# 1 EEG structure

The data is organized in the EEGlab format. The different fields of the structure EEG are:

**EEG.setname**

**EEG.filename**

**EEG.filepath**

**EEG.subject**

**EEG.group**

**EEG.condition**

**EEG.session**

**EEG.comments**

**EEG.nbchan** Number of channels

**EEG.trials** Number of trials (epochs)

**EEG.pnts** Number of samples

**EEG.srate** Sampling rate (Hz)

**EEG.xmin** Minimum time

**EEG.xmax** Maximum time

**EEG.times** Time. It is express in samples for non epoched data and in milliseconds for epoched data. Size = 1 x pnts

**EEG.data** Matrix with the data. Size = nbchan x pnts x trials

**EEG.icaact**

**EEG.icawinv**

**EEG.icasphere**

**EEG.icaweights**

**EEG.icachansind**

**EEG.chanlocs** Localization of the channels

**EEG.urchanlocs** Original channel localization

**EEG.chaninfo**

**EEG.ref**

**EEG.event** Structure with the events in the present data. It has fields

`.event(i).type` Name of each event

`.event(i).latency` Latency of each event (samples). If the data is not epoched it is expressed in samples in terms of the whole data. If the data is epoched the samples are counted as if the data were reshaped by concatenating the epochs along the time dimension.

`.event(i).urevent` Original event number. Index to recover original information regarding th event in EEG.urevent.

`.event(i).info` Other fields can be present is extra information was recovered from the event file

**EEG.urevent** Original events. Structure with all the original events. It has fields

`.urevent(i).type` Name of each event

`.urevent(i).latency` Latency of each event (samples). If the data is not epoched it is expressed in samples in terms of the whole data. If the data is epoched the samples are counted as if the data were reshaped by concatenating the epochs along the time dimension.

**EEG.eventdescription**

**EEG.epoch** Cell array, one cell per epoch. Each cell contains the events during that epoch. It has fields

`.epoch{i}.eventtype` Name of each event

`.epoch{i}.eventlatency` Latency of each event (samples). If the data is not epoched it is expressed in samples in terms of the whole data. If the data is epoched the samples are counted as if the data were reshaped by concatenating the epochs along the time dimension.

`.epoch{i}.eventurevent` Original event number. Index to recover original information regarding th event in EEG.urevent.

`.epoch{i}.eventinfo` Other fields can be present is extra information was recovered from the event file

**EEG.epochdescription**

**EEG.reject** rejection structure created by eeglab

**EEG.stats**

**EEG.specdata**

**EEG.specicaact**

**EEG.splinefile**

**EEG.icasplinefile**

**EEG.dipfit**

**EEG.history**

**EEG.saved**

**EEG.etc.eeglabvers**

**EEG.datfile**

**EEG.eventFile**

**EEG.artifacts** structure containing different fields regarding the artifacts identified in the data

**EEG.reference** data used as reference. Size 1 x pnts x trials

**EEG.description** structure containing the mean and standard deviation for each channel

# 2 Import the data

- **eega_importraw** Imports the .raw data to the EEGlab format. If the file with the channels localization is provided this information is added to the EEGl structure.

**Import Data 1:** *eega_importraw*

```
EEG = eega_importraw(filename, filechanloc)
```

Inputs:

`filename` string with the full path and file name.
`filechanloc` string with the full path and file of the channels localization

Outputs:

`EEG` EEGlab structure.

- **eega_addchinfo** The localization of the channels can also be added at any other moment of the analysis by using this function.

**Import Data 2:** *eega_addchinfo*

```
EEG = eega_addchinfo(EEG, filechanloc)
```

Inputs:

`EEG` EEGlab structure
`filechanloc` string with the full path and file of the channels localization

Outputs:

`EEG` EEGlab structure.

- **eega_infoevents** Extra information of the events can be imported from the event file at any moment using this function.

**Import Data 3:** *eega_addchinfo*

```
EEG = eega_infoevents( EEG, pathIn, eventtype )
```

Inputs:

`EEG` EEGlab structure
`pathIn` string with the full path and file to the events file (.txt)
`eventtype` cell array with the names of the events for which extra information will be recovered

Outputs:

`EEG` EEGlab structure.

- **eega_infomatfile** Extra information can be imported from a matlab file at any moment using this function.

**Import Data 4:** *eega_infomatfile*

```
EEG = eega_infomatfile( EEG, pathIn, varnames )
```

Inputs:

`EEG` EEGlab structure
`pathIn` string with the full path and file to the events file (.mat)
`varnames` cell array with the names of the variables to recover

Outputs:

`EEG` EEGlab structure.

# 3 Check the events

# 4 Filtering

Hight pass filtering or a notch filter have to be applied in order to remove line noise (50Hz).
A low pass filter is necessary to remove slow drifts in the signal. Usually data is filter above 0.2-0.5Hz. However, with data from neonates I have observed that filtering at 0.5 kills part of the signal.

- **eega_filter** Filters the data and if required subtracts the mean per electrode.

**Import Data 5:** *eega_filter*

```
EEG = eega_filter( EEG, 'zeromean', zeromean, 'fcutoff', fcutoff, 'ftype', ftype,
'wtype', wtype, 'filtorder', filtorder, 'minphase', minphase )
```

Inputs:

`EEG` EEGlab structure
`zeromean` (Optional - default 0)
`fcutoff` Frequency limits (Hz) for the filter. One or two numbers. (Optional - default [0.2 40])
`ftype` (Optional - default 'bandpass')
`wtype` (Optional - default 'blackman')
`filtorder` (Optional - default ceil( (3.3 / (0.2 / 250) ) / 2) * 2)
`minphase` (Optional - default 'false')

Outputs:

`EEG` EEGlab structure.

# 5 Motion artifacts detection

Different algorithm are applied in order to identify artifacts. The data identified as containing artifacts is stored in EEG.artifacts, which has fields:

**`EEG.artifacts.BCT`** Logical matrix of size NumChan x NumPnts x NumTrials, indicating the data detected as bad.

**`EEG.artifacts.BC`** Logical matrix of size NumChan x 1 x NumTrials, indicating the channels detected as bad per epoch.

**`EEG.artifacts.BT`** Logical matrix of size 1 x NumPnts x NumTrials, indicating the time points detected as bad per epoch.

**`EEG.artifacts.CCT`** Logical matrix of size NumChan x NumPnts x NumTrials, indicating the data interpolated.

**`EEG.artifacts.summary`** Table with the summary of the amount of data rejected (BCT), channels rejected (BC), time points rejected (BT), and data corrected (CCT).

**`EEG.artifacts.algorithm.stepname`** Names of each of the algorithms applied.

**`EEG.artifacts.algorithm.parameters`** Parameters used for each of the algorithms applied.

**`EEG.artifacts.algorithm.rejxstep`** Amount of data rejected in each rejection step.

Each algorithm gives as output a variable BCT containing the data rejected by that algorithm, and updates BCT in EEG.artifacts, such as it contains the old and new rejected data.
Each algorithm rejects data when a certain measure is above or/and below a threshold. The threshold for rejection can be set in three different ways.

1. An absolute threshold is provided.
2. The threshold is defined based on the distribution of specific measure along the whole recording for each electrode.
3. The threshold is defined based on the distribution of specific measure along the whole recording for all the electrodes

Thresholds defined relative to the distribution are estimated as thres = M -/+ n*IQ, where M is the median of the distribution and IQ is the interquartile range (percentile 75 - percentile 25).
If required data can also be z-score before the detection is performed. In that case the mean and standar deviation for each electrode will be used.
Notice that the data has to be filtered (low pass filter) in order for the algorithms to work properly

The different motion artifact interpolation algorithms are run by the function *eega_tArtifacts*. The parameters to run the algorithms are provided to the function as an input in a structure. An example of a structure with the parameters can be created by using the functions *Example_Art1* and *Example_Art2*. The *Art* structure specifies:

**`Art.KeepRejPre`** Defines what it is done with previously rejected data.

**1** Keep data previously rejected as rejected. Default.
**0** Set previously rejected data as good.

**`Art.KeepRejCause`** Keep or not the information regarding which data was rejected by each algorithm.

**1** Save which data samples were rejected by each algorithm.
**0** Do not save. Default

**`Art.Filter.do`** Specify if the data should be filtered before detecting artifacts.

**1** Filter.
**0** Do not filter. Default

**`Art.Filter.frq`** If Art.Filter.do = 1 it species the frequency band to filter the data in Hz. Default [0.2 40].

**`Art.Detect.MaxLoop`** Maximum number of rejection loops.

**`Art.Detect.Tolerance`** Maximum tolerance for the new data rejected (%).

**`Art.Detect.Ref.loop`** Indexes indicating for which loops the data has to be referenced to the mean before running the algorithms. Default = [].

**`Art.Detect.Ref.baddata`** Indicates what to do with the bad data when calculating the reference. It can be: 'none' / 'replacebynan' / 'replacebymean'. Default = 'none'.

**`Art.Detect.steps{i}`** Name of the algorithm to apply.
The algorithm to apply independently are:

- *eega_tRejSpectrum*
- *eega_tRejTimeVar*
- *eega_tRejAmp*
- *eega_tRejAmpDiff*
- *eega_tRejFastChange*

- *eega_tRejNetStation*
- *eega_tRejAmpElecVar*
- *eega_tRejAmpEpochVar*

The algorithms which reject data based on the data already rejected by the previous algorithms are:

- *eega_tMask*
- *eega_tRejShortGood*
- *eega_tRejShortBad*
- *eega_tRejSmplPercCh*
- *eega_tRejChPercSmpl*

**`Art.Detect.loop{i}`** Vector with indexes indicating in which loops the algorithm i should be applied.

**`Art.Detect.P{i}`** Parameters for the algorithm i.
Depending of the algorithm it can contain different fields. Some fields are the following:

**`.dozscore`** Z-score or not the data.

**`.twdur`** For algorithms which are applied on a time window, duration of the time window (s).

**`.twstep`** For algorithms which are applied on a time window, step for the sliding time window (s).

**`.thresh`** Values for the threshold. If absolute, absolutes values, if relative number of IQs from the median.

**`.relative`** Set relative thresholds (1) or absolute thresholds (0).

**`.xelectrode`** If reletive thresholds, estimate it per electrode or for all the electrodes.

Bellow all the functions and their inputs and outputs are described.

- **eega_tArtifacts** This function performs the rejection based on the parameters provided in the structure *Art*. The function does loops of rejection. In each loop, it first implements independently a series of algorithm to detect artifacts. Afterwards, another set functions can be applied in order to add or remove some rejected data based on the data already data. These second set of functions are applied in the order specified in *Art*. After each loop the amount of new data rejected is computed. Rejection loops keep going till when a maximum number of loops is reached or till when the amount of rejected data is under a tolerance threshold. Before starting the rejection the data can be filtered (this is useful for cases in which the original data is not filter and not filtered data is required). For each rejection loop it is possible to specify if the rejection will be done on the original data or on the data referenced to the mean.

  NOTE: the data saved in EEG.data will alway be the original data, NO filtered and NO reference.

  **Artifacts Detection 1:** ***eega_tArtifacts***

  ```
  [ EEG ] = eega_tArtifacts( EEG, Art )
  ```

  Input

  `EEG` EEGlab structure.

`Art` structure with the parameters to perform the rejection

Output:

`EEG` EEGlab structure with EEG.artifacts updated.

- **eega_tRejSpectrum** Reject epochs by electrodes when the power a given frequency band is too high or too low relative to the total power. The ratio is log transformed (decibels).

**Artifacts Detection 2:** *eega_tRejSpectrum*

```
[ EEG, BCT, T ] = eega_tRejSpectrum( EEG, [-4 4], 'dozscore', 0, finterval, [20 40],
'relative', 1 )
```

Inputs:

`EEG` EEGlab structure

`thresh` Threshold for each frequency band. Size n x 2. The first column is the threshold for the lower limit and the second column for the upper limit.

`dozscore` Z-score or not the data. (Optional - default 0).

`finterval` Frequency bands to check. Matrix of size n x 2, where n is the number of frequency bands. (Optional - default [20 40]).

`relative` Calculate relative (1) or absolute (0) thresholds. (Optional - default 0).

Outputs:

`EEG` EEGlab structure with EEG.artifacts updated.

`BCT` logical matrix of size nbchan x pnts x trials, indicating the data detected as bad by the algorithm.

`T` thresholds for each electrode and each frequency band.

- **eega_tRejTimeVar** Rejects samples in a time window when the variance across samples is bigger or lower than a threshold

**Artifacts Detection 3:** *eega_tRejTimeVar*

```
[ EEG, BCT, T ] = eega_tRejTimeVar( EEG, [-Inf 4], 'dozscore', 0, 'twdur', 0.5,
'twstep', 0.25, 'relative', 1, 'xelectrode', 0 );
```

Inputs:

`EEG` EEGlab structure

`thresh` threshold for the variance across samples. The first column is the threshold for the lower limit and the second column for the upper limit.

`dozscore` Z-score or not the data. (Optional - default 0).

`twdur` time window duration (s). (Optional - default 0.5).

`twstep` time window step (s). (Optional - default 0.25).

`relative` Calculate relative (1) or absolute (0) thresholds. (Optional - default 0).

`xelectrode` Estimate thresholds independently for each electrode (1) or an unique threshold for all the electrodes (0).(Optional - default 1).

Outputs:

`EEG` EEGlab structure with EEG.artifacts updated.

`BCT` Logical matrix of size nbchan x pnts x trials, indicating the data detected as bad by the algorithm.

`T` thresholds for each electrode.

- **eega_tRejAmp** Rejects samples with a too high (or low) amplitude.

**Artifacts Detection 4:** *eega_tRejAmp*

```
[ EEG, BCT, T ] = eega_tRejAmp( EEGout, [-4 4], 'dozscore', 0, 'relative', 1,
'xelectrode', 0 );
```

Inputs:

`EEG` EEGlab structure

`thresh` threshold for the amplitude. The first column is the threshold for the lower limit and the second column for the upper limit.

`dozscore` Z-score or not the data. (Optional - default 0).

`relative` Calculate relative (1) or absolute (0) thresholds. (Optional - default 0).

`xelectrode` Estimate thresholds independently for each electrode (1) or an unique threshold for all the electrodes (0).(Optional - default 1).

Outputs:

`EEG` EEGlab structure with EEG.artifacts updated.

`BCT` Logical matrix of size nbchan x pnts x trials, indicating the data detected as bad by the algorithm.

`T` thresholds for each electrode.

- **eega_tRejAmpDiff** Rejects samples when the change in amplitude is too high or low (first derivative).

NOTE: This algorithm is working fine for identifying very sharp changes in amplitude, like the heart beats or jumps in the signal. This type of artifacts are successfully corrected by the target PCA algorithm. See below.

**Artifacts Detection 5:** *eega_tRejAmpDiff*

```
[ EEG, BCT, T ] = eega_tRejAmpDiff( EEGout, [-4 4], 'dozscore', 0, 'relative', 1,
'xelectrode', 0 );
```

Inputs:

`EEG` EEGlab structure

`thresh` threshold for the change in amplitude. The first column is the threshold for the lower limit and the second column for the upper limit.

`dozscore` Z-score or not the data. (Optional - default 0).

`relative` Calculate relative (1) or absolute (0) thresholds. (Optional - default 0).

`xelectrode` Estimate thresholds independently for each electrode (1) or an unique threshold for all the electrodes (0).(Optional - default 1).

Outputs:

`EEG` EEGlab structure with EEG.artifacts updated.

`BCT` Logical matrix of size nbchan x pnts x trials, indicating the data detected as bad by the algorithm.

`T` thresholds for each electrode.

- **eega_tRejFastChange** Rejects samples in a brief time window when the maximum change is above a threshold. The time window is moved sample by sample.

NOTE: This algorithm is slow, so it may be better to avoid.

**Artifacts Detection 6:** *eega_tRejFastChange*

```
[ EEG, BCTS, T ] = eega_tRejFastChange( EEG, 4, 'dozscore',0, 'twdur', 0.025,
'tmask', 0,'relative', 1 )
```

Inputs:

`EEG` EEGlab structure

`thresh` threshold for the variance across samples. The first column is the threshold for the lower limit and the second column for the upper limit.

`dozscore` Z-score or not the data. (Optional - default 0).

`twdur` time window duration (s). (Optional - default 0.025).

`tmask` length of a mask to reject samples around the samples detected as bad (s). (Optional - default 0).

`relative` Calculate relative (1) or absolute (0) thresholds. (Optional - default 0).

Outputs:

`EEG` EEGlab structure with EEG.artifacts updated.

`BCT` Logical matrix of size nbchan x pnts x trials, indicating the data detected as bad by the algorithm.

`T` thresholds for each electrode.

- **eega_tRejNetStation** Implements the NetStation algorithm in a sliding time window. The algorithm calculates a fast running average and a slow running average in the time window, and reject samples in the time window if the fast running average is above the threshold, or if the difference between the fast and slow running average is above the threshold.

  NOTE: Do not apply this algorithm with relative thresholds per electrode because the data from some electrodes is rejected even if there are not artifacts. This was happening even if the data was referenced to the mean.

  **Artifacts Detection 7:** *eega_tRejNetStation*

  ```
  [ EEG, BCT, BCT_f, BCT_d, T ] = eega_tRejNetStation( EEG, 4, 4, 'dozscore', 0,
  'twdur', 0.5, 'twstep', 0.25, 'relative', 1, 'xelectrode', 0 )
  ```

  Inputs:

  `EEG` EEGlab structure

  `thresh_fa` threshold for the fast algorithm.

  `thresh_da` threshold for the difference algorithm.

  `dozscore` Z-score or not the data. (Optional - default 0).

  `twdur` time window duration (s). (Optional - default 0.5).

  `twstep` time window step (s). (Optional - default 0.25).

  `relative` Calculate relative (1) or absolute (0) thresholds. (Optional - default 0).

  `xelectrode` Estimate thresholds independently for each electrode (1) or an unique threshold for all the electrodes (0).(Optional - default 1).

  Outputs:

  `EEG` EEGlab structure with EEG.artifacts updated.

`BCT` logical matrix of size nbchan x pnts x trials, indicating the data detected as bad by the algorithm.

`BCT_f` logical matrix of size nbchan x pnts x trials, indicating the data detected as bad by the fast algorithm.

`BCT_s` logical matrix of size nbchan x pnts x trials, indicating the data detected as bad by the difference algorithm.

`T` thresholds for each electrode for the fast and difference running average.

- **eega_tRejAmpElecVar** Rejects samples when the amplitude is too high relative to the amplitude in the other electrodes. M +/- n*IQ, where the median and IQ are calculated across electrodes for each time point.

**Artifacts Detection 8:** ***eega_tRejAmpElecVar***

```
[ EEG, bcts ] = eega_tRejAmpElecVar( EEGout, [-4 4] );
```

Inputs:

`EEG` EEGlab structure

`thresh` threshold for the amplitude in number of interquartile ranges.

Outputs:

`EEG` EEGlab structure with EEG.artifacts updated.

`BCT` Logical matrix of size nbchan x pnts x trials, indicating the data detected as bad by the algorithm.

- **eega_tRejAmpEpochVar** Rejects samples when the amplitude is too high relative to the amplitude in the other epochs. M +/- n*IQ, where the median and IQ are calculated across epoch for each time point and electrode.

**Artifacts Detection 9:** ***eega_tRejAmpEpochVar***

```
[ EEG, bcts ] = eega_tRejAmpEpochVar( EEGout, [4 4] );
```

Inputs:

`EEG` EEGlab structure

`thresh` threshold for the amplitude in number of interquartile ranges.

Outputs:

`EEG` EEGlab structure with EEG.artifacts updated.

`BCT` Logical matrix of size nbchan x pnts x trials, indicating the data detected as bad by the algorithm.

- **eega_tMask** Marks as bad samples around a bad segment.

NOTE: The masking instead of being done during the interpolation can also be done when artifacts are corrected.

**Artifacts Detection 10:** ***eega_tMask***

```
[ EEG, BCTS ] = eega_tMask( EEG, 0.015 );
```

Inputs:

`EEG` structure.

`tmask` time in seconds to reject before and after a bad segment.

Outputs:

`EEG` EEGlab structure with EEG.artifacts updated.

`BCT` logical matrix of size nbchan x pnts x trials, indicating the data detected as bad by the algorithm.

- **eega_tRejShortGood** Marks as bad short segments between two bad segments.

**Artifacts Detection 11:** *eega_tRejShortGood*

```
[ EEG, BCT ] = eega_tRejShortGood( EEG, 0.05 );
```

Inputs:

`EEG` structure.

`tmin` minimum length in seconds required of a good segment to be kept as good (s).

Outputs:

`EEG` EEGlab structure with EEG.artifacts updated.

`BCT` logical matrix of size nbchan x pnts x trials, indicating the data detected as bad by the algorithm.

- **eega_tRejShortBad** Removes from the rejection matrix, BCT, segments that are too short.

NOTE: If very short artifacts (like jumps in the signal or heart beat) want to be detected, then it is better no to use this and accept even very brief artifacts.

**Artifacts Detection 12:** *eega_tIncShortBad*

```
[ EEG, RCT, R ] = eega_tRejShortBad( EEG, 0.05 );
```

Inputs:

`EEG` EEGlab structure.

`minlenght` minimum length in seconds required of a bad segment to be kept as bad.

Outputs:

`EEG` EEGlab structure with EEG.artifacts updated.

`RCT` logical matrix of size nbchan x pnts x trials, indicating the data re-included by the algorithm.

`R` amount of data re-included by the algorithm.

- **eega_tRejSmplPercCh** Rejects all the electrodes at a given time point if too many channels were rejected.

**Artifacts Detection 13:** *eega_tRejSmplPercCh*

```
[ EEG, BCT, isbadsmpl ] = eega_tRejSmplPercCh( EEG, 0.5 );
```

Inputs:

`EEG` EEGlab structure.

`maxBadCh` maximum proportion of bad channels.

Outputs:

`EEG` EEGlab structure with EEG.artifacts updated.

`BCT` logical matrix of size nbchan x pnts x trials, indicating the data detected as bad by the algorithm.

`isbadsmpl` logical matrix of size 1 x pnts x trials, indicating the time points detected as bad by the algorithm.

- **eega_tRejChPercSmpl** Rejects a channel in an epoch if too many samples were rejected.

**Artifacts Detection 14:** ***eega_tRejChPercSmpl***

```
[ EEG, BCT, isbadch ] = eega_tRejChPercSmpl( EEG, 0.25 );
```

Inputs:

`EEG` EEGlab structure.

`maxBadSmpl` maximum proportion of bad samples.

Outputs:

`EEG` EEGlab structure with EEG.artifacts updated.

`BCT` logical matrix of size nbchan x pnts x trials, indicating the data detected as bad by the algorithm.

`isbadch` logical matrix of size nbchan x 1 x trials, indicating the channels detected as bad by the algorithm.

# 6 Artifacts Correction and interpolation

The corrections and interpolation involves

1. To define channels that are not working properly during a whole epoch (or during all the recording time). This is done based on the amount of artifacts identified for each channel. It means defining BC.
2. To define periods with strong artifacts. This is done based on the amount of channels containing artifacts and the duration of the artifacts. It means defining BT.
3. To interpolate / correct the data when it is possible. The corrected

The correction is done in an iterative way. When it is possible, detected artifacts are corrected, and afterwards the detection is run again. The precess can be repeated several times.

The correction and interpolation of artifacts can be done in two basic ways:

**Target PCA** During periods containing artifacts we can assume most of the variance comes from the artifacts. Under this assumption we can estimate principal components associated with artifacts during this periods, and this components can be subtracted to the real data in order to correct the artifacts. In practice, bad data segments are concatenated, and PCA is applied only on this subset of data. Then, the components explaining most of the variance are subtracted to the real data subset. Finally, corrected segments are splice into the whole data and aligned to it to avoid discontinuities. It has to be noticed that the variance of electrodes it is normalized before the method is applied, in order to have equal variability across them. Data is re-scaled at the end. Moreover, when the bad segments are concatenated they are previously de-mean.
This methods is applicable for short segments of data (no much longer than 100ms) affected by artifacts, an in particular when they have a specific spatial structure. It is particularly good for removing heart beats and jumps in the signal.

**Spatial Interpolation** Data from some electrodes is interpolated using the data from other electrodes by using using a spherical spline. It is applicable when a few electrodes are not working properly.

For both methods it is necessary to low pass filter the data after they are applied (on non epoched data), because drifts in the signal appear after splicing the corrected data segments.

And the set of functions to do so are:

- **eega_tTargetPCAEEG** This function corrects bad segments of data using target PCA. PCA is apply only on the bad. The selection of the periods to apply PCA is made based on the length of the artifacts and the amount of bad electrodes.

  NOTE: The selection of the bad segments is not trivial, the next function deals with this issue by applying the method recursively by electrode.

  **Artifacts Correction 1:** ***eega_tTargetPCAEEG***

  ```
  EEG ] = eega_tTargetPCAEEG( EEG, nSV, vSV 'maxtime', maxtime, 'mask', mask,
  'alltime', alltime, 'allchannels', allchannels, 'savecorrected', 1 );
  ```

  Inputs:

  `EEG` EEGlab structure.

  `nSV` Number of principal components to remove from the data.

  `vSV` Amount variance to remove from the data (it removes the first n components of the data that removes a fraction of the variance up to vSV).

  `maxtime` Maximum duration of a segment in order to apply the correction (in seconds). Default 0.200.

  `mask` The correction is also applied on segments around bad periods (in seconds). Default 0.

  `alltime` Logical value defining if the correction is applied on all samples which are bad in BCT (1) or only in that ones defined as bad in BCT but for which the an artifact was defined in BT (0). Default 0.

  `allchannels` Logical value defining if the correction is applied on all channels (1) or only in channels defined as good in BC (0). Default 0.

  `savecorrected` Logical value defining if the corrected data will be marked as good in BCT (1) or not (0). Default 1.

  Outputs:

  `EEG` EEGlab structure with the update rejection structure.

- **eega_tTargetPCAxElEEG** This function corrects bad segments of data using target PCA. PCA is apply only on the bad data and independently for each channel. This avoids the difficulty of determining the subset of data on which the correction is applied. For each channel, *j*, bad segments are determined based on *BCT(j,:,:)*; data from this segments are concatenated (including all electrodes!); and PCA is applied on it. The components explaining most of the variance are subtracted to the data of channel *j*. Finally, corrected segments are splice into the whole data. This is done independently for each channel.

  **Artifacts Correction 2:** ***eega_tTargetPCAxElEEG***

  ```
  EEG ] = eega_tTargetPCAxElEEG( EEG, nSV, vSV 'maxtime', maxtime, 'mask', mask,
  'alltime', alltime, 'allchannels', allchannels, 'savecorrected', 1 );
  ```

  Inputs:

`EEG` EEGlab structure.

`nSV` Number of principal components to remove from the data.

`vSV` Amount variance to remove from the data (it removes the first n components of the data that removes a fraction of the variance up to vSV).

`maxtime` Maximum duration of a segment in order to apply the correction (in seconds). Default 0.200.

`mask` The correction is also applied on segments around bad periods (in seconds). Default 0.

`alltime` Logical value defining if the correction is applied on all samples which are bad in BCT (1) or only in that ones defined as bad in BCT but for which the an artifact was defined in BT (0). Default 0.

`allchannels` Logical value defining if the correction is applied on all channels (1) or only in channels defined as good in BC (0). Default 0.

`savecorrected` Logical value defining if the corrected data will be marked as good in BCT (1) or not (0). Default 1.

Outputs:

`EEG` EEGlab structure with the update rejection structure.

- **eega_tInterpSpatial** This function use spherical spline to interpolate electrodes not working during an entire epoch.

**Artifacts Correction 3:** ***eega_tInterpSpatial***

```
EEG, Interp_out ] = eega_tInterpSpatial( EEG, p_int, excbad );
```

Inputs:

`EEG` EEGlab structure.

Outputs:

`EEG` EEGlab structure with the update rejection structure.

- **eega_tInterpSpatialSegment** This function use spherical spline to interpolate segments of bad data, which do not extend during the whole epoch, for each electrode.

**Artifacts Correction 4:** ***eega_tInterpSpatialSegment***

```
EEG, Interp_out ] = eega_tInterpSpatialSegment( EEG, p_int );
```

Inputs:

`EEG` EEGlab structure.

Outputs:

`EEG` EEGlab structure with the update rejection structure.

In order define bad segments and bad channels the following function is used. After each interpolation the data marked as good in BCT is marked as good. Therefore it is necessary to define again BT and BC based on BCT. Nevertheless, the definition of BT and BC during the interpolation does not make a big difference. BC will define which channels are spatially interpolated during the whole time, thus only very bad channels should be define as bad. BT will defined on which periods the target PCA interpolation and the spatial interpolation by segments will be applied. Because each of this functions have anyway they threshold, it is better no to be strict to define BT in this stage.

- **eega_tDefBTBC** This function defines bad channels (BC) and data points (BT) based on the percentage of bad samples and bad channels respectively.

**Artifacts Correction 5:** *eega_tDefBTBC*

```
EEG, BTnew, BCnew ] = eega_tDefBTBC( EEG, rLimCh, rLimSmp, aLimCh, aLimSmp,
'minGoodTime', minGoodTime, 'minBadTime', minBadTime, 'maskTime', maskTime,
'badChtotdata', badChtotdata, 'keeppre', 0, 'plot', 1 );
```

Inputs:

`EEG` EEGlab structure.

`rLimCh` relative threshold for the number of bad channels, determining that a sample is bad. It is the number of IQ from the 3rd quartile.

`rLimSmp` relative threshold for the number of bad samples, determining that a channels is bad. It is the number of IQ from the 3rd quartile.

`aLimCh` absolute lower and upper limits for the maximum number of bad channels to define a bad sample (proportion). It's a 2 elements vector with numbers between 0 and 1.

`aLimSmp` absolute lower and upper limits for the maximum number of bad samples to define a bad channel (proportion). It's a 2 elements vector with numbers between 0 and 1.

`minGoodTime` Periods with non artifact shorter than minGoodTime (in seconds) are marked as having artifacts. Default 0.

`minBadTime` Periods with artifacts shorter than minBadTime (in seconds) are not included. Default 0.

`maskTime` Samples around a bad segment marked as bad in BT (in seconds). Default 0.

`badChtotdata` Logical value indicating if bad channels are defined based on all the recording (1) or per epoch (0). Default 0.

`keeppre` Logical value indicating if keeping (1) or not (0) the previous BT and BC. Default 0.

`plot` Logical valued indicating if producing and output plot (1) or not (0). Default 0.

Outputs:

`EEG` EEGlab structure with EEG.artifacts updated.

`BTnew` New bad segments.

`BCnew` New bad channels.

# 7 Obtain ERP

In order to obtain the ERP the data has to be locked to an event (epoched), reference (in principle to the average), and eventually z-scored and/or a baseline remove. Removing a baseline seems to make statistics worst, while normalizing by z-score using the mean and standar desviation during a long baseline seems to improve statistics.
In order to avoid distortions of the topology ERP do not correct artifacts after referencing to the average.

- **eega_epoch** This function epochs the data relative the provided event and using the provided time interval.

- **eega_tDefBEart** This function defines bad epochs based on the percentage of data rejected and interpolated.

**Artifacts Correction 6:** *eega_tDefBEart*

```
EEG, isbadepoch ] = eega_tDefBEart( EEG, maxBCTS, maxBTS, maxBCS, maxCCTS,
'relative', 1, 'keeppre', 1, 'maxcicles', 4, 'plot', 1, 'savefigure', 1);
```

Inputs:

`EEG` EEGlab structure.

Outputs:

`EEG` EEGlab structure with the update rejection structure.

- **eega_tDefBEdist** This function defines bad epochs based on the percentage of data rejected and interpolated.

**Artifacts Correction 7:** *eega_tDefBEdist*

```
EEG, isbadepoch ] = eega_tDefBEdist( EEG, maxMeanDist, maxMaxDist, 'relative', 1,
'keeppre', 1, 'maxcicles', 4, 'plot', 1, 'savefigure', 1 );
```

Inputs:

`EEG` EEGlab structure.

Outputs:

`EEG` EEGlab structure with the update rejection structure.

- **eega_refavg** This function subtracts the average across electrodes at each data point.

- **eega_normalization** This function normalizes the data of each electrode and epoch by z-score. The mean and standard deviation are calculated during the time interval indicated in the input.

- **pop_rmbase** This function subtracts a baseline to each electrode and epoch by removing the mean over the period indicated in the input.

- **eega_infoevents** This function recovers extra information from a text file about the events in the EGG structure. It can be applied on epoched and not epoched data.

- **eega_definefactors** This function defines relevant factors and its value for each epoch. The factors have to be fields of EEG.epoch based on the event information present in EEG.epoch.

- **eega_avgdata** This function calculates the average across epochs. The epochs are grouped and averaged together, based on the condition they belong, therefore the size of output EEG.data is electrodes x samples x conditions. Conditions are defined based on a table provided in the input, which indicates for each condition the value of the relevant factors. Consider that this function needs that EEG.F exists, thus it has to be run after *eega_definefactors*.