**Updated - March 5, 2020**

**BadChannels**

This class includes methods to aid in the process of finding bad channels – channels with outlier erratic behaviors.

The class is not intended to be a fully automatized recognition tool of bad channels, but to run tests on the channels whose results can point to a possibly atypical behavior in channels, which should be then manually verified. The assumption is that no one single test is enough, but a combination of positive results should raise suspicion regarding a channel.

The class also includes several additional methods aside from the method that runs tests which can help in the process of finding bad channels.

Bad channels are marked in the macro montage in the field badChannel which can have the following values:

|  |  |  |
| --- | --- | --- |
| **Field name** | **Possible values** | **Interpretation** |
| badChannel | 1 | bad channel |
| 2 | big and atypical response to stimulus (effectively these are also regarded as bad channels) |
| 3 | high correlation to adjacent channel (there are many channels of this sort, effectively this flag is disregarded) |
| 4 | opposite polarity, macro montages may also include the field ‘flipped’, which will have the value 1 if the data in this channel was flipped (data = -data) and then resaved. |
|  | 5 | Contact not seen in CT – resides outside the brain (added in Feb 2022) |
| flipped | 0/1 | value 1 if the data in this channel was flipped (data = -data) and then resaved. |
| highSpikesActivity | 0/1 | channel has a spike rate higher than the set threshold (by default 5 spikes/min). |

Before running:

Note that before running any of the methods, properties of the class which are properties of the current patient’s dataset should be set:

**sourceFolderMacro** – the folder in which that data files are saved; the method assumes the format of the data file is ‘CSC<#channel number>’.

**macroMontageFileName** – file name (including path) of the macro montage file.

**spikeResultsFileName** (optional) – file name (including path) of the prefix of spike times results, the method assumes the file name format is <spikeResultsFileName><<#channel number>. If not provided, spikes outliers will not be taken into account.

The method ***loadHeaderExpData*** should be run before using other methods, and receives as input the filename (including path) of the EXP\_DATA struct.

The methods included in the class:

*findBadChannels*

This method runs all the tests that are designed to help in detecting bad channels. The output of the method is a matrix #of channels \* #of tests, where each row is a channel (according to channel indices) and each column is a test the method runs, where 1 represents a positive result of the test and 0 a negative result. The property badChannelsDict is a dictionary mapping from test names to test indices in the matrix.

The tests which are run are:

A. Norm based tests – the data is divided to short segments in the length timeWinForAmp (by default 1 sec), and the norm is calculated for all the segments. The method calculates parameters of the distribution of norms per channel – standard deviation, skewness and kurtosis. These parameters are then compared to the median of the parameters of all the channels **in a given area** and the test is positive for channels that pass the threshold.

For example – the standard deviation (std) of the 1-sec norms is calculated for each channel of the 8 channels in the ‘LAH’ region. The method calculates the median of these 8 numbers. It then compares the std of each channel to three different thresholds – low, medium and high. The low is by default 1.5\*median, medium is by default 2\*median and high is by default 4\*median. Channels for which the std pass the thresholds are marked accordingly in the output matrix (in the columns whose names are: LowThreshAmpSTD, MidThreshAmpSTD and HighThreshAmpSTD). The skewness and kurtosis only have two thresholds – low and high. In total there are 7 tests of this sort: LowThreshAmpSTD, MidThreshAmpSTD, HighThreshAmpSTD, LowThreshAmpSkewness, HighThreshAmpSkewness, LowThreshAmpKurtosis, HighThreshAmpKurtosis.

B. Correlation based tests – these tests are based on the calculation of correlation between adjacent channels within each area (channel N and channel N+1). There are three types of tests:

HighCorrAdj – tests whether the correlation coefficient is above a threshold (highCorrThresh, by default 0.95). This test is designed to find duplicate channels. Note: this threshold may be too low, empirically many many channels pass this test and we didn’t rule them out because of that.

LowCorrAdj – tests whether the correlation coefficient is below a threshold (lowCorrThresh, by default 0.3), which may be atypical behavior as most adjacent channels are very similar to each other.

NegCorrAdj – tests whether the correlation coefficient is negative, which may point to a channel with opposite polarity.

C. Channels with too many NaNs – channels whose percentage of NaNs in the data is above a threshold (NaNThreshPerc, by default 0.3, i.e. 30%).

D. Channels with high spikes – if the spikeResultsFileName property is not empty the method loads spike results files and calculates spike rate (spikes/min). Channels with spikes rate above a threshold (SpikesRateThresh, by default 5) are marked as HighSpikes.

*saveEDF*

This is a help function that saves EDF files of channels in order to view them in an EDF viewer. A small amount of channels should be saved each time because the files are very big. The method receives:

chansToSave – array of channel indices to save in the file.

nameEDF – filename (including path) of the EDF file.

The code of this method is based on code from Maya and the method uses the external method saveEDF which should be downloaded from Matlab Central first.

*saveSpikesResults*

In case no spike detection files are saved in advance, this method runs spikes detection on all the channels of the patient and saves them with the filenames as defined in the property spikeFileName.

*saveSpikesResultsRevPolar*

Detects and saves spike times only for channels which have opposite polarity (the badChannel field in the Macro Montage is 4). The method assumes the data wasn’t flipped and flips it before running the detection.

*findHighSpikes*

A method that finds channels with high spike rate. *findBadChannels* also checks whether the spike rate in each channel is above the threshold, so it’s unnecessary to run this method in addition to *findBadChannels*. It’s intended for situations: a. where this result is required without the other tests. B. where the spike rate itself (and not only whether it is above the threshold) is of interest.

The method receives no input and returns as output:

highSpikesInds – the indices of the channels which passed the threshold for this patient.

spikeRates – the spike rates of all the channels for this patient (i.e. not just those that passed the threshold).

*correctChanWithNaN*

Some channels have bad segments which need to be NaNed (rather than marking the entire channel as bad). This method receives indices of the segments, loads the data, marks the segments as NaN and saves the data.

The input is a matrix (nanInds) with the following format:

Each row includes 3 numbers: <channel index> <start point of bad segment in seconds> <duration of bad segment in seconds)

*getChansByArea*

This method returns two variables:

uArea - a list of all the areas for the current patient

mapAreaChan – a map from each area to its list of channel indices. i.e. mapAreaChan(‘RAH’) will return a list of the RAH channel indices.