Truly, there are lies, brazen lies, and statistics, but let’s not, my friends, forget the psychology!

Environments for efficient contemporary research in neuroimaging

Yaroslav O. Halchenko & Michael Hanke

Dartmouth College, USA
Otto-von-Guericke-University Magdeburg, Germany

UPENN 2012, Philadelphia PA, USA
Acknowledgements

Swaroop Guntupalli
Andrew Connolly
Per B. Sederberg
Emanuele Olivetti
Valentin Haenel
James M. Hughes
Scott Gorlins

FOSS developers of
Python, NumPy, SciPy,
Matplotlib, H5Py, Rpy,
Shogun, scikit-learn,
Inkscape, ...

Debian Community

James V. Haxby
Stephen J. Hanson
Stefan Pollmann

Dr. Yaroslav O. Halchenko
Dartmouth College, NH, USA

yoh@onerussian.com

about the slides:
should become available at
© 2007-2012
Yaroslav O. Halchenko & Michael Hanke,
Arno Klein
inspired by Stefano Zacchiroli
CC BY-SA 3.0 — Creative Commons Attribution-ShareAlike 3.0
The task of neural science is to explain behavior in terms of the activities of the brain

*Eric Kandel, Principles of Neural science, 4th ed., 2000*
The task of neural science is to explain behavior in terms of the activities of the brain.


**Behavior**

- Response time
- Accuracy
  
**Brain Activity**

- Extracellular Recordings
- Electroencephalography (EEG)
- Magnetoencephalography (MEG)
- Functional Magnetic Resonance Imaging (fMRI)
  
...
Univariate analyses

Research Question
Experiment Design
Stimuli

SPM
GLM

Hypothesis Testing
Data-driven Approach: **Reverse the flow!**

- **Research Question**
- **Experiment Design**
- **Stimuli**
- **Neural Processing and Encoding**

**Information Integration Models**

**Decoder**

GLM
Data-driven Approach: Analysis

Research Question

Experiment Design

Stimuli

Decoder

Neural Processing and Encoding

Information Integration
Data-driven Approach

Stimuli

Reconstruction

Decoder

Research Question

Experiment Design

Stimuli

Neural Processing and Encoding
Data-driven Approach

Miyawaki et al. (2008)

Presented image (10 x 10 patches)

Local image bases (elements)

Reconstructed contrast pattern

Multi-voxel pattern decoders

Multi-scale image representation

Presented contrast pattern

Reconstructed contrast pattern

Mean of reconstructed contrast pattern

Miyawaki et al. (2008)

H₂ (Dartmouth; Magdeburg)
Data-driven Approach:

Contrast & Localization

Decoder

Research Question

Experiment Design

Stimuli

Neural Processing and Encoding
Data-driven Approach

GLM → MVPA Decoder → Models
Data-driven Approach
Exploratory Analysis

Decoder

Research Question → Experiment Design → Stimuli → Neural Processing and Encoding

$H_2$ (Dartmouth; Magdeburg)
Data-driven Approach

Poldrack et al. (2009)

Dimension 1
- audition
- language
- decisionmaking
- emotion
- implicitmemory
- phonology
- positiveemotion
- responseinhibition
- semantics
- spatialprocessing
- speech
- workingmemory

Dimension 2
- language
- orthography
- phonology
- responseinhibition
- reward
- semantics
- speech
- spatialprocessing
- vision
- workingmemory

H₂ (Dartmouth; Magdeburg)

PyMVPA

UPENN 2012
Data-driven Approach

neurosynth.org \textit{beta}

NeuroSynth is a platform for large-scale, automated synthesis of functional magnetic resonance imaging (fMRI) data extracted from published articles.

Our goal is to turn this:

\begin{center}
\includegraphics[width=\textwidth]{neurosynth_images}
\end{center}

Into this:

\begin{center}
\includegraphics[width=\textwidth]{brain_images}
\end{center}

\url{http://neurosynth.org}
Data-driven Approach

Welcome to Cognitive Atlas

The Cognitive Atlas is a collaborative knowledge building project that aims to develop a knowledge base (or ontology) that characterizes the state of current thought in cognitive science. The project is led by Russell Poldrack, Professor of Psychology and Neurobiology at the University of Texas at Austin in collaboration with the UCLA Center for Computational Biology (A. Toga, PI) and UCLA Consortium for Neuropsychiatric Phenomics (R. Bilder, PI). It is supported by grant RO1MH82795 from the National Institute of Mental Health.

http://cognitiveatlas.org
Advantages of multivariate approaches

- Discover distributed (contiguous or networks) functional units or biomarkers
- Encourage model testing
- Are capable of per-trial/sample analysis
- Account for various sources of variance and covariance/causal structure (Sato et al., 2008)
- Can relax modeling assumptions of the signals
- Benefit from inter-disciplinary methodological developments
- Can aggregate functional information without precise spatial correspondence
... there is also shadows

## Pitfalls of MVPA

- K. Friston about why PCA approach didn’t pick up in 1990s: “This reflects the fact that the scientific process in neuroimaging is essentially Popperian and is almost universally, driven by hypotheses.”

- Difficulties with the magic: **But is it significant?**
K. Friston about why PCA approach didn’t pick up in 1990s: “This reflects the fact that the scientific process in neuroimaging is essentially Popperian and is almost universally, driven by hypotheses.”

Difficulties with the magic: **But is it significant?**

- Multivariate methods are more flexible thus might be more sensitive to confounds and preprocessing
  - Trial-order/sample-groups effects
  - *Am I decoding what am I interested in?*

- Variety of multivariate methods with different properties
- Variety of *feature extraction* methods without an absolute winner (Mumford et al., 2011)
Available Generic Software

there is also shadows

### Available Generic Software


### Available Software Tailored Towards fMRI

- 3dsvm (AFNI plugin, LaConte et al., 2005)
- MATLAB® MVPA Toolbox (Detre et al., 2006)
- MATLAB® PROBIT Toolbox ([http://brainmap.co.uk/](http://brainmap.co.uk/))
- MATLAB® SearchMight ([http://minerva.csbmb.princeton.edu/searchmigh](http://minerva.csbmb.princeton.edu/searchmigh))
- NISL (NeuroImaging with the Scikit-Learn) ([http://nisl.github.com](http://nisl.github.com))
... there is also shadows

Available Generic Software


Unavailable Software

- Kriegeskorte et al. (2006), searchlight algorithm
- Kamitani and Tong (2005), unknown
- Pessoa and Padmala (2006), custom combination of LIBSVM and pieces of the MVPA MATLAB® toolbox
- ...
Software: Desired Features

- User-centered programmability with a intuitive and documented user interface
  - Concise scripting interface in Python, illustrated user manual

- Extensibility
  - Modular architecture to connect extensions in multiple languages

- Transparent reading and writing of neural data sets
  - e.g. NIfTI support for input and output

- Portability
  - Runs on anything from mainframes to cell phones

- Reliability/Consistency
  - Unit-, doc-, example- tests

- Open source software
  - MIT-licensed free software
Software: PyMVPA Features

- User-centered programmability with an intuitive and documented user interface
  - Concise scripting interface in Python, illustrated user manual

- Extensibility
  - Modular architecture to connect extensions in multiple languages

- Transparent reading and writing of neural data sets
  - e.g. NIfTI support for input and output

- Portability
  - Runs on anything from mainframes to cell phones

- Reliability/Consistency
  - Unit-, doc-, example- tests

- Open source software
  - MIT-licensed free software
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

```python
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

```
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'})
```

- **Experimental Runs**
- **Experimental Conditions**
- **fMRI Volumes**
- **Chunks**
- **Targets**
- **Samples**
- **Time**
- **Features**
- **Samples**
- **Targets**
- **Chunks**
- **.sa**
- **.targets**
- **.chunks**
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'})
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'})

print(dataset.summary())
```

<table>
<thead>
<tr>
<th>Dataset / int16  81 x 577</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniq: 3 chunks 3 labels</td>
</tr>
<tr>
<td>stats: mean=1670.84 std=344.597 var=118747 min=430 max=2707</td>
</tr>
</tbody>
</table>

Counts of labels in each chunk:
```
<table>
<thead>
<tr>
<th>chunks</th>
<th>bottle</th>
<th>cat</th>
<th>chair</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>1.0</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>2.0</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>
```

Summary per label across chunks
```
<table>
<thead>
<tr>
<th>label</th>
<th>mean</th>
<th>std</th>
<th>min</th>
<th>max</th>
<th>#chunks</th>
</tr>
</thead>
<tbody>
<tr>
<td>bottle</td>
<td>9</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>cat</td>
<td>9</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>chair</td>
<td>9</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>
```

Summary per chunk across labels
```
<table>
<thead>
<tr>
<th>chunk</th>
<th>mean</th>
<th>std</th>
<th>min</th>
<th>max</th>
<th>#labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>
```
Analysis Example: Classification

```python
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()
```
Analysis Example: Classification

```python
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
print cv.ca.stats.plot()
```
Analysis Example: Classification (Everyone matters)

\[
\begin{align*}
\text{attr} &= \text{SampleAttributes('attributes.txt')} \\
\text{ds} &= \text{fmri_dataset(samples='bold.nii.gz', targets=attr.targets, chunks=attr.chunks, mask='mask_brain.nii.gz', add_fa='vt:vt.nii.gz'})
\end{align*}
\]

\[
\begin{align*}
\text{clf} &= \text{LinearCSVMC()} \\
\text{cv} &= \text{CrossValidation(clf, NFoldPartitioner(), enable_ca=['stats'])} \\
\text{print cv.ca.stats.errors=cv(ds)} \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>predict.\targets</th>
<th>3kHz</th>
<th>7kHz</th>
<th>12kHz</th>
<th>20kHz</th>
<th>30kHz</th>
<th>song1</th>
<th>song2</th>
<th>song3</th>
<th>song4</th>
<th>song5</th>
</tr>
</thead>
<tbody>
<tr>
<td>3kHz / 38</td>
<td>84</td>
<td>42</td>
<td>27</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td>7kHz / 39</td>
<td>43</td>
<td>94</td>
<td>16</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>24</td>
<td>183</td>
</tr>
<tr>
<td>12kHz / 40</td>
<td>21</td>
<td>16</td>
<td>103</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>20kHz / 41</td>
<td>1</td>
<td>2</td>
<td>13</td>
<td>158</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>30kHz / 42</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>162</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>song1 / 43</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>160</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>173</td>
</tr>
<tr>
<td>song2 / 44</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>171</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>song3 / 45</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>170</td>
<td>2</td>
</tr>
<tr>
<td>song4 / 46</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>139</td>
<td>7</td>
</tr>
<tr>
<td>song5 / 47</td>
<td>10</td>
<td>14</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>104</td>
</tr>
</tbody>
</table>

| Per target:      |      |      |       |       |       |       |       |       |       |       |
| N                 | 1560 | 1560 | 1561  | 1561  | 1561  | 1561  | 1561  | 1561  | 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: |     |      |       |       |       |       |       |       |       |       |
| ACC               | 0.78 |      |       |       |       |       |       |       |       |       |
| ACC\%             | 77.57|      |       |       |       |       |       |       |       |       |
| # of sets         | 8    |      |       |       |       |       |       |       |       |       |

Hanke&Halchenko et al., Frontiers in Neuroinformatics, 2009 (Suppl.)
## Analysis Example: Classification

### Code Snippet

```python
attr = SampleAttributes('attributes.txt')
ds = ...
```

### Analysis

#### Data Description
- **Attributes:**
  - Sample attributes are loaded from `attributes.txt`.

#### Dataset Configuration
- **Image Files:**
  - `bold.nii.gz`
  - `mask_brain.nii.gz`

- **Targets:**
  - `attr.targets`

- **Chunks:**
  - `attr.chunks`

- **Feature Additions:**
  - **vT:**
    - `vt.nii.gz`

### Cross-Validation Setup
- **Cross-Validation:**
  - `CrossValidation(clf, NFoldPartitioner(), enable_ca=['stats'])`

- **Classification Model:**
  - `clf = LinearCSVMC()`

### Error Calculation
- **Errors:**
  - `errors = cv(ds)`

### Visualization

#### Confusion Matrix
- `chunk 1`
- `chunk 2`
- `chunk 3`
- `chunk 4`
- `chunk 5`

#### Confusion Matrix Details
- **Row:**
  - **3kHz**
  - **7kHz**
  - **12kHz**
  - **20kHz**
  - **30kHz**

- **Column:**
  - **targets**
  - **song1**
  - **song2**
  - **song3**
  - **song4**
  - **song5**

#### Color Scale
- **Range:**
  - 0 to 162

#### Color Mapping
- **Red:**
  - 162
- **Blue:**
  - 0

### Additional Notes
- `H2 (Dartmouth; Magdeburg)`
- `PyMVPA`
- `UPENN 2012`
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()

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

errors = cv(ds)

print cv.ca.stats

print cv.ca.stats.plot()
```
Analysis Example: Searchlight

```python
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'})
cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])
cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])
clf = LinearCSVMC()
... = cv(ds)
```

The searchlights over London
Are like the fingers of a woman,
Wandering over the dead form of a lover. . . .

Analysis Example: Searchlight

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'])

<table>
<thead>
<tr>
<th></th>
<th>3kHz</th>
<th>7kHz</th>
<th>12kHz</th>
<th>20kHz</th>
<th>30kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>song1</td>
<td>84</td>
<td>43</td>
<td>21</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>song2</td>
<td>42</td>
<td>94</td>
<td>16</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>song3</td>
<td>27</td>
<td>16</td>
<td>103</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>song4</td>
<td>4</td>
<td>0</td>
<td>13</td>
<td>158</td>
<td>1</td>
</tr>
<tr>
<td>song5</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Per target:

- P: 174
- N: 1560
- TP: 84
- TN: 1221

Summary Means:

- ACC: 0.78
- ACC%: 77.57

Haynes and Rees, Current Biology, 2007
Analysis Example: Searchlight

```python
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'})
```

<table>
<thead>
<tr>
<th></th>
<th>3kHz</th>
<th>7kHz</th>
<th>12kHz</th>
<th>20kHz</th>
<th>30kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>song1</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td>song2</td>
<td>43</td>
<td>94</td>
<td>16</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>song3</td>
<td>21</td>
<td>16</td>
<td>103</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>song4</td>
<td>1</td>
<td>2</td>
<td>13</td>
<td>158</td>
<td>1</td>
</tr>
<tr>
<td>song5</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>162</td>
</tr>
</tbody>
</table>

Per target:

```plaintext
<table>
<thead>
<tr>
<th></th>
<th>3kHz</th>
<th>7kHz</th>
<th>12kHz</th>
<th>20kHz</th>
<th>30kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>174</td>
<td>174</td>
<td>173</td>
<td>173</td>
<td>173</td>
</tr>
<tr>
<td>N</td>
<td>1560</td>
<td>1560</td>
<td>1561</td>
<td>1561</td>
<td>1561</td>
</tr>
<tr>
<td>TP</td>
<td>84</td>
<td>94</td>
<td>103</td>
<td>158</td>
<td>162</td>
</tr>
<tr>
<td>TN</td>
<td>1261</td>
<td>1251</td>
<td>1242</td>
<td>1187</td>
<td>1183</td>
</tr>
</tbody>
</table>
```

Summary Means:

```plaintext
<table>
<thead>
<tr>
<th></th>
<th>3kHz</th>
<th>7kHz</th>
<th>12kHz</th>
<th>20kHz</th>
<th>30kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>0.78</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACC%</td>
<td>77.57</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># of sets</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

$H_2$ (Dartmouth; Magdeburg)

PyMVPA
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: Searchlight

\[
\text{e} = ()
\]

\[
\Sigma(e)
\]

\[
\Sigma(e)
\]

\[
\text{e} = (,,,,)
\]

\[
\text{attr} = \text{SampleAttributes('attributes.txt')}
\]

\[
ds = \text{fmri_dataset(samples='bold.nii.gz', targets=attr.targets, chunks=attr.chunks, mask='mask_brain.nii.gz', add_fa={'vt':'vt.nii.gz'})}
\]

\[
\text{clf} = \text{LinearCSVMC()}
\]

\[
\text{cv} = \text{CrossValidation(clf, NFoldPartitioner(), enable_ca=['stats'])}
\]

\[
\text{fwm} = \text{sphere_searchlight(cv, radius=3)}
\]

\[
\text{fwm_map} = \text{fwm(ds)}
\]
Analysis Example: Searchlight (Data says a lot)

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

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

cclf = sphere_searchlight(cv, radius=3)
fwm_map = fwm(ds)
```
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'})
cv = CrossValidation(clf, NFoldPartitioner(),
enable_ca=['stats'])
fwm = sphere_searchlight(cv, radius=3)
fwm_map = fwm(ds)
Analysis Example: Searchlight

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'})

cf = LinearCSVMC()

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

fwm = sphere_gnbsearchlight(GNB(), N FoldPartitioner(), radius=3)
fwm_map = fwm(ds)
Complete Analysis Example: Searchlight

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: Searchlight

Hanke & Halchenko et al., Neuroinformatics, 2009
Complete Analysis Example: Searchlight

```python
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'})

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: Searchlight**

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, N FoldPartitioner(),
                     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)
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'})

1D 3D
0.0 0.36 0.24 0.12
1 mm 5 mm 10 mm 20 mm
y=-61 y=-69 y=-59 y=-54

--

3kHz  7kHz  12kHz 20kHz 30kHz
song1  song2  song3  song4  song5

predict.
targets  38  39  40  41  42  43  44  45  46  47

3kHz / 38  84  42  27  4  4  2  1  0  15  19
7kHz / 39  43  94  16  0  1  1  1  2  1  24
12kHz / 40  21  16  103  5  2  2  0  0  6  13
20kHz / 41  1  2  13  158  1  0  0  1  3  1
30kHz / 42  3  0  2  3  162  0  0  0  0  0
song1 / 43  3  1  1  0  1  160  0  0  2  5
song2 / 44  1  1  0  0  0  0  171  0  0  0
song3 / 45  1  1  1  0  0  0  0  170  2  0
song4 / 46  7  3  3  2  2  2  0  0  139  7
song5 / 47  10  14  7  1  0  7  0  1  5  104

Per target:  

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:  

ACC  0.78
ACC%  77.57
# of sets  8
3kHz
7kHz
12kHz
20kHz
30kHz
song1
song2
song3
song4
song5
predictions

map2nifti(ds, fwm_map).
to_filename('out.nii.gz')
h5save('out.hdf5', fwm_map)
fwm = sphere_searchlight(cv, radius=3)
fwm_map = fwm(ds)

cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])
clf = LinearCSVMC()
Analysis Example: Spatio-temporal Searchlight

\[
\begin{align*}
\text{chunk 1} & \\
\text{chunk 2} & \\
\text{chunk 3} & \\
\text{chunk 4} & \\
\text{chunk 5} & \\
\text{attr} & = \text{SampleAttributes('attributes.txt')} \\
ds & = \text{fmri_dataset(samples='bold.nii.gz', targets=attr.targets, chunks=attr.chunks, mask='mask_brain.nii.gz', add_fa={vt: 'vt.nii.gz'})} \\
\end{align*}
\]

\[
\begin{align*}
\Sigma(e) & \\
\end{align*}
\]

\[
\begin{align*}
\text{map2nifti(ds, fwm_map).to_filename('out.nii.gz')} \\
\text{h5save('out.hdf5', fwm_map)} \\
fwm & = \text{sphere_searchlight(cv, radius=3)} \\
fwm_map & = \text{fwm(ds)} \\
\text{cv} & = \text{CrossValidation(clf, NFoldPartitioner(), enable_ca=['stats'])} \\
\text{clf} & = \text{LinearCSVMC()} \\
\end{align*}
\]
Analysis Example: Searchlight

```
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()

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)
```
Analysis Example: Sensitivities

```
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 = clf.get_sensitivity_analyzer()
fwm_map = fwm(ds)
map2nifti(ds, fwm_map).to_filename('out.nii.gz')
h5save('out.hdf5', fwm_map)
```
Analysis Example: Sensitivities

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'])

h5save('out.hdf5', fwm_map)

Hanke&Halchenko et al., Frontiers in Neuroinformatics, 2009
Analysis Example: Sensitivities

```
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()

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

fwm = clf.get_sensitivity_analyzer()

fwm_map = fwm(ds)

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

h5save('out.hdf5', fwm_map)
```
Analysis Example: SVM-Sensitivities

\[ e = ( , , , , ) \]

\[ \Sigma(e) \]

\[ M(s) \]

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' })
cv = CrossValidation(clf, NFoldPartitioner(),
enable_ca=[ 'stats' ])
fwm = clf.get_sensitivity_analyzer()
fwm_map = fwm(ds)

map2nifti(ds, fwm_map).to_filename('out.nii.gz')
h5save('out.hdf5', fwm_map)
Analysis Example: GPR-Sensitivities

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 = GPR()
cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])
fwm = clf.get_sensitivity_analyzer()
fwm_map = fwm(ds)

map2nifti(ds, fwm_map).to_filename('out.nii.gz')
h5save('out.hdf5', fwm_map)
Analysis Example: SMLR-Sensitivities

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 = SMLR()
cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])
fwm = clf.get_sensitivity_analyzer()
fwm_map = fwm(ds)
map2nifti(ds, fwm_map).to_filename('out.nii.gz')
h5save('out.hdf5', fwm_map)
Analysis Example: Beware

Figure 1. The highest expected chance accuracy as a function of the number of examples and classifiers. Each line represents a different number of examples. The x-axis is the number of classifiers and the y-axis is the accuracy.

Palatucci&Carlson, On the Chance Accuracies of Large Collections of Classifiers, 2008
And With That We Could Do:

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

H2 (Dartmouth; Magdeburg)
And With That We Could Do: EEG → fMRI
And With That Others Could Do:

Studies employing PyMVPA

- **Raizada and Connolly, Cognitive Neuroscience (In press):** What makes different people’s representations alike: neural similarity-space solves the problem of across-subject fMRI decoding. Preprint PDF and code are available
- **Jimura and Poldrack, Neuropsychologia (2011):** Analyses of regional-average activation and multivoxel pattern information tell complementary stories
- **Carlin et al., Current Biology (2011):** A head view-invariant representation of gaze direction in anterior superior temporal sulcus
- **Kaunitz et al., Frontiers in Perception Science (2011):** Intercepting the first pass: rapid categorization is suppressed for unseen stimuli.
- **Carlin et al., Cerebral Cortex (2011):** Direction-Sensitive Codes for Observed Head Turns in Human Superior Temporal Sulcus.
- **Kubilius et al., Psychological Science (2011):** Emergence of perceptual gestalt in the human visual cortex: The case of the configural superiority effect. Complete suite of sources from stimuli delivery (PsychoPy) to data analysis (PyMVPA) is available
- **Manelis et al., Cerebral Cortex (2011):** Dynamic Changes In The Medial Temporal Lobe During Incidental Learning Of Object–Location Associations.
- **Meyer et al., Cerebral Cortex (2011):** Seeing Touch Is Correlated with Content-Specific Activity in Primary Somatosensory Cortex.
- **Clithero et al., NeuroImage (2010):** Within- and cross-participant classifiers reveal different neural coding of information.
- **Cohen et al., Frontiers in Human Neuroscience (2010):** Decoding Developmental Differences and Individual Variability in Response Inhibition Through Predictive Analyses Across Individuals.
- **Meyer et al., Nature Neuroscience (2010):** Predicting visual stimuli based on activity in auditory cortices.
- **Manelis et al., Human Brain Mapping (2010):** Implicit memory for object locations depends on reactivation of encoding-related brain regions.
- **Sun et al., Biological Psychiatry (2009):** Elucidating an MRI-Based Neuroanatomic Biomarker for Psychosis: Classification Analysis Using Probabilistic Brain Atlas and Machine Learning Algorithms.
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: Searchite
  - Measures
  - Searching, searching, searching, ...
  - For real!
- Part 6: Looking Without Searching – Sensitivity Analysis
  - It’s A Kind Of Magic
Similarity Analyses

Andrew Connolly: http://haxbylab.dartmouth.edu/ppl/andy.html
Similarity Analyses: V1 vs behavioral models

A. Behavioral ratings DM

B. Similarity searchlight: Behavioral DM

C. V1 model DM

D. Similarity searchlight: V1 model DM

Similarity Analyses: Cross-subject agreement

A. SVM searchlight

B. Cross-subject similarity correlation searchlight

Similarity Analyses . . .

- provide richer picture than mean generalization error maps
- investigate the transformation of the representations through processing streams
- unravel functional regions with information represented according to theoretical models of perception or behavior observations
- relax requirement for precise anatomical correspondence across subjects
Hyperalignment

J. Swaroop Guntupalli: http://haxbylab.dartmouth.edu/ppl/swaroop.html

H₂ (Dartmouth; Magdeburg)

PyMVPA

UPENN 2012 22 / 14
Led by Dartmouth's James Haxby, neuroscientists unlock shared brain codes

Published: Thursday, October 20, 2011 - 14:34 in Psychology & Sociology

A team of neuroscientists at Dartmouth College has shown that different individuals' brains use the same, common neural code to recognize complex visual images. Their paper is published in the October 20, 2011, issue of the journal Neuron. The paper's lead author is James Haxby, the Evans Family Distinguished Professor of Cognitive Neuroscience in the Department of Psychological and Brain Sciences. Haxby is also the director of the Cognitive Neuroscience Center at Dartmouth and a professor in the Center for Mind/Brain Sciences at the University of Trento in Italy. Swaroop Guntupalli, a graduate student in Haxby's laboratory, developed software for the project's methods and ran the tests of their validity.

Haxby developed a new method called hyperalignment to create this common code and the parameters that transform an individual's brain activity patterns into the code.
Hyperalignment: Deriving a common space
Hyperalignment: Classification (VT cortex)

![Classification accuracy chart](chart.png)

**Classification accuracy** (±SE)

- **Between-subject classification**
  - 1000 hyperaligned dimensions
  - 35 hyperaligned PCs
  - 1000 anatomically-aligned voxels

- **Within-subject classification**
  - 1000 voxels

- **Movie 18” segments**
- **Faces & objects**
- **Animal species**
Hyperalignment: Classification (movie segments)

Between subject classification of 18s movie time segments (chance~ 1 in 1000) using data from AC, VT, and STS and their combinations. Using AC+VT or STS+VT, we can predict which movie segment a subject is watching with ~80% accuracy based on other subject’s data.
Hyperalignment: Functional specificity

Figure 6. Contrast-Defined Category-Selective Profiles in the Common Model Space Projected into the Native Voxel Spaces of Two Subjects
(A) The topography associated with the contrast between mean response to faces as compared to the mean response to nonface objects (houses, chairs, and shoes). Note the tight correspondence of the regions with positive weights and the outlines of individually defined FFAs.
(B) FFA and PPA regions defined by contrasts in group data projected into the native voxel spaces of two subjects. For each subject, that subject’s own data were excluded from the calculation of face selectivity and house selectivity, yielding category-selective regions that were based exclusively on other subjects’ data. Each subject’s individually defined FFAs and PPAs are shown as outlines to illustrate the tight correspondence with model-defined category-selective regions.

Hyperalignment...

- aligns functional topologies when aligning anatomically is not adequate
- transformation
  - provides a common model of reduced dimensionality
  - transfers across experiments
  - can be used to derive subject-specific functional localizers from group data

---

Haxby, Neural Computation workshop (2011) video:
http://haxbylab.dartmouth.edu/meetings/ncworkshop11.html#jim-haxby-dartmouth
Hyperalignment...

- aligns functional topologies when aligning anatomically is not adequate
- transformation
  - provides a common model of reduced dimensionality
  - transfers across experiments
  - can be used to derive subject-specific functional localizers from group data
- is highly sensitive

Haxby, Neural Computation workshop (2011) video:
http://haxbylab.dartmouth.edu/meetings/ncworkshop11.html#jim-haxby-dartmouth
PyMVPA: Standing on the shoulders of giants

The PyMVPA Framework

Data Storage & Transformation
- Dataset: Mapped, Masked, Channel, Nifti
- Sample Attributes: Labels, Chunks
- Mapper: Array, Boxcar, Wavelet, ...

Preprocessing
- Detrend, Z-Score, ...

Cross-Validation
- Resampling: N-Fold, Half, Odd-Even

Plotting
- Topography, Distances, ...

States & Harvester
- Meta: Feature-Selection, Binary, Combined, Multi-Class, Mapped, Split
- Basic: PLR, SMLR, kNN, GPR, GNB, BLR, ...

Machine Learning

Feature Selection
- Simple Thresholding, RFE, IFS

Featurewise Measures
- Meta: Noise-Perturbation, Combined
- Basic: ANOVA, I-RELIEF, Classifier weights for SVM, SMLR, GPR

Mapper
- PCA, ICA

LibSVM
- Shogun

MDP

R
- via RPy

NumPy

SciPy

Nifti
- via PyNifti

.mat
- via SciPy.io

HDF5
- via H5Py

Any GNU/Linux Distribution

Mac OS X™

Windows™
Python in Neuroimaging

Stimuli delivery & Data access

- PsychoPy, Opensesame, VisionEgg
- Nibabel (superseeded PyNIfTI, PyMGH, etc.), h5py, PyTables
Python in Neuroimaging: PsychoPy

For stimulus generation and experimental control in python.

PsychoPy depends on your feedback. If something doesn't work then let me/us know at psychopy-users@googlegroups.com

Jonathan Poline

http://www.psychopy.org

Output

AL lib: alc 1.17.6; exit() closing 1 Device
AL lib: alc 1.16.0: alcCloseDevice(): destroying 1 Context[6]

Running: /home/joh/dev/git/pk-g-expsy/psychopy/psychopy/demos/code/gabor.py

# Initialising some stimuli

gabor = visual.PatchStim(win, tex='sin', mask='gauss', texRes=256,

... size=[1.0, 1.0], sf=4.0,

... ori=0)

message = visual.TextStim(win, pos=(0.0, -0.9), text='Hit Q to quit')

trialClock = core.Clock()
Python in Neuroimaging

Stimuli delivery & Data access

- PsychoPy, Opensesame, VisionEgg
- Nibabel (superseeded PyNIfTI, PyMGH, etc.), h5py, PyTables

Data analysis

- NumPy, SciPy, NetworkX
- MDP, scikit-learn, statsmodels, pandas, simpy
- IPython, Sage
- OpenMEEG, Dipy, NiPy, Nitime, NiPype, BrainVisa

NIPY features

- preprocessing: space-time realignment of fMRI data
- fMRI data analysis: GLM model (model specification, fit)
- inference:
  - parametric tests (false discovery rate, Gaussian Random Field theory)
  - non-parametric tests (voxel-level, cluster-level, mixed effects, various statistics)
- spatial models:
  - anatomo-functional parcellation,
  - structural models (brain functional landmarks)
Python in Neuroimaging: NiPype

Neuroimaging in Python
Pipelines and Interfaces

```python
import nipype.interfaces.fsl as fsl
mybet = fsl.Bet()
mybet.inputs.infile = 'foo.nii'
mybet.inputs.outfile = 'bar.nii'
result = mybet.run()
```

```python
import nipype.interfaces.spm as spm
from glob import glob
allepi = glob('epi*.nii')
allepi.sort()
realigner = spm.Realign()
realigner.inputs.infile = allepi
result = realigner.run()
```
Python in Neuroimaging

Data visualization

- matplotlib, Mayavi2
- ConnectomeViewer
- PySurfer, Anatomist

http://www.connectomeviewer.org

http://matplotlib.sourceforge.net/gallery.html

http://code.enthought.com/projects/mayavi/
Python in NeuroImaging

Find the community @ http://www.nipy.org

Stimuli Delivery

PsychoPy
http://www.psychopy.org

PsychoPy is an easy, precise, platform-independent package for stimulus presentation. Suitable for psychophysics, neuroimaging, and all areas of psychology.

- Huge variety of stimuli generated in real-time
- Cross-platform – run the same script on Linux, Win or OS X
- Flexible stimulus units (degrees, cm, or pixels)
- Coder interface for those that like to program
- Builder interface for those that don’t
- Input from keyboard, mouse, joystick or button boxes
- Multi-monitor support
- Automated monitor calibration (supported photometers)

OpenSesame
http://www.cogsci.nl/software/opensesame

OpenSesame is a graphical experiment builder for the social sciences.

- A comprehensive and intuitive graphical user interface
- WYSIWYG drawing tools for creating visual stimuli
- Cross-platform
- Python scripting for complex tasks
- A plug-in framework
- Compatibility (through plug-ins) with commonly used devices: (e.g. Eyelink eye trackers, serial response boxes, Mantra object tracker)
- Compatibility with popular Python libraries: PsychoPy, PyGame, PyOpenGL, etc.

Data I/O

Nibabel
http://nipy.org/nibabel

Nibabel provides read and write access to some common medical and neuroimaging file formats, including: ANALYZE (plain, SPM99, SPM2), GIFTI, NIfTI1, MINC, as well as PAR/REC. Nibabel is the successor of PyNIfTI.

The various image format classes give full or selective access to header (meta) information and access to the image data is made available via NumPy arrays.

Analysis

BrainVISA
http://brainvisa.info

BrainVISA is an open-source, modular and customizable software platform built to host heterogeneous tools dedicated to neuroimaging research. It aims at helping researchers in developing new neuroimaging tools, sharing data and distributing their software.

- Databasing capabilities
- Massive computation facilities using Soma-workflow
- Open environment, with many toolboxes
- Specialized toolboxes for T1 MRI processing, sulci and gyri morphometry, diffusion imaging and fibers tracking, surfacic and structural analysis, 3D histology...
- Links with other software like SPM, FSL, FreeSurfer, or CIVET

DiPy
http://nipy.org/dipy

DiPy is an international FOSS project for diffusion magnetic resonance imaging analysis. DiPy is multiplatform and will run under any standard operating system such as Windows, Linux, Mac OS X. Some of our state-of-the-art applications are:

- Reconstruction algorithms e.g. GQI, DTI
- Tractography generation algorithms e.g. EuDX
- Intelligent downsampling of tracks
- Ultra fast tractography clustering
- Resampling datasets with anisotropic voxels to isotropic
- Visualizing multiple brains simultaneously
- Finding track correspondence between different brains
- Warping tractographies into another (e.g. MNI) space
- Support of various file formats e.g. Trackvis or NIfTI
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?**

- Impossibility to obtain funding for software development and maintenance (alone)
- Development of software tools often not considered scientific progress
We crave for *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 crave for *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 “loose 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 ([Hanke & Halchenko, Front. Neuroinf., 2011](#))

- Upgrading requires finding webpages, getting accounts, reading documentaion, downloading huge archives, running various installers, scripts [da capo al fine]
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 (Hanke & Halchenko, Front. Neuroinf., 2011)
- Upgrading requires finding webpages, getting accounts, reading documentaion, downloading huge archives, running various installers, scripts [da capo al fine]

But

Quick dissemination of new features and bug fixes is essential for efficiency
Why don’t we all use the same platform... 

- that works on all devices, operating systems, ...
- that is guaranteed to be available for as long as we want, wherever we want
- that we can freely share with anyone
- 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
Aww, come on!
Seriously, how do we get there?

Role model **debian**

- Origin of most active software distributions
- Vast archive of maintained software (≈30000 binary packages) – proven procedures
- Self-governed, “do-ocracy”, no need to earn money, going strong for 20 years
Seriously, how do we get there?

Role model: debian

Origin of most active software distributions

Vast archive of maintained software (≈ 30,000 binary packages) – proven procedures

Self-governed, “do-ocracy,” no need to earn money, going strong for 20 years

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

http://distrowatch.com
Seriously, how do we get there?

**Role model Debian**

- Origin of most active software distributions
- Vast archive of maintained software (≈30000 binary packages) – proven procedures
- Self-governed, "do-ocracy", no need to earn money, going strong for 20 years

**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
NeuroDebian from a researcher’s perspective

<table>
<thead>
<tr>
<th>Task</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install simple editor</td>
<td><code>apt-get install gedit</code></td>
</tr>
<tr>
<td>Install complex MRI analysis package</td>
<td><code>apt-get install fsl</code></td>
</tr>
<tr>
<td>Install software collection for psycho-physics</td>
<td><code>apt-get install science-psychophysics</code></td>
</tr>
<tr>
<td>Keep the whole system up-to-date</td>
<td><code>apt-get upgrade</code></td>
</tr>
</tbody>
</table>
After X years and the contributions of many people:
Researchers/users

- Configure Debian/Ubuntu box or download virtual machine – visit http://neuro.debian.net for instructions
- `apt-get install mricron` (thousands of packages)
- `apt-get upgrade` (full system update)
- Get support at neurodebian-users@lists.alioth.debian.org
NeuroDebian: The virtual machine

- 32/64bit, multi-core
- Most convenient solution for Mac OS X, Windows
- Base image with setup wizard, fully functional within minutes
- Great for teaching, workshops, development, analysis

http://neuro.debian.net/vm.html
Walkthrough: NeuroDebian on Mac OS X

Research environment on other operating systems. This virtual machine initially comes as a compact Debian installation that can, once installed, be equipped with a large variety of neuroscience software with just a few mouse clicks (e.g. AFNI, Caret, FSL, PyMVPA).

Downloads

- NeuroDebian 6.0.0 image (amd64) [-570MB]
- VirtualBox download page (Windows, Linux, Mac, Solaris)

Installation

First download and install a recent version of VirtualBox. VirtualBox is a virtualization software that is freely available for Windows, MacOS X, Solaris, and Linux. VirtualBox comes with a comprehensive manual that should answer potential problems with virtual box installation.
Walkthrough: NeuroDebian on Mac OS X
Walkthrough: NeuroDebian on Mac OS X
Walkthrough: NeuroDebian on Mac OS X

Welcome to the NeuroDebian virtual machine!

This is the first time you have started this virtual machine. We will now perform a few simple steps to finalize its configuration. This includes upgrading the system to the latest available versions and installing additional software packages upon your request.
Walkthrough: NeuroDebian on Mac OS X
Walkthrough: NeuroDebian on Mac OS X
Walkthrough: NeuroDebian on Mac OS X

H2 (Dartmouth; Magdeburg) NeuroDebian
## NeuroDebian focus

### Researchers/users

- Configure Debian/Ubuntu box or download virtual machine
- `apt-get install mricron` *(thousands of packages)*
- `apt-get upgrade` *(full system update)*
- Get support at neurodebian-users@lists.alioth.debian.org

*Developers*

- Low-latency dissemination channel to thousands of users
- Platform-specific expertise available

*Maintainers*

- Uniform environment for cross-project improvements and QA
## NeuroDebian focus

### Researchers/users
- Configure Debian/Ubuntu box or download virtual machine
- `apt-get install mricron` (thousands of packages)
- `apt-get upgrade` (full system update)
- Get support at neurodebian-users@lists.alioth.debian.org

### Developers
- Low-latency dissemination channel to thousands of users
- Platform-specific expertise available
NeuroDebian focus

Researchers/users

- Configure Debian/Ubuntu box or download virtual machine
- `apt-get install mricron` (thousands of packages)
- `apt-get upgrade` (full system update)
- Get support at neurodebian-users@lists.alioth.debian.org

Developers

- Low-latency dissemination channel to thousands of users
- Platform-specific expertise available

Maintainers

- Uniform environment for cross-project improvements and QA
NeuroDebian popularity statistics

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
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
What are inside outs of NeuroDebian?

- QA/Testing
  - Build-time tests
  - Interoperability testing
  - Maintainer tests
  - Reproducible research
  - Portability
  - Teaching
  - Virtual machine

- Packaging
  - Maintenance
  - Mentorship
  - Library/compiler transitions
  - Proxy bug reports
  - Portability patches

- Repository
  - Software
    - Debian
      - i386
      - amd64
    - Ubuntu
      - i386
      - amd64
  - Data
    - Usage statistics
    - Snapshots
  - DE
  - GR
  - US-CA
  - US-NH
  - US-TN
  - Mirrors

- Communication
  - Online
    - IRC (#neurodebian on OFTC)
    - Publications
    - Mailing lists
    - NeuroDebian Insider Blog
    - Team@neuro.debian.net

- Personal
  - Talks
  - Conference booth

- Software
  - Blends
  - Debian Science
  - Debian Med

- Archive
  - BTS
  - QA
    - i386
      - amd64
      - armel
      - mips(elf)
      - kfreebsd-i386/amd64
      - ...
  - Derivatives

- Mirrors
  - Debian
    - Windows
    - Mac OS X
    - Other

- Derivatives
  - MacOSX
  - Other
For whom is NeuroDebian?

You want to . . .

- have **readily usable** software at your fingertips
- have the **latest developments** of research software
- use a **rock-solid** operating system
- **try something new**, without investing much time
- offer **students** a fully functional “take-away” research environment
- **efficiently collaborate** with other researchers
- **escape limitations** of an institutional computing environment
- **waste less time** maintaining computers
- have **your own software** easily available for other’s to use
- **develop neuroscience software** without worrying about dependencies
- **help** make NeuroDebian more robust, and/or **built on top** of it

[http://neuro.debian.net](http://neuro.debian.net)
Get involved!

- Find and evaluate software
- Report bugs, send patches
- Support: Mailing list, IRC
  (http://neuro.debian.net/#contacts)
- Post on the NeuroDebian blog
- Help to (co-)maintain a package
- Package your own software
- Send us tests
- Spread the word

WE NEED HELP!

http://openhatch.org/+projects/NeuroDebian
Brain Download:

iz compltes.
Acknowledgements

Swaroop Guntupalli
Andrew Connolly
Per B. Sederberg
Emanuele Olivetti
Valentin Haenel
James M. Hughes
Scott Gorlins

FOSS developers of Python, NumPy, SciPy, Matplotlib, H5Py, Rpy, Shogun, scikit-learn, Inkscape, ...

Debian Community

Dr. Yaroslav O. Halchenko
Dartmouth College, NH, USA
yoh@onerussian.com

James V. Haxby
Stephen J. Hanson
Stefan Pollmann

about the slides:
should become available at

© 2007-2012
Yaroslav O. Halchenko & Michael Hanke,
Arno Klein

snake-brain © 2010
inspired by Stefano Zacchiroli

slide style

CC BY-SA 3.0 — Creative Commons Attribution-ShareAlike 3.0

H2 (Dartmouth; Magdeburg)
NeuroDebian
UPENN 2012
OS market share

A) Proportion of participants

Operating systems
- GNU/Linux (L)
- Debian based
- Red Hat based
- Windows (W)
- Mac OS X (M)
- Other OS

Usage time
- Always
- < 50%
- > 50%
- Never


