MNE-Connectivity: An Electrophysiology Connectivity Package in the MNE Environment

Sub-Organization Information:

MNE-Python

Student Information:

Name: Adam Li
Time zone: EST
Source control username: http://github.com/adam2392/
Blog: https://adam2392.github.io/categories.html
GSoC Blog RSS feed: https://adam2392.github.io/categories.html

Code Samples:
https://github.com/mne-tools/mne-python/pull/7667
https://github.com/mne-tools/mne-python/pull/7768
https://github.com/mne-tools/mne-python/pull/8176
https://github.com/mne-tools/mne-python/pull/8402
https://github.com/mne-tools/mne-python/pull/8460
https://github.com/mne-tools/mne-python/pull/9087
https://github.com/mne-tools/mne-bids/pull/446

University Information:

University: Johns Hopkins University
Major: Biomedical Engineering
Current Year and Expected Graduation Date: 5th year and December 2021
Degree: PhD

Project Proposal Information:

Proposal Title:

MNE-Python (Connectivity): Neural Connectivity with MNE-Python with a Consistent API

Proposal Abstract:

The main objective of the project is to build a consistent API leveraging MNE-Python to perform connectivity analysis of neural time-series data, such as MEG, EEG and iEEG. The result would be “mne-connectivity” a Python3.6+ software package that allows scientists to explore connectivity based analysis using published algorithms with few barriers because it leverages the widely used open-source platform of MNE-Python.

Detailed Description:

Introduction:

Causality in the form of a graph structure, and dynamical networked systems are difficult to infer from electrophysiological data. For example, this can mean correlations with consistent delays in time that estimate the connectivity structure between brain regions. This type of analysis allows the neuroscientist to make inferences about which brain regions form interconnected networks and conduct experiments to determine the role of these putative networks in behavior. There is thus a need to have a connectivity software package that is implemented in Python for the MEG, EEG and iEEG neuroscience research community. Over the past few decades, neuroimaging, EEG and iEEG have been increasingly used to analyze connectivity patterns in the brain (Friston 2011; Bassett and Sporns 2017). Connectivity analysis generally consists of defining coupling functions on some transformation of the data. This can be done in the time, or frequency domain of the signal. For example, one might be interested in the Pearson correlation between EEG electrodes, or the coherence between EEG electrodes. Moreover, it can be defined as a linear, or nonlinear coupling function, and in addition, a dynamic or static function. For example, one might be interested in linear dynamical system representation vs static correlation graphs.

There are existing packages that perform different connectivity computations, such as estimating the connectivity structure, or performing downstream statistical tasks on top of the connectivity. There are existing packages for graph computation (networkx), and graph statistics (graspologic). In terms of dynamic connectivity, this can be represented in the realm of multivariate time-series models, or linear systems in dynamical systems models. Vector autoregressive models (VAR) are available through the open-source package, statsmodels. In MNE-Python, there are existing ways to compute connectivity from neural time series data,

such as with the `mne.connectivity` sub-module, or source-connectivity toolbox (i.e. scot). However, there are limitations in software maintenance, API consistency and also up-to-date algorithms that users will look for. These issues lead to adoption issues because users will look for a variety of different algorithms that are robustly tested packaged with a consistent API. This project will create a new package that can leverage these up-stream up-to-date software packages to allow for a consistent API to go from MEG/EEG/iEEG data to connectivity analyses. By providing this package, MNE-Python adoption will increase and connectivity analyses within the MNE framework will be streamlined.

Existing code - MNE-Python contains a mne.connectivity submodule, but presents difficulties in maintenance due to the size and current goals of the repository. There are only four existing algorithms implemented in this sub-module. In addition, according to the core-developers, the sub-module is not actively developed and maintained. Rather than having connectivity be a sub-module within MNE-Python, this project proposes i) refactoring the mne.connectivity sub-module into its own repository and ii) implementing additional modern connectivity algorithms that are useful to users.

Scot is a more general usage toolbox, but the last release was in 2016 and does not support Python3.6+ yet. However, there is definitely interest in this type of project with 45 stars and 17 forks. Some of the developers of Scot have expressed interest in collaborating on a community-driven effort that leverage MNE as the platform. This project would use existing implementations in Scot, but rewrite and consolidate these algorithms to be MNE compliant. Permission for adaptation from the authors of the original code has been obtained. Statsmodels, and networkx are BSD licensed, and graspologic and Scot are MIT licensed, which all facilitate usage in a downstream open-source package.

Dynamical Connectivity Improvements - In addition to static connectivity measures in the time-domain (i.e. Pearson correlation) and frequency-domain (i.e. coherence), this project aims at adding dynamical connectivity algorithms that are based on Vector Auto-Regressive (VAR) models in time-series, Dynamic Mode Decomposition (DMD) and linear dynamical systems theory. Generally, these models are generative of the form:

$$x(t+1) = Ax(t)$$

Where we will have some state matrix, A, that governs how signals evolve over time and each component in this matrix will tell us how each EEG electrode at a previous time point affects another EEG electrode at a future time point in the dynamical sense. Compared to existing static connectivity implementations in Scot, these connectivity measures fall under the category of “dynamic” (Ju and Bassett 2020). Dynamical models (i.e. VAR models) can be implemented

by leveraging the open-source software packages: statsmodels. This will leverage upstream bug-fixes and algorithmic improvements.

Downstream Tasks (Statistical Analysis and Graph Analysis) - By creating a new "mne-connectivity" repository that handles all connectivity related analysis steps, we can also leverage optional dependencies on 3rd party packages, such as networkx and graspologic that are not desirable within the MNE-Python repository. These packages are robust in computing downstream graph statistics and performing statistical inference tasks given the connectivity structure. This will also leverage upstream bug-fixes and algorithmic improvements that result in a more robust connectivity package for neural time series.

API

The API will be documented with Sphinx in the style of MNE and syntax will look something like the following:

First, we need the raw data imported via MNE-Python. This can be for example, a raw EEG dataset:

```
>>> from mne.io import read_raw_edf
>>> raw_fname = './sub-01_eeg.edf'
>>> raw = read_raw_edf(raw_fname)
```

Then, we can compute various measures of connectivity. Perhaps we will compute spectral connectivity (i.e. coherence) between EEG electrodes over a sliding window of time of 5 seconds.

```
>>> from mne_connectivity.spectral import compute_coherence
>>> conn = compute_coherence(raw, win_size=5)
```

Alternatively, one might be interested in estimating a linear dynamical system from the data (this is equivalent to a VAR model of order 1; one can estimate a VAR model of higher order also). That is, a model of the form:

$$x(t+1) = Ax(t)$$

Or for general VAR models

$$x(t+1) = \sum_{i=0}^{p-1} A_i x(t-i)$$

Where x(t) represents EEG electrode activity at time point t. Then we might use a sliding window of 2 seconds and use a least-squares estimation procedure in the backend to estimate an order-1 model.

```
>>> from mne_connectivity.var import compute_var
>>> conn = compute_var(raw, method='least-squares', order=1, win_size=2)
```

At the end, one will have arrays of the computation that represents their corresponding connectivity pattern. To facilitate usage of the computation in downstream usage that is in-line with MNE-Python, this project will implement a general-usage connectivity data structure that at its core represents: "channels X channels", possibly over time. By integrating the connectivity visualization module inside "mne.viz", or "pyvista", we could possibly generate figures, such as:

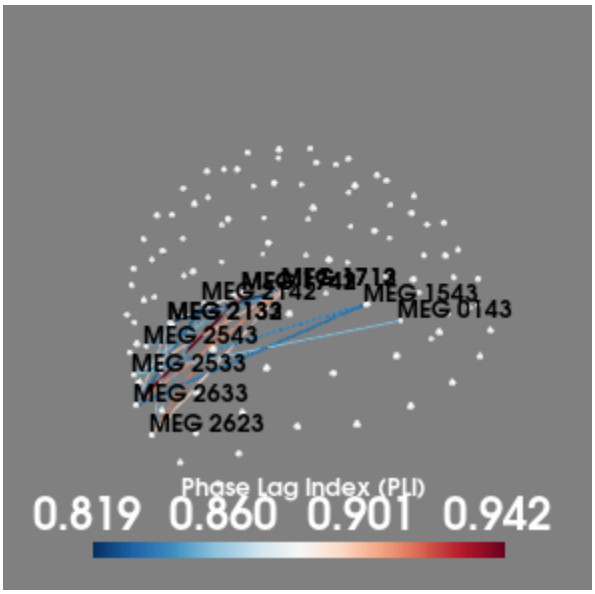

Figure 1: An example of connectivity structure, in the form of a phase lag index (PLI) being visualized on MEG sensors. PLI could be computed using "mne_connectivity.spectral.compute_phase_lag_index".

Implementation Details:

Some notes on potential implementation roadblocks and challenges along with potential solutions are outlined below:

- **Computing the connectivity:** Since there are many ways to compute connectivity from neural time series data, this project will approximately group these semantically under a "time" and "frequency" domain connectivities. In addition, there will be options to compute dynamical connectivity structures (such as the linear dynamical system, or VAR model).
- **Performing post-hoc network analysis:** Once the connectivity structure is estimated from the data via some algorithm exposed via the API, then post-hoc network analysis can be performed by requiring "networkx", or "graspologic" dependency. Currently,

these are very actively developed repositories, so relying on them to perform analysis on top of the MNE connectivity structure can be done by wrapping calls to their API.

- **Writing functional tests:** Since the connectivity API needs to be refactored from existing code that is not maintained, or updated recently, then tests need to apply the latest MNE-Python and Python3.6+ features.
- **Writing documentation that explicitly demonstrates metrics with pros and cons for various methods**: There are a multitude of ways to compute connectivity, but it is still not 100% clear what are the advantages of different approaches based on different properties of the dataset. The documentation will attempt to be as clear as possible, by operating on a select few datasets to demonstrate varying behavior of different connectivity algorithms. This will allow the user to understand pros and cons in the context of their own dataset. For example, if we have an EEG dataset with sampling rate of 200 Hz, then we most likely cannot compute spectral connectivity (i.e. coherence) in the high gamma frequency band (90-200 Hz), and so would instead opt for a time-domain connectivity measure, such as Pearson correlation.
- **Generic MEG/EEG/iEEG connectivity container:** Since mne-connectivity will utilize MNE-Python's data structures as a platform, then this project can easily create a generic connectivity container that is useful for the user in terms of plotting, doing post-hoc statistics, or saving to a file. The inheritance from MNE's Info object would look like this:

  ```
  >>> conn = Connectivity(connectivity_array, info)
  ```

  Where info, is the MNE Info object. Thus one can easily perform actions such as:

  ```
  >>> conn.ch_names  # get the channel names associated with the connectivity
  >>> conn.get_montage()  # get the montage of the channels
  >>> conn.info['sfreq']  # get the original sampling frequency of the data
  ```

  This conn object would be returned by *all* functions that compute connectivity, to provide a consistent data structure object that extends what users are familiar with in MNE-Python to connectivity.

Proposal Timeline:

The goal is the development of the "mne-connectivity" repository.

June 7, 2021 - June 11, 2021

- Analyze what is up-to-date in `mne.connectivity` sub-module and general missing areas of interest, such as “dynamical system” connectivity modeling.
- Generate a plan for:
    - Refactoring existing open-source algorithms under the `mne-connectivity` module
    - Refactoring the entire `mne.connectivity` sub-module to this new repository that is maintained.
- Begin refactoring `mne.connectivity` into a new repository “mne-connectivity” with continuous integration, unit tests and example datasets
    - In order to validate that refactoring was successful, i) the unit tests must be constructed exactly as it is done in MNE-Python’s `mne.connectivity` sub-module and ii) using sample (real) data, demonstrate that the two implementations are identical.

June 14, 2021 - June 18, 2021

- Finish refactoring mne.connectivity code into a new repository
    - API goals:
        - mne_connectivity.spectral_connectivity
        - mne_connectivity.phase_slope_index
        - mne_connectivity.post.compute_degree
    - Validate that unit tests pass and existing documentation is successfully built on the new repository
- Begin refactoring “scot” existing algorithms into mne-connectivity.

June 21, 2021 - June 25, 2021

- Finish refactoring scot code into mne-connectivity. Although not all code is desirable, the scot package does contain code for VAR models, Partial coherence models, Directed transfer function and cross spectral density.
    - In order to validate that refactoring was successful, i) the unit tests must be constructed exactly as it is done in scot and ii) using sample (real) data, demonstrate that the two implementations are identical for the functions that are desired.
- API goals:
    - Mne_connectivity.var
    - Mne_connectivity.partial_coh
    - Mne_connectivity.directed_transfer_fun
    - mne_connectivity.cross_spectral_density

June 28, 2021 - July 2, 2021

- Implement dynamic connectivity representations based on dynamical systems
- API goal: mne_connectivity.linear_dynamical_system

July 5, 2021 - July 9, 2021

- Incorporate mne.viz and pyvista functionality to visualize different connectivity structures on the MEG/EEG/iEEG sensors
- API goal: mne_connectivity.viz

July 12, 2021 - July 16, 2021

- Implement a light-weight data structure that inherits MNE's info object and stores the connectivity array
    - API goal: mne_connectivity.Connectivity data structure that stores connectivity arrays of channels X channels (optionally over time) and inherits MNE's `Info` object

July 19, 2021 - July 23, 2021

- Write tests for core functionality and algorithms

July 26, 2021 - July 30, 2021

- Finish tests to reach 90+% coverage

August 2, 2021 - August 6, 2021

- Write examples and documentation demonstrating usage of API on datasets from openneuro

August 9, 2021 - August 16, 2021

- Write examples and documentation demonstrating usage of API on datasets from openneuro

Conclusion

Overall, this project is well-posed and is expected to be finished in the allotted time span. Having a consistent API that is compliant with MNE data structures and preprocessing will i) increase the user base of MNE and ii) enable faster connectivity analysis of neural time series, powered by open source software. Moreover, leveraging MNE as a platform, this project can create light-weight generic containers for connectivity data from MEG/EEG/iEEG data. Once connectivity structures are estimated from data, this project can expose downstream graph analysis and statistical inference using robust 3rd party packages, such as "networkx" and "graspologic". In addition, by depending on downstream visualization packages, such as "pyvista", this project can easily generate connectivity visualizations overlaid on a brain. These features will enable researchers to quickly explore any type of connectivity they want and help generate scientific hypotheses about the brain and functions of its networks.

References

Bassett, Danielle S., and Olaf Sporns. 2017. "Network Neuroscience." *Nature Neuroscience* 20 (3): 353–64.

Friston, Karl J. 2011. "Functional and Effective Connectivity: A Review." *Brain Connectivity* 1 (1): 13–36.

Ju, Harang, and Danielle S. Bassett. 2020. "Dynamic Representations in Networked Neural Systems." *Nature Neuroscience* 23 (8): 908–17.