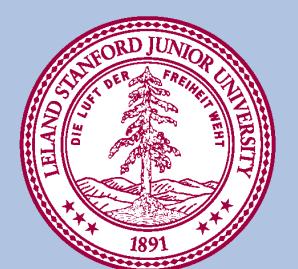




# NIPY: Neuroimaging in Python

<http://neuroimaging.scipy.org>



NeuroSpin

INRIA



Matthew Brett (1), Christopher Burns (1), K. Jarrod Millman (1), Fernando Perez (1),  
Alexis Roche (3), Jonathan Taylor (2), Bertrand Thirion (4), Mark D'Esposito (1)

1. Helen Wills Neuroscience Institute, University of California, Berkeley, U.S.A 2. Dept. of Statistics, Stanford University, U.S.A.  
3. Neurospin, Commissariat à l'Energie Atomique, Gif-sur-Yvette, France. 4. Parietal team, INRIA, Saclay, France.

## Why Python?

- High-level, interactive, easy-to-use, versatile, freely available
- Easy computation-efficient extension with cython/c
- Well suited to design relatively complex neuroimaging objects
- Easy parallel computing
- Presence of well-developed and tested scientific libraries:  
**numpy** (numeric python): array functions  
**scipy** (scientific python): various utilities  
**matplotlib & mayavi** : visualization

## Documentation

- PDF, HTML included in distribution
- Interactive Session help and examples
- Web-based: <http://neuroimaging.scipy.org>



Home | Documentation | User Guide > Tutorials >

**Basic Data IO**

Accessing images using nipy:

While **Nifti** is the primary file format Analyze images (with associated .mat file), and MNI files can also be read.

**Load Image from File**

```
from nipy.io.api import load_image
infile = 'myimage.nii'
myimg = load_image(infile)
myimg.shape
```

**Access Data into an Array**

This allows user to access data in a numpy array.

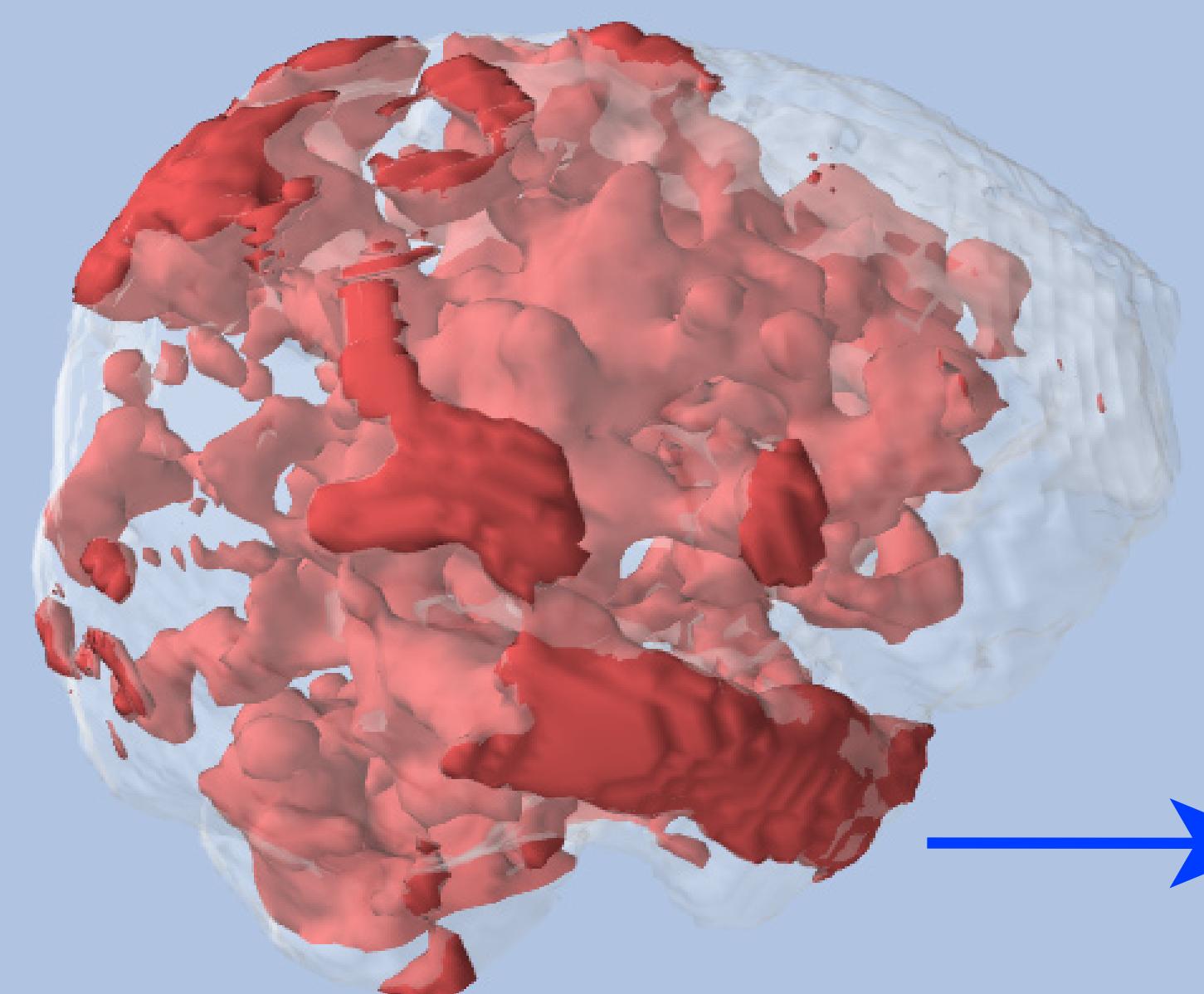
```
Note
This is the correct way to access the data as it applies the proper
intensity scaling to the image as defined in the header
from nipy.io.api import load_image
import numpy as np
myimg = load_file('myfile')
```

In [34]:  
In [35]:  
In [36]: load\_image?  
Load an image from the given filename.  
Load an image from the file specified by ``filename``.  
Parameters  
-----  
filename : string  
 Should resolve to a complete filename path.  
mode : Either 'r' or 'r+'  
Returns  
-----  
image : An 'Image' object  
 If successful, a new 'Image' object is returned.  
See Also  
-----  
save\_image : function for saving images  
fromarray : function for creating images from numpy arrays  
Examples  
-----  
>>> from nipy.io.api import load\_image  
>>> from nipy.testing import onfile  
>>> img = load\_image(onfile)  
>>> img.shape  
(25, 35, 25)

## What's in Nipy Now?:

- GLM Model (Model specification/fit)
- Parametric tests: (false discovery rate, Gaussian Random theory)
- Non-parametric Tests: (voxel-level, cluster-level, mixed effects)
- Spatial Models:  
~region-of-interest based analysis  
~anatomo-functional parcellations  
~structural models (brain functional landmarks)

## Visualize Statistical Results



```
def view_thresholded3d(design, contrast, threshold, inequality=np.greater):
    """A mayavi isosurface view of thresholded t-statistics

    Parameters
    -----
    design: ('block', 'event')
        Experimental design to view.
    contrast: ('speaker_0', 'speaker_1', 'sentence_0', 'sentence_1')
        Which contrast of interest to view.
    threshold: float
        Value of the threshold the t-map.
    inequality: (np.greater, np.less)
        Measure of inequality to compare t-map against `threshold`.

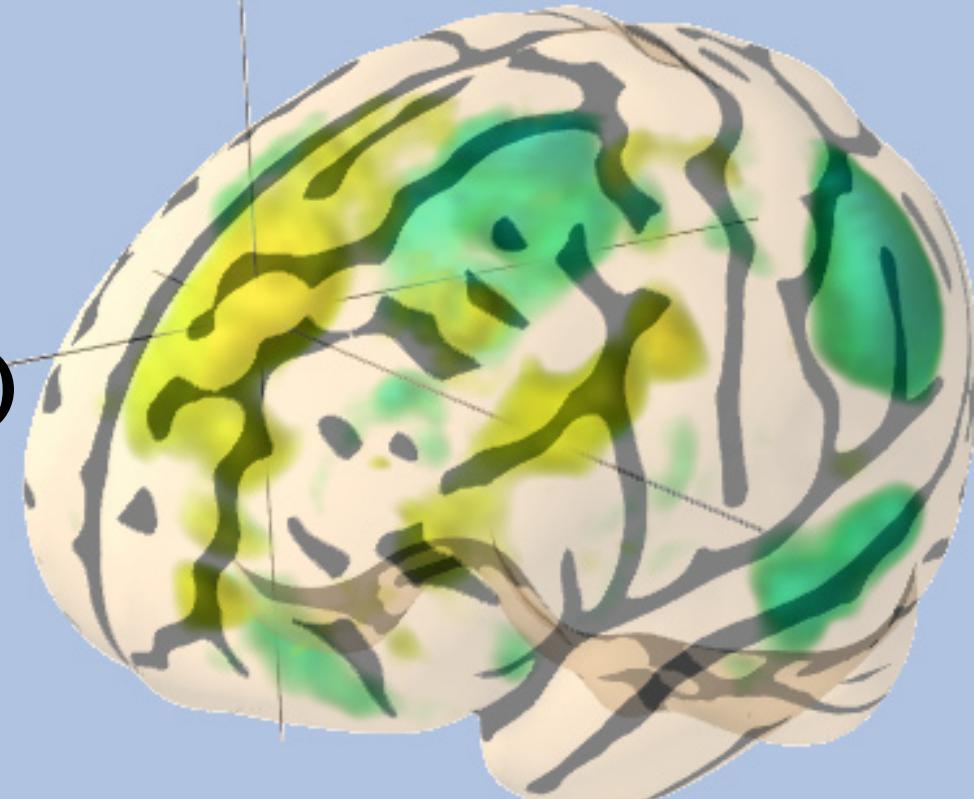
    """
    maska = np.asarray(MASK)
    tmap = np.array(load_image_fiac('group', design, contrast, 't.nii'))
    test = inequality(tmap, threshold)
    tval = np.zeros(tmap.shape)
    tval[test] = tmap[test]
    avaganata = np.array(AVGANAT)
    avaganat_iso = ML.contour3d(avaganata * maska, opacity=0.3, contours=[3600], color=(0.8, 0.8, 0.8))
    avaganat_iso.actor.property.backface_culling = True
    avaganat_iso.actor.property.ambient = 0.3
    tval_iso = ML.contour3d(tval * MASK, color=(0.8, 0.3, 0.3), contours=[threshold])
    return avaganat_iso, tval_iso
```

**Scientific Python  
is a high level  
platform to support  
long-term sustainable  
development in  
neuroimaging**

**Automated Test Framework**  
Ensures robust and reproducible software  
Bugs found early in development cycle  
Tests documentation and code

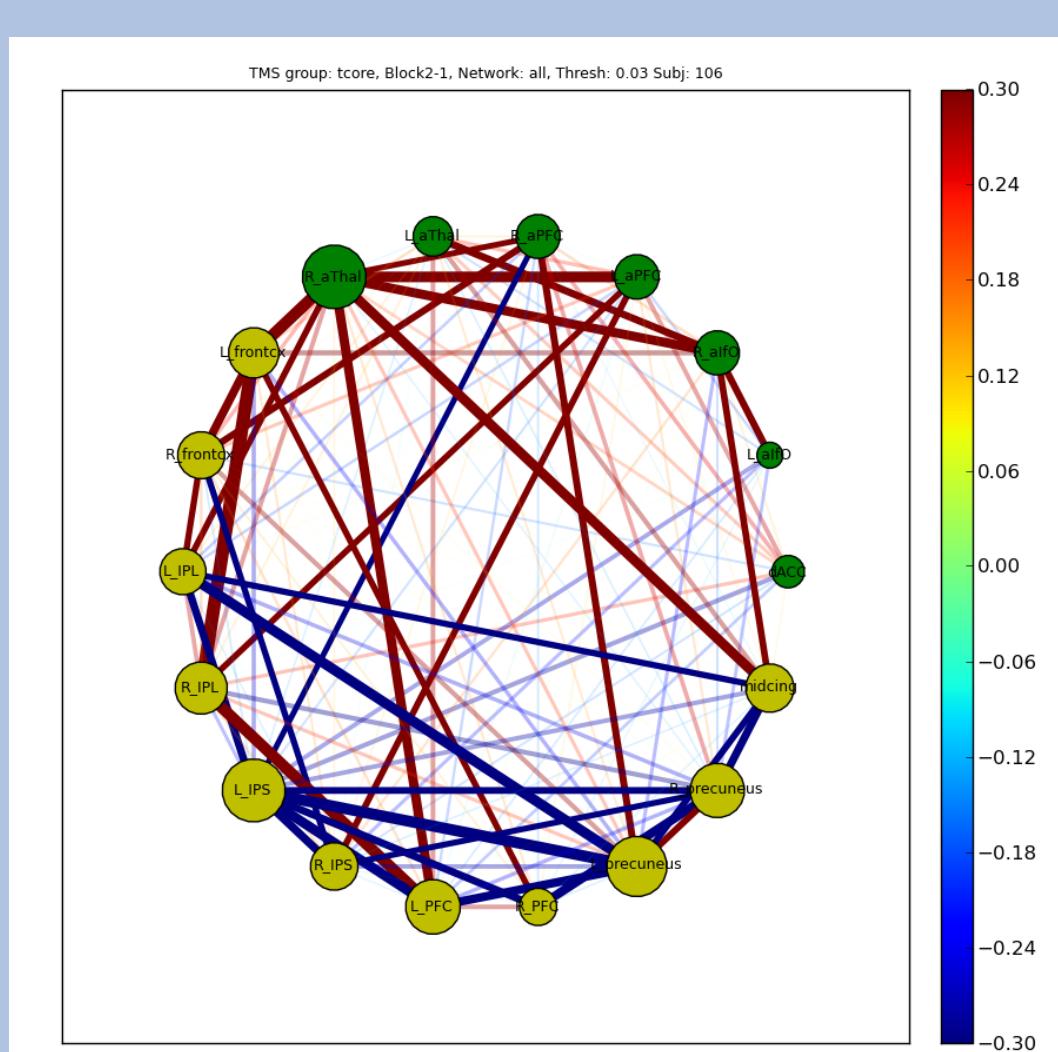
```
def test_file_roundtrip():
    img = load_image(datajoin('avg152T1.nii.gz'))
    fd, name = mkstemp(suffix='.nii.gz')
    tmpfile = open(name)
    save_image(img, tmpfile.name)
    img2 = load_image(tmpfile.name)
    data = np.asarray(img)
    data2 = np.asarray(img2)
    tmpfile.close()
    os.unlink(name)
    # verify data
    yield assert_true, np.allclose(data2, data)
    yield assert_true, np.allclose(data2.mean(), data.mean())
    yield assert_true, np.allclose(data2.min(), data.min())
    yield assert_true, np.allclose(data2.max(), data.max())
    # verify shape and ndims
    yield assert_equal, img2.shape, img.shape
    yield assert_equal, img2.ndim, img.ndim
    # verify affine
    yield assert_true, np.allclose(img2.affine, img.affine)
```

**Resting State ICA  
(2D/3D Visualization)**  
G.Varoquaux  
(Poster 536 F-PM)

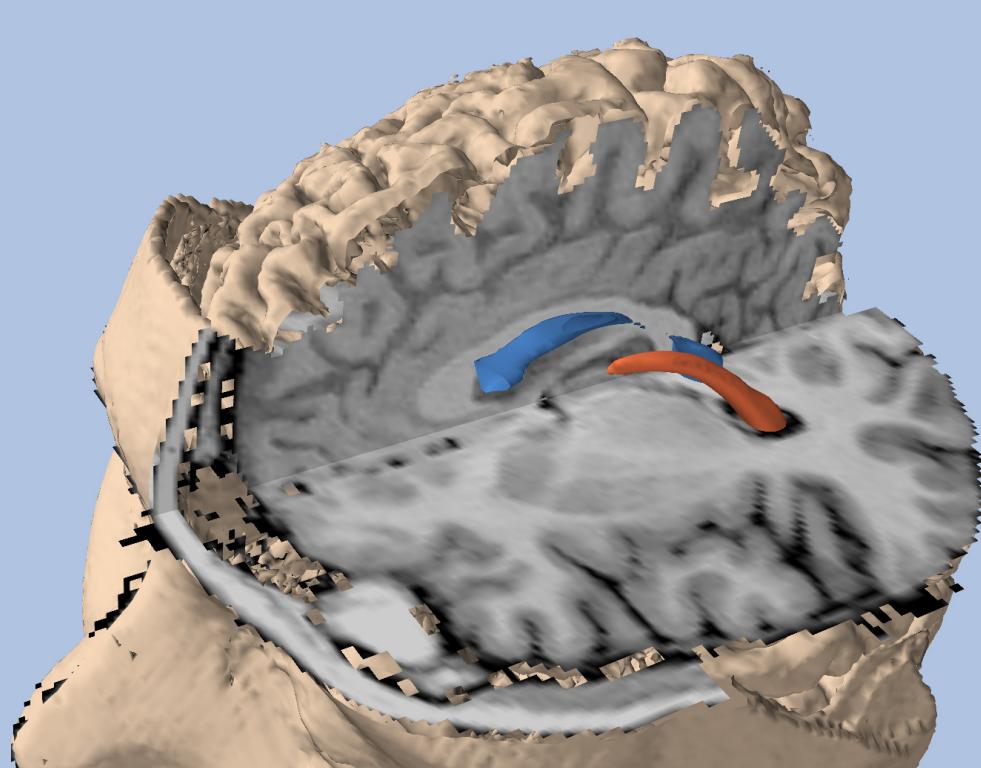
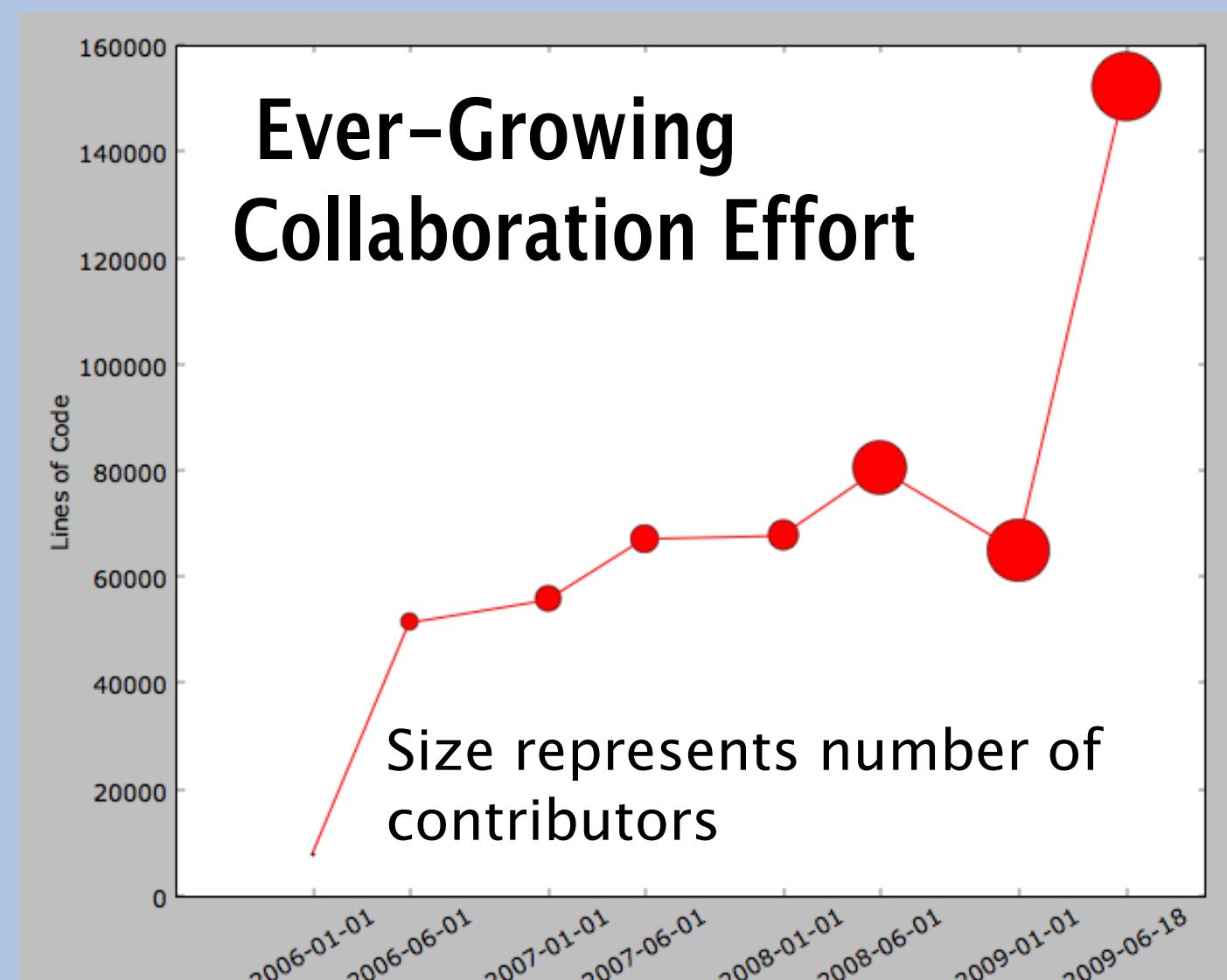


## Access to Other Scientific Tools

- pymvpa : multivariate analysis tools in python
- mlpy: machine learning toolbox
- pygifti: python binding of gifti library
- pyNN and many other Neural Network packages
- sympy: symbolic mathematics
- brainvisa: interface to many anatomical/diffusion and visualization functions



Visualizing changes in coherence of resting state functional connectivity networks after 10 minutes of repetitive TMS (1 Hz). Thanks to E. Nomura, C. Gratton and M. D'Esposito (Poster 282 M-PM)



## What's Currently Being Developed

- spatial normalization
- handling regressors/contrasts in symbolic python formalism
- connectivity models: inference of brain connectivity/ ICA
- Data Quality checking Methods
- Timeseries/Coherence Models
- Probabilistic Spatial Models
- Haemodynamic Response Estimation
- Interfaces to other existing packages for cross validation
- Pipeline architecture/ parallelization