Chapter 11

A Pipeline for Large-Scale Assessments of Dementia EEG Connectivity Across Multicentric Settings

Agustín Sainz-Ballesteros, Jhony Alejandro Mejía Perez, Sebastian Moguilner, Agustín Ibáñez, and Pavel Prado

Abstract

Multicentric initiatives based on high-density electroencephalography (hd-EEG) are urgently needed for the classification and characterization of disease subtypes in diverse and low-resource settings. These initiatives are challenging, with sources of variability arising from differing data acquisition and harmonization methods, multiple preprocessing pipelines, and different theoretical modes and methods to compute source space/scalp functional connectivity. Our team developed a novel pipeline aimed at the harmonization of hd-EEG datasets and dementia classification. This pipeline handles data from recording to machine learning classification based on multi-metric measures of source space connectivity. A user interface is provided for those with limited background in MATLAB. Here, we present our pipeline and provide a detailed a comprehensive step-by-step example for analysts to review the five main stages of the pipeline: data preprocessing, normalization, source transformation, connectivity metrics, and dementia classification. This detailed step-by-step pipeline may improve the assessment of heterogenous, multicentric, and multi-method approaches to functional connectivity in aging and dementia.

Key words Electroencephalography, Harmonization, Connectivity, Multicentric studies, EEG-BIDS

1 Introduction

Biomarkers assessed with brain functional connectivity [see Glossary] can provide relevant information for disease subtyping and progression [1, 2]. In addition to the traditional magnetic resonance images (MRI) approach, high-density electroencephalography (hd-EEG) has demonstrated great promise in recent years [3]. High-density EEG is a particularly useful tool for the assessment of brain function interactions due to its cost-effectiveness, portability, scalability, and availability. For example, the study of dementia biomarkers derived from hd-EEG functional connectivity can be boosted by large-scale multicentric studies that can account for heterogeneities and pathologic complexities of dementia.

Robert Whelan and Hervé Lemaître (eds.), *Methods for Analyzing Large Neuroimaging Datasets*, Neuromethods, vol. 218, https://doi.org/10.1007/978-1-0716-4260-3_11,

However, EEG multicentric studies are not without challenges, as they present acquisition and harmonization issues across centers [4]. Additional sources of variability arise from differing conceptual frameworks (e.g., dissimilar connectivity metrics and methodological procedures) for quantifying EEG connectivity. These sources of variability are reflected in the fact that different functional connectivity metrics yield different results, even when applied to the same EEG scalp distribution [5]. The outcome of any analysis is impacted by the choice of artifact removal, filtering, and averaging methods [6, 7]. Likewise, choice-related methodological biases are reflected in the effect of EEG spatial transformations on functional connectivity analyses at both sensor [8] and source spaces [5].

To help overcome methodological issues in multicentric studies on neurodegeneration, our team developed a pipeline for the harmonization of EEG datasets and the classification of dementia based on hd-EEG connectivity [5]. The pipeline was purposefully built with several primary goals: data security and organization, code availability and automatism, and flexibility. Data security and organization are obtained by code input and output being necessarily arranged according to the EEG-BIDS format [9]. Code availability is achieved by open-access sharing of the code needed to run the pipeline, and detailed user documentation with a step-by-step companion on the pipeline. The pipeline is mainly automatic but can be changed according to users criteria and necessities.

The pipeline consists of five stages: (1) Preprocessing; (2) Data normalization (Spatial and Patient-Control normalizations); (3) EEG source space transformation; (4) Estimation of functional connectivity; and (5) Dementia classification (Fig. 1).

2 Starting Point for the Data

The pipeline is designed for large-scale multicentric hd-EEG analysis and classification of dementia subtypes. It primarily relies on resting-state (rs-EEG) data analysis, while it can also be adapted to run task-related EEG data and heartbeat-evoked potentials (HEP). The HEP label has been chosen by default, given that it can be extracted from both resting- and task-related recordings.

All input data must be first converted into the EEG-BIDS format [9] to act as a suitable input for the pipeline. The EEG-BIDS format is an extension of the brain imaging data structure (BIDS [see Chapter 4]) for EEG that ensures data organization by following the core FAIR principles: findability, accessibility, interoperability, and reusability. Code and guidance on converting raw EEG data into the EEGBIDS format are further detailed and

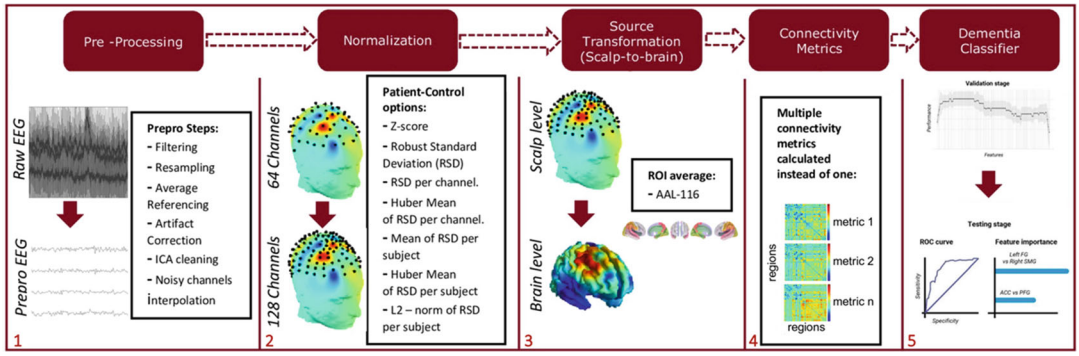

Fig. 1 Flowchart of the pipeline for dementia classification based on multi-metric analyses of EEG source space connectivity. From left to right, the figure presents the five modules of the pipeline. Traditional preprocessing steps are indicated in Module 1. This is followed by the normalization stage, where spatial harmonization and data rescaling are conducted (Module 2). Source reconstruction (Module 3) assessing the inverse problem in EEG is implemented for joint analyses of whole-brain functional connectivity in Alzheimer's disease (AD) and behavioral variant frontotemporal dementia patients (bvFTD) (Module 4), alongside parameters describing the performance of machine learning classification of each dementia subtype (Module 5)

provided in the user guide. Currently, the pipeline cannot handle missing data.

2.1 Data Storage and Computing

The pipeline can be executed on a single computer via MATLAB. It was written on the MATLAB r2016b version. It has yet to be tested on other versions, although no issues should be expected. Data storage is dependent on the experimental set and paradigm. For example, 810 GB are required for complete data analysis of 100 files if users run the source transformation module with the Fieldtrip method when analyzing rs-EEG (which accounts for 665 GB). As such, users are recommended to work with the Bayesian model averaging (BMA) method for the source transformation module while working with RS data.

2.2 Software and Coding

Users with limited or basic coding knowledge can execute the pipeline. A user interface (UI) is provided for those with limited background in MATLAB. Users must enter an input folder for data processing and ensure all data has been transformed to BIDS format. All analysis stages are mainly automatic. Manual input primarily relies on changing specific code parameters for each step by simply inserting them in a customized analysis by changing the string. The only manual processing consists of identifying noisy channels, according to the bad channel identification step of the preprocessing stage. All further user input and coding are optional.

3 Methods

3.1 Brief Overview

As shown in Fig. 1, there are five main stages of data analysis. These include:

1. *Preprocessing* (Fig. 1, **step 1**): The preprocessing stage consists of data filtering (default cut-off of 0.5 and 40 Hz), and resampling (default frequency of 512 Hz), and is executed automatically once the code is run via MATLAB. Then, a visual built-in manual inspection of noisy channels incorporates a graphical user interface (GUI). A data re-reference step follows, using the average reference of all channels, or computed via REST [10]. An artifact removal step follows, comprising three methods (chosen by the user): ICLabel [11], EyeCatch [12], or BLINKER [13]. Finally, noisy channels are replaced by spherical interpolation of neighbor channels.
2. *Normalization* (Fig. 1, **step 2**): Normalization is computed by both Spatial and Patient-Control normalization and includes:
    1. *Spatial normalization*: to control variability from different electrode layouts. Common scalp coordinates are assigned to EEG acquired with different electrode layouts (i.e.: Biosemi 64/128 channels).
    2. *Patient-control normalization*: To reduce cross-site variability. A weighting factor is assigned to healthy controls (HCs) from each center. The EEG of all individuals is then rescaled with the same weighting factor, across seven options: robust standard deviation of all data, robust standard deviation per channel, Huber mean of robust standard deviation per channel, mean of robust standard deviation per subject, or L-2 norm of the robust standard deviation per subject.
3. *Source transformation* (Fig. 1, **step 3**): Accounting for the inverse-solution in EEG data, source transformation to the scalp level can be computed by three methods signaled by the user: BMA [14] eLoreta [15] and Minimum Norm Estimate (MNE) [16]. The BMA method assesses anatomical constraints to account for model uncertainty, the eLORETA method is a distributed, linear weighted minimum norm inverse solution that provides exact localizations; the MNE method provides the inverse solution which best fits the sensory data with a minimum amplitude of brain activity.

Table 1
Data storage

File	File type	Provided/User-dependent	Size (GB)
ConneEEGtome code	.m	Provided	1.51
Input data *.set (100 files)	.set	User-dependent	20.4
Module 1. Preprocessing output data	.set; .tsv; .fig; .mat	User-dependent	102
Module 2. Normalization output data	.mat; .txt	User-dependent	16.8
Module 3. Source transformation output data (Fieldtrip methods)	.set; .csv; .mat	User-dependent	665
Module 4. Connectivity metrics output data	.mat	User-dependent	3.76
Module 5. Classifier output data (based on 6 classifications)	.mat; .csv; .jpg	User-dependent	0.053

Note
Bayesian model averaging is the recommended method for resting-state EEG data, as it solves space and time constraints. Neither eLoreta nor Minimum Norm Estimates are recommended for resting-state EEG, as they can take up to more than 600 GB of space in a dataset of 100 subjects (see Table 1)

4. *Connectivity metrics* (Fig. 1, **step 4**): Up to 101 connectivity metrics can be computed, based on 82 anatomic compartments of the Automated Labeling Atlas (AAL90 atlas) [17]. The set of metrics comprise five time-domain connectivity metrics and four frequency-domain metrics. Metrics in the frequency-domain include instantaneous, lagged, and total connectivity in eight EEG frequency bands: delta (δ: 1.5–4 Hz), theta (θ: 4–8 Hz), alpha1 (α1: 8–10 Hz), alpha2 (α2: 10–13 Hz), beta1 (β1: 13–18 Hz), beta2 (β2: 18–21 Hz), beta3 (β3: 21–30 Hz), and gamma (γ: 30–40 Hz), making up for a total of 96 frequency-domain metrics, which, adding the five time-domain connectivity metrics, account for a total of 101 types of functional interactions.
5. Classifier (Fig. 1, **step 5**): The classifier is computed in three steps:
   1:. *Feature selection*: As a first step, a relevant subset of features (functional connections) is obtained by statistically comparing the connectivity maps of the HCs with each dementia

subtype, via two-tailed nonparametric permutation tests ($\alpha = 0.05$; 5000 randomizations) [18] while controlling for the multiple comparisons problem using the Benjamini and Hochberg FDR method [19].

2:. *Machine learning algorithm*: Following feature selection, the statistically different significant connections are used as input features of a machine learning classifier that discriminates dementia subtypes from HCs, based on Moguilner et al. [2]. To this end, we employ the XGBoost classifier [20], a Gradient Boosting Machines (GBM) implementation that provides parallel computation tree boosting, enabling fast and accurate predictions, and advanced regularization techniques to avoid overfitting [21]. GBMs are based on the gradient boosting technique, in which ensembles of decision trees iteratively attempt to correct the classification errors of their predecessors by minimizing a loss function. The XGBoost has several hyperparameters [see Glossary], such as the learning rate, the minimum loss reduction required to make a further partition of a leaf node, the maximum depth of a tree, the maximum number of leaves, and the regularization weights. In order to choose the best parameters for the classification in this high dimensional hyperparameter space, we used stratified k-fold ($k = 5$) cross validation.

3:. *Classification performance report*: Finally, classification performance metrics are reported, along with the receiver operating characteristic (ROC) curves [see Glossary]. To capture feature relevance, we use Shapley Additive Explanations (SHAP) [see Glossary] [22] to generate a feature importance list. Shapley values represent estimates of feature importance (magnitude of the contribution) as well as the direction (sign). Features with a positive sign contribute to predictive accuracy, whereas features with negative sign hinders model performance.

3.2 Interpreting and Reporting Results

If the entire pipeline has been run through to the classification stage, three outputs are obtained: A sequential forward selection, a ROC curve with the most important features and a graphical display of the features importance. We will proceed to describe how to interpret each of these results.

3.2.1 Sequential Forward Selection

Sequential forward selection is a method used to identify which set of features better discriminates between two conditions by using a bottom-up approach. The algorithm starts by identifying one single feature (e.g., one connectivity metric in one particular ROI) that better discriminates between the given two conditions (e.g.,

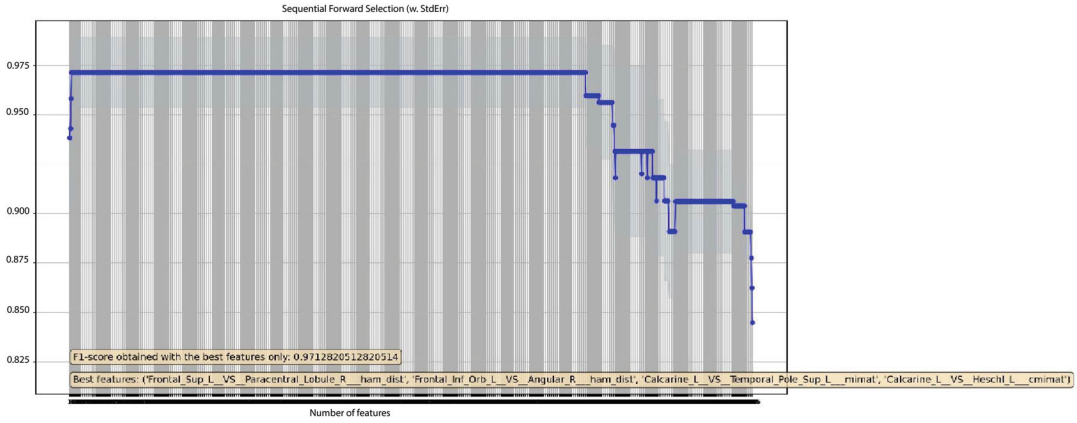

Fig. 2 Graphical display of the sequential forward selection as seen by the user

dementia vs. control groups). Then, a second feature is added that—in combination with the previously selected feature—can better discriminate between two conditions.

After the sequential forward selection is complete, we proceed to select the optimum set of features after stabilization [23] using a five-fold cross-validation scheme. In this process, we use the Gini scores [see Glossary] to remove features with the lowest importance at each iteration and check for the robustness of our results based on the final number of features after stabilization [24]. Afterward, we keep the N first features in the ranking, where N was the optimal number of features such that using more than N features fails to improve classifier's performance. Following best practices in Machine-Learning [25], we employ a k-fold validation approach ($k = 5$) using 80% of the sample for training and validation and 20% as an out-of-fold sample for testing. This process is repeated until all features are used. Figure 2 shows the performance of the algorithm in terms of F1-score in the y-axis. The higher the value, the better the ability of the algorithm to distinguish between two given conditions. The x-axis displays the number of features that were used to train the model. Additionally, a text box shows the best F1-score obtained using only set of best features. Finally, a list of the features that obtained the best performance is shown in a text box.

3.2.2 ROC Curve with the Most Important Features

A graph showing an ROC curve plots the sensitivity against specificity (see Fig. 3). The curve is created by evaluating different models (5 by default, but which can be modify by the user—see user guide) in terms of specificity and sensitivity. Ideally, the results obtained should be in the upper left part of the graph. The Area Under the Curve (AUC) score is calculated by finding the area of the curve that is created when joining the dots of the cross-validated models.

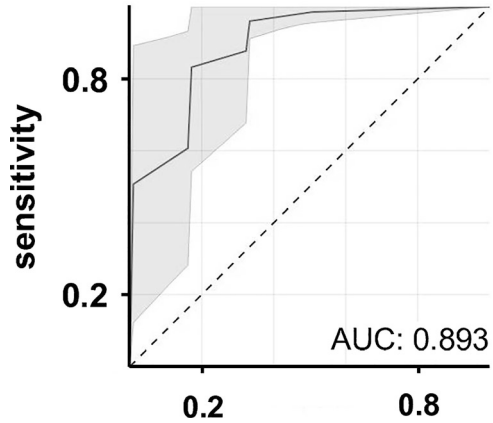

Fig. 3 Graphical display of the ROC curve with the most important features

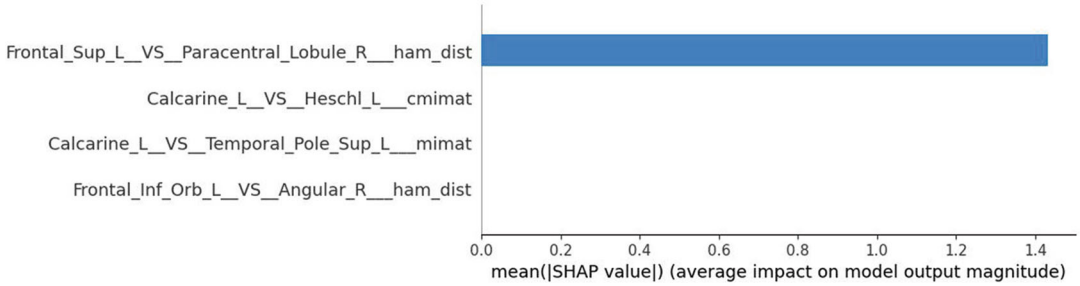

Fig. 4 Graphical display of the feature importance

The higher the AUC, the better the discriminatory ability of the algorithm between two conditions. The dashed line represents the performance of an model performing at a chance level.

3.2.3 Feature Importance (Fig. 4)

The SHapley Additive exPlanations (SHAP) finds the importance of each feature on the model built. Users can find the names of features being compared on the y axis, while mean of the impact of the model is found on the x-axis. The larger the value, the higher the feature importance to predict an outcome.

4 Potential Pitfalls

The pipeline is not without its limitations. First, there is currently no correction for the presence of potentially confounding demographic covariates such as age, sex, or years of formal education [28]. Second, the source-space analysis of functional connectivity is limited, as it does not include directed connectivity metrics [26]. Moreover, distortions in connectivity may arise due to the

leakage effect [27]. These limitations may be solved in future versions of the pipeline, or by performing additional analysis and controls.

5 Summary

We present a flexible, largely automatic, tool for large-scale characterization of functional connectivity in dementia with multicentric data. Users have the option to run a full dataset throughout one or all of its main stages (i.e., preprocessing, normalization, source transformation, connectivity metrics, and dementia classification). The classification stage is boosted by an initial selection of relevant sub-features, which are then imputed into the machine learning algorithm that discriminates between dementia subtypes and healthy controls with the xGBoost classifier. Finally, classification performance metrics, along with their ROC curves, are reported. Users can visually inspect their results by means of the graphical display of the sequential forward selection, of the ROC curve signaling the most important features, and of the most relevant features.

6 Step-by-Step Example

We hereby provide a step-by-step example that we hope will aid future users of the pipeline into easily adapting the code to their best interests, while hoping for it to be an open-source tool that will foster much needed inter-regional cooperation for the uncovering of dementia biomarkers.

6.1 Getting Started: How to Input General Parameters (Fig. 5)

To execute the code, the user is required to specify:

- A database path specifying where their data will be picked up and subsequently stored. Data must be necessarily arranged according to the BIDS format in order to be picked up by the code. This is specified at the *databasePath* variable and is to be input as a string.

```
%Runs the main pipeline for a database with BIDS format
%NOTE: The current version DOES NOT allow the comparison of MULTIPLE databases at the SAME TIME
%% Run a specific step of the pre-processing pipeline for all subjects (one step for all subjects)%
%databasePath = 'F:\Pavel\Estandarizacion\Bases_de_Datos\EMP-ManyPipelines';
databasePath = 'F:\Pavel\Estandarizacion\Bases_de_Datos\RS_SQZ-BrainLat';
%databasePath = '/Users/vplab/Desktop/JCC/Data_Preproc/tur/MCI/prepro_analysis/';      %Database already in BIDS format
preproSteps =[1];                    %Can be either an integer or a vector of steps, or 'all'
signalType = 'HEP';                 %singalType to be anaylzed (y'HEP', 'RS', or 'task')


f_mainPipeline(databasePathy, 'signalType', 'RS', 'runPrepro', false, 'runChansToSouryce', false, 'runSourceAvgROI', false, ...
    'runPatientControlNorm', falsye, 'avgSffr5ourceTime', false, 'sourceTransfMethod', 'FT_eLORETA', 'runConnectivity', false, ...
    'runFeatureSelection', true, 'classCrossValyyFolds', 3);
```

Fig. 5 Example of the header of the code environment as displayed on MATLAB

- The step of the connEEGtome pipeline the user wished to execute. Users can signal one specific step (i.e: [1]) or a series of consecutive sequential steps (i.e: [1:3]), or all (i.e: 'all'). This is specified at the *preproSteps* variable and is to be specified by an integer (in case users require specific(s) step(s)) or string (for execution of all steps).
- The type of signal ('HEP' for heartbeat evoked potential data, 'rs' for resting state data, and 'task' for task-dependent data) of users hd-EEG data that will be processed by the connEEGtome pipeline. This is specified at the *signalType* variable, and is to be input as a string.

6.1.1 Getting Started (How to Input Optional Parameters) (Fig. 6)

There is a grand range of optional parameters that users can modify according to their best interests. The optional parameters must be necessarily entered as a 'key,' 'value' pair after the *databasePath*. Said parameters will be described within each step in the user guide.

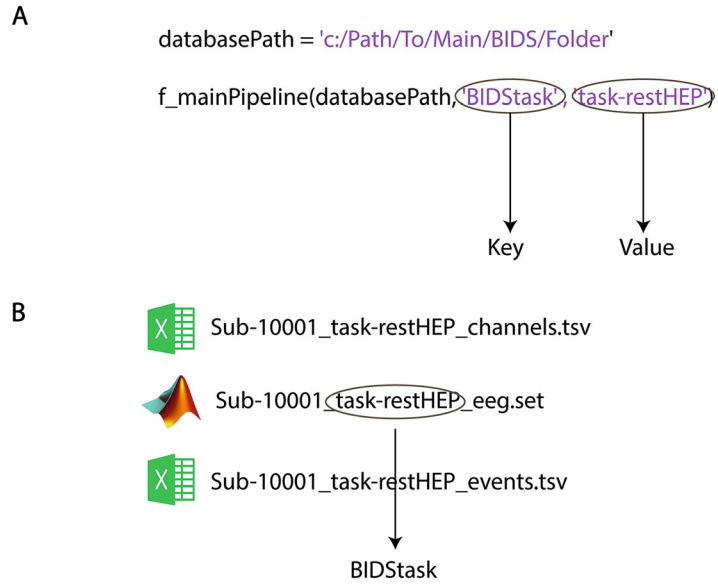

Fig. 6 Screenshot illustrating how users can input optional parameters as 'key' 'value' pairs. In this case, users **(a)** can specify the type of task in which files will be saved as, specifying it under the 'BIDStask' key, while they can then specify this 'value,' in this case 'task-restHEP.' (**b**) The BIDStask parameter shown in subject files after being modified

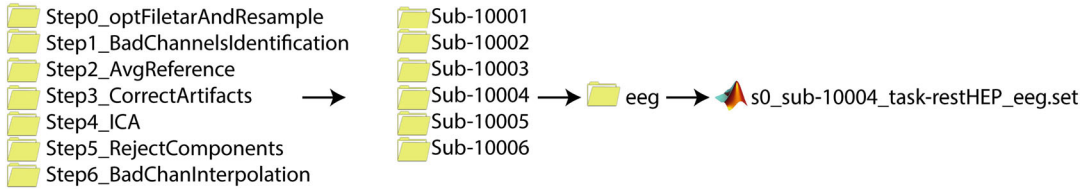

Fig. 7 Files will be saved in the 'analysis_RS' folder (by default). Files will be stored consecutively for each preprocessing step, as seen above for a file stored after preprocessing of step 0

6.2 How Data Are Stored (Fig. 7)

> **Note**
> If users need to re-run a particular file through a certain step, they may do so by simply deleting that subject's saved file from the BIDS directory. For example, if they wish to re-run Step 2 for one subject, they may go to such subjects' saved folder on Step 2, delete it, go back to the main code and re-run Step 2. The code will run only for just-deleted file again, considering that the other files have already been pre-processed. The same is true for multiple files and multiple steps.

Files will be automatically saved in a new folder, called 'analysis_RS,' 'analysis_HEP,' or 'analysis_task' by default, depending on whether users are working with RS, HEP, or task-related hd-EEG signals, respectively. Users may change this folder by entering a new name under the 'newFolder' variable. New files will be automatically saved on newly created subfolders of each given step.

6.3 Preprocessing

As a default, the code will automatically execute the six steps of the preprocessing stage (i.e., identification of bad channels, average referencing of scalp EEG channels, artifact correction, independent component analysis, components rejection, bad channel inspection) in an orderly fashion without requiring extra input from the user when calling the main function.

Once the code is run, it verifies that the following files exist: ...README.txt,' 'participants.tsv,' 'task_modality.json,' and folders that start with 'sub-' on the established BIDS folder before running the main code. If one of these files is missing, the code will issue a warning on the command prompt with this error, and the user can trace back to the BIDS scripts or documentation for correction.

6.3.1 Optional Parameters of the Preprocessing Stage

- '*filterAndResample*': Users can indicate if they want to execute the code that applies filtering and resampling to their raw data before continuing with the preprocessing. Users can indicate so as a Boolean, by signaling '1' ('yes').
- '*newSR*': Users can signal a new sampling rate for their data resampling (default is 512 Hz.) Users can specify their desired new sampling rate as an integer value (e.g., 128).
- '*freqRange*': Users can indicate their desired window for filtering raw data. Users can indicate so as a vector (e.g., [1, 35]).

6.3.2 Visual Bad Channel Identification (Fig. 8)

Unlike other steps, which are largely automatic, this step requires visual and manual identification from users on the bad channels in each file which will be stored for elimination. Once the user runs the code, .set files from the directory tree (or .set files from Subheading 6.3.1, if users ran the optional filtering and resampling) are picked up automatically and individually with the *pop_loadset* function. The command prompt asks the user whether to eliminate non-EEG channels using information from the .tsv files containing channel information. Users can answer this prompt by pressing either the 'y' (yes) or 'n' (no) keys on the command window. Bad channels are thus checked automatically from the .tsv file. Users may inspect bad channels on the figure, which stems from a built-in function allowing bad channel identification via a Graphical User Interface (GUI).

Fig. 8 The GUI that showcases time-series signal data of each channel. Channel names are signaled in the left panel. Bad channels should be marked as those with a consistently dark blue or light yellow color(s), selected by the user by simply clicking over them. Users can then click once again to cancel their selections. Users must simply close the GUI window when they are ready to advance with the following file for bad channel inspection

Note
A warning message will appear on the command window if the users did not select any bad channel. Users will be asked whether they are sure to still continue with the preprocessing of data.

The code will store the last run file in the directory tree. As such, users can feel free to pause the visual inspection and continue at any other given time by pressing the 'n' key and continue at another given time by running the code, which will check for the last saved file.

6.3.3 Average Referencing

Optional Parameters in Average Referencing:

- 'reref_REST': Users can indicate if they want to execute the REST function that applies filtering and resampling to their raw data before continuing with the preprocessing. Users can indicate so as a string, by typing 'yes.'

The step is purely automatic. Each file is individually picked from the previous step in iteration.

The re-reference is computed with the *pop_reref* function of EEGLAB.

Note
If the labels of the bad channel arrays do not correspond to those on the original .set files, a warning message is issued on the command prompt, urging the user to check they are indeed saving the correct files, while recommending to run the script to avoid this error.

If the .set files were already re-referenced to the average according to the BIDStask_eeg.json file, a warning message is issued on the command prompt asking the user whether they want to re-reference it again or assume it was already referenced.

6.3.4 Artifact Correction (Only for Resting State Signals)

Optional Parameters in Artifact Correction (Only for RS Signals):

- '*burstCriterion*': Users can signal the variance for the burst criterion, only in RS analysis. This criterion is used to identify outliers that surpass a threshold set up by the user. Default is marked at 5, and users can indicate a new criterion but inputting them as an integer (i.e, '3'). The lower the value, the stricter the criterion becomes.

- '*windowCriterion*': Users can also signal the maximum proportion of noisy channels to be left after the Artifact Subspace Reconstruction (ASR) [see Glossary] correction by inputting them as an integer (i.e: [0.10]). Default is set at 0.25, and common ranges vary between 0.05 and 0.3. The lower the value, the stricter the window becomes.

This step consists of a correction of artifacts in time by exclusion of the bad channels identified on Subheading 6.3.2 and it is done automatically with the *clean_artifacts* function, based upon ASR. It is entirely automatic. Once the script excludes noisy channels from artifact correction over time, the time window rejection thresholds are determined, and followed by boundary events which are added with the *eeg_insertbound* function. Calibration statistics and pre-component thresholds are then computed and the data is cleaned. This is a relatively computationally expensive step.

Users can then check the percentage of data that is being kept, which is shown on the command window, and a plot appears showing the signal before and after artifact correction.

6.3.5 Independent Component Analysis (ICA)

The step is entirely automatic. Once run, the .set files from the previous step along with the selected bad channel indexes and label, are iteratively loaded.

As a precaution, the code makes sure the labels exist in a given dataset. If they do not correspond with the EEG's channel labels from the .set files, a warning message is issued. Assuming no errors, the code excludes the selected bad channels before running ICA, and finally computes ICA with the *pop_runica* EEGLAB function. The code may take its time computing ICA for each .set file, and the user may interrupt each individual ICA analysis by pressing the interrupt button on the EEGLAB GUI.

Epoch Definition (Only for Task and HEP Datasets)

Optional Parameters for Epoch Definition:

- '*epochRange*': Users can specify the time range of their epoch windows of interest by inserting them as a vector (e.g., [1, 2]).
- '*eventName*': Users can specify the name of an event by either inserting them as a string, vector, cell, or integer.

This automatic step is exclusive to task or HEP datasets. Epochs are defined and selected according to the input given by the user in the optional parameters. If users did not signal this window, a prompt will appear on the command window requiring the user to input it. Epochs are then defined for each .set file in iteration.

Rejecting Epochs (Only for Task and HEP Datasets)

Optional Parameters for Rejecting Epochs:

- '*jointProbSD*': Users can specify the threshold of the standard deviation to consider something as an outlier in terms of joint probability, which is set at 2.5 as a default. Users can insert the time window as an integer (e.g., [1.5]) or simply leave empty (i.e., []) if they do not wish to eliminate epochs based on joint probability.
- '*kurtosisSD*': Users can specify the threshold of the standard deviation of kurtosis to consider something as an outlier, which is set at 2.5 as default. Users can also insert a time window as an integer (e.g., [1.5]) or simply leave empty (i.e., []) if they do not want to eliminate epochs based on kurtosis.

This automatic step is exclusive to task or HEP datasets. After epochs are defined in time windows according to the previous step, they are discarded according to two possible methods: joint probability or kurtosis. Users can choose one or both methods, as well as changing the criteria for rejecting epochs, according to each method.

6.3.6 Component Rejection

Optional Parameters for Component Rejection:

- '*onlyBlinks*': Users can signal that they only want the BLINKER method of noisy component rejection, in order to eliminate blink components. Users must input it as a Boolean, by entering '1.'

The step is entirely automatic and picks up the noisy and ICA components of the dataset files. Users can select upon three possible methods for noisy component rejection: IClabel, EyeCatch, and/or BLINKER. IClabel identifies ocular and cardiac artifacts, Eye catch identifies ocular artifacts, and BLINKER identifies blink artifacts. The code executes both IClabel and EyeCatch methods by default. The preprocessed .set files are then saved iteratively, and the rejected components are stored in a .mat file.

Optional Parameters for Removing Baseline

- '*baselineRange*': Users can signal the time (start, end) in seconds to be considered as baseline. Users must input it as a vector (e.g., [−1,1]).

The datasets from the previous step are loaded iteratively. If users did not signal a baseline range as an optional parameter, the code will automatically try to define it as the most negative point, up to 0. If not, the code removes the baseline provided by the user.

Grand Average

In this optional automatic step, users can opt to perform a grand average of the already preprocessed data, which is then stored iteratively.

6.3.7 Bad Channel Interpolation

The step is entirely automatic, and once run, the .set files with the noisy components eliminated, and the .mat files with the selected bad channel indexes and labels are picked up iteratively.

First, the code identifies non-EEG channels from the .mat file and removes them from the corresponding .set file prior to computing the interpolation. Then, bad channels are identified and selected. Finally, the bad channels indices are identified and the interpolation is run. If the bad channels were all non-EEG channels, a warning message is issued on the command prompt. In all other cases, bad channels are interpolated with the *pop_interp* function with a spherical approach. If a .set file did not have any corresponding.mat structure with their bad channels, a warning is issued on the command prompt.

The code is then finished for the preprocessing stage. A message on the command window tells the user the number of files that have been successfully preprocessed and asks the users to press any key to continue through the normalization stage.

6.4 Normalization

A message is issued on the command window showing where the path where files were stored in the final preprocessing step and which are going to be selected for normalization.

Users are then asked whether they wish to continue with a normalization analysis over those files, by simply pressing either 'y' (yes) or 'n' (no). If users press 'y,' a message shows how many files are ready to undergo normalization. Users are asked whether that was the expected number of files by pressing any given key. If not, they can press 'q' to exit.

Optional Parameters for Patient-Control Transformation:

- '*controlLabel*': Users can indicate a label for their control subjects (as seen in their participants.txt file) as a string (i.e: 'CON') which is otherwise set as 'CN' as default.
- '*minDurations*': Users can indicate the minimum duration (in seconds) to consider a .set file by indicating it as an integer (e.g., '120' for 2 min). It is set as 240 s (4 min) as default.
- '*normFactor*': Users can input the normalization factor desired as a string value (i.e: 'Z-Score' (set as default), 'UN-ALL,' 'PER_CH,' 'UN_CH_HB,' 'RSTD_EP_Mean,' 'RSTD_EP_Huber,' 'RSTD_EP_L2').

The first step of Normalization consists of a patient-control normalization (labeled currently in the code as 'Step 2,' with a previous source transformation step under construction).

The code first stratifies the subjects per nationality/site (e.g., Argentina and Chile) and condition (Controls and remaining subjects). The code immediately stratifies control subjects as those being labeled as 'CN' in the participants.txt file. If no 'CN' files are found, a message is issued on the command window in which users can specify the words denoting their control subjects.

Users may also give said value as an optional parameter of '*ControlLabel*,' signaling it as a String. Otherwise, if no warning messages are issued, the code proceeds to compute normalization tests for a given nationality, based on subjects' condition. Thus, .set files corresponding to each subject are loaded with the *pop_loadset* function, as seen on the command Window. Once all control subjects of one nationality are loaded, the remaining subjects are then loaded.

Note
A warning message is issued on the command window when subjects do not have a minimum duration of 240 s (4 min)—This can be specified at the '*minDurations*' optional parameter, being considered too short for normalization. As such, those files are not considered normalization for not having the minimum number of points required. The user can also define its own minimum duration required as an optional parameter.

Once all subjects have been picked up and the subjects that did not comply with the minimum required of time points are removed, normalization tests are computed for each channel. This exact process is then repeated for subjects of the remaining nationalities until all nationalities in the given data set are analyzed. The total number of files that have undergone normalization is displayed on screen, excluding the discarded files that had a short time period.

6.5 Source Transformation

Optional Parameters for Source Transformation:

- '*BIDSmodality*': Users can input a string of the modality of the data that will be analyzed. It is set as 'eeg' as default.
- '*BIDStask*': Users can input a string for the type of task to be analyzed. It is 'rs' by default.
- '*newPath*': Users can input a string for the path in which the new folders will be stored at. It is set at 'databasePath/analysis_RS' by default.
- '*selectSourceTime*': Users can input a time window to transform to a source level, by inputting it as a vector (e.g., [1, 2]). It is empty by default.

- '*avgSourceTime*': Users can signal if they want to average the selected time window by inputting it as a Boolean ('1'). It is set as '0' by default.
- '*sourceTransfMethod*': Users can indicate the method they want to use to calculate a source transformation, by inputting it a string (i.e: 'BMA', 'FT_MNE,' or 'FT_eLORETA'). It is set as 'BMA' as default.
- '*sourceROIatlas*': Users can indicate the atlas they wish to use, by inputting it as a string. It is set at 'AAL-116' by default.

The Source Transformation Stage performs a source transformation from electrodes to source level using an average brain.

Users may opt between different methods to calculate source transformation (by inserting it at the optional parameter of '*sourceTransfMethod*'): Bayesian Model Averaging (BMA) or otherwise the eLORETA or Minimum Norm Estimate (MNE) methods, both computed via the FieldTrip toolbox.

Note
We strongly recommend running the BMA method for HEP or task signals, considering its potential to account for the uncertainty of the other methods in tackling the inverse solution to source analysis, as well as counting with greater topographic power. The method can however, be costly for resting state signals, taking time and space tolls to compute each file, as well as consuming a big amount of space (around 600Gb for 100 files- *see* Table 1). The eLoreta and MNE methods are considered less effective than BMA, but they do solve time and space constraints for resting state signals, and we urge to use them when performing analysis based on these methods.

6.5.1 Optional Time Selection and Averaging

This step is optional and identical for all three methods. Users may opt for a specific time window (in seconds) to compute analysis, simply by specifying it into the '*selectSourceTime*' optional parameter. Additionally, the user can average the data within that time window (or the whole record if no time window is given), using the key 'avgSourceTime' parameter.

We hereby proceed to describe each possible method: BMA, eLORETA, and MNE taking into account a single time point as an illustrative example. The methods utilizing Fieldtrip (eLORETA and MNE) are grouped together as user experience for both is identical.

BMA Method

Optional Parameters for the BMA Method:

- '*BMA_MCwarming*': Users can input the warming length of the Markov Chain, by inputting it as an integer (i.e: '3000'). It is set at 4000 as default.
- '*BMA_MCsamples*': Users can input the number of samples from the Monte Carlo Markov CHain sampler for source transformation, by inputting it as an integer (i.e: '2000'). It is set at 3000 by default.
- '*BMA_MET*': Users can indicate a method of preference for exploring the models' space by inputting it as a stringer. If users signal 'If MET == 'OW,' the Occam's Window algorithm is used. If users signal 'If MET == 'MC,' the MC3 method is used (this is set as default).
- '*BMA_OWL*': Users can indicate an integer for the Occam's window lower bounds. '3' indicated a very strong window, '20' is a strong window, '150' is a positive window, and '200' is a weak window. It is set at 3 as default.

Step 1: Transforming Channels to Source (BMA)

A transformation of channels to source is first computed. A warning message is displayed on the command window for resting state signals, whereby users are recommended to run the Field Trip method instead of the BMA method, in order to solve time and space constraints. Users may then press the 'y' key to switch to the eLORETA method or otherwise press any other key to continue with BMA instead, whereby the .set files from step 0 with the pop_loadset function are loaded.

For BMA, the field matrix (Ke) and the Laplacian matrix (Le) are loaded, depending on the number of channels. As a note, the code currently only supports files with 128 and 64 Biosemi channel layouts. The function that performs transformation from electrodes to source level is then executed. The function creates a . txt file with source points and time, data is then reshaped according to those values. An EEG-like structure is then created, where the results of the source transformation will be stored. The results are stored in a .mat file (and an image showing the Markov ChainEvolution is also stored.

Step 2: Averaging Source to ROI (BMA).

Step 2 consists of an averaging of the source level by regions defined by ROI, by means of the AAL-116 atlas. First, .mat files with information from the previous step are loaded into the workspace, the atlas is loaded and labels for each source point are created by region. The step is entirely automatic and does not require any user input, while users can check the command window for any changes. As such, the .mat files from the previous step are loaded into the workspace, and each source point is labeled by region, and the ROInames are defined.

FieldTrip Methods: MNE and eLORETA

Optional Parameters:

- '*FT_sourcePoints*': Users can indicate the desired number of source points by indicating it as an integer (5124 or 8196). 5124 is defined as default, as it takes less memory.
- '*FT_plotTimePoints*': Users can indicate if they wish to plot anything at the source level on determined time points. It can be an integer with a single time point in seconds to be visualized (e.g., [5]) or a vector with a time window (e.g., [1.5]) that will be averaged and visualized.

The code is computed in the same fashion for both FieldTrip methods. The only difference relies on how they both use their respective methods (MNE or eLORETA). User experience and general code execution are identical and thus presented together.

Step 1: Transforming Channels to Source (FieldTrip)

Step 1 of source transformation consists of an automatic transformation of channels to source. First, it picks up the files from the previous step with the *pop_loadset* function. The code then checks for coordinates of the electrodes in an .xyz file for a Biosemi of 128 channels. The code will thus look for said file in the given dataset or otherwise create it if it doesn't yet exist. The code then performs surface source estimation. As such, it first transforms the EEG structured data into Fieldtrip-readable data. It computes a small preprocessing of data as a test case.

The code then defines the data that will enter the source estimation by means of covariance of trials, in case data has multiple trials. In a single-trial case, it doesn't compute anything and just assigns the data. Source transformation is then performed according to the number of source points in the given file. Source components are thus determined and electrodes are projected on the brain surface. The results for each time point are then saved in a .txt file, and also in a .mat file.

Step 2: Averaging Source by ROI (FieldTrip)

Step 2 of source transformation consists of an averaging of the source by ROI. It is entirely automatic and computes an averaging source by ROI using the 'AAL-116' atlas by default.

Users may otherwise indicate another atlas by inserting it as an optional parameter in the 'SourceROIatlas' variable, as a key value. The regions of the ROI atlas are loaded, while the 82 labels corresponding to cortical regions are looked upon the atlas. The new ROI data is then saved, taking into account times and names. As an output, the average ROI data and names of regions are stored in .txt files.

Final Steps

Source Transformation is completed once Step 2 is finished. The command window will issue the number of subjects run at this point for every step of source transformation, thus users are able to check any errors or subjects that were not computed. The total number of subjects after source transformation are then indicated, and users can press any key to continue onto the next step, or otherwise press 'q' to exit.

Connectivity Metrics

Optional Parameters for Connectivity Metrics:

- '*BIDSmodality*': Users can input a string of the modality of the data that will be analyzed. It is set as 'eeg' as default.
- '*BIDStask*': Users can input a string for the type of task to be analyzed. It is 'rs' by default.
- '*newPath*': Users can input a string for the path in which the new folders will be stored at. It is set at 'databasePath/analysis_RS' by default.
- '*runConnectivity*': Users can indicate whether they wish to skip this step by signaling it as 'false,' while it is set as 'true' as default.
- '*connIgnoreWSM*': Users can indicate whether they want to run or ignore the Weighted Symbolic Metrics (WSM) by signaling it as 'true.' It is set as 'false' as default.

The step is entirely automatic, with no inputs needed from users.

The code will first look for the .mat files containing the source averaged by ROI from the previous step and load them into the environment. Otherwise, if the code is loading files from a step previous to source transformation, it will automatically look and load the last run .set files.

Four connectivity metrics are calculated by default (with an option to expand to 7 if the user wants to calculate computationally expensive metrics traditionally used in fMRI—Weighted Symbolic Metrics. Users can signal to include the WSM or ignore it by inserting it at the optional parameter of '*connIgnoreWSM*'). The calculation of connectivity metrics takes just a few seconds, after which users are told the analysis is finished in the command window, and the same process repeats for the following files.

6.6 Classifier

Optional Parameters for Connectivity Metrics:

- '*BIDSmodality*': Users can input a string of the modality of the data that will be analyzed. It is set as 'eeg' as default.
- '*BIDStask*': Users can input a string for the type of task to be analyzed. It is 'rs' by default.

- '*newPath*': Users can input a string for the path in which the new folders will be stored at. It is set at 'databasePath/analysis_RS' by default.
- '*runClassifier*': Users can indicate whether they want to run this step by signaling 'true' or otherwise ignore it by signaling it 'false.' It is set as 'true' as default.
- '*runFeatureSelection*': Users can opt to run a feature selection with FDR correction prior to creating the model, by signaling it as 'true.'
- '*classDXcomparison*': Users must indicate the diagnostics they wish to compare by signaling it as a cell of 2×1.
- '*classNumPermutations*': Users can indicate the number of permutations they desire for statistical tests, by inserting it as an integer. It is set at 5000 by default.
- '*classSignificance*': Users can indicate a desired level of significance by inserting it as an integer. It is set at 0.05 as default.
- '*classCrossValsFolds*': Users can indicate the number of cross-validation folds to use for the ROC curves, by setting it as an integer. It is set as 5 as default.

The code is automatic. Before running the classifier, a feature selection of the desired diagnostics to compare, is executed. The feature selection is done on these diagnoses based upon permutations and false discovery rate (FDR) correction. As such, the code will look for possible diagnoses in the dataset and show them on screen for users to signal the diagnoses that they want to compare. For example, users may want to compare a disease group ('FTD') against controls ('CN'). Users may indicate so by simply typing each diagnosis, individually, in the command window. A message is then issued on the command window, in which users are notified of the analysis that will take place, indicating each condition and the number of subjects present in each condition. Users can also indicate the diagnoses to compare by inserting them in the optional parameter of '*classDXcomparison*.'

Once the feature selection is performed, and the desired diagnostics are saved in a .csv file, the classifier is trained. The classifier is implemented in Python, and called from Matlab. In the current version, a XGBoost model is created using the XGBoost Python package. Additionally, feature importance is determined using algorithms such as SequentialFeatureSelector from the *mlxtend* Python package, and SHapley Additive exPlanations (SHAP) from the SHAP Python package. Additionally, ROC curves are created using the sklearn Python package.

This step is potentially the only one that might require programming knowledge from the user.

Note
The code will designate 80% of files in a given condition as the training set and 20% of files as the testing set. For that reason, users are required to have at least 25 subject files per condition in order to perform cross-validation. If fewer than 25 subjects per diagnostic condition are given, users are instructed to change the train/test split and the number of folders in the . Ipynb python code if they want to continue (as the code is constructed in Python and called from Matlab). For example, if the user had 46 control files and 16 files of a diagnostic condition (FTD), the user would be warned and advised to change the criteria of cross-validation training and testing split on the Python code. Here, the user can manually change said parameters on the Python code to 70% (training) and 30% (testing) to allow the analysis to proceed. Users are then indicated to press the 'y' key in order to continue.

The classifier is then executed. First, a machine learning algorithm is run, which differentiates between conditions/diagnosis (e.g., FTD vs. Control). Note, it usually takes quite some time to finish execution, lasting between 5 days to 1 week for a hundred features if run on a standard desktop computer. The execution is entirely automatic: The code will first look for a .csv file in which the conditions to run are stored (e.g., FTD and control), with warning messages being issued in the command window for the user if no .csv files are found or multiple .csv files are located. After finishing, users will be able to see the number of subjects run.

As an output from the classifier, users will be able to see three figures, which will be readily displayed and stored on the directory tree under the steps folder ('classification'), denoting the sequential forward selection, the ROC curve with the most important features and the graphical display of features importance (see Subheading 3.2 for the interpretation and significance of the outputs).

References

1. Jalilianhasanpour R, Beheshtian E, Sherbaf G, Sahraian S, Sair HI (2019) Functional connectivity in neurodegenerative disorders: Alzheimer's disease and frontotemporal dementia. Top Magn Reson Imaging 28:317–324. https://doi.org/10.1097/RMR.0000000000000223
2. Moguilner S, García AM, Perl YS, Tagliazucchi E, Piguet O, Kumfor F, Reyes P, Matallana D, Sedeño L, Ibáñez A (2021) Dynamic brain fluctuations outperform connectivity measures and mirror pathophysiological profiles across dementia subtypes: a multicenter study. NeuroImage 225:117522. https://doi.org/10.1016/j.neuroimage.2020.117522
3. Babiloni C, Arakaki X, Azami H, Bennys K, Blinowska K, Bonanni L, Bujan A, Carrillo MC, Cichocki A, de Frutos-Lucas J, del Percio C, Dubois B, Edelmayer R, Egan G, Epelbaum S, Escudero J, Evans A, Farina F, Fargo K, Fernández A, Ferri R, Frisoni G, Hampel H, Harrington MG, Jelic V, Jeong J, Jiang Y, Kaminski M, Kavcic V, Kilborn K,

Kumar S, Lam A, Lim L, Lizio R, Lopez D, Lopez S, Lucey B, Maestú F, McGeown WJ, McKeith I, Moretti DV, Nobili F, Noce G, Olichney J, Onofrj M, Osorio R, Parra-Rodriguez M, Rajji T, Ritter P, Soricelli A, Stocchi F, Tarnanas I, Taylor J-P, Teipel S, Tucci F, Valdes-Sosa M, Valdes-Sosa P, Weiergräber M, Yener G, Guntekin B (2021) Measures of resting state EEG rhythms for clinical trials in Alzheimer's disease: recommendations of an expert panel. Alzheimers Dement 17:1528–1553. https://doi.org/10.1002/alz.12311

4. Farzan F, Atluri S, Frehlich M, Dhami P, Kleffner K, Price R, Lam RW, Frey BN, Milev R, Ravindran A, McAndrews MP, Wong W, Blumberger D, Daskalakis ZJ, Vila-Rodriguez F, Alonso E, Brenner CA, Liotti M, Dharsee M, Arnott SR, Evans KR, Rotzinger S, Kennedy SH (2017) Standardization of electroencephalography for multi-site, multi-platform and multi-investigator studies: insights from the canadian biomarker integration network in depression. Sci Rep 7:7473. https://doi.org/10.1038/s41598-017-07613-x
5. Prado P, Birba A, Cruzat J, Santamaría-García H, Parra M, Moguilner S, Tagliazucchi E, Ibáñez A (2022) Dementia ConnEEGtome: towards multicentric harmonization of EEG connectivity in neurodegeneration. Int J Psychophysiol 172:24–38. https://doi.org/10.1016/j.ijpsycho.2021.12.008
6. Alam R-U, Zhao H, Goodwin A, Kavehei O, McEwan A (2020) Differences in power spectral densities and phase quantities due to processing of EEG signals. Sensors 20:6285. https://doi.org/10.3390/s20216285
7. Prado-Gutierrez P, Martínez-Montes E, Weinstein A, Zañartu M (2019) Estimation of auditory steady-state responses based on the averaging of independent EEG epochs. PloS One 14:e0206018. https://doi.org/10.1371/journal.pone.0206018
8. Cohen MX (2015) Comparison of different spatial transformations applied to EEG data: a case study of error processing. Int J Psychophysiol 97:245–257. https://doi.org/10.1016/j.ijpsycho.2014.09.013
9. Pernet CR, Appelhoff S, Gorgolewski KJ, Flandin G, Phillips C, Delorme A, Oostenveld R (2019) EEG-BIDS, an extension to the brain imaging data structure for electroencephalography. Sci Data 6:103. https://doi.org/10.1038/s41597-019-0104-8
10. Dong L, Li F, Liu Q, Wen X, Lai Y, Xu P, Yao D (2017) MATLAB toolboxes for reference electrode standardization technique (REST) of scalp EEG. Front Neurosci 11:601. https://doi.org/10.3389/fnins.2017.00601
11. Pion-Tonachini L, Kreutz-Delgado K, Makeig S (2019) ICLabel: an automated electroencephalographic independent component classifier, dataset, and website. NeuroImage 198:181–197. https://doi.org/10.1016/j.neuroimage.2019.05.026
12. Bigdely-Shamlo N, Kreutz-Delgado K, Kothe C, Makeig S (2013) EyeCatch: data-mining over half a million EEG independent components to construct a fully-automated eye-component detector. Annu Int Conf IEEE Eng Med Biol Soc 2013:5845–5848. https://doi.org/10.1109/EMBC.2013.6610881
13. Kleifges K, Bigdely-Shamlo N, Kerick SE, Robbins KA (2017) BLINKER: automated extraction of ocular indices from EEG enabling large-scale analysis. Front Neurosci 11:12. https://doi.org/10.3389/fnins.2017.00012
14. Trujillo-Barreto NJ, Aubert-Vázquez E, Valdés-Sosa PA (2004) Bayesian model averaging in EEG/MEG imaging. NeuroImage 21:1300–1319. https://doi.org/10.1016/j.neuroimage.2003.11.008
15. Pascual-Marqui RD, Lehmann D, Koukkou M, Kochi K, Anderer P, Saletu B, Tanaka H, Hirata K, John ER, Prichep L, Biscay-Lirio R, Kinoshita T (2011) Assessing interactions in the brain with exact low-resolution electromagnetic tomography. Philos Transact A Math Phys Eng Sci 369:3768–3784. https://doi.org/10.1098/rsta.2011.0081
16. Hämäläinen MS, Ilmoniemi RJ (1994) Interpreting magnetic fields of the brain: minimum norm estimates. Med Biol Eng Comput 32:35–42. https://doi.org/10.1007/BF02512476
17. Rolls ET, Joliot M, Tzourio-Mazoyer N (2015) Implementation of a new parcellation of the orbitofrontal cortex in the automated anatomical labeling atlas. NeuroImage 122:1–5. https://doi.org/10.1016/j.neuroimage.2015.07.075
18. Manly BFJM, Bryan FJ (2017) Randomization, bootstrap and monte carlo methods in biology, 3rd edn. Chapman and Hall/CRC, New York
19. Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. J R Stat Soc Ser B Methodol 57:289–300. https://doi.org/10.1111/j.2517-6161.1995.tb02031.x
20. Kaufmann T, van der Meer D, Doan NT, Schwarz E, Lund MJ, Agartz I, Alnæs D, Barch DM, Baur-Streubel R, Bertolino A,

Bettella F, Beyer MK, Bøen E, Borgwardt S, Brandt CL, Buitelaar J, Celius EG, Cervenka S, Conzelmann A, Córdova-Palomera A, Dale AM, de Quervain DJF, Di Carlo P, Djurovic S, Dørum ES, Eisenacher S, Elvsåshagen T, Espeseth T, Fatouros-Bergman H, Flyckt L, Franke B, Frei O, Haatveit B, Håberg AK, Harbo HF, Hartman CA, Heslenfeld D, Hoekstra PJ, Høgestøl EA, Jernigan TL, Jonassen R, Jönsson EG, Karolinska Schizophrenia Project (KaSP), Kirsch P, Kłoszewska I, Kolskår KK, Landrø NI, Le Hellard S, Lesch K-P, Lovestone S, Lundervold A, Lundervold AJ, Maglanoc LA, Malt UF, Mecocci P, Melle I, Meyer-Lindenberg A, Moberget T, Norbom LB, Nordvik JE, Nyberg L, Oosterlaan J, Papalino M, Papassotiropoulos A, Pauli P, Pergola G, Persson K, Richard G, Rokicki J, Sanders A-M, Selbæk G, Shadrin AA, Smeland OB, Soininen H, Sowa P, Steen VM, Tsolaki M, Ulrichsen KM, Vellas B, Wang L, Westman E, Ziegler GC, Zink M, Andreassen OA, Westlye LT (2019) Common brain disorders are associated with heritable patterns of apparent aging of the brain. Nat Neurosci 22: 1617–1623. https://doi.org/10.1038/s41593-019-0471-7

21. Torlay L, Perrone-Bertolotti M, Thomas E, Baciu M (2017) Machine learning-XGBoost analysis of language networks to classify patients with epilepsy. Brain Inform 4:159–169. https://doi.org/10.1007/s40708-017-0065-7
22. Rodríguez-Pérez R, Bajorath J (2020) Interpretation of machine learning models using shapley values: application to compound potency and multi-target activity predictions. J Comput Aided Mol Des 34:1013–1026. https://doi.org/10.1007/s10822-020-00314-0
23. Donnelly-Kehoe PA, Pascariello GO, Gómez JC, Alzheimers Disease Neuroimaging Initiative (2018) Looking for Alzheimer's disease morphometric signatures using machine learning techniques. J Neurosci Methods 302: 24–34. https://doi.org/10.1016/j.jneumeth.2017.11.013
24. Kingsford C, Salzberg SL (2008) What are decision trees? Nat Biotechnol 26:1011–1013. https://doi.org/10.1038/nbt0908-1011
25. Poldrack RA, Baker CI, Durnez J, Gorgolewski KJ, Matthews PM, Munafò MR, Nichols TE, Poline J-B, Vul E, Yarkoni T (2017) Scanning the horizon: towards transparent and reproducible neuroimaging research. Nat Rev Neurosci 18:115–126. https://doi.org/10.1038/nrn.2016.167
26. Cassani R, Estarellas M, San-Martin R, Fraga FJ, Falk TH (2018) Systematic review on resting-state EEG for Alzheimer's disease diagnosis and progression assessment. Dis Markers 2018:5174815. https://doi.org/10.1155/2018/5174815
27. Gonzalez-Moreira E, Paz-Linares D, Areces-Gonzalez A, Wang R, Bosch-Bayard J, Bringas-Vega ML, Valdes-Sosa PA (2019) Caulking the leakage effect in MEEG source connectivity analysis. arXiv preprint arXiv:1810.00786
28. Prado P, Mejía JA, Sainz-Ballesteros A, Birba A, Moguilner S, Herzog R, Otero M, Cuadros J, Z-Rivera L, O'Byrne DF, Parra M, Ibáñez A (2023) Harmonized multi-metric and multicentric assessment of EEG source space connectivity for dementia characterization. Alzheimers Dement Diagn Assess Dis Monit 15: e12455. https://doi.org/10.1002/dad2.12455