nideconv: Easy deconvolution of neural signals using the general linear model and flexible basis functions

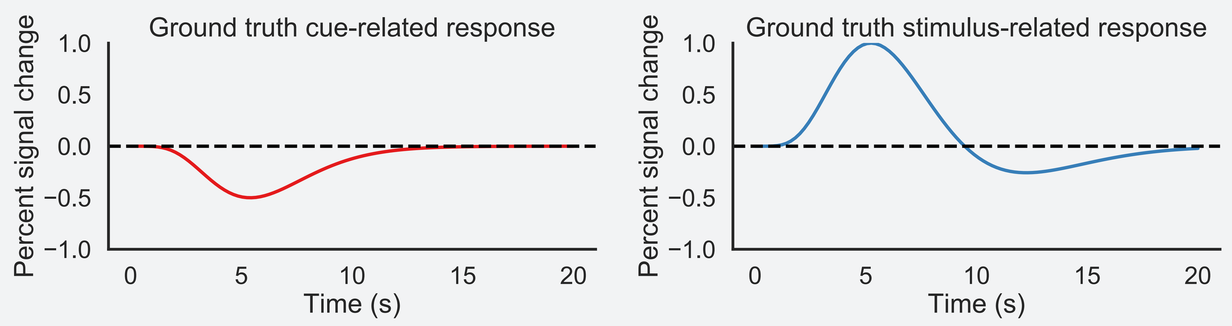
Gilles de Hollander & Tomas Knapen

Summary

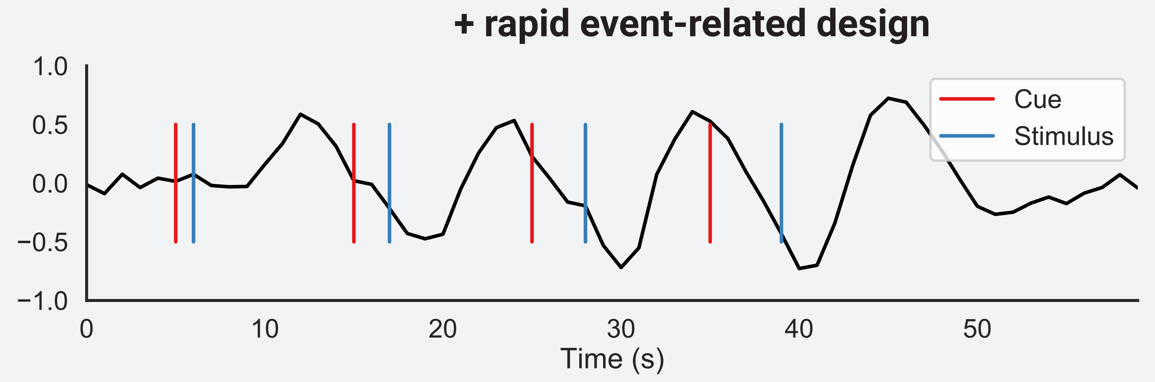
- Many neuroimaging methods (e.g., **fMRI, pupil size tracking**) yield time series that contain **overlapping event-related responses**.
- Overlapping event-related responses can be deconvolved using the general linear model (**GLM**). The GLM with the **canonical HRF** is still the workhorse of most task-based fMRI research.
- The **canonical HRF** is often misspecified and more flexible basis functions can be warranted. New **accelerated imaging techniques** with TRs in the order of a second now also allow us to do so.
- **Nideconv** offers an **easy-to-use framework** within the **Python neuroimaging ecosystem** for fitting GLMs with flexible basis functions like finite impulse response functions (**FIR**) and **Fourier** sets.
- It also offers easy-to-use **plotting capabilities, utility functions** and a **Hierarchical Bayesian** version of the **GLM**. This estimation procedure increases statistical power and reliability of parameter estimates, especially in noisy regimes

The trouble of overlap

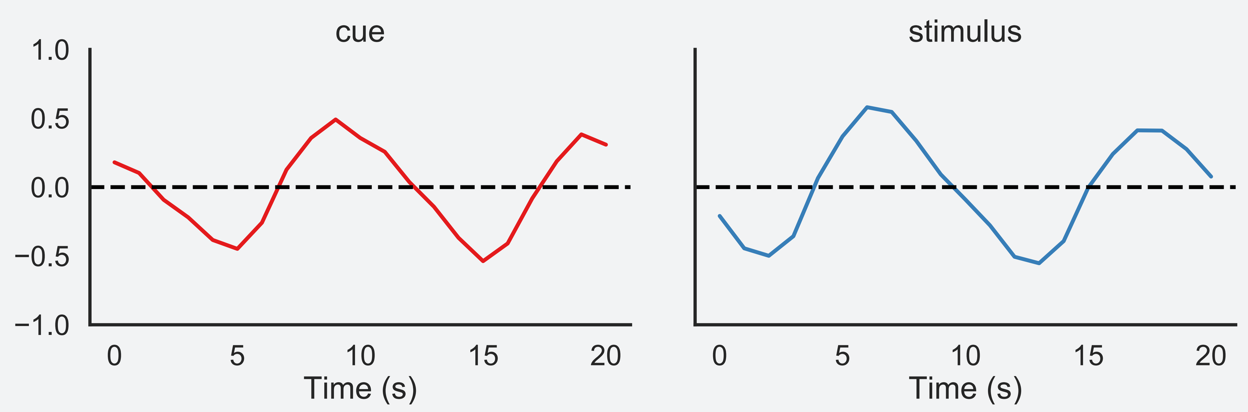
Different event-related responses



+ rapid event-related design



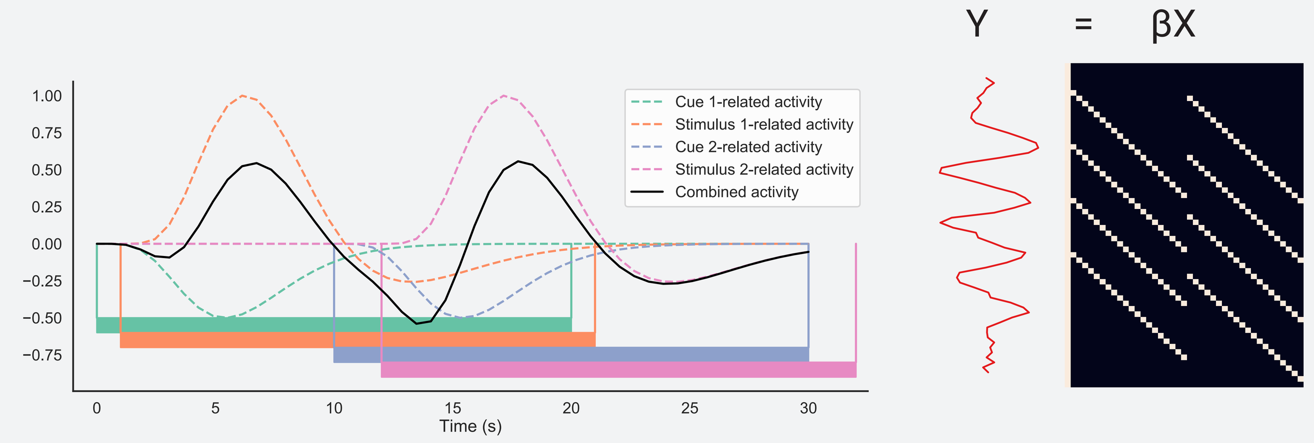
+ epoching



...means trouble

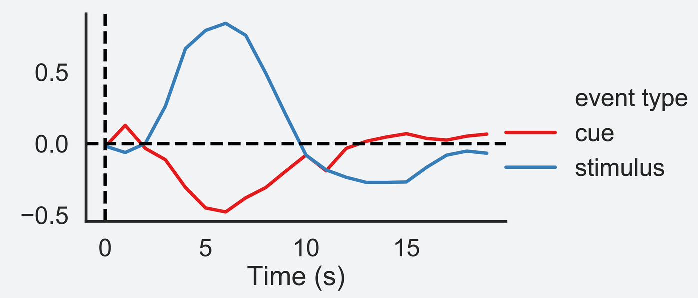
The GLM framework

Model overlapping responses as linear combinations of **basis functions** and solve a GLM



Nideconv allow you do this with just a few lines of code , yielding a nice little plot, sprinkled with some *seaborn* `FacetGrid` magic:

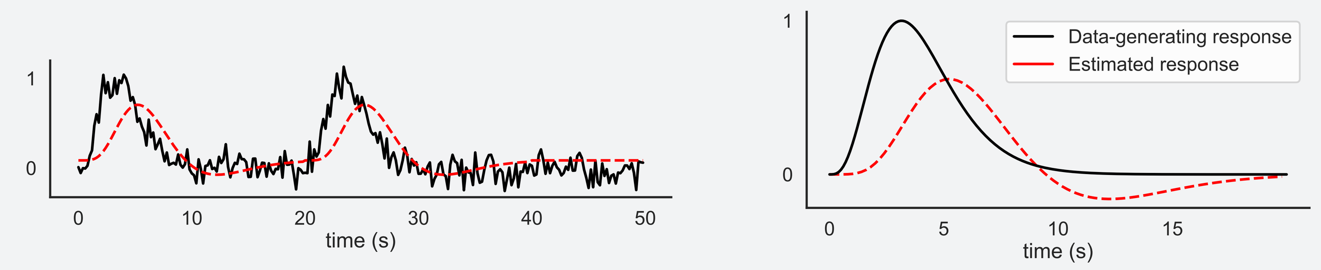
```
import nideconv
rf = nideconv.ResponseFitter(input_signal=data,
                             sample_rate=1)
rf.add_event(event_name='cue',
             onsets=onsets.loc['cue'].onset,
             basis_set='fir',
             interval=[0,20])
rf.add_event(event_name='stimulus',
             onsets=onsets.loc['stim'].onset,
             basis_set='fir',
             interval=[0,20])
rf.fit()
rf.plot_timecourses()
```



Not all HRFs are created equal

The canonical HRF is not appropriate for all brain areas and task conditions. This can lead to misspecified models

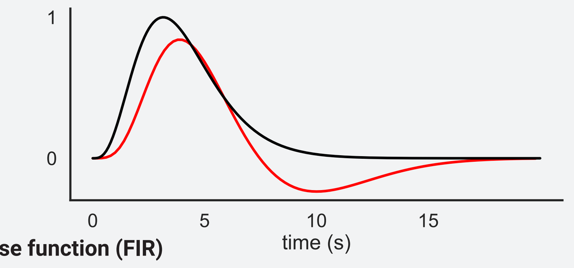
Example: cHRF fitted on simulated data with an early peak and no post-stimulus undershoot.



More flexible basis functions

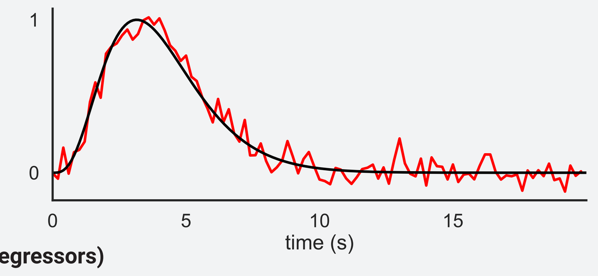
Canonical HRF with derivative wrt time

```
rf = ResponseFitter(input_signal=data, sample_rate=5)
rf.add_event('stimulation',
             onsets.loc['stimulation'].onset,
             interval=[0, 20],
             basis_set='canonical_hrf_with_time_derivative')
rf.fit()
```



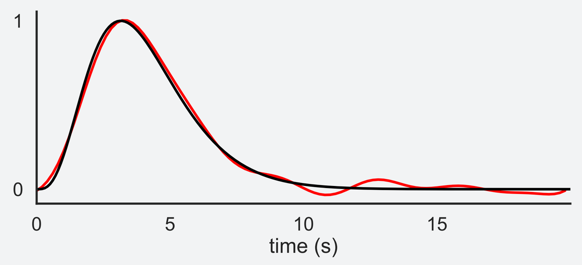
Finite Impulse Response function (FIR)

```
rf = ResponseFitter(input_signal=data, sample_rate=5)
rf.add_event('stimulation',
             onsets.loc['stimulation'].onset,
             interval=[0, 20],
             basis_set='fir')
rf.fit()
```



Fouriet set (13 regressors)

```
rf = ResponseFitter(input_signal=data, sample_rate=5)
rf.add_event('stimulation',
             onsets.loc['stimulation'].onset,
             interval=[0, 20],
             n_regressors=13,
             basis_set='fourier')
rf.fit()
```



Hierarchical Bayesian estimation

The standard GLM can be extended to a Hierarchical Bayesian GLM (**HB-GLM**).

A HB-GLM estimates all individual subject parameters, as well as group estimates, in the same model. Therefore, it **stabilizes individual parameters estimates**, especially when **data** is **scarce** or **missing**. A common use case here is error trials. Lastly, the HBGLM can provide **Bayesian credible intervals** that take into account both group- and subject-variance in a graceful manner.

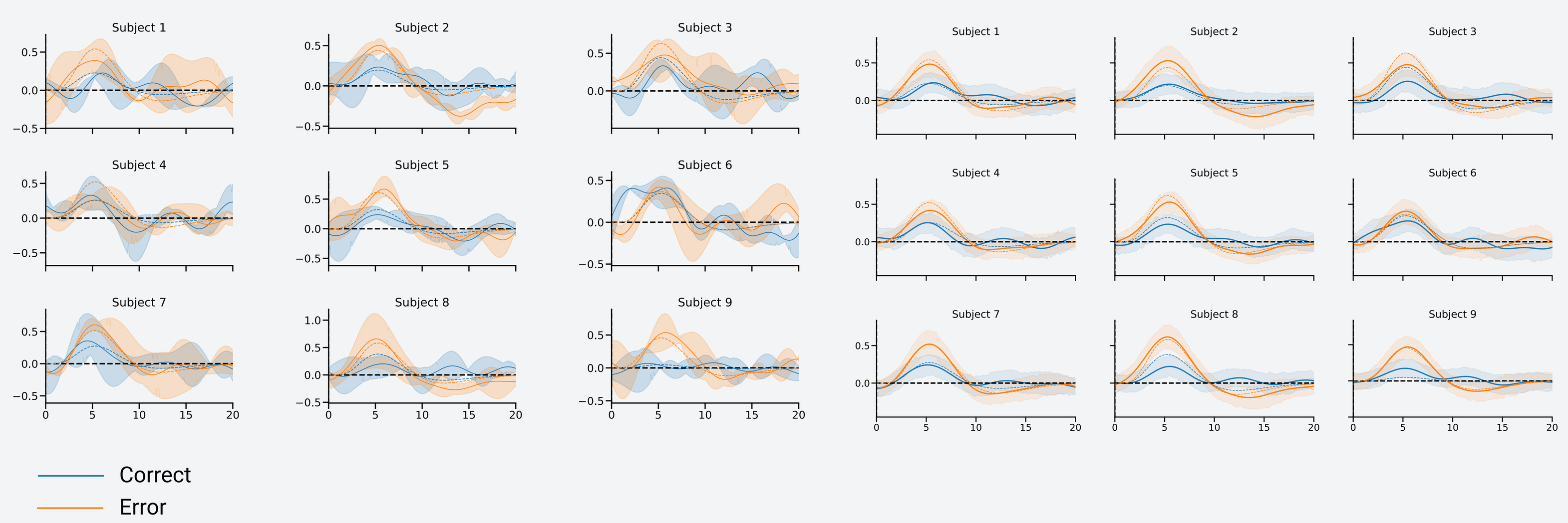
In Nideconv, the HB-GLM can be fit for arbitary models and basis functions, with a few lines of code, using the powerful **NUTS**-sampler of **STAN** in the background:

```
import nideconv
gmodel = nideconv.GroupResponseFitter(data, onsets,
                                      input_sample_rate=1/1.5,
                                      concatenate_runs=False)
gmodel.add_event('Correct', basis_set='fourier', n_regressors=9, interval=[0, 21])
gmodel.add_event('Error', basis_set='fourier', n_regressors=9, interval=[0, 21])

hbmodel = nideconv.HierarchicalBayesianModel.from_groupresponsefitter(gmodel)
hbmodel.build_model()
hbmodel.sample()
```

We simulated data from 9 subjects, with each 3 runs of 5 minutes (TR=1.5s), containing 16 correct and 1 - 6 error trials.
Because of the scarcity of the error trials, standard GLMs show very unstable parameter estimates, unlike the HB-GLM, where individual estimates are regularized (shrinked) towards the group mean.

Frequentist GLM | **Bayesian Hierachical GLM**



https://github.com/VU-Cog-Sci/nideconv/

Spinoza Centre for Neuroimaging

VRIJE UNIVERSITEIT AMSTERDAM