Analysis of Functional Magnetic Resonance Imaging in Python

In this article, we describe a package for analyzing magnetic resonance imaging (MRI) and functional MRI (fMRI) data, which is part of the Neuroimaging in Python (NIPY) project. An international group of leading statisticians, physicists, programmers, and neuroimaging methodologists are developing NIPY for wider use.

Magnetic resonance imaging (MRI) measures induced magnetic properties of tissue. It has long been the chosen technique for creating high-resolution anatomical images of the human brain. Over the past decade, a new technique called *functional MRI* (fMRI) has become a powerful and widely used method for studying human brain function. fMRI measures regional blood flow changes in the brain, which can help researchers identify the most active brain areas during mental tasks such as memory and language.

Functional MRI Analysis

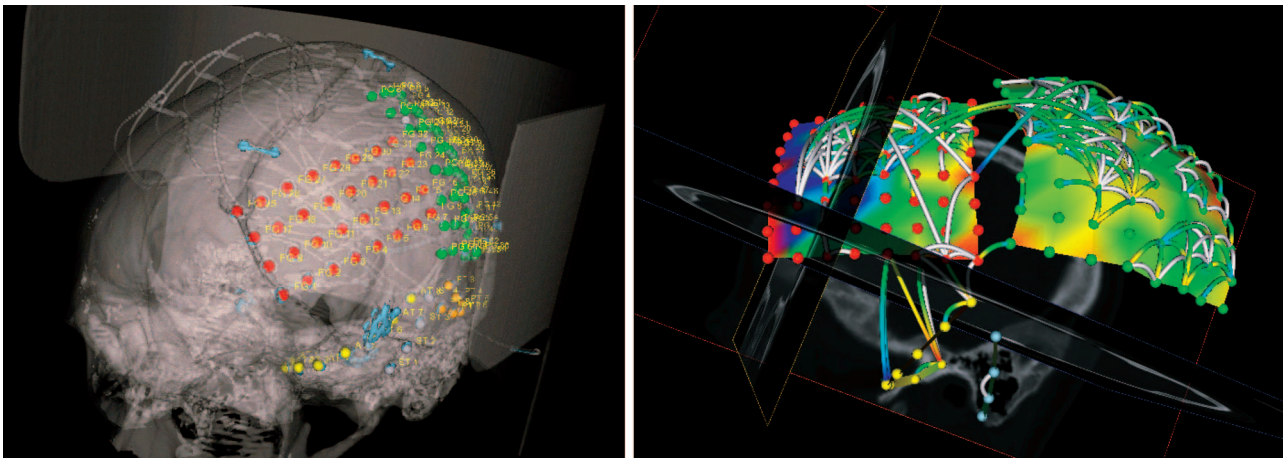
The first step of an fMRI analysis—image reconstruction—takes raw data from the scanner and performs a highly customized inverse Fourier transform to create a time series of 3D functional images. A typical next step is to estimate the movement between scans via an automated image-matching algorithm and then use that estimate to remove artifacts due to motion. Researchers commonly relate the activity detected in the low-resolution functional images to a high-resolution structural image of the same subject. However, to compare between subjects, the functional or structural data must be warped to match some standard brain, a process that requires sophisticated models of brain anatomy. Finally, statistical techniques are used to determine which brain regions are related to certain tasks or activities.

Clearly, fMRI data analysis comes with several challenges. First, it has a wide variety of computationally intensive spatial and statistical processing steps. Thus, an analysis software package must cover the range from file system and network interaction through complex image processing to advanced statistical inference and 3D visualization. Second, such analysis involves a massive volume of data, often reaching several hundred gigabytes.

The most common software package in use today is SPM (www.fil.ion.ucl.ac.uk/spm/), which is written in Matlab. Other common packages include FSL (www.fmrib.ox.ac.uk/fsl/) and AFNI (http://afni.nimh.nih.gov/afni/), written in C and C++. Although these packages are well-designed and supported, an increasing number of imaging scientists have found that Matlab is not powerful enough to support the industrial level of code size and complexity that neuroimaging requires, and that C and C++ are too low-level for rapid development.

1521-9615/07/$25.00 © 2007 IEEE
Copublished by the IEEE CS and the AIP

K. Jarrod Millman
University of California, Berkeley
Matthew Brett
MRC Cognitive Brain Science Unit, Cambridge, UK



(a) (b)

Figure 1. Python visualization widgets. Images from PBrain showing, (a) the position of the electrode arrays on the surface of the brain superimposed with a surface reconstruction of the skull from a CT and (b) measures of frequency and coherence of electrical activity overlaid on an image of the electrode positions.

Neuroimaging and Python

Python has become a natural choice for neuroimaging because it is high level, object-oriented, and interactive. All these features make it particularly well suited to scientific programming. It also has robust libraries for system interaction, image processing, matrix algebra, and visualization. Moreover, Python has very good tools for providing scripting support to software written in other languages. Finally, because of Python's strong integration with C and C++, it is often used as a type of high-level language glue for calling routines in a huge array of high-quality C/C++ libraries.

Accordingly, several significant neuroimaging packages have already been written in Python. Led by Daniel Sheltraw at Berkeley, researchers recently developed a set of Python tools for MRI reconstruction that provides specialized corrections for the artifacts and geometric distortions associated with fMRI (https://cirl.berkeley.edu/view/BIC/ReconTools). John Hunter, the author of the matplotlib Python plotting library, developed PBrain, a sophisticated application for analyzing and visualizing the data from electrical recordings on the brain surface of epileptic patients (see Figure 1).[5] Finally, BrainVISA is a comprehensive pipeline analysis tool for anatomical data developed by a team of researchers in Orsay, France (www.brainvisa.info).

Integrating Python Development in Neuroimaging

In this article, our main focus is on NIPY (http://neuroimaging.scipy.org), a new initiative to create a unified, open source, and open development environment for the analysis of neuroimaging data.[2] In particular, we will focus on the FMRI component of NIPY, which is based on theBrainSTAT package that Jonathan Taylor wrote at Stanford University.[3] We are also working with John Hunter and researchers at the University of Chicago to better integrate PBrain into the NIPY framework.

Let's look more closely at NIPY's basic image model, data diagnostic tools, and statistical analysis capabilities.

Image Model

An MRI scanner produces images that represent slices of brain tissue, and several of these slices together constitute a 3D whole brain scan. The values in each voxel (volume element) in a 3D image slice represent measurements from a small volume of the brain.

A major problem in neuroimaging is that different analysis packages and scanners use different output image formats that are not readily compatible. To address this, NIPY provides read and write capabilities for all the popular image formats in neuroimaging, as well as access to binary data images and Python arrays in memory. Images can be compressed upon read or write, loaded from an arbitrary URL (with local caching), and managed by memory mapping where possible. Images also have iterators for reading data in slices and other subsets; Python iterators offer an elegant mechanism for constructing lazily instantiated sequential data structures, a perfect abstraction for intuitively representing sequential data (such as time series, spatial slices,

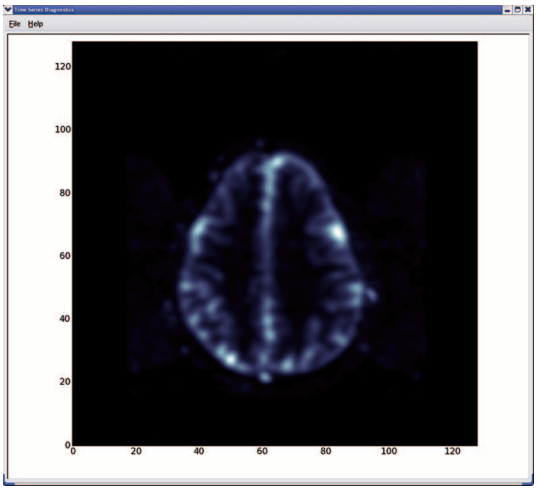


Figure 2. Mean image. By summarizing data, the mean image can highlight obvious problems with the overall brain shape or discover the source of severe background noise.

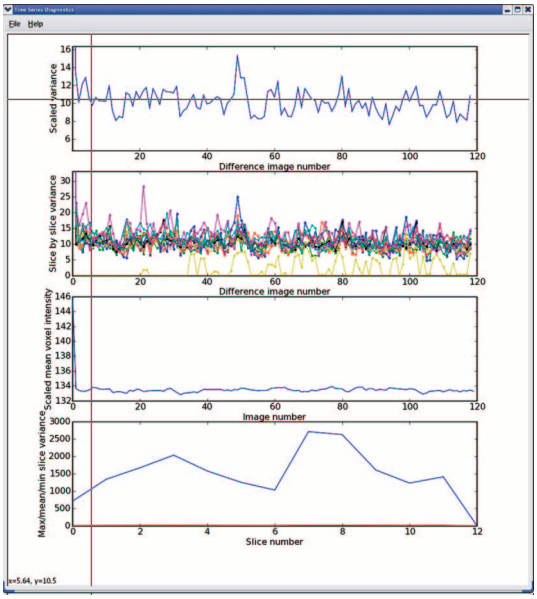


Figure 3. Time-series diagnostics. Four plots that can help researchers diagnose potential problems in the time-series data.

or generally $(n - 1) - d$ data bricks from an $n - d$ data set) without sacrificing memory efficiency.

Spatial transforms on images are fundamental to neuroimaging—for movement correction, warping to templates, and many other analysis steps. NIPY offers a general model of image spatial transforms that allow arbitrary combinations of linear and nonlinear transforms, volume-to-surface mappings, and flexible levels of image interpolation. A common analysis technique is to analyze the signal from a restricted region of the brain, or *region of interest* (ROI). NIPY has several ROI objects and functions, including discrete (point-level) and continuous (sphere, ellipse) region definitions, region combination algebra, and region data extraction.

Data Diagnostics

Because the scanner's signal quality can vary from day to day, special attention is needed to ensure good data. Currently, NIPY offers several diagnostic tools to let researchers discover problems with their data before they analyze it.

Images that summarize data across a time series provide a straightforward way to examine fMRI data. Taking an image of the mean signal intensity across time, for example, can highlight obvious problems with the overall brain shape or the sources of severe background noise that can occur with acquisition or reconstruction errors (see Figure 2).

Figure 3 shows four plots that can help diagnose potential problems in the time series. The top plot displays the scaled variance from image to image; the second plot shows the scaled variance per slice; the third plot shows the scaled mean voxel intensity for each image; and the bottom one plots the maximum, mean, and minimum scaled slice variances per slice.

Another powerful technique for data diagnosis is principal components analysis (PCA), which is particularly useful for detecting outlying time points or unexpected sources of spatially coherent noise. PCA on an image time series takes the time points as rows and voxels as columns, and decomposes the data into components that can efficiently express the source of variance in the data. Each component consists of a characteristic time series and the voxel weights contributing to that component, and these weights are viewable as an image. Figure 4 shows images of the weights for the first four principal components of a functional data set.

Statistical Processing

Subjects in a typical fMRI experiment perform some task or receive stimuli while being scanned, thus areas activated by the task or stimulus exhibit voxel time series correlated with the experimental design. Unfortunately, noise in fMRI analysis can come from multiple sources, and the signal is relatively low. This problem with signal detection has led to several standard and more complex statistical methods.[4] Partly for historical reasons, current analysis packages use nonstandard or low-level statistical terminology and interfaces, making them less accessible to scientists with general training in statistics.

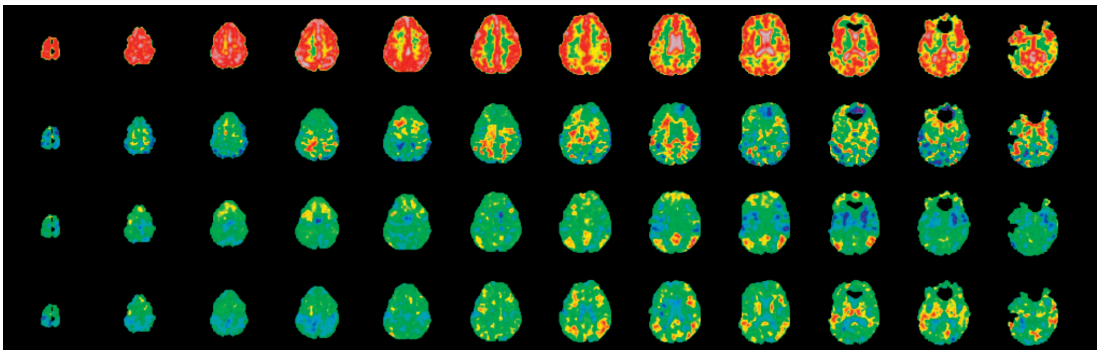


Figure 4. Principal components analysis (PCA). This type of analysis is particularly useful for detecting outlying time points or unexpected sources of spatially coherent noise.

At Stanford, Taylor developed a general set of statistical model objects in Python that use standard statistical terminology and allow flexible high-level statistical designs. Similar to the R statistical language, these objects implement a general series of statistical models and contrasts for data, including functional images. These models form the basis of NIPY statistical analysis and are now maintained as part of the SciPy distribution; we hope these will attract further development from researchers outside brain imaging.

Python's power and generality mean that we have many fruitful directions in which to take NIPY. At the most basic, its high-level language features make it much easier to refactor the code into a well-patterned, high-level interface that scientific developers can quickly pick up. Because Python has such good integration with C/C++, we also have ready access to very powerful visualization and image-processing libraries. Two very important examples are the Visualization ToolKit (VTK) and the Insight ToolKit (ITK). VTK provides high-quality 3D graphical display and has Python wrappers as part of its standard distribution, whereas ITK is a library of image registration and segmentation routines originally developed for the Visible Human Project. Like VTK, Python wrappers are part of the standard ITK distribution.

We intend NIPY to become the standard analysis library in neuroimaging in the medium term, which means we will need to provide the ability to call routines in other packages that are more familiar to researchers. Thankfully, this is a much easier task in Python than in many other languages because of its ability to interact with languages such as C and Matlab.

Python's language features can also help us tackle the problem of provenance tracking. Because imaging analyses and data sets are very diverse, researchers use a variety of analysis packages and rarely record all their data and analysis parameters, making it very difficult for other people to reproduce a published analysis. Fortunately, Python has excellent support for metaprogramming techniques (including metaclasses and function decorators) that can transparently change object behavior. We can thus use these techniques with Python's object introspection to capture the data's nature and processing in a way that can be closely wedded to the analysis.[1] Ultimately, this means that the analysis can become self-documenting, making it far easier to reproduce. CISE

References

1. K.J. Millman and M. D'Esposito, "Data and Analysis Management for Functional Magnetic Resonance Imaging Studies," *Proc. Int'l Advanced Database Conf.*, M. Amin et al., eds., US Education Service, 2006, pp. 24–28.
2. J.E. Taylor et al., "BrainPy: An Open Source Environment for the Analysis and Visualization of Human Brain Data," *Neuroimage*, vol. 26, no. 763, 2005, pp. T–AM.
3. J.E. Taylor and K.J. Worsley, "Inference for Magnitudes and Delays of Responses in the FIAC Data Using BRAINSTAT/FMRISTAT," *Human Brain Mapping*, vol. 27, no. 5, 2006, pp. 434–441.
4. A.W. Toga and J.C. Mazziotta, eds., *Brain Mapping: The Methods*, 2nd ed., Elsevier Science, 2002.
5. J.D. Hunter et al., "Locating chronically implanted subdural electrodes using surface reconstruction," *Clinical Neurophysiology*, vol. 116, no.8, 2005, pp. 1984–7.

K. Jarrod Millman *is the director of computing for the Helen Wills Neuroscience Institute at the University of California, Berkeley. His research interests include functional brain imaging, informatics, configuration management, and computer security. Millman has a BA in mathematics and computer science from Cornell University. Contact him at millman@berkeley.edu.*

Matthew Brett *is a senior investigator scientist at the Medical Research Council (MRC) Cognition and Brain Science Unit in Cambridge, UK. His research interests include functional brain imaging and localization of brain function. Brett has an MD from Cambridge University. Contact him at matthew.brett@gmail.com.*