Nilearn

Python library for MRI data analysis

What is Nilearn?

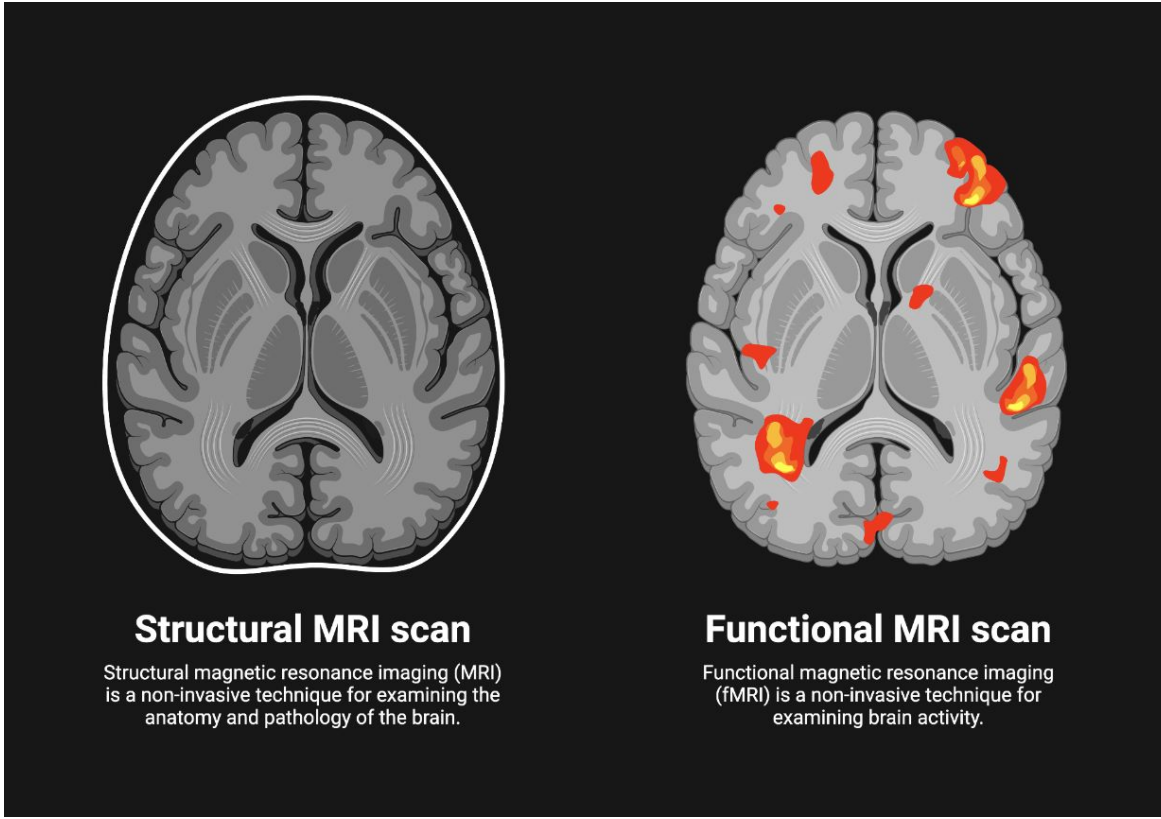
- open-source Python package with a set of tools for loading, preprocessing, analyzing, and visualizing neuroimaging data
- it builds upon Python libraries such as NumPy, SciPy, scikit-learn, and matplotlib, providing a high-level interface for working with neuroimaging data
- accessible to users with varying levels of expertise in neuroimaging and programming

What is neuroimaging?

- branch of medical imaging that focuses on the brain
- used for:
  - diagnosing disease
  - assessing brain health
  - studying how the brain works
  - studying how various activities impact the brain

Nilearn is mostly used for MRI and fMRI data:

- structural MRI scan uses a magnet, radio waves and computer processing to generate 3D pictures of the brain
- fMRI scan uses the same MRI machine, but it tracks blood flow in different parts of the brain



https://share.baptisthealth.com/difference-between-mri-and-fmri-scans

Nilearn simplifies the process of analyzing and interpreting neuroimaging data, making it a valuable tool for researchers in neuroscience, psychology, and related fields.

It provides a wide range of functionalities for brain images, including:

- **Preprocessing**
  e.g motion correction, spatial smoothing
- **Statistical analysis**
- **Machine learning**
  integration with machine learning libraries such as scikit-learn
- **Visualization**
  tools for plotting brain images, statistical maps, and connectivity matrices
- **Connectivity analysis**
  functional connectivity in brain networks

neuroimaging data

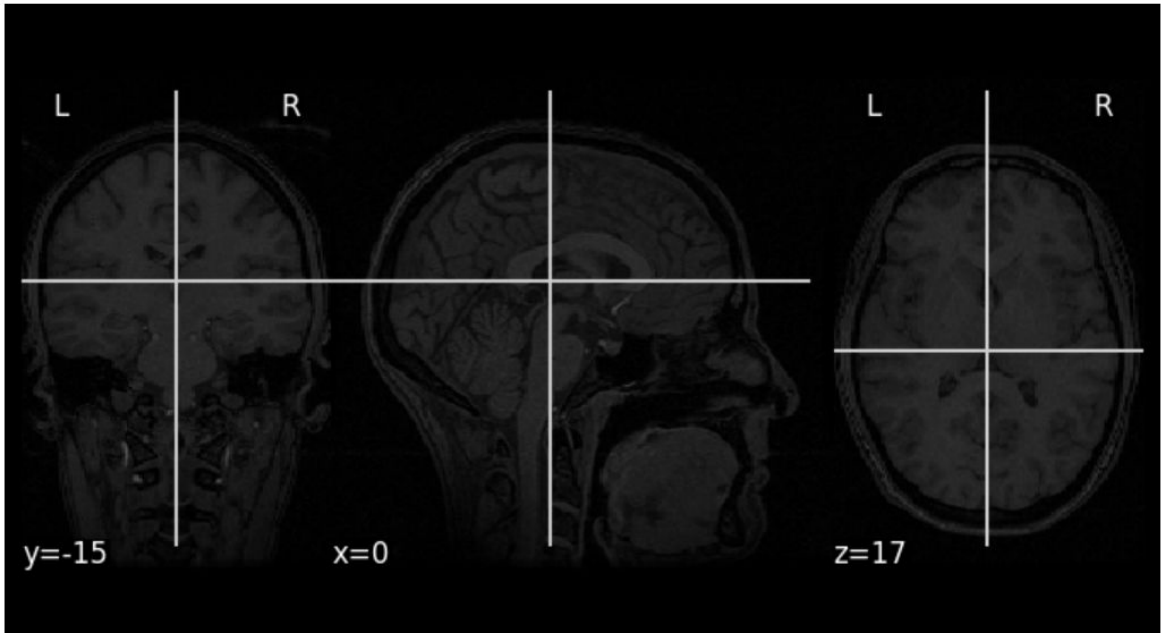
- NIfTI (Neuroimaging Informatics Technology Initiative) format for handling neuroimaging data
- NIfTI files typically have the extension ".nii"
- NifTi data structure is the standard way of sharing data in neuroimaging research. Three main components are:
  - raw scans in form of a numpy array `nilearn.image.get_data()`
  - affine transformation matrix `nilearn.image.affine`
  - header `nilearn.image.header`
- Nilearn supports 3D and 4D image objects

```
from nilearn import image

img = image.load_img(
    'Mouse_rest_3xTG/sub-jgrADc1NT/ses-1/func/sub-jgrADc1NT_ses-1_task-rest_acq-EPI_bold.nii')
print(img)
```

```
<class 'nibabel.nifti1.Nifti1Image'>
data shape (90, 60, 28, 600)
affine:
[[ 0.18888889  0.          0.         -8.59444427]
 [ 0.          0.          0.40000001 -4.51052999]
 [ 0.         -0.15000001  0.          9.64795017]
 [ 0.          0.          0.          1.        ]]
metadata:
<class 'nibabel.nifti1.Nifti1Header'> object, endian='<'
sizeof_hdr      : 348
data_type       : b''
db_name         : b''
extents         : 0
session_error   : 0
regular         : b'r'
dim_info        : 0
dim             : [  4  90  60  28 600   1   1   1]
intent_p1       : 0.0
intent_p2       : 0.0
intent_p3       : 0.0
intent_code     : none
datatype        : int16
bitpix          : 16
slice_start     : 0
pixdim          : [1.         0.18888889 0.15       0.4        1.         1.
 1.         1.        ]
vox_offset      : 0.0
scl_slope       : nan
scl_inter       : nan
slice_end       : 0
slice_code      : alternating increasing
xyzt_units      : 10
cal_max         : 0.0
cal_min         : 0.0
slice_duration  : 0.0
toffset         : 0.0
glmax           : 255
glmin           : 0
descrip         : b'5_EPI_multi:ACQ_BRUKER_PVM'
aux_file        : b''
```

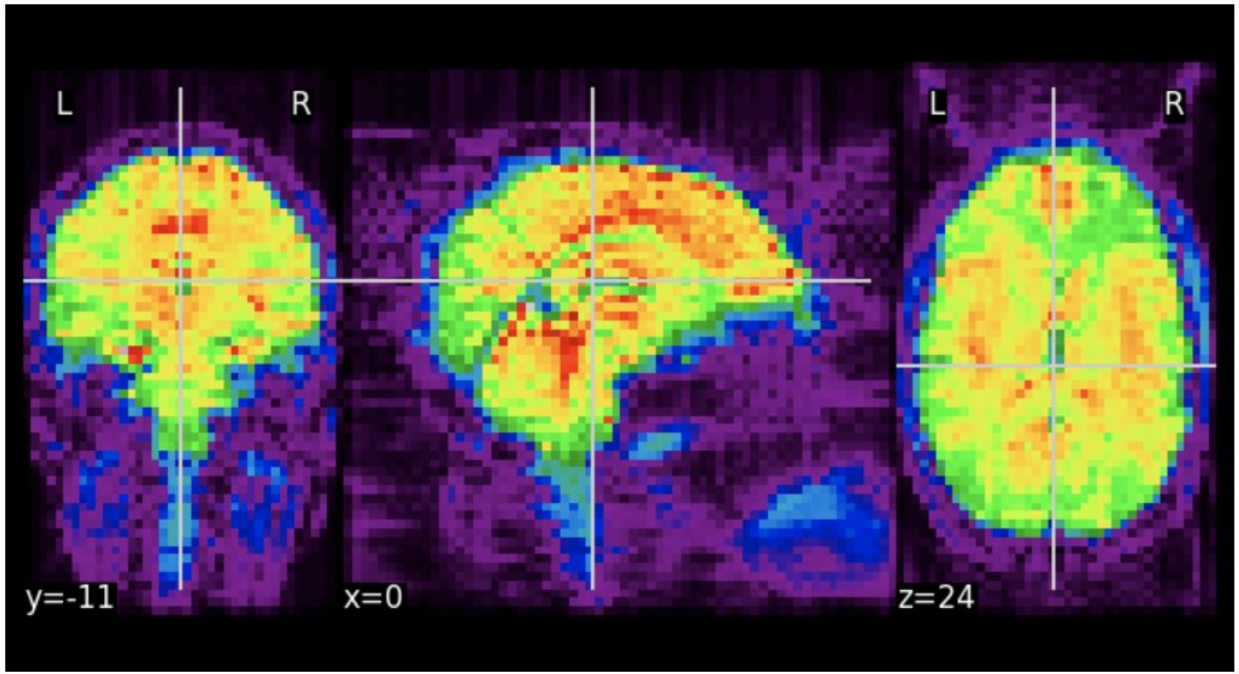
```
from nilearn import datasets
from nilearn import plotting

haxby_dataset = datasets.fetch_haxby()
print(
    f"First subject anatomical nifti image is at: {haxby_dataset.anat[0]}"
)

plotting.plot_anat(haxby_dataset.anat[0])
```

```
First subject anatomical nifti image is at: C:\Users\michalina/nilearn_data\haxby2001\subj2\anat.nii.gz
<nilearn.plotting.displays._slicers.OrthoSlicer at 0x27281006e10>
```

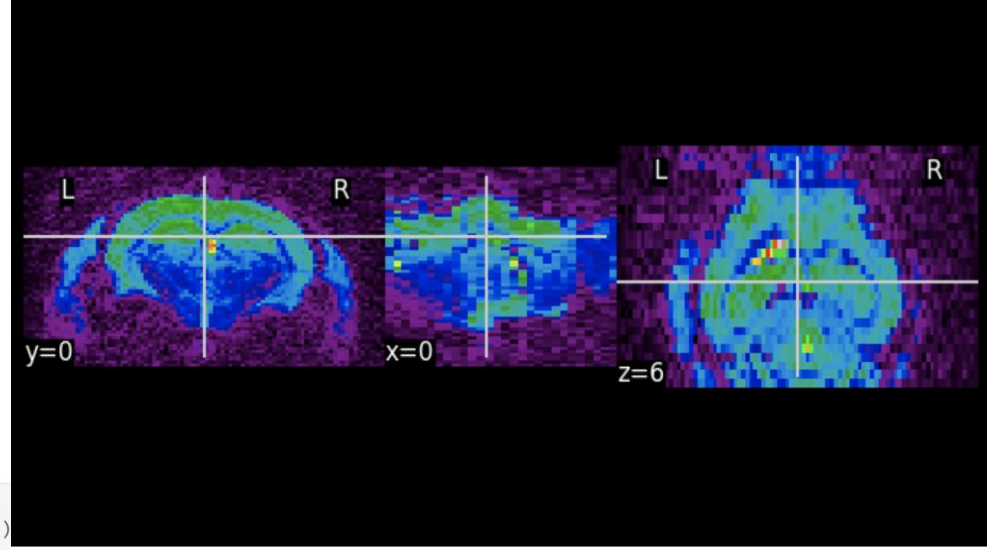


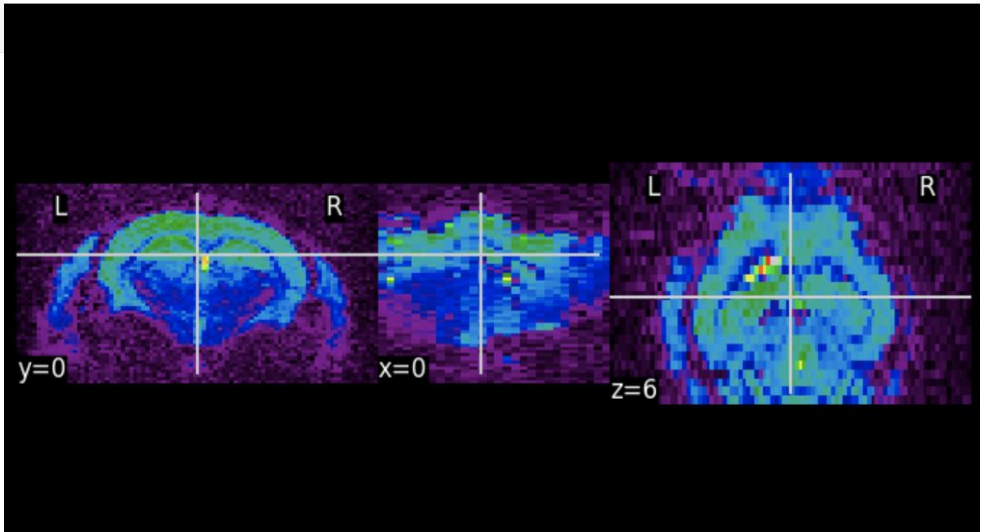
```
from nilearn.image import mean_img

print(
    f"First subject functional nifti image is at: {haxby_dataset.func[0]}"
)

plotting.plot_epi(mean_img(haxby_dataset.func[0]))
```

```
First subject functional nifti image is at: C:\Users\michalina/nilearn_data\haxby2001\subj2\bold.nii.gz
<nilearn.plotting.displays._slicers.OrthoSlicer at 0x27281004290>
```



```
img = image.load_img(
    'Mouse_rest_3xTG/sub-jgrADc1NT/ses-1/func/sub-jgrADc1NT_ses-1_task-rest_acq-EPI_bold.nii')
plotting.plot_epi(image.index_img(img, 0))
plotting.plot_epi(image.index_img(img, 599))
```





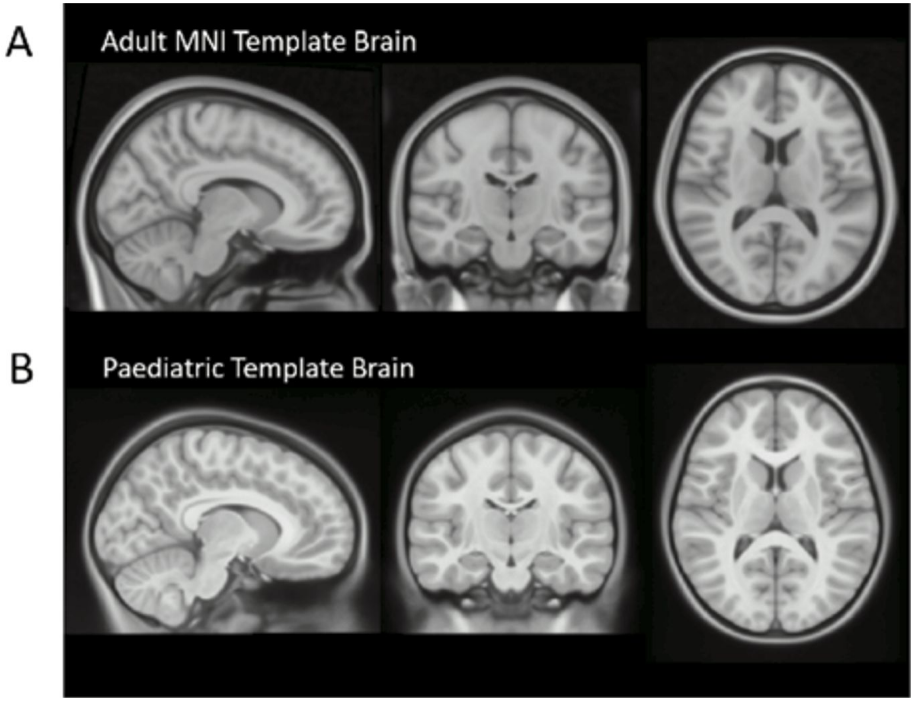
Atlases and templates

Nilearn includes multiple built-in atlases and templates commonly used in neuroimaging research, particularly in the context of region-based analyses and spatial normalization.

They enable users to perform a wide range of neuroimaging analyses and visualize results in a standardized and interpretable manner.

Aspect	Atlases	Templates
Definition	Predefined maps dividing the brain into regions based on anatomical or functional criteria.	Standardized brain images used as a reference space for spatial normalization and alignment.
Purpose	Used for region-based analyses, defining Regions of Interest (ROIs) for statistical analysis or visualization.	Used to warp individual subject data to a common reference space, enabling comparison across subjects or studies.
Content	Contains specific brain regions or networks defined by neuroscientific criteria.	Represent the entire brain in a standardized coordinate system, facilitating alignment and comparison across different subjects.
Examples	Harvard-Oxford, AAL, Desikan-Killiany, etc.	MNI152, Talairach, etc.
Alignment	May require alignment to a common reference space for consistency across studies.	Serves as the reference space to which individual subject data are aligned for consistency across studies.

Example brain templates

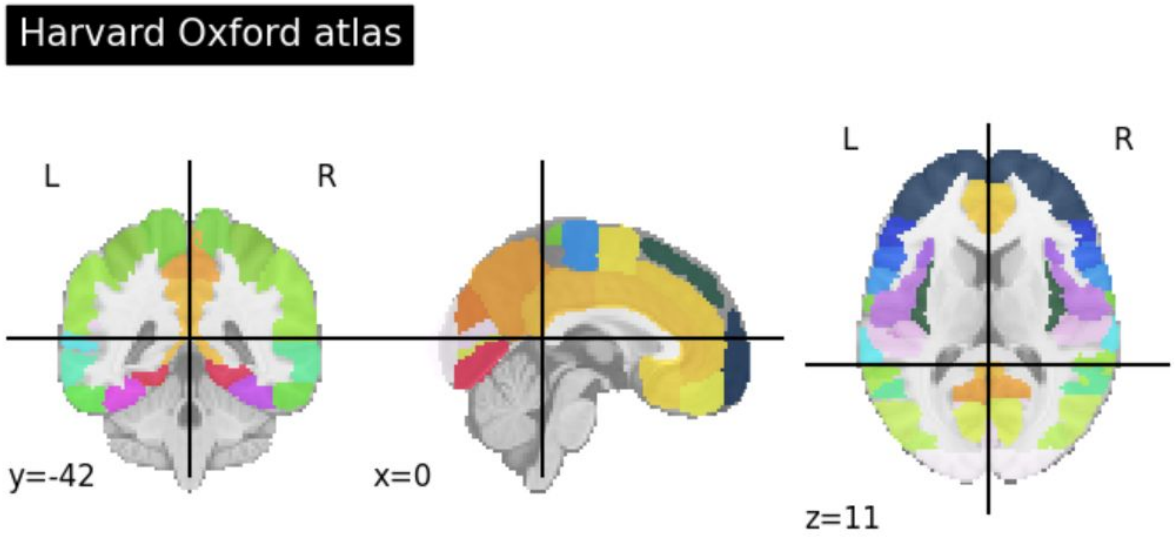


A) Adult MNI template brain commonly used as the standard space image for MRI analysis.
B) Paediatric template brain created using images from 112 children of ages between 7 and 11 years. Use of an age appropriate template is important because of differences between the brains of adults and children.

Regions of a reference atlas

```
from nilearn import datasets, plotting

atlas = datasets.fetch_atlas_harvard_oxford("cort-maxprob-thr25-2mm")
plotting.plot_roi(atlas.filename, title="Harvard Oxford atlas")
```

```
<nilearn.plotting.displays._slicers.OrthoSlicer at 0x27281184980>
```



Functional connectome

By combining functional connectome data with machine learning techniques, one can use Nilearn to build models that can be used to understand cognitive states, predict clinical outcomes, or identify biomarkers of disease.

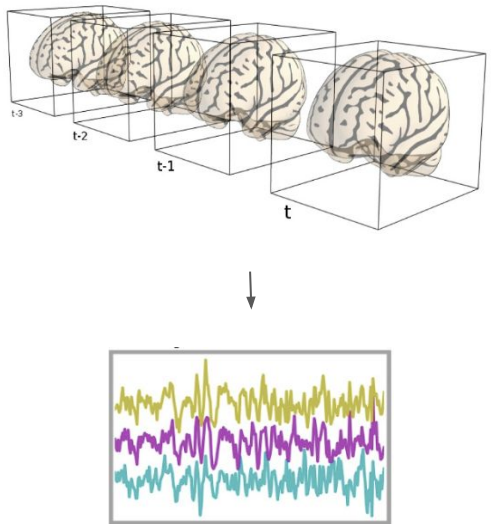
Nilearn's visualization capabilities aid in interpreting connectivity patterns and understanding the underlying neural mechanisms.

functional connectivity - statistical relationship between specific physiological signals in time

functional connectome - set of connections representing brain interactions between regions

Obtaining functional connectiome with Nilearn

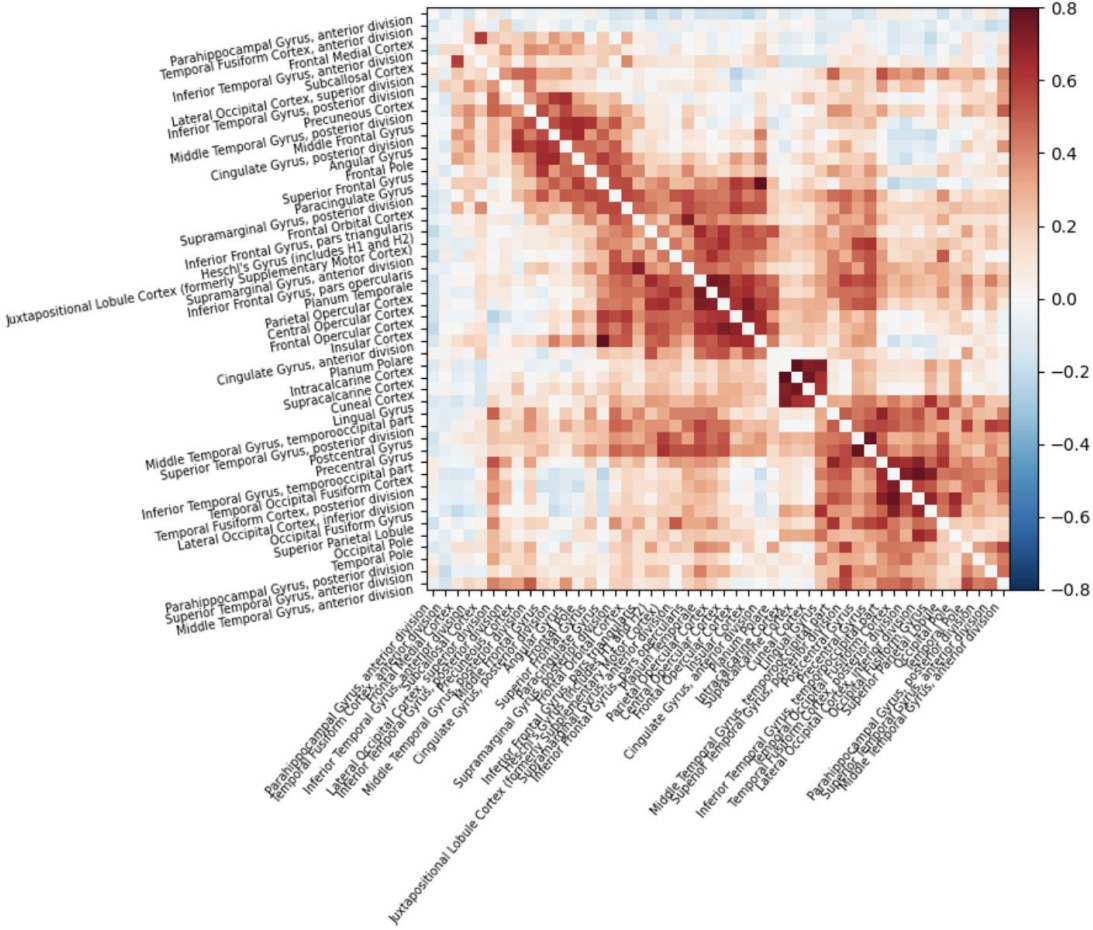
1. Extract time series representing the average activity within each ROI over the course of the fMRI scan (transform 4D data into 2D data)

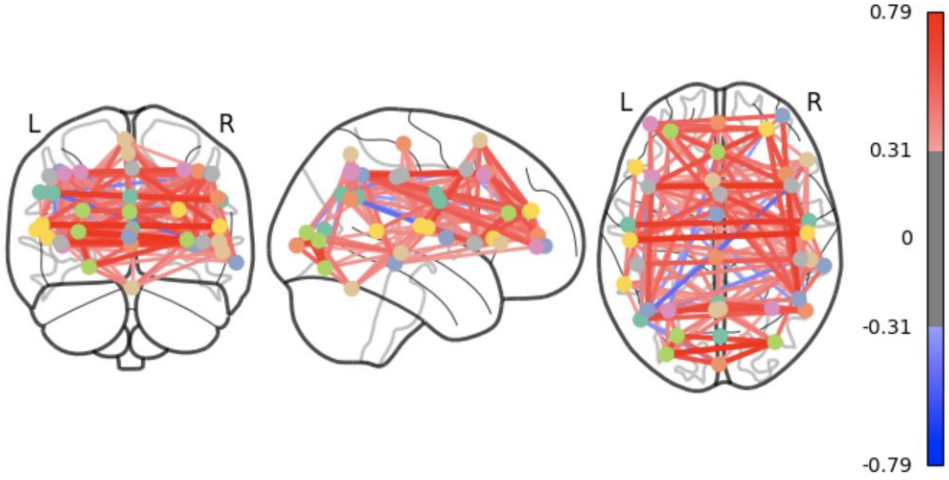


2. Calculate functional connectivity between pairs of ROIs (typically done using correlation-based measures) and organize the values into a connectivity matrix, representing the functional connectome of the brain

`nilearn.connectome.ConnectivityMeasure`

3. Use connectivity matrices as features in machine learning algorithms to build predictive models





connectivity matrix represented as a network graph, where nodes represent ROIs and edges represent functional connections between them

`nilearn.plotting.plot_connectome`

The End