Semi-simulated ECoG data

Written by J. Schrouff

This tool allows to create semi-simulated ECoG data with varying signal to noise ratio (SNR) and sparsity (i.e. number of channels containing 'signal'), based on the rest session from a patient to which we added a fake design.

This code was created by J. Schrouff for use in 2 publications:

1. *Interpreting weight maps in terms of cognitive or clinical neuroscience: nonsense? J. Schrouff & J. Mourao-Miranda (submitted). Deposited on arXiv:* http://arxiv.org/abs/1804.11259

This work investigated the recovery of univariate and multivariate analyses, at the feature (i.e. time point per channel) and region of interest level (i.e. channel). Simulated signal was added on condition A, based on the noise in condition B, to estimate their discrimination using different techniques.

2. *Face Information is Anatomically Localized but Distributed in Time Across the Human Temporal Cortex. J. Schrouff, S. Salehi, O. Raccah, V. Rangarajan, S. Baek, J. Mourão-Miranda, Z. Helili, A.L. Daitch & J. Parvizi (submitted)*

This work used simulated data to assess whether our response onset latency (ROL) detection depended on the signal's shape (amplitude or slope) or not, as supplementary information. The ROL for condition A was estimated on the simulated signals for varying levels of SNR, knowing the ground truth (i.e. times(1), defined as input).

Please see (1) for further details and cite when using.

Pre-requisites:

Matlab and SPM12. This tool was used with the pipeline that can be found at https://github.com/LBCN-Stanford/Preprocessing_pipeline.

An epoched SPM MEEG file must be provided. The file provided along the code is a rest session from a single patient with ECoG, to which a fake design was design (see below for details on design and preprocessing).

Usage:

Inputs:

The code takes as input:

- **fname**: the name of the raw signal file, epoched on the created fake conditions
- **options**: a structure with the different options to add signal

The options available are:

- **.isgood**: whether to add only on 'good' channels (1) or all (0, default)

- **.times**: vector of times in ms to define the onset and offset of simulated signal (e.g. [50 200], default: all window)
- **.condA**: conditions to add signal to, in cell array of strings (e.g. {'A'}, default: all)
- **.condB**: condition to consider as 'noise', cell array of strings (e.g. {'B'}, default: baseline signal in [-minimum 0] of all conditions)
- **.newname**: string with appended name for simulated files (default: 'Simulated')
- **.permute_chans**: flag to permute channels randomly before adding signal (default: 0)
- **.SNR**: range of imposed SNR to simulate, each generating a separate file
- **.sparsity**: range in % of sparsity, representing the % of channels to add signal to (default: 5:5:100)
- **.window**: substructure to specify the 'type' ('rect' or 'ramp'), the parameters ('args', e.g. 3 for slope of ramp) and the smoothing ('smooth.smooth': 1 or 0, 'smooth.win': value in ms) of the window to add to the signal. Default is rectangular window, no smoothing ('.window.type='rect'','.window.smooth.smooth=0')

Note: the sparsity is re-calculated to add signal on an integer numbers of channels. So the final sparsity may vary from the values input.

Outputs:

For each SNR-sparsity combination, a SPM MEEG file with the simulated data. In addition, a 'Results_SNR_Sparsity.mat' is created, containing metrics on the simulated data:

- **effsnrsite**: matrix of size number of (good) channels * number of sparsity levels * number of SNR levels. It contains the effective SNR when comparing condA with condB after adding the signal. This value should be close to the desired SNR, but typically deviates a bit from it as SNR is estimated on average across trials.
- **sitesord**: the order of the channels on which signal is added. If options.permute_chans is set to 0, this represents the index of the (good) channels in the object. Otherwise, it displays a random permutation of these indices.
- **insnrsite**: same as effsnrsite, but with imposed SNR per channel and sparsity
- **insparse**: same as insparse, keeping track of the sparsity level for each combination
- **minSignal**: mean of input signal, i.e. average simulated 'signal' amplitude for each channel and each SNR-sparsity combination (i.e. amplitude of added window). Of dimension number of (good) channels * number of sparsity levels * number of SNR levels
- **meffSignal**: mean of effective signal, i.e. average signal + noise for each channel and combination (i.e. amplitude of obtained signal in conditions condA). Of dimension number of (good) channels * number of sparsity levels * number of SNR levels

Examples:

(1) Create smoothed rectangular windows, on 'A' trials, using 'B' trials as noise:

fname = spm_select;

options.isgood = 1;

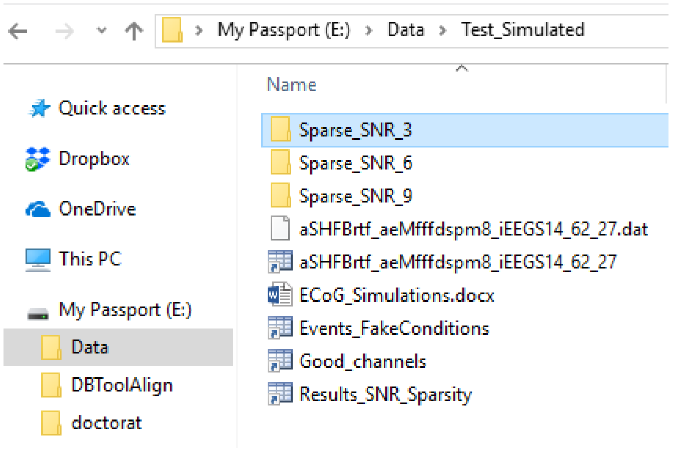
options.condA = {'A'};

options.condB = {'B'};

options.newname = 'Rect_';

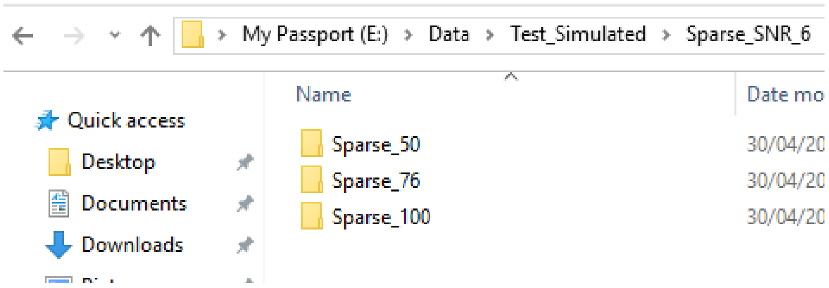
```
options.permute_chans = 1;
options.SNR = [3,6,9];
options.sparsity = 50:25:100;
options.window.type = 'rect';
options.window.smooth.smooth =1;
options.window.smooth.win =50;
options.times=[100 900];
[effsnrsite] =  create_semi_simulated_ECoG(fname,options);
```

The output signal is saved in subfolders: for each level of SNR, one folder is created. Within each SNR folder ('Sparse_SNR_xx'), one folder is created per level of sparsity ('Sparse_yy'). Within these, the simulated data is saved, with appended options.newname, as a .mat and .dat SPM MEEG format.



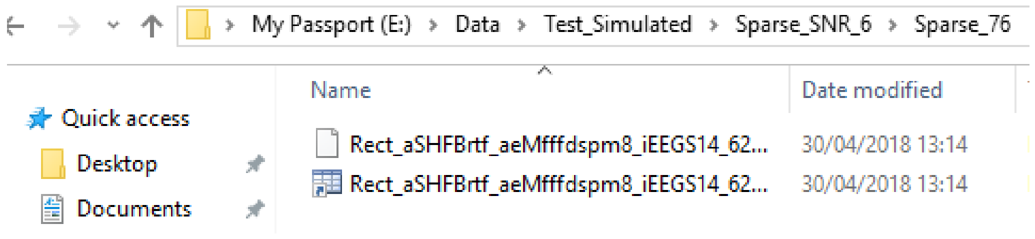
Sparse_SNR_3
Sparse_SNR_6
Sparse_SNR_9
aSHFBrtf_aeMfffdspm8_iEEGS14_62_27.dat
aSHFBrtf_aeMfffdspm8_iEEGS14_62_27
ECoG_Simulations.docx
Events_FakeConditions
Good_channels
Results_SNR_Sparsity

Figure 1: Folders created for each SNR



Name	Date mo
Sparse_50	30/04/20
Sparse_76	30/04/20
Sparse_100	30/04/20

Figure 2: Subfolders created for each sparsity level


My Passport (E:) > Data > Test_Simulated > Sparse_SNR_6 > Sparse_76
Quick access
Desktop
Documents

Name	Date modified
Rect_aSHFBrtf_aeMfffdspm8_iEEGS14_62...	30/04/2018 13:14
Rect_aSHFBrtf_aeMfffdspm8_iEEGS14_62...	30/04/2018 13:14

Figure 3: Files saved with appended 'Rect_' as specified in options.newname

We then use our pipeline to review the obtained signals, for the combination of SNR=3 and sparsity = 76.

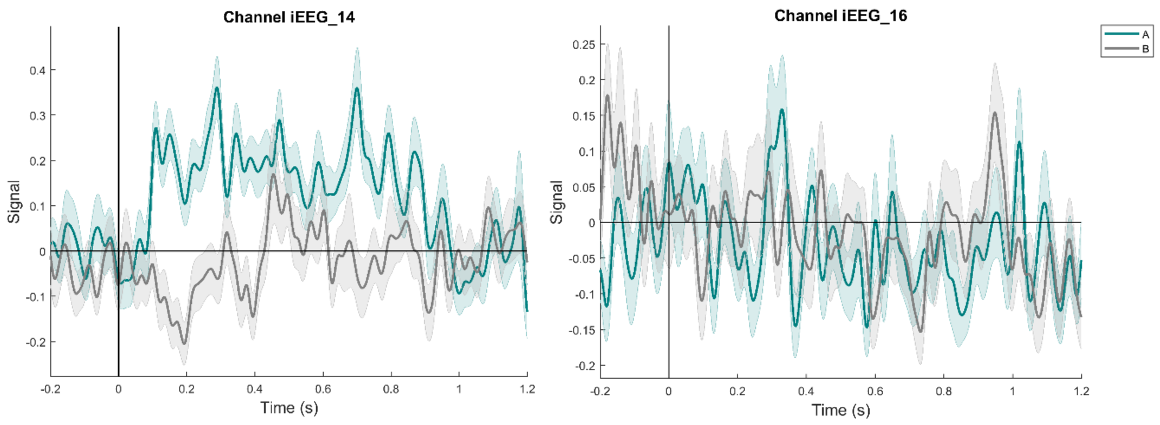


Figure 4: examples of signal obtained. Left: simulated signal was added on channel 14, with SNR =3 over the window [100 900]ms. Right: no signal was added on channel 16.

(2) Modify options to create ramp windows, on 'A' trials, using 'B' trials as noise:

```
options.newname = 'Ramp_';
options.permute_chans = 0;
options.sparsity = [];
options.window.type = 'ramp';
options.window.args = 2;
options.window.smooth.smooth =0;
[effsnrsite] =  create_semi_simulated_ECoG(fname,options);
```

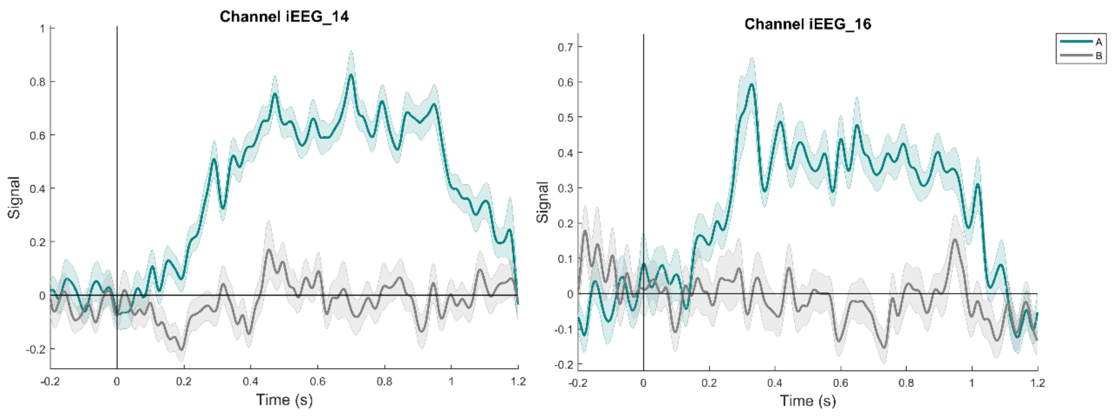


Figure 5: examples of signal obtained. Simulated signal was added on both channels 14 and 16 with SNR =9 over the window [100 900]ms

(3) Modify options to create ramp windows, on all trials, using the baseline of all trials as noise:

```
options.condA = [];
options.condB = [];
options.newname ='Baseline_';
[effsnrsite] = create_semi_simulated_ECoG(fname,options);
```

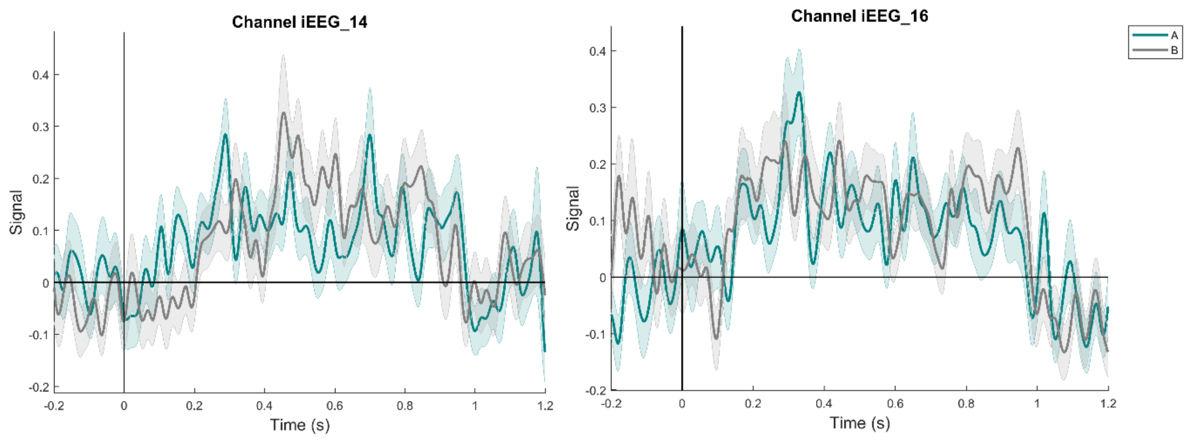


Figure 6: examples of signal obtained. Simulated signal was added on both channels 14 and 16 with SNR =6 over the window [100 900]ms, based on the baseline in the [-200 0]ms time window.

<u>Input epoched data</u>:

Raw data: The data was recorded from intracranial electrodes implanted in a patient with pharmaco-resistant epilepsy. This study was approved by the Stanford IRB and the patient gave written consent to participate in the study. 64 electrodes were implanted and signal was recorded using a Tucker Davis system (sampling rate: 1536Hz). The signal was acquired during a 5-minute wakefulness rest period, with eyes closed. Electrodes displaying pathological activity were discarded from further analysis.

Simulated design: A fake experimental design was simulated: 2 conditions, ‘A’ and ‘B’, presented at random every 1.9 seconds. The stimuli are further assumed to last for 1 second. This yielded 146 stimuli, 73 for each category.

Pre-processing: Signal pre-processing was performed with specific ECoG routines (https://github.com/LBCN-Stanford/Preprocessing_pipeline) using Matlab (www.mathworks.com) and SPM12 (www.fil.ion.ucl.ac.uk/spm). First, the data was converted to SPM format and downsampled to 1kHz. The continuous signal was filtered for line noise and harmonics (stop-band: 57-63Hz, 117-123Hz, 177-183Hz) and an automatic quality assessment identified ‘noisy’ or ‘spiky’ channels based on their variance and number of ‘jumps’ (i.e. signal derivative>100µV), leaving 38 ‘good’ channels. The data was re-referenced to the average of all good channels before being epoched in the [-400, 1400]ms window around ‘onset’ (as defined by the simulated design) and baseline corrected using the [-400, 0]ms window. Epochs displaying flat segments of more than 4ms or ‘jumps’ larger than 100µV were discarded from further analysis. The signal was then decomposed using a 5-wavelets decomposition in the 70 to 170Hz frequency band (step: 10Hz, avoiding 120Hz) to estimate High Frequency Broadband (HFB) power. The time-frequency signal was z-scored based on the pooled baselines of all events in the [-300, 0]ms window before onset to avoid edge effects. The signal in the [-200, 1200]ms window was finally smoothed using a 50ms Gaussian window. Epochs displaying z-scores larger than 8 were discarded, leaving 60 trials for condition ‘A’ and 56 for ‘B’. This pre-processing procedure is standard in stimulus-based ECoG studies.