pyatuomagic

Release 0.0.1

C. Zurn, R. Bechtold, A. Lawrence, L. Xiang, S. Meza, D. Soni

Dec 20, 2019

CONTENTS:

CHAPTER ONE

PROJECT

class `Project.`**`Project`**(*name*, *d_folder*, *file_ext*, *montage*, *sampling_rate*, *params*)

Object containing all methods for creating a new project.

Parameters

- **name** (*str*) – The name of the project
- **d_folder** (*str*) – The folder where the raw data is stored
- **file_ext** (*str*) – File extension
- **montage** (*str*) – Montage to be used
- **sampling_rate** (*int*) – Sampling rate for the txt file
- **params** (*dict*) – Preprocessing parameters for the new project

Variables

- **`quality_thresholds`** – The thresholds to rate te quality of the datasets
- **`ds_rate`** – Sampling rate to create reduced files
- **`rate_cutoffs`** – Sampling rate to the recorded data
- **`config`** – Configuration file with all the project constants
- **`params`** (*`dir`*) – The default parameters to be used
- **`visualization_params`** (*`dir`*) – The default visualisation parameters to be used
- **`CGV`** (*`dict`*) – Constant Global Variables

`create_ratings_structure`()

Method that creates and initializes all data structures based on the data on both data folder and results folder

Parameters None

Returns

- *block_list*
- *processed_list*
- *block_map*
- *n_processed_subjects*
- *n_processed_files*
- *n_block*

- *current*
- *interpolate_list*
- *good_list*
- *bad_list*
- *ok_list*
- *not_rated_list*

static dir_not_hiddens(*folder*, *extn*)

Returns the list of files in the folder, excluding the hidden files

Parameters

- **folder** – The folder in which the files are listed
- **extn** – Extension of the raw file

Returns List of files that are not hidden

Return type files

get_current_block()

Returns the block pointed by the current index

Parameters None

Returns Current block

Return type block

get_quality_ratings(*cutoffs*)

Parameters cutoffs (*dict*) – cutoffs on which the ratings will be decided

Returns ratings – list of block-wise ratings

Return type list

get_rated_count()

Parameters None

Returns

Return type Count for no. of blocks that are yet to be rated

interpolate_selected()

Interpolates all the channels selected to be interpolated

Parameters None

Returns

Return type None

list_preprocessed_subjects()

Method that lists all folders in the results folder

Parameters None

Returns listb – List of all folders in the results folder

Return type list

list_subject_files()

Method that lists all folders in the data folder

Parameters **None**

Returns **listb** – List of all folders in the data folder

Return type list

`static list_subjects`(*root_folder*)
Returns the list of subjects in the folder

Parameters **root_folder** (*str*) – The folder in which the subjects are looked for

Returns **subjects** – List of subjects in the root folder

Return type list

`static make_rating_manually`(*block*, *q_rate*)
Returns q_rate if the block is not rated manually.

Parameters

- **q_rate** (*float*) – The rate to be returned
- **block** – Block for which the rate is returned

Returns Return q_rate if the block is not manually rated. If it is rated manually return 'Manually rated'

Return type rate

`preprocess_all`()
Preprocess all files in the data folder of the project

Parameters **None**

Returns

Return type None

`save_project`()
saves the project information to a JSON file

Parameters **None**

Returns

Return type None

`set_data_folder`(*folder*)
Sets the path folder where the data is stored

Parameters **folder** (*str*) – Path to the raw data folder

Returns **self.data_folder** – Path to the raw data folder

Return type str

`set_name`(*name*)
Sets the name of a new project

Parameters **name** (*str*) – Name of the current project

Returns **self.name** – Name of the new project

Return type str

`set_results_folder`(*folder*)
Sets the path folder where the results will be stored

Parameters **folder** (*str*) – Path where is the data stored

Returns **self.results_folder** – Path to the results folder

Return type str

`to_be_interpolated_count()`

Parameters **None**

Returns

Return type Count for no. of blocks that are yet to be interpolated

`update_project`(*preprocessed*)

Updates the project information dictionary with each blocks information

Parameters **preprocessed** – Preprocessed files

Returns

Return type None

`update_rating_lists`(*block*)

Updates the rating lists according to the rating of the block.

Parameters **block** – block for which the rating list has to be updated

Returns

Return type None

CHAPTER

TWO

BLOCK

class Block.**Block**(*root_path*, *data_filename*, *project*, *subject*)

Object for all operations on an individual dataset.

Initialized using the name and path of the raw data. Preprocess, interpolate, rate for quality, and store those files.

Parameters

- **root_path** (*str*) – root directory of the BIDS project
- **data_filename** (*str*) – BIDS filename with extension
- **project** (*object*) – project object to which this block belongs
- **subject** (*object*) – subject object to which this block belongs

Variables

- **unique_name** (*str*) – raw file name minus extension, used for saving results as well
- **file_ext** (*str*) – raw file extension
- **params** (*dict*) – parameters for preprocessing and calculating quality metrics
- **sampling_rate** – sampling rate of raw data file
- **result_path** (*str*) – directory path to where results are stored for the block
- **rate** (*str*) – current rating of the file (good, bad, ok, not rated)
- **to_be_interpolated** (*list*) – list of channel indices that are to be interpolated
- **auto_bad_chans** (*list*) – list of channel indices detected as bad
- **final_bad_chans** (*list*) – list of channel indices determined to be bad after checks
- **quality_scores** (*dict*) – contains all metrics of quality calculated for the dataset
- **times_committed** (*int*) – used to track how many changes were made to the evaluation of the data

preprocess()

run the block through preprocessing steps, calc quality scores, save files, write log

interpolate()

interpolate the dataset, update quality scores and rating, save files, write log

find_result_path()

Identifies the directory path pointing to where results stored

Following BIDS requirements, we only have either the subject folder or both subject and session.

Parameters **none**

Returns **result_path** – location of results files within BIDS folder

Return type str

`interpolate()`

Interpolates bad channels to create new data and updates info

Parameters **none**

Returns

Return type none

`load_data()`

Load raw data from BIDS folder

Allowing for a number of extensions, loads file

Parameters **none**

Returns

Return type raw MNE object

`preprocess()`

Preprocesses the raw data associated with this block

none

results: dict dictionary containing all the new updates to the block and the preprocessed array

`save_all_files`(*results*, *fig1*, *fig2*)

Save results dictionary and figures to results path

Parameters

- **results** – MNE raw object with info attribute containing
- **fig1** – Figure of ??
- **fig2** – Figure of ??

Returns

Return type none

`update_rating`(*update*)

Takes update about ratings and stores in object

From project level object, get an update on rating info.

Parameters **update** (*dict*) – dictionary of updates

Returns

Return type none

`update_rating_from_file()`

Updates block information from the file currently stored

Checks for results file, if it's there, and informaation, we update. No direct returns, but updates block fields.

Parameters **none**

Returns

Return type none

`write_log`(*updates*)

Writes a log for all of the updates its making/actions performed :Parameters: **updates** (*dict*)

Returns

Return type Updates in log file

CHAPTER

THREE

SUBJECT

class Subject.**Subject**(*data_folder*)

SUBJECT is a class representing each subject in the dataFolder. A Subject corresponds to a folder, which contains one or more Blocks. A Block represents a raw file and it's associated preprocessed file, if any (See Block).

static extract_name(*address*)

extract the name of the subject from subject data folder. :Parameters: **address** (*str*) – path of the subject folder

Returns **name** – name of the subject

Return type str

result_path(*data_folder*)

finds the result folder path for the corresponding subject data folder according to the BIDS folder hierarchy. :Parameters: **data_folder** (*str*) – subject data folder

Returns **result** – corresponding result folder

Return type str

update_addresses(*new_data_path*, *new_project_path*)

This method is to be called to update addresses in case the project is loaded from another operating system and may have a different path to the dataFolder or resultFolder. This can happen either because the data is on a server and the path to it is different on different systems, or simply if the project is loaded from a windows to a iOS or vice versa. :Parameters: * **new_data_path** (*str*) – updated path of the data folder

- **new_project_path** (*str*) – updated path of the project folder

Returns

Return type None

CHAPTER

FOUR

CALCULATE QUALITY

calcQuality.**calcQuality**(*data: numpy.ndarray*, *bad_chans: List*, *overallThresh: float = 50*, *timeThresh: float = 25*, *chanThresh: float = 25*, *apply_common_avg: bool = True*)

Calculates four essential quality metrics of a data set and returns a dictionary of the quality metrics and thresholds/settings used

The four quality metrics: Overall high amplitude data points Timepoints of high variance across channels Ratio of bad channels Channels of high variance across time

Mean absolute voltage is also calculated.

Parameters

- **Data** (*np.ndarray*) – a channels x timepoints data array of EEG
- **bad_chans** (*list*) – a list of the numbers of bad channels
- **overallThresh** (*float*) – overall threshold for rejection of ...
- **timeThresh** (*float*) – threshold for rejecting time segments
- **chanThresh** (*float*) – threshold for rejecting channels
- **apply_common_avg** (*bool*) – if average referencing should be applied

Returns

quality_metrics: a dictionary of the quality metrics and thresholds/settings used The four quality metrics: Overall high amplitude data points Timepoints of high variance across channels Ratio of bad channels Channels of high variance across time

Mean absolute voltage also calculated

Return type dict

CHAPTER

FIVE

RATE QUALITY

`rateQuality.`**`rateQuality`**(*quality_metrics: dict*, *overall_Good_Cutoff: float = 0.1*, *overall_Bad_Cutoff: float = 0.2*, *time_Good_Cutoff: float = 0.1*, *time_Bad_Cutoff: float = 0.2*, *bad_Channel_Good_Cutoff: float = 0.15*, *bad_Channel_Bad_Cutoff: float = 0.3*, *channel_Good_Cutoff: float = 0.15*, *channel_Bad_Cutoff: float = 0.3*)

Rates datasets, based on quality measures calculated with calcQuality().

The possible ratings: Good overall rating Regular overall rating Bad overall rating

Parameters

- **quality_metrics** (*dict*) – a dictionary containing the quality metrics to rate the dataset.
- **overall_Good_Cutoff** (*float*) – cutoff for "Good" quality based on overall high amplitude data points [0.1].
- **overall_Bad_Cutoff** (*float*) – cutoff for "Bad" quality based on overall high amplitude data point [0.2].
- **time_Good_Cutoff** (*float*) – cutoff for "Good" quality based on time points of high variance across channels [0.1].
- **time_Bad_Cutoff** (*float*) – cutoff for "Bad" quality based on time points of high variance across channels [0.2].
- **bad_Channel_Good_Cutoff** (*float*) – cutoff for "Good" quality based on ratio of bad channels [0.15].
- **bad_Channel_Bad_Cutoff** (*float*) – cutoff for "Bad" quality based on ratio of bad channels [0.3].
- **channel_Good_Cutoff** (*float*) – cutoff for "Good" quality based on channels of high variance across time [0.15].
- **channel_Bad_Cutoff** (*float*) – cutoff for "Bad" quality based on channels of high variance across time [0.3].

Returns **dataset_qualification** – a dictionary indicating is the dataset if "Good" = 100, "Regular" = 50 or "Bad" = 0.

Return type dict

CHAPTER

SIX

PREPROCESS

class preprocess.**Preprocess**(*eeg*, *params*)

Preprocess class for pyautomagic preprocessing pipeline: preprocess performs pyprep's prep_pipeline, then filters the eeg data, performs optionaleog_regression, preforms RPCA to remove noise, and lastly can output plots of the data at various stages in the process.

Parameters

- **eeg** (*mne.io.Raw*) – mne raw object containing eeg data and all the data's information.
- **params** (*dict*) – dictionary of parameters <default> params = {'line_freqs' : 50,

  'filter_type' : 'high', 'filt_freq' : None, 'filter_length' : 'auto', 'eog_regression' : False, 'lam' : -1, 'tol' : 1e-7, 'max_iter': 1000, 'interpolation_params': {'line_freqs' : 50,

  'ref_chs': eeg.ch_names, 'reref_chs': eeg.ch_names, 'montage': 'standard_1020'}

Variables

- **eeg** (*mne.io.Raw*) – mne raw object containing eeg data and all the data's information.
- **eog** (*mne.io.Raw*) – mne raw object containing eog data and all the data's information.
- **bad_chs** (*List*) – list of the names of all the detected bad channels
- **params** (*dict*) – dictionary of parameters described above
- **index** (*numpy.array*) – array of bad channel indices
- **filtered** (*mne.io.Raw*) – mne raw object containing eeg data after filtering
- **eeg_filt_eog** (*mne.io.Raw*) – mne raw object containing eeg data after filtering, and eog_regression
- **eeg_filt_eog_rpca** (*mne.io.Raw*) – mne raw object containing eeg data after filtering, eog_regression, and robust PCA (final cleaned data)
- **noise** (*numpy.array*) – array of the noise removed from rpca
- **automagic** (*dict*) – automagic holds information about the progress of the pipeline
- **fig1** (*matplotlib.pyplot.figure*) – figure of 6 subplots at each stage of the preprocess pipeline
- **fig2** (*matplotlib.pyplot.figure*) – Figure of the final cleaned eeg data

perform_prep()

Calls pyprep's PrepPipeline and detects bad channels in the data.

perform_filter():
Performs initial filter (high, low, or band-pass) and removes line noise.

def perform_eog_regression
If requested, it will remove artifact from eog data.

perform_RPCA()
Uses Robust Principal Component Analysis to remove noise from the data.

plot(self, show=True):
Outputs plots of data at each point in the preprocess pipeline.

fit(self):
Perform the full preprocessing pipeline for pyautomagic (modeled from matlab's automagic package).

fit()
Fit Perform the full preprocessing pipeline for pyautomagic (modeled from matlab's automagic package).

Returns

- **eeg_filt_eog_rpca** (*mne.io.Raw*) – Corrected Data
- **fig1** (*matplotlib.pyplot.figure*) – figure of 6 subplots at each stage of the preprocess pipeline
- **fig2** (*matplotlib.pyplot.figure*) – Figure of the final cleaned eeg data

perform_RPCA()
Uses Robust Principal Component Analysis to remove noise from the data.

Returns

- **eeg_filt_eog_rpca** (*numpy.array*) – eeg data after filtering, eog_regression, and robust PCA (final cleaned data)
- **noise** (*numpy.array*) – array of the noise removed from rpca

perform_eog_regression()
If requested, it will remove artifact from eog data.

Returns **eeg_filt_eog** – Filtered eeg data, with eog regression

Return type mne.io.Raw

perform_filter()
Performs initial filter (high, low, or band-pass) and removes line noise.

Returns **filtered** – Filtered eeg data

Return type mne.io.Raw

perform_prep()
Calls pyprep's PrepPipeline and detects bad channels in the data.

Returns **bad_chs** – List of all the bad channel names

Return type List

plot (*show=True*)
Outputs plots of data at each point in the preprocess pipeline.

Returns

- **fig1** (*matplotlib.pyplot.figure*) – figure of 6 subplots at each stage of the preprocess pipeline
- **fig2** (*matplotlib.pyplot.figure*) – Figure of the final cleaned eeg data

CHAPTER SEVEN

PERFORM EOG REGRESSION

`perform_EOG_regression.`**`perform_EOG_regression`**`(`*EEG*, *EOG*`)`

The artifacts due to EOG activity are removed from the EEG data using the subtraction method that relies on the linear transformation of the EEG signal

Parameters

- **EEG** (*np.ndarray*) – The input EEG data
- **EOG** (*np.ndarray*) – The input EOG data

Returns **clean_EEG** – Cleaned EEG signal from EOG artifacts

Return type np.ndarray

References

[1] Pedroni, A., Bahreini, A., & Langer, N. (2019). Automagic: Standardized preprocessing of big EEG data. Neuroimage, 200, 460-473. doi: 10.1016/j.neuroimage.2019.06.046

CHAPTER

EIGHT

PERFORM FILTER

performFilter.**performFilter**(*EEG*, *sfreq*, *filter_type=None*, *filt_freq=None*, *filter_length='auto'*)

This function filters EEG data using Hamming windowed sinc FIR filter with input filter type and parameters.

Parameters

- **EEG** (*ndarray, shape (…, n_times)*) – Input EEG data to be filtered.
- **sfreq** (*float* | *None*) – The sample frequency in Hz.
- **filter_type** (*str*) – The filter type, can only take 'low', 'high' or 'notch', defaults to None
- **filt_freq** (*float* | *None*) – The filter frequency in Hz, defaults to None
- **filter_length** (*str* | *int*) – Length of the FIR filter to use, defaults to 'auto'.

Returns The filtered EEG data.

Return type ndarray, shape (…, n_times)

CHAPTER

NINE

ROBUST PRINCIPAL COMPONENT ANALYSIS

`rpca.`**`rpca`**(*M*, *lam=-1*, *tol=1e-07*, *maxIter=1000*)

Perform Robust Principle Component Analysis:

Performs a Robust Principal Component Analysis on the EEG data with the specified parameters: Lamda, Tolerance, and Maximum number of Iterations. The function outputs the EEG data with the noise removed as well as the nosie that was removed.

Adapted from Cian Scannell - Oct-2017 (https://github.com/cianmscannell/RPCA) Computes rpca separation of M into L(low rank) and S(Sparse) using the parameter lam this uses the alternating directions augmented method of multipliers.

Parameters

- **M** (*npumpy.darray*) – 1st parameter, EEG Data (must include)
- **lam** (*double*) – 2nd parameter, Lamda paramter for RPCA (default = 1/(sqrt(# of Colunms))
- **tol** (*double*) – 3rd parameter, Tolerance (defalut = 1e-7) RPCA param
- **maxIter** (*int*) – fourth parameter, Maximum Iterations (deafult = 1000)

Returns

- **Data** (*npumpy.darray*) – Corrected Data (Low rank matrix)
- **Error** (*npumpy.darray*) – Noise removed from the data (Sparse Matrix) note: M = L + S

`rpca.`**`soft_thres`**(*x*, *eps*)

Cian Scannell - Oct-2017 Soft thresholds a matrix x at the eps level i.e ST(x,eps)_ij = sgn(x _ij) max(**|x _ij|** - eps, 0)

Parameters

- **x** (*npumpy.darray*) – first parameter, values to be thersholded
- **eps** (*double*) – second parameter, thershold

Returns **np.multiply(a,b)** – thersholded values, where anythign under the threshold was set to zero

Return type npumpy.darray

PYTHON MODULE INDEX

INDEX