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

– A. and B. Stroogatskie *“The bug in an ant hill”*, 1979

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


RESE H LABO TOR
debian/rules
debian

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`

Goal of Neuroscience

> ***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*

Goal of Neuroscience

> ***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*

Behavior → **Brain Activity**

Response time

Accuracy

. . .

Extracellular Recordings

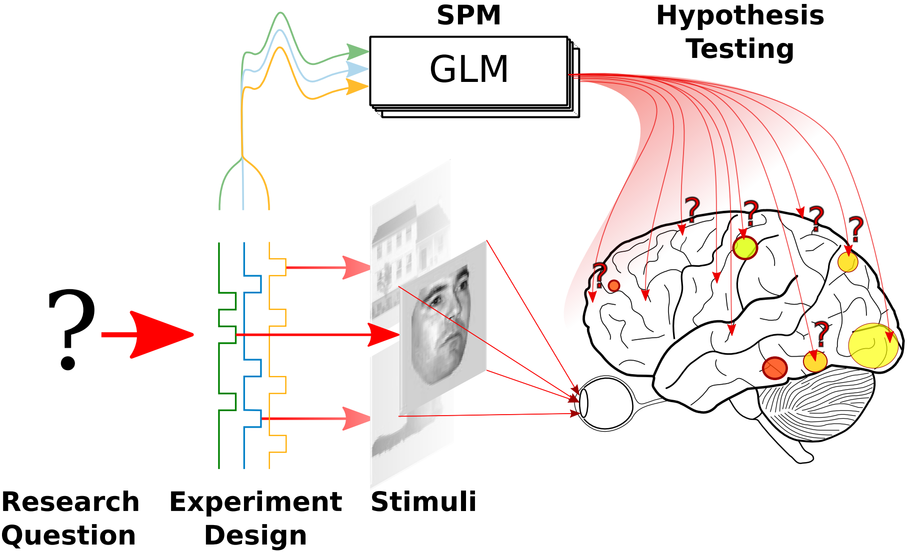
Electroencephalography (EEG)

Magnitoencephalography (MEG)

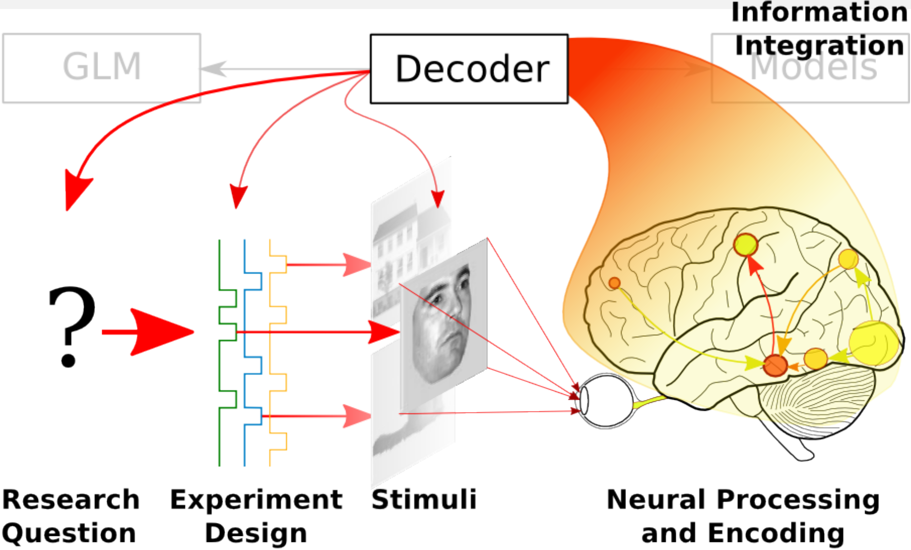
Functional Magnetic Resonance Imaging (fMRI)

. . .

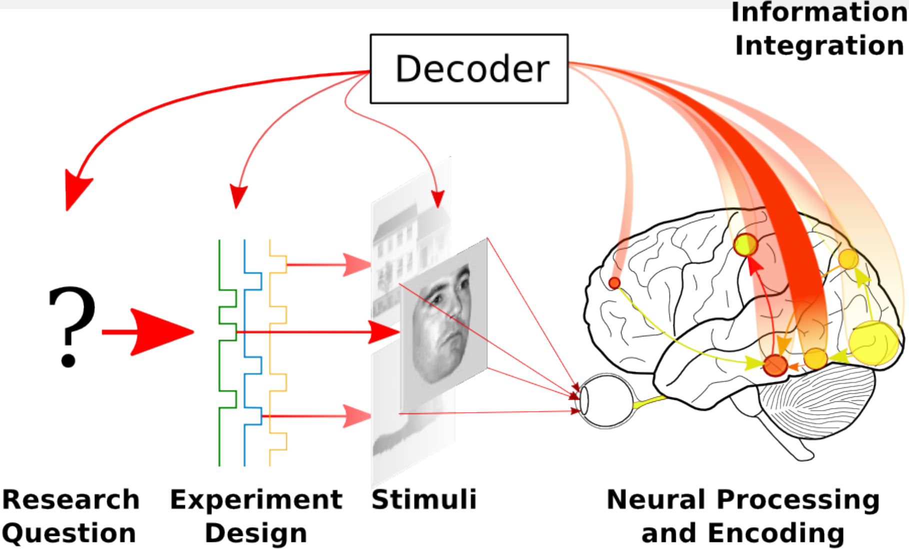
Univariate analyses



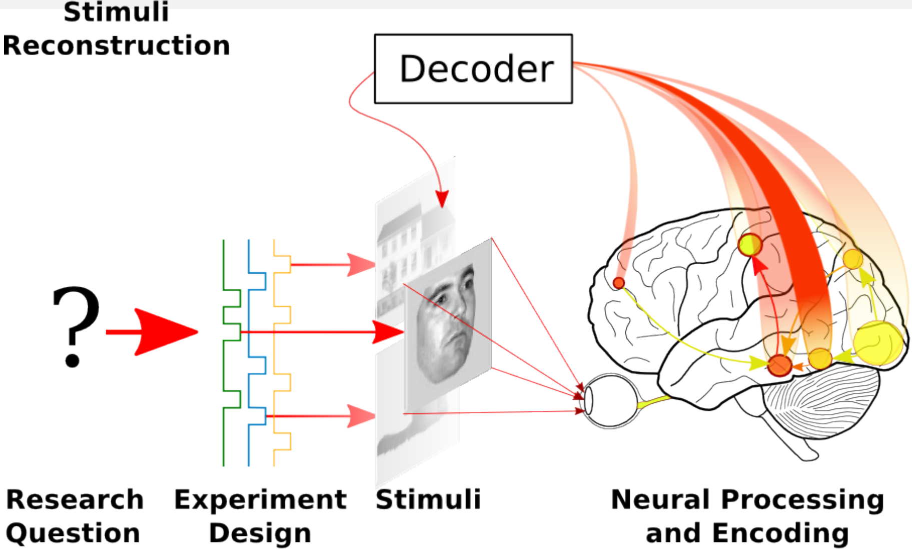
Data-driven Approach:Reverse the flow!



Data-driven Approach: Analysis



Data-driven Approach



Stimuli Reconstruction

Decoder

?

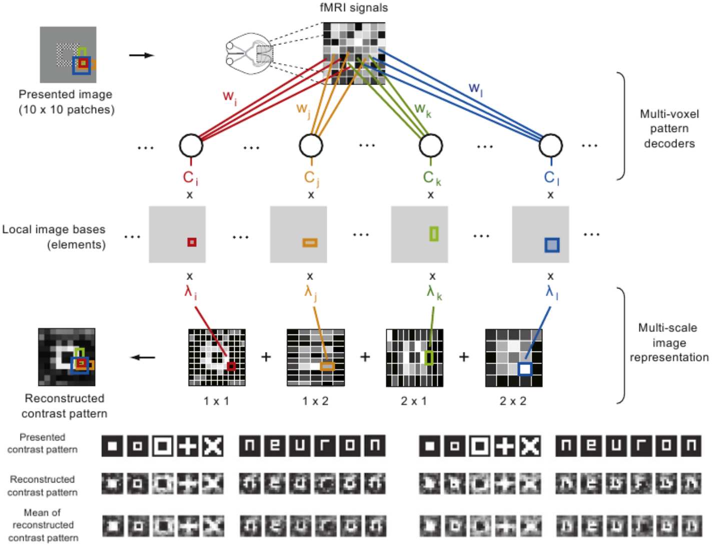
Research Question

Experiment Design

Stimuli

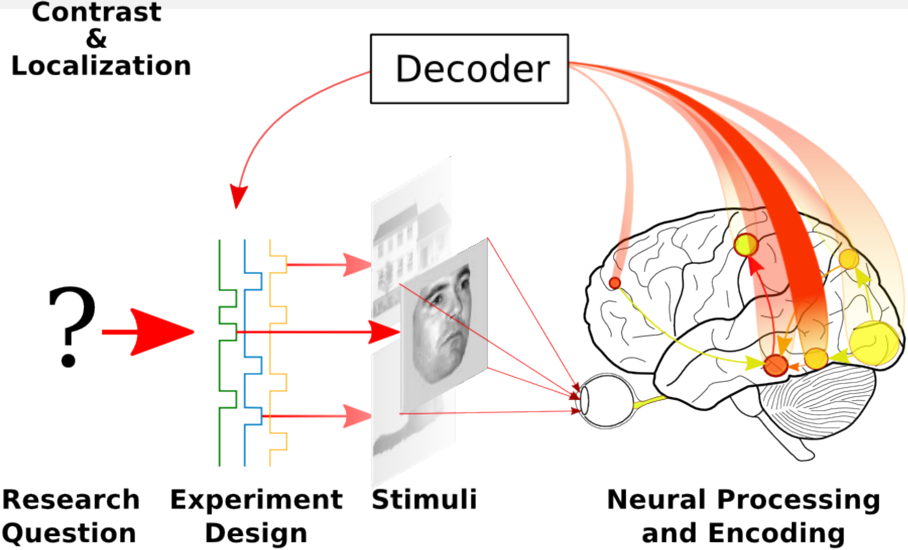
Neural Processing and Encoding

Data-driven Approach



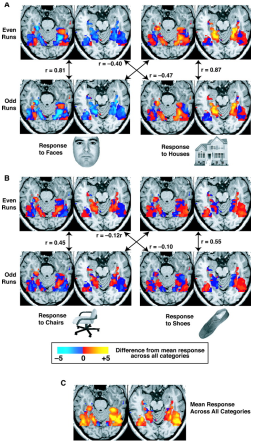
Miyawaki et al. (2008)

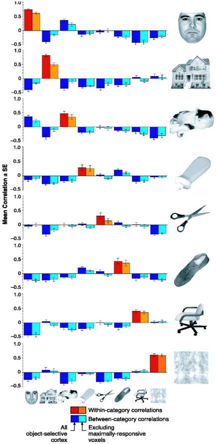
Data-driven Approach



Data-driven Approach

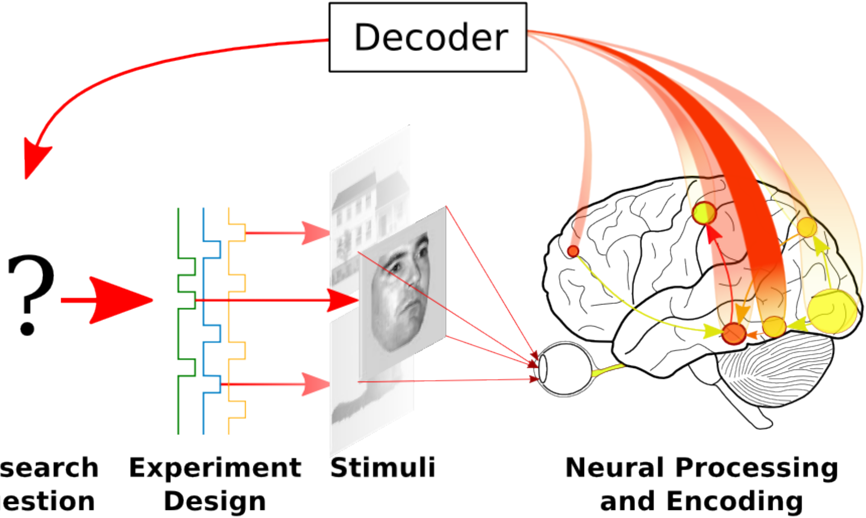
GLM ← **MVPA Decoder** → Models



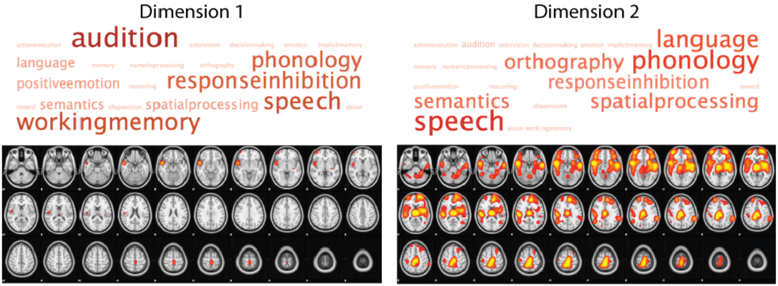


Data-driven Approach

Explorative Analysis

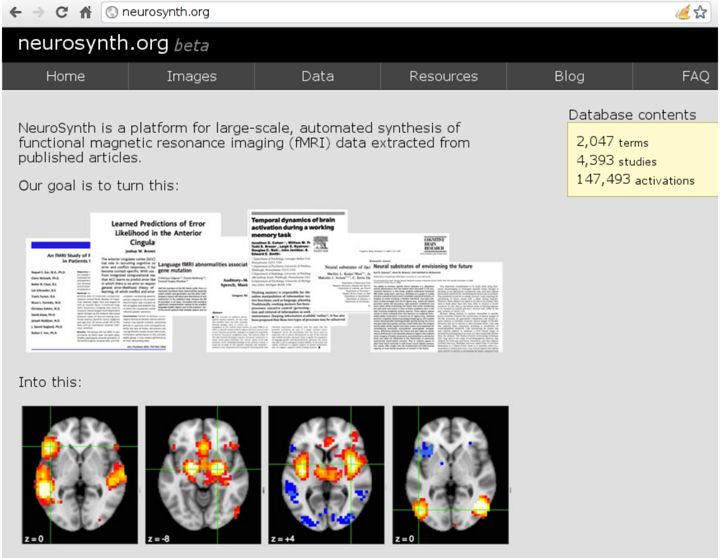


Data-driven Approach



Poldrack et al. (2009)

Data-driven Approach



Database contents

2,047 terms
4,393 studies
147,493 activations

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

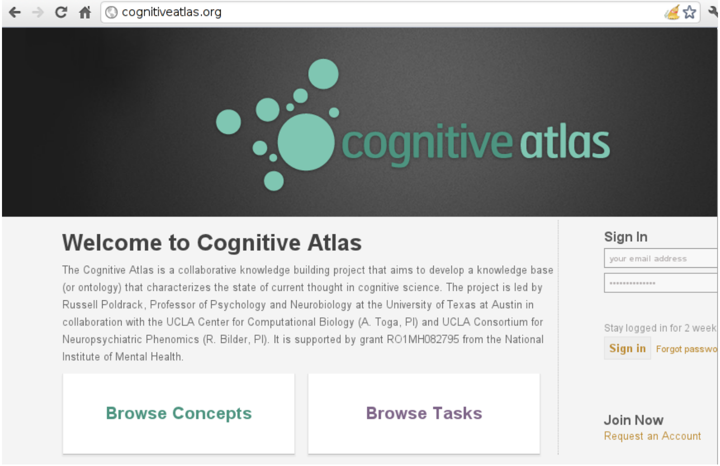
Our goal is to turn this:

Into this:

---

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 RO1MH082795 from the National Institute of Mental Health.

---

http://cognitiveatlas.org

Where there is light . . .

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?**

. . . 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?**
- 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)

. . . there is also shadows

Available Generic Software

- Machine Learning Open Source Software (`http://mloss.org`) lists 380 packages.

. . . there is also shadows

Available Generic Software

- Machine Learning Open Source Software (`http://mloss.org`) lists 380 packages.

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/`)
- *MATLAB® SearchMight* (`http://minerva.csbmb.princeton.edu/searchmigh`)
- *NISL (NeuroImaging with the Scikit-Learn)* (`http://nisl.github.com`)

. . . there is also shadows

Available Generic Software

- Machine Learning Open Source Software (`http://mloss.org`) lists 380 packages.

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

- Extensibility

- Transparent reading and writing of neural data sets

- Portability

- Reliability/Consistency

- Open source software

Software: PyMVPA 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

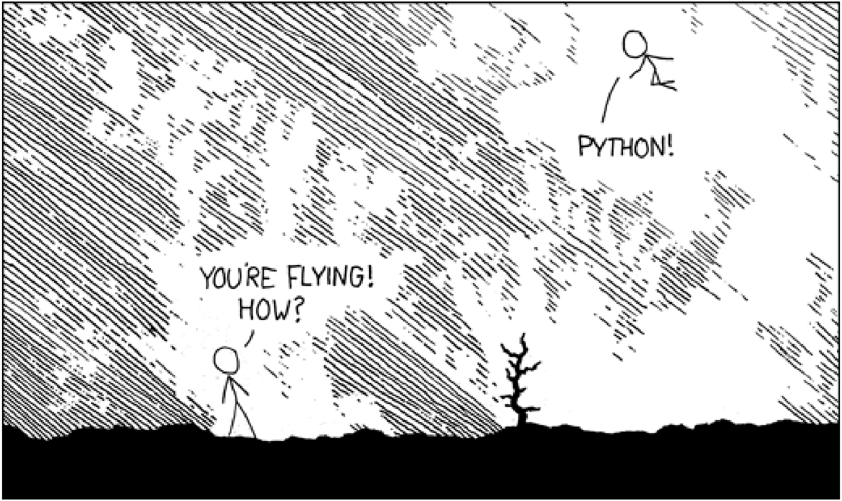
  ⇒e.g. NIfTI support for input and output
- Portability

  ⇒Runs on anything from mainframes to cell phones
- Reliability/Consistency

  ⇒Unit-, doc-, example- tests
- Open source software

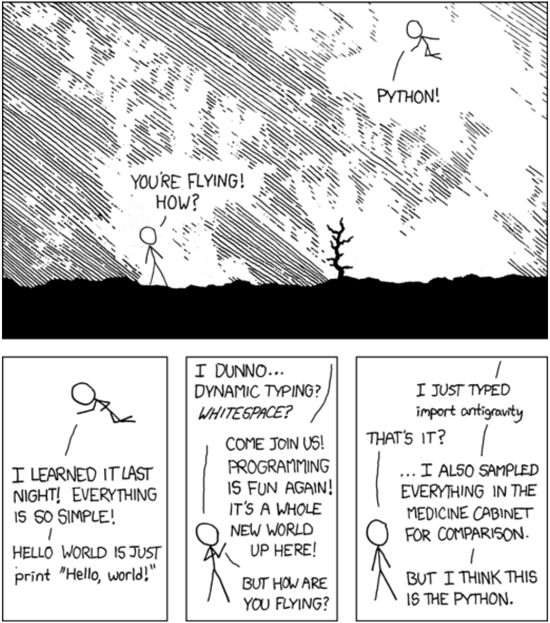
  ⇒MIT-licensed free software

Python



`http://xkcd.com/353/` Randall Munroe, CC BY-NC 2.5 License

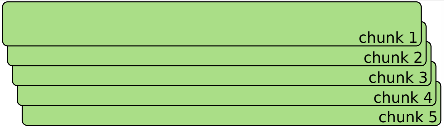
Python


YOU'RE FLYING! HOW?
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”

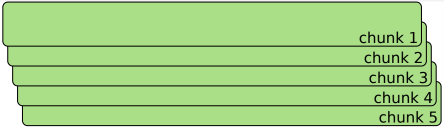
```
from mvpa2.suite import *
```

Analysis Example : Datasets

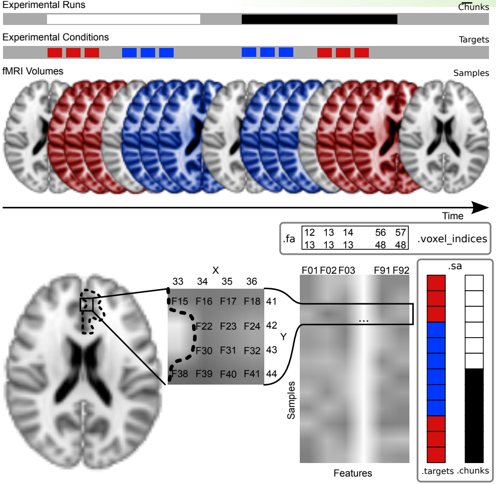


```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets,
                  chunks=attr.chunks,
                  mask='mask_brain.nii.gz',
                  add_fa={'vt':'vt.nii.gz'})
```

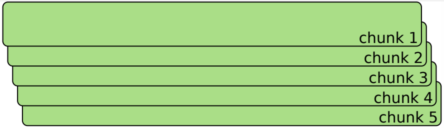
Analysis Example : Datasets



```
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'})
```



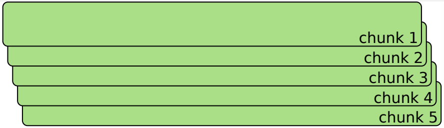
Analysis Example : Datasets



```
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]
```

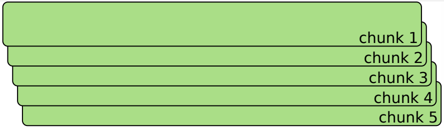
Analysis Example : Datasets



```
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()
```

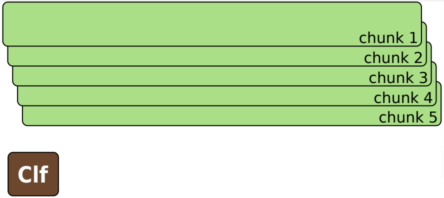
Analysis Example : Datasets



```
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



chunk 1

chunk 2

chunk 3

chunk 4

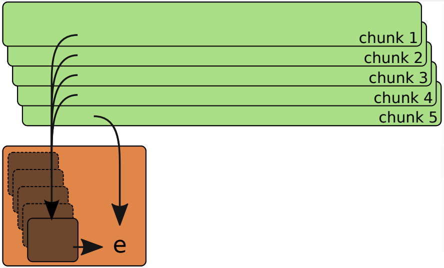
chunk 5

Clf

```
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'})
```

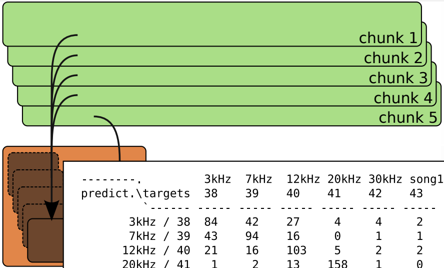
```
clf = LinearCSVMC()
```

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'})
```

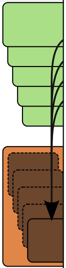
```
clf = LinearCSVMC()
```

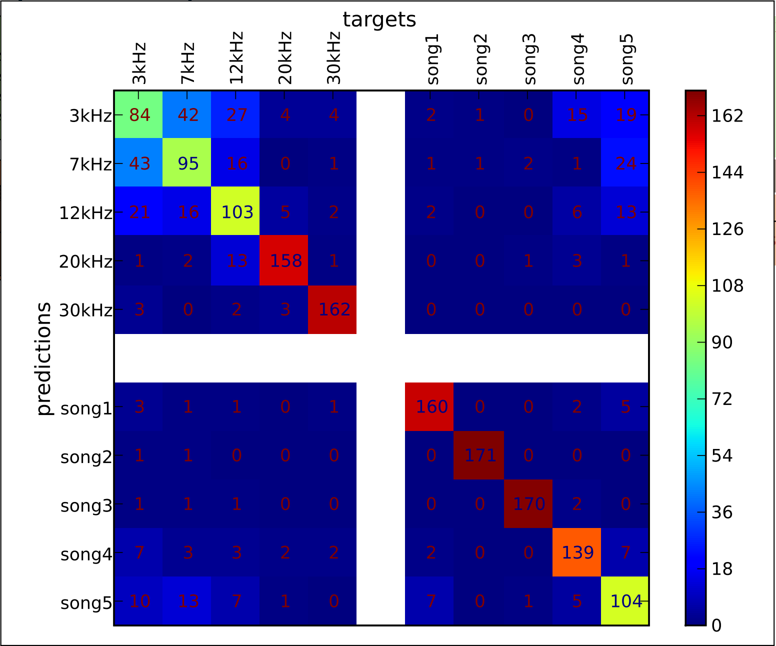
```
oner(),
'])
```

```
--------.         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

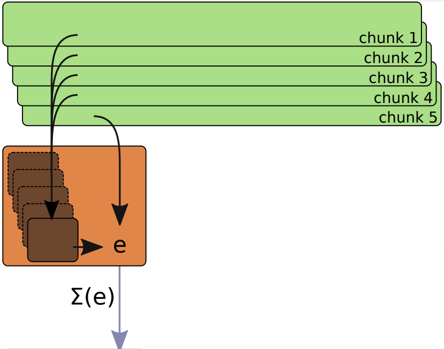




	3kHz	7kHz	12kHz	20kHz	30kHz	song1	song2	song3	song4	song5
3kHz	84	42	27	4	4	2	1	0	15	19
7kHz	43	95	16	0	1	1	1	2	1	24
12kHz	21	16	103	5	2	2	0	0	6	13
20kHz	1	2	13	158	1	0	0	1	3	1
30kHz	3	0	2	3	162	0	0	0	0	0
song1	3	1	1	0	1	160	0	0	2	5
song2	1	1	0	0	0	0	171	0	0	0
song3	1	1	1	0	0	0	0	170	2	0
song4	7	3	3	2	2	2	0	0	139	7
song5	10	13	7	1	0	7	0	1	5	104

```
xt')
z',
s,

i.gz',
i.gz'})

oner(),
']) 
```

Analysis Example: Classification



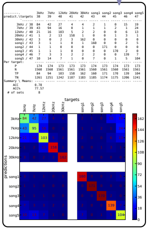
```python
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets,
                  chunks=attr.chunks,
                  mask='mask_brain.nii.gz',
                  add_fa={'vt':'vt.nii.gz'})
```

```python
clf = LinearCSVMC()
```

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

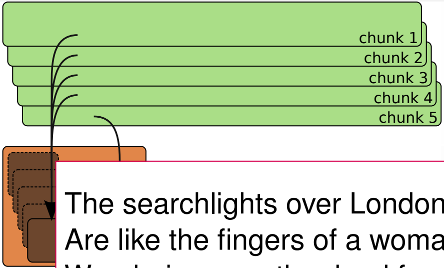
```python
errors = cv(ds)
```

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



```
---------.         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:        ----- ----- ----- ----- ----- ----- ----- ----- ----- -----
      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:   ----- ----- ----- ----- ----- ----- ----- ----- ----- -----
     ACC           0.78
     ACC%          77.57
  # of sets        8
```

Analysis Example: Searchlight

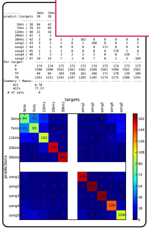


```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets,
                  chunks=attr.chunks,
                  mask='mask_brain.nii.gz',
                  add_fa={'vt':'vt.nii.gz'})
```

```
clf = LinearCSVMC()
```

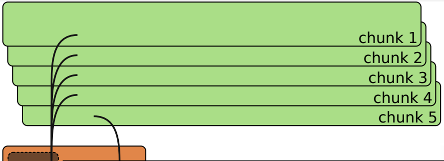
```
er(),
```

> The searchlights over London
> Are like the fingers of a woman,
> Wandering over the dead form of a lover. …
>
> – *John Rodker “The Searchlight”, 1894-1955*

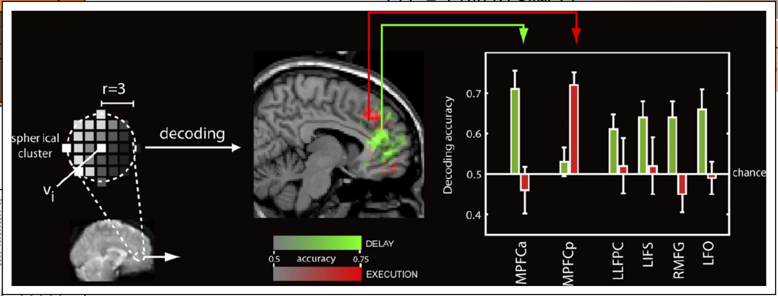


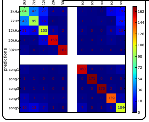
```
print cv.ca.stats
cv.ca.stats.plot()
```

Analysis Example: Searchlight



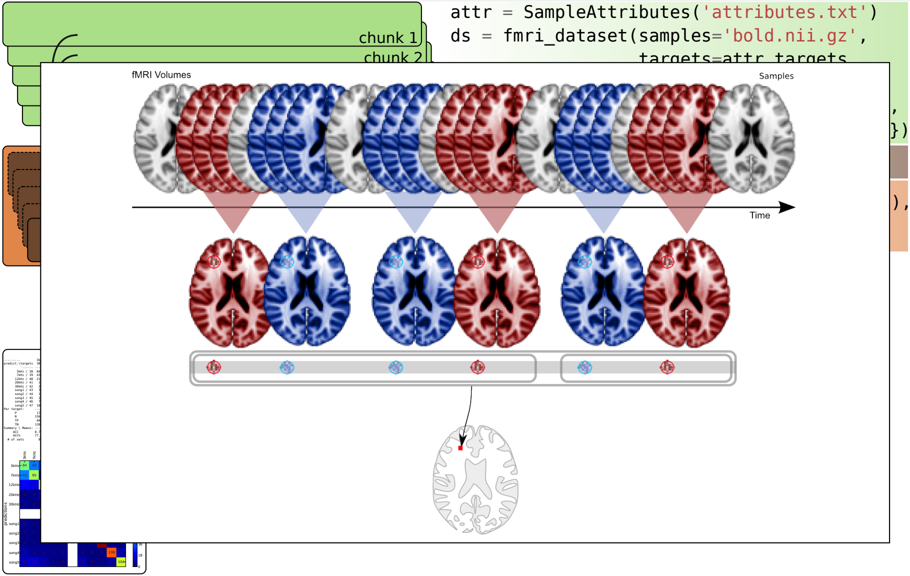
```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets,
                  chunks=attr.chunks,
                  mask='mask_brain.nii.gz',
                  add_fa={'vt':'vt.nii.gz'})
clf = LinearCSVMC()
oner(),
'])
```





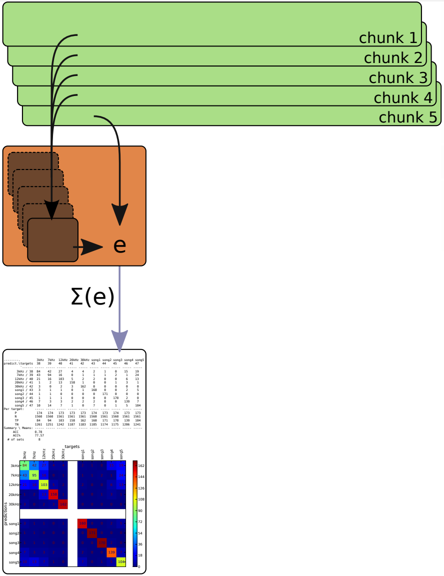
Haynes and Rees, Current Biology, 2007

Analysis Example: Searchlight



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

Analysis Example: Searchlight



```
--------.         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:       ----- ----- ----- ----- ----- ----- ----- ----- ----- -----
         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:  ----- ----- ----- ----- ----- ----- ----- ----- ----- -----
       ACC        0.78
       ACC%       77.57
   # of sets       8
```

```
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'})
```

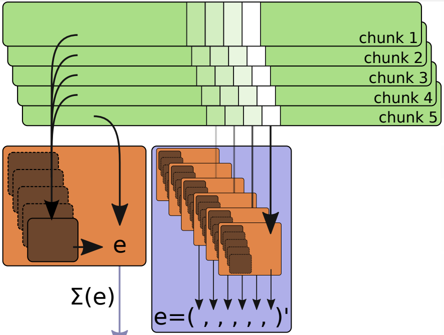
```
clf = LinearCSVMC()
```

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

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

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

Analysis Example: Searchlight

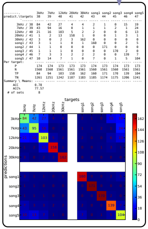


```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets,
                  chunks=attr.chunks,
                  mask='mask_brain.nii.gz',
                  add_fa={'vt':'vt.nii.gz'})
```

```
clf = LinearCSVMC()
```

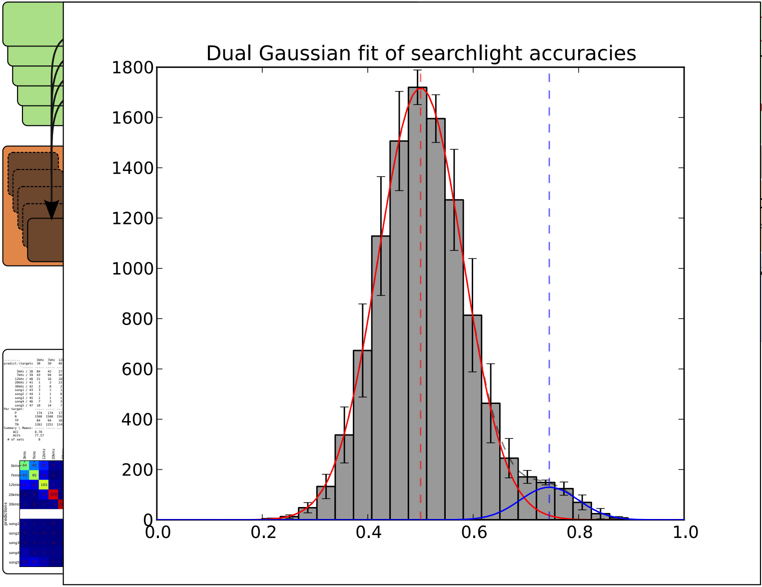
```
cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])
```

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

fwm_map = fwm(ds)
```



```
--------.        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:      ----- ----- ----- ----- ----- ----- ----- ----- ----- -----
     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: ----- ----- ----- ----- ----- ----- ----- ----- ----- -----
      ACC        0.78
      ACC%       77.57
   # of sets      8
```

Analysis Example: Searchlight (Data says a lot)

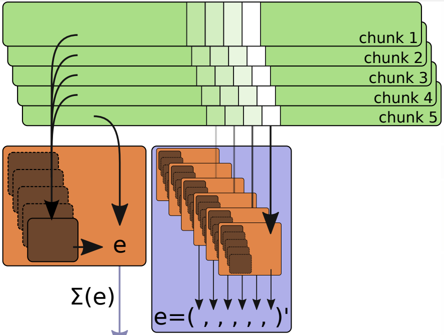


Dual Gaussian fit of searchlight accuracies

```
butes.txt')
d.nii.gz',
.targets,
chunks,
rain.nii.gz',
:'vt.nii.gz'})

dPartitioner(),
=['stats'])

adius=3)
```

PyMVPA: doc/examples/curvefitting.py

Analysis Example: Searchlight

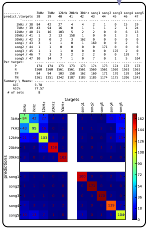


```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets,
                  chunks=attr.chunks,
                  mask='mask_brain.nii.gz',
                  add_fa={'vt':'vt.nii.gz'})
```

```
clf = LinearCSVMC()
```

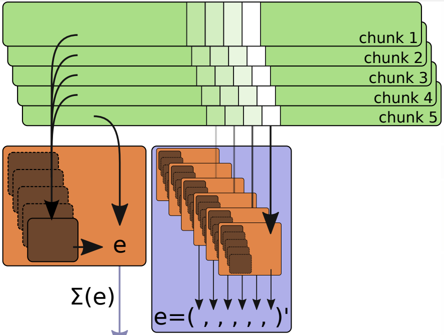
```
cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])
```

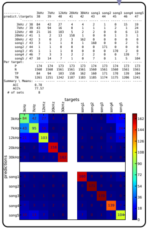
```
fwm = sphere_searchlight(cv, radius=3)

fwm_map = fwm(ds)
```



```
---------.        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:       ----- ----- ----- ----- ----- ----- ----- ----- ----- -----
         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:  ----- ----- ----- ----- ----- ----- ----- ----- ----- -----
       ACC        0.78
       ACC%       77.57
   # of sets       8
```

Analysis Example: Searchlight





```
--------.         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:      ----- ----- ----- ----- ----- ----- ----- ----- ----- -----
      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: ----- ----- ----- ----- ----- ----- ----- ----- ----- -----
     ACC         0.78
     ACC%        77.57
  # of sets        8
```

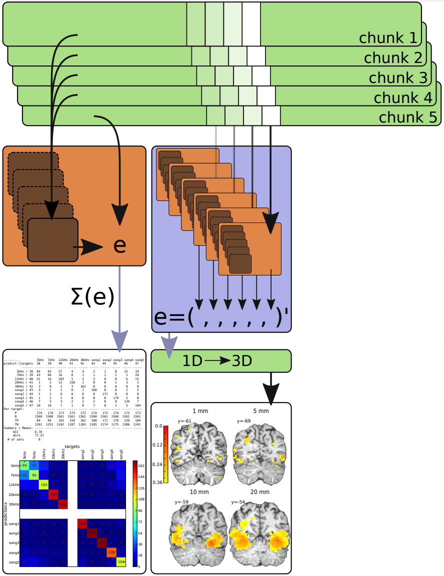
```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets,
                  chunks=attr.chunks,
                  mask='mask_brain.nii.gz',
                  add_fa={'vt':'vt.nii.gz'})
```

```
clf = LinearCSVMC()
```

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

```
fwm = sphere_gnbsearchlight(
    GNB(), NFoldPartitioner(), 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'})
```

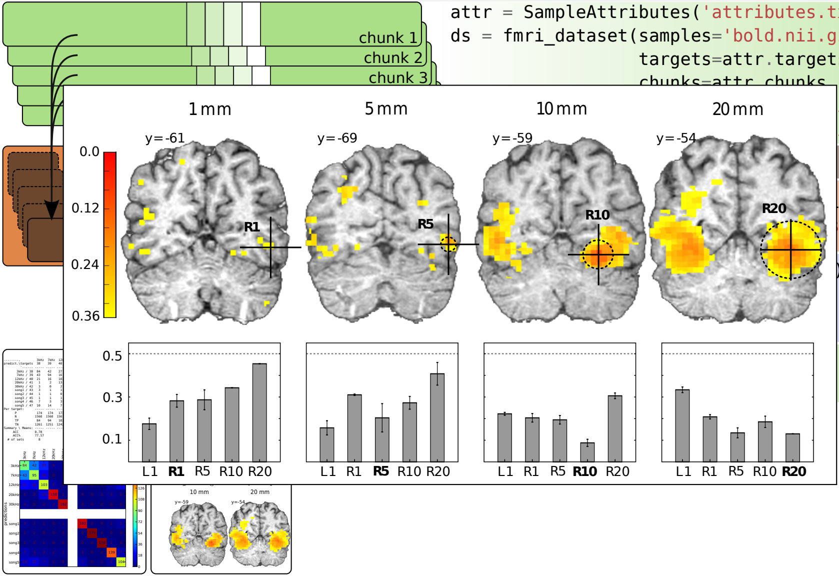
```
clf = LinearCSVMC()
```

```
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')
```

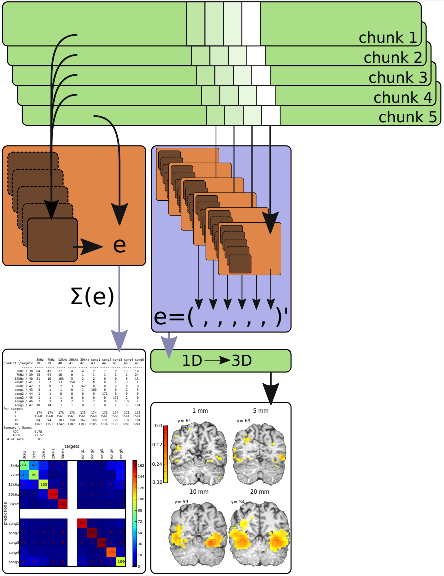
Complete Analysis Example: Searchlight



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

Hanke&Halchenko et al., Neuroinformatics, 2009

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'})
```

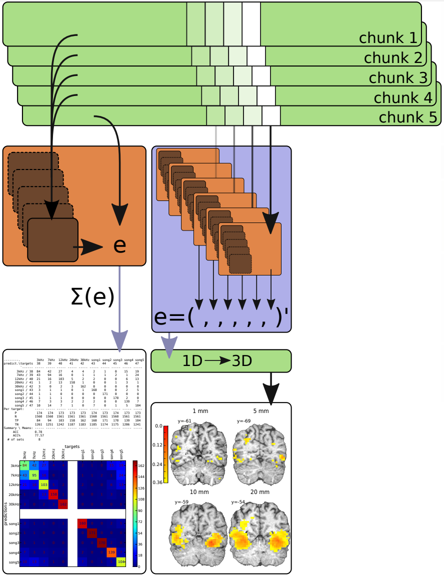
```
clf = LinearCSVMC()
```

```
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')
```

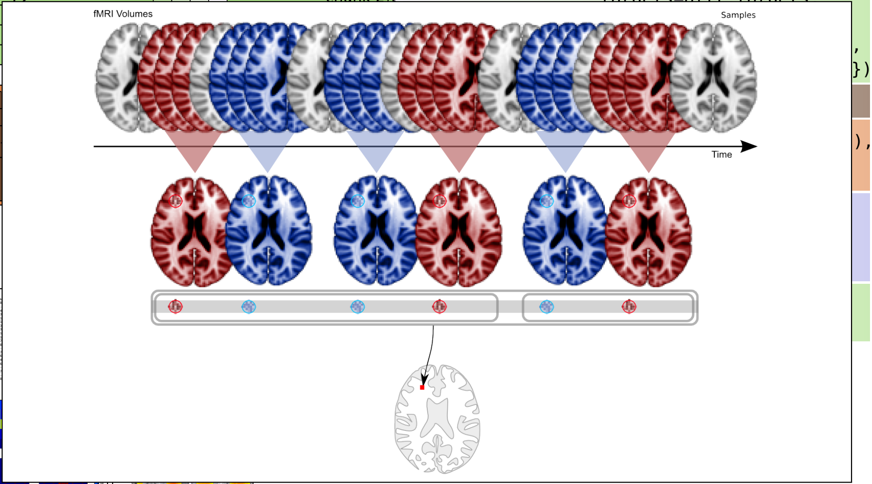
Complete Analysis Example: Searchlight



```
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)
```

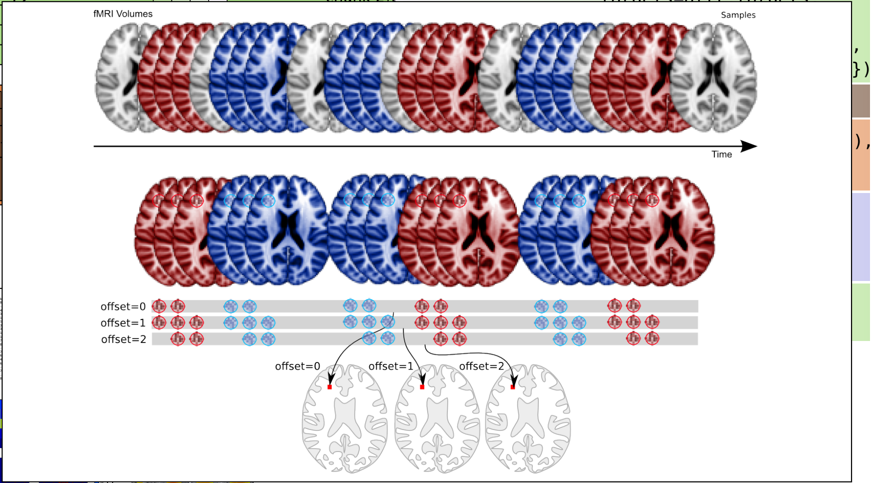
Complete Analysis Example: Searchlight

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

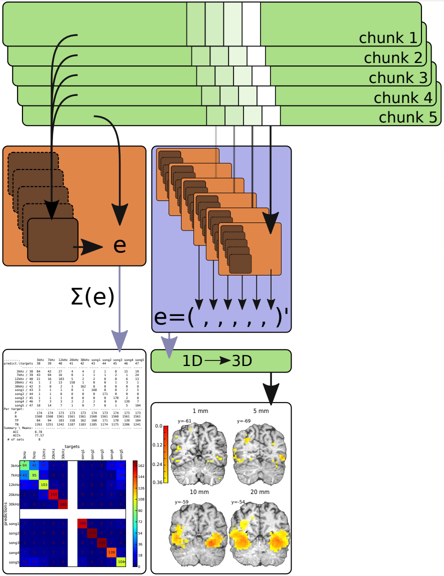


Analysis Example: Spatio-temporal Searchlight

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

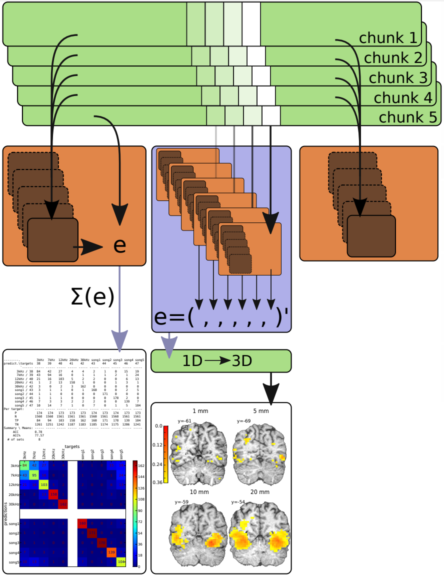


Analysis Example: Searchlight



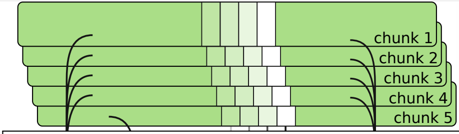
```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets,
                  chunks=attr.chunks,
                  mask='mask_brain.nii.gz',
                  add_fa={'vt':'vt.nii.gz'})

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')

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

Analysis Example: Sensitivities

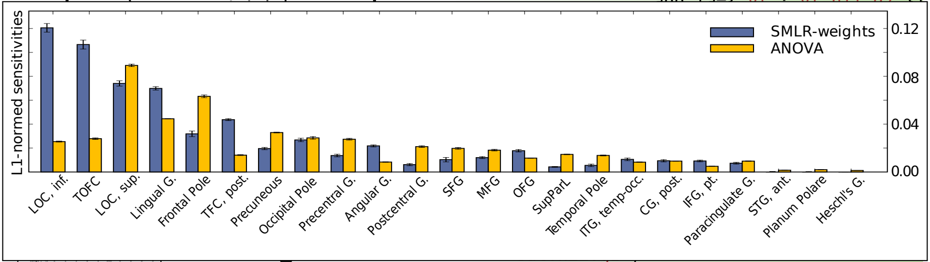


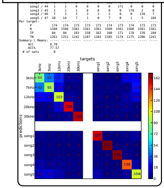
```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets,
                  chunks=attr.chunks,
                  mask='mask_brain.nii.gz',
                  add_fa={'vt':'vt.nii.gz'})

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)
```

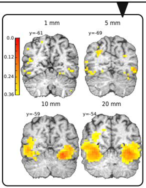
Analysis Example: Sensitivities



```
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'})
```



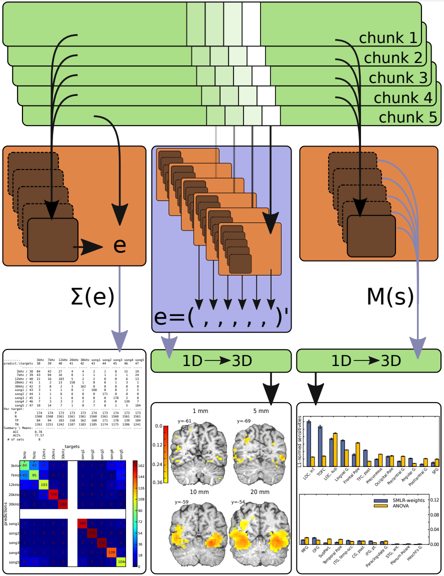




```
h5save('out.hdf5', fwm_map)
```

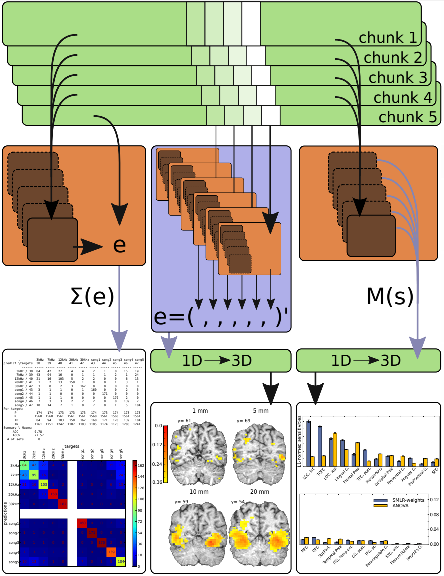
Hanke&Halchenko et al., Frontiers in Neuroinformatics, 2009

Analysis Example: Sensitivities



```
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



```
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()
```

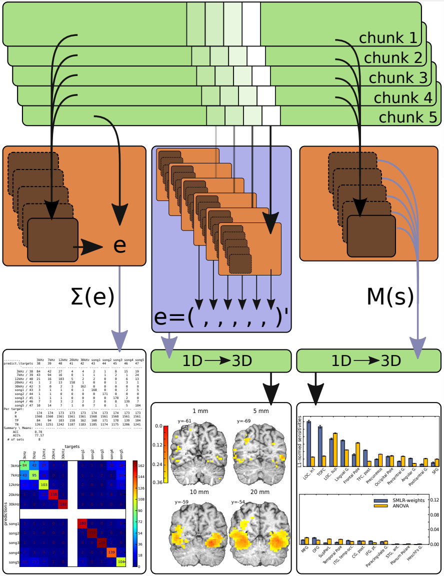
```
cv = CrossValidation(clf, NFoldPartitioner(),
                     enable_ca=['stats'])
```

```
fwm = clf.get_sensitivity_analyzer()

fwm_map = fwm(ds)
```

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

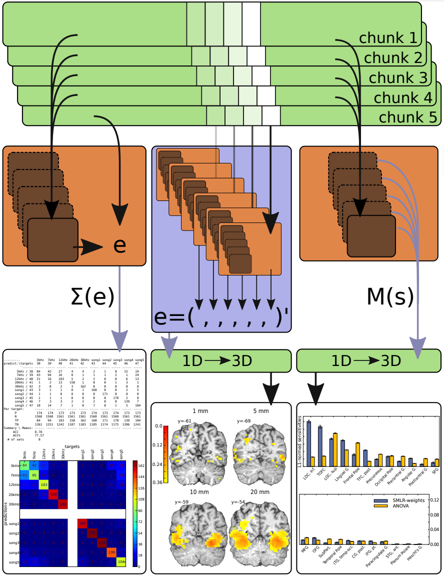
```
h5save('out.hdf5', fwm_map)
```

Analysis Example: GPR-Sensitivities



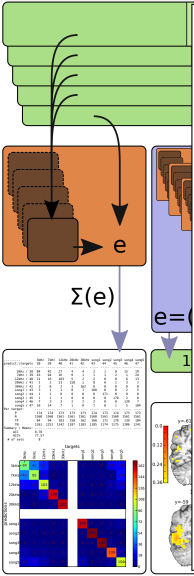
```
attr = SampleAttributes('attributes.txt')
ds = fmri_dataset(samples='bold.nii.gz',
                  targets=attr.targets,
                  chunks=attr.chunks,
                  mask='mask_brain.nii.gz',
                  add_fa={'vt':'vt.nii.gz'})

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



```
ributes.txt')
bold.nii.gz',
ttr.targets,
tr.chunks,
k_brain.nii.gz',
vt':'vt.nii.gz'})
oldPartitioner(),
ca=['stats'])
alyzer()
lename(
```

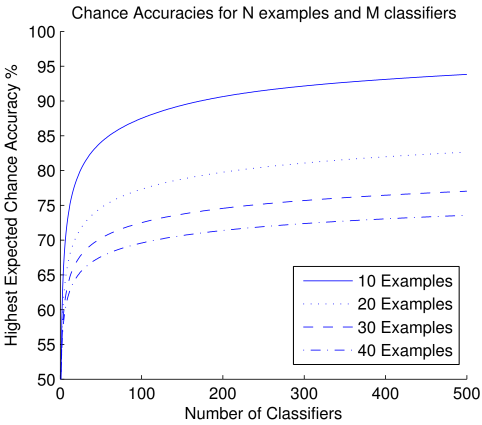
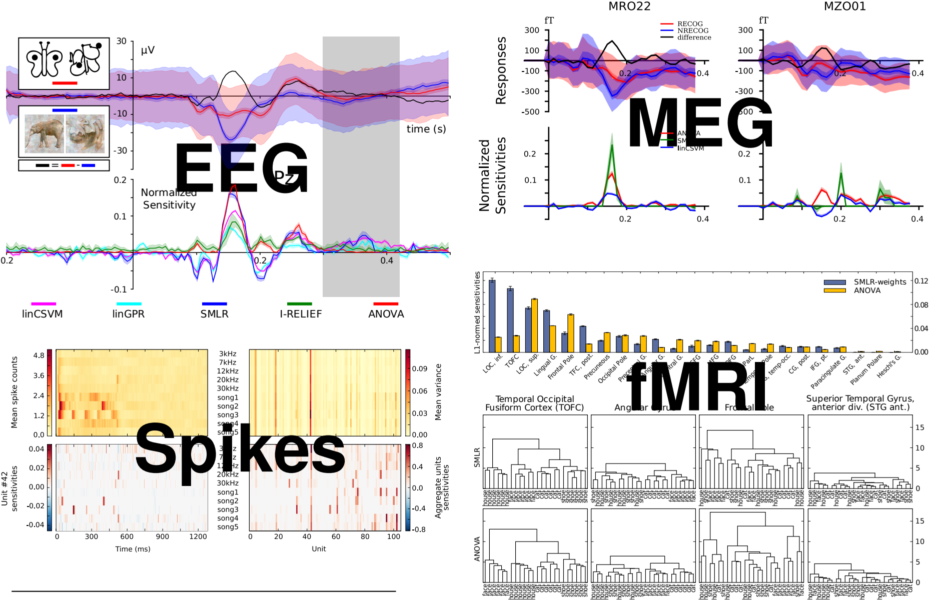


Figure 1. The highest expected chance accuracy as a function of the number of examples and classifiers. Each line represents a different number of examples. The x-axis is the number of classifiers and the y-axis is the accuracy.

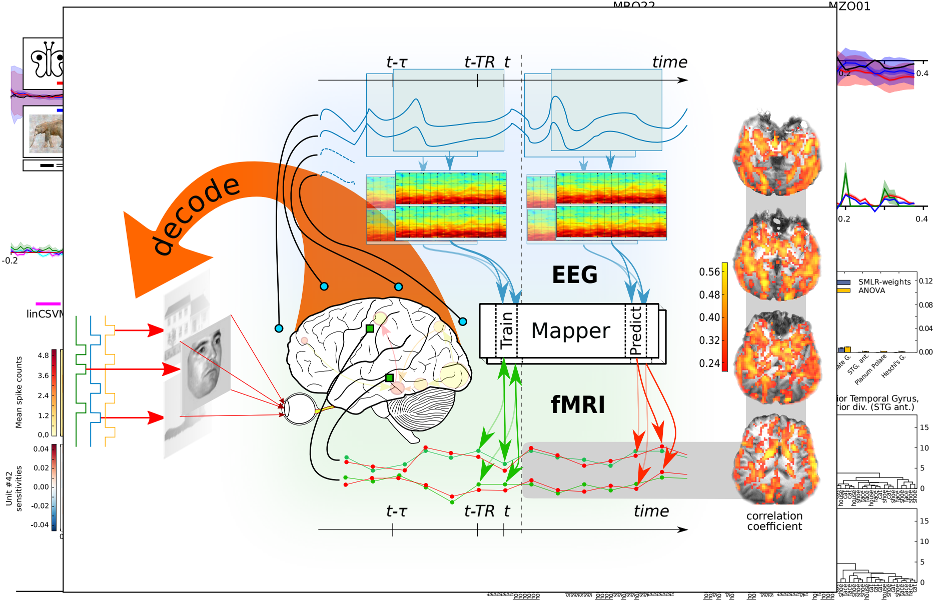
Palatucci&Carlson, On the Chance Accuracies of Large Collections of Classifiers, 2008

And With That We Could Do:



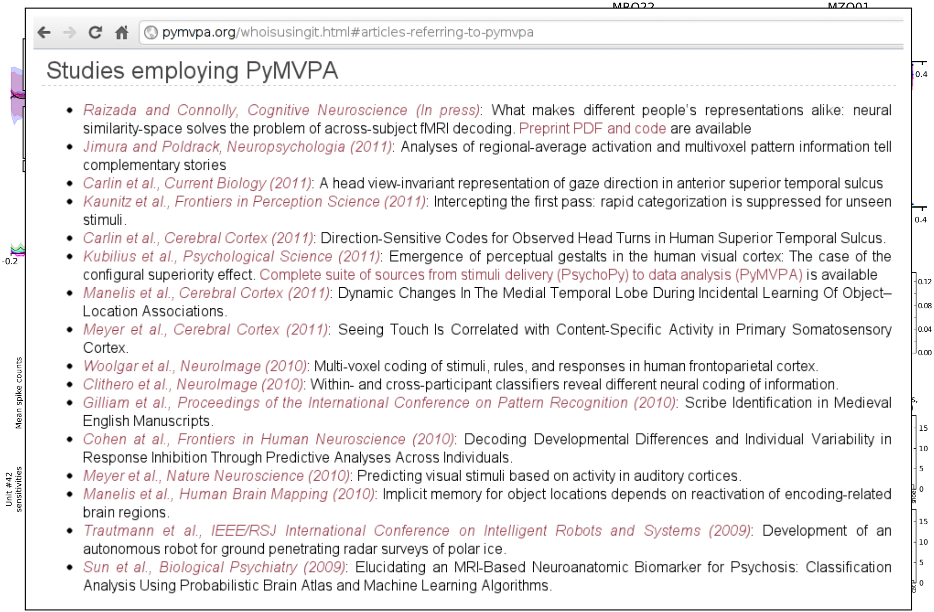
Hanke&Halchenko et al., Frontiers in Neuroinformatics, 2009

And With That We Could Do: **EEG⇒fMRI**



Halchenko, 2009

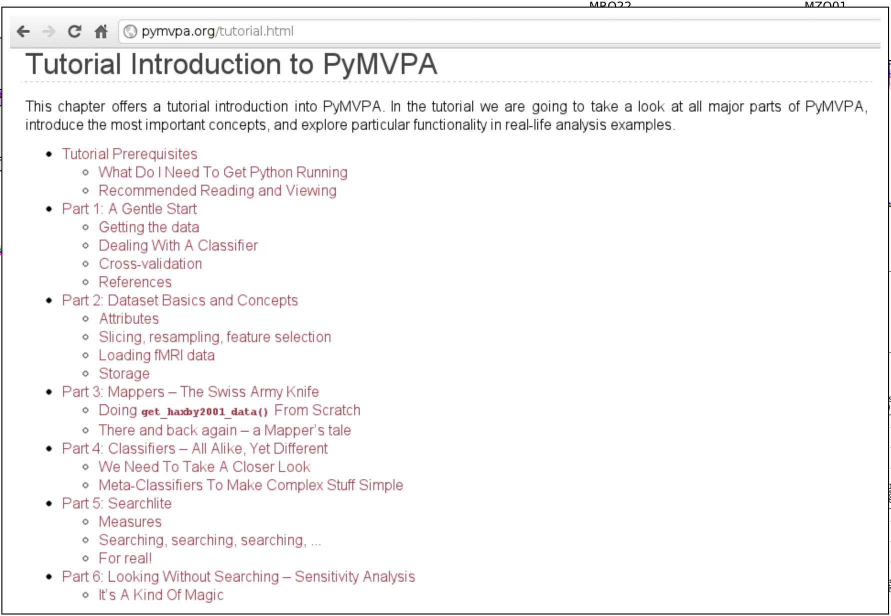
And With That **Others** Could Do:



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 gestalts 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.
- *Woolgar et al., NeuroImage (2010)*: Multi-voxel coding of stimuli, rules, and responses in human frontoparietal cortex.
- *Clithero et al., NeuroImage (2010)*: Within- and cross-participant classifiers reveal different neural coding of information.
- *Gilliam et al., Proceedings of the International Conference on Pattern Recognition (2010)*: Scribe Identification in Medieval English Manuscripts.
- *Cohen at 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.
- *Trautmann et al., IEEE/RSJ International Conference on Intelligent Robots and Systems (2009)*: Development of an autonomous robot for ground penetrating radar surveys of polar ice.
- *Sun et al., Biological Psychiatry (2009)*: Elucidating an MRI-Based Neuroanatomic Biomarker for Psychosis: Classification Analysis Using Probabilistic Brain Atlas and Machine Learning Algorithms.

And With That **You** Could Do: Extended Tutorial



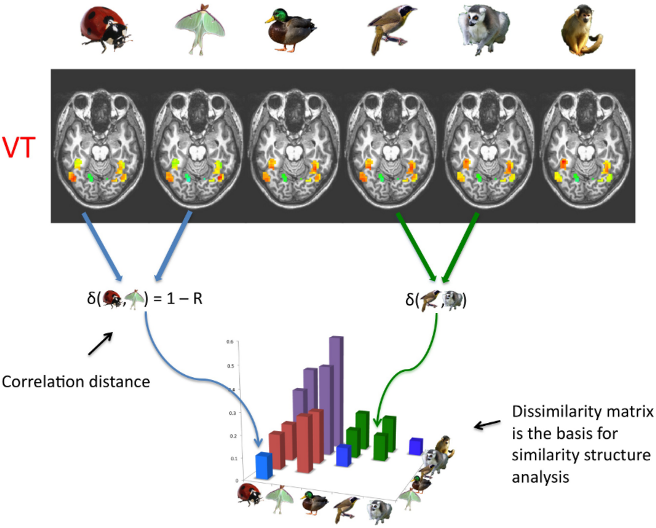
Tutorial Introduction to PyMVPA

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

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

New perspectives from HaxbyLab

Similarity Analyses



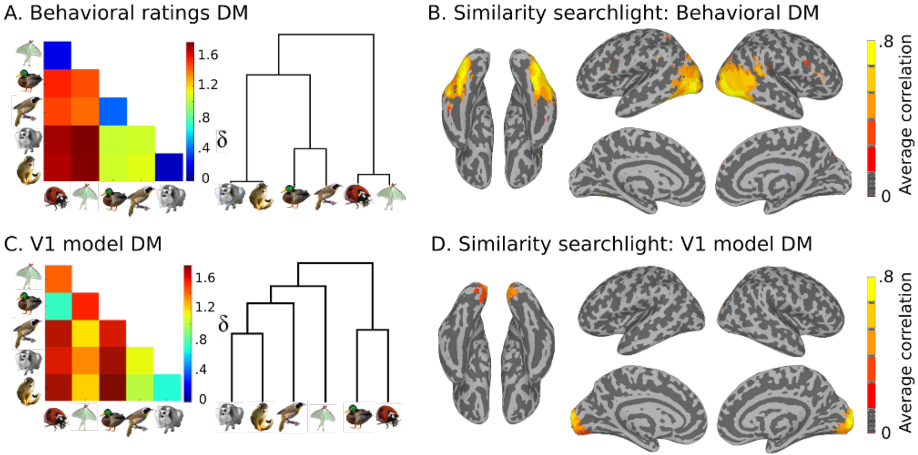
Correlation distance

Dissimilarity matrix is the basis for similarity structure analysis

---

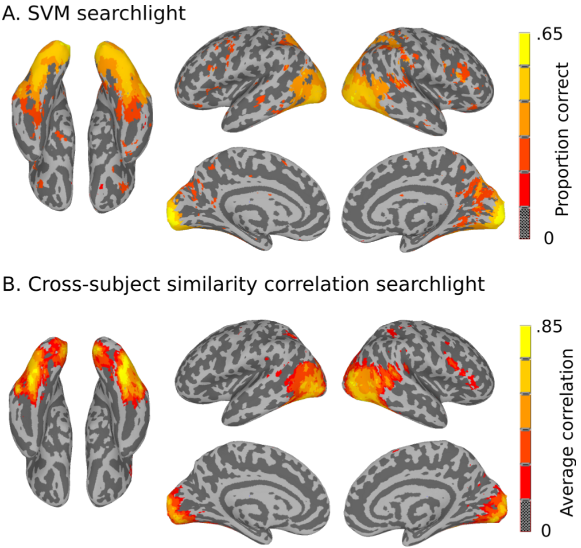
Andrew Connolly: `http://haxbylab.dartmouth.edu/ppl/andy.html`

Similarity Analyses: V1 vs behavioral models



Connolly, A. C. et al, Representation of biological classes in the human brain. Journal of Neuroscience (in press).

Similarity Analyses: Cross-subject agreement

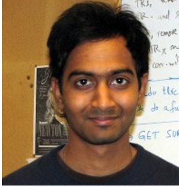


Connolly, A. C. et al, Representation of biological classes in the human brain. Journal of Neuroscience (in press).

Similarity Analyses . . .

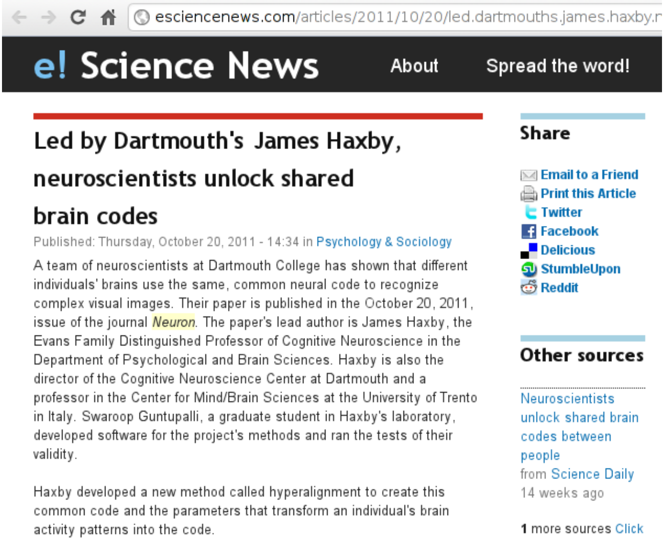
- provide richer picture than mean generalization error maps
- investigate the transformation of the representations through processing streams
- unravel functional regions with information represented according to theoretical models of perception or behavior observations
- relax requirement for precise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

Hyperalignment



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

Hyperalignment



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.

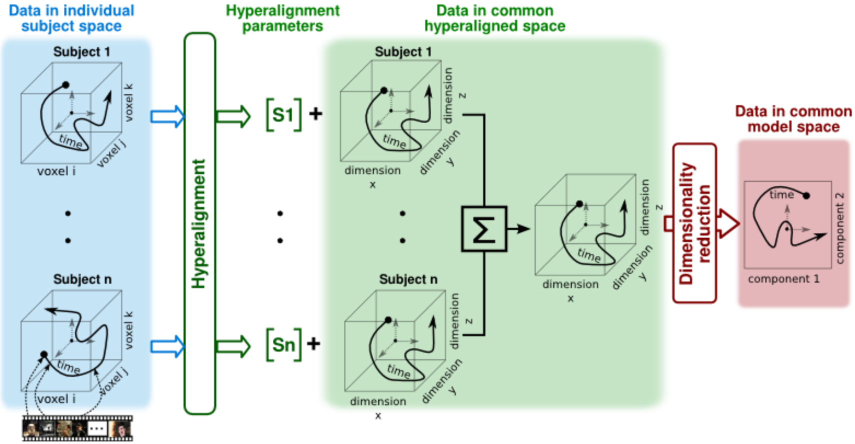
Haxby developed a new method called hyperalignment to create this common code and the parameters that transform an individual's brain activity patterns into the code.

Other sources

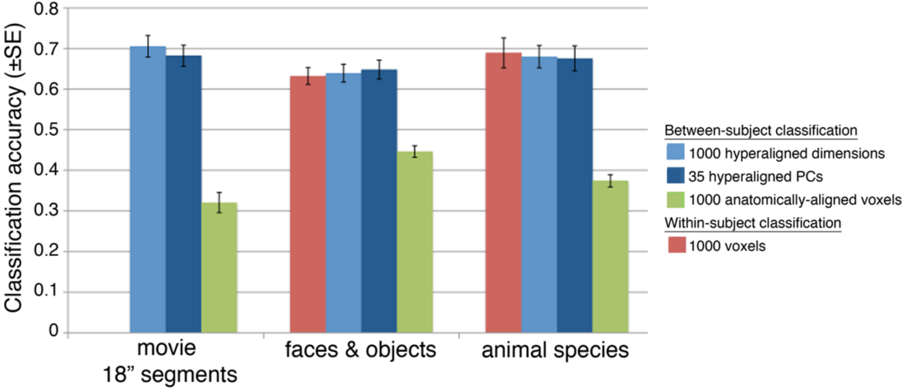
Neuroscientists unlock shared brain codes between people
from Science Daily
14 weeks ago

1 more sources Click

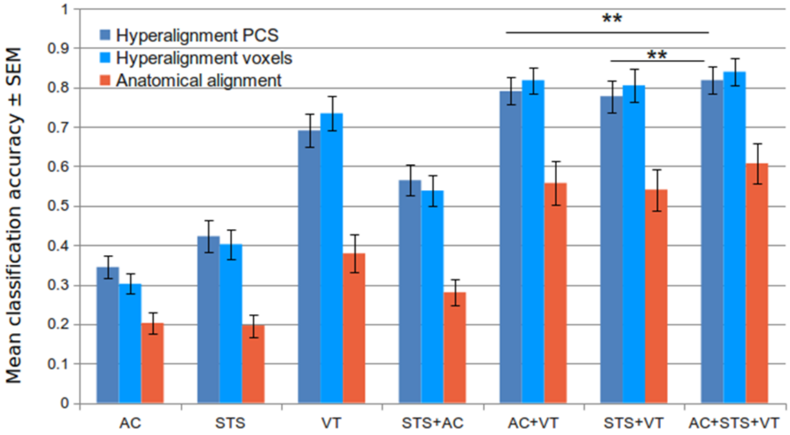
Hyperalignment: Deriving a common space



Hyperalignment: Classification (VT cortex)



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

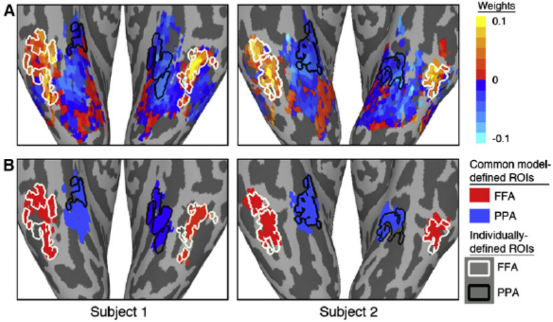


Figure 6. Contrast-Defined Category-Selective Profiles in the Common Model Space Projected into the Native Voxel Spaces of Two Subjects

(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.

Haxby&Guntupalli et al. (2011). A Common, High-Dimensional Model of the Representational Space in Human Ventral Temporal Cortex. Neuron

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&Guntupalli et al. (2011). A Common, High-Dimensional Model of the Representational Space in Human Ventral Temporal Cortex. Neuron

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

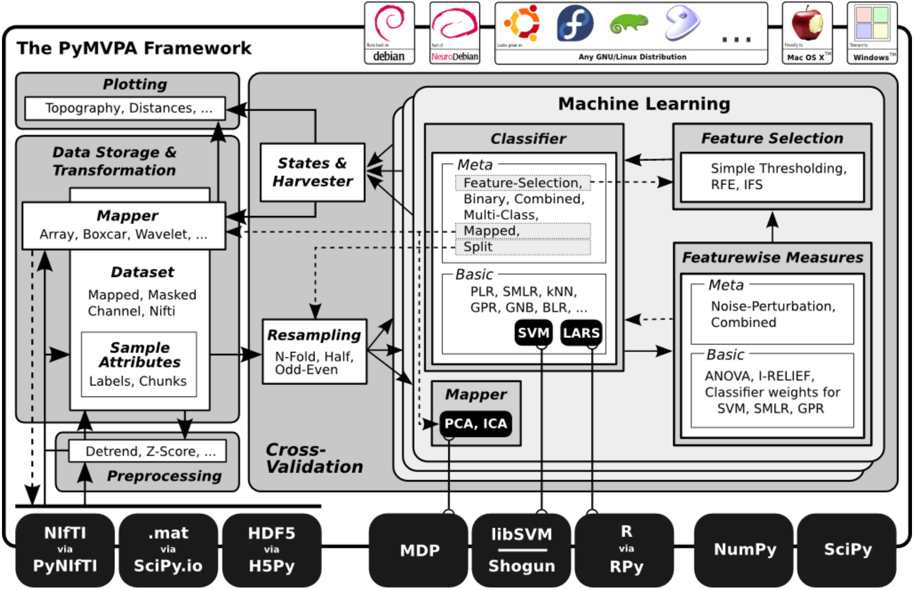
Hyperalignment. . .

- 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&Guntupalli et al. (2011). A Common, High-Dimensional Model of the Representational Space in Human Ventral Temporal Cortex. Neuron

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

debian

NeuroDebian

Any GNU/Linux Distribution

Mac OS X™

Windows™

Plotting

Topography, Distances, ...

Data Storage & Transformation

Mapper

Array, Boxcar, Wavelet, ...

Dataset

Mapped, Masked
Channel, Nifti

Sample Attributes

Labels, Chunks

Detrend, Z-Score, ...

Preprocessing

States & Harvester

Resampling

N-Fold, Half, Odd-Even

Cross-Validation

Machine Learning

Classifier

Meta

Feature-Selection,
Binary, Combined,
Multi-Class,
Mapped,
Split

Basic

PLR, SMLR, kNN,
GPR, GNB, BLR, ...

SVM **LARS**

Mapper

PCA, ICA

Feature Selection

Simple Thresholding,
RFE, IFS

Featurewise Measures

Meta

Noise-Perturbation,
Combined

Basic

ANOVA, I-RELIEF,
Classifier weights for
SVM, SMLR, GPR

NIfTI via **PyNIfTI**

.mat via **SciPy.io**

HDF5 via **H5Py**

MDP

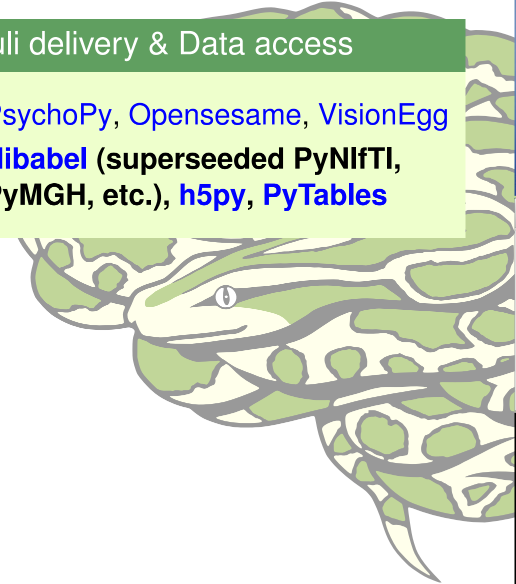
libSVM
Shogun

R via **RPy**

NumPy

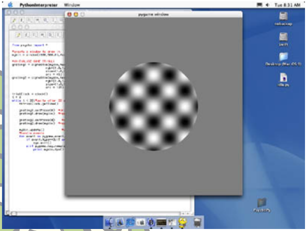
SciPy

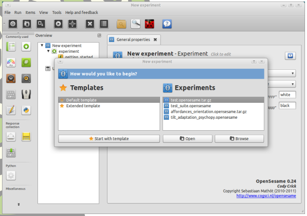
Python in Neuroimaging

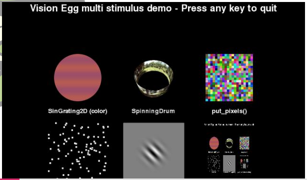


Stimuli delivery & Data access

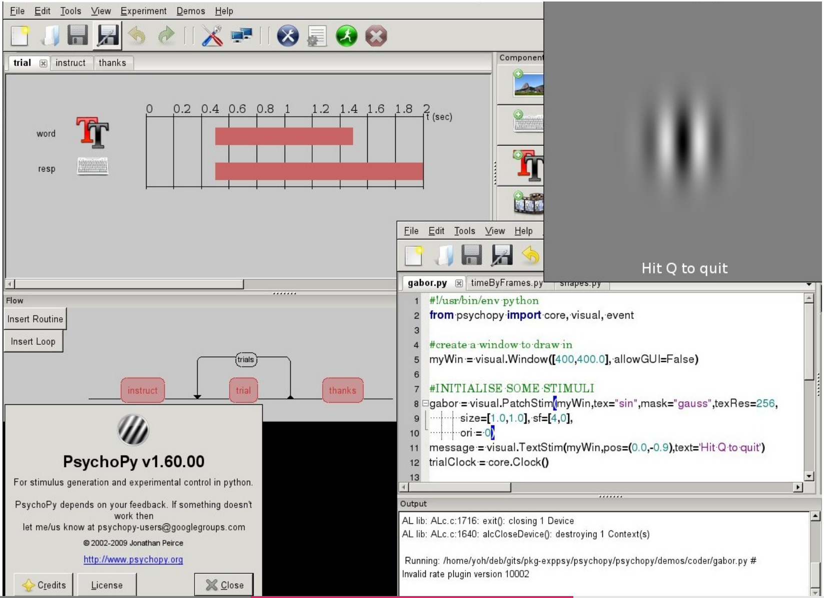
- PsychoPy, Opensesame, VisionEgg
- **Nibabel (superseeded PyNIfTI, PyMGH, etc.), h5py, PyTables**







Python in Neuroimaging: PsychoPy



```python
#!/usr/bin/env python
from psychopy import core, visual, event

#create a window to draw in
myWin = visual.Window([400,400.0], allowGUI=False)

#INITIALISE SOME STIMULI
gabor = visual.PatchStim(myWin,tex="sin",mask="gauss",texRes=256,
        size=[1.0,1.0], sf=[4,0],
        ori = 0)
message = visual.TextStim(myWin,pos=(0.0,-0.9),text='Hit Q to quit')
trialClock = core.Clock()

```

```
AL lib: ALc.c:1716: exit(): closing 1 Device
AL lib: ALc.c:1640: alcCloseDevice(): destroying 1 Context(s)

 Running: /home/yoh/deb/gits/pkg-exppsy/psychopy/psychopy/demos/coder/gabor.py #
Invalid rate plugin version 10002
```

PsychoPy v1.60.00

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

© 2002-2009 Jonathan Peirce

http://www.psychopy.org

Python in Neuroimaging

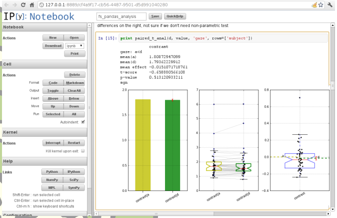
Stimuli delivery & Data access

- 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

http://www.ipython.org/

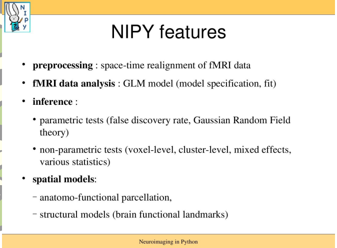


Shift-Enter : run selected cell
Ctrl-Enter : run selected cell in-place
Ctrl-m h : show keyboard shortcuts

differences on the right, not sure if we don't need non-parametric test

```
In [15]: print paired_t_anal(d, value, 'gaze', rows=['subject'])
                   contrast
gaze: a<d
mean(a)            1.80872947099
mean(d)            1.79362229912
mean effect        -0.0151071718761
t-score            -0.658880566108
p-value            0.513120933211
sgn
```

http://nipy.org



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)

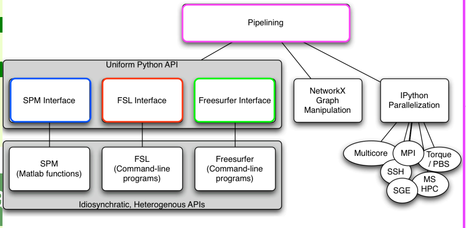
Neuroimaging in Python

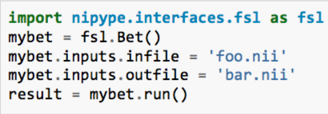
Python in Neuroimaging: NiPype

Sti

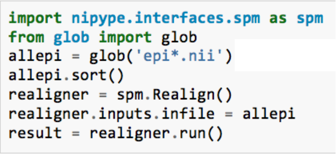
Da

Neuroimaging in Python
Pipelines and Interfaces

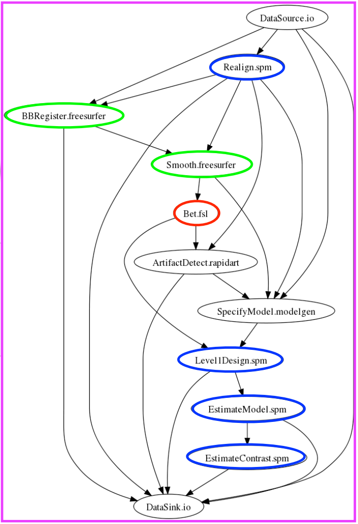




```
import nipype.interfaces.fsl as fsl
mybet = fsl.Bet()
mybet.inputs.infile = 'foo.nii'
mybet.inputs.outfile = 'bar.nii'
result = mybet.run()
```



```
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()
```

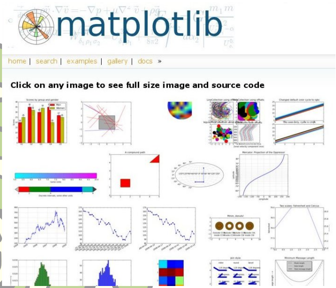


Python in Neuroimaging

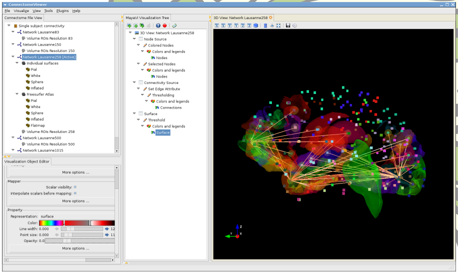
Data visualization

- **matplotlib**, Mayavi2
- ConnectomeViewer
- PySurfer, Anatomist

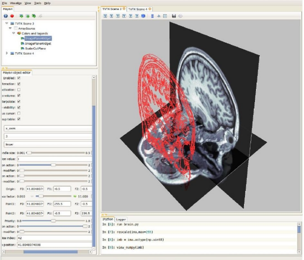
http://matplotlib.sourceforge.net/gallery.html



http://www.connectomeviewer.org



http://code.enthought.com/projects/mayavi/



```
In [6]: run brain.py
In [7]: rescale(ima,max=255)
In [8]: imb = ima.astype(np.uint8)
In [9]: view_numpy(imb)
```

Python in NeuroImaging

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

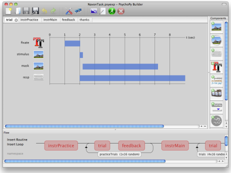
Stimuli Delivery

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)

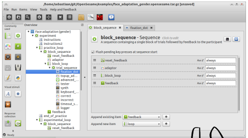


OpenSesame

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.

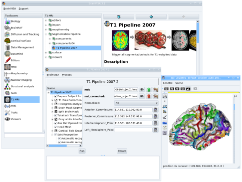
Analysis

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 ang gyri morphometry, diffusion imaging and fibers tracking, surfacic and structural analysis, 3D histology…
- Links with other software like SPM, FSL, FreeSurfer, or CIVET



D. Geffroy, D. Rivière, I. Denghien, N. Souedet, S. Laguitton, and Y. Cointepas. BrainVISA: a complete software platform for neuroimaging. In Python in Neuroscience workshop, Paris, Aug. 2011.

DiPy

http://nipy.org/dipy

Dipy
Diffusion Imaging in Python

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

---

http://neuro.debian.net/_files/nipy-handout.pdf

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

Vision

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

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

Install simple editor

```
apt-get install gedit
```

Install complex MRI analysis package

```
apt-get install fsl
```

Install software collection for psycho-physics

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

Keep the whole system up-to-date

```
apt-get upgrade
```

After X years and the contributions of many people:

afni amide ants biosig brian caret cfflib condor connectomeviewer coop-computing dcmtp debruijn dicomnifti dipy freenect fsl fslview gdf gifti haxby2001 itksnap jist joblib kbibtex klustakwik libsvm lipsia matlab-support mdp medcon minc mipav mitools mlpy mni-autoreg-model mni-colin mni-icbm152-nlin mpi4py mriconvert mricron mrtrix nibabel nifti nipy nitime odin openelectrophy openmeeg openopt openpyxl opensesame psignifit psychofun psychopy psychtoolbox pydicom pyepl pymvpa pynn pyoptical pyxid r-cran-glmnet rorden-mri-tutorial scikits-learn scikits-statsmodels shogun sigviewer slicer spm statsmodels stimfit svgtune sympy via volpack voxbo xmedcon

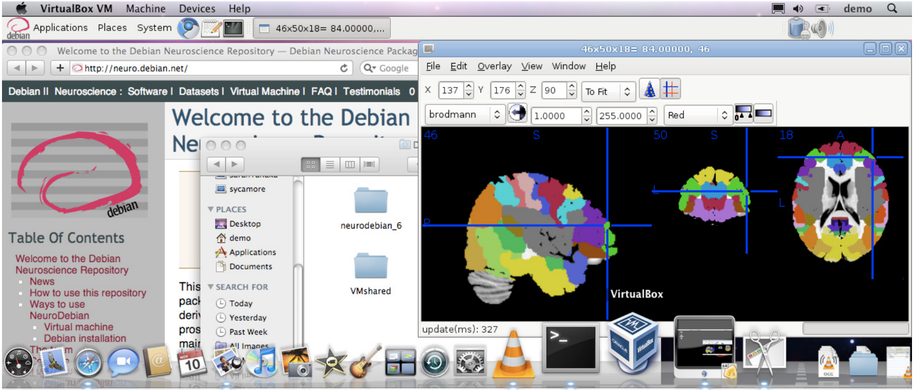
NeuroDebian focus

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`

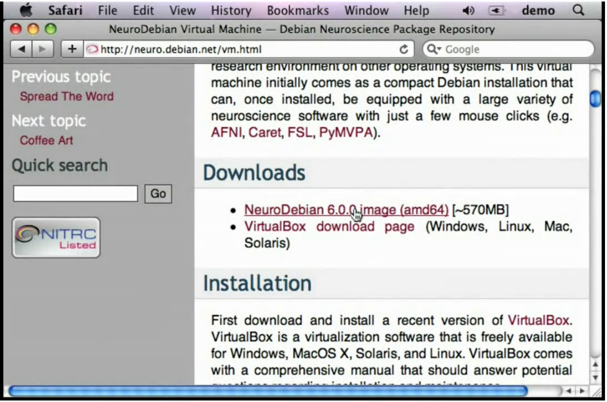
NeuroDebian: The virtual machine

- 32/64bit, multi-core
- Most convenient solution for Mac OS X, Windows
- Base image with setup wizzard, 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).

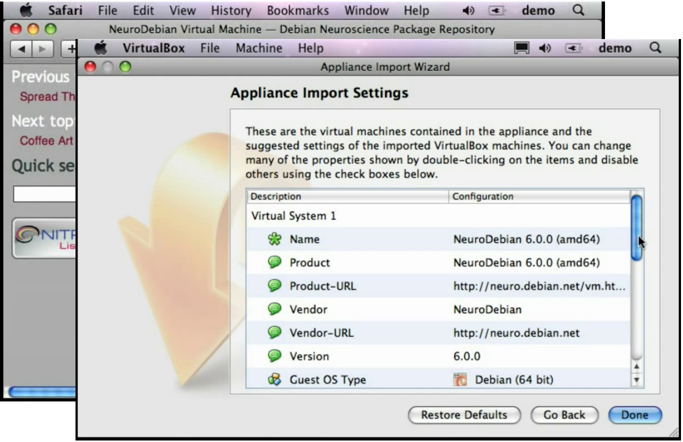
Downloads

- 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

Walkthrough: NeuroDebian on Mac OS X

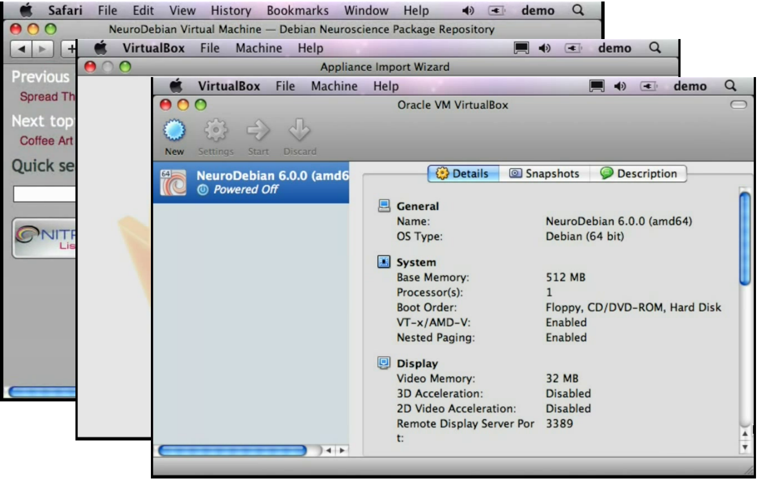


Appliance Import Settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

Description	Configuration
Virtual System 1	
Name	NeuroDebian 6.0.0 (amd64)
Product	NeuroDebian 6.0.0 (amd64)
Product-URL	http://neuro.debian.net/vm.ht...
Vendor	NeuroDebian
Vendor-URL	http://neuro.debian.net
Version	6.0.0
Guest OS Type	Debian (64 bit)

Walkthrough: NeuroDebian on Mac OS X

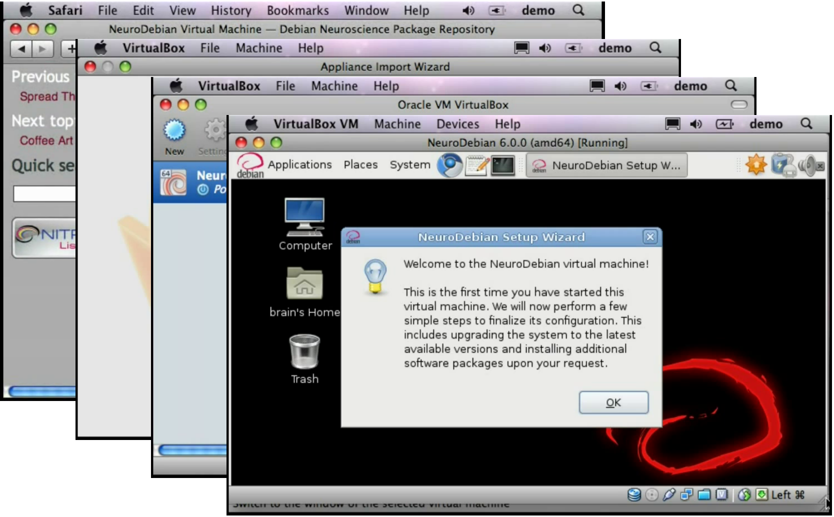


General
Name: NeuroDebian 6.0.0 (amd64)
OS Type: Debian (64 bit)

System
Base Memory: 512 MB
Processor(s): 1
Boot Order: Floppy, CD/DVD-ROM, Hard Disk
VT-x/AMD-V: Enabled
Nested Paging: Enabled

Display
Video Memory: 32 MB
3D Acceleration: Disabled
2D Video Acceleration: Disabled
Remote Display Server Port: 3389

Walkthrough: NeuroDebian on Mac OS X

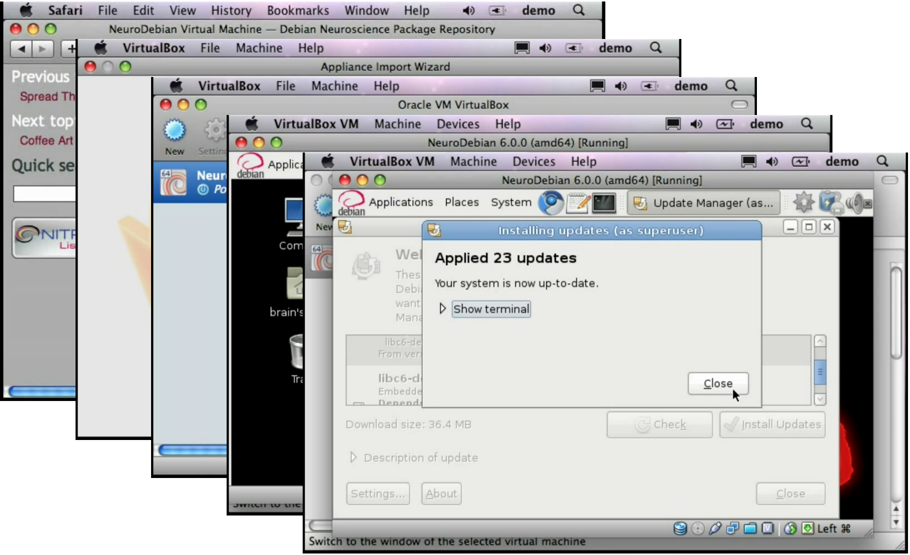


Welcome to the NeuroDebian virtual machine!

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.

Switch to the window of the selected virtual machine

Walkthrough: NeuroDebian on Mac OS X



Installing updates (as superuser)

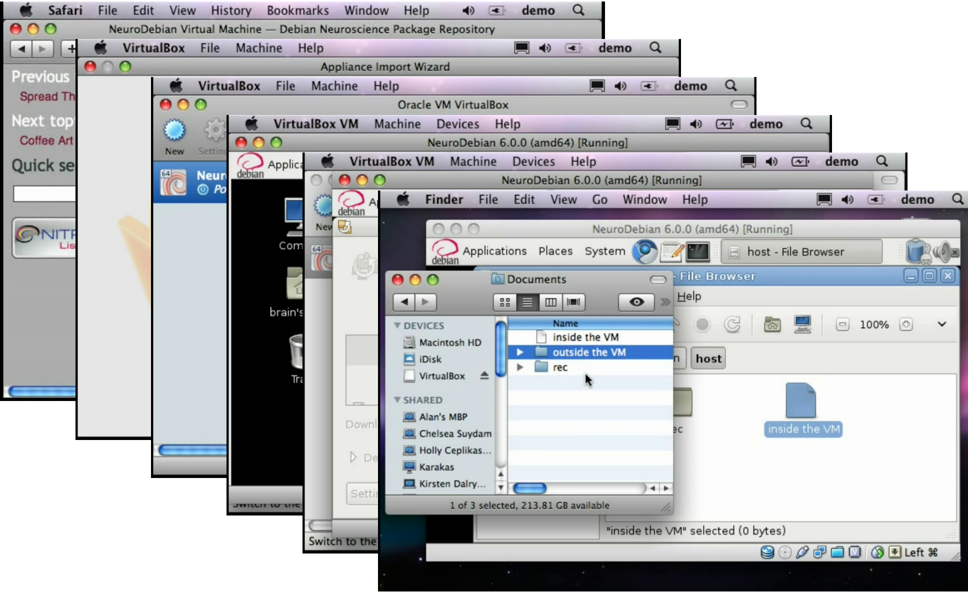
Applied 23 updates

Your system is now up-to-date.

Download size: 36.4 MB

Switch to the window of the selected virtual machine

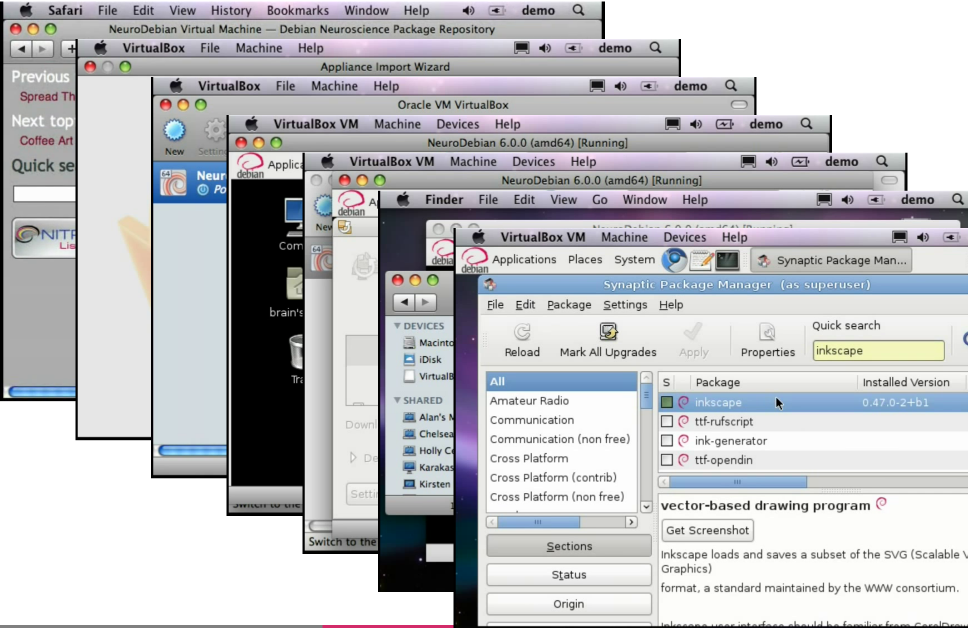
Walkthrough: NeuroDebian on Mac OS X



1 of 3 selected, 213.81 GB available

"inside the VM" selected (0 bytes)

Walkthrough: NeuroDebian on Mac OS X



S	Package	Installed Version
☑	inkscape	0.47.0-2+b1
☐	ttf-rufscript	
☐	ink-generator	
☐	ttf-opendin	

vector-based drawing program

Inkscape loads and saves a subset of the SVG (Scalable V Graphics)

format, a standard maintained by the WWW consortium.

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

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

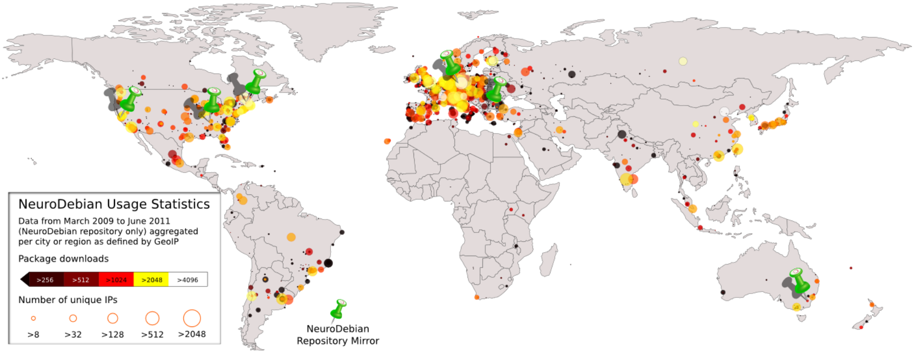
Developers

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

Maintainers

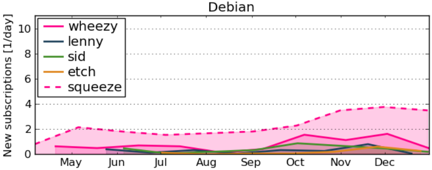
- Uniform environment for cross-project improvements and QA

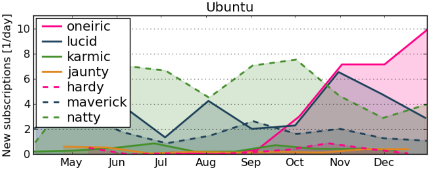
NeuroDebian popularity statistics



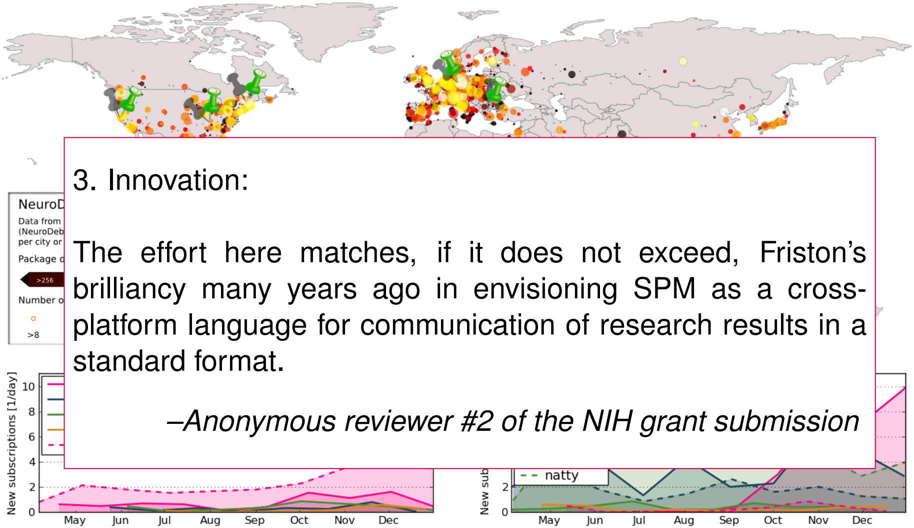
NeuroDebian Usage Statistics

Data from March 2009 to June 2011 (NeuroDebian repository only) aggregated per city or region as defined by GeoIP





NeuroDebian popularity statistics

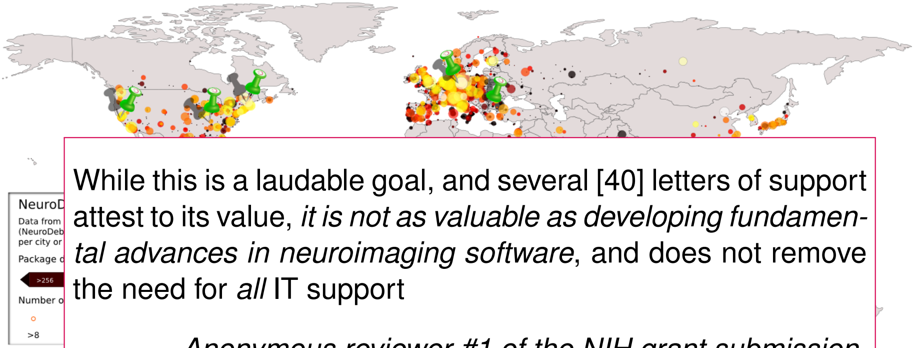


3. Innovation:

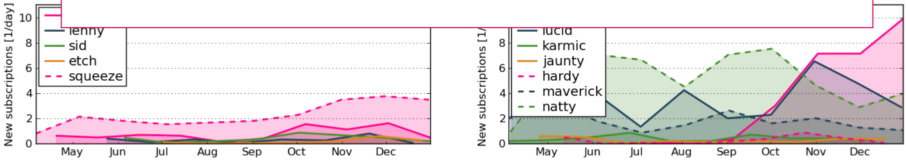
The effort here matches, if it does not exceed, Friston's brilliancy many years ago in envisioning SPM as a cross-platform language for communication of research results in a standard format.

–Anonymous reviewer #2 of the NIH grant submission

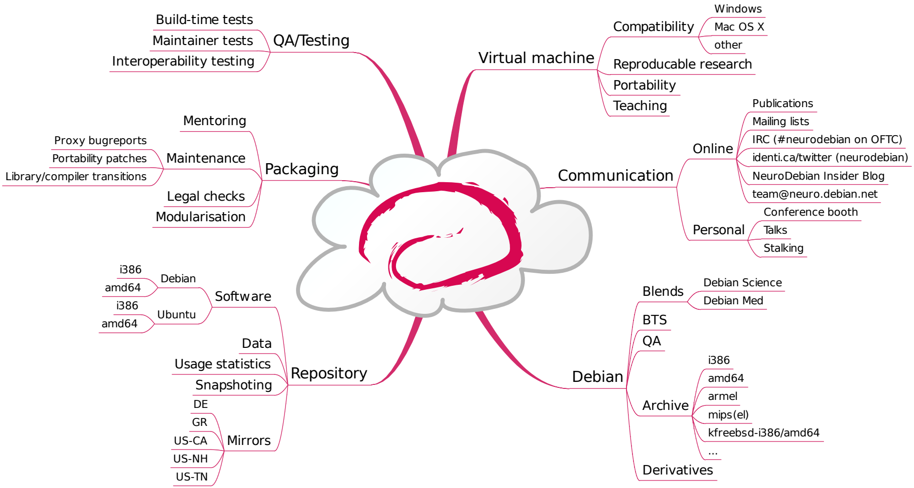
NeuroDebian popularity statistics



> 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
  - Maintainer tests
  - Interoperability testing
- Packaging
  - Mentoring
  - Maintenance
    - Proxy bugreports
    - Portability patches
    - Library/compiler transitions
  - Legal checks
  - Modularisation
- Repository
  - Software
    - Debian
      - i386
      - amd64
    - Ubuntu
      - i386
      - amd64
  - Data
  - Usage statistics
  - Snapshoting
  - Mirrors
    - DE
    - GR
    - US-CA
    - US-NH
    - US-TN
- Virtual machine
  - Compatibility
    - Windows
    - Mac OS X
    - other
  - Reproducable research
  - Portability
  - Teaching
- Communication
  - Online
    - Publications
    - Mailing lists
    - IRC (#neurodebian on OFTC)
    - identi.ca/twitter (neurodebian)
    - NeuroDebian Insider Blog
    - team@neuro.debian.net
  - Personal
    - Conference booth
    - Talks
    - Stalking
- Debian
  - Blends
    - Debian Science
    - Debian Med
  - BTS
  - QA
  - Archive
    - i386
    - amd64
    - armel
    - mips(el)
    - kfreebsd-i386/amd64
    - ...
  - Derivatives

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

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.
ICANHASCHEEZBURGER.COM

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

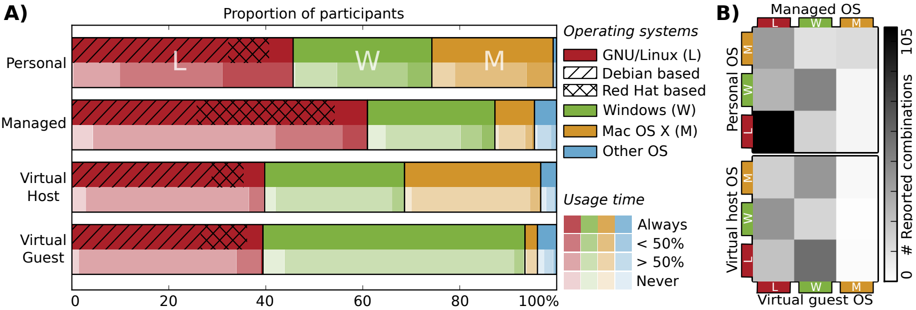
James V. Haxby
Stephen J. Hanson
Stefan Pollmann

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

about the slides:

should become available at	`http://www.pymvpa.org/` & `http://neuro.debian.net`
© 2007-2012	Yaroslav O. Halchenko & Michael Hanke,
snake-brain © 2010	Arno Klein
slide style	inspired by Stefano Zacchiroli
	CC BY-SA 3.0 — Creative Commons Attribution-ShareAlike 3.0

OS market share



Hanke & Halchenko, 2011, Front. Neuroinf.

References

Haxby, J. V., Guntupalli, J. S., Connolly, A. C., Halchenko, Y. O., Conroy, B. R., Gobbini, M. I., Hanke, M., and Ramadge, P. J. (2011). A common, high-dimensional model of the representational space in human ventral temporal cortex. *Neuron*, 72(2):404–416.

Miyawaki, Y., Uchida, H., Yamashita, O., aki Sato, M., Morito, Y., Tanabe, H. C., Sadato, N., and Kamitani, Y. (2008). Visual image reconstruction from human brain activity using a combination of multiscale local image decoders. *Neuron*, 60(5):915–29.

Mumford, J. A., Turner, B. O., Ashby, F. G., and Poldrack, R. A. (2011). Deconvolving bold activation in event-related designs for multivoxel pattern classification analyses. *Neuroimage*.

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

Sato, J. R., Mourão-Miranda, J., da Graça Morais Martin, M., Amaro Jr., E., Morettin, P. A., and Brammer, M. J. (2008). The impact of functional connectivity changes on support vector machines mapping of fMRI data. *Journal of Neuroscience Methods*, In Press, Corrected Proof.