MultiSim: A Python Toolbox for Simulating Datasets with time resolved multivariate effects

Alex Lepauvre[1,2¶], **Qian Chu**[1,3,4,5], **Lucia Melloni**[1,6,7], and **Peter Zeidman**[8]

1 Neural Circuits, Consciousness and Cognition Research Group, Max Planck Institute for Empirical Aesthetics, Frankfurt am Main, Germany **2** Donders Institute for Brain, Cognition and Behaviour, Radboud University Nijmegen, Nijmegen, The Netherlands **3** Max Planck – University of Toronto Centre for Neural Science and Technology **4** Krembil Brain Institute, Toronto Western Hospital, University Health Network, Toronto, ON, Canada **5** Institute of Biomedical Engineering, University of Toronto, Toronto, ON, Canada **6** Department of Neurology, New York University Grossman School of Medicine, New York, NY, USA **7** Predictive Brain Department, Research Center One Health Ruhr, University Alliance Ruhr, Faculty of Psychology, Ruhr University Bochum, Bochum, Germany **8** Wellcome Centre for Human Neuroimaging, Institute of Neurology, University College London, London, UK ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

In MEG/EEG research, validating analysis pipelines is hampered by the lack of ground-truth neural signals in real data. SimMEG fills this gap by generating realistic, time-locked multivariate effects of known magnitude that you can inject into simulated sensor data. You can then run any pipeline—e.g. decoding, sensor-level statistics, or source estimation—against these datasets to benchmark sensitivity and specificity.

Key benefits include:

- Testing whether your pipeline reliably detects effects of a chosen size.

- Providing demonstrable, reproducible benchmarks for reviewers or collaborators.

- Offering a controlled teaching environment for newcomers.

Below, we describe the rationale (Statement of needs), and the data-generation method (Methods), a hands-on example (Results), and potential extensions (Discussion).

Statement of needs

Multivariate pattern analysis (MVPA) is now routine in cognitive neuroscience for probing how the brain represents information {cite}`ritchie2019decoding;haynes2006decoding;kriegeskorte2008re` Applied to high-temporal-resolution electrophysiology signals such as electro and magneto-encephalography (EEG and MEG respectively), decoding techniques reveal the millisecond-by-millisecond unfolding of mental representations {cite}`cichy2014resolving;king2014characterizing;ki` Strinkingly, despite the ubiquity of MVPA techniques, to our knowledge, not method exists to tests the sensitivity and specifity of decoding analysis pipelines, nor to estimate, before data collection, how many trials and how many participants are required to detect an effect of a given size

MultiSim addresses this gap by letting investigators simulate time-resolved multi-channel signals with paramaters matching that of their recording setups, and specify multivariate effects with known timing, spatialization and strength, while controlling channel covariance, sensory noise and between subjects variability. Our algorithm produces multi-subject data sets in which ground truth effects are known. By running their pipeline on these data, researchers obtain a direct read-out of its true-positive rate (can it recover the injected effects?) and false-positive rate (does it raise alarms when nothing is present). In addition, our simulator can be used to perform computational power analysis, to determine the number of trials and subjects, by iterating over these parameters.

This toolbox promotes best-practice MVPA by giving researchers a tailored benchmark for their specific experimental designs, a testbed for developing new decoding methods, and a principled way to check that planned studies are properly powered—ultimately enabling more reliable and efficient investigations of brain function.

Code Quality and Documentation

SimMEG is hosted on GitHub. Examples and API documentation are available on the platform here. We provide installation guides, algorithm introductions, and examples of using the package with Jupyter Notebook. We further provide the full mathetmatical details of our simulation here. The package is available on Linux, macOS and Windows for Python >=3.12 It can be installed with pip install simMEG. To ensure high code quality, all implementations adhere to the PEP8 code style [REF], enforced by ruff [REF], the code formatter black and the static analyzer prospector. The documentation is provided through docstrings using the NumPy conventions and build using Sphinx.

Acknowledgements

References

Supplementary