The Journal of Open Source Software

fitgrid: A Python package for multi-channel event-related time series regression modeling

Thomas P. Urbach[*1] **and Andrey S. Portnoy**[2]

1 Department of Cognitive Science, University of California, San Diego **2** Cerebras Systems

DOI: DOIunavailable

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Reviewers:

- @Pending Reviewers

Submitted: N/A
Published: N/A

Summary

Electroencephalography (EEG) and magnetoencephalography (MEG) are non-invasive human brain imaging modalities with millisecond temporal resolution that track the time course of electro-magnetic fields generated by the synchronous activity of populations of neurons as the brain dynamically processes information. A central challenge for research is that the fields systematically related to events under experimental control (signals) are generally small relative to much larger fields generated by unrelated ongoing brain activity (noise), often differing by one or more orders of magnitude.

In the early 1950s Dawson demonstrated that brain responses to sensory stimulation that were too small to see in wideband EEG could be detected by aligning several recordings with the delivery of the stimulus and averaging the values at each point in time (Dawson, 1951, 1954). As the average accumulates, irregular fluctuations (noise) in the individual recordings tends to cancel out and reveal the time-course of the much smaller systematic stimulus-related response (signal). Dawson attributed the suggestion to J. N. Hunt, noting that while the application to EEG recordings was novel, the approach was generally well-known and an adaptation of Laplace's (unsuccessful) 19th century attempt to isolate small but regular lunar atmospheric tides among the much larger irregular pressure fluctuations (Lindzen & Chapman, 1969). Hunt-Dawson time-domain averaging was a breakthrough. In subsequent decades, laboratory experiments showed that whereas EEG is typically around 25 - 75 μV peak to peak, for n on the order of 10 to 1000, sweeping a sum-and-divide-by-n averager across the time-aligned EEG recordings revealed transient and steady-state brain responses down to fractions of a microvolt over a wide range of frequencies before, during, and after sensory stimulation, motor responses, and internal mental events. Oscilloscopic sweep averagers gave way to multichannel analog-to-digital data acquisition and sum-and-divide averaging in software on general purpose computers. Since the 1970s, the resulting *discrete time-domain average event-related brain potential* or ERP has been a cornerstone of experimental brain research on human sensation, perception, and cognition (Luck & Kappenman, 2013).

In a seminal paper, Smith and Kutas noted that the average of a set of values, y, is identical to the estimated constant, $\hat{\beta}_0$ for the linear model, $y = \beta_0 + e$, fit by minimizing squared error (Smith & Kutas, 2015a). They pointed out that this makes the sum-and-divide time-domain average ERP at time, t, mathematically identical to $\hat{\beta}_0(t)$ and a special case of sweeping a linear regression model along the EEG. Generalizing to more complex models, e.g., multiple regression, $y(t) = \beta_0 + \beta_1 X_1 \ldots \beta_p X_p + e$, likewise produces time series of estimates for the intercept and each regressor coefficient, the $\hat{\beta}_0(t), \hat{\beta}_1(t), \ldots, \hat{\beta}_p(t)$ they dubbed regression ERP (rERP) waveforms. This holds for straight-line fits ("slope" rERPs) as well as models of curvilinear relationship, e.g., via spline regression (Smith & Kutas, 2015b). Still more generally, the approach produces time series for all the basic and derived quantities of the fitted model: parameter estimates and their standard errors, residuals, residual error, likelihood, Akiake's information criterion (AI), and so forth.

*corresponding author, turbach@ucsd.edu

The insight that sum-and-divide averaging is special case of regression modeling encourages a shift in perspective from Laplace-Hunt-Dawson "needle in a haystack" signal detection to the general framework of applied regression for fitting, evaluating, and comparing a range of models to account for systematic variation in the time course of EEG responses. With this shift, however, comes a new problem.

Statement of need

Fitting a regression model is relatively straightforward on any current scientific computing platform. Informative modeling, by contrast, is a laborious process that iterates cycles of data quality control, fitting, data diagnosis, model evaluation, comparison, selection, and interpretation with numerous decision points that require thought and judgment.

Modeling digitized multichannel EEG data as regression ERPs at each time point and data channel multiplies the iterative cycles in a combinatorial explosion of times $\times$ channels $\times$ models $\times$ comparisons. For instance, at a digital sampling rate of 250 samples per second, in 3 seconds of 32-channel EEG data there are 24,000 data sets ($= 3 \times 250 \times 32$). To fit three candidate models requires 72,000 separate model fits, where the size of each data set might range anywhere from a few dozens of observations for a single subject to tens of thousands of observations for a large scale experiment. Nothing can prevent the combinatorial explosion; `fitgrid` is designed to contain it.

The rERP approach is attracting growing attention though to date, the open-source EEG and MEG data analysis landscape has been dominated by toolboxes written for MATLAB such as EEGLAB (Delorme & Makeig, 2004), FieldTrip (Oostenveld et al., 2011), and Brainstorm (Tadel et al., 2011), and this holds for rERP modeling, e.g., Ehinger & Dimigen (2019). Like open-source scientific computing generally, Python and R have been gaining traction for EEG and MEG analysis, as in MNE Python Gramfort et al. (2013) and for regression ERPs in R, Tremblay & Newman (2015). Nevertheless, widely accessible implementations for rERP modeling in the Python remain limited. Development of N. J. Smith's promising rERPy Python package for ERP and rERP analysis appears to have halted at Python 2.6 or 2.7. MNE Python implements a `linear_regression` function for computing rERP coefficients on continuous data as described in Smith & Kutas (2015b) but not time series of OLS or mixed-effects model fits.

fitgrid

`fitgrid` is intended to fill this gap and make rERP modeling as in Smith & Kutas (2015a) accessible to researchers with a working knowledge of scripted data analysis in Python and the symbolic formulae such as $\sim 1 + a + b + a : b$ and $\sim 1 + a * b + (a|s) + (a|i)$ deriving from S (Becker & Chambers, 1984; Chambers & Hastie, 1991) and currently in wide use to specify ordinary and mixed-effects models in Python and R (`patsy` Smith, 2011-2015; `lme4::lmer` Bates et al., 2015; `lm` R Core Team, 2020). The `fitgrid` user interface launches what are routinely hundreds to tens of thousands of model fits with one line of code (computed in parallel if supported by hardware). The fit results across times and channels are available on demand with the same syntax used to access results in a single fit object and the results are returned as tidy indexed `pandas.DataFrames` for further analysis, visualization, and interpretation. While the origins of `fitgrid` are in EEG data analysis, `fitgrid` can also be used with other neuroimaging data such as MEG and more generally with synchronized sensor array time-series data from other domains where event-related regression modeling is appropriate. `fitgrid` enables researchers to conduct this type of computationally intensive modeling flexibly, efficiently, informatively, and reproducibly with familiar scientific computing tools and minimal programming. These features make `fitgrid` well-suited for general use in exploratory data analysis (EDA), e.g., Urbach et al. (2020) and Troyer et al. (2020).

fitgrid TL; DR

Documentation

The `fitgrid` documentation is available online: https://kutaslab.github.io/fitgrid-dev.

- Getting Started gives an overview of the `fitgrid` workflow with notes, figures, and downloadable and executable code.
- The User Guide provides information about usage and specific topics including how the OLS models are fit in Python `statsmodels` (Seabold & Perktold, 2010) and the LMER models are fit in R (`lme4::lmer`, `lmerTest` Kuznetsova et al., 2017) via `pymer4` (Jolly, 2018).
- The API Reference is a complete listing of `fitgrid` classes, methods, attributes, and functions auto-generated with numpy-style docstrings and links to the source code generated by `sphinx-apidoc` (Brandl & others, 2007-2021).
- The Bibliography includes references to relevant experimental and technical literature.
- The Examples Gallery contains `fitgrid` vignettes with simulated data, experimental EEG recordings, and NOAA tide and atmospheric observations that can be downloaded as executable Python scripts or Jupyter notebooks thanks to sphinx-gallery.

  Downloadable Examples Gallery

Installation, Continuous Integration, and Source

The online documentation includes installation instructions and system recommendations. The latest stable release of `fitgrid` and the bleeding edge pre-release development version are packaged for Python 3.6, 3.7, and 3.8 on x86_64 linux and distributed on Anaconda Cloud. The stable release is installed into a fresh conda virtual environment (recommended) along with other compatible packages, if desired, e.g., `jupyter` for running Example Gallery notebooks, like so:

```
$ conda create --name fg_env \
    -c kutaslab -c defaults -c conda-forge \
    fitgrid jupyter
```

`fitgrid` is developed and tested locally on a high-performance 48-core x86_64 CentOS 7 server. Continuous integration (github Actions) for the latest stable release on the main code branch runs nightly. The action runs conda build, conda install, and package pytests on Ubuntu 18.04. Pre-release packages also pass CI conda build, conda install, and pytests before deployment to Anaconda Cloud. The Python 3.6, 3.7 and 3.8 64-bit Intel OSX and Windows packages are also distributed on Anaconda Conda and the Python sdist is uploaded to PyPI but these are not routinely tested (contributed field reports are welcome). The source code is hosted in a public github repository https://github.com/kutaslab/fitgrid-dev) and Issues may be posted there in accordance with the `fitgrid` Code of Conduct.

How it works

TODO ...

Limitations

The `fitgrid` framework can fit OLS models of curvilinear relations between predictors and EEG via formulas because `patsy` supports column variable transformation by arbitrary Python code. Polynomial regression ERPs for U-shaped relations can be computed with, e.g.,

$x + \text{np.power}(x, 2)$ if this seems like a good idea. If spline regression as described in Smith & Kutas (2015b) seems like a better idea, `patsy` also provides built-in functions for generating families of spline bases, although the researcher is responsible for ensuring the data epochs are appropriately mapped to the spline regression variables which may require additional programming. Smith & Kutas (2015b) also generalize the rERP framework from iteratively fitting epochs of length L to continuous data and use it to model temporally overlapping responses. For a design with P predictor variables, this conceptually elegent approach unstacks the L times of the epoch into L $\times$ P predictor variables and codes the values as zeros or non-zeros according to the value of the predictor at each time. The coefficients estimated by a single OLS fit are identical to segmenting the data and fitting models with the P predictors iteratively at each of the L time points. In principle `fitgrid` can ingest and fit the continuous data prepared for the wide $L \times P$ design matrix as a corner case of single-sample epochs but it is not a natural act. The data format with indices for multiple epochs and multiple times is baked into into `fitgrid` implementation, including the parallel processing acceleration. Models are fit separately at each channel and time by design in order to compute and track the timecourse of all the fit attributes, not just the regression coefficients. For fitting the wide L $\times$ P models to continuous data, implementations specifically designed for that purpose such as the implementation in MNE Python may be better suited.

Acknowledgements

We gratefully acknowledge contributions to prototypes by Lauren Liao and testing and feedback during development by Melissa Troyer, Anna Stoermann, and Emily Provenzano. This work was supported by grant NICHD 5R01HD022614 to Marta Kutas.

References

Bates, D., Mächler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4 [Journal Article]. *2015*, *67*(1), 48. https://doi.org/10.18637/jss.v067.i01

Becker, R. A., & Chambers, J. M. (1984). *S: An interactive environment for data analysis and graphics*. CRC Press.

Brandl, G., & others. (2007-2021). *Sphinx - python documentation generator*. https://www.sphinx-doc.org/

Chambers, J. M., & Hastie, T. J. (1991). *Statistical models in s*. CRC Press, Inc.

Dawson, G. D. (1951). A summation technique for detecting small signals in a large irregular background [Journal Article]. *Journal of Physiology*, *115*(1), P2–P3.

Dawson, G. D. (1954). A summation technique for the detection of small evoked potentials [Journal Article]. *Electroencephalography and Clinical Neurophysiology*, *6*(1), 65–84. https://doi.org/10.1016/0013-4694(54)90007-3

Delorme, A., & Makeig, S. (2004). EEGLAB: An open-source toolbox for analysis of single-trial EEG dynamics. *Journal of Neuroscience Methods*, *134*, 9–21.

Ehinger, B. V., & Dimigen, O. (2019). Unfold: An integrated toolbox for overlap correction, non-linear modeling, and regression-based EEG analysis. *PeerJ*, *7*, e7838.

Gramfort, A., Luessi, M., Larson, E., Engemann, D., Strohmeier, D., Brodbeck, C., Goj, R., Jas, M., Brooks, T., Parkkonen, L., & Hämäläinen, M. (2013). MEG and EEG data analysis with MNE-python. *Frontiers in Neuroscience*, *7*, 267. https://doi.org/10.3389/fnins.2013.00267

Jolly, E. (2018). Pymer4: Connecting r and python for linear mixed modeling [Journal Article]. *Journal of Open Source Software*, *3*(31). https://doi.org/10.21105/joss.00862

Kuznetsova, A., Brockhoff, P. B., & Christensen, R. H. B. (2017). lmerTest package: Tests in linear mixed effects models [Journal Article]. *Journal of Statistical Software*, *82*(13), 1–26. https://doi.org/10.18637/jss.v082.i13

Lindzen, R. S., & Chapman, S. (1969). Atmospheric tides [Journal Article]. *Space Science Reviews*, *10*(1), 3–188. https://doi.org/10.1007/BF00171584

Luck, S. J., & Kappenman, E. S. (2013). *The oxford handbook of event-related potential components* [Book]. https://doi.org/10.1093/oxfordhb/9780195374148.001.0001

Oostenveld, R., Fries, P., Maris, E., & Schoffelen, J. M. (2011). FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data [Journal Article]. *Computational Intelligence and Neuroscience*. https://doi.org/10.1155/2011/156869

R Core Team. (2020). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. http://www.R-project.org/

Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. *9th Python in Science Conference*.

Smith, N. J. (2011-2015). Patsy - describing statistical models in python. In *ReadTheDocs*. https://patsy.readthedocs.io

Smith, N. J., & Kutas, M. (2015a). Regression-based estimation of ERP waveforms: I. The rERP framework [Journal Article]. *Psychophysiology*. https://doi.org/10.1111/psyp.12317

Smith, N. J., & Kutas, M. (2015b). Regression-based estimation of ERP waveforms: II. Nonlinear effects, overlap correction, and practical considerations [Journal Article]. *Psychophysiology*. https://doi.org/10.1111/psyp.12320

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*.

Tremblay, A., & Newman, A. J. (2015). Modeling nonlinear relationships in ERP data using mixed-effects regression with r examples. *Psychophysiology*, *52*(1), 124–139. https://doi.org/https://doi.org/10.1111/psyp.12299

Troyer, M., Urbach, T. P., & Kutas, M. (2020). *Toward dissociating general reading experience and domain-specific knowledge sources during RSVP reading: An exploratory rERP analysis* [Conference Poster]. https://doi.org/10.17605/OSF.IO/YNV9K

Urbach, T. P., DeLong, K. A., Chan, W. H., & Kutas, M. (2020). An exploratory data analysis of word form prediction during word-by-word reading [Journal Article]. *Proceedings of the National Academy of Sciences*, 201922028. https://doi.org/10.1073/pnas.1922028117