Cognition and Natural Sensory Processing Workshop (CNSP)
2-4 August 2021

The **Eelbrain** Python Toolkit

Christian Brodbeck

christianbrodbeck.net / christianbrodbeck@me.com

UCONN
UNIVERSITY OF CONNECTICUT

Overview

Agenda

- What you can do with Eelbrain
- How easy it is
- How to get started

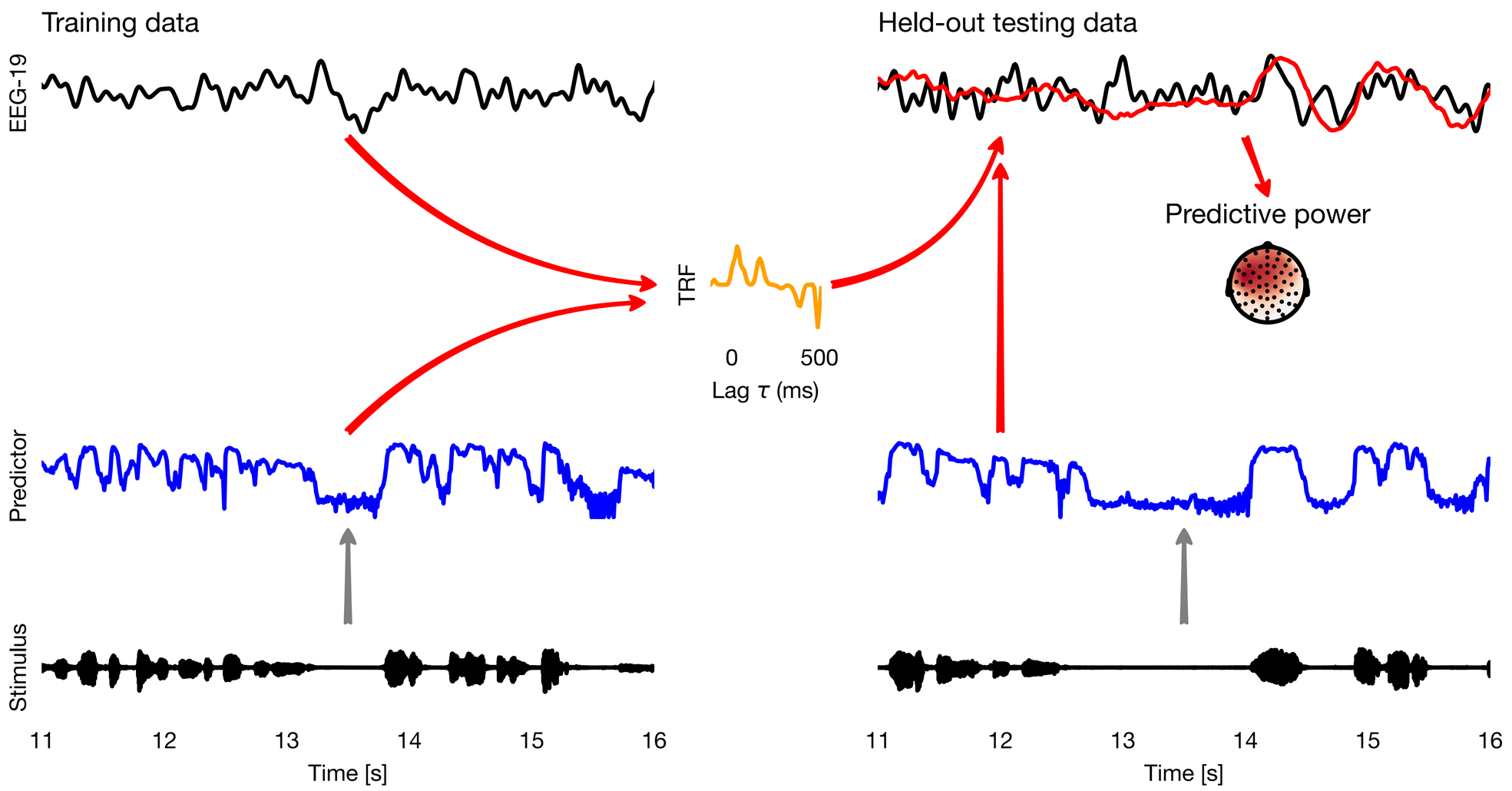
Eelbrain in the Python eco-system

Tutorial manuscript

Components of Eelbrain

- Time-series data
- Deconvolution
- Visualization
- Mass-univariate statistics

Eelbrain: toolkit for deconvolution analysis



Eelbrain in the Python eco-system

MNE

- EEG/MEG data I/O
- Preprocessing
- Many analysis tools

Eelbrain

- N-dimensional time series data
- Deconvolution
- High-level visualization
- Mass-univariate statistics

R

- Export to other platforms

pingouin

- Univariate statistics

scikit learn

- Machine learning (e.g. classification)

. . .

NumPy

- N-dimensional array class
- Optimized array operations

SciPy

- Extends NumPy for scientific applications (linear algebra, signal processing, image processing, …)

matplotlib

- Low-level plotting functions

jupyter

- Notebook environment

Tutorial: manuscript with code

Tutorial for using Eelbrain with the Alice EEG dataset

https://github.com/Eelbrain/Alice/discussions/2

- Introduces deconvolution analysis and demonstrates several applications
- Analysis of the Alice audiobook listening EEG dataset
- Source code from raw data to figures: https://github.com/Eelbrain/Alice

Get the manuscript!

- Tell us how to improve it
- Ask questions on GitHub
  - On the Alice data/analysis: https://github.com/Eelbrain/Alice/discussions
  - On Eelbrain: https://github.com/christianbrodbeck/Eelbrain/discussions

Eelbrain: A Python toolkit for time-continuous analysis with temporal response functions

Christian Brodbeck[1*], Proloy Das[2], Joshua P. Kulasingham[3], Shohini Bhattasali[3], Phoebe Gaston[1], Philip Resnik[3] & Jonathan Z. Simon[3]

1) University of Connecticut
2) Massachusetts General Hospital
3) University of Maryland, College Park

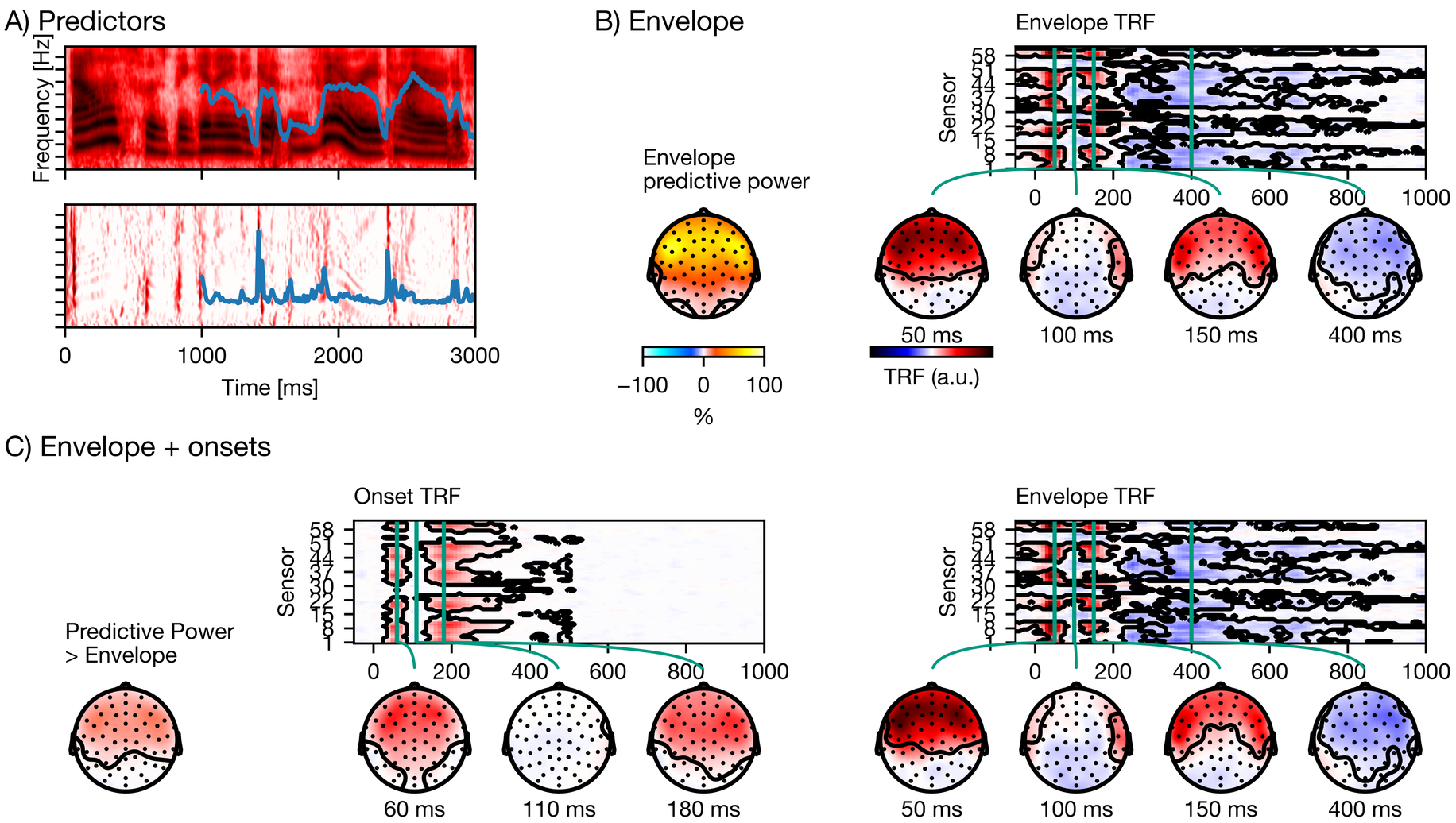
* christianbrodbeck@me.com

Abstract

Even though human experience unfolds continuously in time, it i
entails cascading processes building hierarchical cognitive str
speech perception, humans transform a continuously varying a
words, and meaning, and these levels all have different, interde
Deconvolution analysis has recently emerged as a prom
electrophysiological brain responses related to such complex m
introduce the Eelbrain Python toolkit, which makes this kind of ar
demonstrate its use, using continuous speech as a sample paradi
dataset of audiobook listening. A companion GitHub repository
code for the analysis, from raw data to group level statistics. N
hypothesis-driven approach in which the experimenter specifies
representations that are hypothesized to have contributed to br
as predictor variables for the electrophysiological signal. This
regression problem, but with the addition of the time dimensio
decomposes the brain signal into distinct responses associated with the different predictor
variables by estimating a multivariate temporal response function (mTRF), quantifying the
influence of each predictor on brain responses as a function of time(-lags). This allows asking

bioRχiv
THE PREPRINT SERVER FOR BIOLOGY

Tutorial with code: Auditory TRFs



Design

Easy to install

- Using Anaconda distribution (download the GUI installer)
- Single command to install all libraries you need with an environment file:
  `$ conda env create --file=environment.yml`

Well documented

- Command reference
- Code examples

Open source

- GitHub: https://github.com/christianbrodbeck

16 lines (16 sloc) | 362 Bytes

```yaml
# Environment for Alice TRF analysis
# usage: $ conda env create --file=environment.yml
name: eelbrain
channels:
- conda-forge
dependencies:
- eelbrain >= 0.36
- pip
- ipython
- jupyter
- ipywidgets
- jupytext
- seaborn
- pip:
  - https://github.com/christianbrodbeck/TRF-Tools/archive/win.zip
  - https://github.com/christianbrodbeck/gammatone/archive/fmax.zip
```

Documentation

eelbrain.readthedocs.io

dev

Search docs

eelbrain.boosting

```
eelbrain.boosting(y, x, tstart, tstop, scale_data=True, delta=0.005, mindelta=None, error='l2', basis=0,
basis_window='hamming', partitions=None, model=None, validate=1, test=0, ds=None, selective_stopping=0,
partition_results=False, debug=False)
```

Estimate a linear filter with coordinate descent

Parameters

y (*NDVar*) – Signal to predict.

x (*NDVar* | *sequence of NDVar*) – Signal to use to predict `y`. Can be sequence of NDVars to include multiple predictors. Time dimension must correspond to `y`.

tstart (*scalar* | *sequence of scalar*) – Start of the TRF in seconds. A list can be used to specify different values for each item in `x`.

tstop (*scalar* | *sequence of scalar*) – Stop of the TRF in seconds. Format must match `tstart`.

scale_data (*bool* | *'inplace'*) – Scale `y` and `x` before boosting: subtract the mean and divide by the standard deviation (when `error='l2'`) or the mean absolute value (when `error='l1'`). Use `'inplace'` to save memory by scaling the original objects specified as `y` and `x` instead of making a copy.

delta (`float`) – Step for changes in the kernel.

mindelta (`Optional` [`float`]) – If the error for the training data can't be reduced, divide `delta` in half until `delta < mindelta`. The default is `mindelta = delta`, i.e. `delta` is constant.

error (`Literal` ['l1', 'l2']) –

partition_results (**bool**) – Keep results (TRFs and model evaluation) for each test-partition. This is disabled by default to reduce file size when saving results.

debug (**bool**) – Add additional attributes to the returned result.

See also

`plot.preview_partitions`

preview data partitions for cross-validation

Notes

The boosting algorithm is described in [1].

In order to predict data, use the **`convolve()`** function:

```
>>> ds = datasets.get_uts()
>>> ds['a1'] = epoch_impulse_predictor('uts', 'A=="a1"', ds=ds)
>>> ds['a0'] = epoch_impulse_predictor('uts', 'A=="a0"', ds=ds)
>>> res = boosting('uts', ['a0', 'a1'], 0, 0.5, partitions=10, model='A', ds=ds)
>>> y_pred = convolve(res.h_scaled, ['a0', 'a1'], ds=ds)
```

References

1

David, S. V., Mesgarani, N., & Shamma, S. A. (2007). Estimating sparse spectro-temporal receptive fields with natural stimuli. Network: Computation in Neural Systems, 18(3), 191-212. 10.1080/09548980701609235.

`Return type`

BoostingResult

Examples

eelbrain.readthedocs.io

stable

Search docs

Docs » Examples » Data partitions for boosting

Edit on GitHub

Note

Click here to download the full example code

Data partitions for boosting

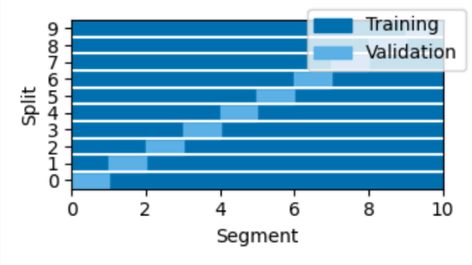
The boosting algorithm can use two different forms of cross-validation: cross-validation as stopping criterion (always on), and cross-validation for model evaluation (optional). This requires paertitioning the data into different segments. The `eelbrain.plot.preview_partitions()` function is for exploring the effect of different parameters on the way the data is split.

Validation

During boosting, every training step consists in modifying one element of the kernel/TRF. After every such step, the new TRF is evaluated against the validation data. For continuous data (without `Case` dimension), the default is to split the data into 10 equal-length segments, and perform 10 model fits, each using one of the segments as validation set. In the plots below, each "Split" shown on the y-axis corresponds to a separate run of the boosting algorithm. The results returned by the `boosting()` function would be based on to the average TRF of those 10 runs.

```
# sphinx_gallery_thumbnail_number = 6
from eelbrain import *


p = plot.preview_partitions()
```



The number of partitions can be controlled with the *partitions* parameter:

Design

Design principles

- Analysis is script-based: completely reproducible
- Concise, high-level commands
- Focus on outcome, not implementation

Examples

- Automatic handling of sensor positions for head-maps
- Meaningful indexing

Code Examples

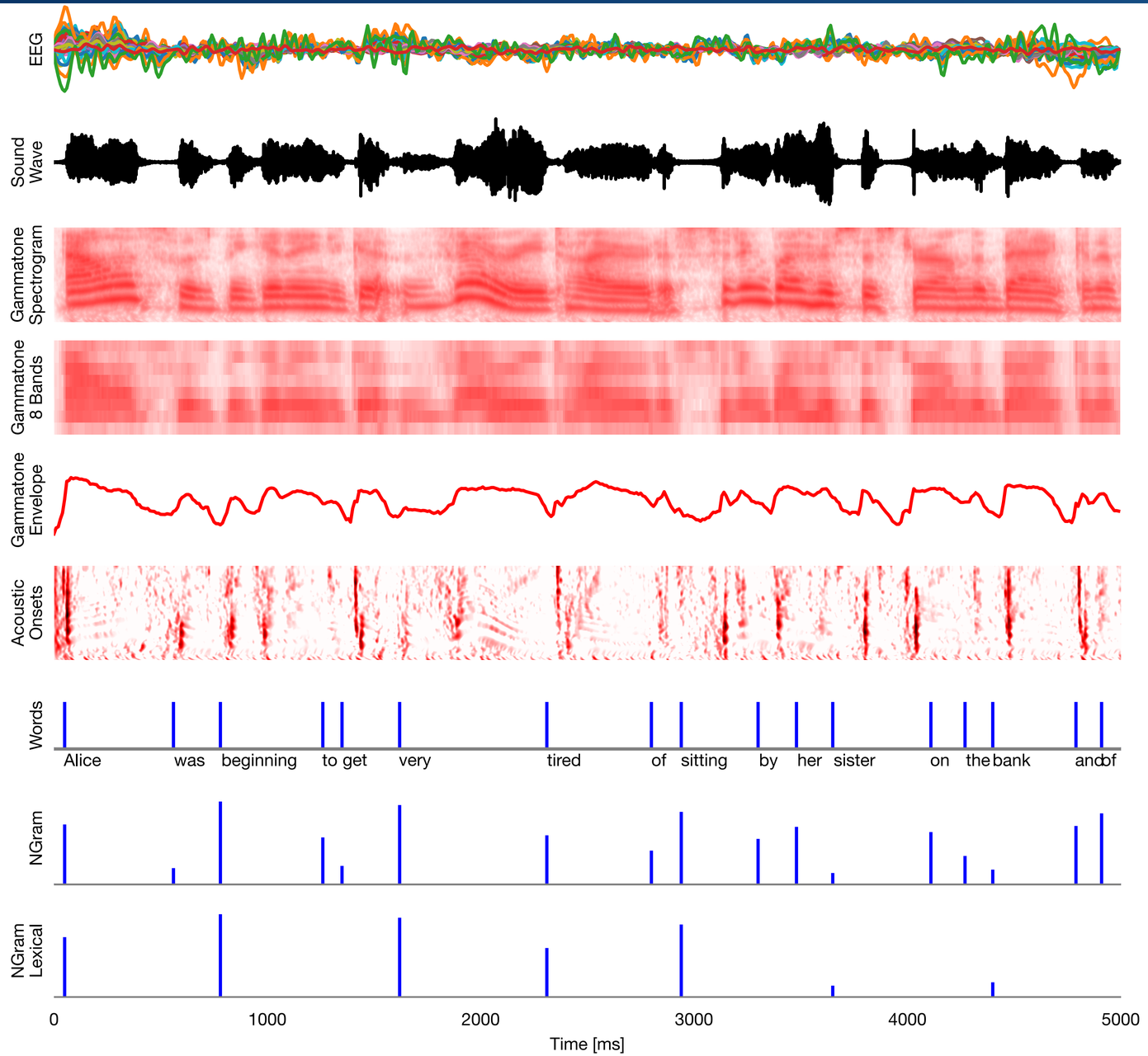
Time-series representations

Represent data with different dimensions

- Time
- Sensors (EEG, MEG)
- Frequency (e.g., of a spectrogram)
- …

Keeps track of meta-information for you

- EEG sensor positions
- Samplingrate
- …



Deconvolution

Boosting algorithm

- Boosting algorithm for estimating sparse mTRFs
  - Coordinate descent
  - Early stopping based on cross-validation to prevent over-fitting
- Can handle
  - Large numbers of predictors
  - Correlated predictors

Built-in *k*-fold cross-validation

- Cross-validated predictive power with one command
- Data partitioning based on trials or continuous time
- Retrieve results from different folds as independent estimates

Statistics

Mass-univariate statistics

- Permutation tests for multiple comparison correction
  - Max-statistic (Nichols & Holmes, 2002)
  - Cluster-mass based correction (Maris & Oostenveld, 2007)
  - Threshold-free cluster enhancement (Smith & Nichols, 2009)
- Works with arbitrary dimensions
- Currently available tests:
  - One-sample, independent and related *t*-tests
  - Pearson correlation
  - ANOVA (fixed, repeated measures, mixed, nested)
  - Two-stage tests for arbitrary linear models

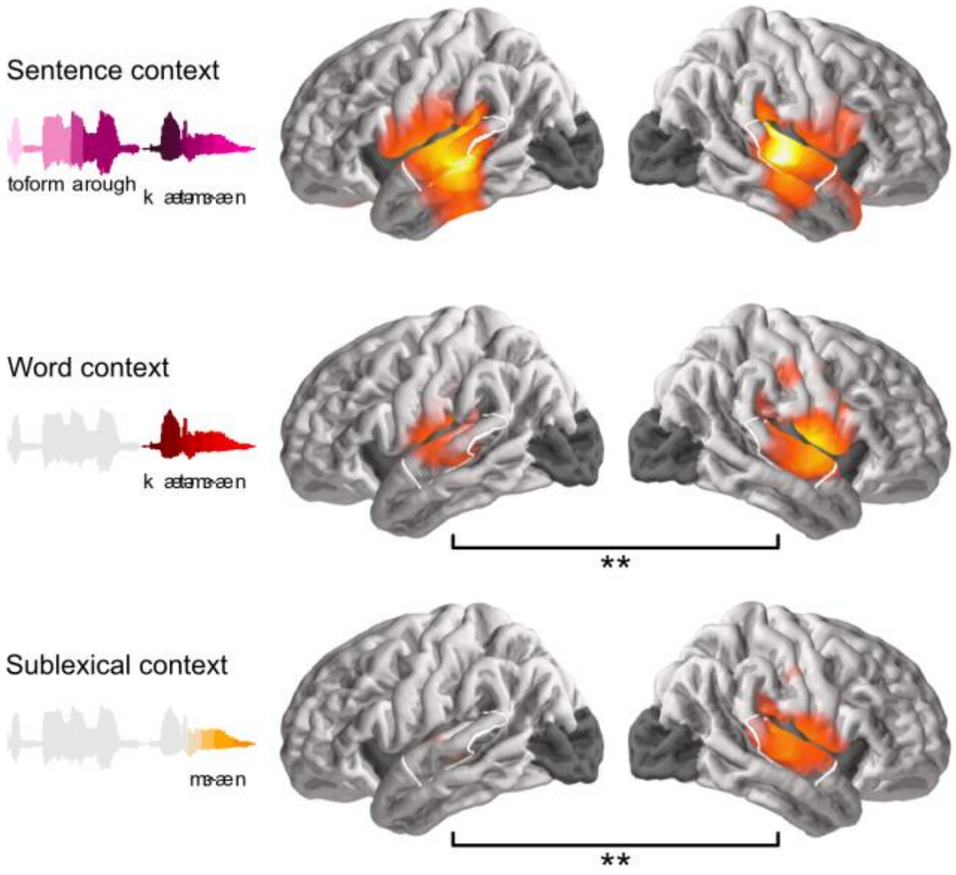
Basic univariate statistics

- ANOVA, *t*-tests, …
- Export to R, …

Extends to source localization

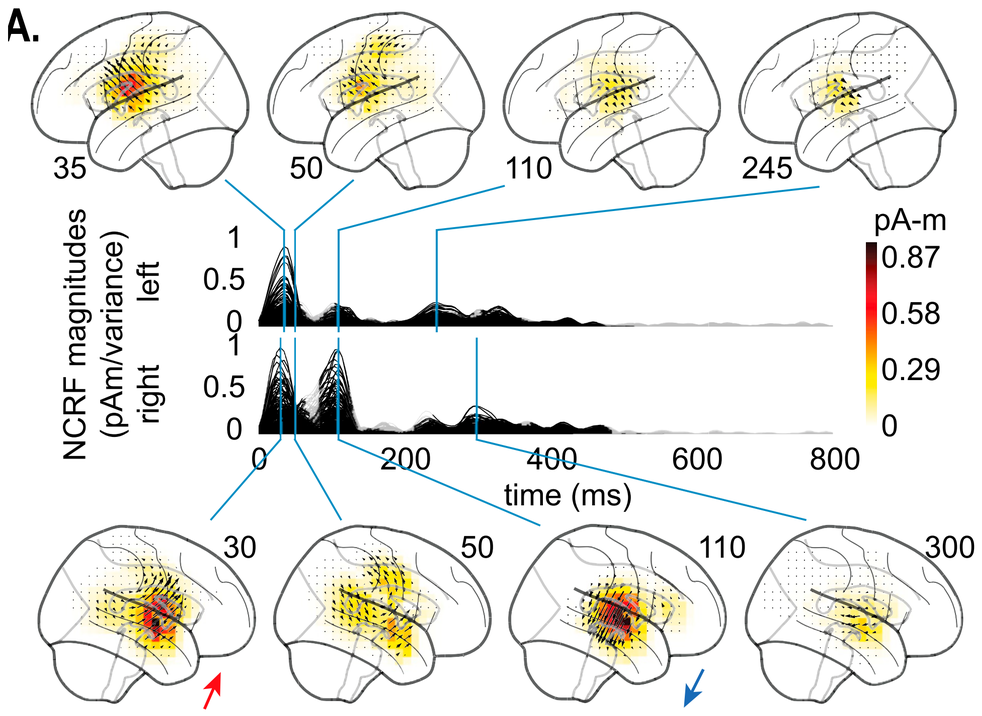
With MNE and boosting

- Example: Brodbeck et al., 2021



Joint source localization and mTRF estimation

- With Neuro-Current Response Functions (Das et al., 2020; https://github.com/proloyd/neuro-currentRF)



Thank you!

If you're interested:

- Download the Eelbrain tutorial preprint: https://github.com/Eelbrain/Alice/discussions/2
- Download the Alice dataset and analysis code: https://github.com/Eelbrain/Alice
  - Reproduce our results
  - Find new, better predictors for the EEG responses
- Ask questions on GitHub Discussions: https://github.com/Eelbrain/Alice/discussions
- Help us make the tutorial better!