Explore the brain with Nilearn

Darya Chyzhyk

Parietal team, INRIA, Paris-Saclay

PyCon Otto, Florence

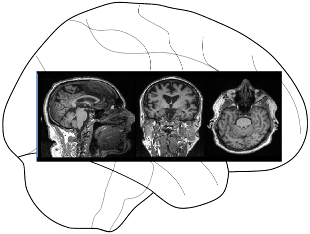
April 6th-9th 2017

Content

- Concept
- MRI
- Data
- Decoding
- Functional connectivity
- Joblib
- Visualization
- Documentation

Concept

ni learn =



+

scikit learn

SciPy

NiPy
Nibabel

matplotlib

NumPy

Concept

How to start

- Data
  - progress and development need more data
  - public data, make more people working on data
- Soft
  - Standardize procedure
  - Interdisciplinary work
- Documentation
  - justification
  - help

Motivation

- Make science open
- Promote research and collective work
- Visibility in the code
- Shared data

Concept

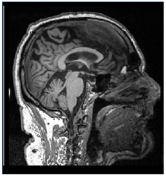
How to start

- Data
  - progress and development need more data
  - public data, make more people working on data
- Soft
  - Standardize procedure
  - Interdisciplinary work
- Documentation
  - justification
  - help

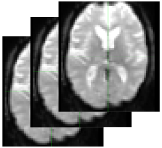
Motivation

- Make science open
- Promote research and collective work
- Visibility in the code
- Shared data

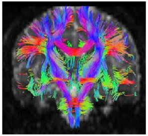
MRI



Anatomical MRI
3D



Functional MRI
4D



DTI

Data

- Clinical data (hospital) from private project
- Open data project:
  - UK Biobank (500,000 participants),
  - Human Connectome Project (HCP)
  - ......
  - Autism Brain Imaging Data Exchange (ABIDE)
  - Alzheimer's Disease Neuroimaging Initiative (ADNI)
- **Fetching**

Data

- Clinical data (hospital) from private project
- Open data project:
  - UK Biobank (500,000 participants),
  - Human Connectome Project (HCP)
  - The Autism Brain Imaging Data Exchange (ABIDE)
  - Alzheimer's Disease Neuroimaging Initiative (ADNI)
- Fetching

Fetch COBRE datasets shipped with Nilearn

In [ ]:
```
# COBRE datasets release version 0.17 preprocessed using NIAK pipeline. This
# release consists of light weight data with standard preprocessing
# steps used in functional MRI setting.

from nilearn import datasets

# all subjects (146 given in n_subjects argument)
cobre_data = datasets.fetch_cobre(n_subjects=146)
print(cobre_data.keys())
```

Fetching

Structural, functional MRI, atlases

7.2. `nilearn.datasets`: Automatic Dataset Fetching

Helper functions to download NeuroImaging datasets

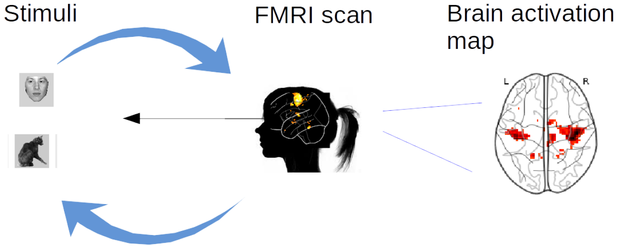
User guide: See the *Fetching open datasets from Internet* section for further details.

Functions:

fetch_atlas_craddock_2012([data_dir, url, ...])	Download and return file names for the Craddock 2012 parcellation
fetch_atlas_destrieux_2009([lateralized, ...])	Download and load the Destrieux cortical atlas (dated 2009)
fetch_atlas_harvard_oxford(atlas_name[, ...])	Load Harvard-Oxford parcellation from FSL if installed or download it.
fetch_atlas_msdl([data_dir, url, resume, ...])	Download and load the MSDL brain atlas.
fetch_coords_power_2011()	Download and load the Power et al.
fetch_atlas_smith_2009([data_dir, mirror, ...])	Download and load the Smith ICA and BrainMap atlas (dated 2009)
fetch_atlas_yeo_2011([data_dir, url, ...])	Download and return file names for the Yeo 2011 parcellation.
fetch_atlas_aal([version, data_dir, url, ...])	Downloads and returns the AAL template for SPM 12.
fetch_atlas_basc_multiscale_2015([version, ...])	Downloads and loads multiscale functional brain parcellations
fetch_coords_dosenbach_2010([ordered_regions])	Load the Dosenbach et al.
fetch_abide_pcp([data_dir, n_subjects, ...])	Fetch ABIDE dataset
fetch_adhd([n_subjects, data_dir, url, ...])	Download and load the ADHD resting-state dataset.
fetch_haxby([data_dir, n_subjects, ...])	Download and loads complete haxby dataset
fetch_icbm152_2009([data_dir, url, resume, ...])	Download and load the ICBM152 template (dated 2009)
fetch_icbm152_brain_gm_mask([data_dir, ...])	Downloads ICBM152 template first, then loads 'gm' mask image.

Decoding/Encoding

Neuroimaging problem



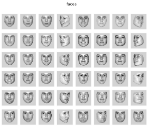
Decoding uses brain activity to predict information about the stimuli

Encoding uses stimuli to predict activity in the brain

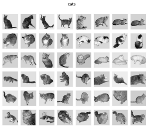
Decoding

Haxby example

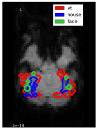
1 subject and two categories: faces and cats



Face stimuli



Cat stimuli



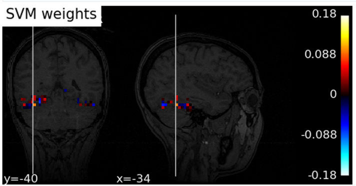
Masks

Decoding with SVM

```python
prediction = svc.predict(fmri_masked)
print(prediction)
```

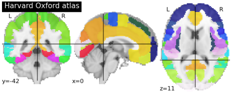
Out:

```
['face' 'face' 'face' 'face' 'face' 'face' 'face' 'face' 'face' 'cat' 'cat'
 'cat' 'cat' 'cat' 'cat' 'cat' 'cat' 'cat' 'face' 'face' 'face' 'face'
 'face' 'face' 'face' 'face' 'face' 'cat' 'cat' 'cat' 'cat' 'cat' 'cat'
 'cat' 'cat' 'cat' 'face' 'face' 'face' 'face' 'face' 'face' 'face' 'face'
```

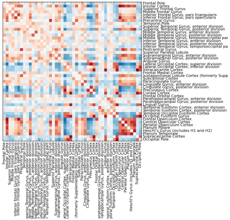


Functional connectivity¶

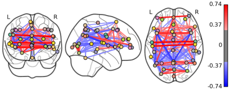
Functional connectivity is the connectivity between brain regions that share functional properties.
FC can be defined as the temporal correlation between spatially remote neurophysiological events



Time series extraction

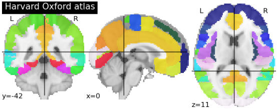


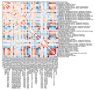
Correlation matrix

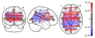


Connectome

Functional connectivity¶





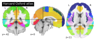


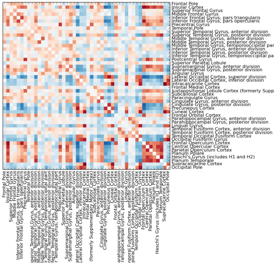
Time series extraction

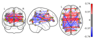
```python
# We use pre-generated atlas (BASC) to parcellate brain into ROIs.
basc_atlases = datasets.fetch_atlas_basc_multiscale_2015()
atlas_img = basc_atlases['scale122']
```

```python
# For timeseries extraction from functional images, we import
# `NiftiLabelsMasker` from input_data module.

from nilearn.input_data import NiftiLabelsMasker

timeseries_extractor = NiftiLabelsMasker(
    labels_img=atlas_img,  # brain atlas
    smoothing_fwhm=6.,  # Smoothing
    standardize=True, detrend=True, # timeseries signals
    memory='nilearn_cache',  # joblib
    memory_level=2,  # Level of caching
    verbose=2)  # useful to see processing

# We call `fit_transform` on each subject fMRI data.
# `fit` will prepare 2D data matrix from 4D functional images

subjects_timeseries = []
for func_img in cobre_data.func:
    signals = timeseries_extractor.fit_transform(func_img)
    subjects_timeseries.append(signals)
```

Functional connectivity¶



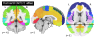


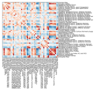


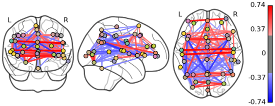
Correlation matrix

```
from nilearn.connectome import ConnectivityMeasure
correlation_measure = ConnectivityMeasure(kind='correlation')
correlation_matrix = correlation_measure.fit_transform([time_series])[0]
```

Functional connectivity¶







Connectome

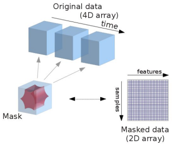
```python
from nilearn import plotting
coords = atlas.region_coords

# We threshold to keep only the 20% of edges with the highest value
# because the graph is very dense
plotting.plot_connectome(correlation_matrix, coords,
                         edge_threshold="80%", colorbar=True)

plotting.show()
```

Joblib

Joblib is a set of tools to provide lightweight pipelining in Python:

Avoid computing twice

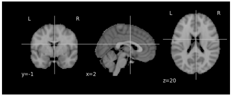
```
from nilearn.input_data import NiftiLabelsMasker

# We initialize the timeseries extractor by setting standard processing
# parameters required for the extraction.

timeseries_extractor = NiftiLabelsMasker(
    labels_img=atlas_img,  # brain atlas
    smoothing_fwhm=6.,  # Smoothing
    standardize=True, detrend=True, # timeseries signals
    memory='nilearn_cache',  # joblib
    memory_level=2,  # Level of caching
    verbose=2)  # useful to see processing
```

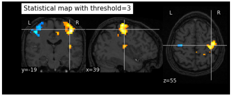


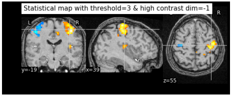
Parallel computing, *n_jobs* - the number of CPUs to use to do the computation

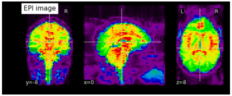
```
from nilearn.input_data import MultiNiftiMasker

masker = MultiNiftiMasker(mask_img,smoothing_fwhm=6.,
                          standardize=True, detrend=True, mask_strategy='epi',
                          n_jobs=5, verbose=10)
```

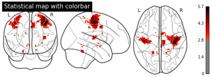
Visualization

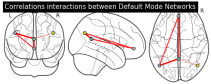












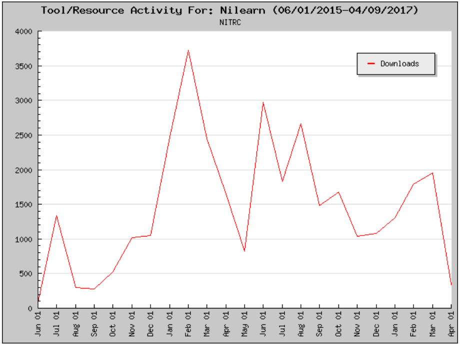
Documentation

- User guide:
  - description
  - references
- Examples:
  - notebook
  - learn from working code

Conclusions

Nilearn is advanced machine learning, pattern recognition and multivariate statistical techniques on neuroimaging data

- Python ecosystem
- Impact in comunity



Thank you!

A. Abraham, L. Estève, E. Dohmatob, D. Bzdok, K. Dadi, A. Mensch, P. Gervais, V. Fritsch, S. Bougacha, B. Cipollini, M. Rahim, M. Perez-Guevara, K. J. Gorgolewski, Ó. Nájera, M. Eickenberg, A. Abadie, Y. Schwartz, A. Hoyos Idrobo, K. Shmelkov, F. Pedregosa, A. Mueller, J. Kossaifi, J. Grobler, A. Gramfort, M. Hanke, B. Thirion, and G. Varoquaux.

Reference:

Abraham, A.; Pedregosa, F.; Eickenberg, M.; Gervais, P.; Mueller, A.; Kossaifi, J.; Gramfort, A.; Thirion, B. & Varoquaux, G. Machine learning for neuroimaging with scikit-learn Frontiers in Neuroinformatics, 2014, 8, 14

https://nilearn.github.io/