Dynamic Functional Connectivity tutorial

Brainhack Tutorial Zurich
Sliding window analysis and innovation-driven co-activation pattern pipelines

March 3rd, 2017

1 Introduction

In this hands-on session, we explore dynamic functional connectivity (dFC) through two different pipelines: (1) a sliding window analysis, where we explore the impact of window characteristics, dynamic graph metrics, and the generation of summarising eigenconnectivity building blocks; and (2) the generation of innovation-driven co-activation patterns (iCAPs), following the previous application of total activation (TA). Participants are expected to have attended the theoretical introduction given by T. Bolton. The data needed to run both pipelines can be found in the *BrainHack/Practice/StartData* folder. It contains:

- mask: Logical vector of size n_voxels $\times 1$ indexing the elements belonging to gray matter (GM) with $+1$.
- fHeader: Header containing information about the considered data (e.g. dimension).
- Innovation: Cell array, where each cell represents the innovation signal matrix (output of the total activation algorithm) of a specific subject, of size n_GM_voxels $\times$ n_timepoints.
- mask1: Cell array, with each cell of size n_GM_voxels $\times$ n_timepoints corresponding to one subject. For each location and time point, the mask contains $+1$ for a positive innovation, -1 for a negative innovation, and 0 for no innovation. This mask refers to voxels that survived thresholding.
- SW_data: Cell array, with each cell of size n_ROI $\times$ n_timepoints corresponding to one subject, and containing atlased time courses.
- CB: Cell array of size n_ROI $\times$ 1, with each cell containing the short name for one region of the used atlas.

The dataset considered here is the same for both subparts of the tutorial: it contains $n_s = 12$ control subjects considered in some previous functional connectivity (FC) works [1, 2]. Acquisition was performed at $TR = 1.1s$, with the subjects lying still with eyes closed. Preprocessing details will not be covered here; for information about the performed preprocessing for the sliding window and iCAP parts of this tutorial, see [2] and [3], respectively.

2 Sliding window analysis

Below, we provide step-by-step guidance on how a basic sliding window analytical pipeline is implemented. A functional script that can be run with various parameter sets is provided in the *BrainHack/Practice* folder (*SW_tutorial_Brainhack_FULL.m*).

2.1 Data loading and parameters

First, we want to load the data required for this part; i.e., the regional activity time courses to consider for the analysis, and the codebook.

```
% SW_data is a cell array, with each cell one subject data (n_ROI x
% n_timepoints)
load SW_data;

% CB is a n_ROI cell array with the short names of atlas areas; used for
% plotting
load CB;
```

Some of the parameters will be kept fixed: in this tutorial, we consider a set of $n_{ROI} = 88$ regions obtained through a parcellation with the Automated Anatomical Labeling (AAL) atlas [4]. Time courses from the bilateral pallidum were removed due to signal dropout.

```
% Number of brain regions contained in the atlas (AAL without Pallidum)
n_ROI = 88;

% Number of subjects
n_subjects = length(SW_data);

% Number of time points
n_timepoints = size(SW_data{1},2);

% Number of eigenconnectivity building blocks to compute
n_components = 20;
```

Some parameters can be changed to observe how it affects the outcomes from the pipeline. It includes basic window characteristics (length, shape and step size), edge density for the computation of dynamic graph metrics (performed on a binarized adjacency matrix), and whether or not to include demeaning of subject-wise connectivity time courses prior to the generation of eigenconnectivity building blocks.

```
% Window length (in TRs)
window_length = 60;

% Window step (in TRs)
window_step = 1;

% Type of window shape desired (0: rectangular, 1: tukey, 2: gaussian,
% 3: exponential)
window = 0;
```

```
% Density of retained edges for dynamic graph analysis
rho = 10/100;

% Perform or not the demeaning for eigenconnectivity generation
% (1 = with, 0 = without)
is_demean = 1;
```

2.2 Computation of connectivity time courses

Connectivity time courses are saved in the cell array CM, with each cell containing the data for one subject, of size n_ROI × n_ROI × n_sliding_windows. Computations are performed sequentially for each subject, for the selected set of parameters, over successive temporal windows. First, weights denoting the importance given to each data point from the window in assessing connectivity are computed according to the desired window shape; second, the function weightedcorrs compute the connectivity information, and stores it in CM.

Q.1: For a few regions of your choice, compare the connectivity time courses obtained with different window lengths. What happens when using very short, and very large, window lengths? Do you know of any rule as for a proper window length selection?

Q.2: For a few regions of your choice, compare the connectivity time courses obtained with different step sizes. What is the influence of this parameter?

Q.3: For a few regions of your choice, compare the connectivity time courses obtained using different window shapes. What shape gives the most fluctuating time courses? What about the least fluctuating ones? Why is it the case?

Q.4: For the same time point, compare the full connectivity matrices obtained for different subjects; are they the same? Why?
Hint: You can use the function View_A(CM_2D,n_ROI,CB) to plot a labeled connectivity matrix, with CM_2D the 2D connectivity information, n_ROI the number of regions, and CB the codebook for the used atlas.

Q.5: For a selected subject of your choice, compare the full connectivity matrices obtained at (1) close time points (separated by only few steps), and (2) remote time points. Do they look similar? Why?

2.3 Dynamic graph analysis and eigenconnectivities

Connectivity data from each subject is concatenated into a Data_s matrix, of size n_connectivity_pairs × n_sliding_windows. If demeaning is included, the Data_s matrices are individually demeaned prior to concatenation together into the final data matrix Data, of size n_connectivity_pairs × (n_sliding_windows × n_subjects).

Graph metrics are computed on each connectivity matrix obtained by the sliding window approach. Matrices are made binary by the function BinarizeAdjacency(SW,rho), where SW is the original

connectivity matrix, and rho is the fraction of edges that should be retained. Two metrics are computed for each node: *clustering coefficient*, with the function ComputeC, and *degree*.

To generate eigenconnectivity building blocks, a singular value decomposition (SVD) is run on the data matrix. The retrieved singular vector matrix u contains, in each column, one principal direction of the dataset; i.e., one eigenconnectivity building block. Each eigenconnectivity vector has size n_connectivity_pairs $\times$ 1, and they are arranged in decreasing order of explained data variance.

```
% Eigenconnectivities are simply the principal directions of the data
% matrix, contained in u
[u,S,v]=svd(Data,'econ');

% SS is a vector containing the singular values
SS = diag(S);

% If needed, we switch a given eigenconnectivity pattern so that its
% maximal value is positive. Because they index a principal direction,
% the sign of the patterns is arbitrary (it combines with the sign of the
% weights, stored in v)
for iter=1:size(u,2),
    [~,idx]=max(abs(u(:,iter)));
    if sign(u(idx,iter))<0,
        u(:,iter)=-u(:,iter);
        v(:,iter)=-v(:,iter);
    end;
end;
```

Q.6: Provide a definition of the clustering coefficient and of the degree metrics computed here. Do they inform on local or on global brain connectivity?

Q.7: For a few regions of your choice and default parameters, plot the graph metrics time courses over time; what do you observe? What conclusion can you make?

Q.8: Do the same for different values of edge density; what do you observe if edge density becomes very low? What happens if it becomes very large?

Q.9: Assuming that subject-wise demeaning has been performed, what does an eigenconnectivity pattern represent? How can you interpret a large value for a given connection in this pattern? Do those answers change if demeaning is not performed anymore?

Q.10: Generate eigenconnectivity building blocks with default parameters, and comment on what you observe (in particular concerning the first eigenconnectivity patterns). Can you recognise any resting-state network?
Hint: You can use the myVisEigenconn(u(:,1:n_components),n_ROI) function to have a global look at the first n_components eigenconnectivity patterns together, and the function View_A for an individual assessment. To convert an eigenconnectivity vector u(:,i) into a 2D matrix format, you can use the function jVecToUpperTriMat(u(:,i),n_ROI).

Q.11: Generate eigenconnectivity building blocks without performing subject-wise demeaning. Do

the results change compared to **Q.10**? Why?

Q.12: How would you perform group comparison with the introduced analytical pipeline? Aside from the already computed variables, do you see any other metrics that could be extracted and used for the purpose?

Q.13: Can you provide a few limitations of the eigenconnectivity approach?

3 Generation of innovation-driven co-activation patterns

Here, we guide you through the implementation of the iCAP generation process from TA outputs. Because the TA step takes a few hours to converge, it has been run beforehand, and you are provided with its outputs (see [5] for details about this part of the pipeline). The script that you are required to fill can be found in the *BrainHack/Practice* folder, and is called *iCAPs_Tutorial_Brainhack_TOFILL.m*. The users that are not willing to go through the implementation steps can directly start from **Q.20**, using the functional script *iCAPs_Tutorial_Brainhack_FULL.m* found in the same folder.

3.1 Data loading and parameters

The data required for this part of this analysis are the innovation signal matrix Innovation and the indicator matrix mask1 for the considered subjects. In contrast to the previous pipeline, the data used here is voxelwise. The variables mask and fHeader are also loaded, but will not be useful for the implementation (they play a role in plotting the results).

```
% Cell array: each cell is n_GM_voxels x n_timepoints. For each location and
% time point, the mask contains +1 for a positive innovation, -1 for a
% negative innovation, and 0 for no innovation
load('mask1.mat')

% Innovation signal: each cell is n_GM_voxels x n_timepoints.
load('Innovation.mat')

% Logical vector of size n_voxels x 1 indexing the elements belonging to
% gray matter with +1. Used for plotting the iCAPs.
load('mask.mat');

% Header containing information about the considered data (dimensions,
% etc.). Used for plotting the iCAPs
load('fHeader.mat');
```

Some of the parameters will be kept fixed throughout: apart from information on the dataset, they stand for some analytical steps (the number of separate times that k-means clustering is run n_folds and the type of distance used for this purpose DistType) or plotting requirements (number of slices to plot on each row of the iCAP figures n_slices_x and ranges of values to display Range_Pos and Range_Neg) that we recommend to leave unchanged.

```
% Number of subjects in the dataset
n_subjects = length(Innovation);
```

```
% Number of time points per subject
n_timepoints = size(Innovation{1},2);

% Number of gray matter voxels
n_GM_voxels = size(mask1{1},1);

% Type of distance to use for k-means clustering
DistType = 'cosine';

% Number of folds for which to run k-means clustering
n_folds = 3;

% Number of slices to display in each row of the iCAP plots
n_slices_x = 6;

% Ranges for the positive and negative values to display in iCAPs plotting
Range_Pos = [1.5,4];
Range_Neg = [-1.5,-4];
```

Some parameters can be modified to influence the iCAP generation scheme. This includes the minimal number of voxels that must show innovation simultaneously so that a given frame is retained in computing the iCAPs n_voxels, and the number of iCAPs into which to separate the retained frames K.

There are also some plotting parameters that can be tuned if needed: typeslice denotes the type of brain slices to plot between *axial*, *sagittal* and *coronal* ones, and slice_range is the associated set of MNI coordinates at which the slices should be drawn. We provide suggested ranges in the code for a general assessment of the maps.

```
% Number of voxels that must show innovation together to retain a given
% frame for the clustering
n_voxels = 500;

% Number of clusters to separate the data into
K = 20;


% Type of slices to display for the iCAP plots: choose between 'axial',
% 'sagittal' and 'coronal'
typeslice = 'axial';

% Associated range of (MNI) values at which to draw the slices. Suggested
% values: for axial slices, -30:10:80; for coronal slices, -100:10:70; for
% sagittal slices, -70:10:70
slice_range = -30:10:80;
```

3.2 Frame selection process

The data to cluster into iCAPs will be contained in the matrix Data, which should ultimately be of size n_retained_frames × n_GM_voxels, and has to be filled iterating through all the subjects. A for loop scaffold is already included in the code, and should be modified appropriately as described

below (**Q14** to **Q19**).
Reminder: For the ones preferring to directly start from a functional script, please work from the *iCAPs_Tutorial_Brainhack_FULL.m* script and start from **Q20**.

```
% Data will contain our inputs for the k-means clustering process
Data = [];

% Looping through subjects...
for i=1:n_subjects

    % I has size n_timepoints x n_voxels and contains the innovations for
    % the considered subject
    I = Innovation{i}';

    %%%%%%% TO FILL %%%%%%%%%
end
```

Q.14: Within the loop, extract the information about which voxels and time points show significant positive and negative innovations for the assessed subject into a matrix of size n_timepoints $\times$ n_GM_voxels. Threshold this matrix so that it contains only 0 (no innovation or negative innovation) or +1 (positive innovation) elements.

Q.15: Extract the innovation data for the assessed subject in a matrix I_pos of size n_timepoints $\times$ n_GM_voxels, thresholded so that it only keeps non-null elements for the voxels and time points with positive innovation.

Q.16: Call the function check_interconnectedness(I_pos,fHeader,mask), so that the thresholded innovation matrix that you constructed is further processed to remove its isolated positive elements. In practice, a non-null voxel must be surrounded by at least 6 non-null neighbours to be retained for further computations.

Q.17: Create a final mask of size n_timepoints $\times$ 1 denoting the frames that must be retained as moments of positive innovation by +1, and the other time points by 0. This is done by checking whether there are at least n_voxels voxels showing significant positive innovation.

Q.18: Use the created mask to extract the thresholded positive innovation frames, and fill Data with them.

Q.19: Still within the loop, perform the same process for negative innovations, so that after going through each subject, Data is filled with the positive and negative thresholded innovation frames retained for this subject.
Hint: Remember to flip the sign of negative innovations into positive values!

Q.20: For each subject, using default parameters, compute the fraction of voxels/time points showing positive and negative innovations. What do you observe?

Q.21: For each subject, using default parameters, compute the fraction of frames showing positive

and negative innovations. What do you observe?

Q.22: For each subject, using default parameters, compute the fraction of frames showing both a positive and a negative innovation together. What is the meaning of a joint innovation?

Q.23: How do the three above measures vary as a function of the n_voxels threshold?

3.3 Generation of iCAPs

Now that the retained frames have been concatenated, k-means clustering is performed on the Data matrix to separate it into K different iCAP maps, using DistType as the distance measure. Z-scoring is then applied on the obtained iCAPs by the function ZScore_iCAPs, and finally, iCAP maps are sorted to appear in descending order of occurrence in the iCAPs_rear matrix, of size K $\times$ n_GM_voxels.

```
% K-means clustering to separate the data into K different iCAPs
[IDX,iCAPs] = kmeans(Data,K,'Distance',DistType,...
    'Replicates',n_folds);

% Z-scoring of the iCAPs
iCAPs_z = ZScore_iCAPs(iCAPs,Data,IDX);

% Rearranges the data and centroids and indices so that '1' denotes
% the cluster with most occurrences in the data
[Data_rear,IDX_rear,iCAPs_rear] = RearrangeTimeCoursesClassic(Data,...
    IDX,iCAPs_z);

% Plotting of all K iCAPs
PlotiCAPs(which('MNI152_T1_2mm_brain.nii'),iCAPs_rear(1:K,:),fHeader,...
    mask,Range_Pos,Range_Neg,n_slices_x,slice_range,typeslice);
```

Q.24 Generate iCAPs with default parameter values; can you recognise some resting-state networks?

Q.25 Try to vary the number of clusters into which to separate innovation frames. How is the quality of retrieved iCAPs affected by using a very high value? What about a very low one?

Q.26 Generate iCAPs with an increasing n_voxels value, going from less than a percent to around a fifth of all gray matter voxels. How are the retrieved iCAPs influenced by this parameter? Can the use of a lower or larger value be justified from a neurophysiological point of view?

Q.27 Can you provide a few possibilities for how group comparison could be performed using the TA/iCAPs pipeline? What metrics could you compare across groups?

Q.28 Do some of the maps that you find with default parameters show spatial overlap? Is this a feature that can be achieved by simpler disentanglement approaches, such as a spatial independent component analysis?

Q.29 Can you come up with a few drawbacks of the TA/iCAPs framework?

References

[1] Jonas Richiardi, Markus Gschwind, Samanta Simioni, Jean-Marie Annoni, Beatrice Greco, Patric Hagmann, Myriam Schluep, Patrik Vuilleumier, and Dimitri Van De Ville. Classifying minimally disabled multiple sclerosis patients from resting state functional connectivity. *Neuroimage*, 62(3):2021–2033, 2012.

[2] Nora Leonardi, Jonas Richiardi, Markus Gschwind, Samanta Simioni, Jean-Marie Annoni, Myriam Schluep, Patrik Vuilleumier, and Dimitri Van De Ville. Principal components of functional connectivity: A new approach to study dynamic brain connectivity during rest. *NeuroImage*, 83:937–950, dec 2013.

[3] Fikret Isik Karahanoglu and Dimitri Van De Ville. Transient brain activity disentangles fMRI resting-state dynamics in terms of spatially and temporally overlapping networks. *Nature Communications*, 6:7751, 2015.

[4] Nathalie Tzourio-Mazoyer, Brigitte Landeau, Dimitri Papathanassiou, Fabrice Crivello, Olivier Etard, Nicolas Delcroix, Bernard Mazoyer, and Marc Joliot. Automated anatomical labeling of activations in spm using a macroscopic anatomical parcellation of the mni mri single-subject brain. *Neuroimage*, 15(1):273–289, 2002.

[5] Fikret Işık Karahanoğlu, César Caballero-Gaudes, François Lazeyras, and Dimitri Van De Ville. Total activation: fmri deconvolution through spatio-temporal regularization. *Neuroimage*, 73:121–134, 2013.