On Using Python to Run, Analyze, and Decode EEG Experiments

Colin Conrad*, Om Agarwal, Carlos Calix Woc, Tazmin Chiles, Daniel Godfrey, Kavita Krueger, Valentina Marini, Alexander Sproul and Aaron Newman

Dalhousie University, Halifax, Canada
*colin.conrad@dal.ca

Abstract. As the NeuroIS field expands its scope to address more complex research questions with electroencephalography (EEG), there is greater need for EEG analysis capabilities that are relatively easy to implement and adapt to different protocols, while at the same time providing an open and standardized approach. We present a series of open source tools, based on the Python programming language, which are designed to facilitate the development of open and collaborative EEG reserach. As supplementary material, we demonstrate the implementation of these tools in a NeuroIS case study and provide files that can be adapted by others for NeuroIS EEG research.

Keywords: Research methods · Python · Machine learning · Open science · Brain-computer interface

1 Introduction

There has been considerable interest recently in the information systems community concerning tools for conducting interdisciplinary experiments. The motivation for this interest is rooted in the growing acceptance that explicit measures of emotional and cognitive states can capture dimensions of technology use and human computer interaction that users would not be able to self-report [1,2]. To address this concern, NeuroIS researchers have called for tools that integrate disparate physiological and questionnaire measures [3]. This has led to efforts in the NeuroIS community to develop tools as either proprietary platforms [4] or open source libraries [5]. While these efforts have considerable potential for streamlining NeuroIS research processes in the long run, there has so far been relatively little discussion about existing tools developed in other research communities that might provide validated building blocks that accelerate progress in NeuroIS without reduplication of effort, and at the same time allowing NeuroIS to more fully integrate with related research enterprises such as cognitive neuroscience. Beyond simply being able to implement NeuroIS research in as efficient and validated a manner as possible, there is growing recognition across scientific disciplines that research processes and results be open, transparent, and fully reproducible [6-8]. Our goal is to describe a software pipeline that enables this.

In this paper, we introduce and demonstrate the utility of a set of open source tools that are relevant to many NeuroIS research projects—particularly those which use EEG to validate elements of user experience, but extending to other physiological measures including eye tracking, heart rate, skin conductance, MEG, and near-infrared brain imaging (fNIRI). All of the tools discussed are written in the Python programming language, are readily inter-operable, are freely available under the Python Software Foundation's open source license. The result is a set of tools that are both powerful and flexible, that can also be adapted to extend beyond traditional EEG analysis. We illustrate the use these tools, which we will henceforth refer to as the "Python stack", through a brief case study provided online as a supplement to this paper. The case study uses the Python stack to build elements of a P3 speller brain computer interface (BCI) [9,10] and includes data collected in this paradigm as part of a neurotechnology hackathon at Dalhousie University, Halifax, Canada in 2019. This application was chosen because of its applicability to attention-related IS constructs that have been described in the IS literature [11-14]. The source files for this case and the related tutorial are provided publicly and can be retrieved from GitHub at `https://github.com/cdconrad/py-bci`.

2 Python tools for experiment design and EEG processing

Table 1 lists the Python packages that comprise the stack we use in our experimental protocols. The base of this is Anaconda [15], a collection of Python packages designed for scientific computing, which come bunded with a "package manager": a tool for installing and updating packages that ensures that all are compatible and interoperable with each other. The value of Anaconda is that by downloading and installing this single package, the user is readily equipped with a wide variety of Python packages that will work together, without the overhead of identifying the necessary set of packages for a task and resolving compatibility issues.

The second tool highlighted in Table 1 is Jupyter [16]. This is a scientific "notebook" application which allows the user to write and execute code, view and save the results, and write rich-text documentation using Markdown formatting, all in a single file that is accessed via a Web browser. This has significant advantages over other approaches to using Python or other programming languages, such as interacting with a command line or using an integrated development environment; because all elements of the process are encapsulated in a single file, it is very easy to share and reproduce analysis pipelines across experiments and between labs.

Another advantage of Jupyter notebooks is that, once a pipeline has been implemented in a notebook, e.g., for the processing of EEG data from an individual participant, notebooks can simply be copied and re-run for each additional participant, and/or easily adapted to new experiments. This means that while proficiency in the Python language is necessary to build the pipelines in the first place, users little to no programming expertise can readily adapt and run these notebooks for new participants or groups of participants. This makes these an excellent entry point for new researchers who wish to engage in NeuroIS research without first learning Python programming—not only because the learning curve is less steep, but because the notebook format makes it easy for senior lab members to audit others' work to ensure quality control. In this regard it is notable that our lab moved to this pipeline several years ago from the Matlab-based EEGlab platform, which is also widely used in cognitive neuroscience research [17]. While EEGlab offers a menu-driven graphical user interface (GUI), users have to choose the appropriate menu items and manually enter the appropriate parameters each time, and these settings are not all recorded in the output—making the process both more error-prone and more difficult to audit.

Table 1. Description of the recommended stack of Python tools for EEG analysis

Tool Name	Developers	Description
Anaconda	Anaconda Inc. [15]	A distribution of the Python programming language for scientific computing.
Jupyter	Pérez et al. [16]	A notebook format for sharing code and computational narratives.
Matplotlib	Hunter et al. [18]	A 2D graphics package for the creation of publication-quality images.
NumPy	van der Walt et al. [19]	A library for scientific computing and analysis.
Pandas	McWinney et al. [20]	A data library optimized for manipulating large and time series data.
PsychoPy	Peirce et al. [21]	An application and library used to run psychology and neuroscience experiments.
MNE-Python	Gramfort et al. [22]	A library for preparing, analyzing and visualizing MEG, EEG and other related data.
scikit-learn	Pedregosa et al. [23]	A machine learning library.

The other packages listed in Table 1 include a set of very widely-used tools for scientific computing (Matplotlib, NumPy, and Pandas), PsychoPy for experimental programming and data collection, MNE-Python (hereafter referred to as MNE) for EEG data preprocessing and analysis, and scikit-learn for machine learning. In what follows we describe the steps involved in running and analyzing the results of an EEG experiment using these packages. Note that the Matplotlib, NumPy, and Pandas packages are not explicitly described but are used by the tools that are described.

2.1 Experimental Protocol and Data Collection

An EEG experiment begins with presentation of stimuli to a participant, time-locked with collection of behavioral, EEG, and possibly other physiological measures. Software capable of precise time-locking is essential here, because measures such as EEG have temporal precision on the order of milliseconds. Some mode of inter-device communication is also required, because physiological data such as EEG is typically recorded on a separate device from that controlling stimulus presentation. The PsychoPy library [20] provides an environment for the presentation of a wide range of stimuli such as images, sounds, and movies, as well as collection of behavioral and vocal responses, and the ability to send precisely time-locked "trigger codes" to other hardware such as EEG data collection systems. These trigger codes are essential for later data analysis as they store, in the EEG data file, both the precise timing of events of experimental interest (e.g., stimulus onset, response times), and the identity of these events (e.g., stimulus type, correct vs. incorrect response). PsychoPy offers the ability to build experiments using either a GUI, which translates the user's design into Python code, or writing Python code directly. This again allows users with varying levels of expertise to participate fruitfully in the research enterprise. In addition to output sent to other devices, PsychoPy will save the precise timing of all events in the experiment to a text file for later analysis in any package the user desires.

2.2 Preprocessing EEG data in Python

Following data collection, EEG data must be preprocessed and analyzed. Preprocessing involves a number of steps designed to improve the signal-to-noise ratio of the data and increase the ability to detect experimental effects, if they are present. In our pipeline, EEG preprocessing and analysis are performed using MNE. MNE provides a collection of data reading and conversion utilities which can be used to import and prepare data from a variety of hardware systems, including most common EEG and MEG systems. MNE converts data into a `mne.raw` object which includes the raw timecourse data, time-locked trigger codes, and metadata such as participant ID, date and time of data collection, the labels for each data channel, etc.

Common preprocessing steps for EEG data [24,25] include: band-pass filtering; removal of data channels (electrodes) and trials contaminated with excessive noise; correction of other well-defined artifacts such as eye blinks, eye movements, and muscle noise; and re-referencing EEG data to an electrode(s) appropriate to the experiment. MNE provides functions dedicated to each of these tasks, which have been designed to implement best practices in EEG/MEG research (e.g., the choice of filter type). This relieves the user of the need to extensively research all of their preprocessing parameter choices yet allows—through the use of command-line options—control over common parameter choices (e.g., filter bandwidth). As data are processed, the data are converted from raw format (continuous EEG data) to MNE's epochs format (segments of data time-locked to experimental events of interest) and finally to MNE's evoked format (averages across all epochs of a given category).

While MNE is under active development, at this time it has a wide variety of tools implementing common functions in EEG preprocessing, such as independent components analysis (ICA) for artifact removal. The supplementary material for this paper includes Jupyter notebooks demonstrating our EEG preprocessing pipeline, including the specific MNE functions and associated parameters used, and documentation elaborating on usage and choice of parameters.

2.3 Analysis and classification

Finally, after preprocessing the data, users can visualize data at the individual or group level, perform analyses to determine if hypothesized effects are present, and/or attempt classification of data based on machine learning. MNE provides tools for the visualization of EEG/MEG data in the time and frequency domains, as both waveform plots at individual or clusters of channels, and as scalp topographic maps. It also includes algorithms for source localization, allowing visualization of data on the cortical surface. MNE also provides some tools for statistical analysis—including parametric (*t*-tests, linear regression) and non-parametric (*t*-test, clustering) approaches and methods for multiple comparison correction—and machine learning decoders. However, perhaps one of the most powerful features of the Python stack is the compatibility between the data formats and machine learning libraries; because MNE is built on the NumPy and Pandas packages, it is easy to convert MNE data to these packages' data objects. This allows the use of a wide variety of other packages in Python, such as scikit-learn, a widely-used package implementing a wide variety of machine learning tools. As well, MNE data objects can be readily exported for use in other statistical packages such as R.

3 Conclusion

As the NeuroIS discipline develops, there will be a greater need for knowledge transfer and open science. Python tools provide an open source alternative for NeuroIS researchers, which has the added benefit of being curated by the Psychology and Neuroscience community. We hope that this technology can be leveraged to benefit the NeuroIS discipline and advance the community's capabilities for EEG research.

References

1. de Guinea, A. O. and Webster, J.: An investigation of information systems use patterns: technological events as triggers, the effect of time, and consequences for performance. In: MIS Quarterly, vol. 37, iss. 4, pp. 1165-1188 (2013)
2. de Guinea, A. O., R. Titah, and P.-M. Léger: Explicit and implicit antecedents of users' behavioral beliefs in information systems: A neuropsychological investigation. In: Journal of Management Information Systems, vol. 30, iss. 4, pp. 179-210 (2014)
3. P.-M. Léger, F. Courtemanche, M. Fredette, and S. Sénécal: A cloud-based lab management and analytics software for triangulated human-centered research. In: Davis, F. D., Riedl, R., vom Brocke, J., Léger, P. M. and Randolph A. B. (eds.) Information Systems

and Neuroscience. Lecture Notes in Information Systems and Organisation, pp. 93-99. Springer International Publishing (2019)

4. Courtemanche, F., et al., Method of and system for processing signals sensed from a user, US Patent US230180035886A1 (2018).
5. Michalczyk, S., D. Jung, M. Nadj, M. T. Knierim, and R. Rissler: BrownieR: the R package for neuro information systems research. In: Davis, F. D., Riedl, R., vom Brocke, J., Léger, P. M. and Randolph A. B. (eds.) Information Systems and Neuroscience. Lecture notes in Information Systems and Organisation, pp.101-109. Springer International Publishing (2019)
6. Baker, M.: Is There a Reproducibility Crisis? In Nature, vol. 533 (2016)
7. Toelch, U., & Ostwald, D.: Digital open science—Teaching digital tools for reproducible and transparent research. PLoS Biology, vol. 16, iss. 7, e2006022 (2018)
8. van der Aalst, W., Bichler, M., & Heinzl, A. Open research in business and information systems engineering. In: Business & Information Systems Engineering, vol. 58, uss. 6, pp. 375-379 (2016)
9. Farwell, L. A. and Donchin, E.: Talkiing off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. In: Electroencephalography and Clinical Neurophysiology, vol. 70, iss. 6, pp. 510-523 (1988)
10. Donchin, E., Spencer, K. M., and Wijesinghe, R.: The mental prosthesis: assessing the speed of a P300-based brain-computer interface. In: IEEE transactions on rehabilitation engineering, vol. 8, iss. 2, pp. 174-179 (2000)
11. Léger, P. M., Davis, F. D., Cronan, T. P., and Perret, J.: Neurophysiological correlates of cognitive absorption in an enactive training context. In: Computers in Human Behavior, vol. 34, pp. 273-283 (2014)
12. Conrad, C. D., and Bliemel, M.: Psychophysiological measures of cognitive absorption and cognitive load in e-learning applications. In ICIS 2016 Proceedings: 37th International Conference on Information Systems, December 11-14 2016, Dublin, Ireland (2016)
13. Conrad, C. and Newman, A.: Measuring the Impact of Mind Wandering in Real Time Using an Auditory Evoked Potential. In: Davis, F. D., Riedl, R., vom Brocke, J., Léger, P. M. and Randolph A. B. (eds.) Information Systems and Neuroscience. Lecture notes in Information Systems and Organisation, pp. 37-45. Springer International Publishing (2019)
14. Gwizdka, J.: Exploring Eye-Tracking Data for Detection of Mind-wandering on Web Tasks. In: Davis, F. D., Riedl, R., vom Brocke, J., Léger, P. M. and Randolph A. B. (eds.) Information Systems and Neuroscience. Lecture notes in Information Systems and Organisation, pp. 47-55. Springer International Publishing (2019)
15. Anaconda, Inc.: Anaconda Destruction: The World's Most Popular Python/R Data Science Platform. Retrieved from `https://www.anaconda.com/distribution/`
16. Perez, F. and Granger, B.: Project Jupyter: Computational Narratives as the Engine of Collaborative Data Science. Retrieved from: `http://archive.ipython.org`
17. Delorme, A. & Makeig, S.: EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. In: Journal of Neuroscience Methods vol. 134, no. 1, pp. 9–21 (2004)
18. Hunter, J. D.: Matplotlib: A 2D graphics environment. In: Computing in Science & Engineering, vol. 9, iss. 3, 90-95 (2007).
19. Van Der Walt, S., Colbert, S. C., & Varoquaux, G.: The NumPy array: a structure for efficient numerical computation. In: Computing in Science & Engineering, vol. 13, iss. 2, (2011)
20. McKinney, W.: Pandas: a foundational Python library for data analysis and statistics. In: Python for High Performance and Scientific Computing, vol. 14 (2011)

21. Peirce, J. W.: Psychopy—psychophysics software in Python. In: Journal of Neuroscience Methods, vol. 162, iss. 1-2, pp. 8-13 (2007)
22. Gramfort, A., Luessi, M., Larson, E., Engemann, D. A., Strohmeier, D., Brodbeck, C., Parkkonen, L. and Hämäläinen, M. S.: MNE software for processing MEG and EEG data. In: Neuroimage, vol. 86, pp. 446-460 (2014)
23. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine Learning in Python. In: Journal of Machien Learning Research, vol. 12 pp. 2825-2830 (2011)
24. Luck, S.: An introduction to the Event-Related Potential Technique, Second Edition. MIT Press (2014)
25. Newman, A.: Research Methods for Cognitive Neuroscience. SAGE Publications (2019)