EEG-preprocessing

Release 0.0.1

Giulia Pezzutti, Melissa Lajtos

Sep 16, 2022

CONTENTS:

CHAPTER

ONE

PYTHON SCRIPTS

1.1 EEGAnalysis.py

class EEGAnalysis.**EEGAnalysis**(*paths*, *dict_info*, *per_run_inspection=False*)

Bases: `object`

Class implemented for the EEG preprocessing and visualization from the reading of .xdf files. This class is intended with the use of BrainVision products, LSL and Psychopy.

Giulia Pezzutti

Assoc.Prof. Dr. Selina Christin Wriessnegger M.Sc. Luis Alberto Barradas Chacon

Institute for Neural Engineering, @ TUGraz

Variables

- **xdf_paths** (*list[str]*) – Filepaths of xdf files to be read.
- **file_info** (*dict*) – Holds information regarding input and output files and directories.
- **eeg_fs** (*int*) – Sampling frequency of EEG signals.
- **eeg_signals** (list(`numpy.ndarray`)) – List of EEG time series; one array per xdf file.
- **eeg_instants** (list(`numpy.ndarray`)) – List of EEG timestamps; one array per xdf file.
- **marker_ids** (*list(list(list(str)))*) – List of (PsychoPy) markers; one list of lists per xdf file (each innermost list contains one marker).
- **marker_instants** (list(`numpy.ndarray`)) – List of marker timestamps; one array per xdf file.
- **channels_names** (*dict*) – Dictionary of all channels with numbers as keys and names as values.
- **channels_types** (*dict*) – Dictionary of channel types with numbers as keys and types as values.
- **evoked_rois** –
- **info** (`mne.Info`) – Info object with information about sensors and methods of measurement.
- **raw** (`mne.io.RawArray`) – Instance of mne RawArray object, holding eeg raw data and related information, like channels names.
- **bad_channels** (*list[str]*) – Names of all the bad channels.
- **events** (`numpy.ndarray` [int]) – Array of events as returned by `mne.events_from_annotations()`.

- **event_mapping** (*dict*) – Event ids as returned by `mne.events_from_annotations()`.
- **epochs** (`mne.Epochs`) – Epochs extracted from attribute `raw`.
- **annotations** (list[`mne.Annotations`]) – List of time series annotations; one instance per xdf file.
- **evoked** (*dict*) –
- **rois_numbers** (*dict*) – Dictionary of different rois with roi names as keys and lists of channel indices as values.
- **input_info** (*dict*) – The parameter *dict_info* passed to the class constructor.
- **t_min** –
- **t_max** –

Constructor of the class: it initializes all the necessary variables, loads the xdf file with all the correspondent information.

Calls *get_info_from_path()* and `load_raws_from_xdfs()`.

Parameters

- **paths** (*list[str] | str*) – paths to .xdfs file containing the data. The filepath must be with following structure: <folder>/sub-<sub-code>_block<num>.xdf (e.g. "/data/subj_001_block1.xdf").
- **dict_info** (*dict*) – dict containing the main information about the processing. It must contain the following keys: streams, montage, filtering, spatial_filtering, samples_remove, t_min, t_max, full_annotation, annotation_durations, epochs_reject_criteria, rois, bad_epoch_names, erp, erds. See the documentation for further info about the dict.

create_annotations(*full=False*)

Annotations creation according to MNE definition. Annotations are extracted from markers stream data (onset, duration and description) and saved in attribute `annotations`.

Parameters

full (*bool*) – Annotations can be made of just one word or more than one. In 'full' case the whole annotation is considered, otherwise only the second word is kept.

create_epochs(*visualize_epochs=False*, *rois=True*)

Creates mne epochs and saves them in attribute `epochs`.

In order to do so, events and event mapping are created from the annotated raw data in attribute `raw` and are saved in the attributes `events` and `event_mapping`.

Optionally calls *visualize_epochs()*.

Parameters

- **visualize_epochs** (*bool*) – Whether to generate epochs plots or not.
- **rois** (*bool*) – Whether to visualize results according to the rois or for each channel. Only has an effect if *visualize_epochs* is *True*.

create_evoked(*rois=True*)

Function to define the evoked variables starting from the epochs. The evoked will be considered separately for each condition present in the annotation and for each ROI (otherwise, in general for the whole dataset).

Parameters

rois (*bool*) – Whether to visualize results according to the rois or just the general results.

create_raw(*eeg_signal*, *eeg_instants*)

Creates MNE RawArray instance from the data, setting the general information and the relative montage. Also creates the dictionary for the regions of interest according to the current data file and saves is in attribute `rois_numbers`.

Parameters

- **eeg_signal** (`numpy.ndarray`) – EEG time series.
- **eeg_instants** (`numpy.ndarray`) – EEG timestamps.

Returns
Instance of mne RawArray object which was created with the given input.

Return type
`mne.io.RawArray`

get_info_from_path()

Gets main information from file path regarding subject, folder and output folder and saves that information in attribute `file_info`.

get_peak(*t_min*, *t_max*, *peak*, *mean=True*, *channels=None*)

Function to extract the peaks' amplitude from the epochs separately for each condition found and returns them or the mean value.

Parameters

- **t_min** – lower bound of the time window in which the algorithm should look for the peak
- **t_max** – upper bound of the time window in which the algorithm should look for the peak
- **peak** – +1 for a positive peak, -1 for a negative peak
- **mean** – boolean value, if the return value should be the mean value or the list of amplitudes
- **channels** – list of channels name to be investigated. If None, all the channels are considered

Returns
if mean=True, mean amplitude value; otherwise list of detected peaks' amplitude and list of the correspondent annotations

load_channels(*dict_channels*)

Loads channel names and types and saves them in attributes `channels_names` and `channels_types`. Also checks for bad channels in attribute `input_info` and saves their names in attribute `bad_channels`.

Parameters
dict_channels (*dict*) – Channel information loaded from an xdf file.

load_raws_from_xdf(*per_run_inspection=False*)

Loads one or multiple xdf files into a raw object which is saved in the attribute `raw`. Populates attribute `info`.

If parameter *per_run_inspection* is True, the data of each xdf file (i.e. run) is plotted individually, allowing the user to select bad channels. These bad channels will be interpolated as soon as the plot is closed.

Gets called by constructor.

Calls *load_xdf()*, *create_annotations()*, and *create_raw()*.

Parameters
per_run_inspection (*bool*) – Whether each run should be plotted individually and bad channels interpolated.

load_xdf(*xdf_path*)

Loads xdf file from the filepath given in input.

The function automatically divides the different streams in the file according to the stream names given in attribute `input_info` and extracts their main information. EEG signals are converted from µV to V: Saves the sampling frequency of the EEG acquisition in attribute `eeg_fs`. Appends EEG and marker data of given file to attributes `eeg_signals`, `eeg_instants`, `marker_ids`, and `marker_instants`.

Calls *load_channels()*. Optionally calls `_fix_lost_samples()`.

Parameters

xdf_path (*str*) – Path to the xdf file to be loaded (whole path from project directory).

preprocess_raw(*visualize_raw=False*, *ica_analysis=False*, *reference=None*)

Perform some preprocessing to the raw data stored in attribute `raw`.

Sets the reference channel if parameter *reference* is given.

Optionally calls (depending on settings) *raw_spatial_filtering()*, *raw_time_filtering()*, *visualize_eeg()*, and *raw_perform_ica()*.

Parameters

- **visualize_raw** (*bool*) – Whether to visualize the data after processing steps.
- **ica_analysis** (*bool*) – Whether to perform EOG removal via ICA.
- **reference** (*str or list[str]*) – List of channels to use as reference, or 'average' to use a virtual reference.

raw_ica_remove_eog()

Old method from Giulia. Deprecated. Will be removed with next commit.

Automatic removal of EOG components, as suggested by MNE, just does not work.

raw_inspect_ica_components()

Plots ICA components of attribute `raw`.

This function is intended to be used for the inspection of ICA components. See *raw_perform_ica()* for applying ICA.

raw_perform_ica()

Removes ICA components according to ICA indices specified in attribute `input_info`.

Before applying ICA, *raw_inspect_ica_components()* should be used to inspect the ICA components and choose the ones to be removed.

raw_spatial_filtering()

Resets the reference in raw data in attribute `raw` according to the spatial filtering type in the input dict.

raw_time_filtering()

Filter raw data in attribute `raw` with a band-pass filter and a notch filter.

run_raw_epochs(*visualize_raw=False*, *save_images=True*, *ica_analysis=False*, *create_evoked=True*, *save_pickle=True*)

Function to run all the methods previously reported. Attention: ICA is for now not used.

Parameters

- **visualize_raw** (*bool*) – Whether raw signals should be visualized.
- **save_images** (*bool*) – Whether epoch plots should be saved. (note: they are never visualized)

- **ica_analysis** (*bool*) – Whether ICA analysis should be performed.
- **create_evoked** (*bool*) – Whether Evoked computation is necessary.
- **save_pickle** (*bool*) – Whether the pickles with data, label and info should be saved.

save_epochs(*track_dropped=False*)

Saves the epochs stored in attribute `epochs` (created by calling *create_epochs()*).

If parameter *track_dropped* is true, the function counts how many epochs are dropped for each condition and writes an entry (row) in a csv file, that has to be created by the user beforehand. CAUTION: This functionality is pretty much hard-coded to my (Melissa's) needs (regarding tasks and conditions), but can be easily adapted.

Parameters

track_dropped (*bool*) – Whether to count dropped epochs and save the numbers in a csv.

save_pickle()

Function to save epochs, labels and main information into pickle files. The first two are saved as numpy arrays, the last one is saved as dictionary.

visualize_eeg(*signal=True*, *psd=True*, *psd_topo=False*, *sensors=False*)

Visualization of raw data using different plots generated with MNE.

Parameters

- **signal** (*bool*) – Whether the raw time signal plot should be generated.
- **psd** (*bool*) – Whether the psd plot should be generated.
- **psd_topo** (*bool*) – Whether the topographic psd plot should be generated.
- **sensors** (*bool*) – Whether the sensors plot (electrode positions) should be generated.

visualize_epochs(*conditional_epoch=True*, *rois=True*)

Saves plots of mean epoch signal.

Still needs to be checked properly.

Parameters

- **conditional_epoch** (*bool*) – boolean, if visualize the epochs extracted from the events or the general mean epoch
- **rois** (*bool*) – Whether to visualize the epochs according to the rois; only if *conditional_epoch = True*.

visualize_evoked()

Function to plot the computed evoked for each condition and for each region of interest.

1.2 sanity_checks.py

Custom to my (Melissa) data.

sanity_checks.**sanity_check**(*task*, *eeg_source_path*)

Performs basic sanity checks on the raw data of the Handedness MI study.

Checks performed are:

- 3 different cues in markers
- 30 trials in total per file

- right number of channels (33 for BrainAmp, 36 for LiveAmp) in EEG data
- effective sampling rate does not deviate more than 0.03 Hz from 500 Hz

Parameters

- **task** (*str*) – The task performed during the runs. Either 'ME' or 'MI'.
- **eeg_source_path** (*str*) – Path to directory which contains xdf files of raw data of one participant.

Returns
True if no issues were detected, else False.

Return type
bool

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX