MiTfAT: A Python-based Analysis Tool for Molecular fMRI Experiments.

Vahid S. Bokharaie[*1]

1 Max Planck Institute for Biological Cybernetics, Tuebingen, Germany

DOI: 10.21105/joss.02827

Software

- Review
- Repository
- Archive

Editor: Ariel Rokem

Reviewers:

- @JonathanReardon
- @emdupre

Submitted: 23 October 2020
Published: 17 February 2021

Summary

Functional Magnetic Resonance Imaging, fMRI, is a technique used in neuroscience to measure brain activity based on any signal that can be measured in an MRI scanner. Normally, fMRI is used to detect changes associated with blood flow, but it can also be used to detect changes in concentrations of molecules with different magnetic properties that might be directly injected into the brain of a subject.

Regardless of the signal that is measured in fMRI recordings, from a computational point of view, fMRI recordings will result in a number of time-series. And then those time-series should be analyzed to find the answers to various questions of interest. The length of the time-series depends on the number of time-steps in which we have measured the signals, and their number depends on how-many voxels we have measured (a voxel is a 3-dimensional pixel or the unit of volume in which each fMRI signal is measured). The size of each of these voxels depends on the magnetic flux density of the MRI scanner, measured in Tesla (T). The higher the magnetic flux density, the smaller the voxels can be, and the higher is the spatial resolution of the measurements. Hence, we end up with one time-series for each of the voxels arranged in a 3-dimensional structure. One characteristic of the fMRI measurements is that while they can have a high spatial resolution, and while they allow us to measure brain activity not only in the cortex but also in deeper regions of the brain, their temporal resolution is not normally high. And that can provide challenges for researchers who need to analyze the fMRI data.

Statement of need

`MiTfAT` is a Python library to analyze fMRI data, with a focus on molecular fMRI experiments. It was primarily developed for the study which is presented in (Savic et al., 2019).

There are already a few Python packages that are used by researchers to pre-process the fMRI time-series and then analyze them, for example (Esteban et al., 2019), and (Kent & Herholz, 2019), which focus on very specific points of the analysis workflow. Even a more comprehensive library such as NiLearn (Abraham et al., 2014), which includes various visualization functionalities and machine learning tools to analyze fMRI data, does not provide a ready-made framework to contain various information and measurements obtained in molecular fMRI experiments. Therefore, the `MiTfAT` library was developed. It can be used for general fMRI time-series analysis, but in particular for signals obtained from molecular fMRI studies, i.e., the cases in which we measure the changes in concentration of molecules that might have been directly injected into the brain. The `MiTfAT` library incorporates all the information and data related to an experiment into a Python class object called `fmri_dataset`. And various attributes of this class can be used to identify, analyze and visualize the data related to each

*Corresponding Author.

experiment. Such datasets can include various MRI measurements of the same subject, for example, T1-weighted and FISP signals measured almost simultaneously. In addition, the dataset can include many trials in which the same set of stimuli is presented or applied to a subject repeatedly.

The basic principle behind `MiTfAT` is that it imports all the relevant data of an fMRI experiment into an object/class of type `fmri_dataset`. Each set of fMRI time-series is a member of this class and is stored as a NumPy array. When the data is loaded into the `fmri_dataset` object, the raw values of the time-series are normalized (to the maximum value of all voxels) and the library uses the normalized values for further analyses unless the user explicitly specifies that the raw values should be used. Also, mean and linear approximations are also saved in separate members of the class. If the time-span of the recording is divided into a number of segments (representing different experimental conditions), mean and linear approximations are performed over each segment independently.

There are various functionalities available to analyze and visualize the data in a number of ways. They include:

- Clustering the time-series using K-means clustering. Clustering can be done based on raw or normalized values in all time-steps, or the mean value of each time-series or slope of a linear-regression passing through the time-series. If the time-span of the recording is divided into a number of segments, clustering can be done on each segment or the combination of the values in the segments. `MiTfAT` uses `scikit-learn` (Pedregosa et al., 2011) for machine learning functionalities. Hence, K-means can be easily replaced with any other clustering algorithm implemented in `scikit-learn`.
- Removing voxels with a low signal to noise ratio. This is done using a clustering algorithm in two stages. In the first stage, the algorithm removes the time-series corresponding to voxels in which signal to noise ratio is not high enough. And in the second stage, the time-series corresponding to the remaining voxels are clustered to identify the distribution pattern of the contrast agent.
- Detrending. We can remove the general trends in time-series to make the transient changes more visible. As an example, if we cannot wait long enough until the concentration of our agent reaches the steady-state level, then transient variations in the signal caused by changes in experimental conditions, which is what we are interested in, might be obscured by such trends in the signal. `MiTfAT` can detrend the time-series and give us a signal which looks like the one we might expect in a steady-state condition. And then, transient changes due to experimental design could be quantifiable.
- Interpolation of time-series under varying conditions. Assume we have an experiment in which the total recording time is divided into 3 segments. In the second segment, we record under a different condition than the first and third segments. This can be, as an example, the occlusion of an artery that changes calcium concentration in the brain when the contrast agent we have injected into the brain binds with calcium and we want to quantify how the signal has changed in the second segment. In order to do so, we should interpolate the time-series in segment 2 based on values of segments 1 and 3, and then compare it with the actual measurements during the second segment. `MiTfAT` provides such a functionality out of the box.
- Averaging over many trials. If we repeat an experiment many times, then it is usually of interest to average the measurements over all the trials. This can be useful if each measurement is noisy and we want to attenuate the effects of noise by averaging. `MiTfAT` provides such functionality.
- Visualization. The `fmri_dataset` class includes various visualization functions. They include functions that can plot the raw, normalized, averaged, or linear approximations

of the time-series in each voxel independently or as part of a plot representing all voxels in each layer of the fMRI mask. Also, various functions to plot centroids resulting from clustering algorithms and also plots to show how voxels corresponding to each cluster are placed in the overall mask. Such plots can be quite informative when we want to study the diffusion of a contrast agent in the brain.

Examples

`MiTfAT` repository includes a manual that contains many examples of the various capabilities of the library. It can be found here.

There are also two scripts in the `tests` folder of the repository that can be found here, accompanied with sample datasets, which you can run to see sample outputs of the library.

References

Abraham, A., Pedregosa, F., Eickenberg, M., Gervais, P., Mueller, A., Kossaifi, J., Gramfort, A., Thirion, B., & Varoquaux, G. (2014). Machine learning for neuroimaging with scikit-learn. *Frontiers in Neuroinformatics*, *8*, 14. https://doi.org/10.3389/fninf.2014.00014

Esteban, O., Wright, J., Markiewicz, C. J., Thompson, W. H., Goncalves, M., Ciric, R., Blair, R. W., Feingold, F., Rokem, A., Ghosh, S., & others. (2019). *NiPreps: Enabling the division of labor in neuroimaging beyond fMRIPrep*. https://doi.org/10.31219/osf.io/ujxp6

Kent, J. D., & Herholz, P. (2019). NiBetaSeries: Task related correlations in fMRI. *Journal of Open Source Software*, *4*(41), 1295. https://doi.org/10.21105/joss.01295

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., & others. (2011). Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, *12*, 2825–2830. https://doi.org/10.5555/1953048.2078195

Savic, T., Gambino, G., Bokharaie, V. S., Noori, H. R., Logothetis, N. K., & Angelovski, G. (2019). Early detection and monitoring of cerebral ischemia using calcium-responsive MRI probes. *Proceedings of the National Academy of Sciences*, *116*(41), 20666–20671. https://doi.org/10.1073/pnas.1908503116