Chapter 7
Exploring Brain Imaging Analysis With Nilearn and Related Python Packages

Kadir Uludag
https://orcid.org/0000-0003-3713-4670
Shanghai Jiao Tong University Mental Health Center, China

ABSTRACT

The utilization of brain imaging information holds promise for predicting the prognosis of diverse psychiatric disorders, including schizophrenia, anxiety, attention-deficit hyperactivity disorder, bipolar disorder, and depression. Researchers are displaying growing interest in harnessing Python applications like Nilearn, Nistats, and Nibabel packages to extract and analyze brain imaging data. The Python Nilearn package enables the decoding, analysis, and creation of predictive models using brain imaging data. With Nilearn, users can perform functional connectivity analysis, manipulate brain image volumes, and conduct advanced statistical research on brain images. Furthermore, Nilearn contributes to the development of standardized methods for extracting brain imaging information, addressing potential variations in imaging methodologies across studies. While there is limited literature available on the Nilearn package and similar tools used for similar purposes, it is crucial to promote Nilearn and similar software to facilitate comparisons between studiess.

1. INTRODUCTION

Nilearn, a Python package developed by the nipy community, is a powerful tool for extracting and analyzing brain imaging data. It was created by individuals associated with the INRIA Parietal Project. Nilearn offers various functions that

DOI: 10.4018/979-8-3693-3840-7.ch007

enable the manipulation, processing, and transformation of brain imaging data. It is particularly useful during the preprocessing stage of functional magnetic resonance imaging (fMRI), where essential steps are taken to organize and standardize the data before conducting statistical analysis. Researchers often create preprocessing workflows using Nilearn to streamline these procedures (Esteban et al., 2019). Over time, an increasing number of studies examining the application of fMRI emerged, captivating the interest of many researchers who were intrigued by the potential of analyzing brain imaging data. However, a minority of critics argued that brain imaging tools were no more beneficial than traditional methods in comprehending the nature of neurological and psychiatric disorders.

Furthermore, within the nipy community, there are several additional Python packages such as dipy, mindboggle, nibabel, nitime, popeye, and niwidgets that can enhance brain imaging analysis. These packages contribute to the overall functionality and performance of Nilearn. The objective of our study was to provide an introduction to utilizing the Nilearn Python Package.

1.1. Methods

1.1.1.Running Sample Nilearn Codes

In order to execute Nilearn package code using Python programming, there are various interfaces available, including Jupyter Lab, Jupyter Notebook, Spyder, and PyCharm. However, if these applications are not accessible, Nilearn samples can still be run using the Mybinder application. Additionally, Jupyter nbviewer allows for the viewing of example codes from the Nilearn package, as mentioned in the official documentation on the Nilearn website (nilearn.github.io). Moreover, human neuroimaging studies typically involve a substantial number of subjects (Cui et al., 2015), and processing neuroimaging data may require heavy software (Huntenburg et al., 2017). Furthermore, utilizing the Nilearn Package necessitates a fundamental understanding of Python programming and familiarity with brain anatomy. Nilearn offers a command-line interface that facilitates the retrieval of atlases (Lawrence et al., 2021). Moreover, the Nilearn package provides the capability to import datasets as illustrative examples. It also offers modules that can be utilized to retrieve sample data for practicing brain imaging processing, such as importing a cortical atlas.

Python's modules consist of bundled classes, objects, functions, and constants employed in various tasks. The installation process involved utilizing pip to install both NumPy and its dependencies, including SciPy, the required libraries (Lemenkova, 2019). We have included seven Nilearn-related code snippets. We have presented a progression of codes, starting from simpler ones and gradually advancing to more complex ones. Subsequently, we have provided a comprehensive explanation of the