Microstate Analysis Tutorial

V0.2 – 05/08/2016

Arjun R. Khanna

Getting Started

- Download and install EEGLab from: http://sccn.ucsd.edu/eeglab/download.php
- Download, unzip, and add microstate analysis folders to MATLAB path from:
- Check the link above for updates, or give me your email address and I will email with update alerts.

Preprocessing

- The microstate analysis scripts use the native EEGLab “ALLEEG” structure.
- All preprocessing of data can be done in EEGLab.
- EEG resting-state microstates arise from predominantly alpha oscillations, so I typically filter <20-30 Hz.
- I also downsample data to ~200 Hz to reduce file sizes and speed up processing.
- What you absolutely need for microstate analysis:
  - All recordings must have the same number of channels – interpolate as necessary
  - Chanlocs must be present in the ALLEEG structure
  - Rejected epochs totally REMOVED from the data, not just marked
  - All filenames must be unique

Loading Sample Data

- In this tutorial, we'll use sample data from a study examining resting-state EEG from a set of Alzheimer's disease patients (AD) and old, healthy controls (OHC). Subjects had EEG recorded before (Pre) and after (Post) stimulation.
- Once you have EEGLab loaded, delete 'ALLEEG' from the workspace, and load the sample data, which will add an ALLEEG structure with the sample data. Then execute
  `eeglab redraw`
- Open the ALLEEG structure in MATLAB to explore its structure…

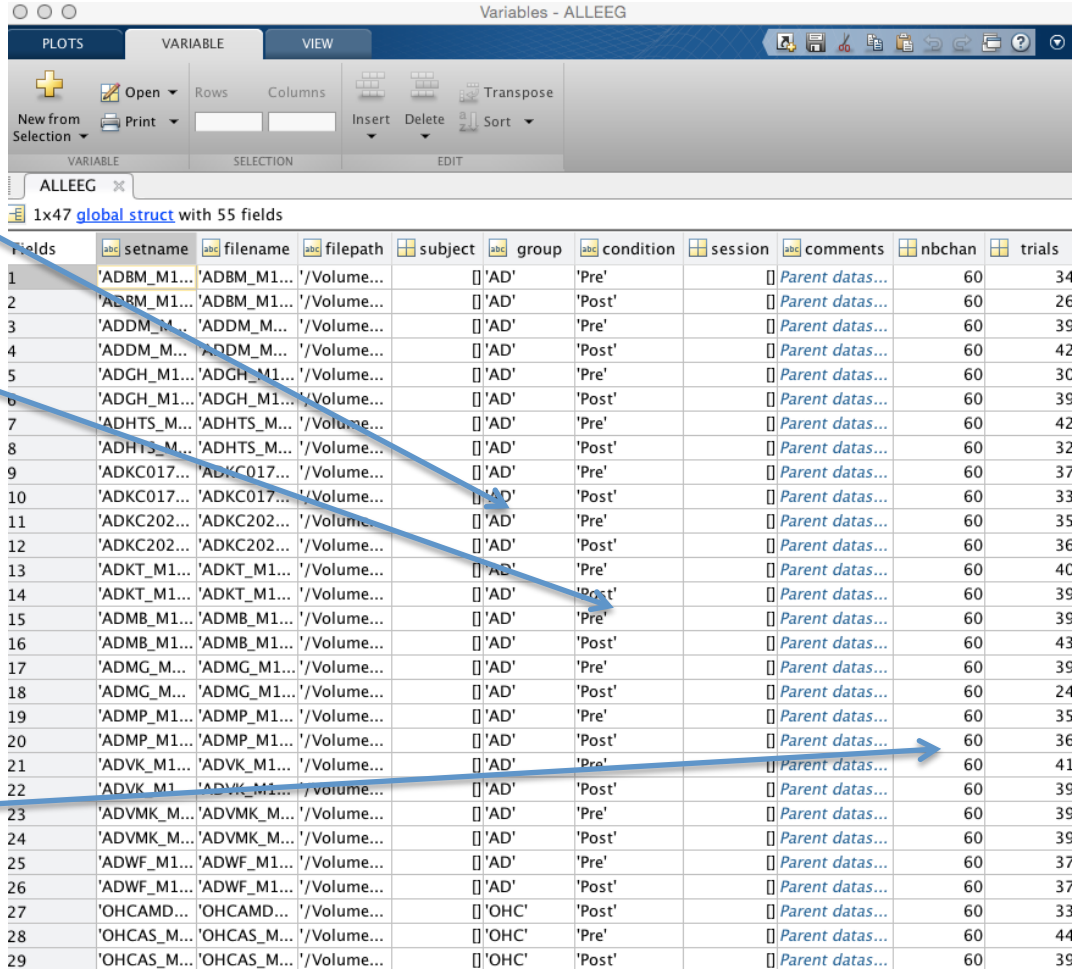
Sample Data ALLEEG

Group labels ‘AD’ or ‘OHC’ in ALLEEG.group field

Condition labels ‘Pre’ or ‘Post’ in ALLEEG.condition field

Group and Condition labels inserted as shown are required if you want to perform analysis by group or condition (or both).

All recordings must have the same number of channels (interpolate them using EEGLab if necessary) and chanlocs must be present

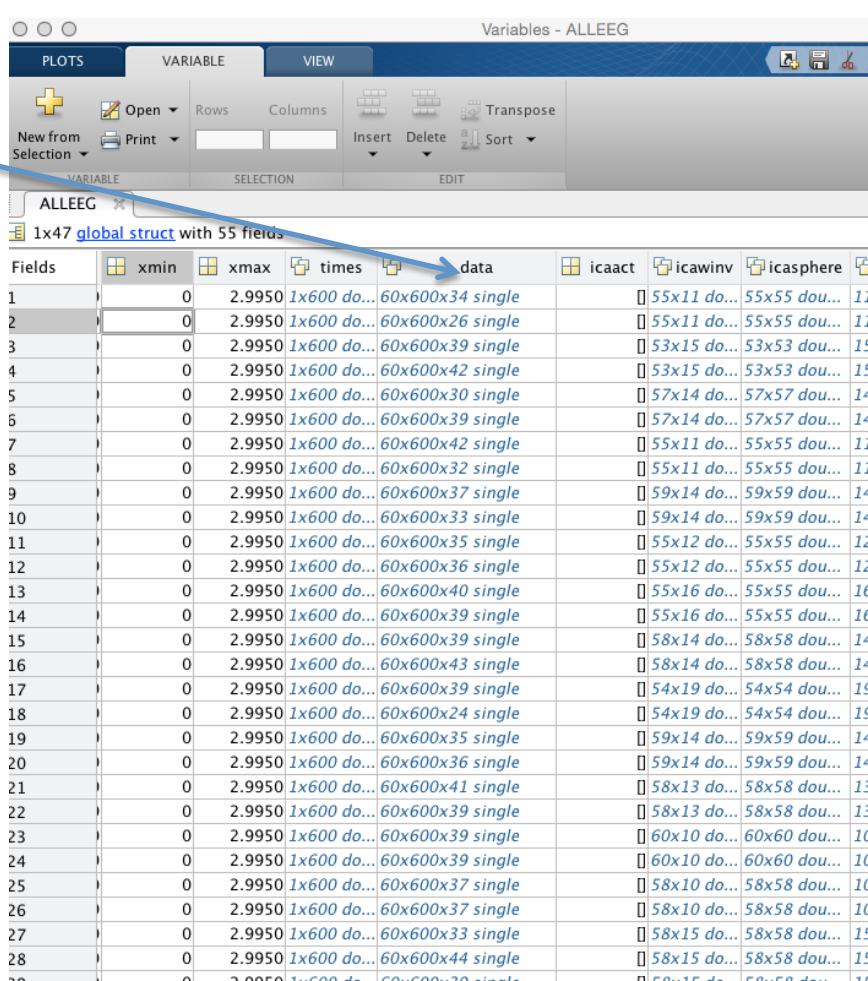


Fields	setname	filename	filepath	subject	group	condition	session	comments	nbchan	trials
1	'ADBM_M1...	'ADBM_M1...	'/Volume...	[]	'AD'	'Pre'	[]	Parent datas...	60	34
2	'ADBM_M1...	'ADBM_M1...	'/Volume...	[]	'AD'	'Post'	[]	Parent datas...	60	26
3	'ADDM_M...	'ADDM_M...	'/Volume...	[]	'AD'	'Pre'	[]	Parent datas...	60	39
4	'ADDM_M...	'ADDM_M...	'/Volume...	[]	'AD'	'Post'	[]	Parent datas...	60	42
5	'ADGH_M1...	'ADGH_M1...	'/Volume...	[]	'AD'	'Pre'	[]	Parent datas...	60	30
6	'ADGH_M1...	'ADGH_M1...	'/Volume...	[]	'AD'	'Post'	[]	Parent datas...	60	39
7	'ADHTS_M...	'ADHTS_M...	'/Volume...	[]	'AD'	'Pre'	[]	Parent datas...	60	42
8	'ADHTS_M...	'ADHTS_M...	'/Volume...	[]	'AD'	'Post'	[]	Parent datas...	60	32
9	'ADKC017...	'ADKC017...	'/Volume...	[]	'AD'	'Pre'	[]	Parent datas...	60	37
10	'ADKC017...	'ADKC017...	'/Volume...	[]	'AD'	'Post'	[]	Parent datas...	60	33
11	'ADKC202...	'ADKC202...	'/Volume...	[]	'AD'	'Pre'	[]	Parent datas...	60	35
12	'ADKC202...	'ADKC202...	'/Volume...	[]	'AD'	'Post'	[]	Parent datas...	60	36
13	'ADKT_M1...	'ADKT_M1...	'/Volume...	[]	'AD'	'Pre'	[]	Parent datas...	60	40
14	'ADKT_M1...	'ADKT_M1...	'/Volume...	[]	'AD'	'Post'	[]	Parent datas...	60	39
15	'ADMB_M1...	'ADMB_M1...	'/Volume...	[]	'AD'	'Pre'	[]	Parent datas...	60	39
16	'ADMB_M1...	'ADMB_M1...	'/Volume...	[]	'AD'	'Post'	[]	Parent datas...	60	43
17	'ADMG_M...	'ADMG_M1...	'/Volume...	[]	'AD'	'Pre'	[]	Parent datas...	60	39
18	'ADMG_M...	'ADMG_M1...	'/Volume...	[]	'AD'	'Post'	[]	Parent datas...	60	24
19	'ADMP_M1...	'ADMP_M1...	'/Volume...	[]	'AD'	'Pre'	[]	Parent datas...	60	35
20	'ADMP_M1...	'ADMP_M1...	'/Volume...	[]	'AD'	'Post'	[]	Parent datas...	60	36
21	'ADVK_M1...	'ADVK_M1...	'/Volume...	[]	'AD'	'Pre'	[]	Parent datas...	60	41
22	'ADVK_M1...	'ADVK_M1...	'/Volume...	[]	'AD'	'Post'	[]	Parent datas...	60	39
23	'ADVMK_M...	'ADVMK_M...	'/Volume...	[]	'AD'	'Pre'	[]	Parent datas...	60	39
24	'ADVMK_M...	'ADVMK_M...	'/Volume...	[]	'AD'	'Post'	[]	Parent datas...	60	39
25	'ADWF_M1...	'ADWF_M1...	'/Volume...	[]	'AD'	'Pre'	[]	Parent datas...	60	37
26	'ADWF_M1...	'ADWF_M1...	'/Volume...	[]	'AD'	'Post'	[]	Parent datas...	60	37
27	'OHCAMD...	'OHCAMD...	'/Volume...	[]	'OHC'	'Post'	[]	Parent datas...	60	33
28	'OHCAS_M...	'OHCAS_M...	'/Volume...	[]	'OHC'	'Pre'	[]	Parent datas...	60	44
29	'OHCAS_M...	'OHCAS_M...	'/Volume...	[]	'OHC'	'Post'	[]	Parent datas...	60	39

Sample Data ALLEEG

Data is stored in a 3-D matrix of shape [channels x samples x epochs]

Rejected epochs must be totally REMOVED from the data, not just marked!!!

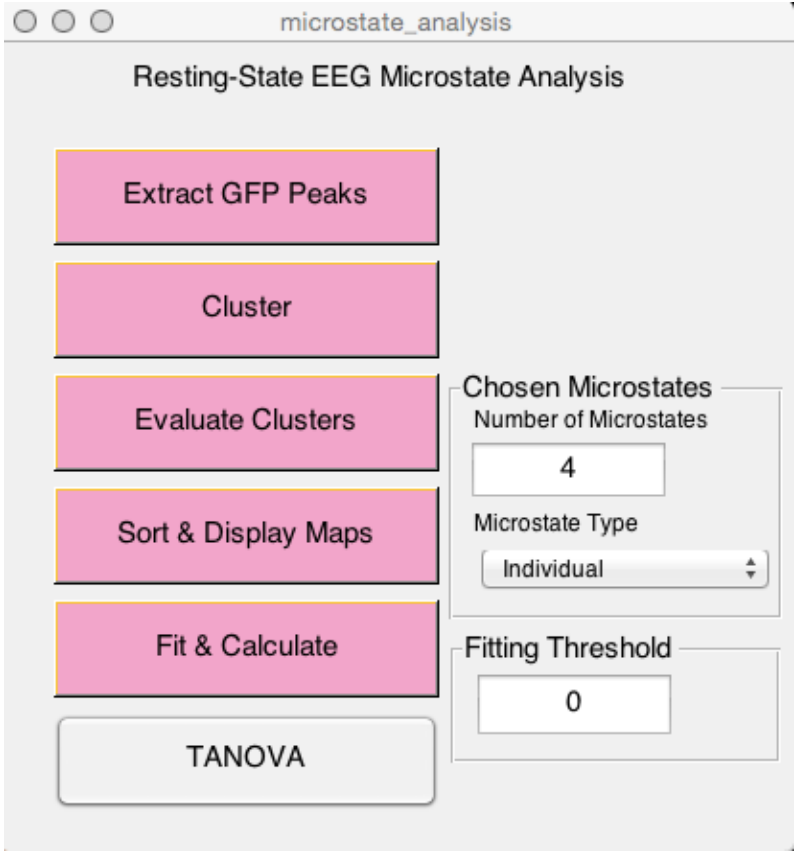


Starting Microstate Analysis

- Once data are loaded into the EEGLab ALLEEG structure and preprocessed as required, you are ready to start microstate analysis.
- Close the EEGLab GUI window; you don't need it anymore.
- Launch microstate analysis by executing `microstate_analysis`

Microstate Analysis

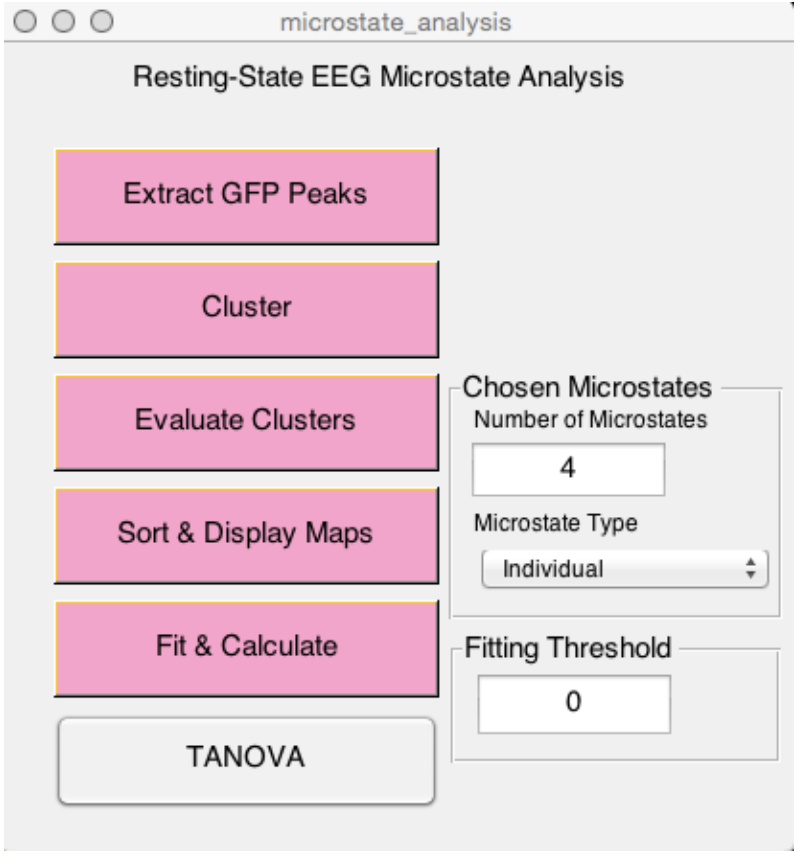
This GUI will pop up



Microstate analysis generally has 5 (or 6) steps, which are listed here in order:

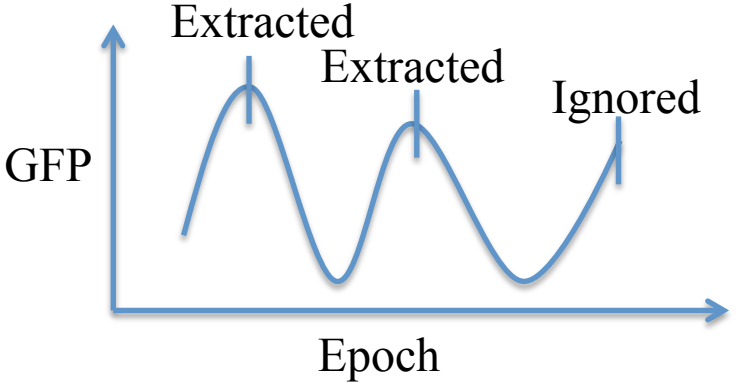
1. Extract GFP Peaks: to data-reduce, we select out the points at local maxima of the GFP curve; these have the highest SNR. These maps are called “original maps”.
2. Cluster: we submit the original maps to mathematical topographic clustering
3. Evaluate Clusters: Choosing how many clusters is most appropriate for the data can be difficult. Evaluation of clusters helps with this task.
4. Sort & Display Maps: In order to compare microstates across groups, there needs to be consistency in labeling.
5. Fit & Calculate: Finally, we fit the desired microstates onto the data and calculate features of interest.
6. TANOVA: If desired, topographic statistical analyses can be performed across groups.

1. Extract GFP Peaks

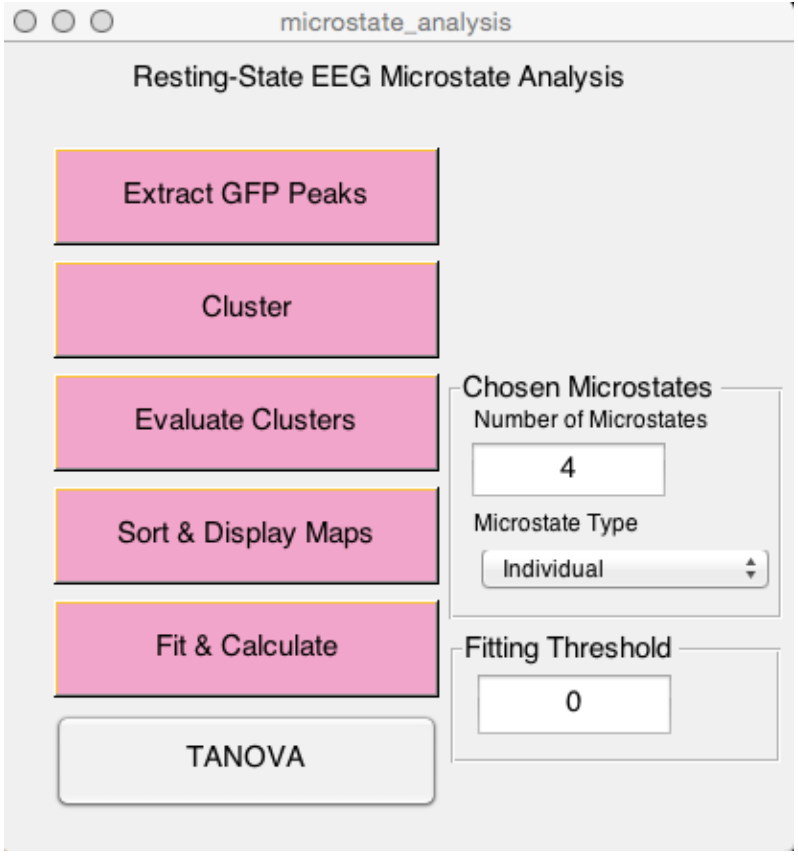


To extract GFP peaks, simply click "Extract GFP Peaks."

The GFP is equivalent to the standard deviation of the signal across all electrodes at any given instant. This script calculates the GFP at every time point and then extracts the local maxima of this curve. (If a local maxima occurs at the edge of an epoch, it is ignored.)



1. Extract GFP Peaks



The MATLAB command console updates you on its status.

```
File 36 of 47...Done.
File 37 of 47...Done.
File 38 of 47...Done.
File 39 of 47...Done.
File 40 of 47...Done.
File 41 of 47...Done.
File 42 of 47...Done.
File 43 of 47...Done.
File 44 of 47...Done.
File 45 of 47...Done.
File 46 of 47...Done.
File 47 of 47...Done.
Finished extracting GFP peaks.
```

The outputs are inserted into the ALLEEG structure as two new fields:

a	gfp_peaks	gfp_times
[]	1x34 cell	1x34 cell
[]	1x26 cell	1x26 cell
[]	1x39 cell	1x39 cell
[]	1x42 cell	1x42 cell
[]	1x30 cell	1x30 cell
[]	1x39 cell	1x39 cell
[]	1x42 cell	1x42 cell
[]	1x32 cell	1x32 cell
[]	1x37 cell	1x37 cell
[]	1x33 cell	1x33 cell
[]	1x35 cell	1x35 cell
[]	1x36 cell	1x36 cell

ALLEEG.gfp_peaks is a cell array of size [1 x epochs]. Each cell is a numerical array of size [channels x time points of GFP peaks] for that epoch.

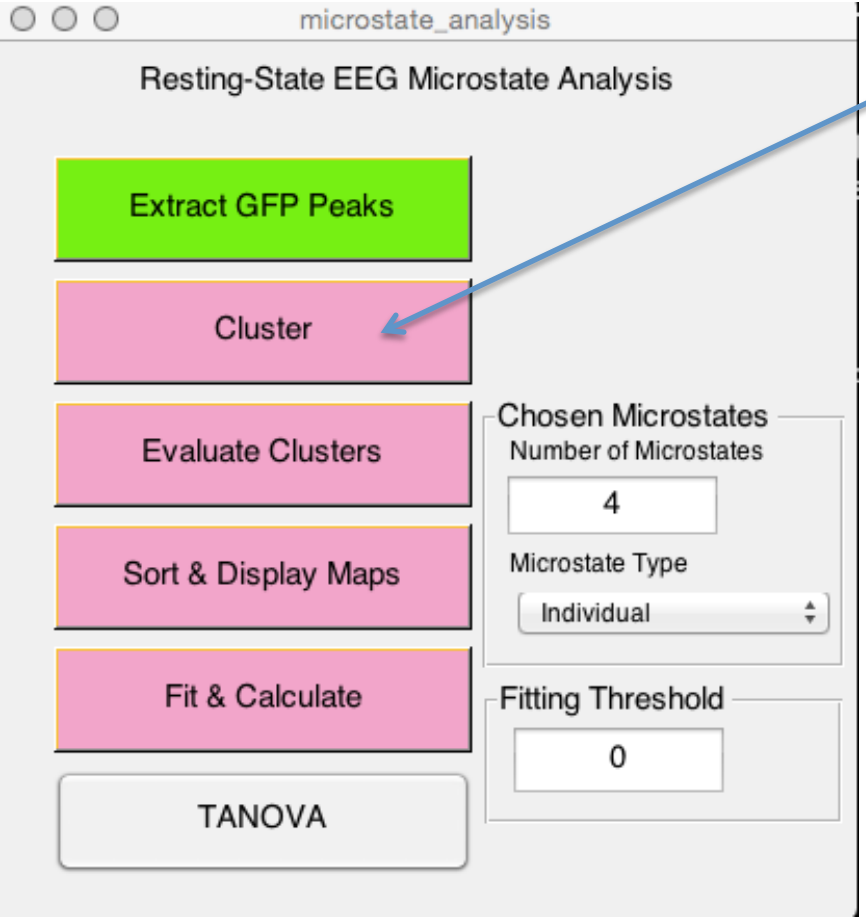
ALLEEG.gfp_times is a cell array of size [1 x epochs]. Each cell is a numerical array of size [1 x time point] for that epoch with time point labels of when each corresponding GFP peak occurs in the epoch. We need this time data to calculate the length of each microstate.

1. Extract GFP Peaks

- When the calculation is complete, the button will turn green.

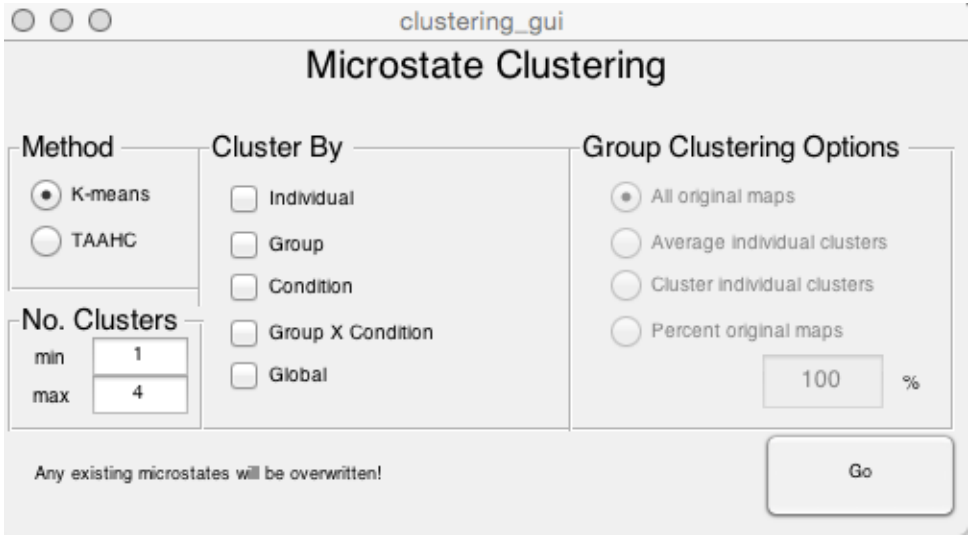
- This is for your bookkeeping purposes only.
- If you close the microstate_analysis GUI and re-launch it (with `microstate_analysis`), the button will be red again, but this doesn't matter.

2. Cluster

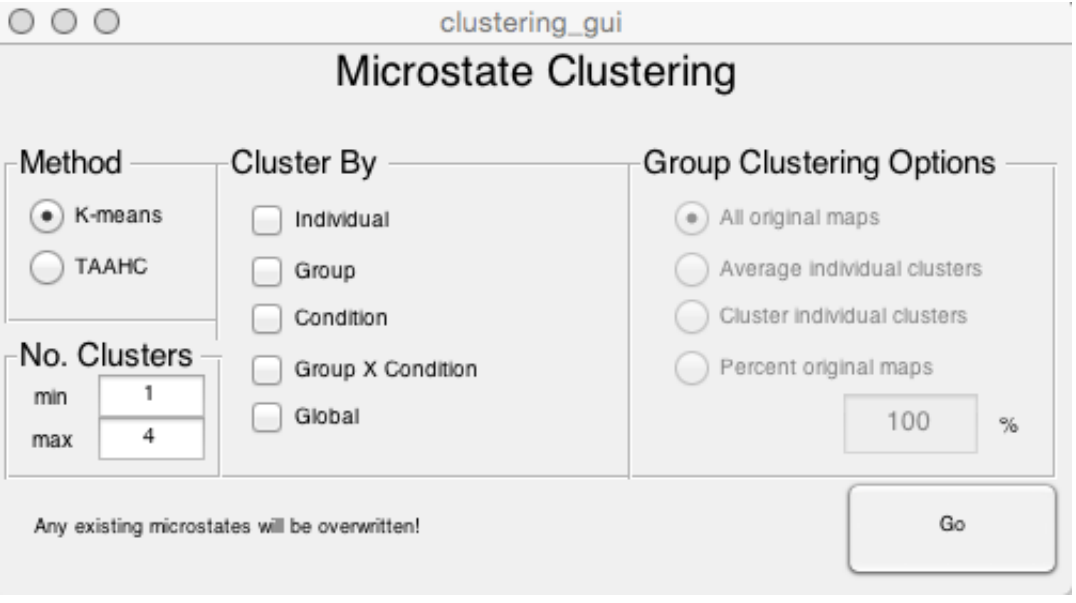


Click on ‘Cluster’

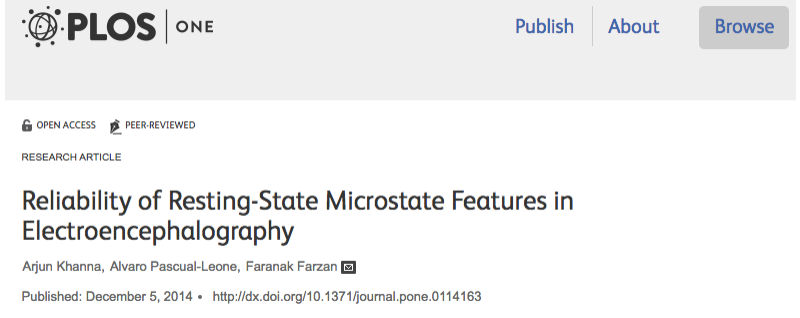
A new GUI will pop up:



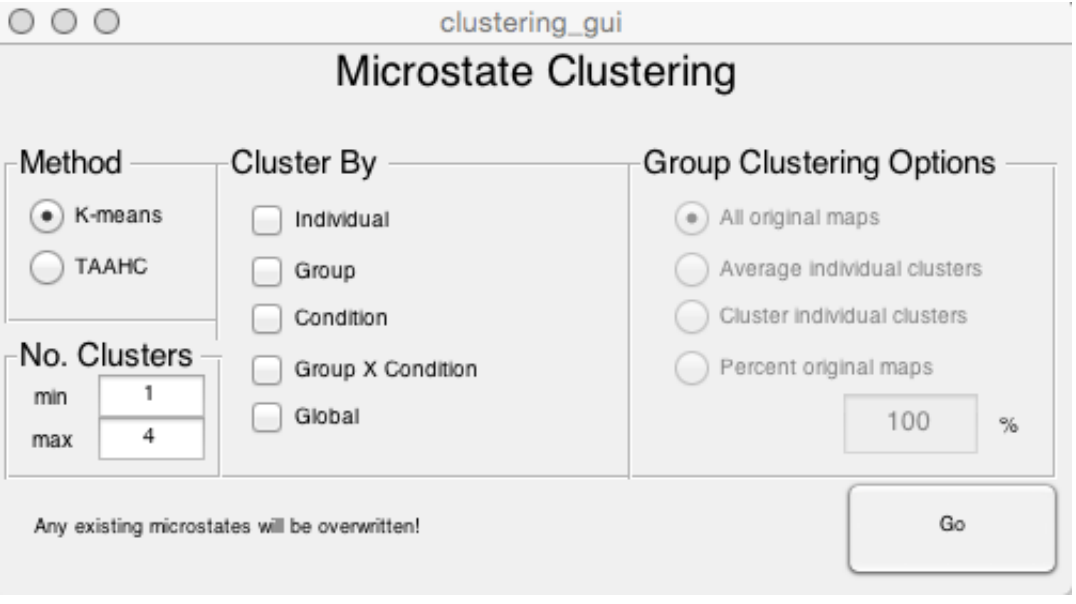
2. Cluster



METHOD refers to the clustering algorithm used. A (modified) k-means clustering algorithm or topographic atomize and agglomerative hierarchical clustering algorithm are available. For more technical details about these two methods, I refer you to the following paper:



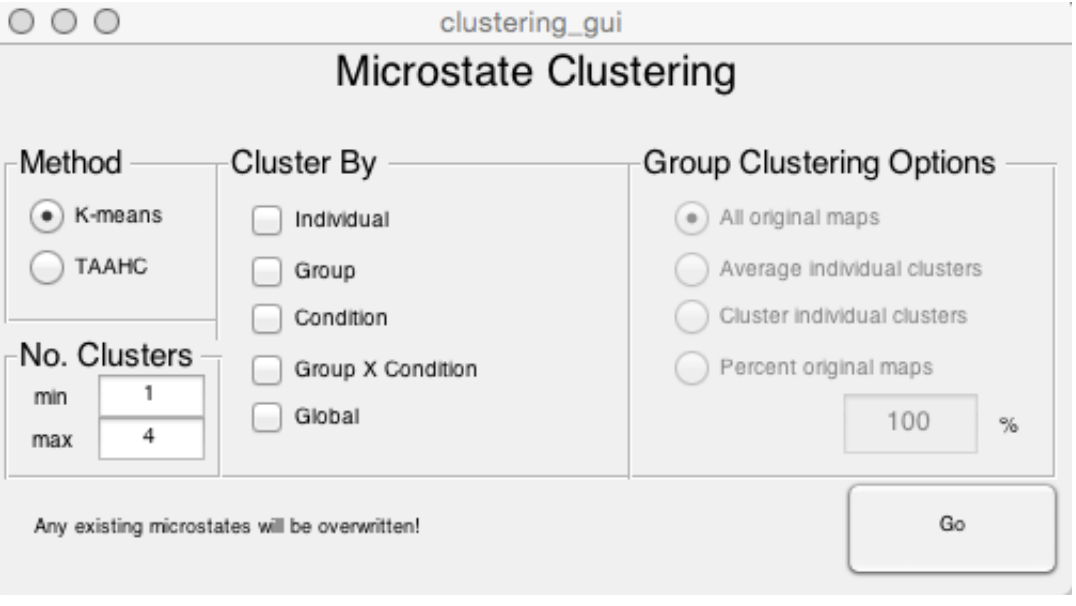
2. Cluster



Currently, TAAHC is *much* slower than k-means, too slow to be practical on most personal systems. For now, I suggest using K-means until a faster TAAHC is rolled out.

No. Clusters: This is the minimum and maximum number of clusters you want to look for in the data. You'll decide what the 'optimal' number of microstates for your data is later.

2. Cluster



Cluster By:

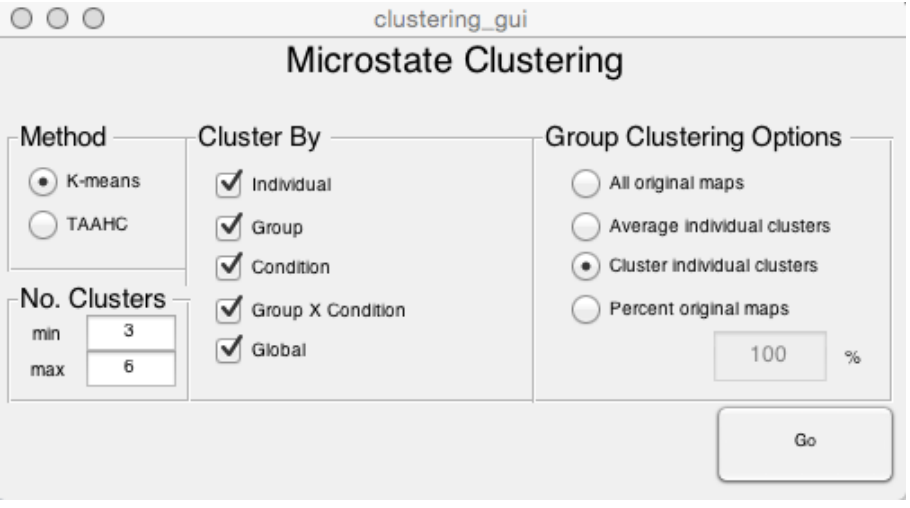
- Individual = Each file (row in ALLEEG) is clustered independently, generating *n* microstate maps separately for each file.
- Group = Each group (e.g. ‘AD’ and ‘OHC’ in the sample data) is clustered independently, generating *n* microstate maps separately for each group.
- Condition = Each condition (e.g. ‘Pre’ and ‘Post’ in the sample data) is clustered independently, generating *n* microstate maps separately for each condition.
- Group x Condition = The data are parsed into meta-groups using group and condition labels. Each file is labeled as ‘group_condition’, e.g. ‘AD_Pre’, ‘AD_Post’, ‘OHC_Pre’, and ‘OHC_Post’ in the sample data. Each group x condition is clustered independently; e.g. in the sample dataset, there are *n* clusters generated for each of the 4 group x conditions.
- Global = A single set of *n* microstates are generated for the entire dataset.

2. Cluster

When any of the group clustering options are selected (Group, Condition, Group x Condition, or Global) the ‘Group Clustering Options’ Panel becomes active.
Here, you must decide how the data will be compiled into each group/condition/etc.

- All original maps = with this option, all the original maps (GFP peaks) from each member of the group are combined and submitted to clustering. Note that this option does not require individual clusters to be found first. However, with large datasets, each group might end up having a huge number of points, which will slow down the analysis. Another important consideration is that if different subjects in the group have vastly varying amounts of data (e.g. one subject has 10 epochs while another has 40), this option will weight the group clusters toward subjects with more data (because they contribute more maps).
- Average individual clusters = in this option, individual microstate maps must be found first. Then, to create group/condition/etc maps, the individual maps from each member are first automatically sorted (explained later, in step 4) and then point-by-point averaged to create group/condition/etc maps.
- Cluster individual clusters = in this option, individual microstate maps must be found first. Then, to create group/condition/etc maps, the individual maps from each member are collected and submitted to another round of clustering. A potential issue is that if a group/condition/etc has very few members, there may not be sufficient maps to run reliable clustering.
- Percent original maps = with this option, a certain percent of original maps from each member of the group are randomly selected, aggregated, and submitted to clustering. The algorithm finds the subject with the fewest number of original maps, multiplies this number by the indicated percent, and then randomly selects this many maps from each file, sorting by group/condition/etc, to submit to clustering. This is a good way to ‘normalize’ against the issue presented above with ‘all original maps.’

2. Cluster



Here are the options I selected for the sample dataset:

- Using the k-means algorithm
- Looking for 3, 4, 5, and 6 clusters
- Cluster individually, by group (AD vs OHC), by condition (Pre vs Post), by group x condition (AD_Pre vs AD_Post vs OHC_Pre vs OHC_Post), and finally globally (everything in one big group).

- For group, condition, group x condition, and global maps, I want to find these by clustering the individual clusters from each member in the respective group/condition/groupxcondition/global.

2. Cluster

Clustering may take some time, depending on how large the dataset is.
The MATLAB console gives you updates on its status.
Occasionally, a warning is generated that a cluster has collapsed. This is usually because there are too few maps being clustered for the given number of clusters being sought. Usually, this is not a problem, and the algorithm keeps going.

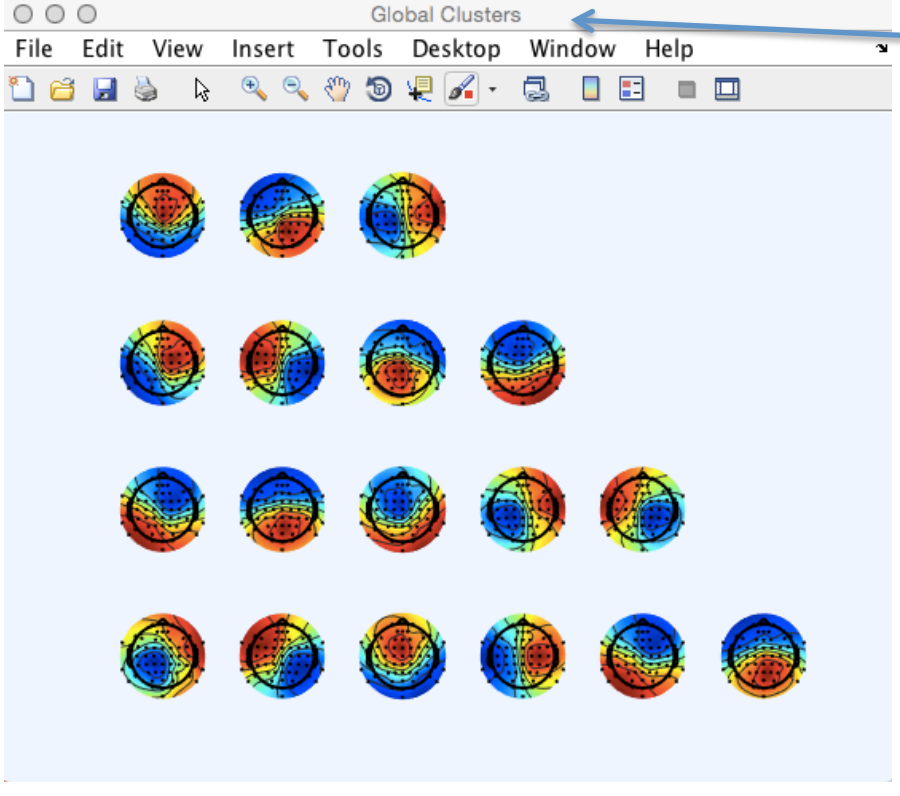
```
Clustering file 46 of 47.
Looking for 3 microstates...
>> Repetition 1: optimized in 24 iterations.
>> Repetition 2: optimized in 29 iterations.
>> Repetition 3: optimized in 31 iterations.
>> Repetition 4: optimized in 43 iterations.
>> Repetition 5: optimized in 25 iterations.
Elapsed time is 3.064026 seconds.
Looking for 4 microstates...
>> Repetition 1: clusters failed to converge after 500 iterations.
>> Repetition 2: clusters failed to converge after 500 iterations.
>> Repetition 3: optimized in 19 iterations.
>> Repetition 4: clusters failed to converge after 500 iterations.
>> Repetition 5: optimized in 35 iterations.
Elapsed time is 30.814503 seconds.
Looking for 5 microstates...
>> Repetition 1: optimized in 52 iterations.
>> Repetition 2: optimized in 48 iterations.
>> Repetition 3: optimized in 29 iterations.
>> Repetition 4: optimized in 43 iterations.
>> Repetition 5: optimized in 11 iterations.
Elapsed time is 4.302941 seconds.
```

```
Warning: Cluster has collapsed. Repeating this repetition. The ratio
of data points to clusters is too low.
> In find_microstates_kmeans (line 117)
  In clustering_gui>go_button_Callback (line 622)
  In gui_mainfcn (line 95)
  In clustering_gui (line 42)
  In matlab.graphics.internal.figfile.FigFile/read>@(hObject,eventdata
  In uiwait (line 81)
  In clustering_gui>clustering_gui_OpeningFcn (line 81)
  In gui_mainfcn (line 220)
  In clustering_gui (line 40)
  In microstate_analysis>cluster_Callback (line 159)
```

If this warning happens a lot, or if the algorithm appears to be hanging because this warning is endlessly appearing, consider choosing a different option for how to find group-level clusters.

2. Clustering

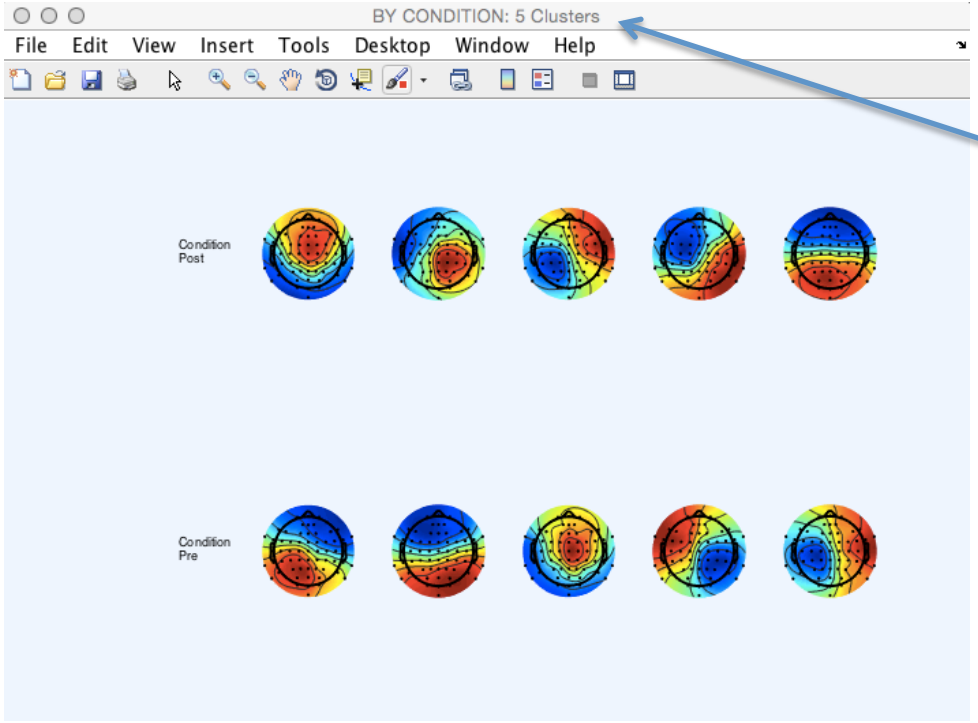
When clustering finishes, a number of figures are generated. These give you an idea of what your clusters look like.



Look at the title of the figure for what is depicted.

Recall that we asked the algorithm to look for 3-6 clusters. Here, the cluster maps for 3, 4, 5, and 6 clusters are displayed. They look nice! Note that for 4 clusters, we clearly see the 'archetypal' states A, B, C and D.

2. Clustering

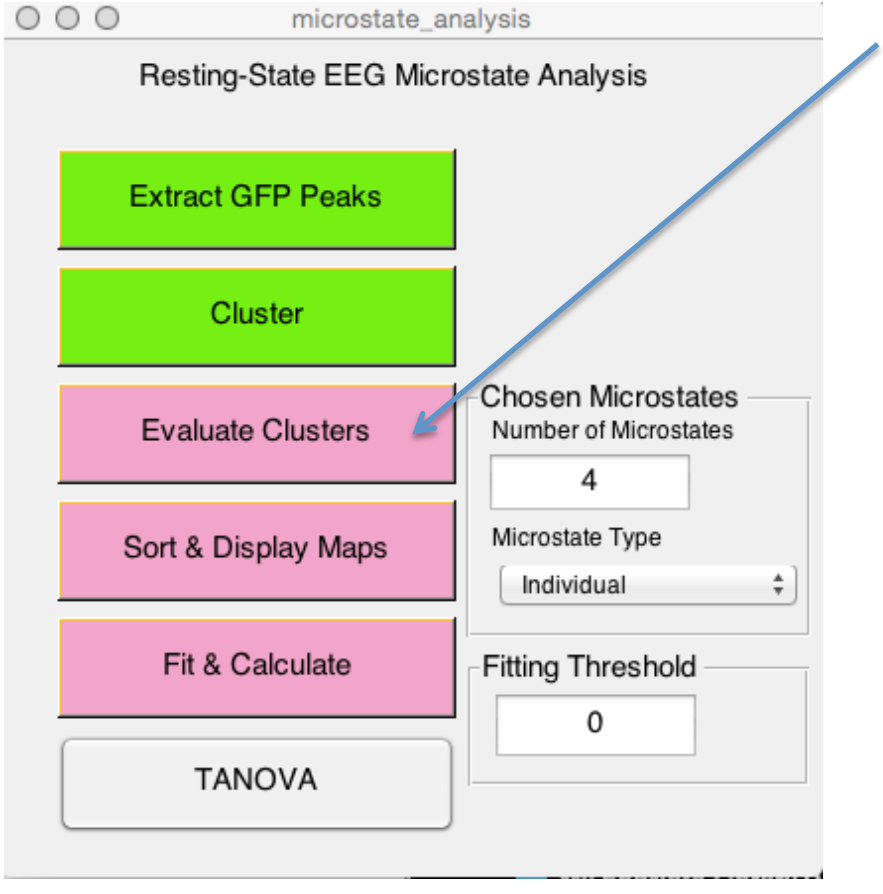


Here is another figure generated – showing what 5 clusters looks like when clustered by condition ('Pre' vs 'Post').

Note that the clusters are not sorted. In other words, cluster 1 in 'Post' in no way corresponds to cluster 1 in 'Pre'. Sorting of maps will happen later, in step 4.

Observing the maps directly can give you some idea of what the "optimal" number of maps might be for your dataset. When you are finished, close the clustering GUI.

3. Evaluate Clusters



The next step is to evaluate clusters. Click 'evaluate clusters.'

Recall that during clustering analysis we found 3, 4, 5, and 6 clusters. However, we have to choose which number of clusters is 'best' for our data.

It is important to always keep in mind that there is no standard accepted method of deciding what the optimal number of clusters for a given dataset is.

In fact, if you *a priori* decided to choose a certain number of microstates, you can skip this step altogether.

3. Evaluate Clusters

Evaluation of clusters can take some time.
All results are spit into the MATLAB console for viewing.
Evaluation is done separately for individual, group, condition, group x condition, and global microstates. For each of these, a table showing the global explained variance (GEV) and cross-validation criteria (CV) are shown.
Note: There might be an error in the calculation of GEV in v0.1, and although their relative values appears to be correct, their absolute values should not be relied upon at this time.
Minimization of the CV criteria is sometimes used to decide what the "optimal" number of clusters is.
All the results are shown in the MATLAB console as well as new fields in the ALLEEG structure.

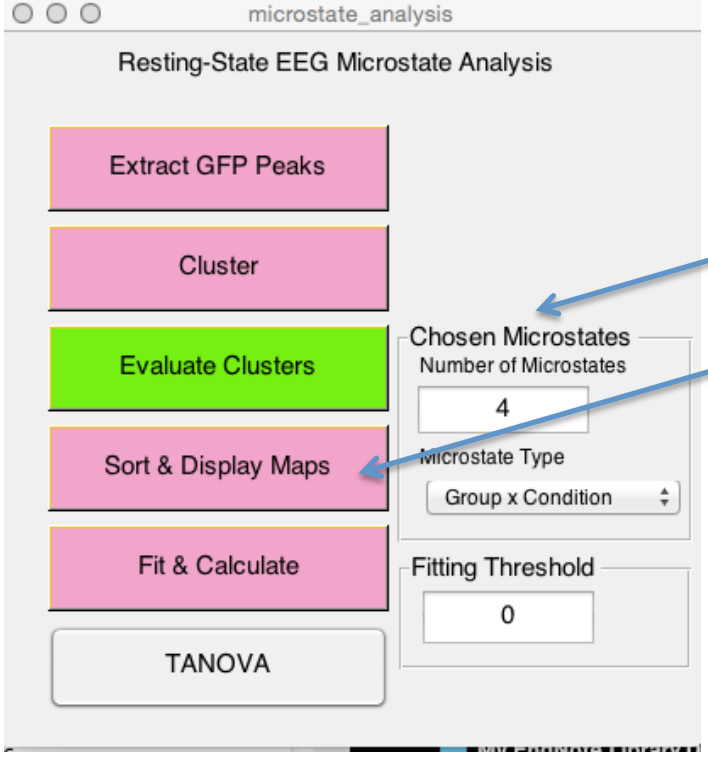
ADWF_M1V1_EC_S12_cleanCh.set	NaN	NaN	0.001446	0.0011748	0.0010185	0.00090816	6
ADWF_M1V1_EC_S12_cleanCh.set_2	NaN	NaN	0.0016472	0.0011957	0.001023	0.00097582	6
OHCAMD_M1V1_EC_S12_cleanCh.set	NaN	NaN	0.0012085	0.00084943	0.00075881	0.00070655	6
OHCAMD_M1V1_EC_S12_cleanCh.set_2	NaN	NaN	0.0011683	0.00080752	0.00066823	0.00057633	6
OHCAS_M1V1_EC_S12_cleanCh.set	NaN	NaN	0.0011525	0.00079785	0.00068591	0.00058044	6
OHCAS_M1V1_EC_S12_cleanCh.set_2	NaN	NaN	0.0015382	0.0012413	0.0010405	0.00092354	6
OHCCG_M1V1_EC_S12_cleanCh.set	NaN	NaN	0.0015169	0.0012008	0.00099969	0.00088754	6
OHCCG_M1V1_EC_S12_cleanCh.set_2	NaN	NaN	0.00090793	0.00079978	0.00063306	0.00059617	6
OHCCM_M1V1_EC_S12_cleanCh.set	NaN	NaN	0.00067867	0.00058497	0.00047567	0.000449	6
OHCCM_M1V1_EC_S12_cleanCh.set_2	NaN	NaN	0.0015138	0.0012062	0.00099636	0.00089921	6
OHCDM_M1V1_EC_S12_cleanCh.set	NaN	NaN	0.0015125	0.0011804	0.001032	0.00088423	6
OHCDM_M1V1_EC_S12_cleanCh.set_2	NaN	NaN	0.0016461	0.0012913	0.0011289	0.0010219	6
OHCEH_M1V1_EC_S12_cleanCh.set	NaN	NaN	0.0016693	0.0013555	0.0011453	0.0010366	6
OHCEH_M1V1_EC_S12_cleanCh.set_2	NaN	NaN	0.0014109	0.0010956	0.00093245	0.00082671	6
OHCGS_M1V1_EC_S12_cleanCh.set	NaN	NaN	0.0014814	0.0012117	0.0009791	0.00087492	6
OHCGS_M1V1_EC_S12_cleanCh.set_2	NaN	NaN	0.0010126	0.00079117	0.00069186	0.000589	6
OHCJF_M1V1_EC_S12_cleanCh.set	NaN	NaN	0.00096885	0.00076511	0.00064926	0.00060953	6
OHCJF_M1V1_EC_S12_cleanCh.set_2	NaN	NaN	0.001565	0.0012303	0.0010821	0.00091525	6
OHCRA_M1V1_EC_S12_cleanCh.set	NaN	NaN	0.0015011	0.0012026	0.0010272	0.00089125	6
OHCRA_M1V1_EC_S12_cleanCh.set_2	NaN	NaN	0.0016189	0.0012583	0.0010999	0.00098765	6
OHCRD_M1V1_EC_S12_cleanCh.set	NaN	NaN	0.0015932	0.0012941	0.0010989	0.00096649	6
OHCRD_M1V1_EC_S12_cleanCh.set_2	NaN	NaN	0.0012893	0.00099662	0.00089049	0.00081844	6
OHCSA_M1V1_EC_S12_cleanCh.set	NaN	NaN	0.0013667	0.0010181	0.00089443	0.00084007	6

In this sample, the mean CV minimum is at 6 clusters.

4. Sort & Display Maps

- By now, you should have chosen how many clusters you want to pursue, and what kind of maps (individual / group / condition / group x condition / global).
- Recall that the clustering algorithm generates maps in no particular order.
- However, to make comparisons within microstate classes (e.g. average life of microstate A is shorter in group 1 vs group 2) the maps need to be sorted so that there is consistency in the labels A, B, C, D, etc…

4. Sort & Display Maps

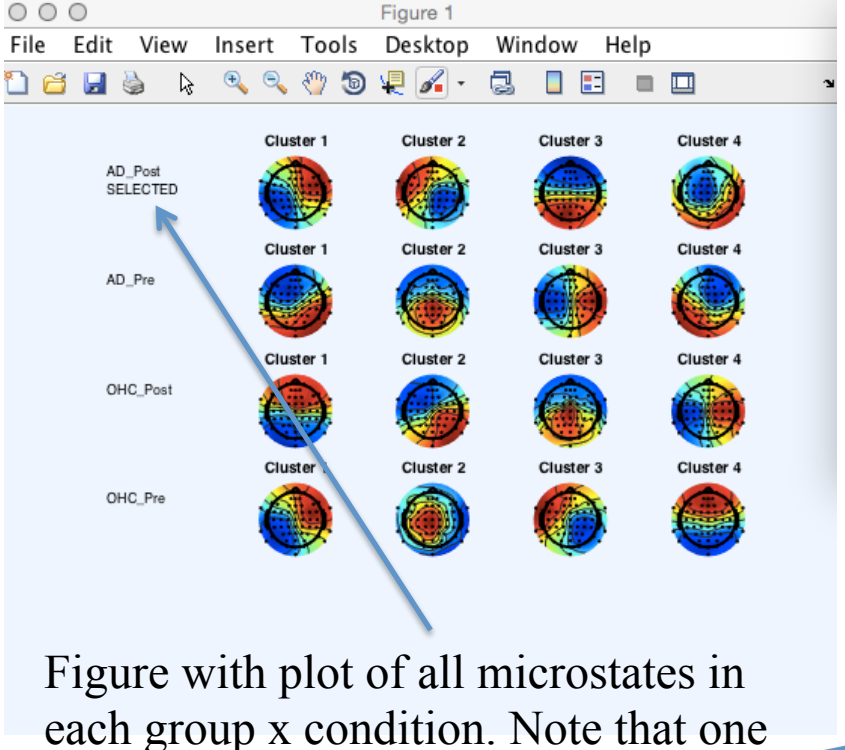


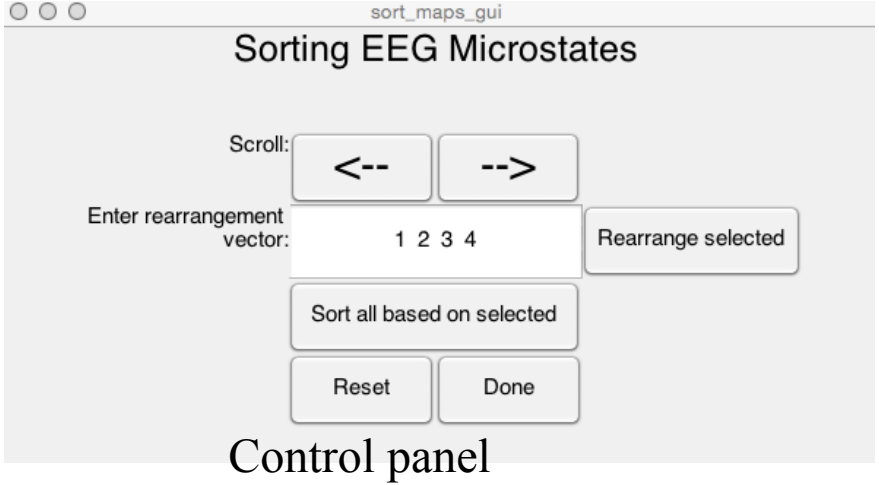
For the sample data, I decided to use 4 microstate maps, and to use the group x condition maps (AD_pre, AD_post, OHC_pre, OHC_post).

Select these options in “Chosen Microstates” panel

Click “Sort & Display Maps”

4. Sort & Display Maps



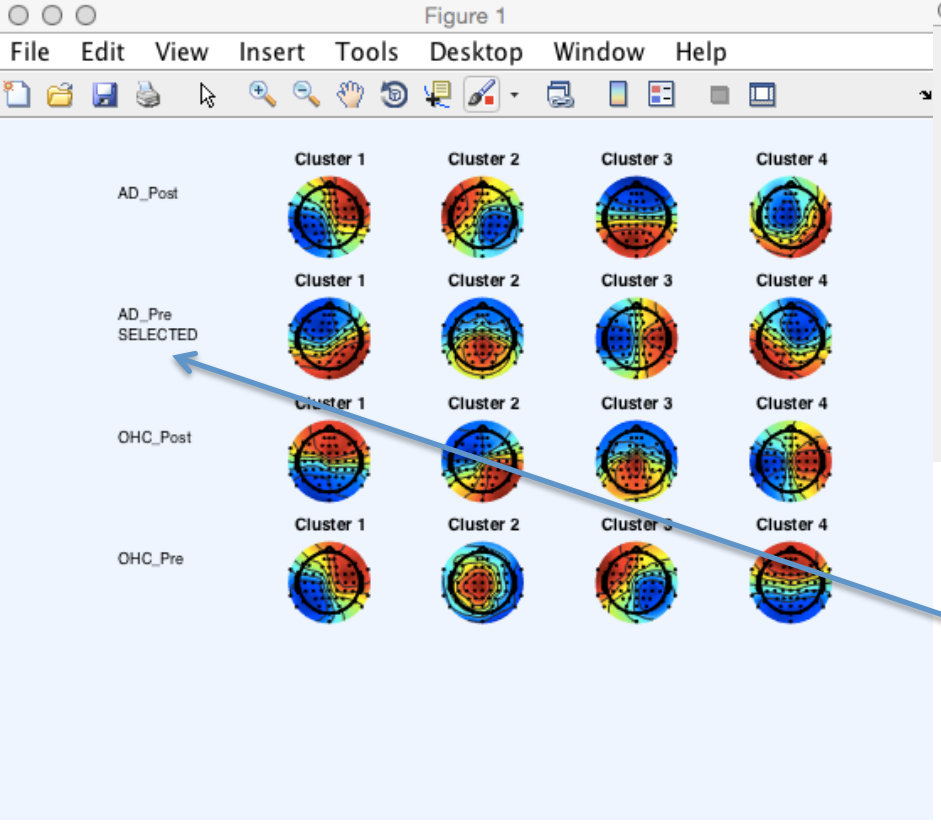


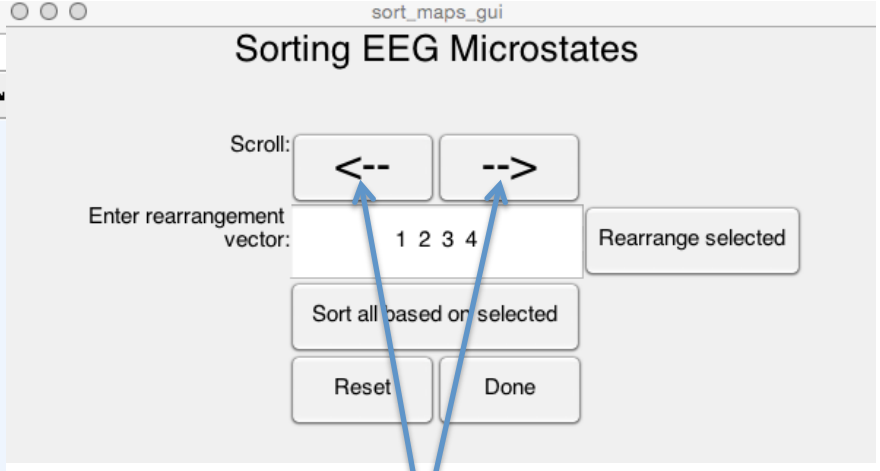
Control panel

Figure with plot of all microstates in each group x condition. Note that one of them is 'SELECTED'.
Also note that a maximum of 5 sets of maps (rows) are shown at a time.

Two new GUIs appear.

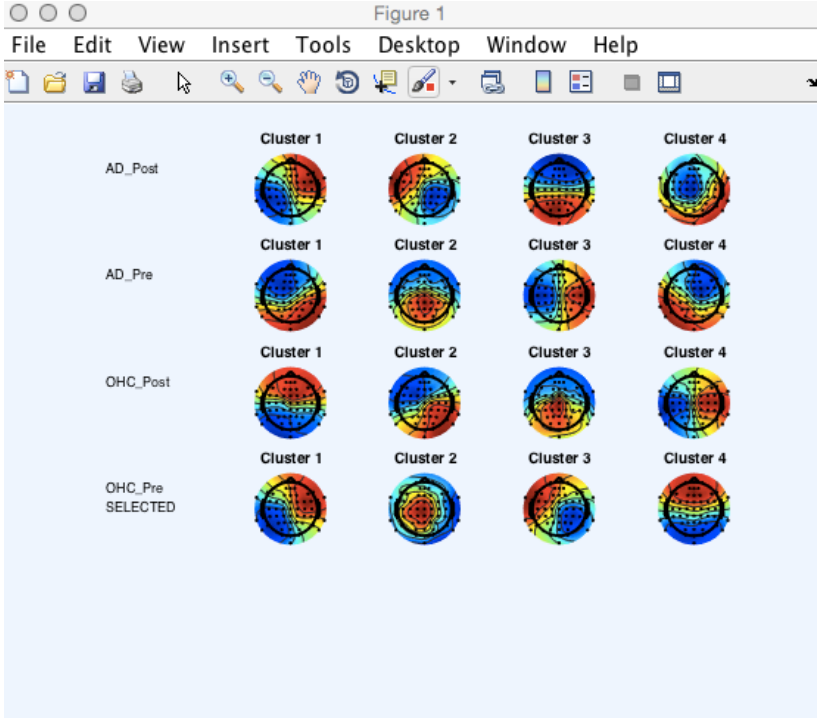
4. Sort & Display Maps

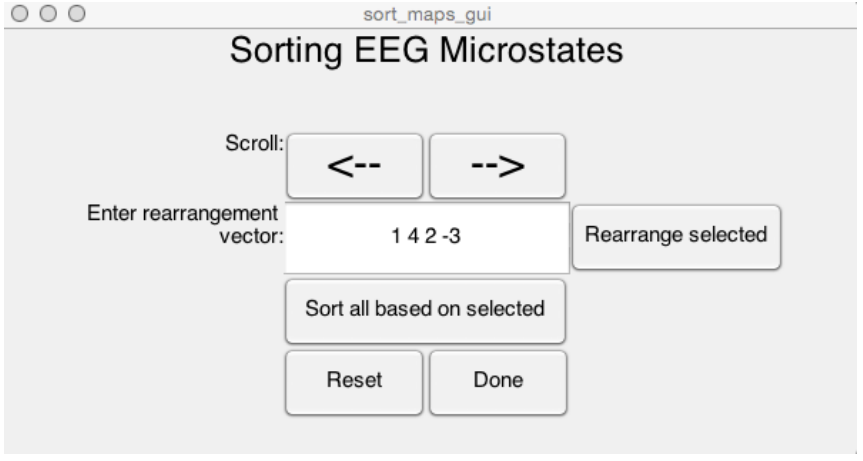




Pressing forward or backward in ‘SCROLL” changes which set of maps is ‘SELECTED’

4. Sort & Display Maps

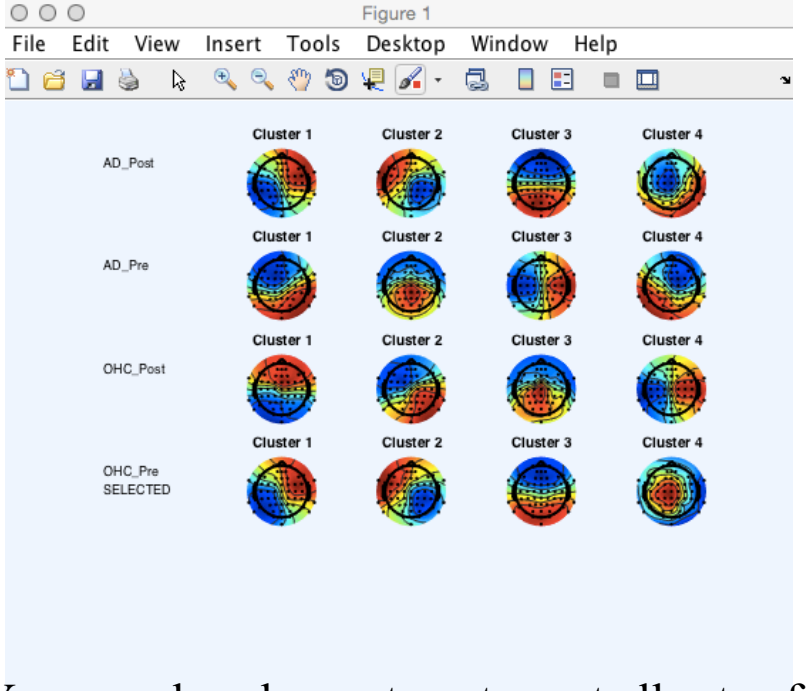


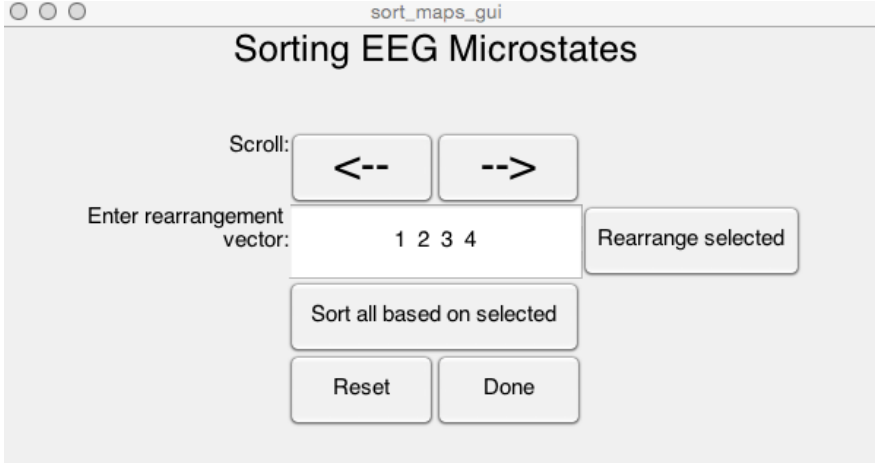


The rearrangement vector allows you to manually rearrange the SELECTED maps.

For example, I want to rearrange the "OHC_Pre" maps to Cluster 1 = A, Cluster 2 = D, Cluster 3 = B, and Cluster 4 = inverted C. To do so, scroll to have OHC_Pre selected and then enter the following rearrangement vector: 1 4 2 -3. Click rearrange selected.

4. Sort & Display Maps

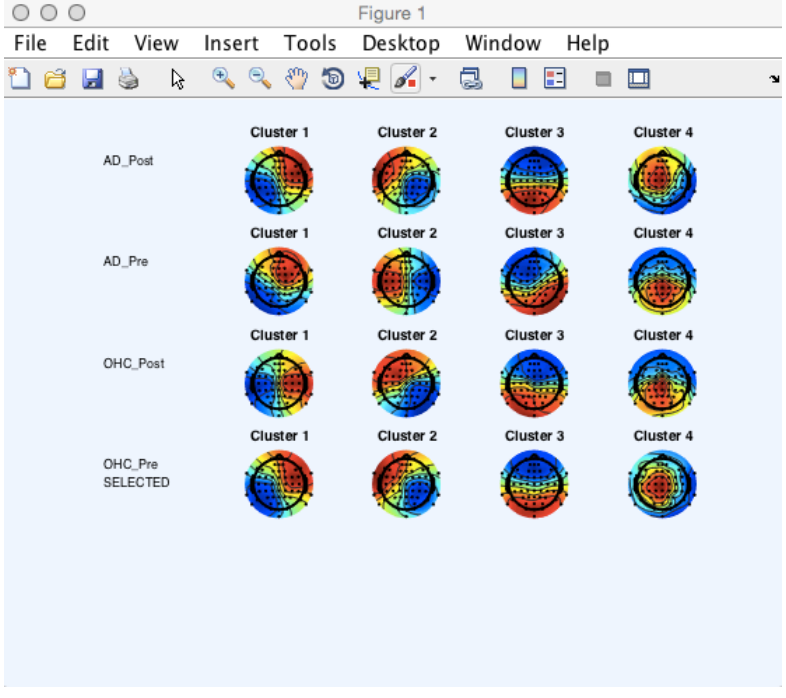




The OHC_Pre maps are now rearranged.

You can also choose to auto-sort all sets of maps using the selected maps as a template. The way this script works is by trying all possible rearrangements of each row, calculating the sum of correlation between each map and the corresponding template map, and choosing the rearrangement which yields the highest total correlation.

4. Sort & Display Maps



Here, all groups have been auto-sorted using the selected OHC_Pre maps as the template.

Important note: The auto-sort works most of the time, but occasionally will identify a rearrangement that, although maximizes summed correlation, is not visually correct. Therefore, I recommend routinely visually inspecting all maps after auto-sort and using rearrangement vectors to rearrange those rows that require fixing.

4. Sort & Display Maps

Another important point: Recall that for identifying group/condition/groupxcondition/global maps, we had the option of using “Average individual clusters” to identify group-level maps. To perform point-by-point averaging of individual maps, we need the individual maps to be sorted. The clustering algorithm uses auto-sort to sort the individual maps before averaging. However, as I just explained, the auto-sort is sometimes wrong.

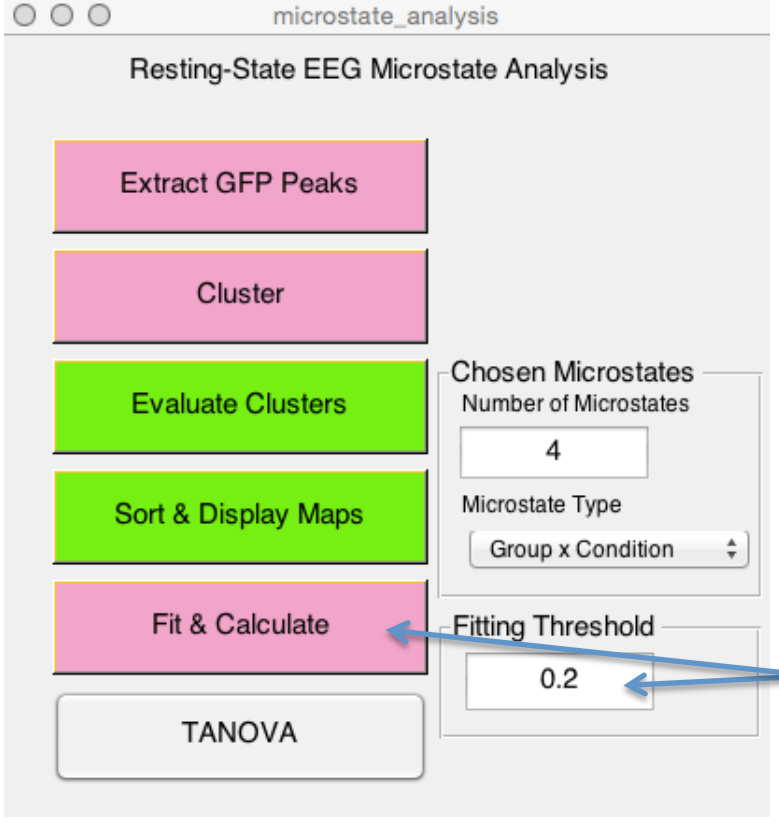
Therefore, if I want to use “average individual clusters” to identify group-level maps, do the following:

- Cluster individual maps ONLY
- Sort & Display individual maps to make sure the sorting of individual maps is appropriate
- Then click on “cluster” again and cluster group/condition/groupxcondition/global maps using the “average individual clusters” option. Do NOT select individual clusters this time, or else the sorted maps will be overwritten with new individual maps.

4. Sort & Display Maps

- Once you are done sorting, click ‘Done’
- If you want to erase all the sorting you have done in the session so far, click ‘Reset’

5. Fit & Calculate



Finally, we fit the chosen microstates onto the data and calculate features of interest.

We can also apply a fitting threshold if we want. Each original map will be labeled A, B, C, or D depending on which template map it most correlates to. If the "best' correlation is below the fitting threshold, that map is called 'unfit' and is not used in the analysis.

Simply enter in a fitting threshold and click 'fit & calculate.'
You will be prompted to identify a directory to save results in and a filename, which should end in "name.csv". Results are also stored in the ALLEEG structure.

TANOVA

In order to use TANOVA, you MUST have sorted INDIVIDUAL maps!!!

Click on ‘TANOVA’ and follow the instructions in the GUI.

You can compare maps between any groups, conditions, or group x conditions that exist in the data.

P-Values are output into the MATLAB Console.

Features Under Development

- Clustering:
  - Faster TAAHC
  - New clustering method using nonlinear analysis (recurrence-based method)
- Evaluation:
  - Correcting any issues in GEV calculation
  - K-L criteria
- Sort & Display:
  - More accurate auto-sort
- Fit & Calculate:
  - Adding transition counts & probabilities