Computers in Biology and Medicine 192 (2025) 110232

Contents lists available at ScienceDirect

Computers in Biology and Medicine

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

SleepEEGpy: a Python-based software integration package to organize preprocessing, analysis, and visualization of sleep EEG data

Rotem Falach [a,1], Gennadiy Belonosov [a,1], Flavio Jean Schmidig [a,b], Maya Aderka [a], Vladislav Zhelezniakov [b], Revital Shani-Hershkovich [c], Ella Bar [b,d], Yuval Nir [a,b,c,e,f,*]

[a] Sagol School of Neuroscience, Tel Aviv University, Tel Aviv, Israel
[b] Department of Physiology and Pharmacology, Faculty of Medical and Health Sciences, Tel Aviv University, Tel Aviv, Israel
[c] The Sieratzki-Sagol Center for Sleep Medicine, Tel Aviv Sourasky Medical Center, Tel Aviv, Israel
[d] Department of Brain Sciences, Weizmann Institute of Science, Rehovot, Israel
[e] Department of Biomedical Engineering, Faculty of Engineering, Tel Aviv University, Tel Aviv, Israel
[f] Sagol Brain Institute, Tel Aviv Sourasky Medical Center, Tel Aviv, Israel

ARTICLE INFO

Keywords:
Electroencephalography (EEG)
sleep
Hypnogram
Graphoelements
Software
Python
Open source

ABSTRACT

Sleep research uses electroencephalography (EEG) to infer brain activity in health and disease. Beyond standard sleep scoring, there is growing interest in advanced EEG analysis that requires extensive preprocessing to improve the signal-to-noise ratio and specialized analysis algorithms. While many EEG software packages exist, sleep research has unique needs (e.g., specific artifacts, event detection). Currently, sleep investigators use different libraries for specific tasks in a 'fragmented' configuration that is inefficient, prone to errors, and requires the learning of multiple software environments. This complexity creates a barrier for beginners. Here, we present SleepEEGpy, an open-source Python package that simplifies sleep EEG preprocessing and analysis. SleepEEGpy builds on MNE-Python, PyPREP, YASA, and SpecParam to offer an all-in-one, beginner-friendly package for comprehensive sleep EEG research, including (i) cleaning, (ii) independent component analysis, (iii) sleep event detection, (iv) spectral feature analysis, and visualization tools. A dedicated dashboard provides an overview to evaluate data and preprocessing, serving as an initial step prior to detailed analysis. We demonstrate SleepEEGpy's functionalities using overnight high-density EEG data from healthy participants, revealing characteristic activity signatures typical of each vigilance state: alpha oscillations in wakefulness, spindles and slow waves in NREM sleep, and theta activity in REM sleep. We hope that this software will be adopted and further developed by the sleep research community, and constitute a useful entry point tool for beginners in sleep EEG research.

1. Introduction

Electroencephalography (EEG) is the main tool in basic and clinically oriented sleep research [1]. EEG is routinely used in conjunction with electrooculography and electromyography to perform sleep scoring and distinguish between vigilance states of wakefulness, rapid eye movement (REM) sleep, and non-REM sleep [2]. Sleep scoring is performed either manually according to established standards [3] or, in recent years, via automatic tools [4–6]. Beyond sleep scoring, there is increased attention toward advanced EEG analysis that focuses on investigating events occurring at specific times, frequencies, and scalp locations or in estimated brain sources [7]. Examples of such sleep EEG analyses include an association between sleep spindles and sleep-dependent memory consolidation [8–13], regional differences in slow-wave activity during development [14], changes in slow-wave-spindle coupling in older age [15], and neural correlates of dreaming [16]. In clinical contexts, advanced analysis that goes beyond sleep architecture reveals an association between disrupted frontal slow waves and β-amyloid pathology in Alzheimer's disease [17], altered central sleep spindles in schizophrenia [18], and how epileptic seizures emerge from sleep oscillations [19–21].

EEG data comprise a mixture of signals of interest from neuronal

* Corresponding author. Sagol School of Neuroscience, Tel Aviv University, Tel Aviv, Israel
E-mail address: ynir@tauex.tau.ac.il (Y. Nir).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.compbiomed.2025.110232
Received 23 October 2024; Received in revised form 14 April 2025; Accepted 16 April 2025
Available online 26 April 2025

origin and noise arising from both physiological and extrinsic sources [22]. Typically, preprocessing (e.g., filtering and artifact rejection) is performed to improve the signal-to-noise ratio (SNR) of EEG signals [23]. While EEG may be "better left alone" in event-related contexts where noise can be reduced by trial averaging [24], it is crucial to preprocess and clean the EEG signal when analyzing the continuous, spontaneous brain activity observed during sleep. For example, many artifacts, such as sweating or eye movements, are dominated by spectral frequencies that overlap with slow-wave activity; hence, removing such artifacts is often necessary to accurately characterize sleep homeostasis as indexed by slow-wave activity [25].

Various established software packages exist for visualization, preprocessing, and analysis of EEG data [26]. Leading general-purpose EEG software packages include open-source MATLAB-based Brainstorm [27], EEGLAB [28], and FieldTrip [29], as well as Python-based MNE-Python [30]. All of these are under continuous development and have witnessed rapidly expanding interest in the scientific community (Fig. 1a). In addition, PyLossless [31] provides a pipeline and a GUI for continuous EEG preprocessing, focusing on automated artifact detection. EEGLAB and Brainstorm, developed earlier, provide a graphical user interface, whereas FieldTrip and MNE-Python are scripting-oriented. Over the last decade, automatic preprocessing pipelines have been developed and are being increasingly used for rejecting artifacts, problematic time intervals, and "bad" electrodes alongside conventional visual annotation [32–41] (Fig. 1b).

Sleep EEG research has many specific preprocessing, analysis, and visualization considerations compared with classic event-related potential (ERP) and resting-state EEG studies. For instance, different intrinsic artifacts may be dominant (e.g., cardiac activity, muscle twitches, and slow or rapid eye movements), whereas other common artifacts during wakefulness, such as eye blinks, are mostly absent [42]. Moreover, sleep EEG activity is predominantly spontaneous and not necessarily time-locked to external events, as in ERP studies. Therefore, continuous data are usually not windowed into epochs. This approach warrants preprocessing and visualization methods closer to the raw data, in line with a recently proposed 'lossless' preprocessing concept [43]. Additional considerations unique to sleep EEG analysis include sleep-stage-based analyses, such as detecting specific events such as slow waves and spindles, as well as examining spectral aspects and associated scalp topography. Thus, specific features of sleep EEG present a need for specific software tools tailored to these specific needs. Moreover, the quantity and complexity of these steps can often be overwhelming for beginners in the field.

Ideally, a complete sleep EEG software package should draw on the advantages of both general-purpose tools (e.g., preprocessing, time-frequency, and topography analyses) and specialized tools for sleep research (e.g., integration with sleep scoring, advanced analyses per sleep stage, and detection of specific sleep oscillations). At present, this is often accomplished by combining multiple existing and custom-made tools, leading to a "fragmented" configuration that can be inefficient, prone to errors, and demanding in terms of learning curve. These issues are an elevated entry barrier for new users and often lead to frustrating initial encounters with sleep EEG data.

In the landscape of Python-based tools for sleep research, Wonambi, Snooz Toolbox, Luna, PyLossless, and SleepEEGpy stand out as solutions targeting different aspects of EEG data analysis. Wonambi and Snooz Toolbox offer feature-rich graphical user interfaces (GUIs) that support manual sleep scoring, automated artifact rejection, and event detection, making them valuable for users who prefer interactive, visual workflows. In this sense, they are comparable to EEGLAB in the MATLAB ecosystem, providing a user-friendly environment with broad functionality accessible through a GUI. Luna, on the other hand, is a command-line-based toolkit designed for high-throughput batch processing of large sleep datasets, with a strong focus on flexibility and scalability for sleep signal analysis. In contrast, SleepEEGpy is designed with a focus on simplicity, flexibility, scalability, and reproducibility, following a pipeline-oriented approach that emphasizes automated preprocessing, report generation, and advanced analysis. Much like FieldTrip, SleepEEGpy offers a modular, script-based API, enabling researchers to work efficiently within Jupyter Notebooks while maintaining full control over each step of the analysis. This design not only paves a standardized workflow for users but also promotes reproducible research practices, which are critical in large-scale and collaborative projects.

Here, we present SleepEEGpy, a package that integrates preprocessing, analysis, and visualization for general sleep EEG data. It is designed to streamline workflows through a simple, script-based API, providing flexibility for both beginners and experienced researchers. We initially developed SleepEEGpy as a tool for new students in our lab to work with EEG data of human sleep. It is meant to offer a user-friendly introduction to sleep data analysis for users with little to no prior experience with EEG, sleep, or programming. Geared for beginners, it is not meant to replace the rich and complex functionalities of highly developed packages that it is based on (e.g., MNE); rather, it facilitates an entry point for newcomers in sleep EEG research. At the same time, its standardized visualization allows more experienced users to quickly assess the quality of the applied sleep-scoring, pre-processing, and analysis of the sleep data and thus enables them to provide helpful feedback and effectively mentor and supervise new users. SleepEEGpy is based on the following Python packages: MNE [30], PyPREP [33], YASA [44], and SpecParam [45] (formerly FOOOF). The choice of Python as an open-source programming language leverages the benefits of its

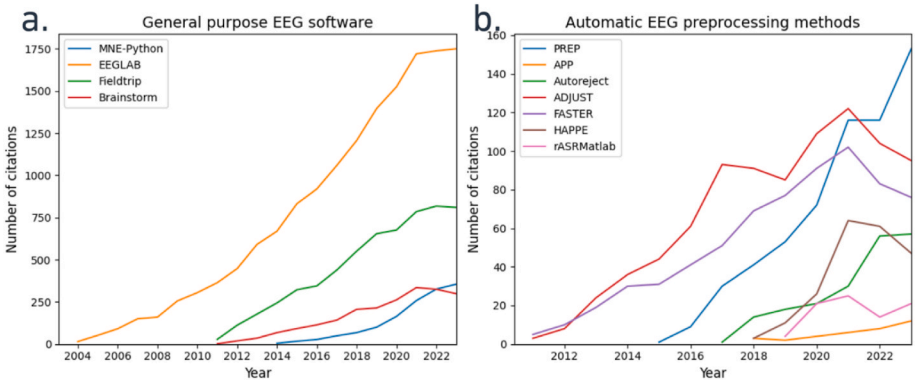


Fig. 1. Citation dynamics of common general-purpose and automatic preprocessing EEG software. A noncumulative number of citations (y-axis) per year (x-axis) for (a) four leading general-purpose EEG packages (EEGLAB, orange; Fieldtrip, green; Brainstorm, red; MNE-Python, blue) and (b) leading software packages implementing automatic preprocessing of EEG data (PREP pipeline, blue; APP, orange; Autoreject, green; ADJUST, red; FASTER, purple; HAPPE, brown; rASRMatlab, pink). Citations are based on the Scopus database (December 17, 2023).

many libraries, including extended machine learning ecosystems, broad documentation, and a dynamic community. To support rapid learning, SleepEEGpy includes example Jupyter notebooks demonstrating its pipeline functions [46]. To contextualize SleepEEGpy within the broader landscape of available tools, Table 1 presents a comparison of several publicly available sleep EEG toolkits across a range of key features. Notably, SleepEEGpy is not meant to replace or improve the packages it is built on. Its value primarily lies in simplifying getting started with these packages by unifying them into one framework and by reducing the functionality to the core necessities for the analysis of general sleep EEG data. In this study, we aim to demonstrate how SleepEEGpy simplifies complex sleep EEG analyses, supports standardized workflows, and enhances reproducibility in sleep research.

2. Methods

2.1. Overview

The SleepEEGpy pipeline (see Fig. 2 for a flowchart) is divided into two sections: preprocessing and analysis. Section A (preprocessing) is further divided into A1 (cleaning) and A2 (independent component analysis, ICA), whereas section B (analysis) is further divided into B1 (events) and B2 (spectral). The preprocessing section aims to increase the SNR by cleaning (e.g., filtering, rejecting bad electrodes or problematic temporal intervals, with rejections done either manually or automatically using MNE and PyPREP) and by regressing out noise components (through an ICA). The analysis section focuses either on specific sleep events (graphoelements such as sleep spindles, slow-waves, or rapid eye movements) or power spectral decomposition performed for individual recordings or multiple datasets. The dashboard and additional visualization tools allow a precise and sleep-tailored visual assessment of the preprocessing and the analysis section. Together, SleepEEGpy offers an integrated pipeline for cleaning, ICA, event detection and analysis, spectral analysis, and visualization, as well as integration with sleep scoring vectors (performed either a priori manually or automatically via YASA).

2.2. Prerequisites: input data, software, and hardware

To utilize SleepEEGpy, input data must consist of non-segmented ('continuous') EEG data in common formats supported by MNE-Python, such as Brain Vision, Meta File Format (MFF), or European Data Format (EDF). For sleep-stage-based functionality (both events and spectral), an additional sleep scoring vector is required in the form of a text file containing an integer per row representing different sleep stages for each epoch. For example, the sleep module of the Visbrain package provides an interface for sleep scoring that is well-suited and compatible with SleepEEGpy. If this data is not provided, SleepEEGpy can perform automatic sleep scoring using the YASA package. SleepEEGpy requires Python version 3.9 or higher, with Python versions between 3.9 and 3.11 recommended. It is advised to install SleepEEGpy in a Python virtual environment (using venv or conda) to avoid conflicts with other packages. Installation of SleepEEGpy is straightforward, and instructions are provided on the GitHub repository. Briefly, users should set up a virtual environment, install the required Python dependencies, and download the necessary notebooks to familiarize themselves with the library's functionalities. The repository also provides a quickstart guide, including a notebook with an end-to-end example of dataset retrieval, preprocessing, and analysis. Particularly for long overnight (6–10h) high-density (128/256-channel) EEG sleep datasets, we highly recommend at least 64 GB of rapid access memory (RAM), especially for event detection tools, even when the sampling rate is not higher than 256 Hz.

2.3. Architecture and typical workflow

Each tool within the pipeline is organized independently and has a corresponding Jupyter notebook. These notebooks serve as exemplars, providing a detailed walkthrough of each tool's functionality and offering step-by-step guidance to new users. More experienced users can always re-organize, reuse, and combine different pipeline tools to support their needs. Fig. 2 depicts a possible prototypic process flow of the SleepEEGpy pipeline.

Section A: Preprocessing. The preprocessing section (Fig. 2A) is divided into two tools: A1, cleaning, and A2, ICA. The dashboard offers a convenient way of reviewing the quality of the applied preprocessing steps. Furthermore, we provide a Jupyter notebook for the cleaning (A1) that can be applied with minimal knowledge of sleep and without further manual inputs. It uses default parameters for resampling, filtering, automatic rejection of bad channels and epochs, and automatic sleep-scoring. Naturally, default parameters will not result in ideal preprocessing, which has to be evaluated carefully. However, it provides a first-pass "low-entry" point for beginners to start exploring the preprocessing of sleep EEG and refine further via additional iterations.

Table 1
Comparison of SleepEEGpy with other sleep EEG analysis toolkits. The table presents a side-by-side overview of different features across SleepEEGpy and several existing software tools for sleep EEG analysis.

Feature/Tool	SleepEEGpy	Wonambi	Snooz Toolbox	Luna	PyLossless	MNE	EEGLAB	FieldTrip
Platform	Python	Python	Python	Command line + R & Python extensions	Python	Python	MATLAB	MATLAB
Interface	Scripting	GUI + scripting	GUI	GUI + scripting	Scripting	Scripting	GUI + scripting	Scripting
Sleep Scoring	Automated via YASA	Manual	No	Automated	No	No	No	No
Artifact Handling	Automated via PyPREP or manual via MNE	Manual	Automated	Automated	Advanced auto-cleaning	Manual	Automated + manual	Automated + manual
Event Detection	Spindles, slow waves, REMs via YASA	Spindles, slow waves	Spindles, slow waves	Spindles, slow waves	No	No	No	No
Sleep Report	Standardized dashboard + pipeline plots	Interactive GUI + CSV sleep stats	TSV sleep stats	Console summaries + pipeline plots	No	Basic plotting functions	Basic plotting functions	Basic plotting functions
Scalability	High	Moderate	Moderate	Very high	Moderate	High	Moderate	High
Prior Knowledge	Low	Moderate	Moderate	High	Low	Moderate	Low	Moderate
Documentation	Online docs + Jupyter tutorials	Online docs	Online docs	Extensive online docs + tutorials	Online docs	Extensive online docs + tutorials	Online docs + community	Online docs + community

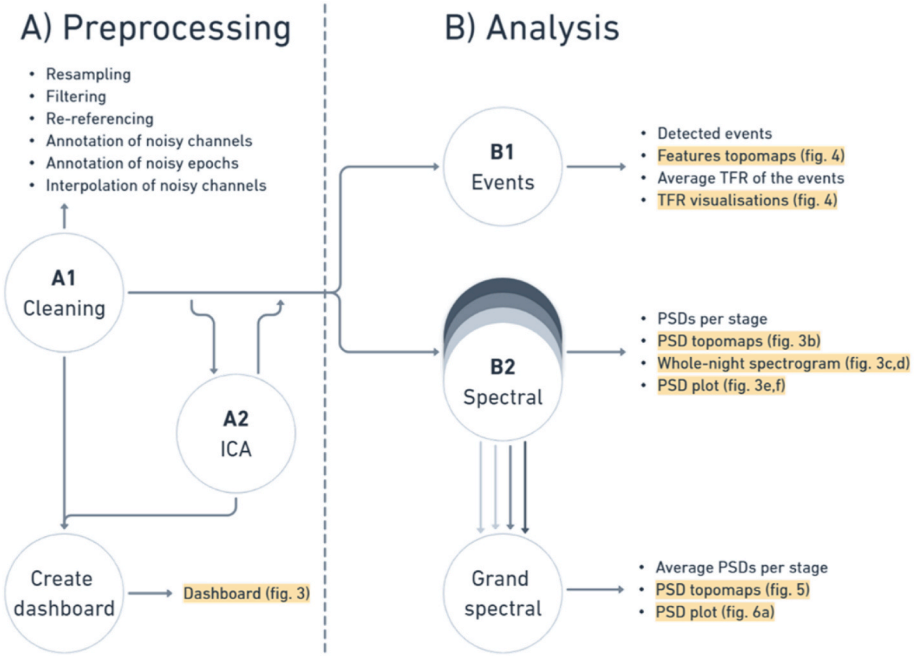


Fig. 2. Flowchart of the SleepEEGpy pipeline. Section A (preprocessing) is divided into A1 (cleaning) and A2 (ICA). The cleaning tool (A1) handles operations such as resampling, filtering, and the annotation of noisy electrodes or epochs, with the option to interpolate affected channels. The ICA tool (A2) provides component visualization and rejection. A dedicated dashboard (Fig. 3) offers a visual summary of preprocessing. Section B (analysis) comprises B1 (detection of sleep events like spindles or slow waves) and B2 (spectral analysis for single or multiple datasets). Arrows indicate transitions between the pipeline tools. Overlapping circles and multiple arrows illustrate the merging of multiple single-subject analyses into a single group-level analysis, while text blocks represent key outputs, with visualizations highlighted. Abbreviations: PSD, power spectral density; TFR, time-frequency representation.

A1. Cleaning. The pipeline cleaning tool consists of the following signal processing techniques: EEG resampling, filtering, re-referencing, noisy channels and temporal intervals annotation, and interpolation of the annotated channels. Resampling is often a necessary step in sleep EEG preprocessing because overnight recordings may require tens of gigabytes of RAM. Filters and data annotations provide a way to improve the SNR by removing artifacts from the signal. For example, by applying a notch filter to remove electrical line noise (50/60Hz) or by annotating the intervals of movements to avoid their subsequent analysis. SleepEEGpy employs the same filtering methods as MNE, ensuring consistency with widely used EEG preprocessing standards. Default parameters, along with their rationale, are presented below. Re-referencing EEG data to the common average is typically used for assessing topography distributions after interpolating noisy channels, ensuring that the mean is not dominated by outliers. Since there is no widely accepted standard for automatically detecting bad channels and noisy epochs in sleep EEG research, the default approach is manual inspection and tagging. However, users can opt for automatic cleaning algorithms tested on our example sleep datasets or combine both methods. Automatic bad channel detection is based on PyPREP, which uses RANSAC, while noisy epoch identification follows MNE's approach, relying on peak-to-peak amplitude thresholds.

A2. ICA. The ICA tool is MNE-based and consists of EEG signal decomposition, selection of artificial components based on a data browser and components' topographies, and subsequent EEG signal reconstruction after regressing out the artificial components. EEG, and sleep EEG specifically, contains noise sources unrelated to brain activity. We can identify and to some extent separate, the noise from the neuronal signal. For example, physiological noise (e.g., electrocardiograph, sweating, rolling eye movements) or external noise (e.g., 50/60Hz line noise) can often be reliably identified based on the components' topographies and time series. Applying ICA is optional because there are advantages and disadvantages in removing or keeping some components, depending on the subsequent analysis and the focus of the investigation. For example, eye movement-related potentials during sleep may mask neuronal activities and be chosen to be removed in some contexts; however, in other studies, the research question may require their detection [47]. Most sleep EEG studies do not employ ICA and prefer to discard entire 20s/30s segments based on manual identification given the rich, long datasets. However, a disadvantage of this approach is that it may limit artifact identification to the few channels used for sleep scoring [48].

Default parameters. By default, the parameters for preprocessing and visualization are set as follows:

- Band-pass filter: the default is high-pass (but not low-pass) filtering of the data with a cutoff of 0.3 Hz. As defined by MNE, the default filter is a zero-phase (non-causal) finite impulse response (FIR) filter using the window method with the hamming window. Stricter high-pass filtering, e.g., 0.75 Hz, may be preferable in the presence of high-amplitude sweat artifacts [48].
- Notch filter: By default, we set 50 Hz and its harmonics to be filtered using a notch filter. Like the band-pass filter, the notch filter is a zero-phase FIR filter with a hamming window and 1 Hz width of the transition band.
- ICA: The default ICA algorithm, based on MNE's implementation, is set to FastICA [49], which performs a contrast-based optimization to maximize non-Gaussianity and extract statistically independent components. Alternatively, users can switch to Infomax [50] which applies a neural-network-inspired gradient-based learning rule, or Picard [51], which leverages a quasi-Newton optimization approach with improved convergence properties. The number of largest-variance PCA components passed to the ICA algorithm is set to 30 by default. Following MNE's recommendation, the signal is by default high-pass filtered at 1.0 Hz before fitting to reduce the ICA algorithm sensitivity to low-frequency drifts.

The rationale behind these default parameters is to provide a

balanced, "ready-to-use" setup for general sleep EEG analyses. Our choices mirror widely adopted practices in the EEG research community (e.g., MNE defaults). Advanced users with specific research questions or data characteristics, such as very slow oscillations in pathological populations, are encouraged to modify these defaults accordingly.

Dashboard: summary and visualization of preprocessing. To provide a visual summary and overview of sleep EEG data preprocessing, we created a dashboard (Fig. 3). It accepts as an input the output of either A1 or A2 (Fig. 2A). Hence, the dashboard can visualize cleaned EEG recordings with or without excluded ICA components. Furthermore, one can optionally refine the visualization by adding a sleep-scoring vector. Based on the input, all visualizations in the dashboard are then computed with or without ICA or sleep scoring. As a pre-configured visualization tool, it displays the characteristic sleep properties of the EEG recording, such that the data quality and SNR of that sleep dataset can be estimated relatively quickly from a "bird's eye" view. The dashboard includes four sections: general preprocessing information (Fig. 3a), topographical power distribution of key oscillations in specific vigilance states (Fig. 3b), time-frequency decompositions (spectrogram, computed using the Multitaper method [52]) superimposed with the corresponding hypnogram, once before (Fig. 3c) and once after (Fig. 3d) the rejection of bad time intervals (and after ICA, if used), and power spectral density (PSD) plots before and after the preprocessing (Fig. 3e and f). A full description of the dashboard results for representative data of overnight sleep in a healthy subject is detailed below in the Results section.

Section B: Analysis. The analysis section (Fig. 2B) is divided into two tools: B1, event-based analysis, and B2**,** spectrum-based analysis. B1 is mainly based on YASA and B2 is based on MNE and SpecParam. Here we describe the main functionalities; for fine-tuning of additional parameters, it is best to refer to the documentation of the original package. The major advantage here is that SleepEEGpy offers the event- and spectrum-based analysis of sleep EEG within the same framework as the preprocessing and thus is easier to get started for new users. Nevertheless, if one is already acquainted with YASA, employing the package directly might be more convenient than submitting the parameters to YASA through SleepEEGpy.

B1. Event detection and analysis tools. Event-based tools in the SleepEEGpy pipeline are used to detect and characterize three different sleep graphoelements, namely sleep spindles, slow waves, and rapid eye movements. Each tool takes an MNE-readable EEG file and the sleep scoring vector as input. The input EEG file is typically the output of the preprocessing section, but this is not mandatory. The detection methods of the event-based tools are completely based on YASA detection algorithms [44] and allow the same input parameters. The output of the detection algorithm provides features of the detected events (for example, number of events, amplitude/frequency characteristics of sleep spindles), as well as the average time-frequency representation averaged per channel across the detected events. The associated visualizations (as seen in Fig. 4) include average event time-course plots, topographical distributions of the events' features, and time-frequency representations.

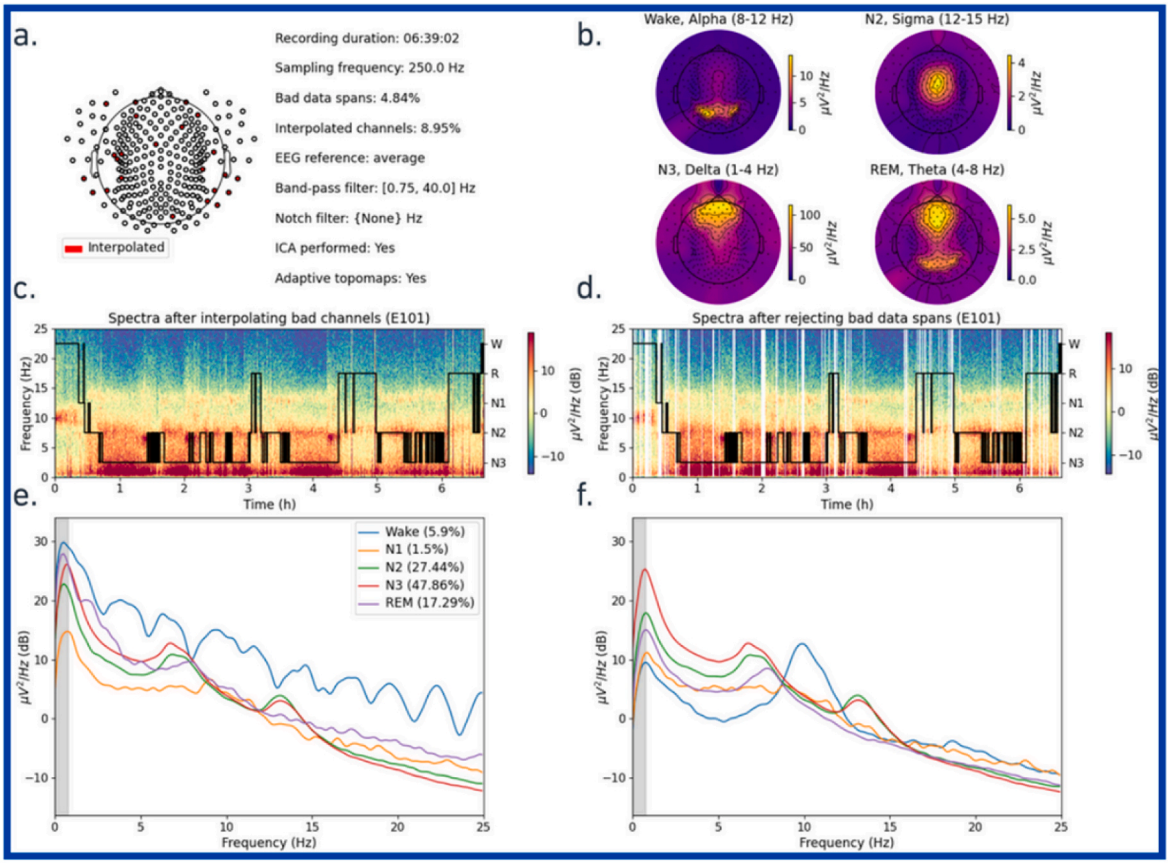
Sleep spindles are detected using YASA's algorithm. The algorithm is based on Lacourse et al. [53] and is similar to our previous work [54]. In brief, detection is performed by the algorithm using a combination of different thresholds (relative sigma power, root mean square, and correlation) separately for the broadband (by default 1–30 Hz) and sigma-filtered (by default 12–15 Hz) EEG signals. Detection signals are resampled to the original time vector of the EEG data using cubic interpolation to facilitate better precision for spindle start time, end time, and duration. The relative sigma power (relative to the total power in the broadband frequency) is computed using a short-term Fourier transform. Each spindle's median frequency and absolute power are computed using the Hilbert transform. Additional spindle properties are computed (e.g., symmetry index) as suggested in Ref. [55]. Detected events with a duration within the range of a prototypical spindle are kept for subsequent analysis (default duration is 0.5–2 s). Finally, an isolation forest algorithm is optionally applied to reject outliers. The EEG signals of the detected spindles can be extracted with a corresponding function, and the properties of the spindles can be accessed through a summary Pandas dataframe.

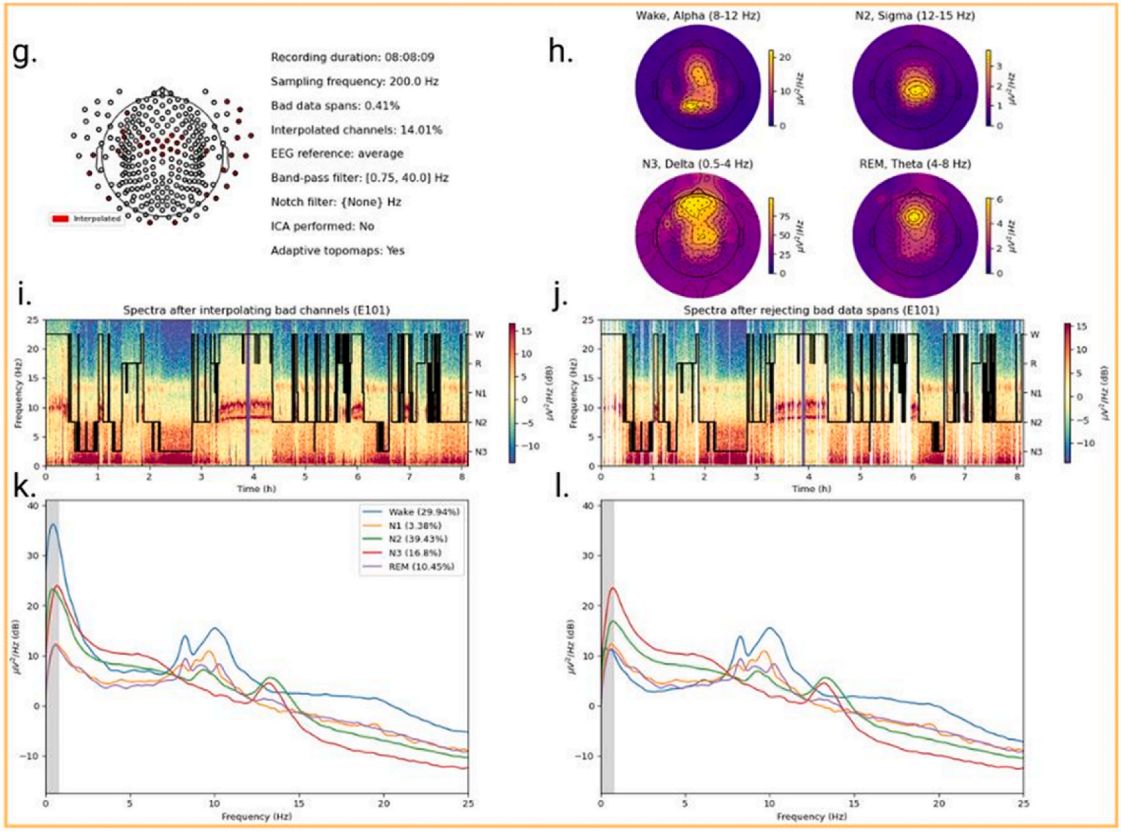
Slow waves are detected via YASA's algorithm, which is based on a study by Massimini and colleagues [56] and similar to our own previous work [57]. In brief, the EEG signal is band-pass filtered in the default range of 0.3–1.5 Hz using an FIR filter with a transition band of 0.2 Hz. Next, negative peaks in the filtered signal with an amplitude of [−40 to −200] uV are detected, as well as all positive peaks with a default amplitude of [10 to 150] uV. For each negative peak (slow wave trough), the nearest positive peak is found, and several metrics are computed, such as the peak-to-peak amplitude, durations of the negative and positive phases, and frequency. A set of logical thresholds, e.g., phase duration, is applied to determine the true slow waves. A Pandas dataframe is created, where each row is a detected slow wave, and each column represents a property of this slow wave. An optional automatic outlier rejection is applied to further remove abnormal slow waves. Similar to the spindle detection algorithm, the EEG signals of the detected slow waves can be extracted.

The detected spindle and slow wave events can be further analyzed using the average time-frequency representation (TFR). Either Morlet wavelets [58] or discrete prolate spheroidal sequence (DPSS) tapers [59] can be used for the TFR computation as implemented by MNE. For the TRF computation of the detected events, we first extracted the EEG signal with a user-defined duration (by default, −1 to 1 s around the central peak for spindles and negative peak for slow waves). Second, the extracted events are used to compute the average TFR, where the TFR is averaged over events in each channel and each sleep stage. Finally, the average TFR is placed inside an MNE-based container ('AverageTFR'). It provides additional extensive data manipulation functionality and various visualizations (e.g., Fig. 4b).

Rapid eye movements (REMs) are also detected using YASA's algorithm. The algorithm uses the rapid eye movement detection method based on left and right outer canthi (left: LOC, right: ROC) EOG data proposed in Ref. [60]. In brief, the algorithm uses amplitude thresholding of the negative product of the LOC and ROC signals. The REM peaks are detected based on the user-defined frequency range (defaults to 0.5–5 Hz), amplitude threshold (defaults to min 50 and max 325 μV), and REM duration (defaults to min 0.3 and max 1.2 s). Similar to spindle and slow wave detection algorithms, the outliers of the detected REMs can be removed using the isolation forest algorithm, and the summary Pandas dataframe and their EEG signal can be extracted.

B2. Spectral tools. Spectral-based tools in the SleepEEGpy pipeline include single recording or multiple dataset analyses. The single recording spectral tool takes an MNE-readable EEG file as input, and the sleep scoring vector (if not provided, can be automatically predicted by the tool with YASA's algorithm). The spectral tool for multiple dataset analysis then takes multiple instances of the single recording tool as input. The numeric output of both tools is PSDs calculated separately for each sleep stage. The PSDs per sleep stage are computed using Welch's method [61] in the following way: first, the EEG signal is divided into regions according to changes in sleep depth, i.e., at the end of the division, there can be multiple segments for each sleep stage. Then, using Welch's method, PSDs are separately computed for each segment. Finally, separately for each sleep stage, a weighted arithmetic mean based on the length of a segment is applied to the PSDs, producing a per-sleep-stage PSD. The PSD computation is based on the MNE's Welch function (psd_array_welch), accepts all the original parameters, and uses the same defaults (e.g., the length of FFT and Welch's segments are set to 256, the segments' overlap is 0, the default window is the hamming type). In addition, the multiple dataset spectral tool averages PSDs over recordings. Finally, the computed PSDs per sleep stage are placed inside MNE-based containers ('SpectrumArray') to preserve the rich spectrum-related functionality of MNE.





(caption on next page)

Fig. 3. Preprocessing assessment with the dashboard. Top, blue box: healthy young adult. Bottom, orange box: healthy older adult. **a)** Left: EEG net montage with spatial distribution of the interpolated channels (in red); right: general information of the recording's preprocessing. **b)** Topographical PSD distribution per sleep stage in its characteristic frequency band (8–12 Hz for Wake, 12–15 Hz for N2, 1–4 Hz for N3, and 4–8 Hz for REM); subfigures **c–f** are based on the E101 electrode (Pz); subfigures **c)** and **d)** present single-channel spectrograms of the recording (colored power distribution in logarithmic scale over time and frequency) overlapped with a hypnogram (black line representing depth of sleep as a function of time). **c)** Spectrogram of the signal after filtering and bad channel interpolation. **d)** Spectrogram after filtering, bad channel interpolation, bad data span rejection, and optionally, exclusion of artificial independent components (ICA). Subfigures **e)** and **f)** present single-channel power spectral distribution (PSD) plots as a function of frequency, with lines representing different sleep stages: blue for Wake, orange for N1, green for N2, red for N3, and purple for REM. The gray area on the left of the plots shows frequencies filtered by the high-pass filter. Note that the cleaning steps yield clearer spectral characteristics, such as a more prominent alpha peak during wakefulness. **e)** PSD plot of the signal after filtering and bad channel interpolation. **f)** PSD plot of the signal with additional bad data span rejection and, optionally, exclusion of artificial independent components. Panels **g–l** (within the orange box) are equivalent to panels **a–f** but for the healthy older adult. Note that the older adult experiences multiple brief awakenings, visible as more frequent transitions to Wake.

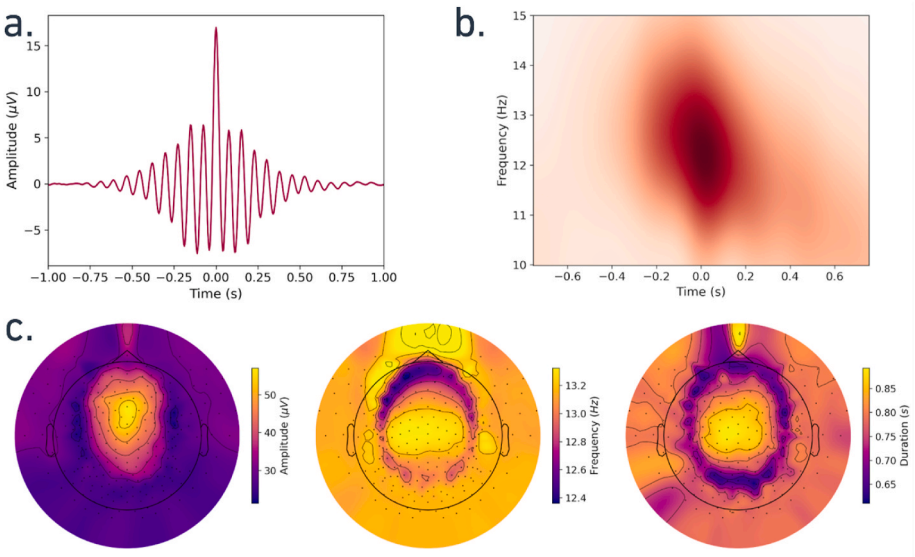


Fig. 4. Visualizations of the detected spindles. Overall, 48057 spindles were detected in 257 channels of 193min N2 sleep in a healthy young adult. (a) Average signal over all detected spindles in all channels centered around the spindle peak. (b) Average time-frequency representation of detected spindles from channel E101 (Pz). (c) Average topographical distribution of spindle characteristics: amplitude (left), frequency (middle), and duration (right). Note the typical separation into slower frontal spindles versus faster centroparietal spindles.

PSD results can be visualized either in a "traditional" (power vs. frequency) plot (as in Fig. 3e/f or Fig. 6a) or with corresponding scalp topography distributions (as in Fig. 3b or Fig. 5). Additional manipulations and visualizations of the PSDs per sleep stage are available through the MNE containers. In-depth spectral analysis can be conducted with SpecParam (formerly FOOOF) [45] (Fig. 6b). This analysis improves the characterization of signals of interest by overcoming the limitations of conventional narrowband analyses, e.g., misinterpretation of physiological phenomena. This is accomplished by parameterizing neural PSDs into periodic and aperiodic components. This algorithm can identify periodic oscillatory parameters, including the center frequency, power, and bandwidth. In addition, offset and exponent parameters can be extracted for the aperiodic component.

Each spectral tool can be computed and viewed at the single recording level (as in Fig. 3e/f) or across multiple datasets (Figs. 5 and 6) recordings, where PSDs are averaged separately across multiple datasets for each EEG electrode and frequency band. Finally, to allow additional preprocessing flexibility and benefit from the interface with diverse approaches, spectral tools can also accept an epoch-based signal annotated with sleep stages as an input (as in ERP studies) rather than a 'continuous' EEG signal typical for sleep studies.

2.4. EEG data

We illustrate SleepEEGpy functionalities by applying it to sleep recordings of 44 healthy, young adult participants (25 females, age 25.86 ± 3.14 years (mean ± STD), ranging from 21 to 36 years) who participated in a research study on sleep and memory consolidation. In addition, we included one healthy older adult (male, age 68) who participated in a sleep and neurodegeneration study (as part of the control cohort). Written informed consent was obtained from each participant. These studies were approved by the Medical Institutional Review Board of the Tel Aviv Sourasky Medical Center. Participants did not have any history of neuropsychiatric or sleep disorders. Each participant arrived at the sleep lab around 21:00 and, after cognitive testing, proceeded to undisturbed sleep (the data presented here). The mean duration of the recording was 7:18:28 ± 0:36:20 (hours:minutes:sec, mean ± STD). We collected polysomnographic data, including high-density electroencephalogram (hd-EEG), EOG, EMG, and video, as described in [13]. hd-EEG was recorded using a 256-channel (plus one reference channel) hydrogel geodesic sensor net (Electrical Geodesics, Inc. [EGI]). Signals were referenced to Cz, amplified using an anti-aliasing filter and an AC-coupled high input impedance amplifier (NetAmps 300, EGI), and digitized at 1000 Hz. Before recording began, after conductive gel application, all sensors' electrode impedance was confirmed to be at least below 50 kΩ.

2.5. Comparison with EEGLAB

To further ensure the consistency and validity of SleepEEGpy's outputs, we conducted a data-driven comparison with a widely used EEG analysis tool, EEGLAB (Fig. 7). Specifically, we compared the topographical distribution of key EEG frequency bands during distinct sleep stages, using our publicly available nap dataset. We focused on four key

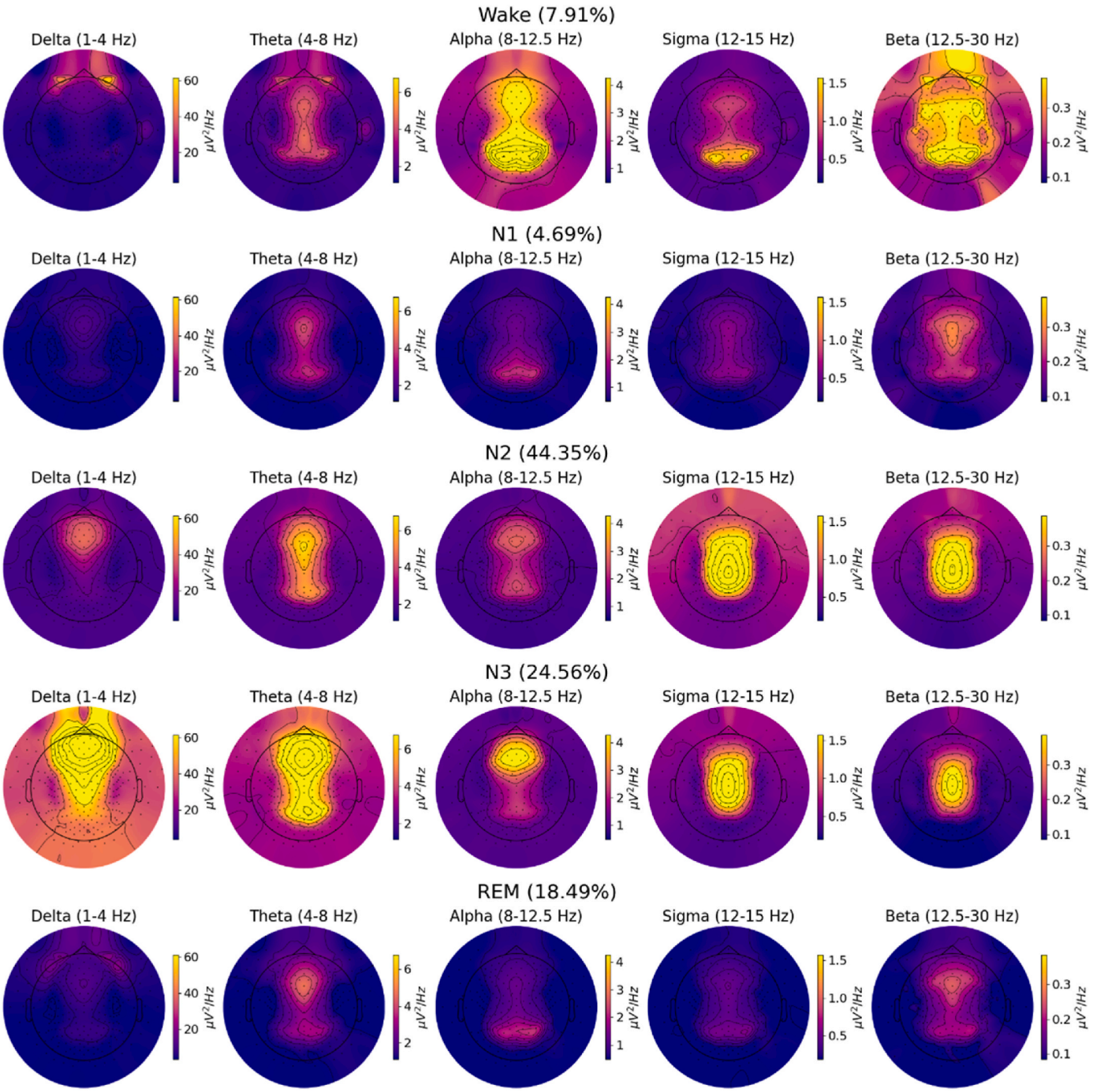


Fig. 5. The topographical distribution of PSD per frequency band and sleep stage averaged over 40 subjects. Each row represents a sleep stage, from top to bottom: Wake, N1, N2, N3, and REM. Each column represents a frequency band, from left to right: Delta (1–4 Hz), Theta (4–8 Hz), Alpha (8–12.5 Hz), Sigma (12–15 Hz), Beta (12.5–30 Hz). Percentages in brackets represent a fraction of the sleep stage signal from the overall data.

combinations also exist in the dashboard: alpha activity (8–12 Hz) during wakefulness, sigma activity (12–15 Hz) during N2, delta activity (0.5–4 Hz) during N3, and theta activity (4–8 Hz) during REM sleep. Both pipelines were run with similar preprocessing parameters, including bandpass filtering, bad channels interpolation, and common average referencing, to ensure consistency across software environments. The scalp topographies were then computed per frequency band and sleep stage using identical time windows.

3. Results

To illustrate the functionality, typical workflow, specific tools, and visualizations of the SleepEEGpy pipeline, we performed preprocessing and analysis of overnight sleep EEG data obtained in 44 young, healthy adults participating in a memory consolidation study (Methods). The analysis was conducted using a structured sequence of Jupyter notebooks, beginning with preprocessing and artifact rejection, followed by event-based detection, and concluding with spectral analysis. Each step is demonstrated using representative datasets before extending the analysis to the full dataset, highlighting both individual case studies and

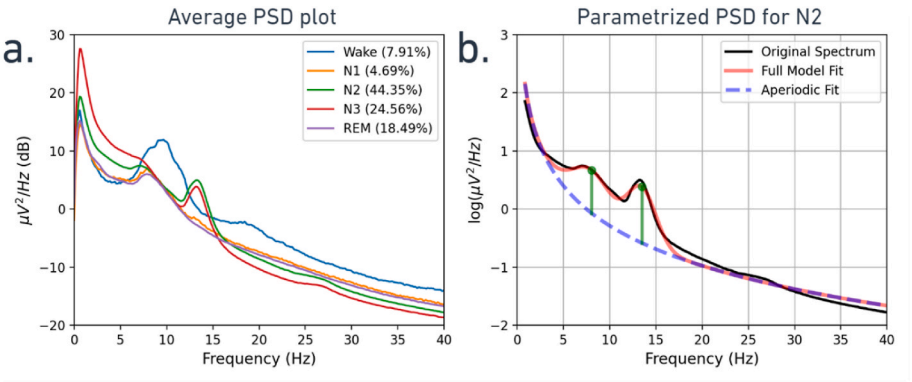


Fig. 6. Group-average PSD plots. (a) PSDs from the E101 (Pz) electrode averaged over 40 overnight recordings. The PSDs were transformed to dB for visualization. Percentages in the legend represent the fraction of a sleep stage signal from the overall signal. (b) Average PSD of the N2 stage parametrized with SpecParam. The PSD values were log-transformed. Vertical green lines represent fitted peaks with central frequencies of ~8.06 and ~13.47 Hz, and their respective power above the aperiodic component of ~0.74 and ~0.97 log(uV^2/Hz). PSD, power spectral density.

overall trends. The following sections will present results that showcase the capabilities of SleepEEGpy, including automated cleaning, visualization, and spectral and event-based analyses. This approach demonstrates how SleepEEGpy can be applied not only to overnight recordings but also to other sleep studies, such as daytime naps or studies in clinical populations. Similar applications can be used for any type of sleep recording, such as a daytime nap or a study in clinical populations. In this study, each dataset consisted of a ~7h undisturbed sleep hd-EEG (256-channel, EGI) combined with polysomnography including EOG, EMG, and video.

3.1. *Section a preprocessing: cleaning and optional ICA*

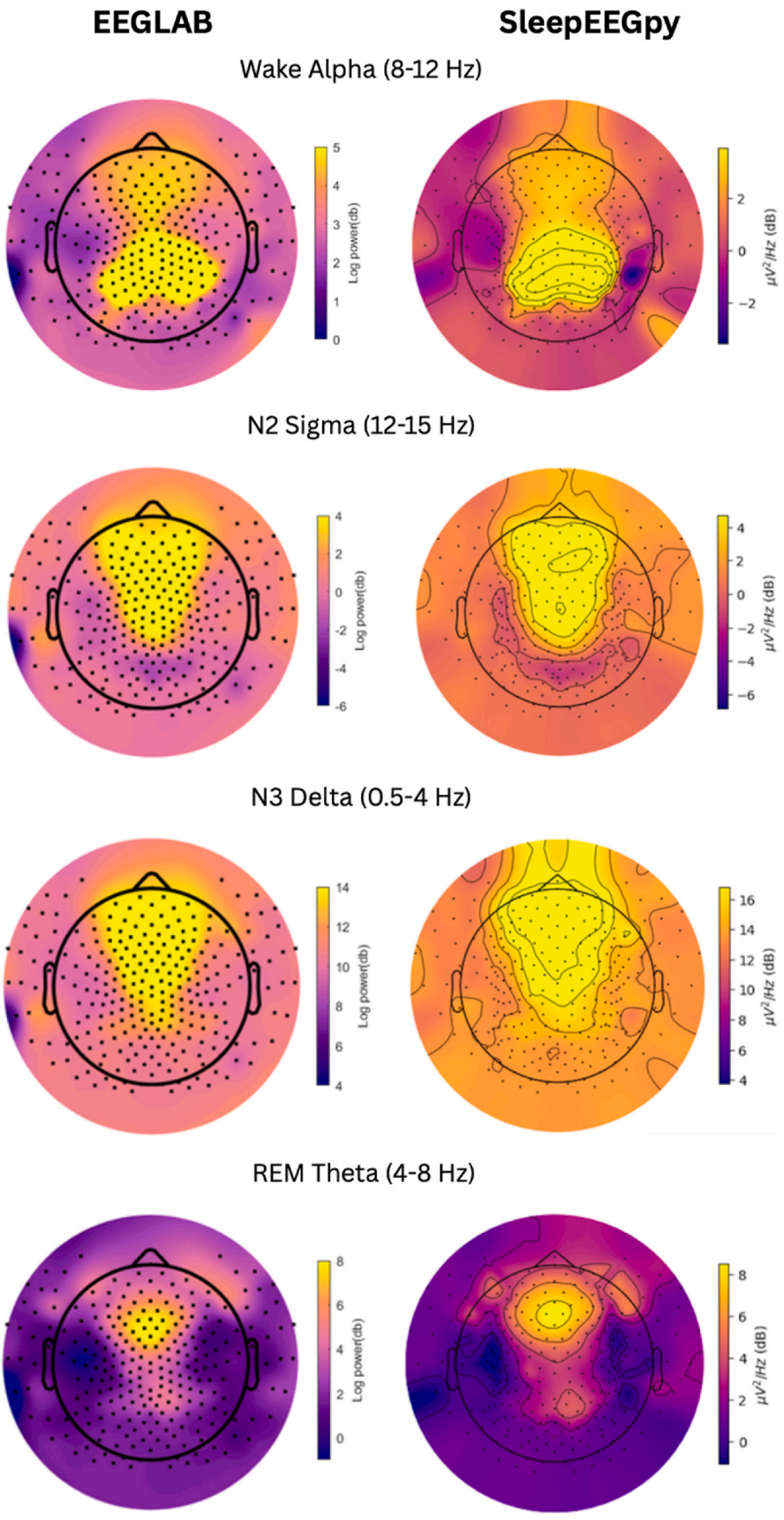
The first step in the pipeline is to perform cleaning (Fig. 2a) via downsampling, filtering, visual annotation, and interpolation of noisy electrodes, as well as identification and exclusion of noisy temporal intervals from subsequent analysis. We illustrate this on one representative dataset (27-year-old female). For the preprocessing, we downsampled the data to 250 Hz; and used a common average reference. Additionally, we choose to perform ICA (see below). In terms of filtering, we used a band-pass filter of 0.75–40 Hz, without applying a specific notch filter.

Next, as part of the SleepEEGpy pipeline, we used an MNE-based data browser to annotate bad channels and bad temporal intervals. Manual inspection and annotation of raw signals were performed in the "butterfly" view and complemented by PSD topography inspection to finalize the exclusion of abnormal channels. In our example of a young adult participant, 8.95 % of the channels were annotated as bad and interpolated using MNE-based spherical spline interpolation [62]. Across the entire dataset (N = 44), 14.0 ± 4.6 % (mean ± STD) were marked as "bad channels", and four subjects with more than 25 % bad channels were excluded, leaving N = 40 for subsequent analysis. Next, we marked temporal intervals in the representative dataset as "bad" (4.84 % of the data in the example, 5.3 ± 2.4 % across the entire N = 40 dataset). Next, sleep scoring was performed manually according to established American Academy of Sleep Medicine (AASM) criteria (3) to create a specific vector text file that could be fed to SleepEEGpy as input for subsequent preprocessing and analysis (Methods). The other 39 datasets were sleep-scored automatically using YASA's algorithm [44]. Lastly, we applied ICA to identify and remove components of the EEG unrelated to brain activity. We chose to annotate and remove components associated with the electrocardiograph/heartbeat before proceeding to the analysis. For the older adult example, we applied fully automated cleaning using MNE and PyPREP, along with manual sleep scoring.

Fig. 3 presents the dashboard, a visual summary after preprocessing. The dashboard serves as an initial assessment tool for preprocessing quality, allowing users to identify noise, detect patterns in sleep stages, and evaluate spectral characteristics before detailed analysis. Here we used a representative dataset of one young adult and one older adult and plotted the activity at the Pz electrode. We created the dashboard to be a helpful, standardized visualization to evaluate the quality of the applied preprocessing. It includes the following subplots: general preprocessing information (Fig. 3a), topography of selected frequency ranges (3b), TFR with hypnogram before (3c) and after (3d) rejection of bad intervals and interpolation of bad electrodes, PSD plots of vigilance state before (3e) and after (3f) cleaning. To assess preprocessing effectiveness, users can look for typical topographies, as seen in Fig. 3b, or compare Fig. 3e and f to confirm noise removal while preserving expected spectral features, such as the clear alpha peak in wakefulness, and the dominant low-frequency power in N3 sleep.

Since the cleaning (no ICA was applied) was successful, the topographic PSD maps of the dominant EEG rhythms for each vigilance state (Fig. 3b) show A) Wakefulness (top left) is characterized by maximal alpha (8–12 Hz) activity over the occipital lobe. B) In N2 sleep (top right), sigma (12–15 Hz) activity predominates over the centroparietal electrodes. C) In N3 sleep (bottom left), slow wave activity (<4 Hz) is maximal over the frontal cortex, and D) REM sleep (bottom right) is characterized by theta (4–8 Hz) activity with its signature scalp topography.

Furthermore, the hypnogram (Fig. 3c and d) demonstrates a "typical" time course of sleep/wake states. As expected, most bad and rejected temporal intervals are associated with wake intervals where locomotion and artifacts occur more readily (Fig. 3d, marked as vertical white bars). The PSD after the preprocessing (Fig. 3f), compared to before (Fig. 3e) shows expected signatures of vigilance states. Specifically, the slow-wave activity (SWA, power < 4Hz) is maximal in N3 sleep, lower in N2 sleep, lower in REM sleep, and lowest in wakefulness; showing that sigma (spindle) activity (12–15 Hz) is dominant in N2 and N3 sleep; highlighting alpha (8–12 Hz) peak in wakefulness, and diffuse theta (4–8 Hz) activity in REM sleep. Note that some of these features do not appear clearly before preprocessing (Fig. 3e, left). Fig. 3 also present a dashboard from an older adult participant (bottom, orange box, panels g–l), illustrating age-related differences in sleep architecture and EEG characteristics. Compared to the representative young adult dataset, the older adult's data shows more frequent transitions between sleep stages and increased wake after sleep onset (WASO, Fig. 3i and j). Additionally, there is a noticeable reduction in frontal delta power during N3 sleep (Fig. 3h–l). Theta power during REM sleep is also weaker compared to the younger participant. Despite these differences, characteristic sleep-stage patterns are still evident: alpha activity (8–12 Hz) dominates during wakefulness although here it extends also to lower frequencies



(caption on next page)

Fig. 7. Comparison of spectral power topographies between EEGLAB and SleepEEGpy Topographic maps of EEG power in canonical frequency bands across sleep stages, generated using EEGLAB (left column) and SleepEEGpy (right column), using the same nap dataset. Rows correspond to specific combinations of sleep stage and frequency band: alpha (8–12 Hz) in wake, sigma (12–15 Hz) in N2, delta (0.5–4 Hz) in N3, and theta (4–8 Hz) in REM. All maps were computed after identical preprocessing steps.

below 8Hz (Fig. 3l), sigma (spindle) activity (12–15 Hz) appears during N2 and N3 sleep, and theta activity (4–8 Hz) is present during REM sleep, albeit with lower amplitude.

Thus, by comparing spectrograms and PSDs before and after cleaning, the user can effectively form an initial impression of the data quality and the effectiveness of the cleaning process (and whether additional iterations may be needed). Overall, the "dashboard" provides a visual summary of a specific dataset, its cleaning/preprocessing, and markers attesting to its quality, which constitute a useful first step for the investigator before proceeding to detailed analysis.

3.2. Section B, analysis: the event-based and spectral-based tools

3.2.1. B1: Sleep spindle detection

To illustrate the event analysis tools (B1), we applied the YASA-based spindle detection in the N2 sleep of our representative dataset. Spindle detection provides an event-based analysis tool for characterizing sleep microstructure, allowing researchers to quantify spindle distribution and topography.

We used the default parameters of YASA, which include a 12–15 Hz spindle frequency range, 1–30 Hz broadband range, spindle duration between 0.5 and 2 s, and 500 ms as the minimal time interval for detecting two distinct spindles. The detection thresholds for a single spindle event were 0.2 relative power, 0.65 moving correlation, and 1.5 STDs above the mean of a moving root-mean-square of the sigma-filtered signal. The signal was re-referenced to a common average reference. With these default parameters, we detected 48,057 spindles across all EEG channels, corresponding to an average of ~187 spindles per channel. Given that the duration of N2 sleep in this dataset was 193 minutes, this reflects a detection rate of ~0.97 spindles/minute. This relatively low rate is reasonable, given that all 257 channels were included. Since spindles are mostly detected over midline scalp electrodes and some electrodes, such as lateral or facial electrodes, only have marginal detections, their inclusion is bound to lower the average rate. The average time course of the detected spindles aligned at the peak is shown in Fig. 4a. Fig. 4b depicts the average time-frequency decomposition (spectrogram) representation of the Pz (E101) channel, showing a slight decrease in spindle frequency from beginning to end. Fig. 4c shows the topographical signatures of different spindle characteristics (left, amplitude; middle, frequency; right, duration), revealing established phenomena such as the prevalence of slower (<13Hz) spindles in frontal electrodes vs. fast (>13 Hz) spindles over centroparietal electrodes.

3.2.2. B2: Spectral tool

We performed MNE-based spectral analysis for the entire dataset separately for each sleep stage and frequency band (slow/delta, theta, alpha, sigma, beta; see Methods). First, we reviewed and edited the default MNE parameters to set FFT and hamming window length to 2048 samples and window overlap to 1024 samples.

The resulting topographical distributions of PSDs per frequency band per sleep stage averaged over all 40 subjects are shown in Fig. 5 and Table 2. In addition to typical activity signatures (described in Fig. 3b above), additional data features can be viewed and assessed here. For example, high-frequency beta activity is maximal during wakefulness; by contrast, delta activity during wakefulness shows hotspots around orbital electrodes due to saccades "injecting" power into this frequency range.

Finally, we display the average PSD plot as a function of the sleep stage across the entire dataset (N = 40) and apply parameterization to the PSDs (Fig. 6). As observed in the representative dataset shown in the "dashboard" (Fig. 3f), this plot reveals expected signatures of vigilance states such as SWA gradient N3>N2, diffuse theta activity in N1 and REM sleep, alpha activity in wakefulness, and maximal high-frequency (>20 Hz) activity in wakefulness. In addition, a peak in sigma activity was observed in the average PSD in N2/N3 sleep. This can also be demonstrated by identifying the periodic component's peak frequency (~13.5 Hz) using SpecParam (Fig. 6b, rightmost vertical green line).

Comparison with EEGLAB. To assess the consistency of spectral outputs across toolkits, we visually compared the topographic distributions of spectral power generated by SleepEEGpy with those obtained using EEGLAB. Fig. 7 shows that the spatial patterns of EEG power in alpha (wake), sigma (N2), delta (N3), and theta (REM sleep) bands are highly similar between the two platforms. For example, both toolkits revealed occipital alpha activity during wakefulness, centrofrontal sigma and delta activity in N2 and N3, and widespread theta power in REM sleep. Note that differences in absolute power scaling between EEGLAB and SleepEEGpy likely reflect their respective approaches to averaging across variable-length time segments: SleepEEGpy uses a weighted average based on segment duration, while EEGLAB applies equal averaging.

4. Discussion

SleepEEGpy is an accessible and user-friendly tool for beginners in sleep EEG data analysis. It offers a comprehensive and user-friendly solution to support sleep EEG research from start to end by providing tools that enable preprocessing, analysis, and visualization of sleep EEG data. By leveraging the MNE-Python library and incorporating features from YASA, PyPREP, and SpecParam (formerly FOOOF), it combines the advantages of general-purpose tools with those of specialized tools. Researchers can benefit from various functionalities, including artifact removal, ICA, event detection, and spectral analyses.

4.1. Simplifying and standardizing sleep EEG analysis

Developed as a streamlined introduction, SleepEEGpy facilitates the learning curve for students and newcomers by unifying essential functionalities. New users should be able to run a standard pipeline without having to worry about the compatibility of data structures, coding

Table 2
EEG power across sleep stages. The first three columns represent the mean ± standard deviation of EEG power (μV^2), averaged across three electrodes for each region: frontal (E21, E22, E14), central (E9, E81, E186), and occipital (E118, E126, E127), calculated from all 40 subjects. The last three columns show the p-values from independent t-tests comparing power between sleep stages.

Measure	Wake	NREM	REM	p(Wake vs NREM)	p(Wake vs REM)	p (NREM vs REM)
Occipital Alpha	12.67 ± 12.45	1.95 ± 1.24	1.86 ± 1.27	8.84e-07***	7.44e-07***	0.757
Frontal Delta	10.58 ± 9.00	67.53 ± 30.84	10.81 ± 5.53	1.62e-17***	0.893	5.95e-18***
Central Sigma	0.86 ± 0.68	1.95 ± 1.24	0.55 ± 0.20	1.64e-10***	0.006**	6.78e-16***
Frontal Theta	3.69 ± 2.56	67.53 ± 30.84	3.46 ± 1.55	4.02e-05***	0.631	3.88e-07**

errors, and arbitrary parameter definitions. Therefore, they can focus on understanding the general high-level steps of analyzing sleep data. Furthermore, SleepEEGpy can be used for learning and teaching specific steps. For example, the dashboard's standardized visualization makes it easy and fast to assess the quality of the preprocessing. Hence, it can function as a useful tool to learn and improve manual preprocessing steps such as annotating bad epochs and bad electrodes.

4.2. A robust and flexible solution for sleep EEG exploration

We demonstrated a typical workflow of SleepEEGpy with continuous hd-EEG data from two healthy participants. With the dashboard (Fig. 3), we summarized and assessed the quality of the preprocessing. The minimal preprocessing included resampling, bandpass filtering, electrode interpolation, and epoch rejection. Finally, an ICA was applied to regress out the components related to electrocardiography/heartbeat. The processed data revealed typical EEG patterns for each vigilance state, which are consistent with prior literature. For example, the topographic distribution of power spectral density (PSD) for each vigilance state (wake, N2, N3, REM) mirrors established findings, such as the dominance of occipital alpha rhythm (8–12 Hz) during wakefulness, central sigma activity (12–15 Hz) in N2 sleep, frontal slow-wave activity (<4 Hz) in N3, and theta activity (4–8 Hz) during REM sleep. These results validate the preprocessing pipeline's effectiveness and its alignment with existing EEG research on sleep [63–65]. To showcase the flexibility of SleepEEGpy, we applied the same workflow to a dataset from an older adult. This participant displayed more frequent transitions between sleep stages, increased WASO, reduced slow-wave sleep, fragmented REM, and weaker frontal delta and theta power, consistent with age-related changes in sleep [66,67]. SleepEEGpy effectively handled both young and older adult datasets, demonstrating its robustness and versatility for sleep research and clinical applications across diverse populations. We further illustrated that with SleepEEGpy, it is possible to reliably analyze events during sleep by detecting sleep spindles (Fig. 4). Specifically, we demonstrated its ability to detect spindles with typical frequency and topographical patterns. Slower spindles (<13 Hz) were observed in frontal regions, while faster spindles (>13 Hz) were found in centroparietal regions, which is consistent with findings in existing literature [54,68,69]. In addition, the decrease in spindle frequency during towards the end of the event aligns with previous findings [54]. Next, we presented group averages of the topographical distribution of PSD per frequency band and sleep stage (Fig. 5, Table 2), highlighting typical specific vigilance-state signatures activity patterns. Finally, we separately inspected the average PSD for each vigilance state, averaged across our participant cohort (Fig. 6), revealing clear peaks in activity associated with specific sleep stages, such as the characteristic sigma peak in N2/N3 sleep (~13.5 Hz). To further demonstrate that SleepEEGpy produces reliable outputs consistent with existing tools, we compared its output scalp topographies to those computed with EEGLAB using the same sleep EEG dataset. The spectral maps of key frequency bands across vigilance states (e.g., alpha in wake, sigma in N2) were nearly identical across both platforms (Fig. 7), supporting the validity of SleepEEGpy's spectral pipeline. These results underscore the tool's ability to capture nuanced spectral features across different vigilance states and sleep stages, contributing valuable insights into sleep dynamics. In summary, SleepEEGpy provides an easy way to perform general analysis and visualization of raw sleep EEG data, as well as overviews of group averages.

4.3. SleepEEGpy as an integrated solution beyond functionality of existing tools

The main advantages of SleepEEGpy relative to other available software tools are its simplicity and all-in-one functionality. Current EEG software packages are either implemented in MATLAB and behind a paywall [28,29,34–36,38,39], optimized for sleep EEG but restricted to either preprocessing [48], sleep scoring [70], or specific analyses [44], or based on Python but not necessarily optimized for sleep research [30]. This often results in the need for researchers to combine multiple software environments to work with sleep EEG data, which can complicate the workflow. Thus, SleepEEGpy helps to address an unmet need by providing a comprehensive package that goes beyond the typical configuration in many labs and combines multiple software environments to work with sleep EEG data. Its free open-source nature ensures accessibility to students and sleep research labs. To complement the discussion, an explicit comparison of core features across popular EEG software tools is provided in Table 1. This overview highlights SleepEEGpy's unique balance of automation, scalability, and ease of use, particularly its support for sleep-specific preprocessing, event detection, and visualization in a single streamlined pipeline. Beyond traditional sleep research, SleepEEGpy also holds potential for interdisciplinary applications. Because it's designed to work end-to-end, from raw data to high quality figures, it can be easily adapted to other domains such as cognitive neuroscience, psychiatry, and neuroengineering - anywhere where sleep-related dynamics and EEG are relevant. For example, it could support clinical studies investigating sleep disruptions in psychiatric populations, research on cognitive performance following sleep interventions, or experiments using wearable EEG devices in real-world settings. By lowering the entry barrier for high-quality EEG analysis, SleepEEGpy opens up opportunities for collaborations between computational scientists, clinicians, educators, and behavioral researchers.

In terms of features, SleepEEGpy enables semi-automated processing, offering an end-to-end pipeline for sleep EEG analysis. After preprocessing, the pipeline provides a standardized dashboard for dataset validation before proceeding to further analysis. Moreover, built-in spectral analysis tools allow users to examine data across multiple patients in a single workflow, enhancing its applicability for large-scale studies. Regarding usability, SleepEEGpy is fully open-source and freely available, ensuring accessibility to researchers at all levels. It offers a complete sleep EEG processing pipeline, allowing beginners to perform analyses with minimal setup using default parameters (e.g., filter type or spindle detection threshold). At the same time, the framework is flexible, enabling experienced users to adjust settings or integrate additional functionality as needed. By addressing these aspects, SleepEEGpy offers a robust solution that simplifies sleep EEG analysis while maintaining flexibility for advanced research.

In terms of performance, SleepEEGpy's processing speed and efficiency are comparable to existing Python-based tools, such as YASA and MNE, since it builds on established libraries. Unlike many existing tools that rely on pre-segmented epochs, the pipeline processes raw continuous EEG data throughout. This approach, while computationally more demanding, ensures a more consistent preprocessing workflow and avoids artificial interruptions caused by segmentation. However, actual performance may vary depending on the hardware used, especially with high-density, long-duration sleep EEG recordings.

4.4. Ease of use

Getting started with SleepEEGpy requires only basic knowledge of Python syntax and Jupyter notebooks. The pipeline is based primarily on classes and their methods, and the most complex task an average user might encounter is writing a 'for' loop to optimize their pipeline. Jupyter notebooks are implemented for each stage of the pipeline, making the tools nearly automatic with embedded explanations at each step. The code repository and notebooks are available at https://github.com/NirLab-TAU/sleepeegpy, where a dedicated notebook (the "complete pipeline" notebook) enables users to download the example datasets and replicate our results. SleepEEGpy is also accompanied by a webpage providing detailed API documentation and published notebooks. The documentation is built using Sphinx [71] and hosted on GitHub. Example datasets, including a full-night and a nap recording from young

adults and a full-night recording from an older adult, can be found at Zenodo [72]: 10.5281/zenodo.10362190. Additionally, the "complete pipeline" notebook provides an end-to-end example of dataset retrieval, preprocessing, and analysis.

4.5. Limitations and future directions

SleepEEGpy could be improved by including additional analysis methods, for example, a module for statistical analysis (parametric and non-parametric tests) or a module for source estimation, which would broaden the scope of its functionality for advanced users. While the current version supports various sleep data types, its data type compatibility could also be further extended, particularly with the EEG-BIDS data structure [73], to better accommodate the growing adoption of this standardized format. We believe that the code availability and free software licenses (SleepEEGpy is released under the MIT license) allow the community to rapidly expand such functionalities.

Additionally, SleepEEGpy has the potential to be extended with more functionality based on machine learning and deep learning algorithms for prediction and classification tasks, such as identifying pathological events during sleep [74] or discovering biomarkers for neuropsychiatric disorders [75]. Recent advancements in these fields, including applications in medical imaging, disease diagnosis, and physiological signal analysis [76–81], highlight their potential for EEG research. In line with these advancements, deep learning and complex network approaches have shown promise in EEG-based brain state classification and neurological research [82,83], while statistical feature selection techniques improve EEG-based classification accuracy [84]. Additionally, advanced simulation-based frameworks, such as Monte Carlo methods [85–88], could be adapted to further refine EEG source modeling or artifact detection in SleepEEGpy. By integrating these techniques, SleepEEGpy could enhance sleep EEG analysis, providing more accurate and personalized insights into sleep dynamics and neuropsychiatric disorders.

An additional limitation is the required knowledge of MNE and YASA for more advanced analyses. At present, if the analysis prompts for an adjustment of a meta-parameter (e.g., spindle detection threshold or specific bandpass filtering), the details are best described in the documentation of MNE or YASA itself, which might be challenging for new users. However, for a new user, the defaults should be sufficient to get to know the processing of sleep EEG. This represents a trade-off between user-friendliness and the flexibility offered by specialized tools. All these limitations could be addressed with additional code and documentation. Hence, we hope that this package will be further developed by the growing community of sleep investigators committed to open science and high-quality open-source software. With the increased use of Python as the preferred programming language and its interface with machine-learning tools, we envision SleepEEGpy as an ideal entry point to become familiar with sleep EEG analysis.

5. Conclusion

SleepEEGpy provides a user-friendly and comprehensive framework for sleep EEG analysis, integrating preprocessing, event detection, and spectral analysis in a single pipeline. By leveraging existing Python-based tools, it streamlines workflows for both novice and experienced researchers, ensuring accessibility and flexibility. Our results demonstrate its ability to effectively process and analyze sleep EEG data across different age groups, capturing key sleep dynamics and spectral features consistent with prior literature. As an open-source package, SleepEEGpy addresses the need for an all-in-one sleep EEG tool, with the potential for further expansion through community contributions.

CRediT authorship contribution statement

Rotem Falach: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Gennadiy Belonosov:** Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis. **Flavio Jean Schmidig:** Writing – review & editing, Writing – original draft, Validation, Methodology. **Maya Aderka:** Validation. **Vladislav Zhelezniakov:** Validation, Methodology. **Revital Shani-Hershkovich:** Validation, Methodology. **Ella Bar:** Data curation. **Yuval Nir:** Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization.

Ethical statement

1) This material is the authors' own original work, which has not been previously published elsewhere.
2) The paper is not currently being considered for publication elsewhere.
3) The paper reflects the authors' own research and analysis in a truthful and complete manner.
4) The paper properly credits the meaningful contributions of co-authors and co-researchers.
5) The results are appropriately placed in the context of prior and existing research.
6) All sources used are properly disclosed (correct citation). Literally copying of text must be indicated as such by using quotation marks and giving proper reference.
7) All authors have been personally and actively involved in substantial work leading to the paper, and will take public responsibility for its content.

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

We thank Gal Zatelman, May Eliyahu, Tomer Cohen, Hadar Nakar, and Shir Frank for their assistance with data collection and sleep scoring; Yarden Mezi for stabilizing the new version of the tool; and Dr. Noa Bar-Ilan Regev for administrative assistance. This study was supported by ERC-2019-CoG 864353 and a grant from the Aufzien Family Center for the Prevention and Treatment of Parkinson's Disease (Y.N.).

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.compbiomed.2025.110232.

References

[1] M.H. Kryger, T. Roth, Principles and Practice of Sleep Medicine, sixth ed., Elsevier - Health Sciences Division, 2016.
[2] Y. Nir, L. de Lecea, Sleep and vigilance states: embracing spatiotemporal dynamics, Neuron 111 (13) (2023) 1998–2011, https://doi.org/10.1016/j.neuron.2023.04.012.
[3] R.B. Berry, R. Brooks, C. Gamaldo, S.M. Harding, R.M. Lloyd, S.F. Quan, M. T. Troester, B.V. Vaughn, AASM scoring manual updates for 2017 (version 2.4), J. Clin. Sleep Med. 13 (5) (2017) 665–666, https://doi.org/10.5664/jcsm.6576.
[4] L. Fiorillo, A. Puiatti, M. Papandrea, P.-L. Ratti, P. Favaro, C. Roth, P. Bargiotas, C. L. Bassetti, F.D. Faraci, Automated sleep scoring: a review of the latest approaches, Sleep Med. Rev. 48 (2019) 101204, https://doi.org/10.1016/j.smrv.2019.07.007.
[5] M. Gaiduk, Á. Serrano Alarcón, R. Seepold, N. Martínez Madrid, Current status and prospects of automatic sleep stages scoring: review, Biomed. Eng. Lett. 13 (3) (2023) 247–272, https://doi.org/10.1007/s13534-023-00299-3.
[6] E. Urtnasan, J.-U. Park, E.Y. Joo, K.-J. Lee, Deep convolutional recurrent model for automatic scoring sleep stages based on single-lead ECG signal, Diagnostics 12 (5) (2022) 1235, https://doi.org/10.3390/diagnostics12051235.

[7] R. Cox, J. Fell, Analyzing human sleep EEG: a methodological primer with code implementation, Sleep Med. Rev. 54 (2020) 101353, https://doi.org/10.1016/j.smrv.2020.101353.

[8] S. Gais, M. Mölle, K. Helms, J. Born, Learning-dependent increases in sleep spindle density, J. Neurosci. 22 (15) (2002) 6830–6834, https://doi.org/10.1523/JNEUROSCI.22-15-06830.2002.

[9] M. Geva-Sagiv, E.A. Mankin, D. Eliashiv, S. Epstein, N. Cherry, G. Kalender, N. Tchemodanov, Y. Nir, I. Fried, Augmenting hippocampal–prefrontal neuronal synchrony during sleep enhances memory consolidation in humans, Nat. Neurosci. 26 (6) (2023), https://doi.org/10.1038/s41593-023-01324-5.

[10] M. Geva-Sagiv, Y. Nir, Local sleep oscillations: implications for memory consolidation, Front. Neurosci. 13 (2019) 813, https://doi.org/10.3389/fnins.2019.00813.

[11] L. Marshall, J. Born, The contribution of sleep to hippocampus-dependent memory consolidation, Trends Cognit. Sci. 11 (10) (2007) 442–450, https://doi.org/10.1016/j.tics.2007.09.001.

[12] M. Schabus, G. Gruber, S. Parapatics, C. Sauter, G. Klösch, P. Anderer, W. Klimesch, B. Saletu, J. Zeitlhofer, Sleep spindles and their significance for declarative memory consolidation, Sleep 27 (8) (2004) 1479–1485, https://doi.org/10.1093/sleep/27.7.1479.

[13] F.J. Schmidig, M. Geva-Sagiv, R. Falach, S. Yakim, Y. Gat, O. Sharon, I. Fried, Y. Nir, A visual paired associate learning (vPAL) paradigm to study memory consolidation during sleep, J. Sleep Res. 33 (5) (2024) e14151, https://doi.org/10.1111/jsr.14151.

[14] S. Kurth, M. Ringli, A. Geiger, M. LeBourgeois, O.G. Jenni, R. Huber, Mapping of cortical activity in the first two decades of life: a high-density sleep electroencephalogram study, J. Neurosci. 30 (40) (2010) 13211–13219, https://doi.org/10.1523/JNEUROSCI.2532-10.2010.

[15] R.F. Helfrich, B.A. Mander, W.J. Jagust, R.T. Knight, M.P. Walker, Old brains come uncoupled in sleep: slow wave-spindle synchrony, brain atrophy, and forgetting, Neuron 97 (1) (2018) 221–230.e4, https://doi.org/10.1016/j.neuron.2017.11.020.

[16] F. Siclari, B. Baird, L. Perogamvros, G. Bernardi, J.J. LaRocque, B. Riedner, M. Boly, B.R. Postle, G. Tononi, The neural correlates of dreaming, Nat. Neurosci. 20 (6) (2017), https://doi.org/10.1038/nn.4545.

[17] B.A. Mander, S.M. Marks, J.W. Vogel, V. Rao, B. Lu, J.M. Saletin, S. Ancoli-Israel, W.J. Jagust, M.P. Walker, β-amyloid disrupts human NREM slow waves and related hippocampus-dependent memory consolidation, Nat. Neurosci. 18 (7) (2015), https://doi.org/10.1038/nn.4035.

[18] F. Ferrarelli, Sleep spindles as neurophysiological biomarkers of schizophrenia, Eur. J. Neurosci. 59 (8) (2023) 1907–1917, https://doi.org/10.1111/ejn.16178.

[19] S.T. Aung, Y. Wongsawat, Modified-distribution entropy as the features for the detection of epileptic seizures, Front. Physiol. 11 (607) (2020), https://doi.org/10.3389/fphys.2020.00607.

[20] F. Edderbali, M. Harmouchi, E. Essoukaki, Transfer learning for epilepsy detection using spectrogram images, IAES Int. J. Artif. Intell. 13 (1) (2024) 1022, https://doi.org/10.11591/ijai.v13.i1.pp1022-1029.

[21] D.M. Tucker, A.C. Waters, M.D. Holmes, Transition from cortical slow oscillations of sleep to spike-wave seizures, Clin. Neurophysiol. 120 (12) (2009) 2055–2062, https://doi.org/10.1016/j.clinph.2009.07.047.

[22] İ. Kaya, A brief summary of EEG artifact handling, in: Brain-Computer Interface, IntechOpen, 2021, https://doi.org/10.5772/intechopen.99127.

[23] M.X. Cohen, Analyzing Neural Time Series Data, MIT Press, 2014, https://doi.org/10.7551/mitpress/9609.001.0001.

[24] A. Delorme, EEG is better left alone, Sci. Rep. 13 (1) (2023), https://doi.org/10.1038/s41598-023-27528-0.

[25] B.A. Riedner, V.V. Vyazovskiy, R. Huber, M. Massimini, S. Esser, M. Murphy, G. Tononi, Sleep homeostasis and cortical synchronization: III. A high-density EEG study of sleep slow waves in humans, Sleep 30 (12) (2007) 1643–1657, https://doi.org/10.1093/sleep/30.12.1643.

[26] R.K. Das, A. Martin, T. Zurales, D. Dowling, A. Khan, A survey on EEG data analysis software, Science 5 (2) (2023), https://doi.org/10.3390/sci5020023.

[27] F. Tadel, S. Baillet, J.C. Mosher, D. Pantazis, R.M. Leahy, Brainstorm: a user-friendly application for MEG/EEG analysis, Comput. Intell. Neurosci. 2011 (2011) e879716, https://doi.org/10.1155/2011/879716.

[28] A. Delorme, S. Makeig, EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis, J. Neurosci. Methods 134 (1) (2004) 9–21, https://doi.org/10.1016/j.jneumeth.2003.10.009.

[29] R. Oostenveld, P. Fries, E. Maris, J.-M. Schoffelen, FieldTrip: open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data, 2010, Comput. Intell. Neurosci. (2011) e156869, https://doi.org/10.1155/2011/156869.

[30] A. Gramfort, MEG and EEG data analysis with MNE-Python, Front. Neurosci. 7 (2013), https://doi.org/10.3389/fnins.2013.00267.

[31] S. Huberty, J. Desjardins, T. Collins, M. Elsabbagh, C. O'Reilly, PyLossless: a non-destructive EEG processing pipeline, bioRxiv (2024), https://doi.org/10.1101/2024.01.12.575323.

[32] N.W. Bailey, M. Biabani, A.T. Hill, A. Miljevic, N.C. Rogasch, B. McQueen, O. W. Murphy, P.B. Fitzgerald, Introducing RELAX: an automated pre-processing pipeline for cleaning EEG data - Part 1: algorithm and application to oscillations, Clin. Neurophysiol. 149 (2023) 178–201, https://doi.org/10.1016/j.clinph.2023.01.017.

[33] N. Bigdely-Shamlo, T. Mullen, C. Kothe, K.-M. Su, K.A. Robbins, The PREP pipeline: standardized preprocessing for large-scale EEG analysis, Front. Neuroinf. 9 (2015) 16, https://doi.org/10.3389/fninf.2015.00016.

[34] S. Blum, N.S.J. Jacobsen, M.G. Bleichner, S. Debener, A riemannian modification of artifact subspace reconstruction for EEG artifact handling, Front. Hum. Neurosci. 13 (2019) 141, https://doi.org/10.3389/fnhum.2019.00141.

[35] J.R. da Cruz, V. Chicherov, M.H. Herzog, P. Figueiredo, An automatic pre-processing pipeline for EEG analysis (APP) based on robust statistics, Clin. Neurophysiol. 129 (7) (2018) 1427–1437, https://doi.org/10.1016/j.clinph.2018.04.600.

[36] L.J. Gabard-Durnam, A.S. Mendez Leal, C.L. Wilkinson, A.R. Levin, The harvard automated processing pipeline for electroencephalography (HAPPE): standardized processing software for developmental and high-artifact data, Front. Neurosci. 12 (2018), https://doi.org/10.3389/fnins.2018.00097.

[37] M. Jas, D.A. Engemann, Y. Bekhti, F. Raimondo, A. Gramfort, Autoreject: automated artifact rejection for MEG and EEG data, Neuroimage 159 (2017) 417–429, https://doi.org/10.1016/j.neuroimage.2017.06.030.

[38] A. Mognon, J. Jovicich, L. Bruzzone, M. Buiatti, ADJUST: an automatic EEG artifact detector based on the joint use of spatial and temporal features, Psychophysiology 48 (2) (2011) 229–240, https://doi.org/10.1111/j.1469-8986.2010.01061.x.

[39] H. Nolan, R. Whelan, R.B. Reilly, FASTER: fully automated statistical thresholding for EEG artifact rejection, J. Neurosci. Methods 192 (1) (2010) 152–162, https://doi.org/10.1016/j.jneumeth.2010.07.015.

[40] R. Somervail, J. Cataldi, A.M. Stephan, F. Siclari, G.D. Iannetti, Dusk2Dawn: an EEGLAB plugin for automatic cleaning of whole-night sleep electroencephalogram using Artifact Subspace Reconstruction, Sleep (2023), https://doi.org/10.1093/sleep/zsad208.

[41] D.C. 't Wallant, V. Muto, G. Gaggioni, M. Jaspar, S.L. Chellappa, C. Meyer, G. Vandewalle, P. Maquet, C. Phillips, Automatic artifacts and arousals detection in whole-night sleep EEG recordings, J. Neurosci. Methods 258 (2016) 124–133, https://doi.org/10.1016/j.jneumeth.2015.11.005.

[42] P. Anderer, S. Roberts, A. Schlögl, G. Gruber, G. Klösch, W. Herrmann, P. Rappelsberger, O. Filz, M.J. Barbanoj, G. Dorffner, B. Saletu, Artifact processing in computerized analysis of sleep EEG – a review, Neuropsychobiology 40 (3) (1999) 150–157, https://doi.org/10.1159/000026613.

[43] J.A. Desjardins, S. van Noordt, S. Huberty, S.J. Segalowitz, M. Elsabbagh, EEG Integrated Platform Lossless (EEG-IP-L) pre-processing pipeline for objective signal quality assessment incorporating data annotation and blind source separation, J. Neurosci. Methods 347 (2021) 108961, https://doi.org/10.1016/j.jneumeth.2020.108961.

[44] R. Vallat, M.P. Walker, An open-source, high-performance tool for automated sleep staging, Elife 10 (e70092) (2021), https://doi.org/10.7554/eLife.70092.

[45] T. Donoghue, M. Haller, E.J. Peterson, P. Varma, P. Sebastian, R. Gao, T. Noto, A. H. Lara, J.D. Wallis, R.T. Knight, A. Shestyuk, B. Voytek, Parameterizing neural power spectra into periodic and aperiodic components, Nat. Neurosci. 23 (2020) 1655–1665, https://doi.org/10.1038/s41593-020-00744-x.

[46] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, Jupyter development team, in: F. Loizides, B. Scmidt (Eds.), Jupyter Notebooks – a Publishing Format for Reproducible Computational Workflows, IOS Press, 2016, pp. 87–90, https://doi.org/10.3233/978-1-61499-649-1-87.

[47] T. Andrillon, Y. Nir, C. Cirelli, G. Tononi, I. Fried, Single-neuron activity and eye movements during human REM sleep and awake vision, Nat. Commun. 6 (7884) (2015), https://doi.org/10.1038/ncomms8884.

[48] S. Leach, G. Sousouri, R. Huber, 'High-Density-SleepCleaner': an open-source, semi-automatic artifact removal routine tailored to high-density sleep EEG, J. Neurosci. Methods 391 (2023) 109849, https://doi.org/10.1016/j.jneumeth.2023.109849.

[49] A. Hyvarinen, Fast and robust fixed-point algorithms for independent component analysis, IEEE Trans. Neural Network. 10 (3) (1999) 626–634, https://doi.org/10.1109/72.761722.

[50] T.-W. Lee, M. Girolami, T.J. Sejnowski, Independent component analysis using an extended Infomax algorithm for mixed subgaussian and supergaussian sources, Neural Comput. 11 (2) (1999) 417–441, https://doi.org/10.1162/089976699300016719.

[51] P. Ablin, J.-F. Cardoso, A. Gramfort, Faster independent component analysis by preconditioning with hessian approximations, IEEE Trans. Signal Process. 66 (15) (2018) 4040–4049, https://doi.org/10.1109/TSP.2018.2844203.

[52] M. Hansson-Sandsten, Optimal multitaper wigner spectrum estimation of a class of locally stationary processes using hermite functions, EURASIP J. Appl. Signal Process. (2011) 1–15, https://doi.org/10.1155/2011/980805.

[53] K. Lacourse, J. Delfrate, J. Beaudry, P. Peppard, S.C. Warby, A sleep spindle detection algorithm that emulates human expert spindle scoring, J. Neurosci. Methods 316 (2019) 3–11, https://doi.org/10.1016/j.jneumeth.2018.08.014.

[54] T. Andrillon, Y. Nir, R.J. Staba, F. Ferrarelli, C. Cirelli, G. Tononi, I. Fried, Sleep spindles in humans: insights from intracranial EEG and unit recordings, J. Neurosci. 31 (49) (2011) 17821–17834, https://doi.org/10.1523/JNEUROSCI.2604-11.2011.

[55] S.M. Purcell, D.S. Manoach, C. Demanuele, B.E. Cade, S. Mariani, R. Cox, G. Panagiotaropoulou, R. Saxena, J.Q. Pan, J.W. Smoller, S. Redline, R. Stickgold, Characterizing sleep spindles in 11,630 individuals from the national sleep research resource, Nat. Commun. 8 (1) (2017), https://doi.org/10.1038/ncomms15930.

[56] M. Massimini, R. Huber, F. Ferrarelli, S. Hill, G. Tononi, The sleep slow oscillation as a traveling wave, J. Neurosci. 24 (31) (2004) 6862–6870, https://doi.org/10.1523/JNEUROSCI.1318-04.2004.

[57] Y. Nir, R.J. Staba, T. Andrillon, V.V. Vyazovskiy, C. Cirelli, I. Fried, G. Tononi, Regional slow waves and spindles in human sleep, Neuron 70 (1) (2011) 153–169, https://doi.org/10.1016/j.neuron.2011.02.043.
[58] C. Tallon-Baudry, O. Bertrand, C. Delpuech, J. Pernier, Oscillatory γ-band (30–70 Hz) activity induced by a visual search task in humans, J. Neurosci. 17 (2) (1997) 722–734, https://doi.org/10.1523/JNEUROSCI.17-02-00722.1997.
[59] D. Slepian, Prolate spheroidal wave functions, fourier analysis, and uncertainty-V: the discrete case, Bell Sys. Tech. J. 57 (5) (1978) 1371–1430, https://doi.org/10.1002/j.1538-7305.1978.tb02104.x.
[60] R. Agarwal, T. Takeuchi, S. Laroche, J. Gotman, Detection of rapid-eye movements in sleep studies, IEEE (Inst. Electr. Electron. Eng.) Trans. Biomed. Eng. 52 (8) (2005) 1390–1396, https://doi.org/10.1109/TBME.2005.851512.
[61] P. Welch, The use of fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms, IEEE Trans. Audio Electroacoust. 15 (2) (1967) 70–73, https://doi.org/10.1109/TAU.1967.1161901.
[62] F. Perrin, J. Pernier, O. Bertrand, J.F. Echallier, Spherical splines for scalp potential and current density mapping, Electroencephalogr. Clin. Neurophysiol. 72 (2) (1989) 184–187, https://doi.org/10.1016/0013-4694(89)90180-6.
[63] J. Zeitlhofer, P. Anderer, S. Obergottsberger, P. Schimicek, S. Lurger, E. Marschnigg, B. Saletu, L. Deecke, Topographic mapping of EEG during sleep, Brain Topogr. 6 (2) (1993) 123–129, https://doi.org/10.1007/BF01191077.
[64] G. Tinguely, L.A. Finelli, H.-P. Landolt, A.A. Borbély, P. Achermann, Functional EEG topography in sleep and waking: state-dependent and state-independent features, Neuroimage 32 (1) (2006) 283–292, https://doi.org/10.1016/j.neuroimage.2006.03.017.
[65] B. Weiss, Z. Clemens, R. Bódizs, P. Halász, Comparison of fractal and power spectral EEG features: effects of topography and sleep stages, Brain Res. Bull. 84 (6) (2011) 359–375, https://doi.org/10.1016/j.brainresbull.2010.12.005.
[66] H.-P. Landolt, A.A. Borbély, Age-dependent changes in sleep EEG topography, Clin. Neurophysiol. 112 (2) (2001) 369–377, https://doi.org/10.1016/S1388-2457(00)00542-3.
[67] B.A. Mander, J.R. Winer, M.P. Walker, Sleep and human aging, Neuron 94 (1) (2017) 19–36, https://doi.org/10.1016/j.neuron.2017.02.004.
[68] R. Cox, A.C. Schapiro, D.S. Manoach, R. Stickgold, Individual differences in frequency and topography of slow and fast sleep spindles, Front. Hum. Neurosci. 11 (2017), https://doi.org/10.3389/fnhum.2017.00433.
[69] J. Zeitlhofer, G. Gruber, P. Anderer, S. Asenbaum, P. Schimicek, B. Saletu, Topographic distribution of sleep spindles in young healthy subjects, J. Sleep Res. 6 (3) (1997) 149–155, https://doi.org/10.1046/j.1365-2869.1997.00046.x.
[70] E. Combrisson, R. Vallat, J.-B. Eichenlaub, C. O'Reilly, T. Lajnef, A. Guillot, P. M. Ruby, K. Jerbi, Sleep: an open-source Python software for visualization, analysis, and staging of sleep data, Front. Neuroinf. 11 (2017), https://doi.org/10.3389/fninf.2017.00060.
[71] T. Komiya, G. Brandl, B. Jean-François, T. Shimizukawa, A. Turner, J.L. Andersen, D. Neuhäuser, S. Finucane, R. Lehmann, T. Kampik, J. Magin, Jacobmason, J. Dufresne, J. Waltman, J.L.C. Rodríguez, A. Ronacher, D. Shachnev, Y. Shibukawa, M. Geier, N. Kaneko, *sphinx-doc/sphinx: Sphinx 7.2.6* (Version v7.2.6), Zenodo (2023), https://doi.org/10.5281/ZENODO.7857310 [Computer software].
[72] R. Falach, G. Belonosov, F. Schmidig, M. Aderka, V. Zhelezniakov, R. Shani-Hershkovich, E. Bar, Y. Nir, SleepEEGpy: a Python-based software integration package to organize preprocessing, analysis, and visualization of sleep EEG data, Zenodo (2025), https://doi.org/10.5281/ZENODO.14914456.
[73] C.R. Pernet, S. Appelhoff, K.J. Gorgolewski, G. Flandin, C. Phillips, A. Delorme, R. Oostenveld, EEG-BIDS, an extension to the brain imaging data structure for electroencephalography, Sci. Data 6 (1) (2019), https://doi.org/10.1038/s41597-019-0104-8.
[74] R. Falach, M. Geva-Sagiv, D. Eliashiv, L. Goldstein, O. Budin, G. Gurevitch, G. Morris, I. Strauss, A. Globerson, F. Fahoum, I. Fried, Y. Nir, Annotated interictal discharges in intracranial EEG sleep data and related machine learning detection scheme, Sci. Data 11 (2024) 1354, https://doi.org/10.1038/s41597-024-04187-y.
[75] E. Jeong, Y. Woo Shin, J.-I. Byun, J.-S. Sunwoo, M. Roascio, P. Mattioli, L. Giorgetti, F. Famà, G. Arnulfo, D. Arnaldi, H.-J. Kim, K.-Y. Jung, EEG-based machine learning models for the prediction of phenoconversion time and subtype in isolated rapid eye movement sleep behavior disorder, Sleep 47 (5) (2024) zsae031, https://doi.org/10.1093/sleep/zsae031.
[76] B. Emek Soylu, M.S. Guzel, G.E. Bostanci, F. Ekinci, T. Asuroglu, K. Acici, Deep-learning-based approaches for semantic segmentation of natural scene images: a review, Electronics 12 (12) (2023), https://doi.org/10.3390/electronics12122730.
[77] A. Gupta, A.G. Ravelo-García, F. Morgado-Dias, Recent advancements in deep learning-based remote photoplethysmography methods, in: Data Fusion Techniques and Applications for Smart Healthcare, Elsevier, 2024, pp. 127–155, https://doi.org/10.1016/B978-0-44-313233-9.00012-6.
[78] B. Kalita, N. Deb, D. Das, AnEEG: leveraging deep learning for effective artifact removal in EEG data, Sci. Rep. 14 (1) (2024) 24234, https://doi.org/10.1038/s41598-024-75091-z.
[79] M. Kalkan, M.S. Guzel, F. Ekinci, E. Akcapinar Sezer, T. Asuroglu, Comparative analysis of deep learning methods on CT images for lung cancer specification, Cancers 16 (19) (2024), https://doi.org/10.3390/cancers16193321.
[80] S. Ozsari, E. Kumru, F. Ekinci, I. Akata, M.S. Guzel, K. Acici, E. Ozcan, T. Asuroglu, Deep learning-based classification of macrofungi: comparative analysis of advanced models for accurate fungi identification, Sensors 24 (22) (2024) 7189, https://doi.org/10.3390/s24227189.
[81] X. Zhang, X. Zhang, Q. Huang, Y. Lv, F. Chen, A review of automated sleep stage based on EEG signals, Biocybern. Biomed. Eng. 44 (3) (2024) 651–673, https://doi.org/10.1016/j.bbe.2024.06.004.
[82] Z. Gao, W. Dang, X. Wang, X. Hong, L. Hou, K. Ma, M. Perc, Complex networks and deep learning for EEG signal analysis, Cognit. Neurodynam. 15 (3) (2021) 369–388, https://doi.org/10.1007/s11571-020-09626-1.
[83] P. Ji, J. Ye, Y. Mu, W. Lin, Y. Tian, C. Hens, M. Perc, Y. Tang, J. Sun, J. Kurths, Signal propagation in complex networks, Phys. Rep. 1017 (2023) 1–96, https://doi.org/10.1016/j.physrep.2023.03.005.
[84] M. Degirmenci, Y.K. Yuce, M. Perc, Y. Isler, Statistically significant features improve binary and multiple Motor Imagery task predictions from EEGs, Front. Hum. Neurosci. 17 (2023), https://doi.org/10.3389/fnhum.2023.1223307.
[85] F. Ekinci, K. Acici, T. Asuroglu, Enhancing tissue equivalence in 7Li heavy ion therapy with MC algorithm optimized polymer-based bioinks, J. Funct. Biomater. 14 (12) (2023), https://doi.org/10.3390/jfb14120559.
[86] F. Ekinci, K. Acici, T. Asuroglu, B. Emek Soylu, MC TRIM algorithm in mandibula phantom in helium therapy, Healthcare 11 (18) (2023), https://doi.org/10.3390/healthcare11182523.
[87] F. Ekinci, T. Asuroglu, K. Acici, Monte Carlo simulation of TRIM algorithm in ceramic biomaterial in proton therapy, Materials 16 (13) (2023), https://doi.org/10.3390/ma16134833.
[88] Y. Gokcekuyu, F. Ekinci, A. Buyuksungur, M.S. Guzel, K. Acici, T. Asuroglu, Comparison of X-ray absorption in mandibular tissues and tissue-equivalent polymeric materials using PHITS Monte Carlo simulations, Appl. Sci. 14 (23) (2024), https://doi.org/10.3390/app142310879.