




Openseize: A digital signal processing package for large EEG datasets in Python

Matthew S. Caudill ^{1,2}

¹ Department of Neuroscience, Baylor College of Medicine, Houston, TX, United States of America ² Jan and Dan Duncan Neurological Research Institute at Texas Childrens Hospital, Houston, TX, United States of America

DOI: [10.21105/joss.05126](https://doi.org/10.21105/joss.05126)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: Samuel Forbes  

Reviewers:

- [@szorowi1](#)
- [@AJQuinn](#)

Submitted: 20 December 2022

Published: 05 April 2023

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

Summary

Electroencephalography (EEG) is an indispensable clinical and research tool used to diagnose neurological disease (Davies & Gavin, 2007; Noachtar & Rémi, 2009; Tierney et al., 2012) and discover brain circuit mechanisms that support sensory, mnemonic and cognitive processing (Nuñez & Buño, 2021; Woodman, 2010). Mechanistically, EEGs are non-stationary time-series that capture alterations in the brain's electromagnetic field arising from synchronous synaptic potential changes across neuronal populations. Linear digital signal processing (DSP) tools are routinely used in EEGs to reduce noise, resample the data, remove artifacts, expose the data's spatio-temporal frequency content, and much more. Openseize is a DSP package written in pure Python that scales to very large EEG datasets, employs an extensible object-oriented architecture, and provides a familiar Scipy-like API (Virtanen et al., 2020).

Statement of need

Scalable

Current DSP software packages (Cole et al., 2019; Delorme & Makeig, 2004; Gramfort et al., 2013; Oostenveld et al., 2011; Tadel et al., 2011) make two critical assumptions. First, that the signals to be analyzed are addressable to virtual memory. Second, that the values returned from a DSP process, such as filtering, and all subsequent processes are likewise addressable to memory. Advances in recording technologies over the past decade are degrading these assumptions. Indeed, thin-film electronics innovations allow for the deposition of a large number of electrode contacts onto a single recording device that can be left implanted for months (Thongpang et al., 2011). These high-channel count long-duration recordings pose a serious challenge to imperatively programmed DSP software in which data is stored as it passes through and between functions within a program.

Openseize takes a declarative programming approach that allows for constant and tuneable memory overhead. Specifically, this approach shuttles iterables (called producers) rather than data between the functions within a program. These memory-efficient producers generate on-the-fly fragments of processed data. Importantly, all of Openseize's functions and methods accept and return producers. This feature allows for the composition of DSP functions into iterative processing pipelines (Figure 1) that yield processed data lazily during an iteration protocol such as a for-loop.

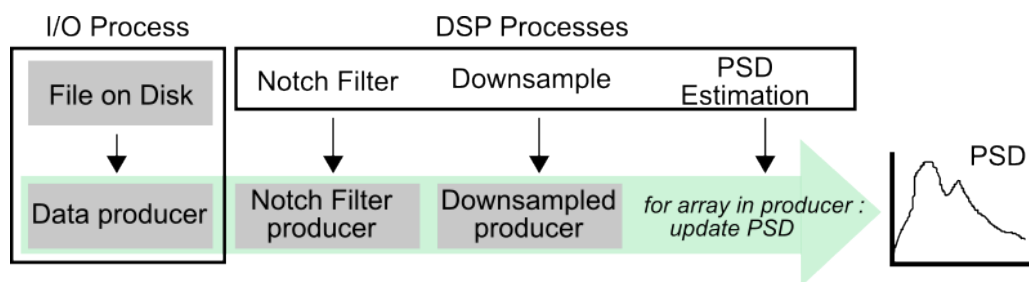


Figure 1: Example DSP pipeline for computing the power spectrum of a large EEG dataset. Each DSP process in the pipeline receives and returns a producer iterable. At the final stage, the power spectral density (PSD) estimator requests an array from the downsampled producer triggering all previous producers to generate a single array.

A consequence of this functional programming approach is that DSP pipelines in Openize, in contrast to pipelines relying on Matlab (MATLAB, 2010) and Scipy (Virtanen et al., 2020), are fully iterative. Restructuring DSP algorithms to support iterative processing is non-trivial because complex boundary conditions and the need to minimize data transfers between disks and virtual memory creates significant challenges. To meet these, Openize uses a first-in first-out (FIFO) queue data structure to cache arrays. FIFO caching allows previously seen data to influence the boundary conditions of in-process data and reduces the number of disk reads. This data structure in combination with both producers and iterative algorithms allows Openize to scale to massive data recordings.

Extensible

In addition to its scalability, Openize employs an extensible object-oriented architecture. This feature, missing in many currently available DSP packages, is crucial in neuroscience research for two reasons. First, there are many different data file types in-use. Abstract base classes (Gamma et al., 1994) help future developers integrate their file types into Openize by identifying required methods needed to create producers that Openize's algorithms can process. Second, DSP operations are strongly interdependent. By identifying and abstracting common methods, the algorithms in Openize are smaller, more maintainable and above all, easier to understand. Figure 2 diagrams the currently available DSP methods grouped by their abstract types or module names.

Readers/Writers	FIR Filters	IIR Filters	Resamplers
EDF EDF Header CSV	Kaiser Rectangular Hanning Hamming Bartlett Blackman Remez	Butterworth Chebyshev I Chebyshev II Elliptical Notch	Upsample Downsample Resample
Producers			Spectral Estimators
Reader Producer Array Producer Generator Producer Masked Producer			Periodogram Power Spectrum Short-time Fourier Transform

Figure 2: Partial list of DSP classes and methods available in Openize grouped by abstract type and/or module (gray boxes). Each gray box indicates a point of extensibility either through development of new concrete classes or functions within a module.

Intuitive API

Finally, Openize has an intuitive application programming interface (API). While under the hood, Openize is using a declarative programming approach, from the end-user's perspective,

the calling of its functions are similar to Scipy's DSP call signatures. The main difference is that producers do not return DSP processed values when created. Rather, the values are generated when the producer is iterated over. To help new users understand the implications of this, Openseize includes extensive in-depth discussions about DSP algorithms and their iterative implementations in a series of Jupyter notebooks (Kluyver et al., 2016). Importantly, to maintain the clarity and extensibility of Openseize's API, graphical user interfaces (GUIs) have been avoided. This decision reflects the fact that many current DSP packages have inconsistent APIs depending on whether the modules are invoked from the command-line or a GUI.

In summary, Openseize fulfills a need in neuroscience research for DSP tools that scale to large EEG recordings, are extensible enough to handle new data types and methods, and are accessible to both end-users and developers.

Acknowledgements

We thank Josh Baker for help in debugging and testing Openseize on real-world EEG data as well as critical reading of the manuscript. This work was generously supported through the Ting Tsung and Wei Fong Chao Foundation.

References

- Cole, S., Donoghue, T., Gao, R., & Voytek, B. (2019). NeuroDSP: A package for neural digital signal processing. *Journal of Open Source Software*, 4(36), 1272. <https://doi.org/10.21105/joss.01272>
- Davies, P. L., & Gavin, W. J. (2007). Validating the diagnosis of sensory processing disorders using EEG technology. *The American Journal of Occupational Therapy*, 61(2), 176–189. <https://doi.org/10.5014/ajot.61.2.176>
- Delorme, A., & Makeig, S. (2004). EEGLAB: Una caja de herramientas de código abierto para el análisis de la dinámica de EEG de un solo ensayo, incluido el análisis de componentes independientes. *J. Neurosci. Métodos*, 134, 9–21. <https://doi.org/10.1016/j.jneumeth.2003.10.009>
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: Elements of reusable object-oriented software* (B. W. Kernighan, Ed.). Addison-Wesley Professional Computing Series. ISBN: 978-0-201-63361-0
- Gramfort, A., Luessi, M., Larson, E., Engemann, D. A., Strohmeier, D., Brodbeck, C., Goj, R., Jas, M., Brooks, T., Parkkonen, L., & Hämäläinen, M. S. (2013). MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7(267), 1–13. <https://doi.org/10.3389/fnins.2013.00267>
- Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). *Jupyter notebooks – a publishing format for reproducible computational workflows* (F. Loizides & B. Schmidt, Eds.; pp. 87–90). IOS Press.
- MATLAB. (2010). *Version 7.10.0 (R2010a)*. The MathWorks Inc.
- Noachtar, S., & Rémi, J. (2009). The role of EEG in epilepsy: A critical review. *Epilepsy & Behavior*, 15(1), 22–33. <https://doi.org/10.1016/j.yebeh.2009.02.035>
- Nuñez, A., & Buño, W. (2021). The theta rhythm of the hippocampus: From neuronal and circuit mechanisms to behavior. *Frontiers in Cellular Neuroscience*, 15. <https://doi.org/10.3389/fncel.2021.649262>

- Oostenveld, R., Fries, P., Maris, E., Schoffelen, J., & others. (2011). *FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data, computational intelligence and neuroscience*. 2011; article ID 156869, 9 p. <https://doi.org/10.1155/2011/156869>
- Tadel, F., Baillet, S., Mosher, J., Pantazis, D., & Leahy, R. (2011). Computational intelligence and neuroscience, 2011. *Brainstorm: A User-Friendly Application for Meg/Eeg Analysis*, 8. <https://doi.org/10.1155/2011/879716>
- Thongpang, S., Richner, T. J., Brodnick, S. K., Schendel, A., Kim, J., Wilson, J. A., Hippensteel, J., Krugner-Higby, L., Moran, D., Ahmed, A. S., & others. (2011). A micro-electrocorticography platform and deployment strategies for chronic BCI applications. *Clinical EEG and Neuroscience*, 42(4), 259–265. <https://doi.org/10.1177/155005941104200412>
- Tierney, A. L., Gabard-Durnam, L., Vogel-Farley, V., Tager-Flusberg, H., & Nelson, C. A. (2012). Developmental trajectories of resting EEG power: An endophenotype of autism spectrum disorder. *PloS One*, 7(6), e39127. <https://doi.org/10.1371/journal.pone.0039127>
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>
- Woodman, G. F. (2010). A brief introduction to the use of event-related potentials in studies of perception and attention. *Attention, Perception, & Psychophysics*, 72(8), 2031–2046. <https://doi.org/10.3758/BF03196680>