Neurocomputing 519 (2023) 127–134

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Original software publication

EEGraph: An open-source Python library for modeling electroencephalograms using graphs

Ana M. Maitin, Alberto Nogales, Pedro Chazarra, Álvaro José García-Tejedor *

CEIEC Research Institute, Universidad Francisco de Vitoria, Ctra.M-515 Pozuelo-Majadahonda Km.1, 800, Pozuelo de Alarcón, 28223 Madrid, Spain

ARTICLE INFO

Article history:
Received 21 April 2022
Revised 10 November 2022
Accepted 15 November 2022
Available online 21 November 2022

Communicated by Zidong Wang

Keywords:
Connectivity
Graph
Electroencephalogram (EEG)
Modeling
Brain
Open source Python library

ABSTRACT

Background and objective: Connectivity studies make it possible to identify alterations in brain connections and to associate these pathologies with different neurological disorders. However, a clinical test is necessary to obtain information about the state of the brain. Electroencephalograms (EEGs) provide this information in addition to being tests with other benefits for the patient (non-invasive, low-cost, high reproducibility). Graph theory can be used to represent both the anatomical and functional connections of the brain by means of connectivity measures. The procedure of transforming an EEG into a graph can be slightly tedious for researchers, especially when implementing different connectivity measures.
Methods: The open-source Python library EEGraph automatically performs the modeling of an EEG through a graph, providing its matrix and visual representation. It recognizes various EEG input formats, identifying the number of electrodes and the location of each electrode in the brain. Moreover, it allows the user to choose from 12 connectivity measures to produce the graph from the EEG, with great flexibility to define specific parameters to adapt them to each study, including EEG time-windows segmentation and separation in frequency bands.
Results: The EEGraph library is developed as a tool, for researchers and clinical specialists in the field of neuroscience, that provides direct information on the connectivity of the brain from electroencephalography signals. Its documentation and source code are available at https://github.com/ufvceiec/EEGRAPH. It can be installed from the Python Package Index using pip install EEGRAPH.
Conclusions: The EEGraph library was built aiming to facilitate the development of connectivity studies based on the modeling of electroencephalography tests through graphs. It includes a wide range of connectivity measures, which, together with the multiple output options, make EEGraph an easy to use and powerful tool with direct applications in both the clinical and neuroscience research fields.

1. Introduction

Connectivity studies are of great importance within the field of neuroscience as they allow the evaluation of both the anatomical and functional organization of the brain. These studies are carried out by identifying and classifying the patterns produced by the connections between the different cerebral regions [1]. To obtain and represent precise information about the state of brain connections, a series of neurobiologically significant measures, sensitive to connectivity changes, are considered. Some of the most used measures can be found in [1]. The study of the connections (both anatomical and functional) of the different brain regions makes it possible to identify alterations in the connectivity patterns and to relate these alterations with anomalies or neuropsychiatric disorders [2–5]. To carry out this type of study, it is necessary to have both a clinical test that contains the information on the state of the brain and a methodology that effectively represents the connection between the brain regions.

Graph theory offers a great way to represent brain connectivity due to its potential to model networks and their connections. To do this, both the nodes and the edges of the graph must be defined according to the considered problem. [6] is a review of the different procedures to define the nodes of the graph according to the type of study that the user wants to carry out. One of the methodologies commonly used consists of relating the nodes of the graph with different regions of the brain, so that the edges determine the connections between them. Furthermore, the use of graph theory allows obtaining the information of the connections in two ways, visual and matrix. The visual representation of the graph can help clinical

* Corresponding author.
E-mail addresses: a.maitin@ceiec.es (A.M. Maitin), alberto.nogales@ceiec.es (A. Nogales), pedrochgb@gmail.com (P. Chazarra), a.gtejedor@ceiec.es (Á.J García-Tejedor).

https://doi.org/10.1016/j.neucom.2022.11.050
0925-2312/

staff in their daily practice, whereas its matrix representation provides information to the researchers as data structures with which to work more comfortably and efficiently in their investigations without losing the information contained in the graph [7]. However, to apply graph theory it becomes necessary to perform a clinical test that extracts the information of the patient's brain.

Electroencephalography is a clinical test that is responsible for recording the electric field that is produced by the activity of pyramidal neurons in the brain, providing a record of the current state of the brain [8]. An electroencephalogram (EEG) is made up of a set of signals, each one associated with each of the scalp electrodes contained in the electroencephalography headset. Therefore, each signal is a time series representing a specific location in the patient's brain. Among the benefits of this test, it may be highlighted that it is non-invasive for the patient, has a high temporal resolution, and is low-cost compared to other types of techniques commonly used in the field of neurology, such as single photon emission computed tomography (SPECT), M-iodobenzylguanidine cardiac scintiscan (MIBG), Magnetic Resonance Imaging (MRI), Computed Tomography (CT scan) [9]. However, the use of this test is not widespread in routine clinical practice due to the complexity of extracting information from these signals through visual analysis, especially in those pathologies that do not present abrupt changes in the state of the patient and thus do not show visual disturbances on EEG [10].

Although graph theory has been previously considered in connectivity studies, the process of constructing graphs from different connectivity measures is still intricate and it is not extended in the clinical practice. Furthermore, tests used to obtain information from the brain are often expensive and invasive for the patient. Even though EEGs are harmless, they are difficult to analyze, and it has not been until recent years that their use in connectivity analysis has grown [11]. These problems reduce the amount of data available, limiting this type of studies and affecting current research. In this work, we present EEGraph, an open-source Python library that provides an EEG representation by means of a graph. It has been built with the main aim of providing a tool that facilites the development of connectivity studies. In order to evaluate connectivity, EEGraph has 12 of the most used measurements in the bibliography [12–16], which allows addressing studies related to both the frequency spectrum and the time domain of the EEG. This wide variety of measures chosen to evaluate connectivity, together with the flexibility offered by the library to define the parameters involved, provide the user with a powerful tool that facilitates the investigation in the area of neuroscience. The output can be obtained as a visual representation, through an image that facilitates a rapid clinical application, and as a matrix representation, through easily manipulable data structures for quantitative analysis. Both representations provide a possible application in the clinical field and a useful tool for neuroscience research field.

2. Methods

The EEGraph library (whose code can be found in GitHub[1]) is a tool that offers the user a fast and efficient way to carry out different connectivity studies from EEGs. It uses graph theory to represent brain connections, identifying the nodes of the graph with the electrodes contained in the electroencephalography headset, preserving their spatial distribution. The connections between the nodes, that is, the graph edges, are determined by the different connectivity measures available in the library. In this section, we make a brief review of the libraries that deal with graphs and EEGs, and we summarize how the library works and how these concepts are defined.

2.1. State of the art

There are Python libraries that either focus on EEG processing, or are dedicated to graph handling, but, to the best of our knowledge, no library exists covering both aspects, i.e., modeling EEG as graphs. This fact highlights the need for a user-friendly and versatile tool that connects both approaches. A brief overview of some of these two groups of Python libraries is provided below.

Within the libraries that carry out EEG processing, *eeglib* [17] is a tool dedicated to EEG feature extraction (through 8 methods), besides including other procedures for signal analysis. However, it does not provide any visual representation of the output. The package *Mne* [18] is designed for the exploration, visualization and analysis of neurophysiological data, EEG included. In particular, it supports 6 connectivity measures and their variations. Although this library allows a great flexibility for the choice of certain parameters, it is rather complex and not very intuitive, and it does not build the associated graph. Finally, *SCoT* [19] is a package dedicated to source decomposition and connectivity estimation in EEG from spectral measures such as Coherence, Partial Directed Coherence, Directed Transfer Function and variations.

Within the Python libraries focused on graph handling, *NetworkX* [20] allows the creation, manipulation, analysis and visualization of graphs, digraphs, and multigraphs. The python libraries *spektral* [21] and *StellarGraph*[2] handle graph-like data-structures for their use in deep learning and machine learning models, respectively.

Hence, it can be appreciated how EEGraph covers the existing gap in EEG modeling through graphs by serving as a link between these two fields. Moreover, it allows a large number of different connectivity measures as well as great flexibility for the parameters in a intuitive environment for the user.

2.2. Nodes construction

Withing the input options, EEGraph admits various EEG formats, including those associated with the output formats of different electroencephalography devices and pre-processing tools. Once the file containing the EEG is read by the library, EEGraph automatically identifies the number of electrodes that it contains, labels each of the signals with the node that corresponds to it, and places each node in the appropriate position with respect to the EEG headset to represent the graph. Building the graph takes into account that smaller mounting systems are contained in larger mounting structures [22]. For this reason, a dictionary with 333-electrode positions has been built and a mesh has been generated where these electrodes with their corresponding labels have been placed in order to later identify the positions of the graph nodes. It has been considered that there are electrodes that can be referred to with more than one label, so the different labeling options have also been included ($T3 = T7, T4 = T8, T5 = P7, T6 = P6$). The result at this point is a graph without edges, which depends on the connectivity measure chosen by the user.

2.3. Edges construction

To select the measures implemented in the library, a bibliographic study was carried out to determine which were the most used ones [12–16]. As a result, EEGraph implements a total of 12 connectivity measures, which are detailed in Section 3. Within these measures, *Power Spectrum*, *Spectral Entropy* and *Shannon Entropy* extract information from a single channel of EEG, while the remaining ones provide information about the relationship between pairs of electrodes. Measurements relating pairs of electrodes provide

[1] https://github.com/ufvceiec/EEGRAPH.

[2] https://stellargraph.readthedocs.io/en/stable.

an immediate relationship between the nodes of the graph. In this way, a fully connected graph is generated where each edge has an associated weight corresponding to the chosen connectivity measure. To facilitate the extraction of patterns, we have introduced a threshold (which can be modified by the user) to represent only those values that exceed this limit. The default value has been defined based on the previous bibliography. In the case of measures that provide information about single electrodes, it becomes necessary to define a criterion that relates the nodes of the graph to establish the connections. The chosen criterion consists in fully connecting a fixed percentage of the nodes with the higher values. The default value for such percentage is 25%, although the user can modify it. In this case, since there is no measure associated with the edges of the resulting graph, their associated weights are fixed to 1.00 to indicate the existence of connections.

Electroencephalography studies can be divided mainly into two groups: temporal domain analysis and frequency domain analysis. Frequency analysis is carried out over different intervals or bands, which are related to various brain functions to study specific characteristics of each of them. The most representative bands are: delta (1–4 Hz), theta (4–8 Hz), alpha (8–13 Hz), beta (13–30 Hz) and gamma (30–45 Hz). The variety of connectivity measures implemented in EEGraph allow the user to carry out both types of studies. Those related to the temporal domain are *Pearson Correlation*, *Cross-Correlation*, *Corrected Cross-Correlation*, *Phase Lag Index* and *Shannon Entropy*, while the measures related to the frequency domain are *Coherence*, *Imaginary Coherence*, *Phase Locking Value*, *Phase Lag Index*, *Weighted Phase Lag Index*, *Directed Transfer Function*, *Power Spectrum*, and *Spectral Entropy*. For these measures, EEGraph allows the users to choose the band or bands with which they want to work. Moreover, some studies divide the EEG into several segments or time windows to analyze them individually or concatenate them to have a temporal sequence. Hence, the division in windows has also been taken into consideration in EEGraph and can be modified by the user.

2.4. Data visualization

The output of EEGraph is two fold. On the one hand, it provides the visual representation of the graph, i.e., the image of the graph with the nodes and the connections between them, which depends on the connectivity measure chosen. Within this representation, there are two possibilities. In case the connectivity measure provides a relationship between two nodes, the value of the measure can be visualized approximating the cursor to the middle point of each edge. Moreover, the width of each line depends on the associated value. In case the connectivity measure provides information about single nodes, the values obtained are represented in the nodes, and the edges represented show a value of 1.00 when the cursor is approximated to their middle point, indicating that a connection exists. This representation is designed to facilitate the acquisition of information directly. On the other hand, EEGraph provides a matrix representation of the graph, which contains all the values of the connectivity measure chosen by the user. If the EEG has been divided into several windows, EEGraph provides both the visual and matrix representation of the graph for each of them, allowing the user to evaluate the temporal evolution of the brain connections.

3. Connectivity measures theory

This section details each of the connectivity measures implemented, providing their mathematical definition, and a brief interpretation of their application to time series, as it is the case of the EEG channels. The interval to which each measure belongs is also specified, along with the meaning of the possible results, in order to facilitate their implementation. In addition, the default threshold associated to each measure (based on bibliographic values) is indicated, although it should be noted that these values can be modified by the user.

3.1. Pearson correlation

Pearson correlation coefficient (C) [23] measures the linear correlation between two variables, two sets of data, or two time series. It is defined as:

$$C_{x,y} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}, \tag{1}$$

where x and y are the two time series, σ_{xy} stands for their covariance, σ_x is the standard deviation of x, and σ_y is the standard deviation of y. Eq. (1) provides a scalar in the range $[-1, 1]$, where 1 means a perfect correlation, -1 represents a perfect inverse correlation, whereas 0 means that there is no correlation between x and y. The function *pearsonr* within the module *scipy.stats* has been used for the computation of this value. The default threshold for this measure is 0.7.

3.2. Cross-correlation

Cross-correlation (CC) measures the similarity of two time series as a function of the relative displacement (or lag) of one of them with respect to the other. Its value is obtained through:

$$CC_{xy}(m) = \frac{R_{xy}(m)}{\sqrt{R_{xx}(0)R_{yy}(0)}}, \tag{2}$$

with

$$R_{xy}(m) := \sum_{j=0}^{N-1} (x_j - \mu_x)(y_{j-m} - \mu_y)^*$$

where x and y are the two time series, $*$ indicates the conjugate, μ_x and μ_y are the respective means of x and y, N denotes both the length of x and y, which match each other since these time series belong to the same EEG, and $y_{j-m} := 0$ if $j - m < 0$ or $j - m \geqslant N$. Here $m \in \mathbb{Z}$ and $R_{xy}(m) = 0$ if $m \leqslant -N$ or $m \geqslant N$. The computation of the value of R_{xx}, R_{yy} and R_{xy} has been carried out using the *correlate* function within the *scipy.signal* module. It should be noted that the definition for the cross-correlation introduced in Eq. (2) is normalized with respect to the zero-lag value so that it corresponds to the Pearson correlation coefficient. Thus, for small lags its value lies in a neighborhood of the interval $[-1, 1]$. To obtain a more representative value associated to the CC, the mean of the 10% of the positive lag values is calculated for each pair of time series, and a threshold of 0.5 is applied to these measures.

3.3. Corrected Cross-Correlation

The Corrected Cross-Correlation (CCC) measures the symmetry of the cross-correlation of two time series, x and y, with respect to the lag (that is, the displacement of y with respect x). Its expression is given by:

$$CCC_{xy}(m) = CC_{xy}(m) - CC_{xy}(-m), \tag{3}$$

where m represents the time lag between x and y, and CC_{xy} is their cross-correlation as established above. As a consequence, for the computations in Eq. (3), the function *correlate* within the *scipy.signal* module is used. Much like in the CC, to obtain a more representative value associated to the CCC, the mean of the 10% of the lag values is

calculated for each pair of time series, and a threshold of 0.1 is applied to these measures.

3.4. Squared Coherence

The Squared Coherence (SC) [24] measures the relationship between the spectrum of two time series, x and y, including, as a consequence, leading, lagged and smoothed relationships. This measure is defined as:

$$SC_{xy}(f) = \frac{|G_{xy}(f)|^2}{G_{xx}(f)G_{yy}(f)} \tag{4}$$

where $G_{xy}(f)$ is the cross spectral density between x and y, and $G_{xx}(f)$ and $G_{yy}(f)$ stand for the power spectral densities of x and y respectively. Eq. (4) provides a series of scalars in the range $[0, 1]$, where 1 represents a perfect correlation in the frequency spectrum, whereas 0 means that there is no correlation between the spectrum of both signals, x and y. The function *coherence* within the module *scipy.signal* is used for the computation of this value. Once it is obtained, its mean in each frequency band (delta, theta, alpha, beta, and gamma) is calculated. The default threshold for this measure is 0.65.

3.5. Imaginary Coherence

Much like the squared coherence, the Imaginary Coherence (IC) measures the relationship between the spectrum of two time series, x and y. However, it is less sensitive to certain external effects compared to the real part of coherence and so compared to SC. Its expression is given by:

$$IC_{xy}(f) = \frac{Imag(G_{xy}(f))}{\sqrt{G_{xx}(f)G_{yy}(f)}} \tag{5}$$

where $G_{xy}(f)$ is the cross spectral density between x and y, and $G_{xx}(f)$ and $G_{yy}(f)$ stand for the power spectral densities of x and y respectively. It should be noted that Eq. (5) provides a series of values in the range $[-1, 1]$, whose mean is then calculated in each frequency band (delta, theta, alpha, beta, and gamma). For the computations in Eq. (5), the functions *welch* and *csd* within the *scipy.signal* module, and the function *imag* within the *numpy* module are used. The default threshold for this measure is 0.4.

3.6. Phase Locking Value

The Phase Locking Value (PLV) [25] measures the fluctuations in the phase difference of two time series, x and y. Its value is obtained through:

$$PLV = |E[\exp(i\,\Delta\phi_{rel}(t))]| \tag{6}$$

where $\Delta\phi_{rel}(t)$ is the phase difference of x and y at time t, shifted to be in the interval $[0, 2\pi)$, $E[\cdot]$ stands for an average in times, and $|\cdot|$ means module, so that for each pair of channels a single real value is returned. According to the definition provided by Eq. (6), the PLV belongs to the interval $[0, 1]$, where 1 represents synchrony between the two time series (little fluctuations on the phase difference), and 0 the absence of synchrony (large fluctuations in the phase difference). Since the phase of the signals is obtained through the Hilbert transform, the *hilbert* function from the *scipy.signal* module, and the *angle* function within the *numpy* module are used. The default threshold for this measure is 0.8. It should be noted that the implementation of the PLV in this work provides the value associated to one or multiple bands that need to be specified.

3.7. Phase Lag Index

The Phase Lag Index (PLI) [26] measures the changes in the sign of the phase differences of two time series, x and y, and thus, the relative changes in the phase of one series with respect to the other. This measure is less sensitive to the influence of certain sources than PLV, and its value is given by the expression:

$$PLI = |E[sign(\Delta\phi(t))]| \tag{7}$$

where $\Delta\phi(t)$ is the phase difference of x and y at time t, shifted to be in the interval $[-\pi, \pi)$. The meaning of $E[\cdot], |\cdot|$ and the functions used to perform the computation of Eq. (7) are detailed in the PLV. The PLI value obtained in Eq. (7) belongs to the interval $[0, 1]$, where 1 represents a uniform relative position of the phase of x with respect to that of y, whereas 0 reveals an equal number of changes in the relative position of such phases. The default threshold for this measure is 0.1, and it can be called either for the whole time series with the option *pli*, or for one or multiple bands (that need to be specified) with the option *pli_bands*.

3.8. Weighted Phase Lag Index

The Weighted Phase Lag Index (WPLI) has a similar objective to the one of the PLI, though it is less sensitive to certain external effects compared to it. It is calculated through:

$$\begin{aligned} WPLI &= \frac{|E[Imag(G_{xy}(f))]|}{E[|Imag(G_{xy}(f))|]} \\ &= \frac{|E[|Imag(G_{xy}(f))| * sign(Imag(G_{xy}(f)))]|}{E[|Imag(G_{xy}(f))|]} \end{aligned} \tag{8}$$

where $G_{xy}(f)$ is the cross spectral density between x and y, $|\cdot|$ means module, and $E[\cdot]$ stands for an average in frequencies, so that for each pair of channels a single real value is returned. The value provided by Eq. (8) lies in the interval $[0, 1]$. The default threshold for this measure is 0.45, and, much like the PLV, the implementation of the WPLI in this work provides the value associated to one or multiple bands that need to be specified.

3.9. Directed Transfer Function

The Directed Transfer Function (DTF) [27] describes the causal influence of a channel *j*on channel *i*for each frequency through the expression:

$$DTF^2_{j\to i}(f) = \frac{|H_{ij}(f)|^2}{\sum_{m=1}^{k}|H_{im}(f)|^2} \tag{9}$$

where $H_{ij}(f)$is an element of a transfer matrix of a MVAR model (multivariate autoregressive model). In particular, Eq. (9) constitutes a normalized version of the *DTF*, which takes values from 0to 1, indicating a ratio between the inflow from channel j to channel i with respect to all the inflows to channel i. In this work, the computation of the DTF is performed through the *SCoT* python package, and the implementation of such a measure is associated to one or multiple bands that need to be specified. For a every band the mean value is considered in the frequency range associated to it. The default threshold for this measure is 0.3.

3.10. Power Spectrum

The Power Spectrum (PS) of a time series, x, describes its distribution in the frequency domain. It is calculated through:

$$PS(f) = |X(f)|^2 \tag{10}$$

where $X(f)$ is the one-dimensional discrete Fourier Transform of x, which is computed using the *rfft* function of the *numpy.fft* module.

The implementation of Eq. (10) allows the user to determine the PS for one or multiple bands (that need to be specified), and it returns the mean value of the PS in the range of frequencies associated to each band. In order to establish a relation between pairs of electrodes, 25% of the channels with the highest values are selected and connected in the final graph.

3.11. Shannon Entropy

The Entropy (H) or Shannon Entropy [28] measures the average amount of information contained in a time series x. Its expression is given by:

$$H(x) = -\sum_i p(x_i)\log\left(p(x_i)\right) \tag{11}$$

where *log* denotes the natural logarithm, and $p(x_i)$ is the unnormalized probability of the event x_i in x. For the computations in (11), the function *entropy* within the *scipy.stats* module is used. It should be noted that the value obtained in Eq. (11) is not normalized, and so it depends on the total number of occurrences, N. If H is normalized by the factor $\log(N)$, then the resulting value belongs to the interval $[0, 1]$. The state of maximum entropy, i.e., normalized entropy equal to 1, is obtained in equiprobable events and means minimum information, whereas minimum entropy, i.e., $H(x) \simeq 0$ represents maximum amount of information. Similar to the measure PS, 25% of the channels with the highest values are selected and connected in the final graph. Thus, its edges do not get influenced by the normalization.

3.12. Spectral Entropy

The Spectral Entropy (SE) is the Shannon entropy in the frequency domain of a time series x. Hence, SE measures the amount of information contained in its spectrum. Its expression is given by:

$$SE(x) = \frac{-\sum_{f=0}^{f_s/2} p(f)\log_2(p(f))}{\log_2(\text{psd\_size})} \tag{12}$$

where log_2 denotes the logarithm in base 2, f_s is the sampling frequency, $p(f)$ stands for the normalized power spectrum density of x estimated through the Welch method, and *psd_size* is the size of the power spectral density of x. Unlike the Shannon entropy measure, the value obtained through Eq. (12) is normalized to be in the interval $[0, 1]$. The complete computations are carried out using the function *spectral_entropy* of the python package *AntroPy*. Once a single value for the SE is obtained for every channel, 25% of the channels with the highest values are selected and connected in the final graph.

4. Results

The EEGraph library has been developed with the main objective of creating a tool, both for researchers and clinical specialists in the field of neuroscience, that provides direct information on the connectivity of the brain from electroencephalography signals. As far as we know, there is nothing implemented in this regard with as many connectivity measures nor modeling EEGs as graphs. Next, installation and usage specifications of EEGraph are detailed.

4.1. Package requirements

This library has been implemented in the Python programming language and can be used in various operating systems. At the time of submission of this article, EEGraph version is 0.1.13. The installation is done using the command "pip install EEGRAPH". It utilizes the following libraries: *Numpy*[3], a package specialized in numerical calculation and data analysis [29], is used to perform the division in frequency bands through the fast Fourier transform, to implement the threshold value, and to define some operations within the computation of the connectivity measures (e.g. PS); *Pandas*[4], a library specialized in data structure management and analysis [30], is used both to structure the data in dataframes and to read the file with the electrode montage specifications; *Mne*,[5] a package for the exploration, visualization and analysis of neurophysiological data [18], is utilized to read the EEG file in the different supported formats; *NetworkX*,[6] a package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks [20], is used for the graphs creation and handling; *Plotly*,[7,8] an interactive plotting library, is used in the implementation of the visual representation of the graphs; *Scipy*,[9] a library that provides mathematical tools and algorithms [31], it is used to estimate Pearson correlation coefficient, the cross correlation, the squared coherence, the power spectral density, the Hilbert transform for the instantaneous phase calculation, the cross power spectral density, and the Shannon entropy, which are crucial in the definition of some of the connectivity measures considered (C, CC and CCC, SC, IC, PLV and PLI, WPLI, and H, respectively); *SCoT*,[10] a Python package for EEG/MEG source connectivity estimation [19], is used for the computation of the DTF; and *AntroPy*,[11,12] a Python package for computing different entropy and complexity metrics of time series, is used for the estimation of the SE. The necessary versions of each of these libraries are specified in the file *requirements.txt* of the GitHub.

4.2. Performance metrics

Some metrics that provide information about the performance of the library, such as execution times and tested versions, are provided below.

4.3. Covered code

The Codecov[13] analysis tool has been used to provide a report of the lines of code executed in the test performed. In particular, as shown in Table 1, 98.27% of the code inside the test folder and 85.97% of the lines of code in the library have been covered.

4.4. Compilation tests

The latest version (v 0.1.13) has passed compilation test successfully for the following Python versions:

- Python 3.7: time – 3 min and 6 s
- Python 3.8: time – 3 min and 6 s

Total compile time is 6 min and 12 s. The tests have been performed on Ubuntu 20.04 LTS Focal Fossa with the following libraries and packages: openssl (Ubuntu package), numpy (1.21.1), pandas (1.1.5), mne (0.22.0), networkx (2.5), plotly (4.14.3), scipy (1.1.0), scot (0.2.1) and antropy (0.1.4).

[3] https://pypi.org/project/numpy/.
[4] https://pypi.org/project/pandas/.
[5] https://pypi.org/project/mne/.
[6] https://pypi.org/project/networkx/.
[7] https://pypi.org/project/plotly/
[8] https://plotly.com/python/.
[9] https://pypi.org/project/scipy/.
[10] https://pypi.org/project/scot/.
[11] https://pypi.org/project/antropy/.
[12] https://raphaelvallat.com/antropy/build/html/index.html.
[13] https://about.codecov.io/.

4.5. Performance of the functions

Fig. A.2 in Supplementary materials shows the functions that have been executed to perform the tests (those within the file test/tests.py). The execution time of the whole program is 23 min (1380 s), where the executed classes are 98.45% of the execution time.

4.6. Input data

EEGraph allows a wide variety of EEGs reading formats as input to cover both raw and processed EEG signals. The formats admitted with their respective extensions are: Brainvision (.vhdr), Neuroscan CNT (.cnt), European data format (.edf), Biosemi data format (.bdf), General data format (.gdf), EGI simple binary (.egi), EGI MFF format (.mff) and eXimia (.nxe). Delving into the use of the library, to load the file associated to an EEG, hereafter, EEG file, one needs to create a *Graph*-like object and then call the "*load_data*" method specifying the "*path*" parameter, which should match the location of such file. If the EEG file does not contain the channel labels, EEGraph offers the option of introducing an additional file through the parameter "*electrode_montage_path*" which allows EEGraph to label and identify each signal with its corresponding electrode. This file must contain at least two columns, one with the numbering of the electrodes in the EEG file and the other with the corresponding channel labels, separated by the delimiters "$\backslash s+,;$", or ":". This option is disabled by default. Lastly, the "*exclude*" parameter allows the user to exclude a list of channels from the EEG file for the subsequent analysis.

4.7. Connectivity measures parameters

Once the EEG has been loaded, the next step is choosing the connectivity measure with which one wants to carry out the study, defining the associated parameters. For this purpose, the user may utilize the *"modelate*" method with the possibility of specifying the value of the parameters *"window_size*", *"connectivity*", *"bands*" and *"threshold*". The *"window_size*" parameter admits either an integer number, which is the time length (in seconds) of the segments into which the EEG is divided, or a list of integer numbers, which allow the user to specify the limits (in seconds) of such intervals. The parameter *"connectivity*" admits as input a string that specifies the name of the connectivity measure that the user wants to use. The specific names of these measures can be seen in detail in the section of *"Modelate Data*" within the GitHub wiki. The parameter *"bands*" admits, as input, a list of strings, in which the user can specify the band or bands for which they want to obtain the information. This parameter must be specified for those measurements that work on the EEG frequency spectrum, i.e., *"squared_coherence*", *"imag_coherence*", *"power_spectrum*", *"spectral_entropy*", *"wpli*", *"plv*", *"pli_bands*", and *"dtf*". Finally, the optional parameter *"threshold*" admits, as input, a float, and its default value can be found in the section *"Modelate Data*" of the GitHub. In the case of measures that relate pairs of electrodes, this parameter specifies a limit above which the connections are represented. In the case of measures that provide information about single electrodes, this parameter represents the percentage of nodes with the higher values that are fully connected. This percentage must be introduced divided by 100, i.e., as parts per unit.

4.8. Output data

To obtain the visual representation of the graph the user can utilize one of the outputs provided by the *"modelate*" method. The output *"graphs*" is a dictionary of all graphs generated from the EEG file as *NetworkX graph-like* objects. Each of these objects contains information about the connections of the graph and the weights of those connections that exceed the value of the parameter *"threshold*", which are the ones that will be represented. To obtain the image of the graph, it is necessary to call the method *"visualize*". This method requires two parameters: *"graph*", which must be a *NetworkX graph-like* object like the ones that made up the dictionary *"graphs*" above, and *"name*", a name for the graph that the user wants to visualize. Hence, the user needs to select the graph contained in the dictionary *"graphs*" that they want to represent and introduce it as the value of the parameter *"graph*". The result of this method is a graph like the one shown in Fig. 1, which is delivered to the user through an HTML file that is automatically saved in the user's folder with the name specified above, and is opened in the web browser. As it can be seen in Fig. 1, there are two display options: *"Show edge markers*", which shows the markers where the user must place the cursor to see the value of the weight associated with that edge (or node in the case of measures that provide information about single channel), and *"Hide edge markers*", which make the markers disappear, although the user can visualize the values of the measure in the same way. It should be noted that the width of each edge shown in this visual representation of the graph depends on the associated value of the chosen measure.

Finally, to obtain the matrix representation of the graph the user can utilize the other output of the *"modelate*" method. Precisely, the output *"connectivity_matrix*" is a *numpy.ndarray* with the form $G \times N \times N$, where G is the number of graphs and N is the number of EEG electrodes or nodes of the graph. It should be noted that *"connectivity_matrix*" contains all the values provided by the connectivity measure regardless of the value of the parameter "*threshold*". Once the matrix representation is obtained through the variable *"connectivity_matrix*" the user may utilize it within a Python script, or save it to a CSV file using the *"savetxt*" function of the *Numpy* library.

Table 1
Report of the Codecov analysis tool.

	Covered	Uncovered	Total
Test folder	284 lines	5 lines	289 lines
Full	576 lines	94 lines	670 lines

4.9. Reports on the library

The library EEGraph was tested by 6 researchers who provided an evaluation on the performance of the tool based on 6 questions. Their responses are presented in Table A.2. The assessments of the 6 researchers agreed that EEGraph was an easy to use library and that the documentation provided on GitHub was very useful to facilitate the handling of the library. One of them suggested that providing a definition of the frequency bands within the documentation would add value.

Regarding the variety of input formats, two of the researchers indicated the possibility of including additional input formats, like numpy arrays, while the remaining four indicated that the formats considered were sufficient and the most common ones, and appreciated the compatibility with the MNE library.

All the researchers agreed that the library included a large number of connectivity measures and that they were helpful. Moreover, even though two of the researchers did not try to modify the variables introduced by default, the remaining ones did, and they indicated the utility of this adaptability and pointed out that the functions from the tools.py file were very useful and that the code (despite not being heavily commented) was well structured and

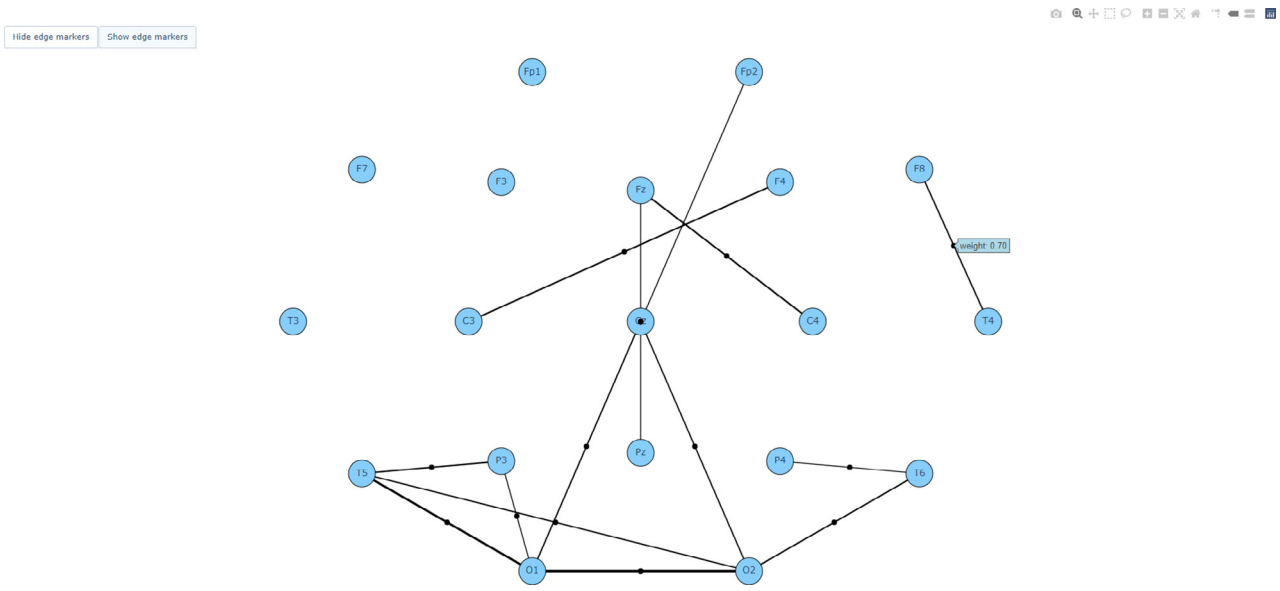

Fig. 1. Example of output for the method "*visualize*" with the display option "*Show edge markers*" activated.

easy to understand, making it easy to modify it or add new functions.

As for the output provided, two of the researchers did not comment, one indicated that the graphical results were easily visualized and understandable, one noted that the data provided about EEG parameters were very useful, in addition to the other outputs, and that although the visualization could be improved it was good enough to extract useful information, and the remaining two researchers focused their interest on the great potential of obtaining the data in a matrix form.

Thus, we can corroborate the favorable impact of the library on the scientific community, offering an easy to use tool, with a detailed guide that allows the user to be guided through the library during its different stages.

5. Discussion

Alterations in brain connectivity patterns can be associated with different neurological pathologies or disorders. To facilitate the study of these anomalies, we have developed EEGraph, a tool implemented in Python that models EEGs using graphs through the application of connectivity measures. On the one hand, EEGs have been chosen because of their easy accessibility and benefits for the patient. On the other hand, the graph theory has been selected to show the connectivity patterns exhibited in the EEG.

In this paper we present the details of EEGraph. The library allows a wide variety of EEGs reading formats as input, being able to identify each signal of an EEG with the position of its associated electrode in the EEG headset. It includes 12 of the most used connectivity measures in the literature, which cover both spectral and temporal features, providing the user with a great flexibility to define specific parameters and adapt them to each study. It enables the segmentation of the EEG in time-windows for their independent analysis, as well as the separation of the EEG signals in their associated frequency bands. All these characteristics offer the users a tool with high adaptability in a programming language known to most researchers. Furthermore, the library outputs the values of the connectivity measure, and an image with the visual representation of this information. This allows EEGraph to be utilized both in daily clinical practice and in neuroscience research.

The source code together with the documentation of the library are available online, on GitHub. We welcome feedback, requests on new connectivity measures, and suggestions on further implementations.

As future work, the available connectivity measures will be expanded and the visualization of results will be improved.

CRediT authorship contribution statement

Ana M. Maitin: Conceptualization, Software, Validation, Formal analysis, Investigation, Writing – review & editing, Writing – original draft, Supervision, Supervision. **Alberto Nogales:** Conceptualization, Software, Validation, Investigation, Writing – review & editing, Supervision, Supervision. **Pedro Chazarra:** Software, Validation, Formal analysis, Investigation. **Álvaro José García-Tejedor:** Conceptualization, Resources, Writing – review & editing, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to acknowledge all of the contributors to the EEGraph library, as well as A. Markitantov, M. Carrilero, A. Alramly, C. Gölz, S. Fernández-Rincón, and J. Pérez, for providing an experience report on the use of this tool.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.neucom.2022.11.050.

References

[1] P. Rossini, R. Di Iorio, M. Bentivoglio, G. Bertini, F. Ferreri, C. Gerloff, R. Ilmoniemi, F. Miraglia, M. Nitsche, F. Pestilli, M. Rosanova, Y. Shirota, C. Tesoriero, Y. Ugawa, F. Vecchio, U. Ziemann, M. Hallett, Methods for analysis of

brain connectivity: An IFCN-sponsored review, Clin. Neurophysiol. 130 (10) (2019) 1833–1858.
[2] C. Stam, B. Jones, G. Nolte, M. Breakspear, P. Scheltens, Small-world networks and functional connectivity in Alzheimer's disease, Cereb. Cortex 17 (2007) 92–99.
[3] D. Bassett, E. Bullmore, B. Verchinski, V. Mattay, D. Weinberger, A. Meyer-Lindenberg, Hierarchical organization of human cortical networks in health and schizophrenia, J. Neurosci. 28 (2008) 9239–9248.
[4] S. Leistedt, N. Coumans, M. Dumont, J. Lanquart, C. Stam, P. Linkowski, Altered sleep brain functional connectivity in acutely depressed patients, Hum. Brain Mapp. 30 (2009) 2207–2219.
[5] C. Stam, W. de Haan, A. Daffertshofer, B. Jones, I. Manshanden, A. van Cappellen, T. van Walsum, J. Montez, J. de Verbunt, B. van Munck, H. Dijk, P. Scheltens Berendse, Graph theoretical analysis of magnetoencephalographic functional connectivity in Alzheimer's disease, Brain 132 (2009) 213–224.
[6] A. Craik, Y. He1, J. Contreras-Vidal, Deep learning for electroencephalogram (eeg) classification tasks: a review, J. Neural Eng. 16 (2019) 031001. doi:10.1088/1741-2552/ab0ab5.
[7] C. Godsil, G. Royle, Algebraic Graph Theory, Springer, 2001.
[8] A. Feyissa, W. Tatum, Chapter 7 – Adult EEG, in: K.H. Levin, P. Chauvel (Eds.), Handbook of Clinical Neurology, vol. 160, Elsevier, 2019, pp. 103–124, https://doi.org/10.1016/B978-0-444-64032-1.00007-2.
[9] Y. Roy, H. Banville, I. Albuquerque, A. Gramfort, T. Falk, J. Faubert, Deep learning-based electroencephalography analysis: A systematic review, J. Neural Eng. 16 (2019) 1–37.
[10] A. Maitín, R. Perezzan, D. Herráez-Aguilar, I. Serrano, M. Castillo, A. Arroyo, J. Andreo, J. Romero, Time series analysis applied to EEG shows increased global connectivity during motor activation detected in pd patients compared to controls, Appl. Sci. 11 (1) (2021) 15.
[11] L. Ismail, W. Karwowski, A graph theory-based modeling of functional brain connectivity based on EEG: A systematic review in the context of neuroergonomics, IEEE Access 8 (2020) 155103–155135, https://doi.org/10.1109/ACCESS.2020.3018995.
[12] F. Fallani, L. Astolfi, F. Cincotti, D. Mattia, A. Tocci, M. Marciani, A. Colosimo, S. Salinari, S. Gao, A. Cichocki, F. Babiloni, Extracting information from cortical connectivity patterns estimated from high resolution EEG recordings: a theoretical graph approach, Brain Topogr. 19 (3) (2007) 125–136.
[13] M. Christodoulakis, A. Hadjipapas, E. Papathanasiou, M. Anastasiadou, S. Papacostas, G. Mitsis, Graph-theoretic analysis of scalp EEG brain networks in epilepsy–the influence of montage and volume conduction, in: 13th IEEE International Conference on BioInformatics and BioEngineering, IEEE, 2013, pp. 1–4.
[14] E. Olejarczyk, W. Jernajczyk, Graph-based analysis of brain connectivity in schizophrenia, PLoS One 12 (11) (2017) .
[15] J. Gomez-Pilar, R. de Luis-García, A. Lubeiro, N. de Uribe, J. Poza, P. Núñez, M. Ayuso, R. Hornero, V. Molina, Deficits of entropy modulation in schizophrenia are predicted by functional connectivity strength in the theta band and structural clustering, NeuroImage: Clinical 18 (2018) 382–389.
[16] M. Anastasiadou, M. Christodoulakis, E. Papathanasiou, S. Papacostas, A. Hadjipapas, G. Mitsis, Graph theoretical characteristics of EEG-based functional brain networks in patients with epilepsy: The effect of reference choice and volume conduction, Front. Neurosci. 13 (221) (2019) 1–18.
[17] L. Cabañero-Gomez, R. Hervas, I. Gonzalez, L. Rodriguez-Benitez, eeglib: A Python module for eeg feature extraction, SoftwareX 15 (2021) , https://doi.org/10.1016/j.softx.2021.100745.
[18] A. Gramfort, M. Luessi, E. Larson, D. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, M. Hämäläinen, MEG and eeg data analysis with mne-python, Front. Neurosci. 7 (267) (2013) 1–13, https://doi.org/10.3389/fnins.2013.00267.
[19] M. Billinger, C. Brunner, G. Müller-Putz, SCoT: a Python toolbox for EEG source connectivity, Front. Neuroinf. 8 (22) (2014) 1–11, https://doi.org/10.3389/fninf.2014.00022.
[20] A. Hagberg, D. Schult, P. Swart, Exploring Network Structure, Dynamics, and Function using NetworkX, in: G. Varoquaux, T. Vaught, J. Millman (Eds.), Proceedings of the 7th Python in Science conference (SciPy 2008), 2008, pp. 11–15.
[21] D. Grattarola, C. Alippi, Graph Neural Networks in TensorFlow and Keras with Spektral, arXiv:2006.12138.
[22] V. Jurcak, D. Tsuzuki, I. Dan, 10/20, 10/10, and 10/5 systems revisited: Their validity as relative head-surface-based positioning systems, NeuroImage 34 (4) (2007) 1600–1611.
[23] K. Pearson, Note on regression and inheritance in the case of two parents, Proc. R. Soc. London A 58 VII (347–352) (1895) 240–242. doi:10.1098/rspl.1895.0041.
[24] J.S. Bendat, A.G. Piersol, Random data, Wiley, 1986.
[25] J.-P. Lachaux, E. Rodriguez, J. Martinerie, F.J. Varela, Measuring phase synchrony in brain signals, Hum. Brain Mapp. 8 (4) (1999) 194–208, https://doi.org/10.1002/(SICI)1097-0193(1999)8:4<194::AID-HBM4>3.0.CO;2-C.
[26] C.J. Stam, G. Nolte, A. Daffertshofer, Phase lag index: Assessment of functional connectivity from multi channel eeg and meg with diminished bias from common sources, Hum. Brain Mapp. 28 (11) (2007) 1178–1193, https://doi.org/10.1002/hbm.20346.
[27] M.J. Kaminski, K.J. Blinowska, A new method of the description of the information flow in the brain structures, Biol. Cybern. 65 (3) (1991) 203–210, https://doi.org/10.1007/BF00198091.
[28] C.E. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. 27 (3) (1948) 379–423, https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.
[29] T. Oliphant, Guide to NumPy, CreateSpace Independent Publishing Platform, 2015.
[30] W. McKinney, pandas: a foundational python library for data analysis and statistics, Python High Performance Scientific Comput. 14 (9) (2011) 1–9.
[31] P. Virtanen, R. Gommers, T. Oliphant, et al., SciPy 1.0: fundamental algorithms for scientific computing in python, Nat. Methods 17 (2020) 261–272, https://doi.org/10.1038/s41592-019-0686-2.

Ana María Maitin studied a BSc in Physics, with a specialization in the theoretical branch, and an MSs in Biomedical Physics at the Complutense University of Madrid (Spain). She has worked in different projects within the physical and medical fields, such us the modeling of the cellular mechanics at the Complutense University of Madrid, and the statistical and non-linear analysis of electroencephalograms at the Francisco de Vitoria University (Spain). She is currently completing her PhD in Biotechnology, Medicine and Bio-Sanitary Sciences at the Francisco de Vitoria University as the recipient of a PhD fellowship. Moreover, she is conducting her research work at the Center for Studies and Innovation in Knowledge Management (CEIEC), where she has focused on the development of Machine Learning techniques, especially in Deep Learning techniques, and their application to different problems within medicine.

Alberto Nogales is a Computer Scientist with a Master in Software Engineer and Artificial Intelligence. He earns a PhD in University of Alcalá (Spain) focused in the field of Semantic Web and Social Network Analysis. He has experience in several European research projects and publications in conferences and journals from JCR. His education has been completed working in Technische Universität Wien (Austria), Tallinna Tehnikaülikool (Estonia) and University of Málaga (Spain). Since November 2017 works as PostDoctoral researcher and lecturer at Universidad Francisco de Victoria, at CEIEC research center. Actually, is working in Deep Learning and its use in Signal Processing and Predictive Models.

Pedro Chazarra has a degree in Computer Science and was intern at CEIEC research institute where he developed some projects in the field of bio signal processing and deep learning. In particular, he worked with epilepsy electroencephalograms with convolutional neural networks and graph convolutional neural networks. He actually works as a data scientist in Banco Santander.

Álvaro José García Tejedor, Ph.D. in Biochemistry. He has been Lecturer in UCM, UC3M and Universidad Francisco de Vitoria, where he is now Associate Professor in Artificial Intelligence. He has been involved in more than 20 R&D projects mainly in AI. As a researcher, he started doing mathematical modelling of biological systems. Since 2007 he heads CEIEC, a Research Institute of the UFV for technological innovation with a social accent. He has produced several serious games for transmission of educational and cultural content and integration of people with Intellectual Disabilities. His current work is focused on neural models, deep learning techniques and their applications in several fields.