The Journal of Open Source Software

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Published: unpublished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

MultiSim is a python package to simulate multivariate patterns in multi-channels time-resolved signals to emulate neural recordings signals (local field potentials, electro and magneto-encephalogram) based on custom experimental design and recording systems. Users can specify the time windows, temporal dynamics and size of the multivariate effects from their design they wish to simulate. The simulated data contain ground truth effects and can therefore be used to develop and benchmark multivariate analysis pipelines to establish their sensitivity and specifity. In addition, the toolbox can be used to perform power analysis, by varying the number of subjects and number of trials per subjects at fixed effect size and noise parameters to identify the optimal combination to ensure that their sample is properly powered.

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 (Haxby et al., 2001; Haynes & Rees, 2006; Kriegeskorte et al., 2008; Poldrack et al., 2009; Ritchie et al., 2019). 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 (Cichy et al., 2014; Consortium et al., 2025; King et al., 2016; King & Dehaene, 2014; Kok et al., 2017).

While several toolboxes can simulate EEG/MEG data—such as MNE-Python (Gramfort et al., 2013), FieldTrip (Oostenveld et al., 2011), Brainstorm (Tadel et al., 2011) and unfoldSim (Schepers et al., 2025)—these are typically designed to model univariate ERP components, source-level activity, or general sensor-level signals. Critically, none allows researchers to specify multivariate effects with controlled timing, spatial structure, and strength, nor to systematically manipulate noise, channel covariance, and between-subject variability. As a result, there is currently no standard method to test the sensitivity and specificity of decoding pipelines, or to estimate, in advance, the number of trials and participants required to detect effects of a given size.

MultiSim addresses this gap by letting investigators simulate time-resolved multi-channel signals tailored to their recording setups, embedding multivariate effects with known spatiotemporal properties while flexibly controlling signal and noise parameters. The core of our simulation engine builds on and extends a function from the SPM toolbox (see DEMO_CVA_RSA.m, Tierney et al., 2025), which we adapted to support dynamic time-resolved signals and to give users direct control over effect size specification.

Functionalities:

The code block below provides a minimal example, highlighting the simplicity with which multivariate effects can be specified with our toolbox (see Figure 1**A** for a visual representation of key parameters):

```
import numpy as np
from meeg_simulator import simulate_data
X = pd.DataFrame(np.random.randn(100, 1), columns=["face-object"]) # 100 trials, 1 exper
t_win = np.array([[0.2, 0.5]])  # Effect between 200-500 ms
effects = [
    {"condition": 'face-object',
      "windows": [0.1, 0.3],
      "effect_size": 0.5
     }
]
sims = Simulator(
   X, noise_std=0.1, n_channels=64, n_subjects=20,
   tmin=-0.2, tmax=0.8, sfreq=250,
   t_win=t_win, effects=effects
  )
sim.summary()  # Should return 20 subjects
```

Our algorithm produces multi-subject data sets in which ground truth effects are known with precise timiming (see Figure 1**B**). Furthermore, our pipelines enable full flexibility regarding the temporal dynamics of the effects (see see Figure 1**C**) as well as the temporal generalization of the injected effects (see Figure 1**D**), enabling the simulation of all patterns presented by (King & Dehaene, 2014) (Figure 2). By running custom analysis pipeline on simulated 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.

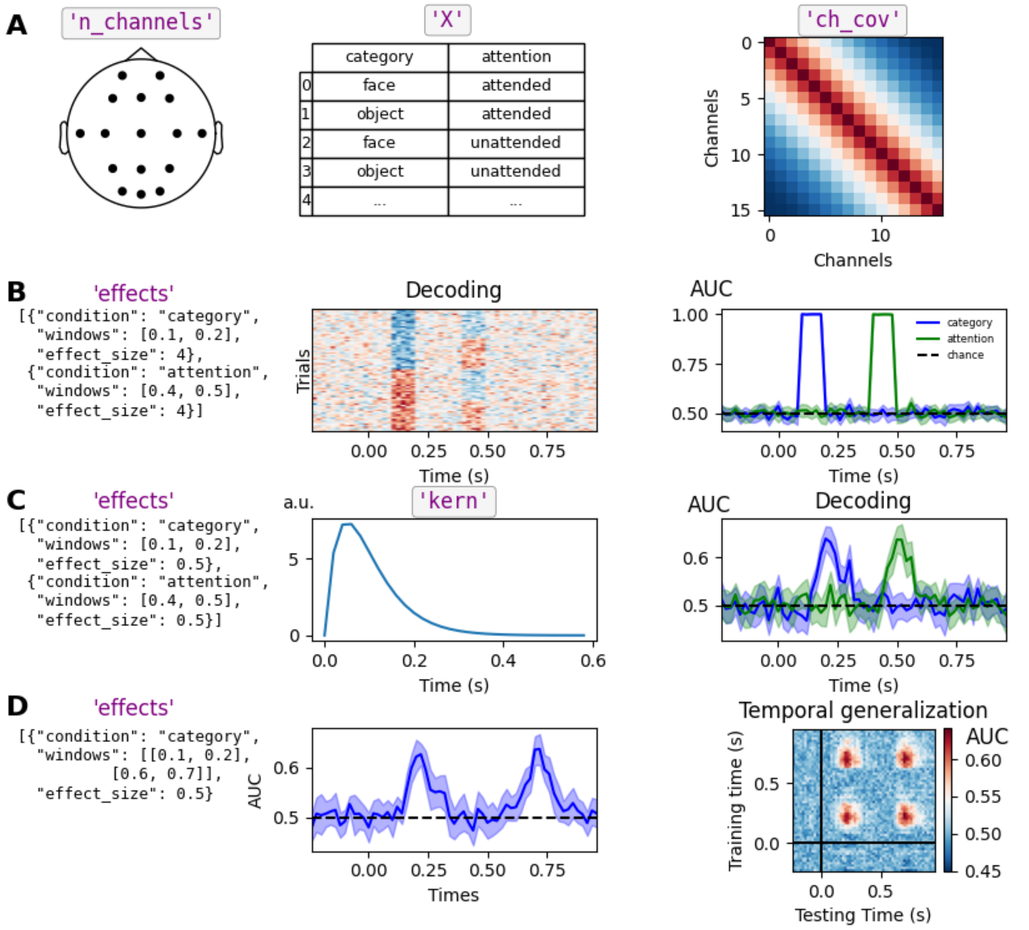

Figure 1: Figure 1. Overview of the simulation and decoding framework. A. General data parameters for the simulation. Left: `n_channels` corresponds to the number of channels in the montage, with 2 experimental conditions. Middle: `X` is the design matrix (each column is an experimental condition and each row a trial). Right: `ch_cov` is the channel by channel covariance matrix of the data to be simulated. **B**. Minimal simulation example with effects for each experimental condition with large effect size. Left: `effects` is a dictionary that specifies the `"condition"`, time window (`"windows"`) and effect size (`"effect_size"`) of each effect to simulate. The example specifies an effect of category from 0.1 and 0.2 s with an effect size of 4, and an effect of the attention condition from 0.4 to 0.5 with an effect size of 4. Middle: example of the activation of a single channel. Right: resulting decoding accuracy (using a Support vector machine classifier). **C**. Example of simulated effects with a an added gamma kernel to simulate effects with biologically plausible temporal dynamcs. Left: `effects` similar to that of B but with effect size of 0.5 for each condition. Middle: gamma `kernel`, specifying the temporal dynamics of the multivariate effect. Right: Resulting decoding accuracy. **D**. example of simulated data with cross temporal generalization of the category effect. Left: `effects` dictionary specifies two different time windows for the effect of category as a list. Middle: resulting decoding accuracy. Right: Temporal generalization of the decoding.

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 mathematical 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

`{bibliography}`

Supplementary

Cichy, R. M., Pantazis, D., & Oliva, A. (2014). Resolving human object recognition in space and time. *Nature Neuroscience*, *17*(3), 455–462.

Consortium, C., Ferrante, O., Gorska-Klimowska, U., Henin, S., Hirschhorn, R., Khalaf, A., Lepauvre, A., Liu, L., Richter, D., Vidal, Y., & others. (2025). Adversarial testing of global neuronal workspace and integrated information theories of consciousness. *Nature*, 1–10.

Gramfort, A., Luessi, M., Larson, E., Engemann, D. A., Strohmeier, D., Brodbeck, C., Goj, R., Jas, M., Brooks, T., Parkkonen, L., & others. (2013). MEG and EEG data analysis with MNE-python. *Frontiers in Neuroinformatics*, *7*, 267.

Haxby, J. V., Gobbini, M. I., Furey, M. L., Ishai, A., Schouten, J. L., & Pietrini, P. (2001). Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, *293*(5539), 2425–2430.

Haynes, J.-D., & Rees, G. (2006). Decoding mental states from brain activity in humans. *Nature Reviews Neuroscience*, *7*(7), 523–534.

King, J.-R., & Dehaene, S. (2014). Characterizing the dynamics of mental representations: The temporal generalization method. *Trends in Cognitive Sciences*, *18*(4), 203–210.

King, J.-R., Pescetelli, N., & Dehaene, S. (2016). Brain mechanisms underlying the brief maintenance of seen and unseen sensory information. *Neuron*, *92*(5), 1122–1134.

Kok, P., Mostert, P., & De Lange, F. P. (2017). Prior expectations induce prestimulus sensory templates. *Proceedings of the National Academy of Sciences*, *114*(39), 10473–10478.

Kriegeskorte, N., Mur, M., & Bandettini, P. A. (2008). Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, *2*, 249.

Oostenveld, R., Fries, P., Maris, E., & Schoffelen, J.-M. (2011). FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data. *Computational Intelligence and Neuroscience*, *2011*(1), 156869.

Poldrack, R. A., Halchenko, Y. O., & Hanson, S. J. (2009). Decoding the large-scale structure of brain function by classifying mental states across individuals. *Psychological Science*, *20*(11), 1364–1372.

Ritchie, J. B., Kaplan, D. M., & Klein, C. (2019). Decoding the brain: Neural representation and the limits of multivariate pattern analysis in cognitive neuroscience. *The British Journal for the Philosophy of Science*.

Schepers, J., Lips, L., Marathe, M., & Ehinger, B. V. (2025). UnfoldSim. Jl: Simulating continuous event-based time series data for EEG and beyond. *Journal of Open Source Software*, *10*(107), 6641.

Tadel, F., Baillet, S., Mosher, J. C., Pantazis, D., & Leahy, R. M. (2011). Brainstorm: A user-friendly application for MEG/EEG analysis. *Computational Intelligence and Neuroscience*, *2011*(1), 879716.

Tierney, T. M., Alexander, N. A., Avila, N. L., Balbastre, Y., Barnes, G., Bezsudnova, Y., Brudfors, M., Eckstein, K., Flandin, G., Friston, K., & others. (2025). SPM 25: Open source neuroimaging analysis software. *arXiv Preprint arXiv:2501.12081*.