**Image Manipulation using Nibabel:**

This Jupyter notebook provides a detailed introduction to working with neuroimaging data using the **Nibabel** library in Python. Below are the key points covered in the notebook:

**Overview**

The notebook focuses on:

* Loading, modifying, saving, and visualizing neuroimages.
* Understanding the data structures involved in neuroimaging.
* Using **Nibabel**, a low-level Python library for handling various neuroimaging formats (e.g., NIfTI, FreeSurfer, DICOM).

**Key Sections**

**1. Setup**

* **Imports**: Essential libraries such as nilearn, nibabel, numpy, and matplotlib are imported to handle image processing and visualization.

python

**from** nilearn **import** plotting

**import** numpy **as** np

**import** nibabel **as** nb

**import** pylab **as** plt

%matplotlib inline

**2. Loading and Inspecting Images**

* **Loading an Image**: A functional MRI image is loaded using nibabel.load(). The image file is in NIfTI format (.nii.gz).

python

img = nb.load('/data/ds000114/sub-01/ses-test/func/sub-01\_ses-test\_task-fingerfootlips\_bold.nii.gz')

* **Inspecting Image Metadata**: The header and affine transformation matrix of the image are printed. These provide details about the image's dimensions, voxel sizes, and orientation.

python

**print**(img) *# Prints metadata such as data shape, affine matrix, and header information.*

**3. Accessing Specific Parameters**

* **Extracting Data**: The image data is extracted into a NumPy array using img.get\_fdata(). This array can be manipulated like any standard NumPy array.

python

data = img.get\_fdata()

affine = img.affine *# Affine transformation matrix*

header = img.header['pixdim'] *# Voxel resolution and TR (repetition time)*

**4. Visualizing Data**

* **Plotting Slices**: A slice of the data is plotted using matplotlib.pyplot.imshow(). The shape of the data is also printed.

python

plt.imshow(data[:, :, data.shape[2] // 2, 0].T, cmap='Greys\_r')

**print**(data.shape) *# (64, 64, 30, 184)*

**5. Exercise: Loading T1 Data**

* An exercise is provided to load a T1-weighted anatomical image and plot it similarly to the functional image.

python

t1 = nb.load('/data/ds000114/sub-01/ses-test/anat/sub-01\_ses-test\_T1w.nii.gz')

data = t1.get\_fdata()

plt.imshow(data[:, :, data.shape[2] // 2].T, cmap='Greys\_r')

**print**(data.shape) *# (256, 156, 256)*

**6. Affine Transformation**

* The affine matrix is used to convert voxel coordinates into real-world coordinates (in millimeters). Additionally, it encodes information about voxel sizes and axis orientation.

python

x, y, z, \_ = np.linalg.pinv(affine).dot(np.array([0, 0, 0, 1])).astype(int)

nb.aff2axcodes(affine) *# Returns axis orientation ('L', 'A', 'S')*

nb.affines.voxel\_sizes(affine) *# Returns voxel sizes ([3.99999995, 4.00000009, 3.99997491])*

**7. Header Information**

* The header contains metadata about the image (e.g., voxel sizes, units). It can be accessed directly or through helper functions.

python

t1.header.get\_zooms() *# Returns voxel size in each dimension (1.0, 1.2993759, 1.0)*

t1.header.get\_xyzt\_units() *# Returns units ('mm', 'sec')*

t1.header.get\_qform() *# Returns qform matrix for spatial orientation*

**8. Creating and Saving Images**

* The notebook demonstrates how to modify image data (e.g., rescaling) and save it as a new NIfTI file while preserving the original header and affine.

python

*# Rescale data to unsigned byte format (uint8)*

rescaled = ((data - data.min()) \* 255. / (data.max() - data.min())).astype(np.uint8)

*# Save new image with same affine and header*

new\_img = nb.Nifti1Image(rescaled, affine=img.affine, header=img.header)

nb.save(new\_img, '/tmp/rescaled\_image.nii.gz')

* The notebook also shows how to correct issues with the header when saving images by using set\_data\_dtype().

python

img.set\_data\_dtype(np.uint8) *# Corrects header dtype before saving again.*

**9. File Size Comparison**

* After rescaling the image data type from int16 to uint8, the file size is reduced significantly.

bash

!du -hL /tmp/rescaled\_image.nii.gz /data/ds000114/sub-01/ses-test/func/sub-01\_ses-test\_task-fingerfootlips\_bold.nii.gz

*# Output shows that the rescaled file is smaller.*

**Conclusion**

The notebook introduces key functionalities of Nibabel for working with neuroimaging data:

* Loading and inspecting NIfTI images.
* Accessing metadata such as headers and affine transformations.
* Visualizing slices of neuroimaging data.
* Modifying and saving images while preserving important metadata like headers and affines.

This foundational knowledge prepares users for more advanced neuroimaging analysis workflows using libraries like Nibabel and Nilearn.