**SlowWavesDetector**

The slow wave detector implements three methods for slow waves detection. The basic concept of all the methods is that slow waves are detected as slow fluctuations (0.16-1.25 Hz) which pass conditions regarding minimal amplitude and minimal and maximal duration. The minimal amplitude threshold can be defined a-priori, relative to the current dataset, or relative to a larger dataset. The duration thresholds (min and max) are predefined. The difference between the methods is how slow waves candidates are defined, how the candidates’ amplitude is defined, and how the thresholds are set.

findSlowWavesStaresina

Based on Staresina et al 2015. Note this is the method used for our analyses so far.

The method performs the following steps:

* If sleep scoring is provided – sets all non NREM sleep points to NaN.
* If IIS times are provided – sets the vicinity of IIS to NaN (+- 500 ms).
* Filters the data to the slow waves frequency range using bandpass filter (the properties which set the range are lowLimitPhase and highLimitPhase, by default 0.16-1.25 Hz).
* Find “cycle start points” – zero crossings of the data in which it changes its sign from positive to negative, i.e. the candidates in this method are from zero to zero (with the same gradient sign). The variable isPosToNeg sets whether the detected zero crossings will be a change from positive to negative or the other way around, by default it’s set to true.
* Removes candidates which are too short or too long, as defined by the properties minLengthThresh and maxLengthThresh.
* For each cycle the amplitude is defined as peak value - trough value (max-min).
* Leaves only cycles with amplitude at the top percentile, as defined by percentileForAmplitudeStaresina (by default – top 25%).
* The output is an array with the peak times of the slow waves.

findSlowWaves:

A combination of the methods described in Lafon et al (2017), Staresina et al (2015), and Molle et al (2002).

The method performs the following steps:

* If sleep scoring is provided – sets all non NREM sleep points to NaN.
* If IIS times are provided – sets the vicinity of IIS to NaN (+- 500 ms).
* Filters the data to the slow waves frequency range using bandpass filter (the properties which set the range are lowLimitPhase and highLimitPhase, by default 0.16-1.25 Hz).
* Extracts the instantaneous phase from the filtered dataset using Hilbert transform (preserving NaN sample points by turning them to 0 before the transform and turning them back to NaN after).
* Find “cycle start points” – points in which the phase jumps from pi to -pi, i.e. a cycle is defined from trough to trough as detected by looking at phase change.
* For each cycle the amplitude is defined as its max value.
* There are three options for filtering the slow waves based on their amplitude:
  + usePredefinedThresholdAmplitude is true – uses an a-priori predefined threshold for the amplitude, which is stored in the property predefinedThreshForAmp.
  + Filter in relative to current data – set the threshold at the upper percentileForAmplitude percentile of amplitudes in the current data (by default – leaves 40% of the slow waves).
  + The threshold is set previously based on other data (e.g. a larger set) by using the method setSlowWaveThresh. In which case the property thresholdAmplitude is not NaN but the property usePredefinedThresholdAmplitude is false (because this is not an a-priori threshold, it is data driven, only on a different set of data).
* After filtering the slow waves based on amplitude, the slow waves are also filtered by their duration, where only slow waves within minLengthThresh to maxLengthThresh remain.
* The output is an array with the size #slow waves\*2 with the limits of the slow waves – from trough to trough and an array of times of slow waves peaks.

setSlowWaveThresh

The method calculates a threshold for the amplitude of the slow waves based on input data (which of course can be different, e.g. larger, than the data on which the detection takes place).

The method performs the same calculations on the data as findSlowWaves: bandpass filtering, instantaneous angle using Hilbert transform, finding slow wave cycles, calculating the amplitude for each cycle, and filtering out cycles (and their corresponding amplitudes) which do not pass the minimal or maximal duration condition. It then sets the threshold as the percentileForAmplitude of the amplitudes it calculated (by default, the top 40% percentile).

findSlowWavesMaingret

Based on Maingret et al 2016.

The method performs the following steps:

* If sleep scoring is provided – sets all non NREM sleep points to NaN.
* If IIS times are provided – sets the vicinity of IIS to NaN (+- 500 ms).
* Filters the data to the slow waves frequency range using bandpass filter (the properties which set the range are lowLimitPhase and highLimitPhase, by default 0.16-1.25 Hz).
* Find “cycle start points” – zero crossings of the derivative of the data in which it changes its sign from negative to positive. i.e. the candidates in this method are from trough to trough, as detected by comparing the derivative to zero.
* Removes candidates which are too short or too long (as defined by the properties minLengthThresh and maxLengthThresh).
* Calculated the zscore of the filtered data.
* For each cycle, the amplitude is defined as max value of the zscored data.
* Leaves only cycles for which either: A. The amplitude was above zscoreThreshPeak (by default 2), B. The amplitude was above zscoreThreshPeakWithEnd (by default 1) and the value at the end of the cycle was above zscoreThreshEnd (by default -1.5).
* The output is an array with the peak times of the slow waves.

A note about Staresina’s method vs Maingret’s method

Maingret’s method results in a considerably smaller amount of slow waves detections.

The reason is that Maingret find their slow wave condidates by finding local troughs (zero crossings in the derivative of the signal) - and a candidate would be a segment between two adjacent local troughts. As there are many small insignificant local troughs many of their candidates end up being very short (and are obviously not slow waves). Staresina find their candidates by finding negative-to-positive zero crossings in the actual signal, which is less sensitive to fast noise. Both methods also employ a duration criteria – only candidates which are not too short or too long pass may be detected as a slow wave.

In the following figure the red asterisks signify the limits of the Maingret candidates (corresponding to local troughs) and the green the limits of the Staresina candidates (corresponding to zero crossings). It’s clear that many of Maingret's candidates would be filtered out based on a short duration.

This is what led us to prefer Staresina’s method.

A picture containing text

Description automatically generated