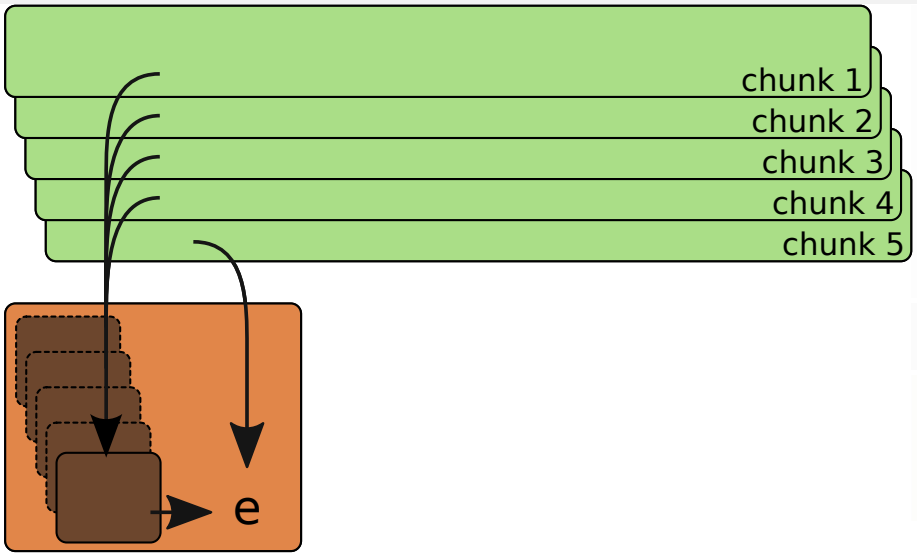
Analysis example: Classification




```
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()
```

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'})
```

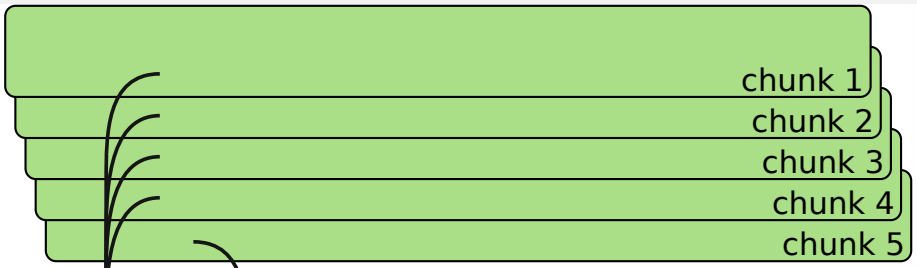
```
clf = LinearCSVMC()
```

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

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

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

Analysis example: Classification (Everyone matters)




```
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'})
```

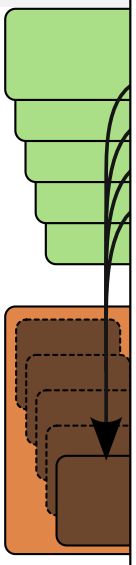
```
clf = LinearCSVMC()
```

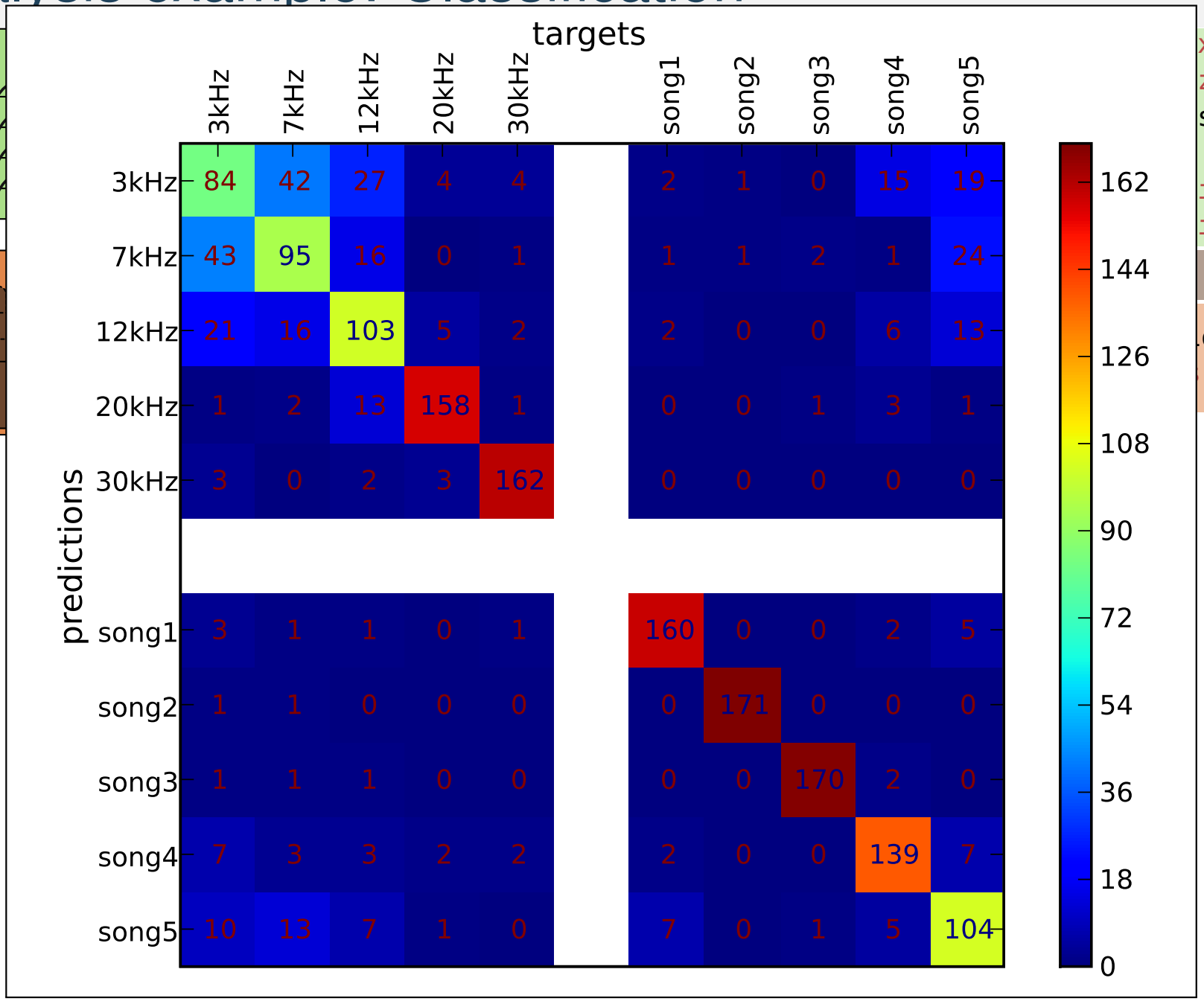
```
oner(),
'])
```

-------.	3kHz	7kHz	12kHz	20kHz	30kHz	song1	song2	song3	song4	song5							
predict.\targets	38	39	40	41	42	43	44	45	46	47							
`------	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	P'	N'	FP	FN	PPV	NPV	TPR
3kHz / 38	84	42	27	4	4	2	1	0	15	19	198	1351	114	90	0.42	0.93	0.48
7kHz / 39	43	94	16	0	1	1	1	2	1	24	183	1331	89	80	0.51	0.94	0.54
12kHz / 40	21	16	103	5	2	2	0	0	6	13	168	1312	65	70	0.61	0.95	0.6
20kHz / 41	1	2	13	158	1	0	0	1	3	1	180	1202	22	15	0.88	0.99	0.91
30kHz / 42	3	0	2	3	162	0	0	0	0	0	170	1194	8	11	0.95	0.99	0.94
song1 / 43	3	1	1	0	1	160	0	0	2	5	173	1199	13	14	0.92	0.99	0.92
song2 / 44	1	1	0	0	0	0	171	0	0	0	173	1176	2	2	0.99	1	0.99
song3 / 45	1	1	1	0	0	0	0	170	2	0	175	1179	5	4	0.97	1	0.98
song4 / 46	7	3	3	2	2	2	0	0	139	7	165	1240	26	34	0.84	0.97	0.8
song5 / 47	10	14	7	1	0	7	0	1	5	104	149	1310	45	69	0.7	0.95	0.6
Per target:	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----							
P	174	174	173	173	173	174	173	174	173	173							
N	1560	1560	1561	1561	1561	1560	1561	1560	1561	1561							
TP	84	94	103	158	162	160	171	170	139	104							
TN	1261	1251	1242	1187	1183	1185	1174	1175	1206	1241							
Summary \ Means:	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	173	1249	38	39	0.78	0.97	0.78
ACC	0.78																
ACC%	77.57																
# of sets	8																

Hanke&**Halchenko** et al., Frontiers in Neuroinformatics, 2009 (Suppl.)

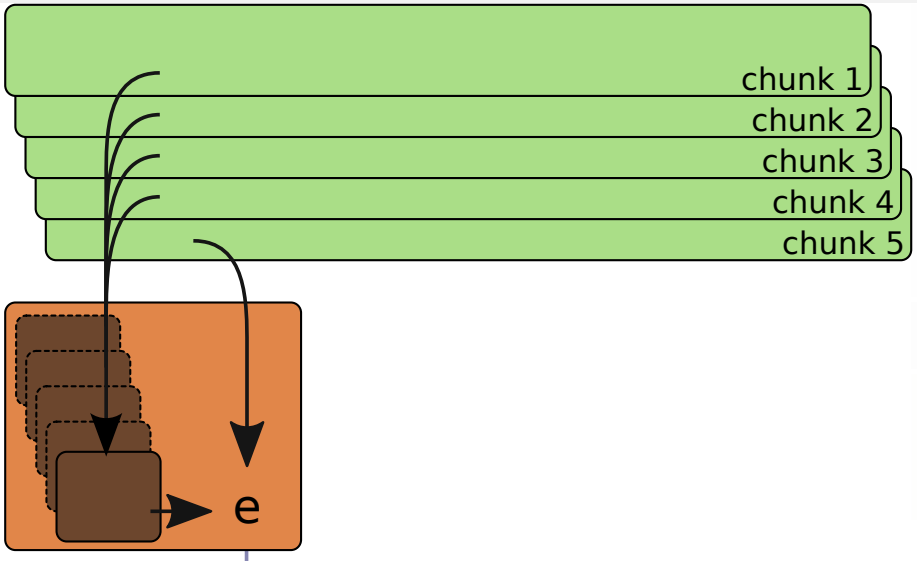
Analysis example: Classification

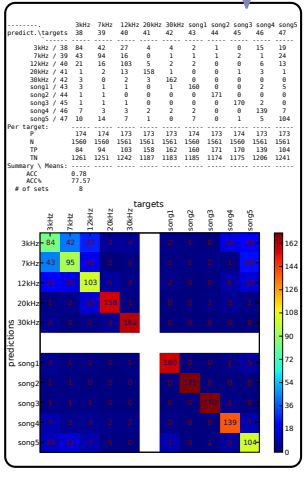





```
xt')
z',
s,

i.gz',
i.gz'})

oner(),
'])
```

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'})
```

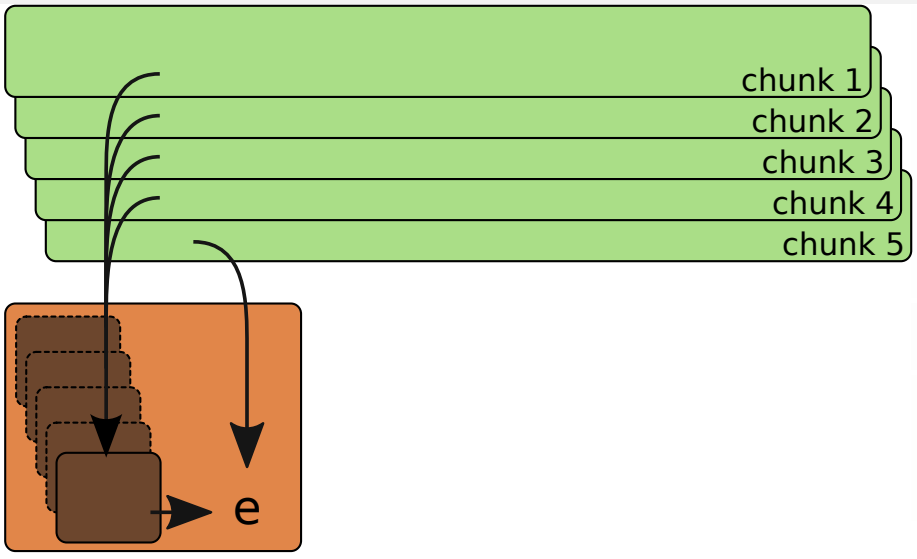
```
clf = LinearCSVMC()
```

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

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

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

Analysis example: Searchlights



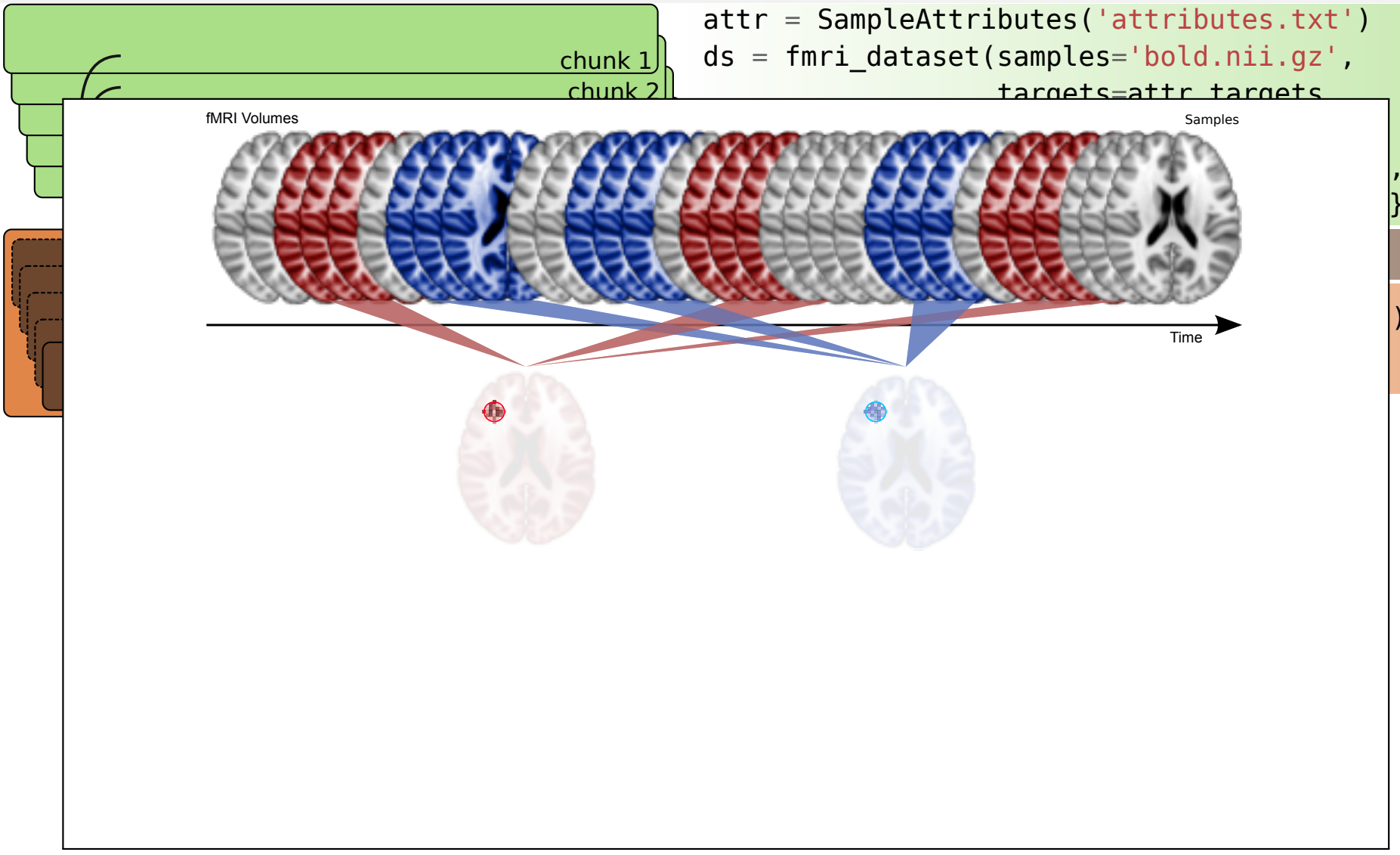

```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets,
                  chunks=attr.chunks,
                  mask='mask_brain.nii.gz',
                  add_fa={'vt':'vt.nii.gz'})
```

```
clf = LinearCSVMC()
```

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

Analysis example: Searchlights

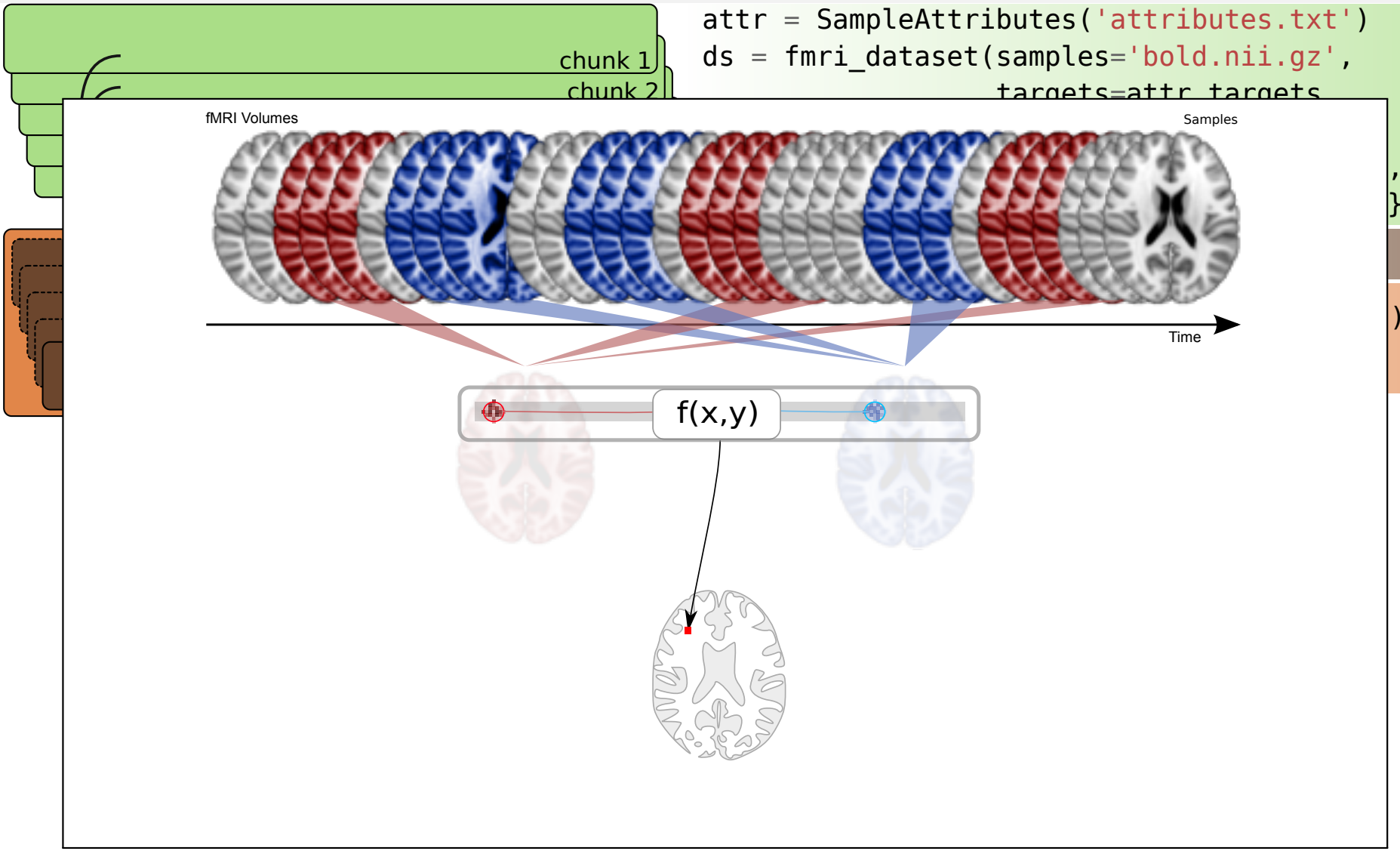
```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets
                  ,
                  })
                  ),
```



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

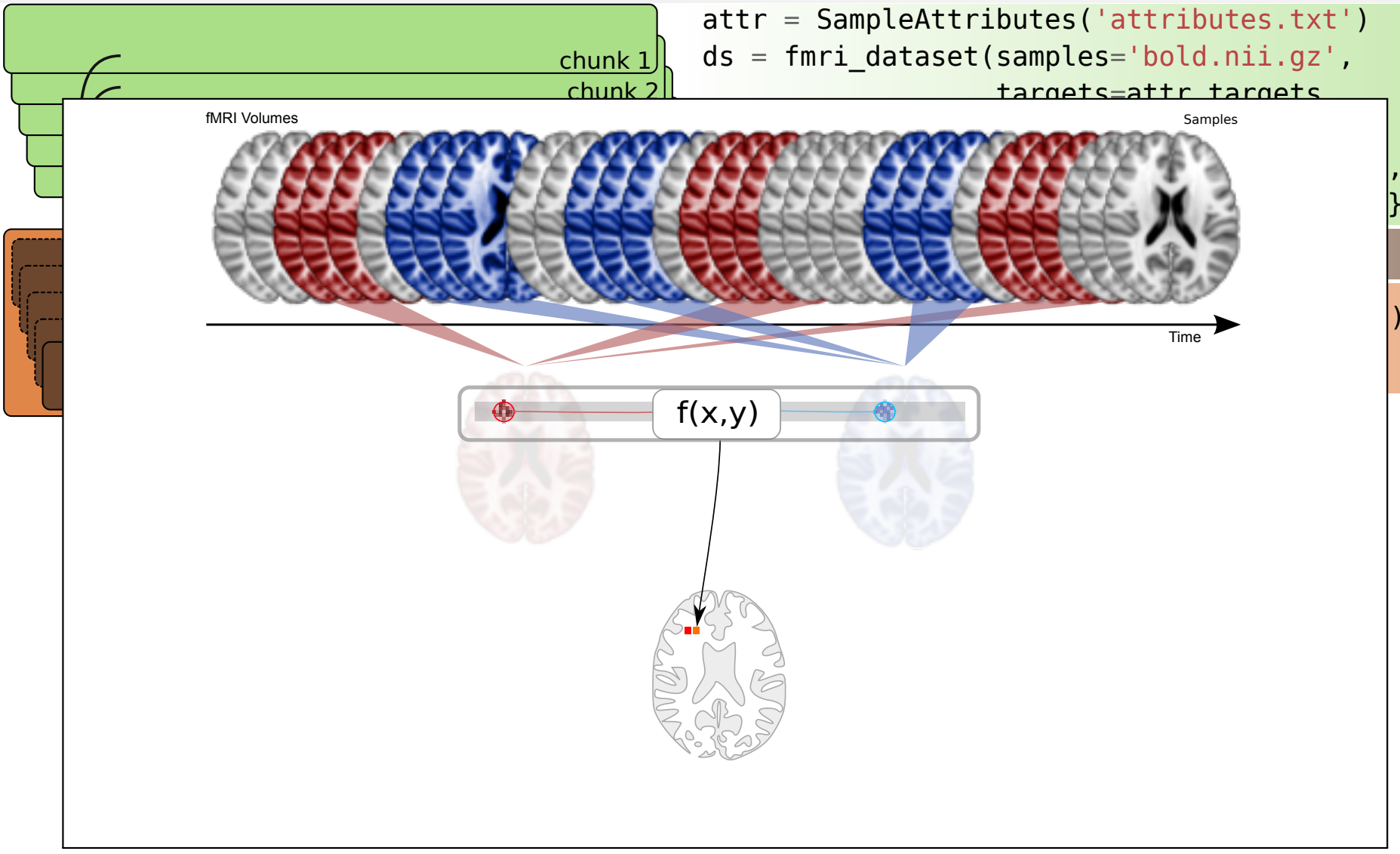
```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets
```



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

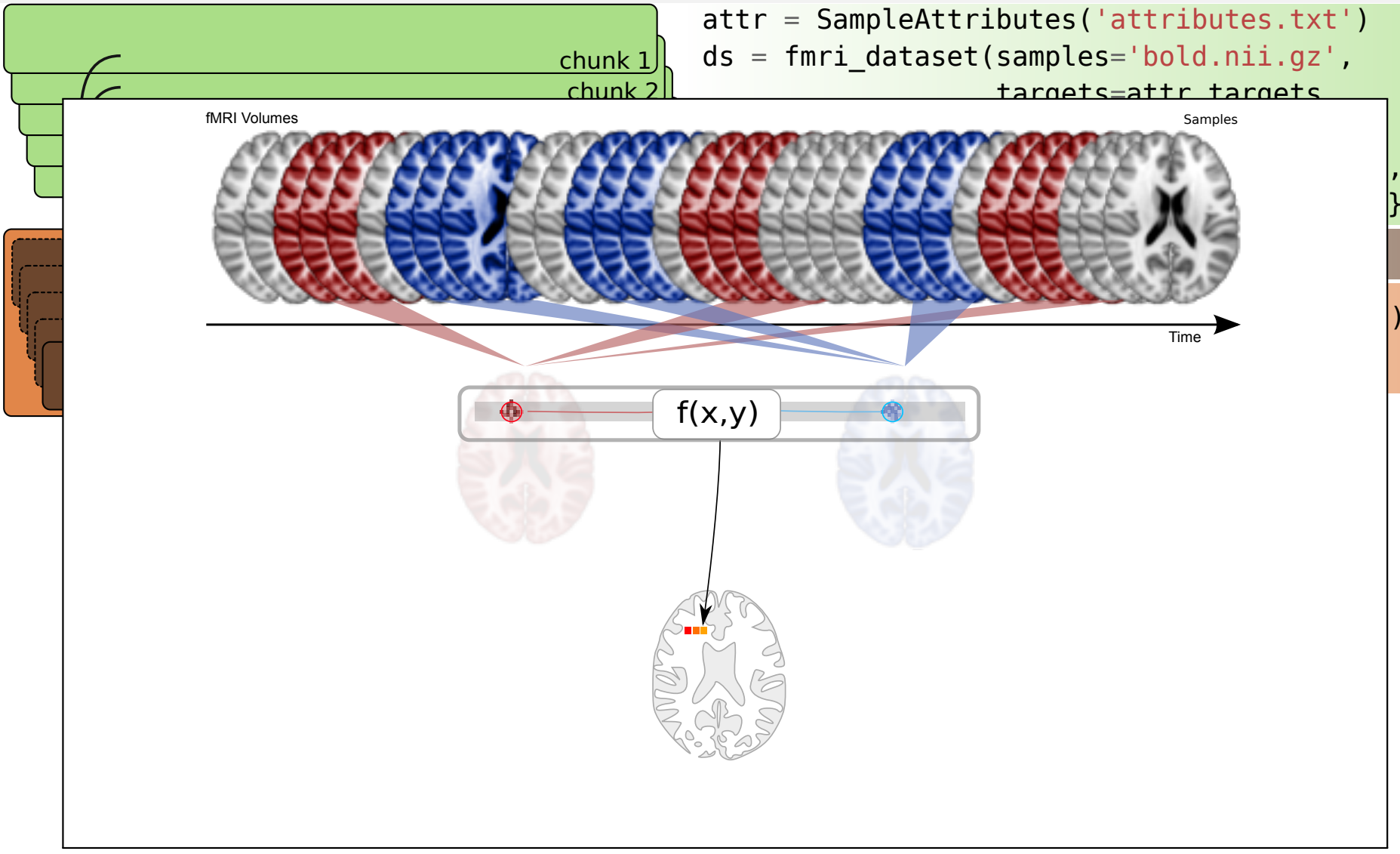
```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets
                  ,
                  })
                  ),
```



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

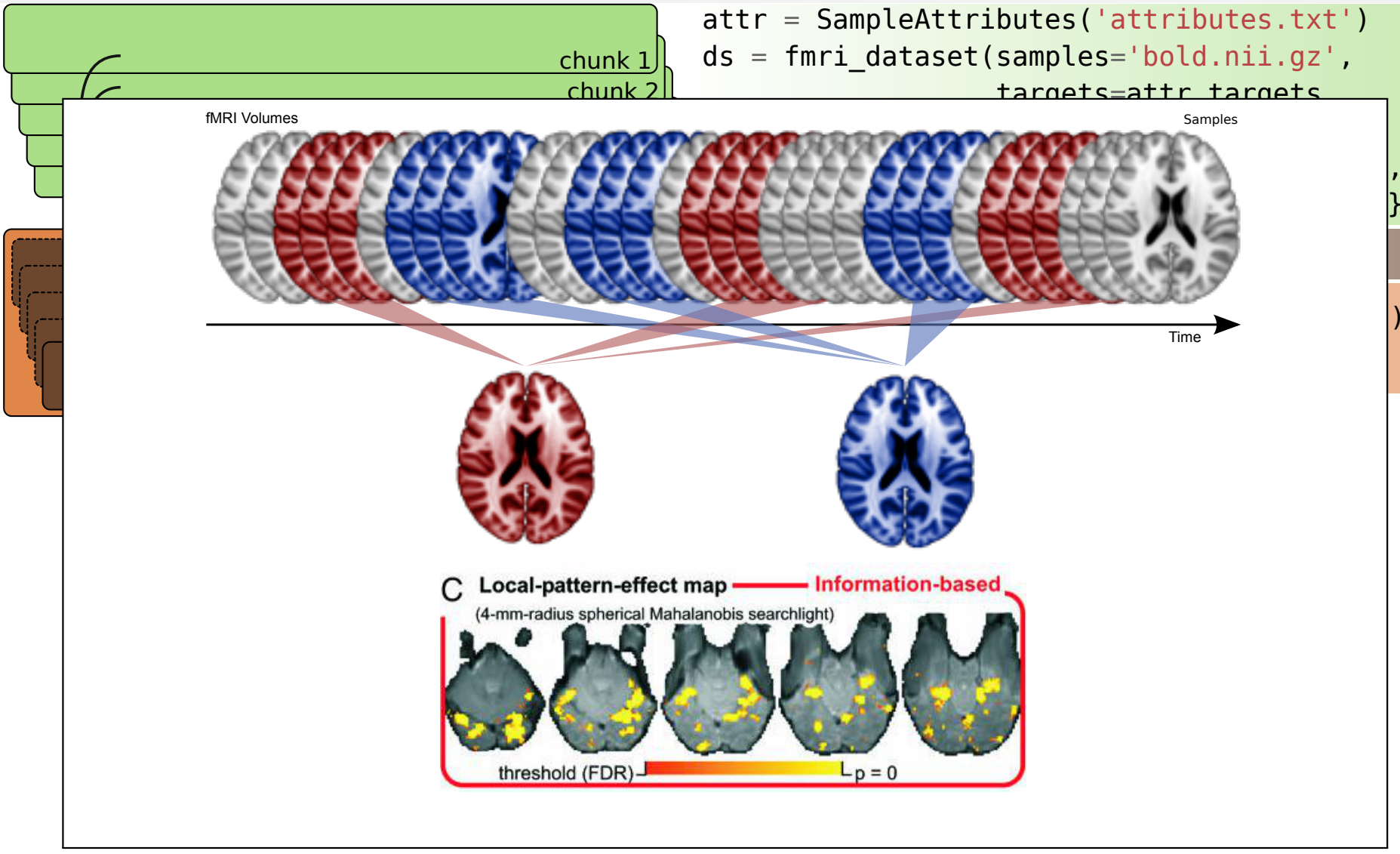
```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets
```



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

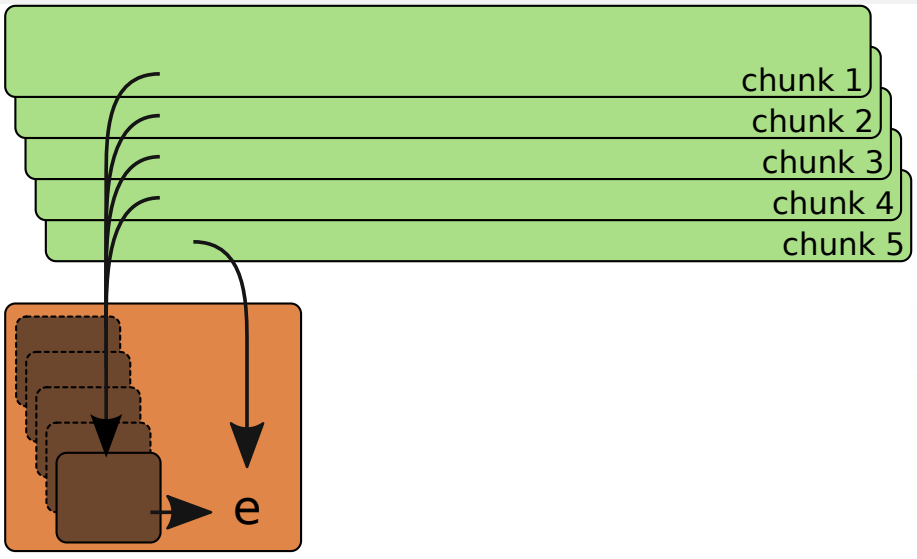
Analysis example: Searchlights

```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets
```



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

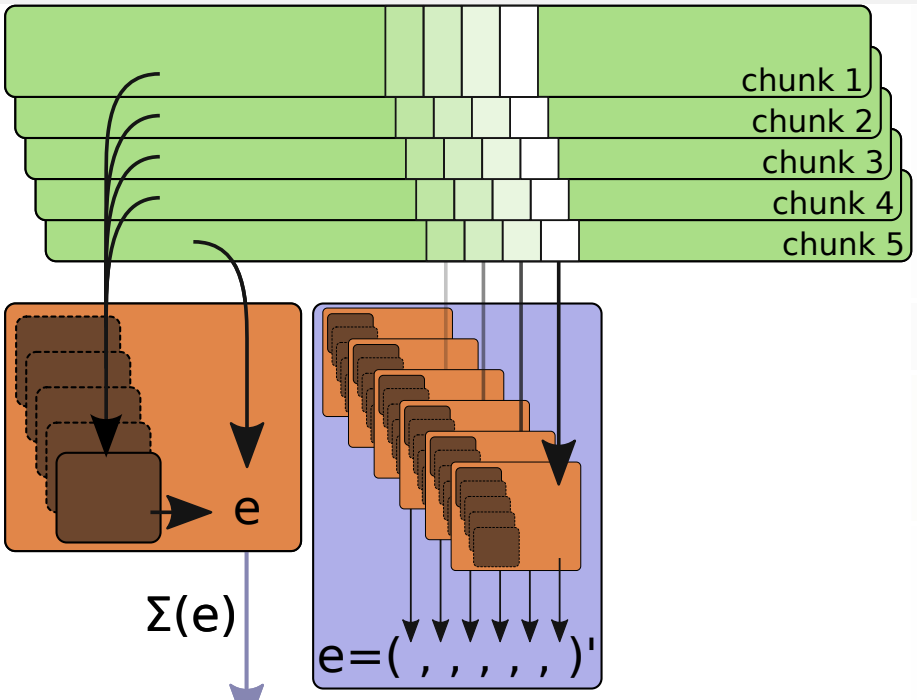


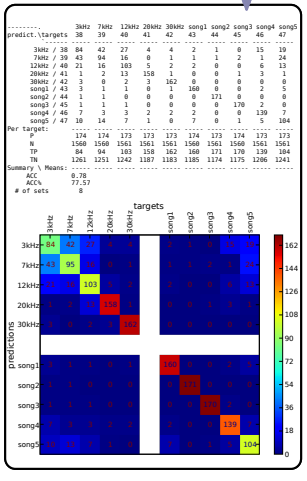

```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets,
                  chunks=attr.chunks,
                  mask='mask_brain.nii.gz',
                  add_fa={'vt':'vt.nii.gz'})
```

```
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'})
```

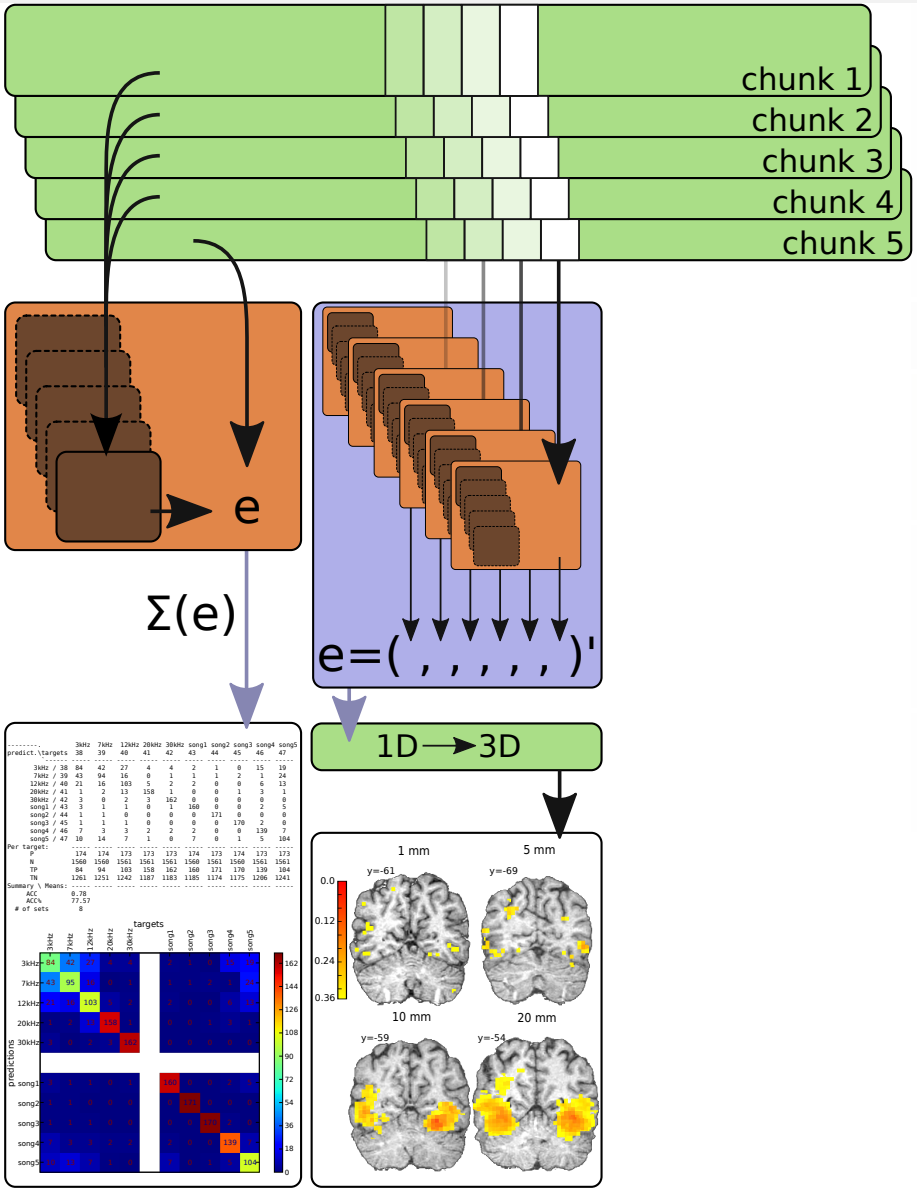
```
clf = LinearCSVMC()
```

```
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'})
```

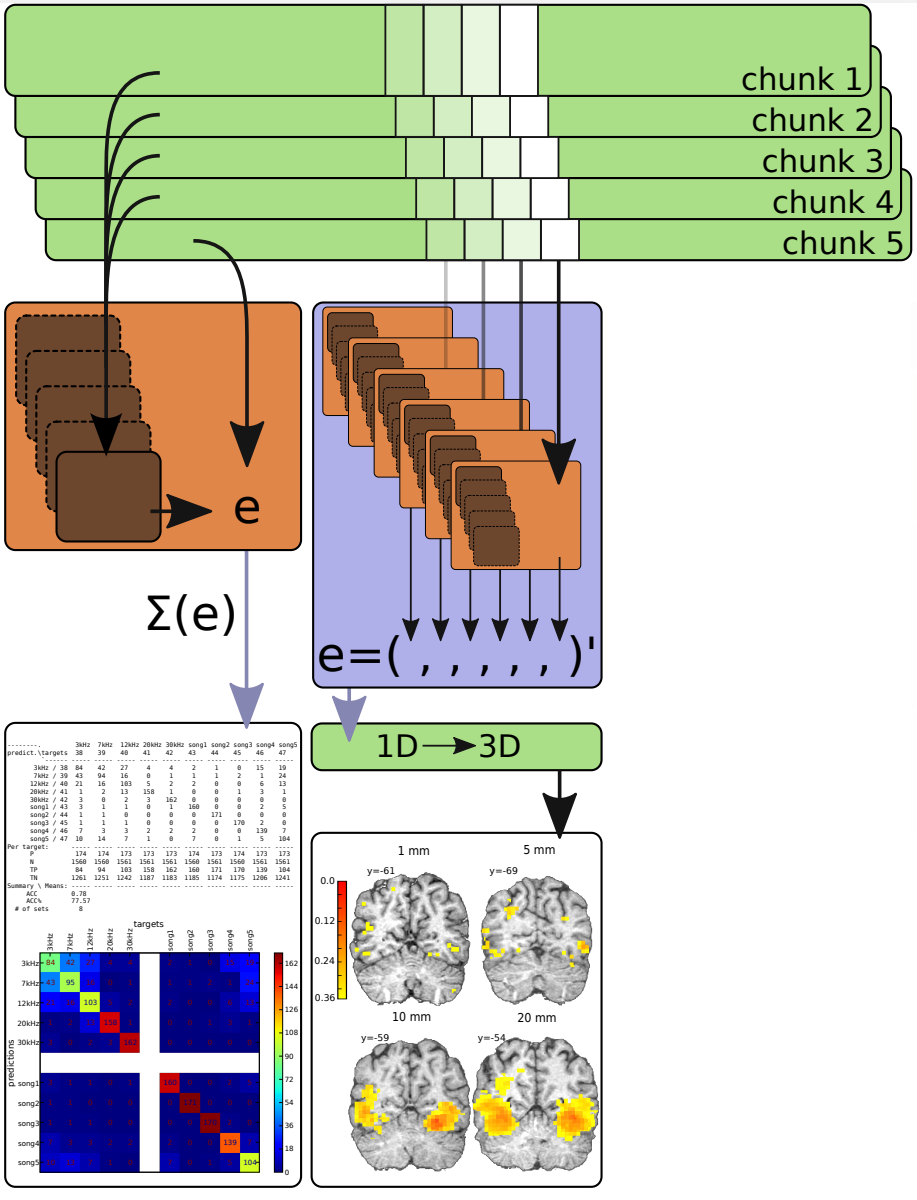
```
clf = LinearCSVMC()
```

```
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()
```

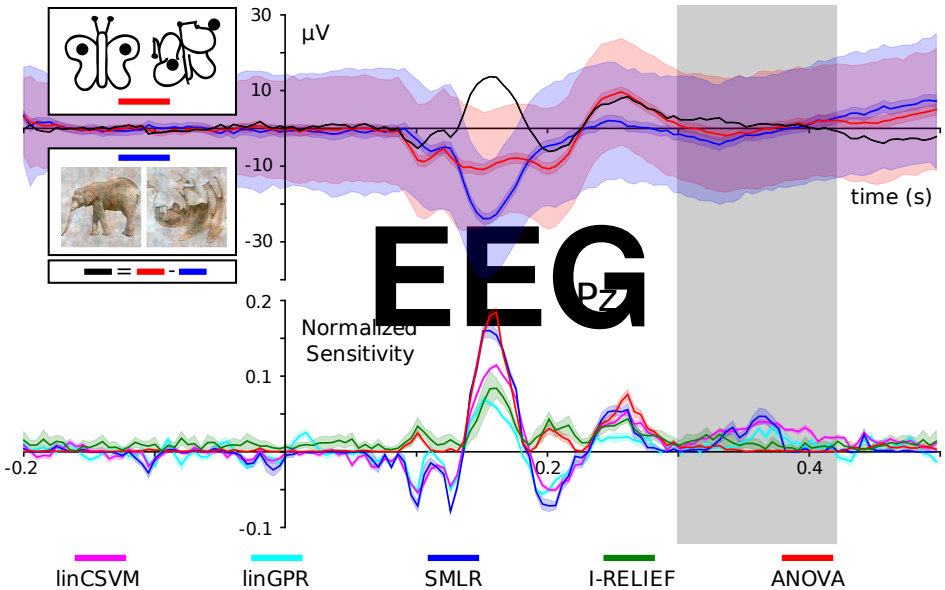
```
cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])
```

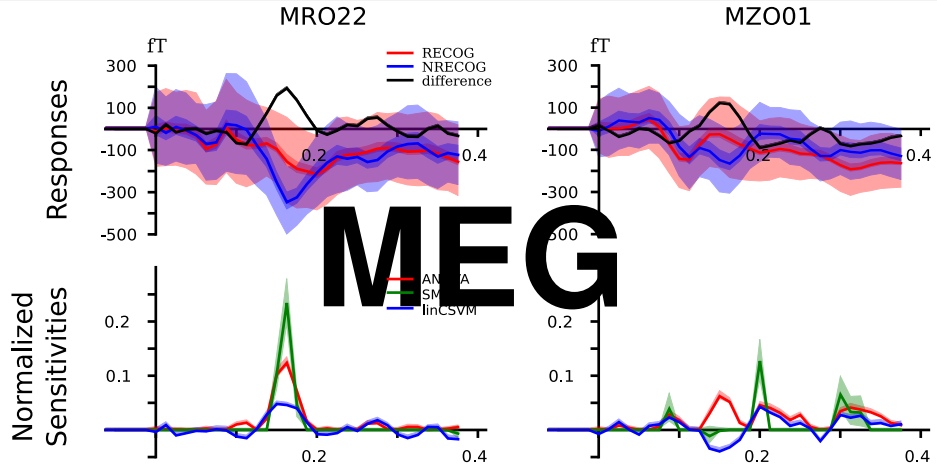
```
fwm = sphere_searchlight(cv, radius=3)


fwm_map = fwm(ds)
```

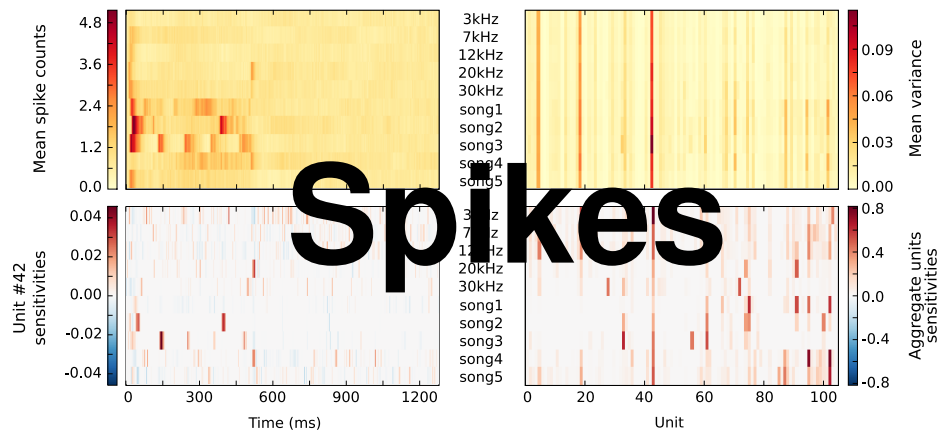
```
map2nifti(ds, fwm_map).to_filename(
    'out.nii.gz')
```

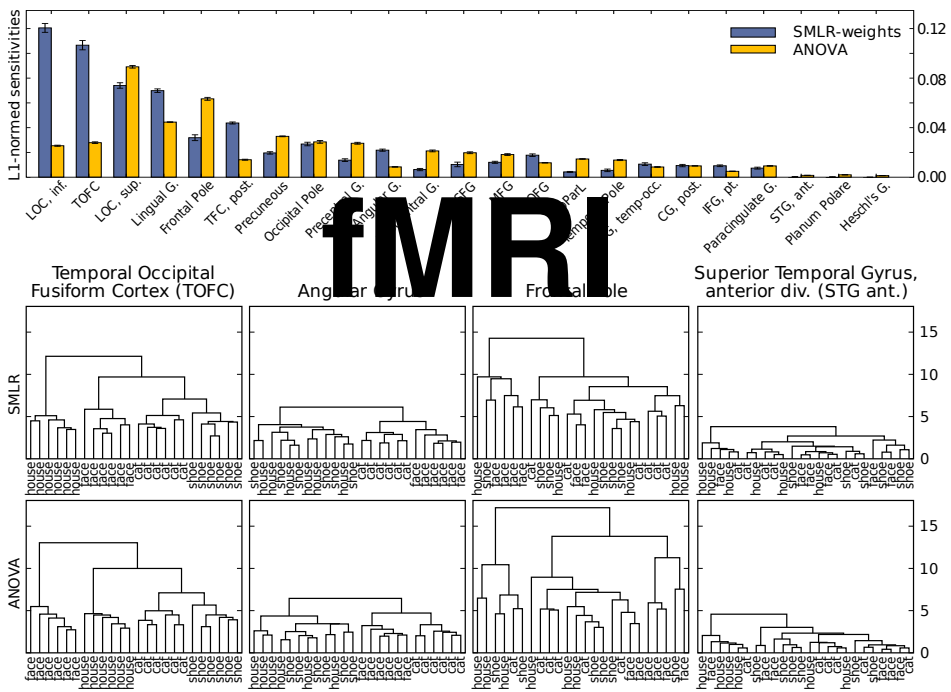
```
h5save('out.hdf5', fwm_map)
```

And with that we could do:



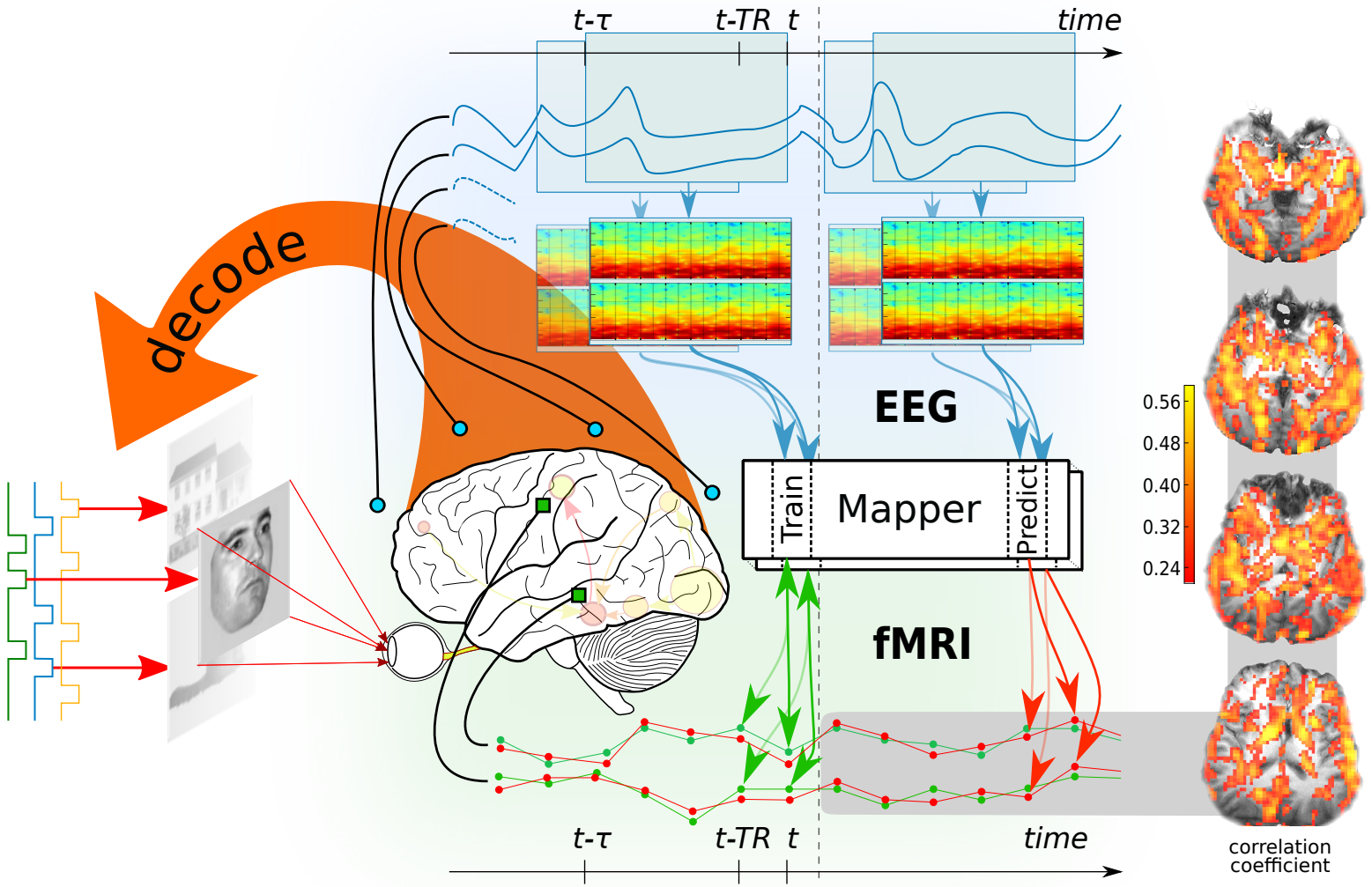






Hanke&**Halchenko** et al., Frontiers in Neuroinformatics, 2009

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 **others** could do:

www.pymvpa.org/whoisusingit.html

Studies employing PyMVPA

2016

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

2015

- Floren et al., Frontiers in Human Neuroscience (2015). Accurately decoding visual information from fMRI data 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 fM
- Danelli et al., Frontiers in Psychology (2015). Framing effects reveal discrete lexical-semantic and sublexica 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 Pathwa
- Greisel et al., arXiv (2015). Photometric redshifts and model spectral energy distributions of galaxies from the S DR10 data
- McNamee et al., Journal of Neuroscience (2015). Characterizing the associative content of brain structures invo 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 i cortex

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

pymvpa.org/tutorial.html

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. "

---

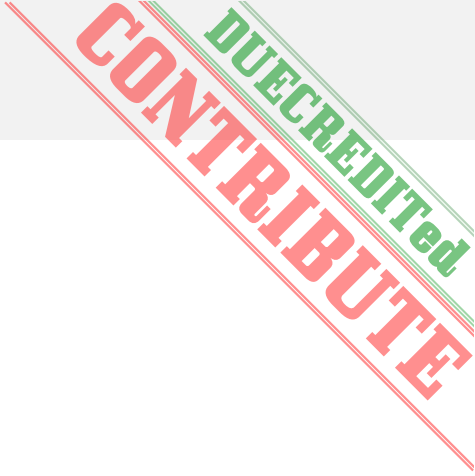
http://kvaden.blogspot.com/2009/03/installing-pymvpa-on-leopard-mac-os.html

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!

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. ..."

http://kvaden.blogspot.com/2009/03/installing-pymvpa-on-leopard-mac-os.html

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! "

---

http://kvaden.blogspot.com/2009/03/installing-pymvpa-on-leopard-mac-os.html

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

We crave *brand new* software, but are afraid of it

We want. . .

- latest research software to get access to bleeding edge technology and stay connected with the field
- latest tools for faster and “more interesting” publications

We **don’t** want. . .

- to “lose results” with a new version for mysterious reasons
- to jeopardize system stability with buggy and unstable research software

We can’t bother to make upgrades an *everyday* habit

- **It simply takes too much time!**
  The average neuroscientist on Windows spends about 14 h/month on non-research maintenance tasks
- Upgrading requires finding webpages, getting accounts, reading documentation, downloading huge archives, running various installers, scripts [lather, rinse, repeat]

Hanke, M. and Halchenko, Y. O. (2011). Neuroscience runs on GNU/Linux. *Front. Neuroinform.*, 5:8. PMC3133852

We can’t bother to make upgrades an *everyday* habit

- **It simply takes too much time!**
  The average neuroscientist on Windows spends about 14 h/month on non-research maintenance tasks
- Upgrading requires finding webpages, getting accounts, reading documentation, downloading huge archives, running various installers, scripts [lather, rinse, repeat]

But . . .

quick dissemination of new features and bug fixes is essential for efficiency

---

Hanke, M. and Halchenko, Y. O. (2011). Neuroscience runs on GNU/Linux. *Front. Neuroinform.*, 5:8. PMC3133852

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: NeuroDebian

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

http://distrowatch.com

Deployment resolution: NeuroDebian

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 NeuroDebian, add fancy logo

More detail on why and how

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:

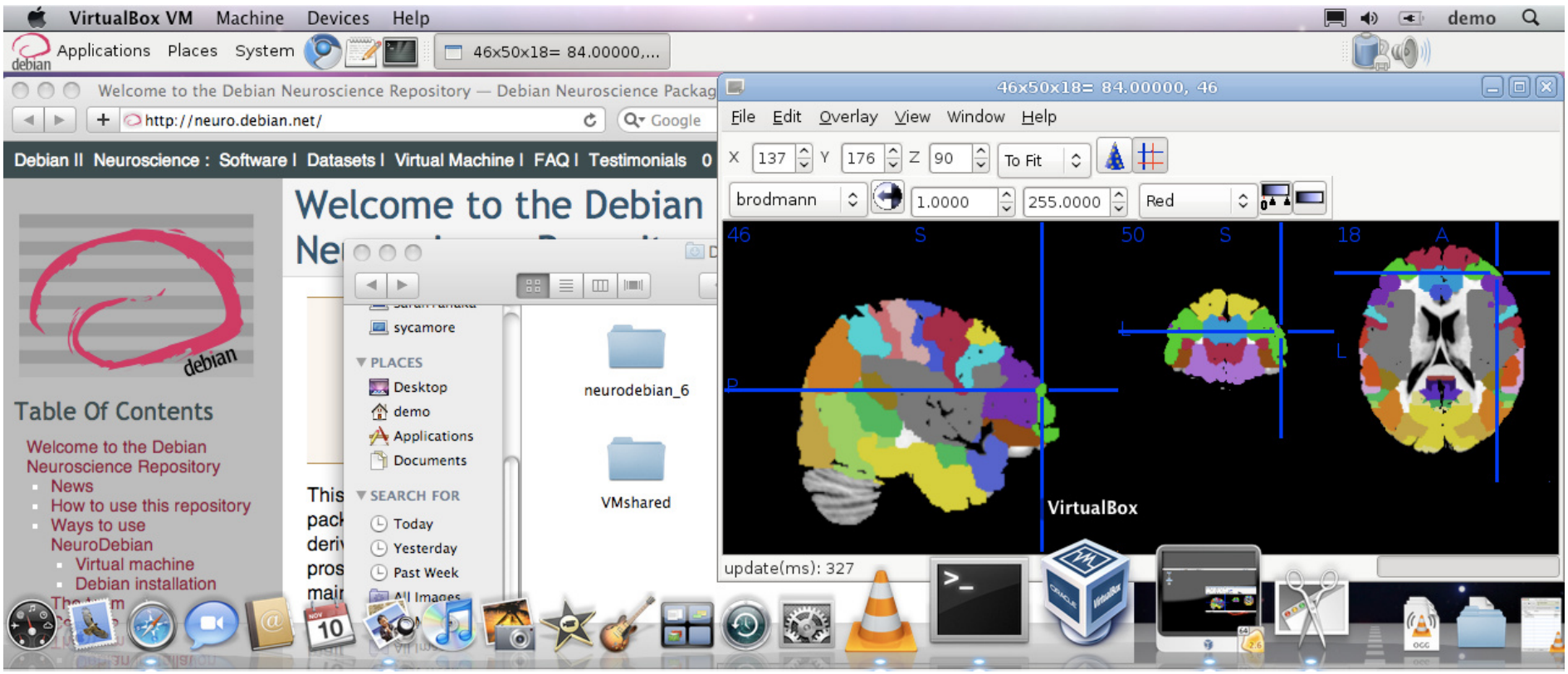
```
debootstrap --include=emacs20,python-base \
   potato /tmp/potato http://archive.debian.org/debian
```

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: 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):



http://neuro.debian.net/vm.html

After X years and the contributions of many people:

Great feedback

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 #2 of the NIH grant submission

Not so great feedback

While this is a laudable goal, and several [40] letters of support attest to its value, *it is not as valuable as developing fundamental advances in neuroimaging software*, and does not remove the need for *all* IT support.

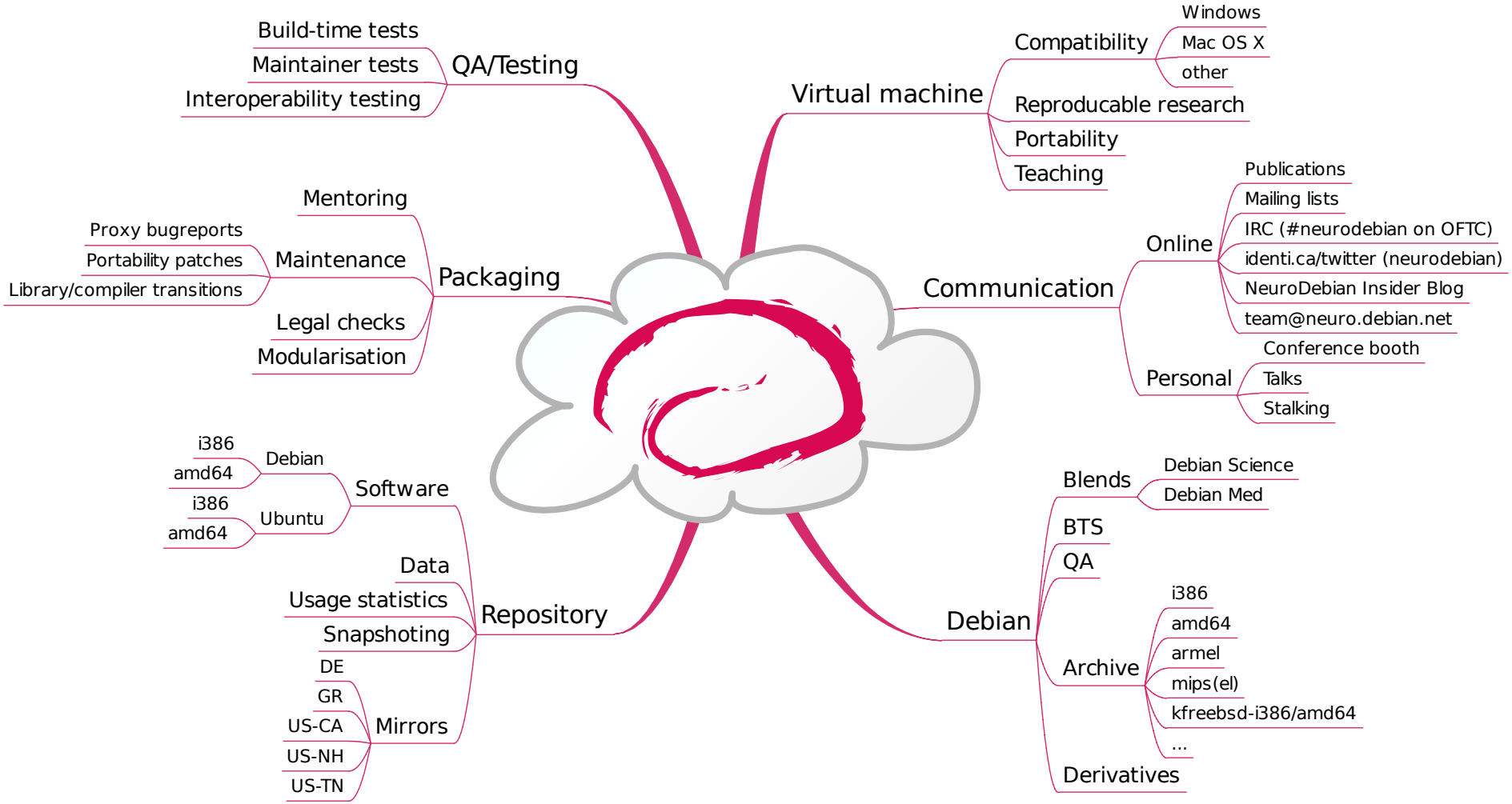
–Anonymous reviewer #1 of the NIH grant submission

End result

I have been increasingly using NeuroDebian in a Virtual Machine because Linux operating systems are not supported by the university. . . . Moreover, powered by NeuroDebian's openness, I see a reason to publish the full code of our experimental and analysis scripts. [NeuroDebian] makes the goal of open science finally viable

–PsychoPy and PyMVPA user Jonas Kubilius, Belgium

What are the *inside outs* of NeuroDebian?



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
- have **your own software** easily available for other’s to use
- **develop software** without worrying about dependencies

Get involved!

- 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

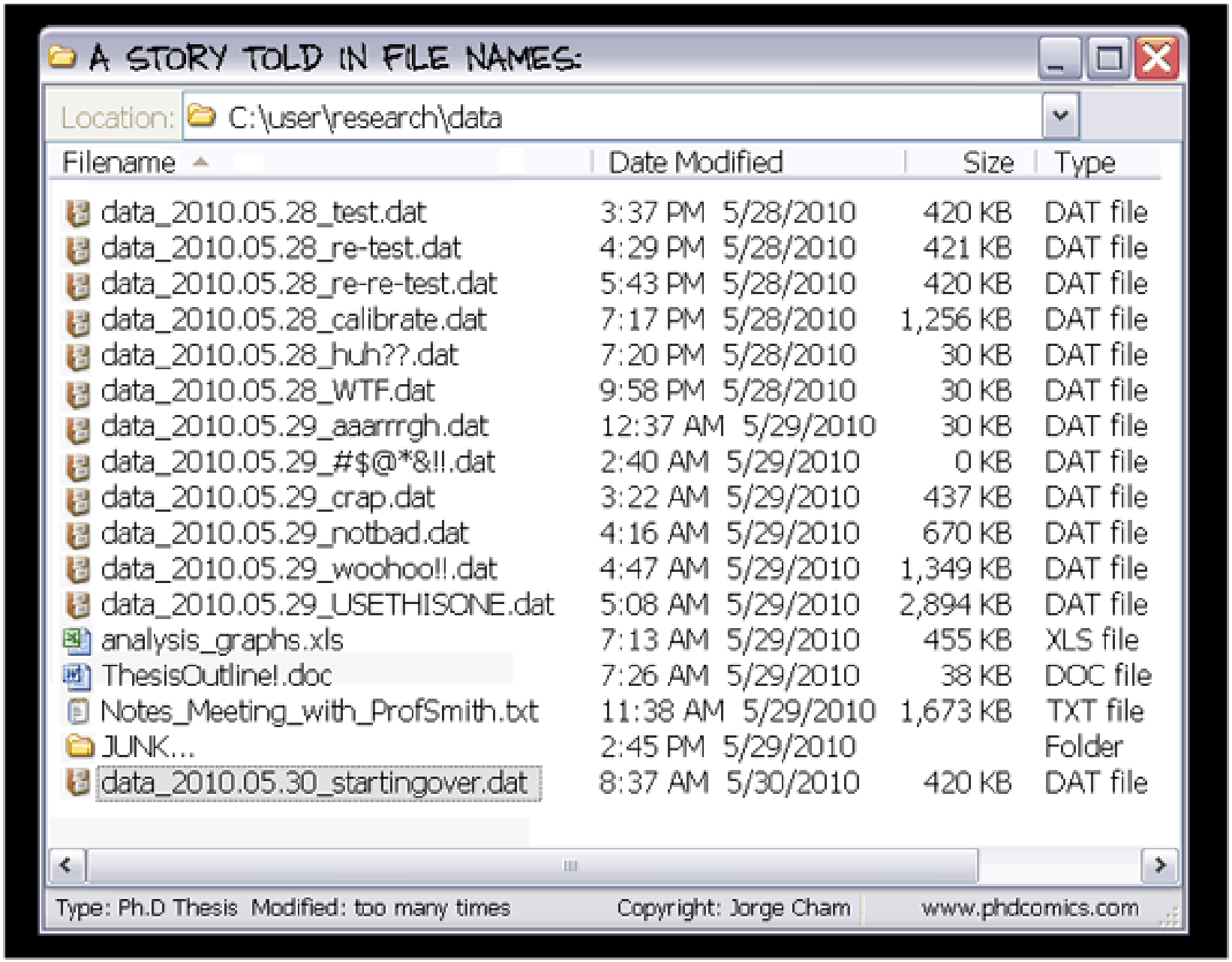
http://datalad.org

Why?

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

derived and/or curated data does change!

Why?


A STORY TOLD IN FILE NAMES:
Location: C:\user\research\data
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_aaarrrgh.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
Type: Ph.D Thesis Modified: too many times
Copyright: Jorge Cham
www.phdcomics.com

ge!

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)*

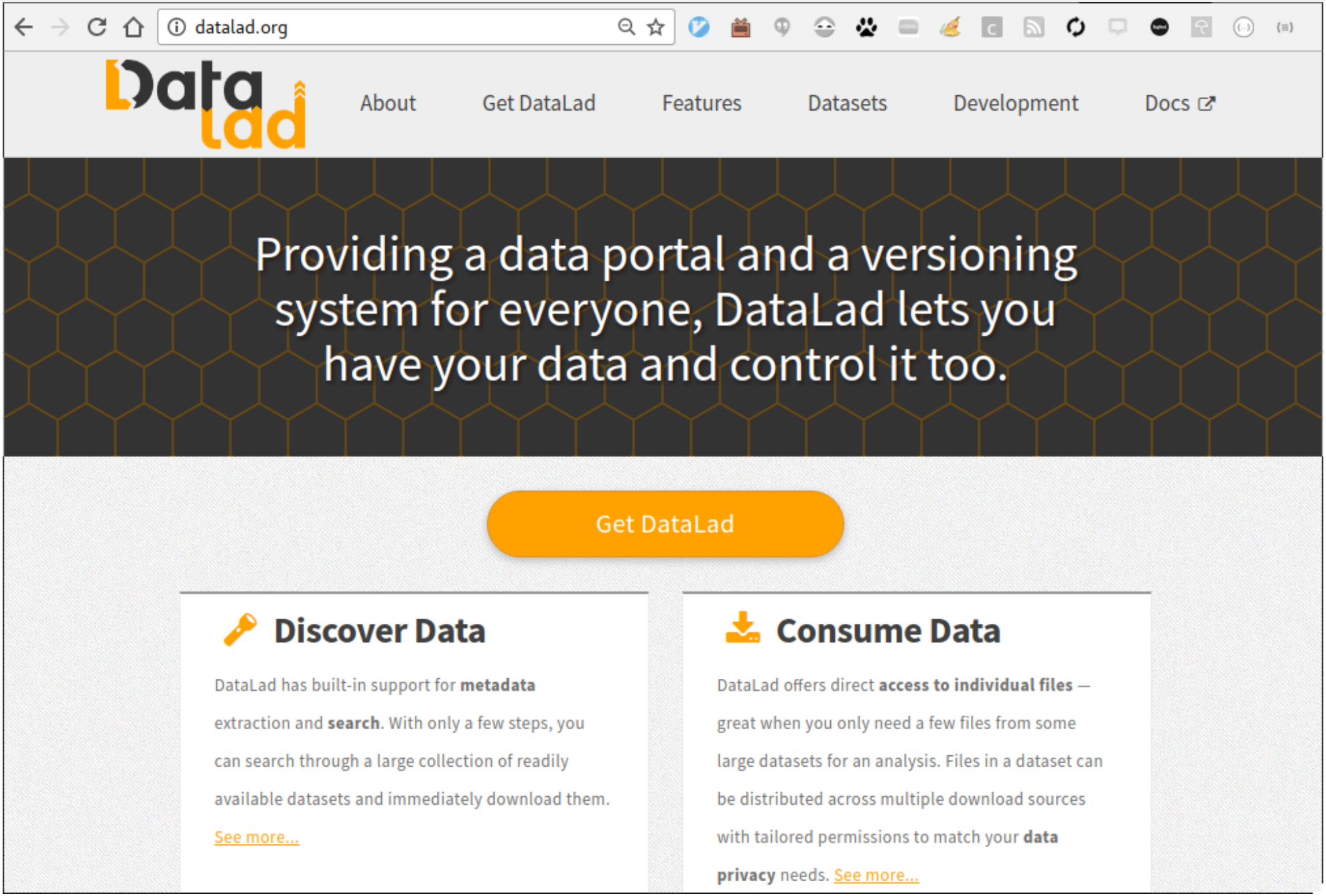
- **difficulty to share** new or derivative data

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



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, bittorent, 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, bittorent, 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*

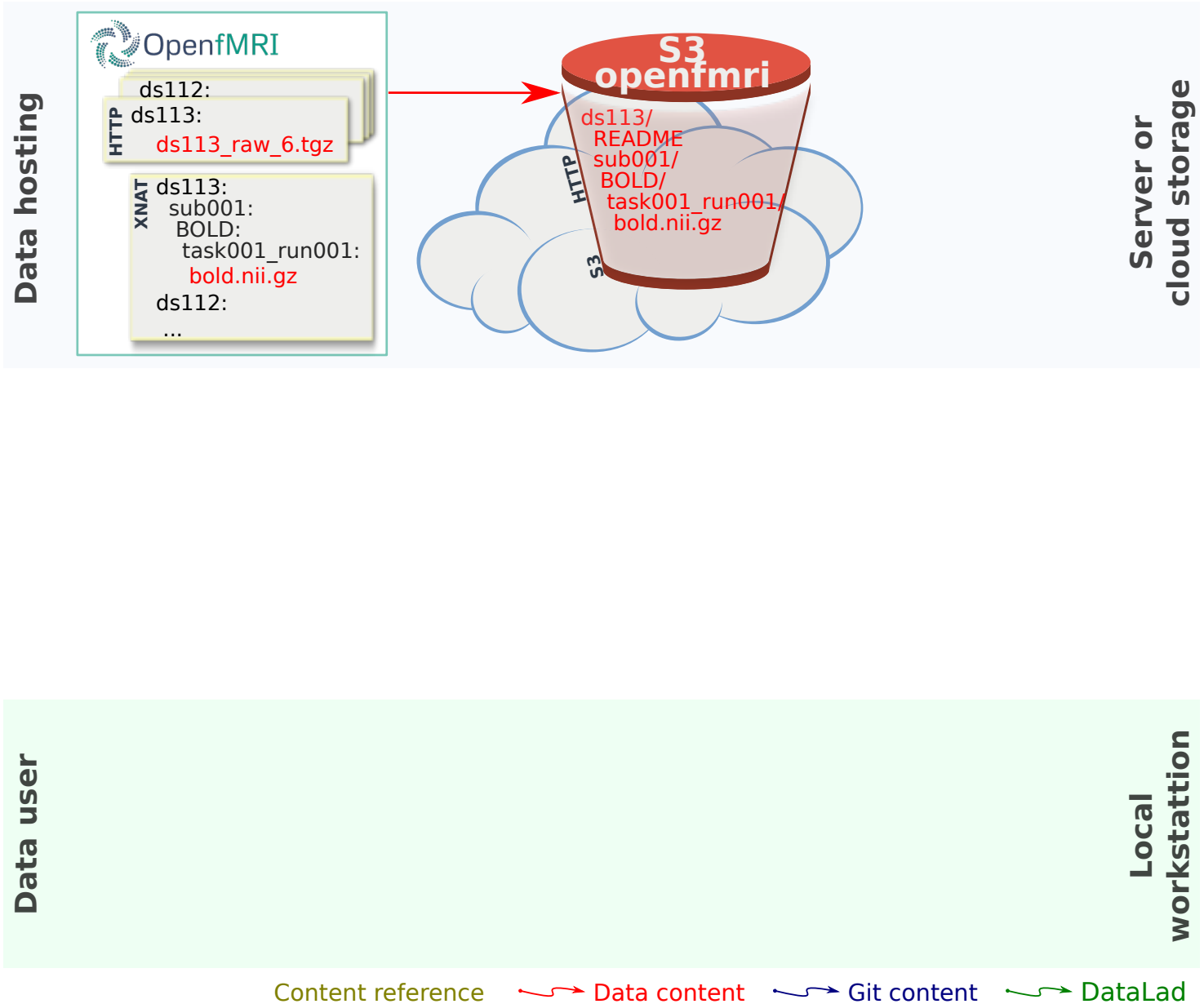
Both Git and git-annex largely work on a single repository level

How: Foundation #2 – Git-annex

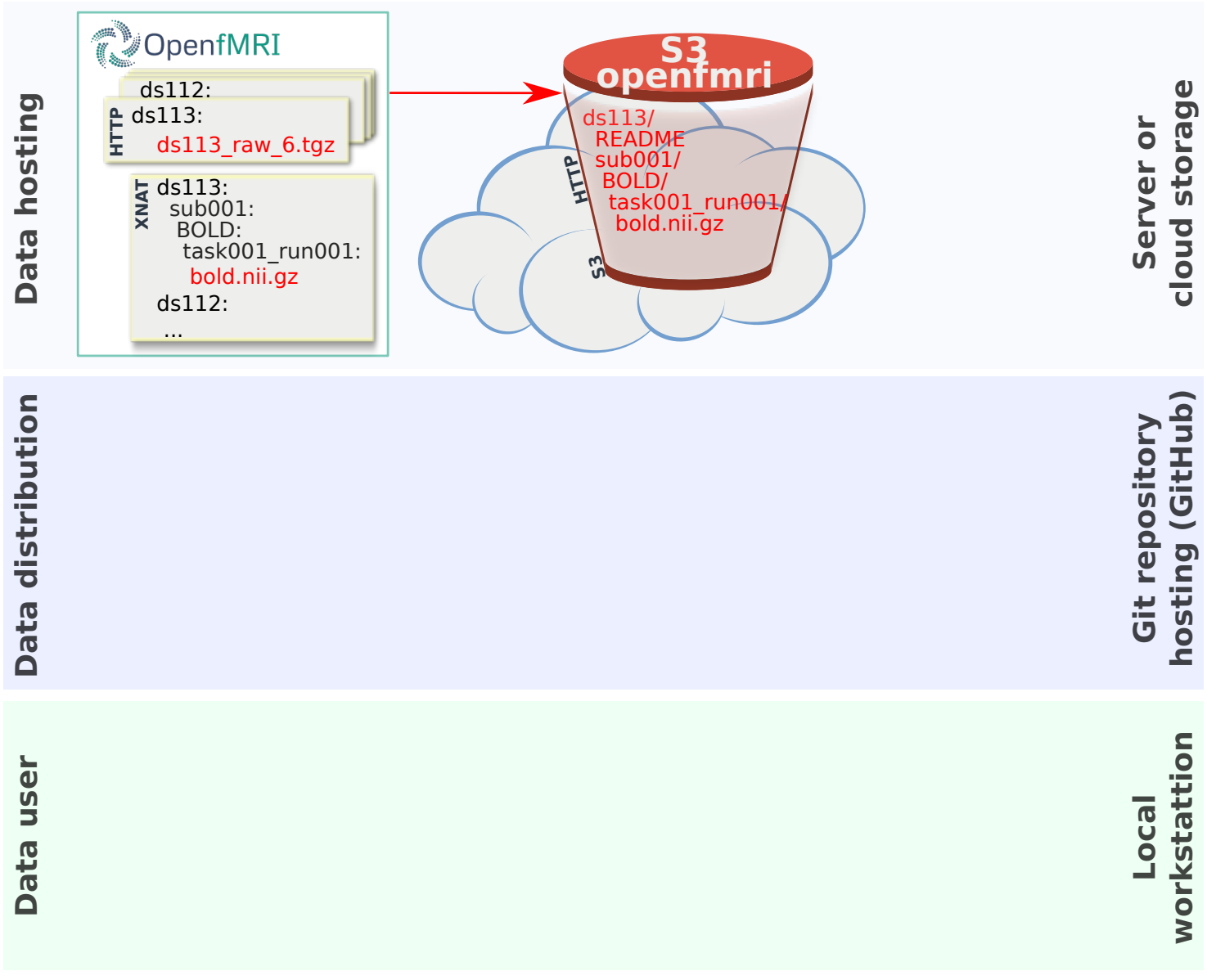
- is **built on top of Git**
- provides **access to data content** from variety of sources: HTTP, FTP, Webdav, bittorent, RSYNC, Amazon S3, *etc.*
- allows for **custom extensions** to get access to offload data: Dropbox, Google Drive, Box.com (will demo later) *etc.*
- features optional Dropbox-like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

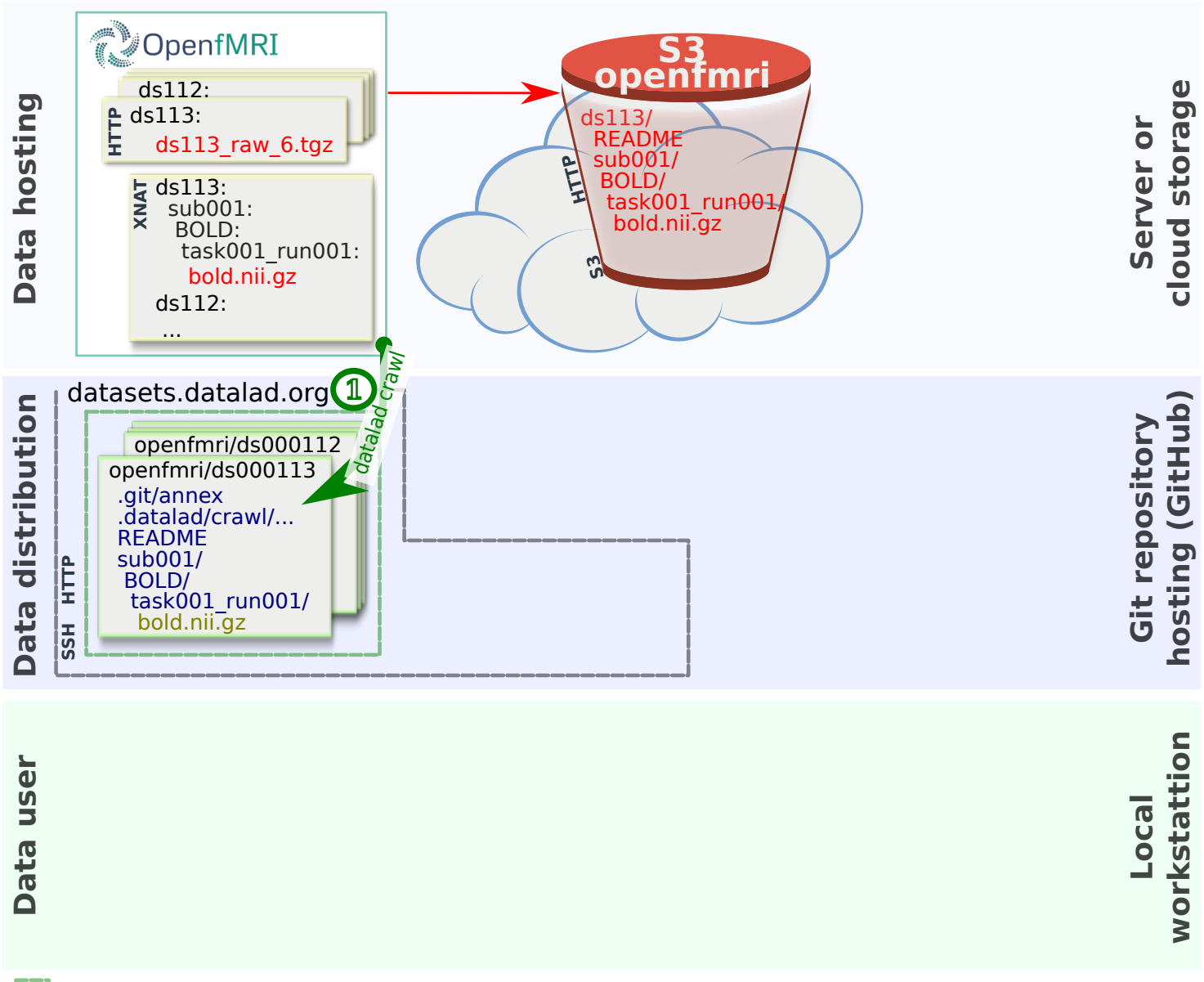
DataLad data distribution: Data life cycle



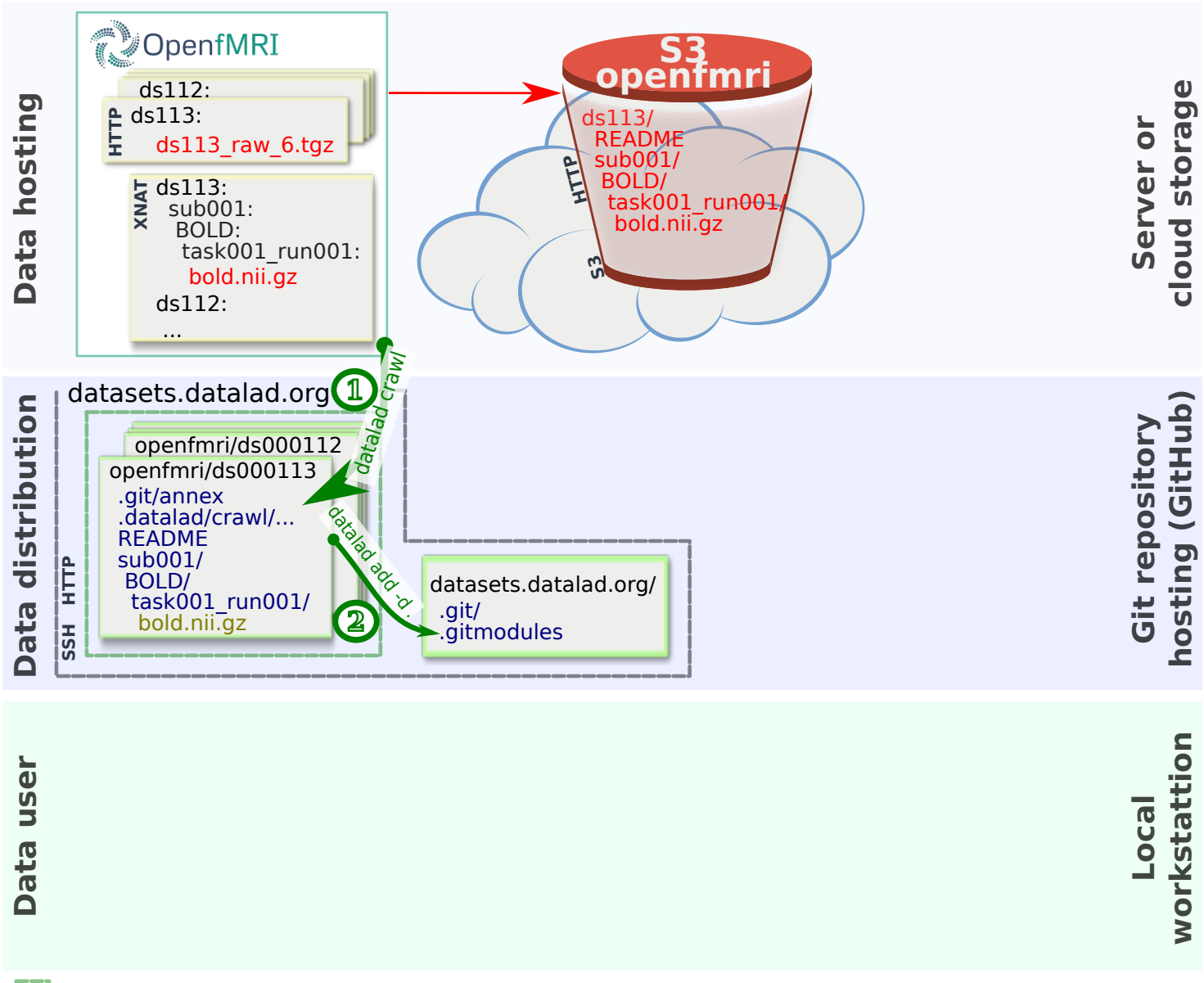
DataLad data distribution: Data life cycle



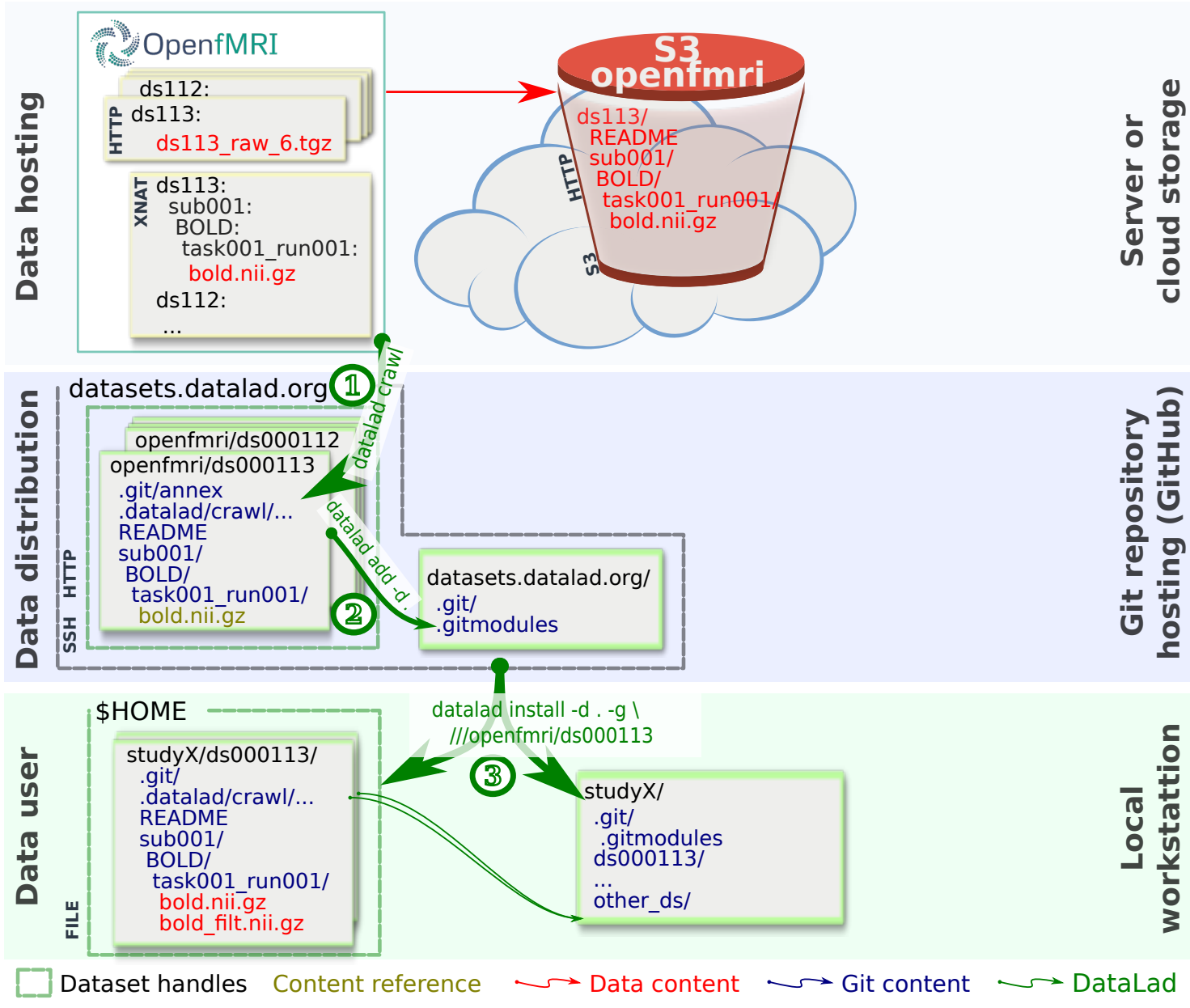
DataLad data distribution: Data life cycle



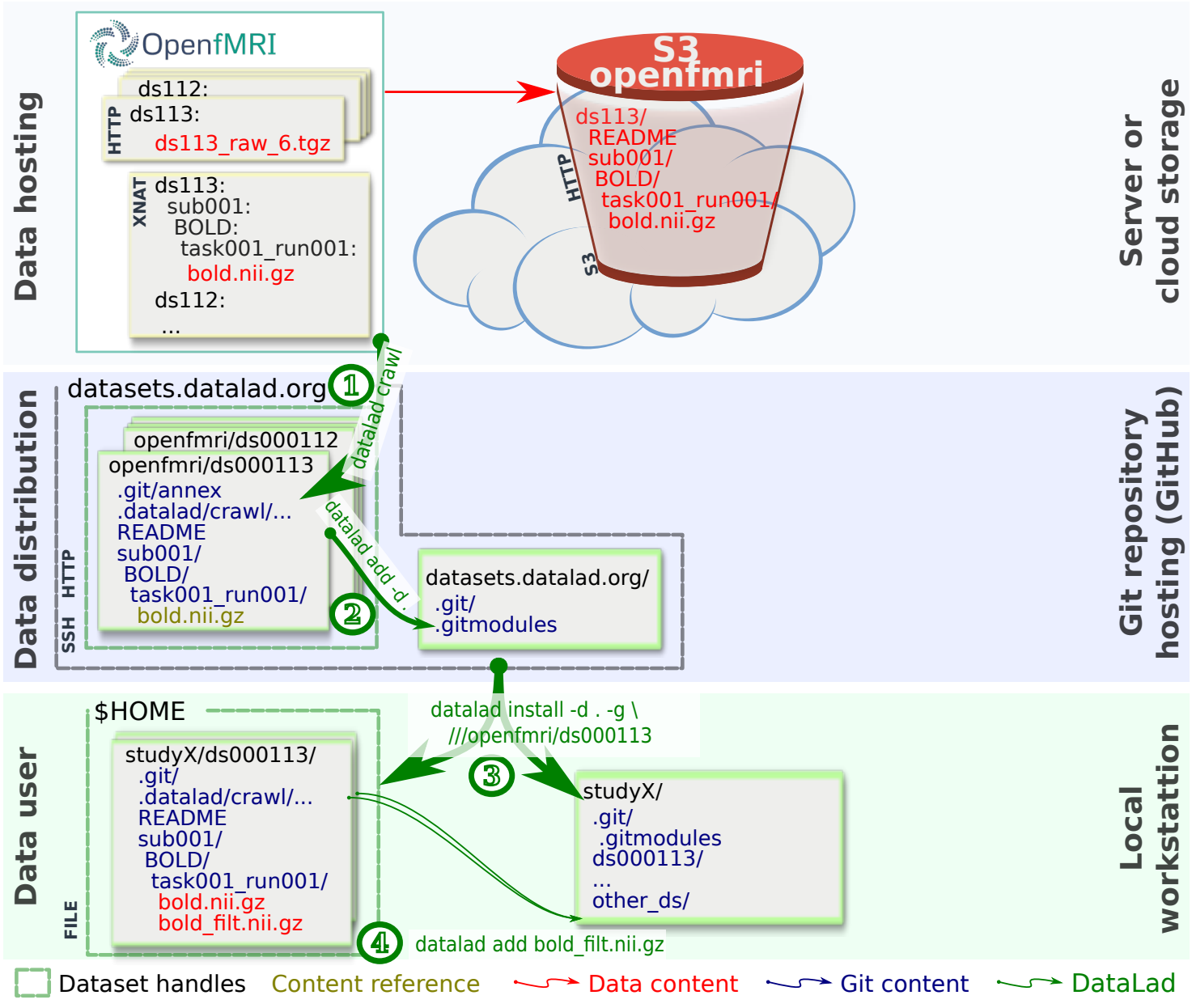
DataLad data distribution: Data life cycle



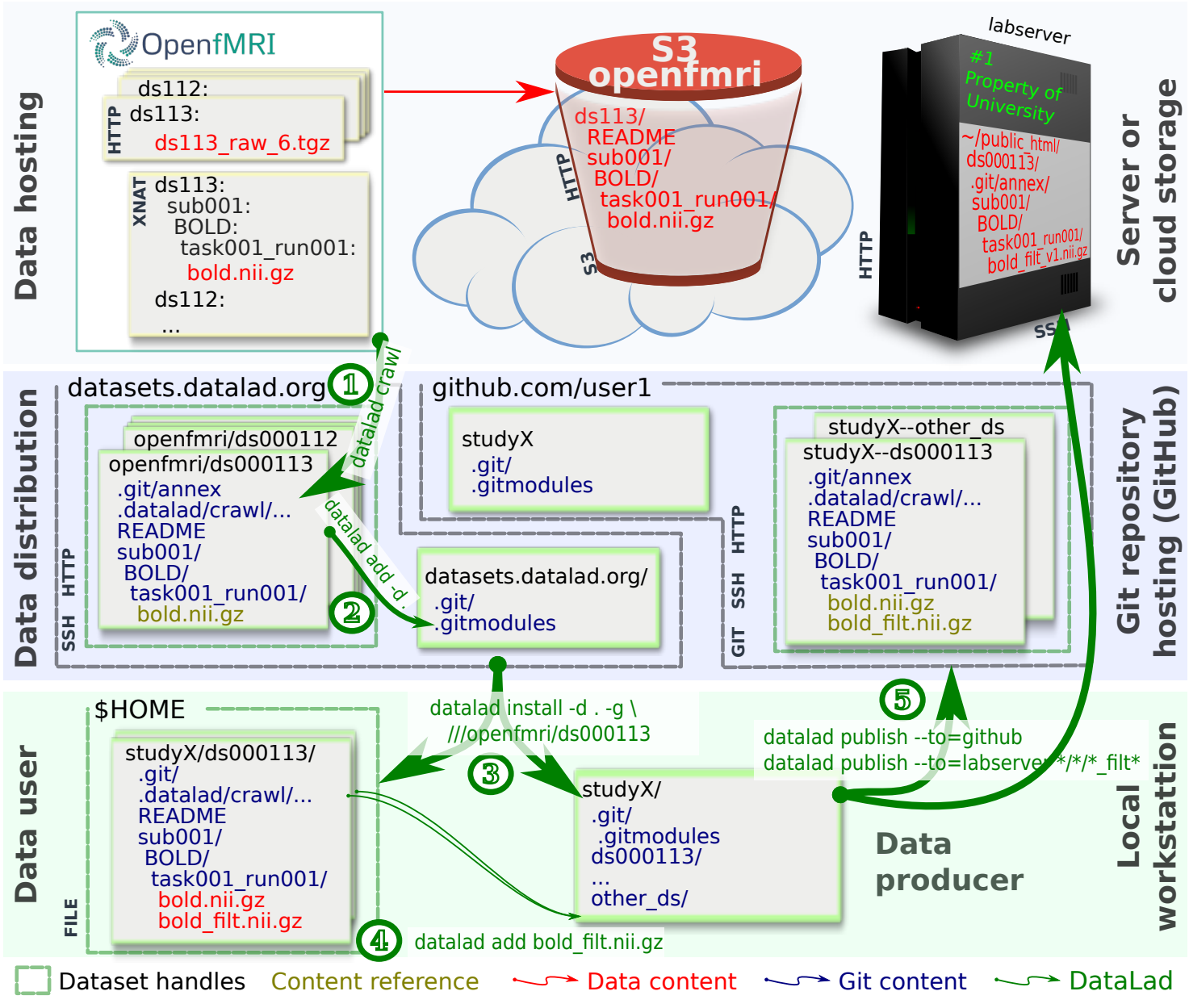
DataLad data distribution: Data life cycle



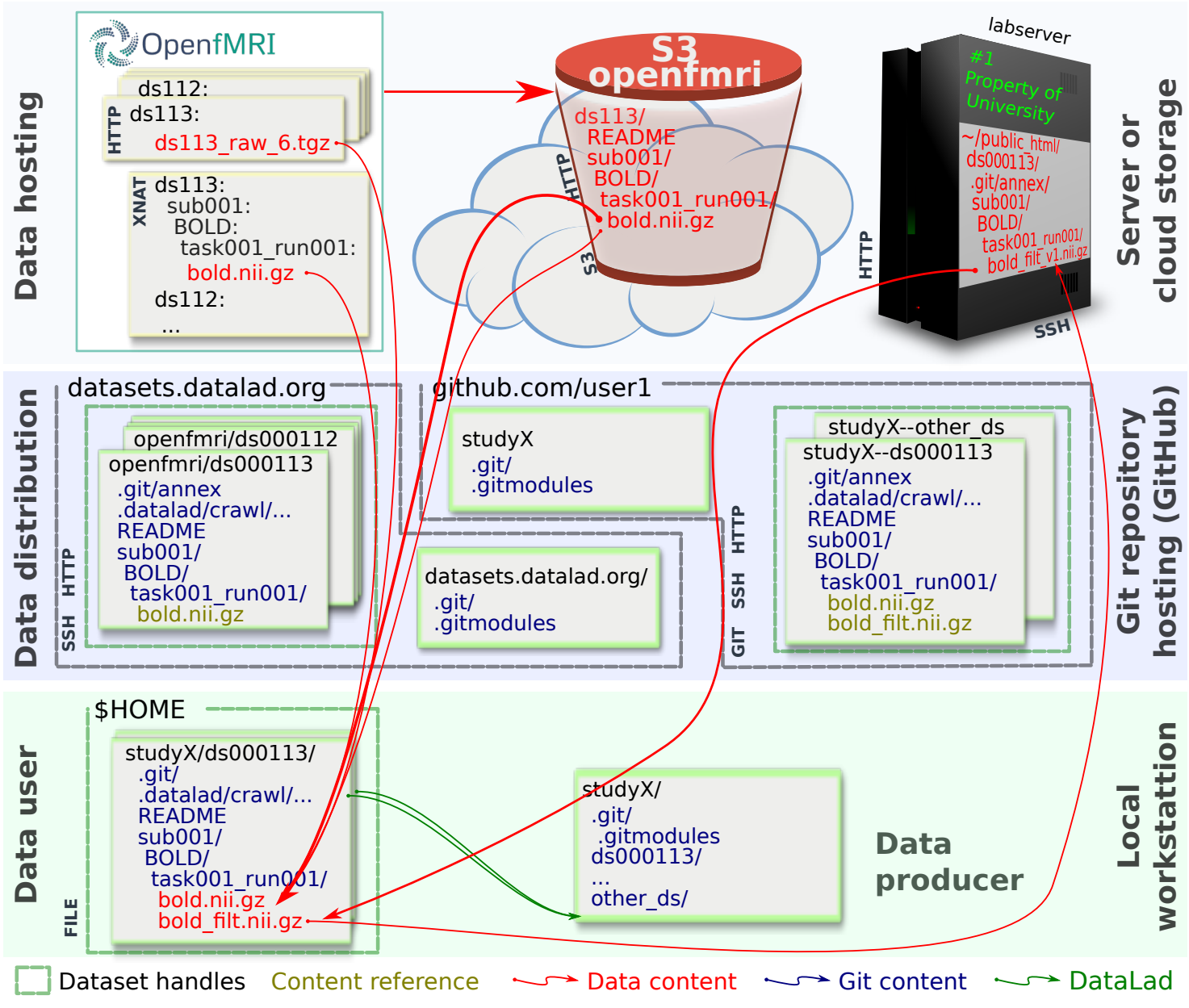
DataLad data distribution: Data life cycle



DataLad data distribution: Data life cycle



DataLad data distribution: Data life cycle



How #1+2=#3: DataLad

- manages **multiple repositories** organized into “super-datasets” using standard git sub-modules mechanism

How #1+2=#3: DataLad

- manages **multiple repositories** organized into “super-datasets” using standard git sub-modules mechanism
- supports both **git and git/annex** repositories

How #1+2=#3: DataLad

- manages **multiple repositories** organized into “super-datasets” using standard git sub-modules mechanism
- supports both **git and git/annex** repositories
- can **crawl** external online data sources, and update git/annex repositories upon changes

How #1+2=#3: DataLad

- manages **multiple repositories** organized into “super-datasets” using standard git sub-modules mechanism
- supports both **git and git/annex** repositories
- can **crawl** external online data sources, and update git/annex repositories upon changes
- is **scalable** since data stays with original data providers

How #1+2=#3: DataLad

- manages **multiple repositories** organized into “super-datasets” using standard git sub-modules mechanism
- supports both **git and git/annex** repositories
- can **crawl** external online data sources, and update git/annex repositories upon changes
- is **scalable** since data stays with original data providers
- **unifies access** to data regardless of its origin (custom portals with authentication, S3, *etc.*) or serialization (*e.g.*, tarballs)

How #1+2=#3: DataLad

- manages **multiple repositories** organized into “super-datasets” using standard git sub-modules mechanism
- supports both **git and git/annex** repositories
- can **crawl** external online data sources, and update git/annex repositories upon changes
- is **scalable** since data stays with original data providers
- **unifies access** to data regardless of its origin (custom portals with authentication, S3, *etc.*) or serialization (*e.g.*, tarballs)
- aggregates datasets’ **meta-data** and allows for quick **search**

How #1+2=#3: DataLad

- manages **multiple repositories** organized into “super-datasets” using standard git sub-modules mechanism
- supports both **git and git/annex** repositories
- can **crawl** external online data sources, and update git/annex repositories upon changes
- is **scalable** since data stays with original data providers
- **unifies access** to data regardless of its origin (custom portals with authentication, S3, *etc.*) or serialization (*e.g.*, tarballs)
- aggregates datasets’ **meta-data** and allows for quick **search**
- can **publish** original or derived datasets publicly (a web server, github) or for internal use (e.g., via ssh), while possibly keeping data available from elsewhere

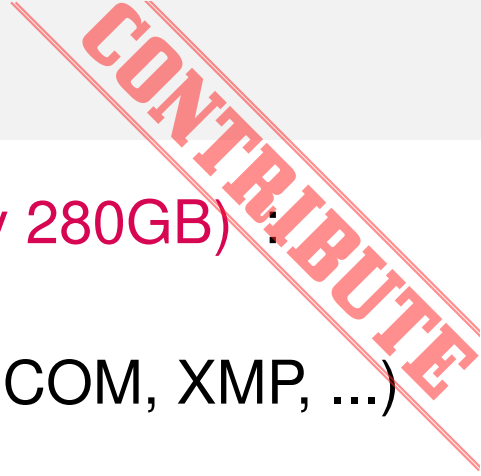
How #1+2=#3: DataLad

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

How #1+2=#3: DataLad

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

DataLad



Growing data "distribution" (>12TB, hosting locally only 280GB) :

- http://datasets.datalad.org
- Extended meta-data support (BIDS, DICOM, XMP, ...)

Covered :

- Neuroimaging: http://openfmri.org, http://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)
- ... *(have interesting data provider? File github issue: github.com/datalad/datasets.datalad.org)* ...
- Integration: NeuroDebian; OSF, Zenodo

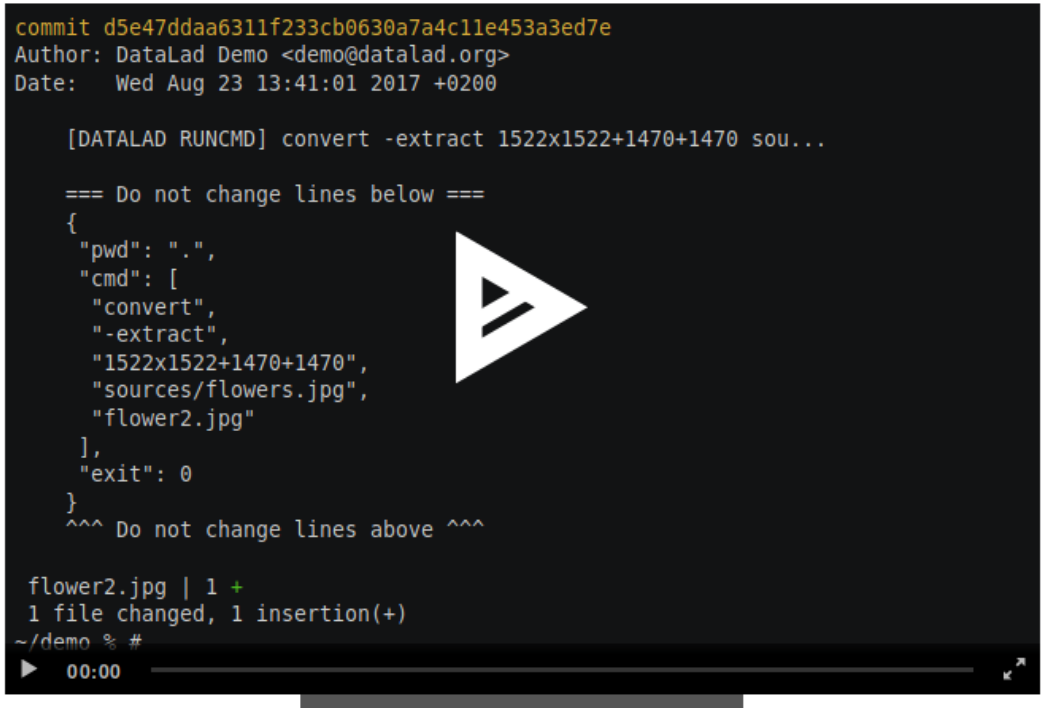
Integrations : ReproIn, WiP: OpenNeuro

Later just go to datalad.org/features.html

https://www.datalad.org/for/reproducible-science 110%

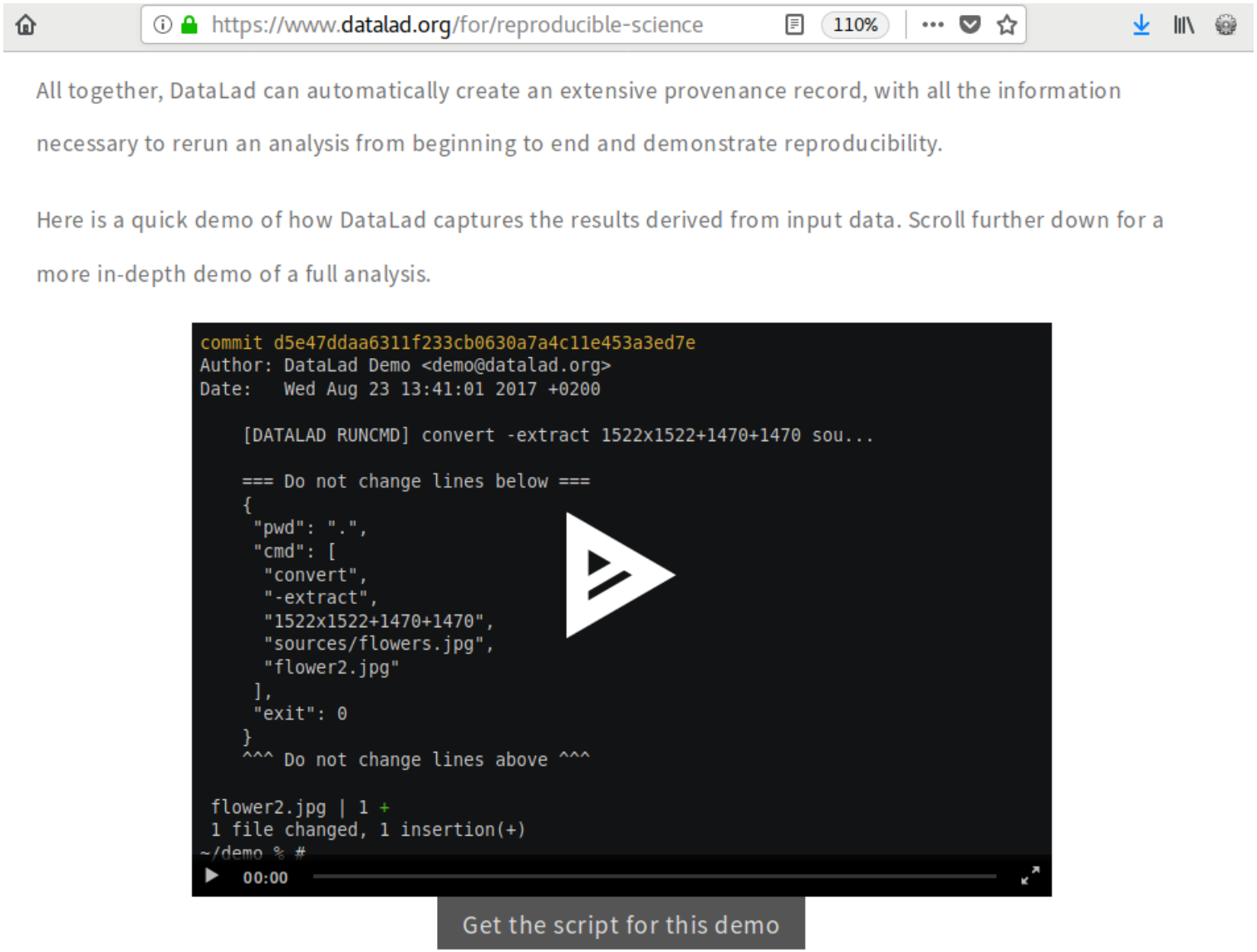
All together, DataLad can automatically create an extensive provenance record, with all the information necessary to rerun an analysis from beginning to end and demonstrate reproducibility.

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.



Get the script for this demo

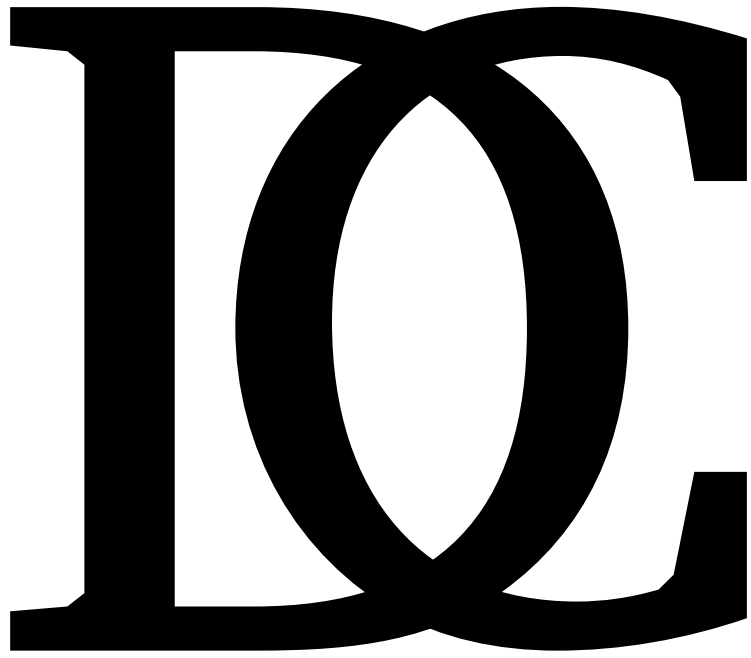
Later just go to datalad.org/features.html



Managing data can be as simple to managing code and software

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 demostrate 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!

- 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 publicly
- Test (unit-, regression-) your analysis and assumptions
- (Re)use public datasets
- Collect information about your computation environment and analysis
- Create your own shareable (legally!) virtualized/conteinerized 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 publicly
- Test (unit-, regression-) your analysis and assumptions
- (Re)use public datasets
- Collect information about your computational environment and analysis
- Create your own shareable (virtualized/containerized) computational environments from unambiguous specification
- Automate your analysis as much as possible

If originally might have sounded Utopian?
You saw now that Neuroimaging community is blessed with great initiatives and tools to make it actually 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 publicly
  - BIDS, ReproIn, DataLad (git, git-annex), GitHub, ...
- Test (unit-, regression-) your analysis and assumptions
  - PyTest, MOxUnit, CTest, Travis-CI, CircleCI, ...
- (Re)use public datasets
  - 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/conteinerized 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.
ICANHASCHEEZBURGER.COM

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. [Retrived 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.

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.