Parameters Pipeline RELAX

Créé par	DeeBeeBOL
Heure de création	@25 juin 2024 10:23
Étiquettes	EEG Matlab pipeline processing

The following is a summary and explanation of the parameters that need to be defined to run the RELAX pipeline.

The pipeline was tested with these parameters on **posttest**, **resting-state data** (first resting-state block) for **s020** using a MWFDelayPeriod of 40ms. It took a total of 22minutes to run, including all three MWF steps and wICA.

Important information for users

The script CLV_CreateRELAX.m creates the RELAX_cfg.mat (the configuration containing the parameters). To be able to run the pipeline on your own computer, you will need to change the following three paths defined at the beginning of this script:

- **Relax_cfg.caploc**: Path to the channel location *.mat file (ChanLocs128.mat)
- **Relax_cfg.myPath**: Path to the raw EEG data.
- **Relax_cfg.OutputPath**: Path to the folder in which the processed data will be saved. This directory will be automatically created if not already present.

In addition, two dependencies are added to the matlab path:

- **PrepPipeline** (this could also be installed in EEGLAB's plugins folder).
- **mwf-artifact-removal-master** (MWF toolbox for EEG artifact removal).

If adding both dependencies, you need to change the paths to each of the above tools in the "Adding dependencies to matlab path" section of the main calling script, CLV_RunRELAX.

💡 To run the pipeline, you can type **CLV_RunRELAX** in the Matlab command window. CLV_RunRELAX is the main, calling script.

💡 Note that the variable **testtype,** which can be either "**posttest**" or "**pretest**" is set to "**posttest**" in the current version of **CLV_RunRELAX**.

💡 The **RELAX_cfg.mat** file is saved for each participant-level dataset and, in addition, a ***.json*** version of this configuration file is saved also for each participant. This is for BIDS compatibility. A lore complete BIDS structure is in preparation.

RELAX pipeline parameters

Need to verify that the parameters used for the RELAX pipeline are the same as those used by Berangère.

RELAX parameter definition

The RELAX_cfg.mat file is created by calling the **CLV_CreateRELAX()** function. A parameter called ***testtype*** is passed to this function. This variable can be either ***Posttest*** or ***Pretest***.

RELAX Parameters

Electrodes to exclude

In the RELAX_cfg file this parameter is *ElectrodesToDelete.* This information should be based on electrodes noted in the *cahier de manip.* By default it is blank. In the CLV implementation the following parameters are marked for exclusion:

- GSR1, GSR2, Erg1, Erg2, Resp, Plet, Temp, EXG1, EXG2, EXG3, EXG4, EXG5, EXG6, EXG7, EXG8

Filter parameters

- Apply notch filter between 47Hz and 53Hz - 4th order Butterworth filter.
- Apply a high-pass filter with a cutoff at 0.25Hz - 4th order Butterworth filter.

> 💡 We do not carry out lowpass filtering at this stage, prior to MWF cleaning, as doing so yields temporal dependencies in the data that lead to rank deficiency when carrying out MWF cleaning, with the effect that a single participant almost 2hours to clean.

Downsampling

Downsampling should, ideally, be applied after lowpass filtering. But a sampling frequency of 4096Hz (the original sampling frequency of the data) the RELAX pipeline takes far too long to run and may not finish.

- Data is downsampled to 1024Hz using the following code:

```
newFS = RELAX_cfg.DownSample_to_X_Hz;
EEG = pop_resample(EEG, newFS);
```

> 💡 A sampling frequency of 512Hz was tested but systematically yielded an error when detecting noisy time intervals. This may be due to the fact that the noisy time intervals are detected by segmenting the continuous data into 1second epochs with a 500ms overlap.

Bad Channel Rejection

The ***findNoisyChannels()*** function from the PREP pipeline is applied. This divides the continuous data into 1 second time windows. For each 1 second time window, if **>5%** of data within the window presents **extreme values**, this window is rejected.

- In GUI: Maximum proportion of electrodes that can be deleted as bad:
  RELAX_cfg.MaxProportionOfElectrodesThatCanBeDeleted = 0.20
- In GUI: Extreme noise proportion electrode deletion threshold:
  RELAX_cfg.PropotionOfExtremeNoiseAboveWhichToRejectChannel = 0.05 (5%)

This value is applied in the ***findNoisyChannel()*** function of the PREP pipeline to detect bad electrodes.

In the findNoisyChannels() function extreme values are determined according to the following criteria:

MAD (mean absolute deviation) from median voltage shift within 1 second epoch. It is the threshold MAD from the median in all epochs for each electrodes against the same electrode in different epochs. If set lower than 20MAD, it would catch less severe voltage shifts.

- In GUI: MAD from median voltage shift: **RELAX_cfg.ExtremeVoltageShiftThreshold** = 20 MAD

MAD from within blink affected epochs. How many MAD from the median across blink affected epochs should be excluded as extreme data.

- In GUI: MAD from median voltage shift in blink affected epochs: **RELAX_cfg.ExtremeBlinkShifThreshold** = 8 MAD

Log-power, log-frequency slopes. Slope of log frequency, log power below which to reject as drift without neural activity.

- In GUI: Log-frequency Log-power for detecting drift threshold: **RELAX_cfg.ExtremeDriftSlopeThreshold** = -4

Absolute Voltage Threshold. Min and max voltage (microVolts) threshold beyond which data will be excluded from cleaning and deleted.

- In GUI: Absolute voltage shift: **RELAX_cfg.ExtremeAbsoluteVoltageThreshold** = 500μV

Other criteria applied in the ***findNoisyChannels()*** to detect noisy electrodes:

- In GUI: Single channel kurtosis: **RELAX_cfg.ExtremeSingleChannelKurtosisThreshold** = 8
- In GUI: All channel kurtosis: **RELAX_cfg.ExtremeChannelKurtosis** = 8

Multi-Channel Wiener Filter

Three muti-channel Wiener filters are applied sequentially to address:

- Muscle activity
- Blinks
- Horizontal eye movements and drift.

Wiener Filter and Multi-Channel Wiener Filter: Background Information

According to basic Wiener theory, the coefficients of a Wiener filter are calculated to minimise the average square distance between the filter output and a desired signal. In its basic form, the Wiener theory assumes that the signals are stationary processes. But, if the filter coefficients are periodically recalculated for every block of N signal samples then the filter adapts itself to the average characteristics of the signals within the block and block adaptive. Here, the activity underlying the EEG signal cannot be considered stationary and so blocks of N samples need to be defined. In the pipeline, these N samples define the **MWF delay period** and need this delay period needs to be calculated as a function of the sampling frequency.

If the desired delay period (D) is 40ms (0.04seconds) then N (the MWF delay period in samples) is given by $N = floor(D * F_s)$ where F_s is the sampling frequency (512Hz).

In GUI MWF Delay Period : **RELAX_cfg.MWFDelayPeriod** = 20 (samples)

💡 A delay period of 8 samples (16ms) is used by default for all MWF cleaning. This delay period implements a positive and negative time lag for applying the MWF filter; this turns the spatial MWF into a spatio-temporal FIR filter. However, if the RELAX pipeline detects eigenvector deficiency, which can occur with longer delay periods and, this can impair the MWF cleaning performance. If generalised eigenvector deficiencies are detected, the algorithm will reduce the period by 1 sample and run the MWF again. This can be repeated up to three times.

In the RELAX pipeline the MWF is carried out in three rounds.

- The **first round** concerns mainly **muscle artifact** cleaning. But eye-blink cleaning can be included in the first round if wished. This implies integrating the eye-blink mask into the noise-mask. The first round is carried out if **RELAX_cfg.Do_MWF_Once** = 1 .
- The **second round** concerns mainly the detection of **eye-blinks** that may have been masked by muscle activity and, thus, missed in the first round of MWF. the second round of MWF is carried out if **RELAX_cfg.Do_MWF_Twice** = 1
- The **third round** concerns the detection of **drift-related and hEOG artifacts**. The third round of MWF is carried out if **RELAX_cfg.Do_MWF_Thrice** = 1 .

There is the question of whether or not we need to carry out all these MWF steps?

Muscle activity

Muscle artifacts are dealt with in the first round of MWF. To create a template for the detection of muscle activity for the MWF cleaning, the data is separated into 1second epochs with 500ms overlap.

The log frequency/log power slope above which to mark an artifact as a muscle artifact. The more negative the value, the more muscle artifact is removed. Less stringent = -0.31, medium stringency = -0.59 and high stringency = -0.72. The default value for this parameter is -0.59.

- In GUI: Log-frequency Log-power slope muscle artifact threshold for MWF cleaning: Relax_cfg.MuscleSlopeThreshold = -0.31.

The function *RELAX_muscle()* creates a template of clean and muscle artefacted data to be used for cleaning using MWF. Note that if data has been low pass filtered below 75Hz, muscle artifact slopes cannot be computed. In this template, muscle artifact periods are marked as 1 and clean periods are marked as 0; thus creating a mask.

- In GUI: Max proportion marked as muscle for MWF cleaning: **RELAX_cfg.MaxProportionOfDataCanBeMark** = 0.5

The following function (below) creates a template of clean and muscle-artifacted data to be used for cleaning in MWF.

```
[continuousEEG, epochedEEG] = RELAX_muscle(continuousEEG, epochedEEG, RELAX_cfg)
```

Blink Activity

The blink activity is mainly dealt with in the second round of MWF. However, it is also possible to detect eye-blinks in the first round, in which case the eye-blink mask will be integrated into the noise mask. To allow eye-blink cleaning in round 1:

- **RELAX_cfg.MWFRoundToCleanBlinks** = 1

In the second round of MWF, the user needs to define the probability that the current data contains blink artifacts. The choices are:

- data almost certainly has blinks: **RELAX_cfg.ProbabilityDataHasNoBlinks** = 0
- data might not have blinks : **RELAX_cfg.ProbabilityDataHasNoBlinks** = 1
- data definitely does not have blinks: **RELAX_cfg.ProbabilityDataHasNoBlinks** = 2

In GUI: Does the data contain blinks? : **RELAX_cfg.ProbabilityDataHasNoBlinks** = 0.

The aim is to detect eye-blinks that were masked my muscle activity and, thus, missed in round one of MWF.

To detect eye-blinks, a copy of data was bandpass filtered using a 4th order Butterworth filter between 1-25Hz. Data in channels identified as blink-relevant was averaged. And blinks were marked as the maximum point within each time-period that exceeded the value of the upper quartile of all voltages + the interquartile range (IQR)*3.

This is carried out by the function in the following code if it is specified that the probability that the dataset does not have blinks is 0.

```
[continuousEEG, epochedEEG] = RELAX_blinks_IQR_method(continuousEEG, epochedEEG,..
 RELAX_cfg)
```

Thus, a blink mask was created for MWF cleaning by marking the 800ms surrounding all blink maximums as artifacts.

Define blink-relevant channels

- In GUI: Blink affected electrodes: **RELAX_cfg.BlinkElectrodes** = C29, C17, C16, C30, C8, C28, C27, C21, C13, C10.
- In GUI: Left sided HEOG affected electrodes: **RELAX_cfg.HEOGLeftpattern** = C30, D7, C8, D9, D23, D10, D22, D24, C31.
- In GUI: Right sided HEOG affected electrodes: **RELAX_cfg.HEOGRightpattern** = C8, C7, B27, B28, B26, B29, B24, B14, C9.

Drift and hEOG Detection

To detect drift, the continuous EEG is re-referenced using PREP's robust average referencing approach and epochs showing an amplitude at any electrode greater than a threshold were marked as artifact periods in the template used for MWF cleaning. Note that the re-referencing is only applied to identify drift and the re-referenced data is not used in the MWF cleaning.

Drift detection is carried out with the following function:

```
[continuousEEG, epochedEEG] = RELAX_drift(continuousEEG, epochedEEG, RELAX_cfg);
```

The drift threshold is the mean absolute deviation (MAD) from the median of all electrodes.

- In GUI: Single electrode drift threshold for MWF cleaning: **RELAX_cfg.DriftSeverityThreshold** = 10.

The maximum proportion of epochs to include in the mask from the drift artifact type.

- In GUI: Max proportion marked as drift for MWF cleaning :
  RELAX_cfg.ProportionWorstEpochsForDrift = 0.3

The horizontal eye-movement threshold calculated as the MAD deviation from the median. if both lateral electrodes show activity above this threshold for a certain duration (the number of timepoints (ms) that exceed the horizontal eye-movement threshold), the activity is marked as horizontal eye-movement.

In GUI: Horizontal eye movement threshold (MAD from median) for MWF cleaning:
RELAX_cfg.HorizontalEyeMovementThreshold = 2

Note: the number of time points (ms) that exceed the horizontal eye-movement threshold within a defined test period before the period is marked for hEOGs. The test time period is calculated as follows:

$$TestWindow = (2 - HorizontalEyeMovementThreshold) - 1$$

A buffer window that defines masking periods before and after the time where the hEOGs exceed threshold is also defined.

- **RELAX_cfg.HorizontalEyeMovementFocus** = 200 (ms)

It is not defined in the RELAX GUI.

Interpolate bad electrodes

Choice of interpolating rejected electrodes to maintain the same number of channels across different data sets.

- In GUI: Interpolate rejected electrodes back into the data after cleaning?
  RELAX_cfg.InterpolateRejectedElectrodesAfterCleaning = 'yes'