Introduction

This document gives details regarding the **CREx_ICA_calc()** function. This function allows the user to carry out Independent Component Analysis (ICA) on EEG data and gives the user the choice of carrying out a Principle Component Analysis (PCA) to reduce the dimensionality of the data entered into the ICA. In addition, it applies functions from the Adjust toolbox (Mognon et al., 2011) to automatically detect ICs (Independent Components) that correspond to artifacts using both spatial and temporal features of the components. The Adjust functions were adapted to work when ICA has been calculated on data whose dimensionality has been reduced using PCA.

This script was programmed to facilitate the use of PCA with ICA with the use of the ADJUST toolbox for the automatic detection of ICs that correspond to artifacts, and that, therefore are candidates for rejection. Of course, the EEGLAB functions can also be used, in particular for the visualization of the time-course of the ICs.

Requirements:

1. EEGLAB (Delorme, 2004), mainly for the:
   - *runica()* function,
   - *visualization of component activity as topography,*
   - *loading and saving data (as *.set files).*
2. ADJUST toolbox (Mognon et al, 2011), an EEGLAB plugin, for:
   - Automatic detection of composants corresponding to artifacts.
   - Visualisation of statistics resuming the spatial and temporal features of the components.
3. The following functions:
   - **CREx_ICA_calc()** function
     - Which calls the "**showICinfo()**" function to visualize the IC components and characteristics in detail.
     - The "**pop_prop_ADJ_crex()**", which is essentially the "**pop_prop_ADJ()**" of the ADJUST plugin that has been adapted to work in the case that PCA has been carried out before ICA.

The ADJUST plugin can be downloaded from: https://www.nitrc.org/frs/?group_id=739

Background:

To carry out the ICA, it uses the EEGLAB function, *runica()*. This function applies the **Infomax** (Information Maximization) algorithm, an optimization algorithm first applied to neural networks, which was applied to ICA by Bell & Sejnowski (1997). It identifies signals that are maximally independent by minimizing the mutual information or maximizing the joint entropy of components. One important thing to note about BSS methods in general, is that the distribution of the data plays an important role in the efficacity of the decomposition. In

the case of the Infomax algorithm, it works best for data with a super-gaussian distribution, such as eye-blinks. This implies that its performance will be compromised for a signal characterized by, say, large line noise (50Hz). However, in this script we run the extended version of the Infomax algorithm, which allows one to extract sub-gaussian signals, such as line noise.

Principal Component Analysis (PCA)

It does not decompose a signal into its independent components, but rather it separates signals that are uncorrelated or orthogonal. If we think in terms of matrices, we can say that PCA finds the direction that best explains the variance in the data. So, PCA will give us components that, rather than corresponding to the latent variables in our signals, will correspond to a combination of variables; it will create new combinations of uncorrelated signals…but just because the signals are uncorrelated, does not imply that they are independent. In addition, because PCA extracts signals that have a gaussian or normal distribution, applying PCA to neurophysiological signals would imply assuming that our latent variables are gaussian…and we can't do this…

However, PCA can be used in the context of ICA, as a means of reducing the dimensionality of the data. Not only can this speed up the calculation of the ICA, which can be practical if we are working on continuous data, it can also help reduce the redundancy. If we run ICA on 64-channel EEG data, it will yield 64 ICs. But, under the sound assumption that we don't necessarily have 64 latent variables, many of these ICs will not correspond to a pertinent underlying signal and so can be considered redundant. Reducing the dimensionality of our data prior to ICA can help reduce this redundancy.

Calculating the number of principle components

So, **how can we decide how many principal components to extract?** The approach used in the **CREx_ICA_calc()** function is to calculate the explained variance as a function of the number of principal components. Then we find the number of components required to explain 99%+ of the variance in the data. Figure 1 presents a graph of the cumulative summation of the explained variance plotted against the number of components. The 99% explained variance point is marked by a red dot and this corresponds to 34 principal components.

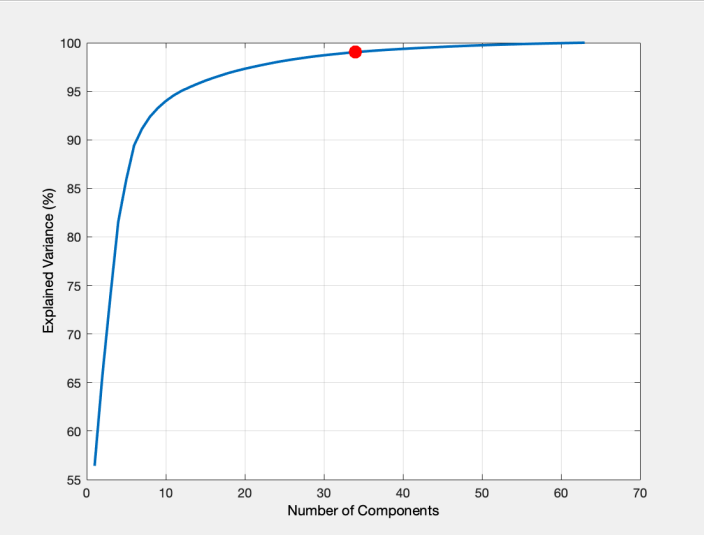

Figure 1: *The explained variance (%) as a function of the number of principle components. The cumulative sum of the explained variance (EV) is plotted. A red dot marks the 99% explained variance, which requires 34 components.*

Guidelines for using the CREx_ICA_calc() function

The following is an example of how to call the function and pass the necessary information to the function. The calling script of CREx_ICA_calc() is CREx_ICA_calc_Main().

```
% CREx_ICA_calc_Main().
% CALL OF FUNCTION TO CARRY OUT ICA ON CONTINUOUS DATA
% It uses functions from the ADJUST toolbox to detect thos ICs that
% correspond to artifacts.
% It can also apply a PCA before carrying out ICA to reduce the number of
components and speed up ICA computation for continuous data.
% The number of PCA components is calculated automatically based on
% explained variance (99% explained variance).

% Read in path information from a parameters text-file
paramfile_nom = 'parameters-myprojectname.txt';  % The title of the
parameters file.
% The path to the parameters path. This should be the only path that needs
% to be changed.
paramfile_path = fullfile(path_to_parameterfile,paramfile_nom)% Put the
path to your parameters file here.

fid2 = fopen(paramfile_path);     % Define a file identifier.
mydata = textscan(fid2,'%s %s'); % Scan text-file...

for i = 1:length(mydata{1,1})                      % generate a parameters
structure from the parameters text file
    Params.(genvarname(mydata{1,1}{i})) = mydata{1,2}(i);
end
currsuj = 'XXX';   % This should correspond to the name of the folder in
which the current subject's processed files are saved.

% Define options
dopca = 1; % or 0 if not doing PCA before ICA.
doICA = 1; % or 0 if not doing ICA calculation (and only rejection of
components).
doRej = 0; % or 1 if only rejecting components and carrying out back-
projection of retained components
```

SCRIPT FOR APPLYING ICA TO EEG DATA

```
doVis = 0; % or 0 if you do not want to visualise and data before and after
IC rejection.

% Call of function to calculate ICA
CREx_ICA_calc(dopca, doICA, doRej,doVis, Params, currsuj);
```

Paths:

The script is in current version assumes that the paths to the subject-level pre-processed data and the channel location file are stored in a text-file, called the "parameters-projname.txt". A template parameters text-file is provided. The data in this text-file is loaded into a structure variable called "**Params**". This structure is passed to the CREx_ICA_calc() function. If the paths are incorrectly defined, the script won't run.

The "**currsu**j" variable has to correspond to the name of the folder in which the pre-processed data for the current subject are saved. Generally, the folder name corresponds to the current subject title, for example, "S10MMN" etc…So the "currsuj" variable should correspond to the title of folder in which the data on which you want to carry out ICA is stored. When you run the script, the folder defined by "currsuj" will automatically open and you will be prompted to select the dataset (*.set file) on which you want to carry out the ICA. However, after ICA calculation, your dataset will be saved automatically to the same folder.

Defining Options:

The CREx_ICA_calc() script offers the following possibilities:

- Carry out ICA on the data: **doICA = 1.** If you don't want to carry out ICA, **doICA=0**.
- Carry out PCA before ICA: **doPCA = 1**. Note that doICA also has to be 1, PCA is not run without ICA. If you don't want to run PCA before ICA: doPCA=0.
- Reject ICA components that have already been marked for rejection: **doRej = 1**. If you only want to carry out ICA and reject the components later: **doReg=0**. Note that you can only do this if ICA has already been run and ICs to reject have been defined.
- Visualise the EEG data before and after the rejection of ICs: **doVis=1**. Note that this can only be carried out on data for which the ICs have been computed-an error message will appear. If you don't want to visualize because you need to run ICA first: **doVis=0**.

Running ICA…

As described earlier, when you chose to carry out PCA before ICA (doPCA=1) the number of principle components (PCs) required to explain 99% of the variance of the data is computed. Therefore, you have no need to arbitrarily choose the number of PCs. A graph of the explained variance (%) versus the number of PCs will appear (figure 1).

Once ICA has completed, a figure showing the topographies all X number of ICs will appear. Those components that are likely to correspond to artifacts, such as eye-blinks, have a title in red showing the component number and "Artifact" (figure 2).

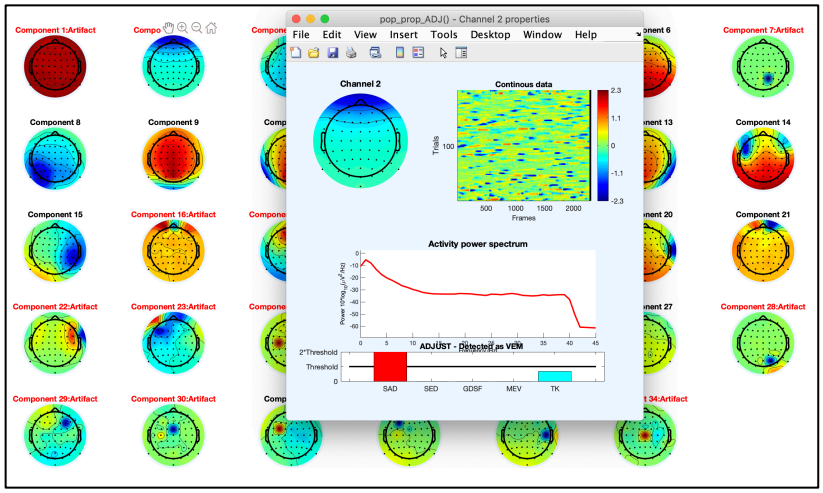

Figure 2: Main figure presenting the independent components' topographies. As PCA was carried out before ICA, there are 34 ICs instead of 64. By clicking below the topography of a component, a second window will open giving details on that component.

To get more information concerning an IC, click the space just below the topography of that component. A new figure will open presenting an erp-image of the component, its spectrum and an over-view of the spatial/temporal features that indicate that it is artifact-like or otherwise (figure 3).

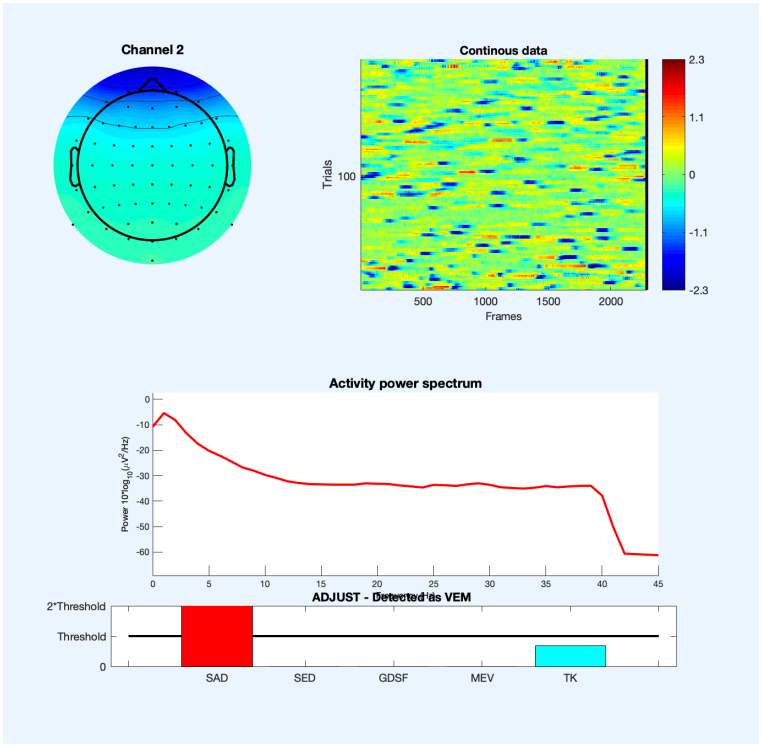

Figure 3 : Figure showing topography, erp-image, spectrum of the component 2. Information regarding the spatial/temporal features calculated by the ADJUST toolbox are plotted at the bottom. We can see that this artifact has been automatically identifies as a VEM (vertical eye movement).

Once you close the main figure showing the component topographies, a dialogue box will appear prompting you to enter the number/index of the component that you want to reject. In figure 4, the second component has been entered. If you want to enter several components, just leave a space between each index.

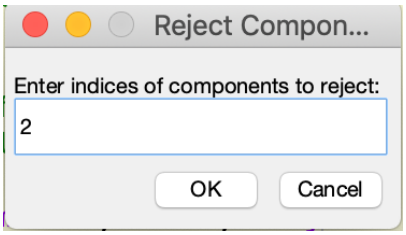

Figure 4 : Dialogue box prompting user to define the indices of the components to be rejected. They will stored in the EEG.reject.gcomp structure.

Important: At this point, the ICs that you entered have not been rejected. They have been stored in EEG.reject.gcomp structure. The "gcomp" sub-field of the EEG.reject structure is a binary vector with a number of elements equal to the number of ICs. Those ICs marked for rejection are set to 1; those ICs to be retained are set to 0.

The dataset with the ICs calculated will be saved to the current subject folder and can be identified as "-ica" will be added to the end of the dataset's name.

Visualising data and rejecting components:

Once ICA has been run and the components to be rejected have been stored in the EEG.reject.gcomp structure, you can visualize the EEG data before and after rejected the marked ICs, to check the effectiveness of the ICA.

Run the CREx_ICA_calc() function again but this time set the following options:

- doICA = 0 (we have already extracted the ICs).
- doPCA = 0 (if we're not doing ICA, then we're not doing PCA)
- doVis = 1 (as we want to visualize)
- doRej = 1, if you want to reject ICs straight after visualization.

Note: You can choose just to visualize the data and run the IC rejection later, by re-running the CREx_ICA_calc() function…

When you visualize the EEG data with/without the IC components stored in the datasets EEG.reject.gcomp structure, the signal before IC rejection will be marked in blue and the signal after IC rejection will be marked in red (figure 5).

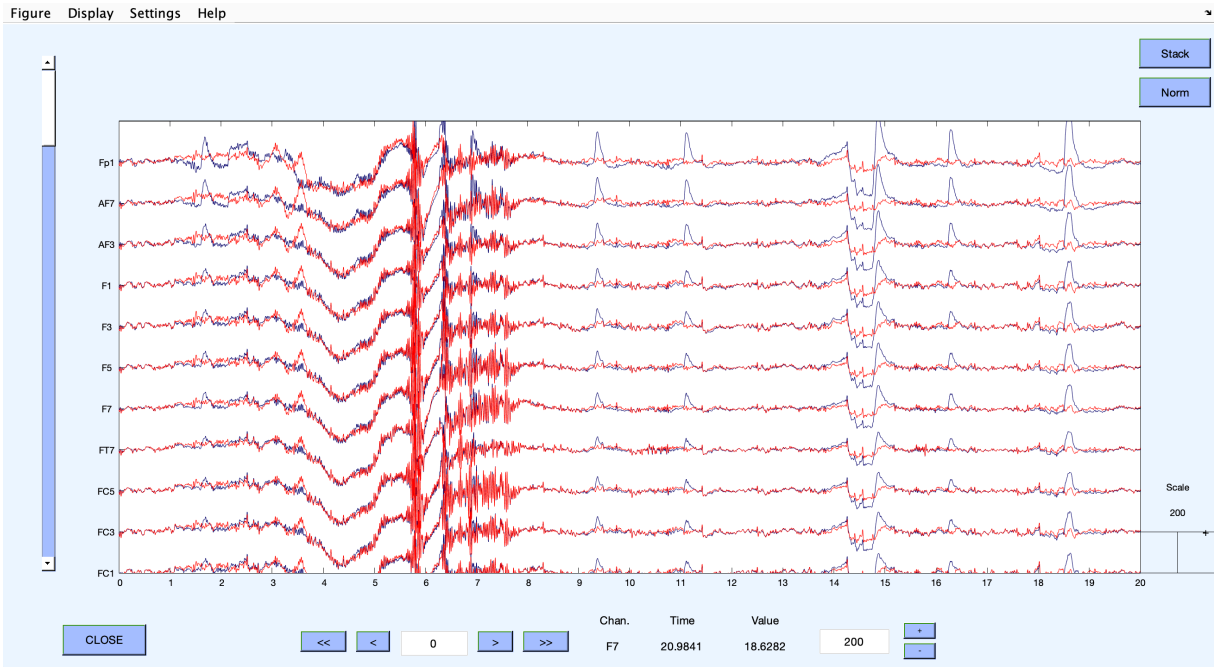

Figure 5: Visualisation of EEG electrode signals before (in blue) and after (in red) IC rejection.

If you choose to carry out IC rejection, a dialogue box will open prompting you to confirm the components that you wish to remove. The dialogue box will show, by default, the indices of the components stored in the EEG.reject.gcomp structure.

Once the ICs have been rejected and the remaining components have been back-projected onto the EEG signals, the dataset is automatically saved (as a *.set file) in the current subject's folder with "*ica-rej*" added to the end. After rejection, a figure showing the EEG before and after IC rejection will again be shown.

Note that the functions available via the EEGLAB GUI can be used to visualize the components as topographies and over time; EEGLAB has some very good functions.

Spatial and Temporal Features from ADJUST plugin: Overview

References

Mognon, A., Jovicich, J., Bruzzone, L., & Buiatti, M. (2011). ADJUST: An automatic EEG artifact detector based on the joint use of spatial and temporal features. *Psychophysiology*. https://doi.org/10.1111/j.1469-8986.2010.01061.x