rERPy Documentation

Release 0.2.1

Nathaniel J. Smith

February 04, 2014

Contents

rERPy is a Python toolkit for doing ERP/ERF analysis of brainwave data (EEG or MEG), using both traditional averaging-based ERP/ERF estimation, and a fancy new regression-based technique for estimating ERP/ERF waveforms, which we call *rERP (or rERF)* <*http://vorpus.org/rERP/*> for short. rERP analysis is a strict generalization of ERP analysis – every published ERP is also a rERP. But with rERP analysis, you can estimate ERP waveforms regardless of whether your design is factorial or continuous or both, whether it is orthogonal or partially confounded, whether your continuous covariates have linear or non-linear effects, and whether your events of interest produce overlapping ERPs or not.

Contents:

CHAPTER 1

Vital statistics

Documentation: not yet

Downloads: not yet

Dependencies:

- Python 2.7 (not Python 3 yet, sorry – patches accepted!)
- numpy
- scipy
- pandas
- patsy

Optional dependencies:

- nose: needed to run tests

Install: probably not a great idea yet

Mailing list: not yet, but in the mean time you can hassle nathaniel.smith@ed.ac.uk

Code and bug tracker: https://github.com/rerpy/rerpy

License: GPLv2+, see LICENSE.txt for details.

CHAPTER 2

User manual

rERPy doesn't work quite like other ERP analysis tools. rERP analysis allows for stimuli that vary in complex ways on multiple dimensions – therefore, we need a way to describe such stimuli. rERP analysis is flexible, allowing for many potentially different analyses of the same data, which means that we may need to spend some time exploring different options – therefore, we would like the process of specifying a new model to be as quick and simple as possible. And rERP analysis requires us to set up predictors, which for categorical variables requires the application of dummy, treatment, or other more exotic coding schemes, and for continuous variables may involve transformations like centering, log transformations, the use of spline bases, and so forth. Therefore, we would like a simple way to describe complex sets of predictors.

To accomplish these goals, rERPy has two key features. First,

2.1 Getting oriented

rERPy holds EEG/MEG data in `Dataset` objects. These objects store several types of information, all bundled up together:

First, they hold the actual EEG/MEG data, divided into contiguous spans of recording. Since we need some way to refer to these spans, we call them *recspans*. If you combine two subjects' data for an analysis, you will have multiple recspans in one `Dataset`. If you record from a subject for a bit, and then pause the recording, and then record some more, then you will have multiple recspans. We find this easier and less error-prone then the other common technique of storing all data concatenated into one long array, with a separate record of where the "boundary points" are between subjects/pauses, etc.; by having one array per recspan, we can never accidentally treat non-contiguous data as if it were contiguous. If you have a `Dataset`, then `data_set[0]` gives you the first recspan (represented as a `pandas.DataFrame`), `len(data_set)` tells you how many recspans there are, `for recspan in data_set: ...` lets you iterate over all of them, etc.

Second, they hold basic metadata needed to interpret the EEG/MEG: sampling rate, units, channel names, etc., represented as a `DataFormat` object. This can be accessed with `data_set.data_format`. This provides various convenience methods; e.g., you can conveniently convert between millisecond-based and tick-based representations of time using `DataFormat.ms_to_ticks()` and `DataFormat.ticks_to_ms()`.

Third, they hold a record of what events have occurred, when they occurred (relative to the recording), and arbitrarily detailed information about each event.

rERPy's system for storing event data is very different from that used in other systems, and is extremely rich and powerful. For each event, we record: * In which recspan it occurs. * Which tick it starts on. * Which tick it ends on (to allow for temporally extended events,

e.g., marking the extent of an artifact).

-

2.2 Loading data

XX

2.3 Examining data

XX

2.4 Estimating rERPs

XX

2.5 Visualizing results

TBD

2.6 Exporting results

TBD

CHAPTER 3

Compatibility notes

CHAPTER 4

Coming from ERPSS

XX

CHAPTER 5

Coming from EEGLAB

TBD

CHAPTER 6

API Reference

6.1 Core objects

6.2 IO

CHAPTER 7

Indices and tables

- *genindex*
- *modindex*
- *search*

r