Chapter 6

End-to-End Processing of M/EEG Data with BIDS, HED, and EEGLAB

Dung Truong, Kay Robbins, Arnaud Delorme, and Scott Makeig

Abstract

Reliable and reproducible machine-learning enabled neuroscience research requires large-scale data sharing and analysis. Essential for the effective and efficient analysis of shared datasets are standardized data and metadata organization and formatting, a well-documented, automated analysis pipeline, a comprehensive software framework, and a compute environment that can adequately support the analysis process. In this chapter, we introduce the combined Brain Imaging Data Structure (BIDS) and Hierarchical Event Descriptors (HED) frameworks and illustrate their example use through the organization and time course annotation of a publicly shared EEG (electroencephalography) dataset. We show how the open-source software EEGLAB can operate on data formatted using these standards to perform EEG analysis using a variety of techniques including group-based statistical analysis. Finally, we present a way to exploit freely available high-performance computing resources that allows the application of computationally intensive learning methods to ever larger and more diverse data collections.

Key words EEG, Neuroinformatics, BIDS, HED, EEGLAB

1 Introduction

As demonstrated in a recent study [1], the complex and varied landscape of individual and phenotypic brain differences may require studies of thousands of individuals to establish reliable and reproducible associations between brain dynamics and function, and personal experience and behavior. Machine-learning approaches can now facilitate these discoveries but typically require large collections of *diverse* input training data, including well-labelled data, to create generalizable models. Public data archives that use common data formatting standards are thus critical infrastructure for enabling large-scale neuroimaging data analysis (and "meta=" or "mega-analysis" [*see* Glossary]) applied within and across individual studies and data recordings. Standardized data formats enable tool interoperability [*see* Glossary]. Machine-actionable [*see* Glossary] metadata [*see* Glossary] supports the

Robert Whelan and Hervé Lemaître (eds.), *Methods for Analyzing Large Neuroimaging Datasets*, Neuromethods, vol. 218, https://doi.org/10.1007/978-1-0716-4260-3_6,

interpretation of the results [*see* Chapter 4]. Finally, complete specification and accurate documentation of the applied computational pipelines promote reproducibility and testing.

This chapter presents an end-to-end overview of an electroencephalography (EEG) data analysis process based on open community standards, beginning with the identification of suitable raw data and ending with the presentation of results suitable for discussion and publication. The process begins with the conversion of raw data (here, collected EEG plus sensory and behavioral event descriptions and timings) to standard BIDS (Brain Imaging Data Structure [*see* Glossary]) archival format [2]. The BIDS project, which initially focused on specification of file organization for functional magnetic resonance imaging (fMRI) datasets, is evolving into a widely adopted, community-driven set of data format specifications for a variety of imaging modalities including fMRI, positron emission tomography (PET), diffusion-weighted imaging (DWI), EEG, magnetoencephalography (MEG), intracranial EEG (iEEG), and microscopy data [3–5] [*see* Chapter 4]. Further BIDS specifications are under development for body motion capture, eye-tracking data, and multiple modality data. The in-common BIDS formatting and metadata annotation standards allow tools to be built using standard APIs (application program interfaces) enabling automated ingestion and processing of datasets representing either one or more than one study and experimental paradigm.

While initial BIDS efforts focused on raw data formats and layouts, recent efforts are also underway to standardize derivative datasets (i.e., containing results of computations performed on the raw data) for different imaging modalities including structural and functional fMRI, electrophysiology, MEG, and PET. Also underway is an initiative to develop a framework for statistical models (BIDS Stats Models) allowing computations to be specified using a JSON file [*see* Glossary] so that they can be more easily documented and reproduced. Finally, a BIDS-mega specification is being proposed to standardize the way that multiple BIDS datasets can be integrated for large-scale computations. (A full listing of BIDS extensions proposals under development can be found at https://bids.neuroimaging.io/get_involved.html.)

A second requirement for interpretable large-scale, cross-study analyses is a standardized metadata specification. While the BIDS conventions include specifications for basic metadata (who, what, when, where, etc.), one important annotation category has remained nearly unstandardized and very often underspecified—the nature of the *events* recorded during or later discovered to have occurred during time series recordings. That is, the answer to the question, "What exactly did the participant(s) experience and do during this recording?

The Hierarchical Event Descriptors (HED or 'H-E-D' system is a standardized method of capturing information about events in dataset in a common metadata format to produce event annotation

ready for machine analysis [*see* Resources]. HED, first proposed by Nima Bigdely-Shamlo over a decade ago [6] and now in its third major version, has many tools and features that facilitate production and validation of standardized annotations that can be searched and used in analyses [7, 8]. HED was formally accepted as part of all BIDS modality standards in 2019 (BIDS v1.2.1-). Current and in-progress HED tools support data annotation, validation, search, summary, and analysis [https://www.hed-resources.org]. The HED framework includes a base standardized vocabulary and a tool suite supporting annotation, validation, and analysis in a combination of online tools, Python-based command line scripts and notebooks, and MATLAB scripts and plug-in tools for EEGLAB [*see* Resources]. See https://osf.io/8brgv/ for links to HED and other resources discussed in this chapter.

HED metadata can enable intelligent combining of event-related data from different recordings and studies in sophisticated analyses including those using machine learning. Because EEG, MEG, and iEEG data, in particular, have very fine time resolution (much quicker than our thoughts and actions), event-related analysis is crucial in processing of neuroelectromagnetic data. The dominant EEG analysis approaches including event-related potential (ERP), time/frequency, and dynamic connectivity averaging rely on event descriptions and time markers to isolate sets of similar data excerpts (or epochs) for meaningful comparison. HED annotation is equally applicable to any other time series data, including fMRI [*see* Chapter 7].

HED also now supports a 'library' mechanism using which specialized research communities can develop additional HED "library schemas" to specify HED term vocabularies for event description of neuroimaging research subfields including language, body movement control, clinical neurophysiology, and others. HED's first library schema, a library of terms used in interpretation of clinical EEG recordings (now available at https://www.hedtags.org/display_hed.html), incorporates the standardized SCORE (Standardized Computer-based Organized Reporting of EEG) for annotation of clinical EEG [9].

The past (and still most current) conception of experiment events conflates *event processes* (unfolding over time) with *event-phase markers* that point (typically) to the *onsets* or to other time points of interest in event processes on the experiment timeline. HED distinguishes in principle between *event processes* (having duration) and *event-phase markers* (pointing to a single moment on the experiment timeline). For many experiment events, the second most important *event phase* to mark is its *offset*. BIDS allows (but does not demand) that event onset markers include a measure (in seconds) of the *duration* of the event, from which the event *offset* moment can be calculated. HED allows other (inset) event phases to be annotated as well, facilitating detailed analysis of complex events.

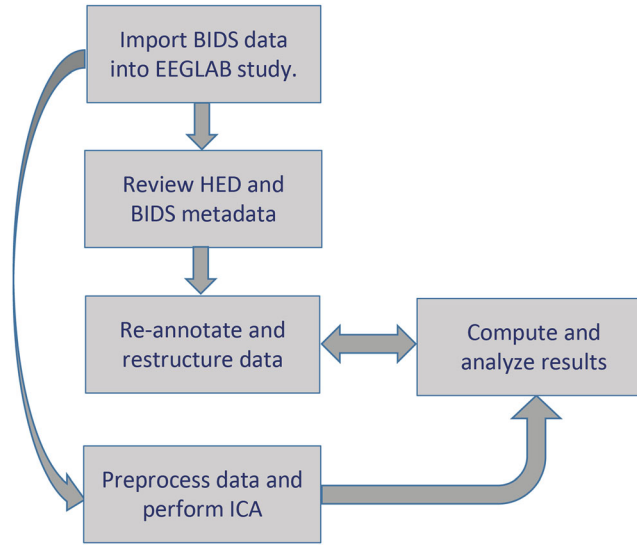

Fig. 1 Chapter overview of the end-to-end process of analyzing EEG data incorporating HED annotation

A further pillar supporting reproducible data analysis is the use of well-documented and automated analysis pipelines. This process is highly dependent on the tools and tool platform used in the analysis. In this chapter, we demonstrate an approach to such an analysis using tools from the EEGLAB environment [10], a widely used processing platform for M/EEG data that fully supports both BIDS and HED. Here, we provide practical guides on the input/output (I/O) workflow between BIDS, HED, and the EEGLAB environment, and show how researchers using EEGLAB can create and/or import BIDS-formatted, HED-annotated datasets. We also provide practical comments on event-related processing pipelines that can be applied *within* or *across* datasets, including those using different task paradigms.

Figure 1 gives a compact overview of the topics covered in this chapter. After reviewing some background material, we demonstrate how to import a BIDS dataset as an EEGLAB Study. We introduce tools available for reviewing and correcting HED annotation and other BIDS metadata, and perform re-annotation as necessary. Independent of metadata review, we demonstrate preprocessing the data including ICA decomposition [11], and as an example show how to test a simple hypothesis concerning source-resolved event-related potentials (ERPs) to sensory presentations of different classes of letters in a demonstration experiment.

2 Methods

2.1 Starting Point: Obtaining the Data

This chapter discusses end-to-end processing of shared or newly collected M/EEG data, from raw data to end results. We assume that the raw data have been made available in BIDS format from an open repository using BIDS formats such as OpenNeuro or its neuroelectromagnetic data portal NEMAR (https://nemar.org/). We also demonstrate tools for importing new or unannotated raw data into BIDS using EEGLAB tools.

2.2 Data Storage and Computing

We use data from one participant in the 24-participant Sternberg Modified Memory Task [12], available in BIDS format with HED annotation in OpenNeuro and NEMAR under accession number *ds004117*. Its paradigm has recently been chosen for replication as part of the EEGManyLabs reproducibility study [13]. The single-participant (~380 MB) demo dataset used here is available at https://osf.io/8brgv/ [*see* Chapter 2].

We assume for demonstration purposes that users will download the demo data on their local computer to experiment with using the tools we describe. Users should have at least 16 GB of memory on their local machine to comfortably run even the single-participant analysis. The Neuroscience Gateway (NSG, https://www.nsgportal.org) at the San Diego Supercomputer Center is a world neuroscience community resource that supports free use of high-performance computing resources to run user-defined analyses built on any of multiple analysis environments (e.g., MATLAB, python, R) and toolsets (EEGLAB, Freesurfer, Open Brain, TensorFlow, PyTorch, NEURON, etc.). NSG enables users to submit analysis scripts of their own design to process either their own or publicly shared (on NEMAR.org) data using either the nsgportal EEGLAB plug-in or the NSG web-browser interface. Data shared via the NEMAR resource (www.nemar.org) are immediately available for analysis via NSG without data download and re-upload. EEGLAB users can also use EEGLAB *nsgportal* plug-in tools to submit jobs to NSG directly from an EEGLAB MATLAB session, as discussed later in this chapter.

2.3 Software and Coding

Although some basic understanding of analysis scripting may be needed, the tools discussed here focus on those providing support for the use of GUIs (graphical user interface) based analysis pipelines. The demonstration analyses use EEGLAB running on MATLAB (The Mathworks, Inc.). To run the demos, users must have MATLAB installed and must download and install EEGLAB as described in https://eeglab.org/download/. An extensive EEGLAB tutorial is available at https://eeglab.org/tutorials/. Preliminary assessment of event structure and annotation using

Hierarchical Event Descriptors (HED) can be done entirely using online HED tools at https://hedtools.org/hed, with no coding or software installation required. We discuss two paths for running an EEGLAB analysis pipeline, one using EEGLAB GUI windows and/or command line scripts that run locally, and another using online high-performance computational resources freely available to neuroscience researchers via the Neuroscience Gateway (NSG).

2.4 Computational Requirements

Computational requirements for importing and annotating the data, as shown here, are minimal. In general, analysis time per data recording depends on the complexity of the analysis, while processing time on the entire dataset grows linearly with the number of processed recordings. When processing time is a limiting factor (making some analyses extend beyond the practical compute horizon), processing large bodies of data via NSG can take advantage of parallel processing across 128 or more cores, with GPU resources also available.

2.5 Background

In this section we review some useful background material about the demo data, BIDS data organization, EEG data formats, and some HED basics. If you are already familiar with these topics, you may skip this section or skip to Subheading 3.

2.5.1 Sternberg Working Memory Dataset

The example data recordings used here are from the Modified Sternberg Working Memory dataset [12], which can be examined on NEMAR (https://nemar.org) and downloaded from NEMAR (or from OpenNeuro.org) as dataset *ds004117*. The single-participant demonstration dataset used here is available separately at https://osf.io/8brgv/. Figure 2 shows a schematic timeline of each experiment task trial. Under BIDS, onset markers for each event are stored in an *events.tsv* file for each recording, as well as, here, in the EEG data itself in EEGLAB format. In both cases the event record has a tabular structure with each row representing an event onset marker and columns recording event aspects. One column always records the time of the event marker relative to the experiment timeline.

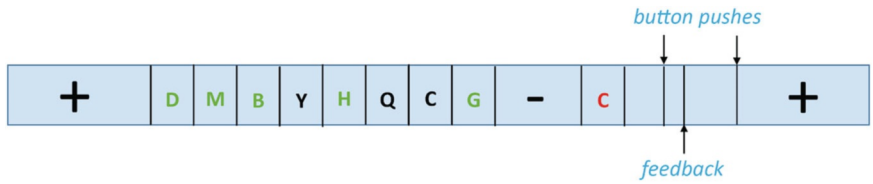

Fig. 2 Schematic timeline of the sequence of sensory presentation and participant action events in each trial of the Modified Sternberg Working Memory experiment. See the text for details including the meanings of the letter colors

In this experiment, each task trial begins with the display of a central fixation cross for 5 s followed by a sequence of 8 centered single letter presentations, each displayed for 306 ms, followed by a 1.4-s empty-screen delay. Each 8-letter sequence includes between 3 and 7 black letters "to be remembered" as well as (5 to 1) green letters "to be ignored". A central dash is then displayed for between 2 s and 4 s (the "memory maintenance" period). A red probe letter then appears, prompting the participant to click either a right-hand controller button (using their dominant hand index finger) if the probe letter was presented in the preceding sequence as a black (to be remembered) letter, or otherwise a left button (with their thumb). All participants were right-handed. The participant response was followed by a feedback sound—a "beep" for "correct" or a "buzz" for "incorrect". Thereafter the participant pressed either controller button to indicate their readiness to proceed to the next trial.

Note

As in nearly all actual experiment datasets, participants were not always able to perform each pair of required trial button press actions appropriately. Across the dataset, there were a few trials in which the participant pressed a button multiple times prior to receiving the trial feedback sound, a few trials in which they did not press a response button, and one session in which for some reason the nature and timing of the auditory feedback was not recorded. The raw dataset also included event marker sequences indicating incomplete trial presentations (e.g., trials in which no letters were presented).

Trial irregularities make it more difficult to perform automated analysis without building specialized programs to handle such irregularities. To help downstream users minimize required special handling, we recommend that, where appropriate, a trial-number column be included in the *events.tsv* file to allow analysis scripts to identify and then efficiently process valid trial events in the data.

If you are annotating a dataset new to you, it is useful to study the experimental event sequence and check, for each participant, whether all its recording segments (runs) and run elements (trials, if any) conform to the expected syntax. The Modified Sternberg Working Memory dataset paradigm is well-structured, producing a well-defined sequence of event onsets that should be present and accounted for in each data trial, making it easy to check for and exclude "bad" (non-standard) trials. If an experiment paradigm is not trial-structured or does not have an easily defined trial

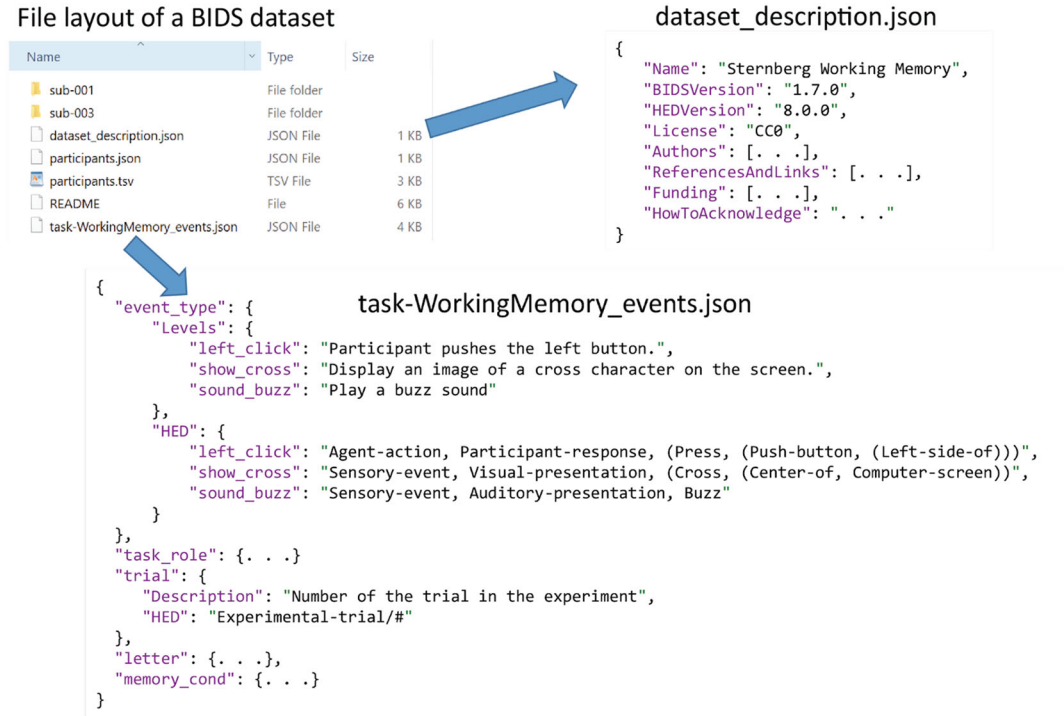

Fig. 3 The Sternberg Working Memory Demo dataset in BIDS format. Top left: the top-level file organization. Top right: The *dataset_description.json* file providing basic dataset identification. Lower: The *task-WorkingMemory_events.json* file giving event-related metadata for the dataset. In particular, it gives text descriptions and HED strings for events whose event types (here using any convenient titles) are indicated in a column of the event list file

sequence, this may require custom coding. The chapter by Denissen et al. introduces some tools that support event file restructuring without programming.

2.5.2 BIDS Dataset Format

BIDS specifications specify data and metadata formats and dataset disk file organization. Dataset metadata are stored in a variety of *JSON* (*.json*) and *tab-separated value* (*.tsv*) ASCII files. The top-level file organization for the Modified Sternberg Working Memory demo dataset is shown in the upper left of Fig. 3. Contents of two important top-level metadata files, *dataset_description.json* and *task-WorkingMemory_events.json*, are shown in the upper right and bottom center respectively.

BIDS datasets are organized by participant (subject), with each participant's data organized in a unique folder (named '*sub-xxx*'). The Dataset EEG data are stored in a participant *eeg/* subdirectory. Optionally, intervening sub-folders may provide organization of the participant's data into session subfolders, as in our example. Several excellent tutorials in the *BIDS Starter Kit* explain this file organization and its implications in more detail (see https://bids-standard.github.io/bids-starter-kit/).

The required top-level files (by name *dataset_description.json*, *participants.tsv*, and *README*) contain some overall information about the data. Of immediate interest in the *dataset_description.json* are the versions of BIDS and HED formatting used in building this dataset. A list of relevant reports describing the experiment and its goals and interpretation may also be included (though perhaps unfortunately BIDS does not require them).

Note
The top-level BIDS *dataset_description.json* file does not contain a text description of the experiment. A full experiment description and a description of the original experiment goals, written in plain language, should instead be contained in the top-level *README* file.

A BIDS file of particular interest for this chapter is the *task-WorkingMemory_events.json* file, which contains metadata about the types of events recorded in the data, including HED annotations that enable machine-actionable event analysis. This file will be discussed in more detail later.

2.5.3 EEG Data Formats

EEG datasets include continuous recordings of multi-channel EEG. Associated with these continuous recordings are event markers (in common practice, event *onset* markers), typically first recorded in log files produced by the experiment control software. BIDS allows EEG data themselves to be stored in any of four widely used formats: EEGLAB (*.set*, or *.set* plus *.fdt*), European Data Format (*.edf*), Biosemi (*.bdf*), and BrainVision (*.vhdr* + *.vmrk* + *.eeg*). EEGLAB has tools for converting each of these formats to its internal *.set* file format.

Note
European and Biosemi data formats store event and channel information within their respective *.edf* or *.bdf* files, while BrainVision data format stores event markers separately (in an *.vmrk* file).

An EEGLAB-based BIDS dataset can store data for a participant session either in a single, continuous *.set* file or in a combination of an *.fdt* file (containing the binary data) and a *.set* file (containing metadata). The latter format allows metadata for a

large number of *.set* files to be gathered and stored in computer memory into an EEGLAB *Study* (equivalent to a BIDS *dataset*) and then manipulated, without needing to actually load the much larger *.fdt* data files.

Once loaded, either by calling the EEGLAB *pop_loadset* function or using the EEGLAB GUI, the data is stored in a MATLAB (EEG) data structure. Dataset event onset markers are stored in the *EEG.event* field of this structure, while channel electrode location and other information are stored in the *EEG.chanlocs* field. The EEG data itself are stored in *EEG.data* as a two-dimensional array, scalp channels by time samples.

Note
BIDS requires that event and channel information be stored separately from the data, in *events.tsv* and *channels.tsv* files, respectively. In any of the four supported EEG data formats, data event and channel information are also stored in the EEG data itself. There is, however, no guarantee (and BIDS does not check) that this internally stored information is consistent with the BIDS *.tsv* files. Hopefully, the BIDS formatted *.tsv* file data will be at least as complete and hopefully more complete than that stored in the raw datafiles. EEGLAB allows users to choose which information to import from the *.tsv* files and which to use from the stored data record itself.

2.5.4 HED Quickstart

As mentioned in the introduction, the HED system consists of a standardized term vocabulary of terms organized hierarchically to in the HED standard schema), as well as an extensive tool base supporting HED annotation, validation, and analysis. Figure 4 gives an overview of the standard HED term schema.

The HED schema is organized around six top-level HED tag subcategories (*Event*, *Action*, *Agent*, *Item*, *Property*, and *Relation*). Users create "HED string" event marker annotations as comma-separated lists of ("HED tag") terms from the schema. Because each term can appear in only one place in the schema, during the annotation process users can specify only the end tags (leaves) in the typically shallow tag hierarchies (subtrees) [*see* Appendix], leaving HED machine tools to fill out their full schema tag paths.

A second important point about the HED schema design is that terms lower in the hierarchy are subcategories of supervening terms (i.e., child nodes that satisfy an *is-a* relationship). This is important for search generalization. For example, a search for annotations containing the term *Event* will also return annotations containing *Sensory-event* or *Agent-action*.

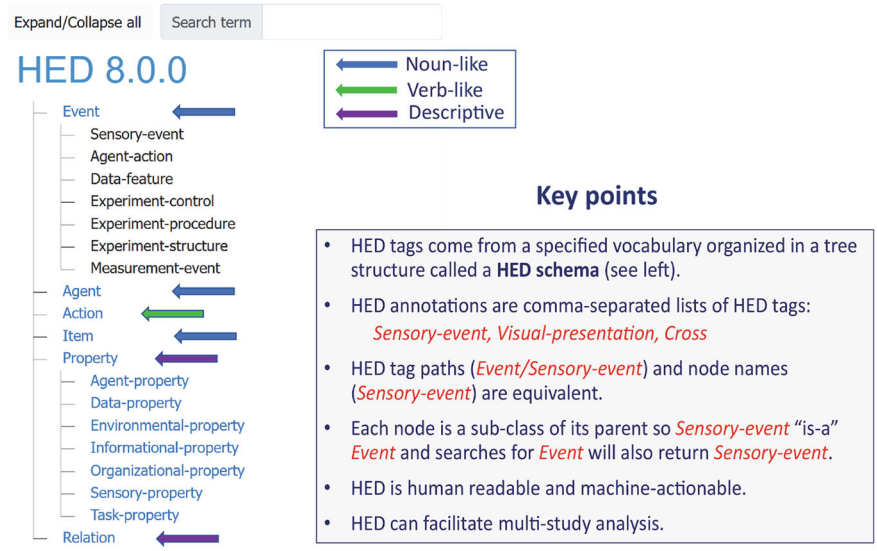

Fig. 4 The standard HED (Hierarchical Event Descriptor) term schema. Left: Partial view of the top-level HED standard schema. See https:/www.hedtags.org/display_hed.html to explore the full schema using an expandable accordion view. Arrows point to top-level tag sub-categories. Right: Some key points are noted

The HED official vocabulary is stored on GitHub in the *hed-schemas* repository of the *hed-standard* organization (https://github.com/hed-standard/hed-schemas). The official HED specification document is available at https://hed-specification.readthedocs.io/en/latest/. The vocabulary is versioned using semantic versioning; HED tools and tools that use HED tools (including the BIDS validator, https://github.com/bids-standard/bids-validator) retrieve a copy of the schema (in XML format [*see* Glossary]) during processing.

A HED schema is needed to perform validation because users usually give a HED annotation as a list of single terms and the software validator must verify that these terms are in the allowed vocabulary and that the term usage is consistent with its properties. For example, terms that take values must have consistent value type and units. Fetching of the schema and processing are done behind the scene by HED tools. These tools (and pertinent tutorials) are described below.

2.5.5 HED in BIDS

As mentioned in the introduction, the HED system is integrated into BIDS [*see* Chapter 4] and is the only BIDS-endorsed mechanism for documenting events using dataset-independent, machine-actionable metadata. However, HED is new to many BIDS users. Thus, (e.g.) a large number of time series datasets on OpenNeuro do not yet have HED tags.

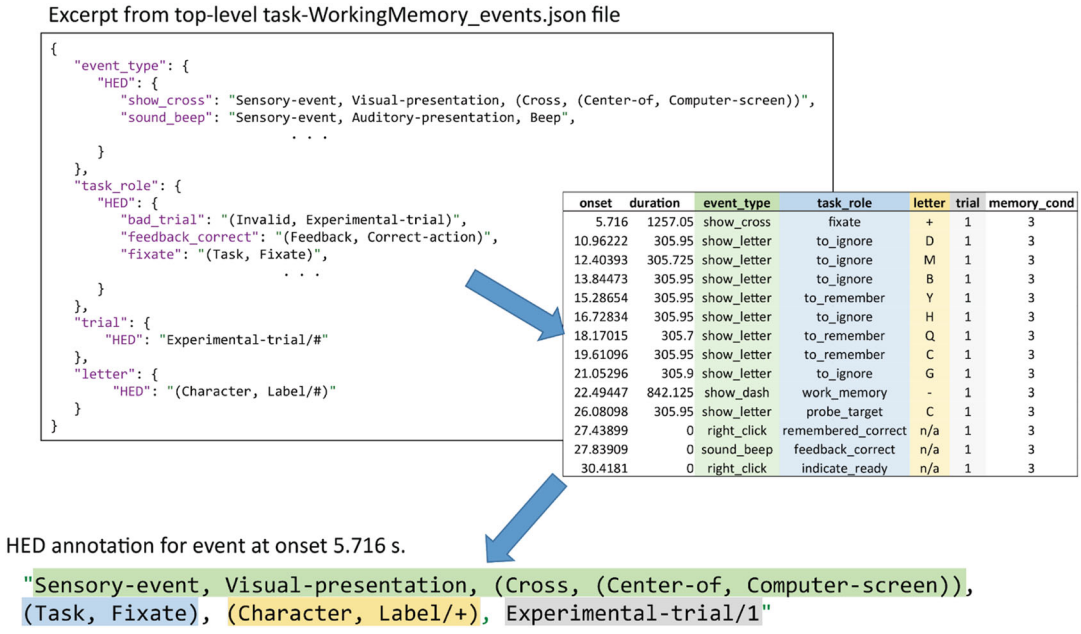

Fig. 5 The assembly of HED annotations for a BIDS *events.tsv* file. Each column value in the events file (here, "show_letter", "to_ignore", etc.) is defined using a list of HED tags keyed to its name in the JSON *events.json* dictionary. The HED tags for each column value in a row of the .tsv events table are assembled to form the HED annotation for the event

As illustrated schematically in Fig. 3, HED annotation for most BIDS datasets consists of providing a single JSON file containing a dictionary associating values in the columns of the dataset event files with HED annotations. Once this dictionary is provided, tools can automatically take advantage of this metadata during event processing. This means that users can contribute HED annotations post hoc, and that re-annotation focused on enabling further analysis only involves modifying a single text file.

Figure 5 shows the mechanism by which this single top-level *events.json* file is used in conjunction with the dataset events files to produce HED annotations for each event. The top left of Fig. 5 shows an excerpt from the *task-WorkingMemory_events.json* file located in the dataset root directory. This JSON file is a text file containing a dictionary whose keys are the column names of the *events.tsv* files (as in the excerpt on the right in Fig. 5). For example, the *event_type* column has a small number of discrete code values (*show_cross*, *show_letter*, *right_click*, etc.). The JSON file associates a HED annotation with each value in the *event_type* column. Here, for example, the term *show_cross* is associated with the HED annotation "*Sensory-event, Visual-presentation, (Cross, Center-of, Computer-screen)*". The code-value names are arbitrary, but should best be informative.

Additional HED tags might be added to these event type definitions—here, for example, to record the color and/or size of the displayed cross. HED tags for such details might be added later by researchers to the archived and shared data to allow them to investigate questions about brain activity different from those of the original data authors—if these experiment details were preserved in some other way, for example in stimulus images, screenshots or recordings, or were specified in the preserved experiment control script. In general, data authors cannot be expected to anticipate all possible research interests of future data users—particularly when stimuli and/or tasks involved are more complex than in this experiment. Thus, there may thus often be some tension between the aim of data authors to document the exact nature of experiment events and the level of effort and imagination required of them to fully accomplish this.

The *events.tsv* files are columnar text tables with tab characters separating columns. Each row in an *events.tsv* file represents an event-time marker—currently nearly always a marker of its onset. BIDS requires that event files have an *onset* column giving the time in seconds of the event marker relative to the start of the associated data recording. HED also supports event *offset* tags and *inset* tags marking any intervening event phase transitions (for example, the moment of maximum amplitude of a musical sound or the maximum velocity of a movement arc).

BIDS has strict naming conventions for determining which JSON files are associated with each event file [*see* Chapter 4]. A discussion of these conventions is beyond the scope of this chapter, but suffice it to say that the JSON file named *task-XXX_events.json* applies to all *events.tsv* files that have *task-XXX* in their file names. So a single JSON file can be (and typically is) used to hold HED annotations of all the event types in an entire dataset. (See https://bids-specification.readthedocs.io/en/stable/02-common-principles.html#the-inheritance-principle in the BIDS specification document for details.)

Some columns in *events.tsv* such as, here, *letter* and *trial* can have many different numeric or text values. Rather than provide an HED annotation for each value separately, HED allows a single annotation with placeholder (#) to apply to every value in the column. The specific column value for each event marker is substituted for the # when the annotation is assembled.

The bottom of Fig. 5 shows the result of final assembled HED annotation. For each row (event marker) in the *events.tsv* file, HED annotations for the individual row column values are assembled from the JSON dictionary. If an annotation exists, it is included; if no annotation is present, the column value is omitted from the final assembled annotation.

Note
During event annotation the annotator should carefully check that all terms used in the *events.tsv* table are either values that will replace a # placeholder or are separately defined in the corresponding *events.json* file. Otherwise, the table term will not contribute to the assembled HED annotations.

HED has a full suite of online tools (https://hedtools.org/hed) to support HED annotation and processing. We encourage readers to download the Sternberg Working Memory demo dataset (available at https://osf.io/8brgv/) to explore these options. Figure 6 shows a screenshot of the operation to assemble HED annotations for the event file corresponding to the first recording (*run 1*) of participant (subject) *sub-001*. Fill in the options as indicated for the downloaded demo data and then press *Process* to see the output event file containing the assembled HED string event descriptions.

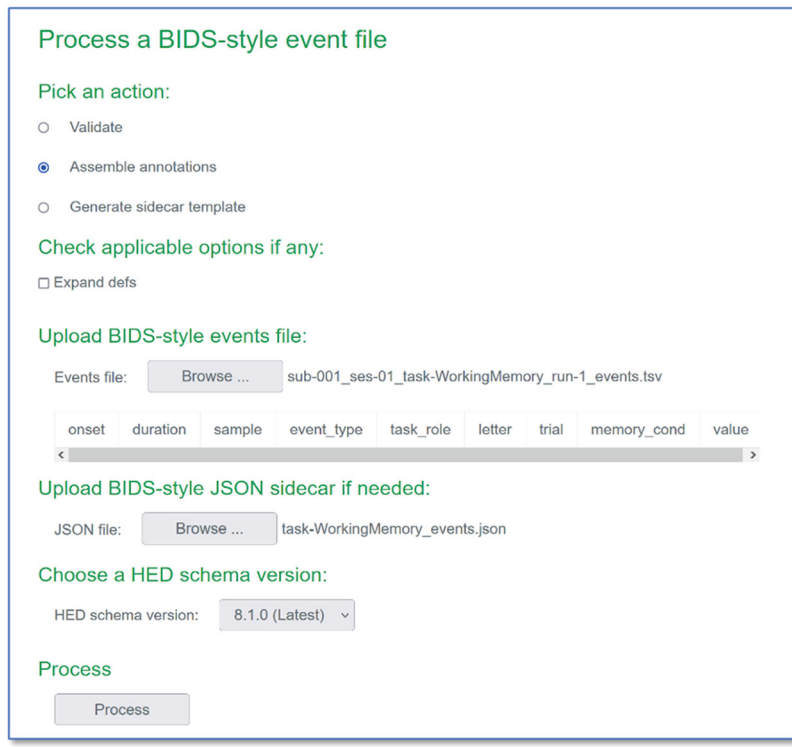

Fig. 6 An example of using the online HED tools to assemble HED annotations for a BIDS events file

The standard process for creating HED annotations in BIDS is to create a JSON template file from a representative *events.tsv* file using the online tool *Generate sidecar template* and then edit the resulting JSON text file directly or using the *Ctagger* tool GUI for assistance in selecting the appropriate HED tags to annotate the data.

A step-by-step process for annotation in BIDS is provided in the following quickstart tutorial:

https://www.hed-resources.org/en/latest/BidsAnnotationQuickstart.html.

CTagger, the HED annotation tool, can be run standalone or EEGLAB plug-in. A step-by-step guide to installing and using CTagger is available in the *Tagging with CTagger* tutorial:

https://www.hed-resources.org/en/latest/CTaggerGuiTaggingTool.html.

All of the tools provided by the HED Online Tools are also available as web REST services. A tutorial on how to use these services within MATLAB is available at:

https://www.hed-resources.org/en/latest/HedMatlabTools.html.

Sample code for calling these services from MATLAB can be found at:

https://github.com/hed-standard/hed-matlab/tree/main/hedmat/web_services_demos.

The most challenging part of learning to perform HED annotation is getting started making annotations. The HED annotation quickstart tutorial:

https://www.hed-resources.org/en/latest/HedAnnotationQuickstart.html

provides a step-by-step recipe for performing basic HED annotations.

A number of annotated sample datasets are available in GitHub *hed-examples* repository of *hed-standard*: https://github.com/hed-standard/hed-examples/tree/main/datasets.

Note

HED annotations vary in their complexity and completeness, and many datasets currently on OpenNeuro do not yet include HED annotations. However, basic HED annotations can now be added quite easily to a BIDS dataset using the methods we describe. As suggested earlier in this chapter, HED annotations (as well as event logs and files themselves) can be enhanced and/or modified at any later time to support analysis goals. This may become particularly important for research using novel objectives, or investigations across studies.

3 Methods

3.1 Setup

3.1.1 Software Installation

The remainder of this tutorial assumes that you are working with EEGLAB in MATLAB. If you do not already have EEGLAB loaded, download and install it by following the instructions available at https://eeglab.org/tutorials/01_Install/Install.html. You should also install the *bids-matlab-tools* and the *HEDTools* EEGLAB plug-ins. To download and install these plug-ins, go to the *Manage EEGLAB extensions* submenu of the *File* menu on the main EEGLAB, then search for and install these plug-ins directly through this menu.

3.1.2 Downloading Data

The demos in this chapter use the Sternberg Working Memory demo dataset, which is available for download at https://osf.io/8brgv/. This single-participant dataset is part of a 24-participant dataset from an experiment performed by Onton et al. (2005) [12] that is available on OpenNeuro (https://openneuro.org/datasets/ds004117).

3.1.3 Importing the BIDS Dataset

The analysis in this chapter assumes that the dataset is in EEGLAB Study format. The EEGLAB *bids-matlab-tools* [14] allow easy import and conversion through the *BIDS tools* submenu item of the *File* menu item in the main EEGLAB GUI. Figure 7 shows the BIDS import tool menu when the demo dataset is imported.

The EEGLAB BIDS import tool does not check the consistency of the BIDS files, but it does allow users to select which external information related to channels and events will overwrite the internal data. MATLAB scripts for checking the consistency between the BIDS external representation of events and of channels are available at:

https://github.com/hed-standard/hed-matlab/tree/main/hedmat/utilities.

3.2 Preprocessing and ICA Data Decomposition

For this demonstration, our preprocessing of the single-participant data began with high pass filtering the data with a cutoff at 1.5 Hz, as this is useful for subsequent ICA decomposition. The data were then re-referenced to common average and excessively noisy portions of the data were removed by running the *pop_clean_rawdata* EEGLAB plug-in [15, 16] *without* using data interpolation. Here, no excessively noisy ("bad") channels were detected or removed in this process. The data were then decomposed using Adaptive Mixture ICA (AMICA) [17] with its automated data rejection option engaged using default parameters. AMICA uses data rejection only during training, to learn an 'unmixing' matrix linearly transforming the data from the input channels to a set of independent component (IC) processes that, when back-projected through the

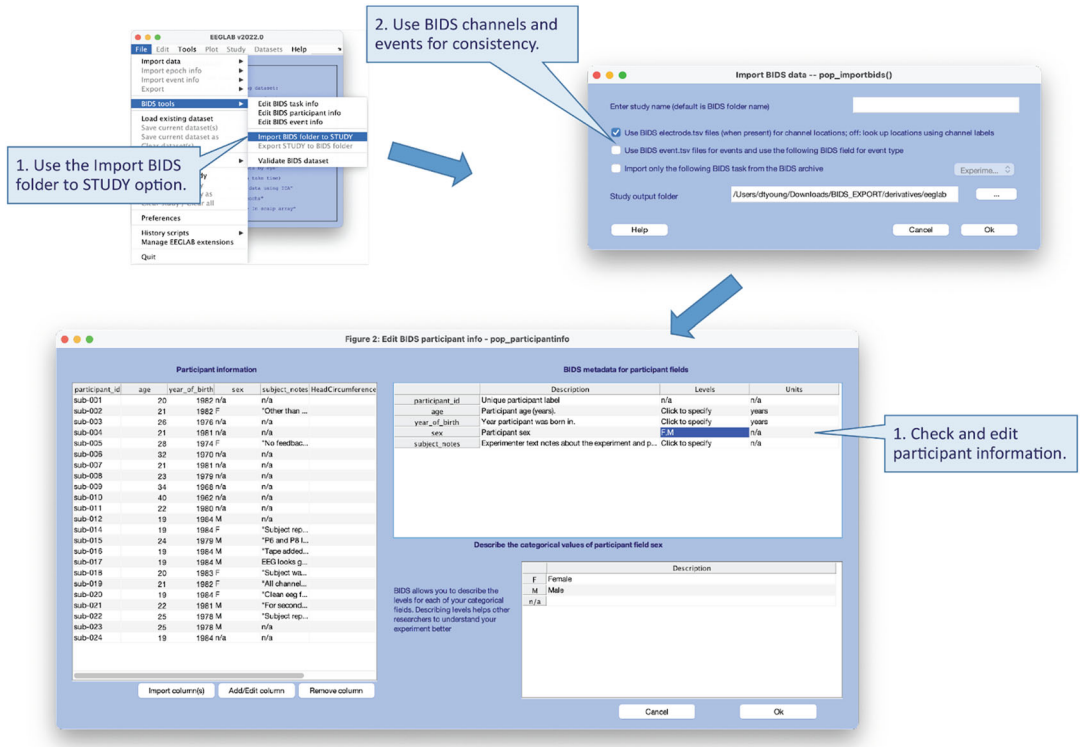

Fig. 7 The BIDS tools import tool for EEGLAB for the demo dataset

'mixing' matrix (the inverse of the unmixing matrix), will reconstitute the original data. ICA decomposition of EEG data finds ICs whose time courses are maximally temporally independent.

ICA decomposition is a powerful source separation method used to isolate both brain and non-brain source contributions in EEG data and data measures including ERPs [18]. The brain generated ICs that ICA discovers in the data may be better called "effective brain sources", as considered at either the neuronal or larger spatial scales the whole cortex is always active. However, local-scale activity in cortex, projected through the brain tissues, skull, and scalp and then summed at the scalp electrodes, is very largely canceled out by phase cancellation (positive voltage projections canceling negative projections). What dominates the "far field" scalp EEG signals are projections from larger areas of local field potential coherence that arise spontaneously by mechanisms that have not yet been well studied or understood. These projected potentials sum to contribute appreciable voltages (positive or negative) at the scalp channels, thereby constituting "effective brain sources" of EEG signals. ICA decomposition separates out the

appreciable effective sources from spatially and functionally distinct non-brain sources (potentials contributing to the scalp channels from eye movements, line noise, scalp and neck muscle activities, etc.). We have shown, using mutual information reduction criteria, that AMICA is the most effective ICA decomposition approach, though also the most computationally complex. Other, less computationally complex algorithms approach AMICA in effectiveness [19]. Recently, a very efficient version of AMICA has been compiled for use on high-performance computer resources supported by and made freely available through the Neuroscience Gateway (NSG).

Another advantage of AMICA is that it performs its own data rejection internally, so as to not be misled by atypical noisy data patterns that appear in the data with unique (non-stereotyped) spatial nature incompatible with the assumption of spatial source stationarity used in ICA derivation. However, by default AMICA does not return the data after rejection, and work is just beginning to apply its rejection decisions to the data for subsequent processing. Thus, here we applied *pop_clean_raw_data* for this purpose prior to ICA decomposition. This cleaning appropriately adjusted the set file's EEG.events table to reflect the new locations of event markers remaining in the clean data, while in the EEG.urevents subfield storing pointers to the retained events in their original sequence and timing.

The EEGLAB plug-in *zapline-plus* [20] was then used to remove line noise from the IC timecourses ('activations'). This is a recently introduced approach to line noise removal; another is the *cleanline* plug-in [21]. Removing line noise contamination without losing contributions of other brain sources at the line frequency can be difficult since head movements, moment-to-moment changes in the electrical environment, or the possible presence of electrical line frequency sources with phase differences, together create spatial instability in the line noise contamination pattern throughout the data. Thus, although (as here) ICA decomposition typically gathers much of the line noise contamination in one or a few (here two) ICs, substantial line noise contamination was still present in other, by scalp map and power spectrum clearly effective brain source ICs.

This was supported by applying *ICLabel* [22], a neural network classifier trained on a large body of expert-labeled IC data, to automatically classify components as representing brain sources or as any of several classes of non-brain sources (see discussion below). Applying *ICLabel before* removing line noise correctly identified two strong line noise ICs but also caused *ICLabel* to classify effective brain sources still containing substantial noise as most likely representing line noise rather than brain source activity. *After* running *zapline-plus* on the scalp data and then applying the AMICA weights to the cleaned data to obtain the IC time courses (activations), *ICLabel*, now applied to the IC scalp maps and line-noise

```
% high-pass filter the data and remove (60-Hz) line noise
EEG = pop_eegfiltnew(EEG, 'locutoff',1.5); % High-pass filter

% re-reference the data channels to common average
EEG = pop_reref(EEG, []);

% Remove excessively noisy ('bad') data portions
EEG = pop_clean_rawdata(EEG, 'FlatlineCriterion', 'off', 'ChannelCriterion', 'off',
'LineNoiseCriterion', 'off', 'Highpass', 'off', 'BurstCriterion', 'off',
'WindowCriterion', 0.25, 'BurstRejection', 'off', 'Distance', 'Euclidian',
'WindowCriterionTolerances', [-Inf 20] ); ;

% Perform AMICA decomposition of the data
amicaout = [pwd '/amicaout'];
EEG = pop_runamica(EEG, 'outdir', amicaout, 'numprocs', 1, 'do_reject', 1, 'rejstart',
3, 'numrej', 3, 'rejint', 3, 'pcakeep', EEG.nbchan);

% Remove line noise from data using Zapline_plus plug-in
EEG = clean_data_with_zapline_plus_eeglab_wrapper(EEG,struct('noisefreqs','line'));

% Remove line noise from IC activations using the line noise-removed data
EEG.icaact = EEG.icaweights*EEG.icasphere*EEG.data;

% Apply ICLabel to categorize component types
EEG = pop_iclabel(EEG, 'default');
```

Fig. 8 Preprocessing pipeline script for the examples used in this chapter. The data are first high pass filtered above 1.5 Hz (using the *pop_eegfiltnew* function). Then the data (recorded to a common reference electrode placed behind the right ear) are transformed to average reference. Next we use the *clean_rawdata* plug-in of EEGLAB to remove artifacts, here using conservative parameters (no channel rejection, data-portion rejection with default parameters, no channel interpolation). AMICA decomposition of the data is performed, then 60-Hz line noise is removed, here by *zapline-plus*. The independent component (IC) activations are recomputed following line noise removal, and the ICs are again categorized by *ICLabel*

cleaned activations, confidently classified the ICs in question as representing brain sources, while classifying the two near-wholly line noise ICs only as "Other" (e.g., non-brain).

Figure 8 shows a MATLAB script containing the entire preprocessing pipeline used here. The script can be run on a single participant (as in Fig. 8) or can be applied in a loop or as part of an automated pipeline executed on remote compute resources, as discussed later in this chapter.

Figure 9 shows results of the AMICA decomposition of the 71-channel data in the form of the scalp maps of the largest 35 (of 71) IC processes. All 71 IC scalp maps and activations were then input to the *ICLabel* EEGLAB plug-in for component identification. IC7 and IC15 accounted for much, but not all, of the line noise contamination in the data; the associated scalp maps are quite incompatible with local field activity projecting from a single (or even strongly-connected dual) cortical source area.

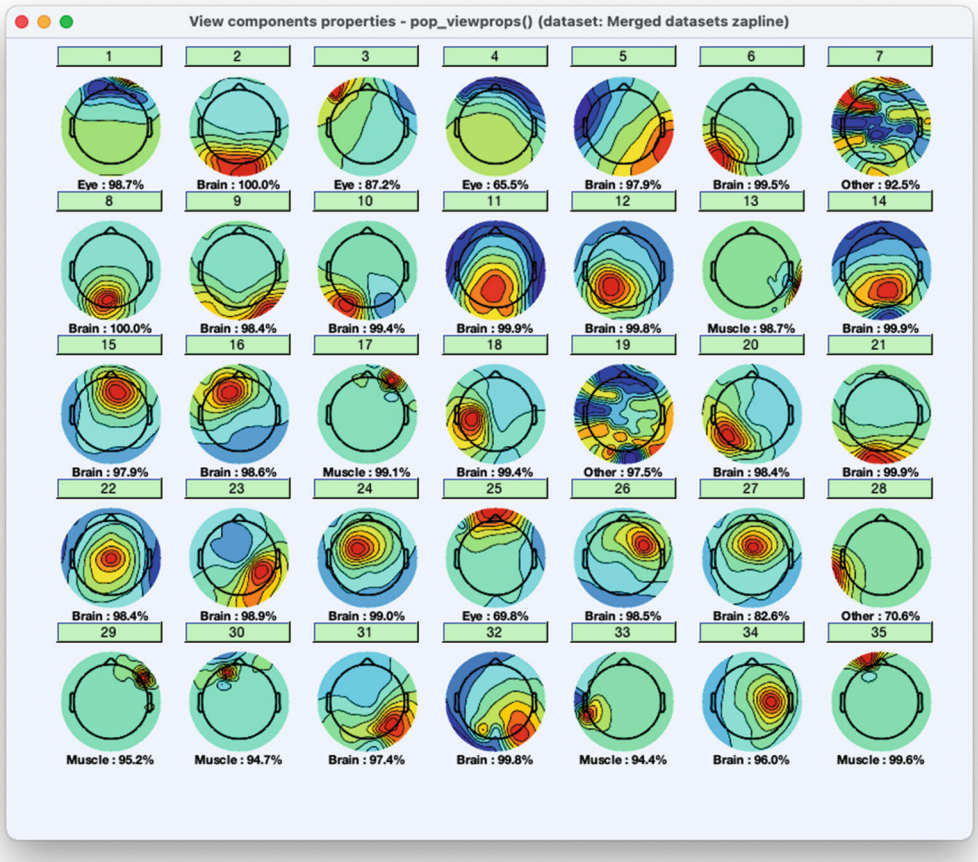

Fig. 9 Output of ICLabel after removing (60-Hz) line noise using *zapline-plus*. Here, ICA decomposition was performed before removing the line noise. The returned ICs accounting for most line noise in the data (here, ICs 7 and 19) are thus here labelled by *ICLabel as 'Other' rather than as 'Line Noise'*

Before removing line noise, *ICLabel* mis-classified brain source ICs such as IC8 as "Line noise" because of remaining line noise contamination. After removing line noise from the IC activations, *ICLabel* here correctly classified ICs such as IC8 as effective brain sources (with high likelihood), while the two major line noise sources (ICs 7 and 19) are now classified simply as *Other*. Several ICs classified as *Muscle* (ICs 13, 17, 29, 30, 33, 35) are compatible with effective sources of surface-recorded electromyographic (EMG) activity, which projects most strongly to the skin from the ends of individual scalp/neck muscles (i.e., from the muscle/tendon interface).

In recent years, several EEG data preprocessing pipelines have been developed and published by different laboratories. To our knowledge, there has been no systematic review of these, nor is it

quite clear what measures should best be used for fair comparison. It is best that EEG researchers acquaint themselves with the problems involved in adequate data preprocessing and test for themselves the particular pipeline they use or construct for this purpose. Because there is no agreed-upon standard [23], we hesitate to promote a particular approach (see https://osf.io/8brgv/ for an example of an automated pipeline). We prefer to involve ICA decomposition in this process as it is shown to perform well in identifying and separating out several classes of non-brain source signals typically mixed in the scalp data (eye movements, scalp muscle activities), as well as identifying major, spatially localizable effective brain sources that together account for much of the brain's (largely cortical) contribution to the scalp data.

3.3 Epoching with Event Codes and HED

Onton et al. (2005) analyzed the dynamics of frontal midline theta in this modified Sternberg dataset during presentations of three different letter types. The mean trends in this study replicated previous findings (i.e., stronger frontal midline theta during letter presentations corresponded to more letters being held in memory). Yet these results accounted for relatively little of the trial-to-trial variation in theta power in the frontal sources. Therefore, theta dynamics were compared across working memory loads and then further decomposed into event-related spectral perturbations (ERSPs) across single trials to obtain additional insight. For researchers who are new to the dataset and interested in replicating or extending the analysis results, an integral step after running a preprocessing pipeline is to identify experimental events to use for epoch extraction.

"Epoching" of EEG data refers to extracting sections (epochs) of the data of equal duration time locked to particular classes of experiment events of interest, in order to compute data measures and perform statistical comparison on these and/or the epoched data. The standard method of data epoch extraction in EEGLAB is to use the *pop_epoch* function, specifying specific values in a particular column of the EEGLAB *EEG.event* structure, typically using the *EEG.event.type* field to select the desired class(es) of events. This requires studying the event type terms used by the original investigators, which may likely be idiosyncratic and are often opaque (e.g., "type 17")—and often fail to record distinctions that might in future prove fruitful to analyze.

EEGLAB now also supports epoching using (more detailed and informative) HED tag information. HED tags are stored in the *EEG* structure under EEG.etc. HED, as search terms using the *pop_epochhed* function. Using HED tags allows researchers to query events of interest in a more semantically meaningful way, such as "*Green*, Letter" and "*Press*, Push-*button*" instead of having to work with the originally-assigned cryptic alphanumeric codes (e.g., event types "17" and "256").

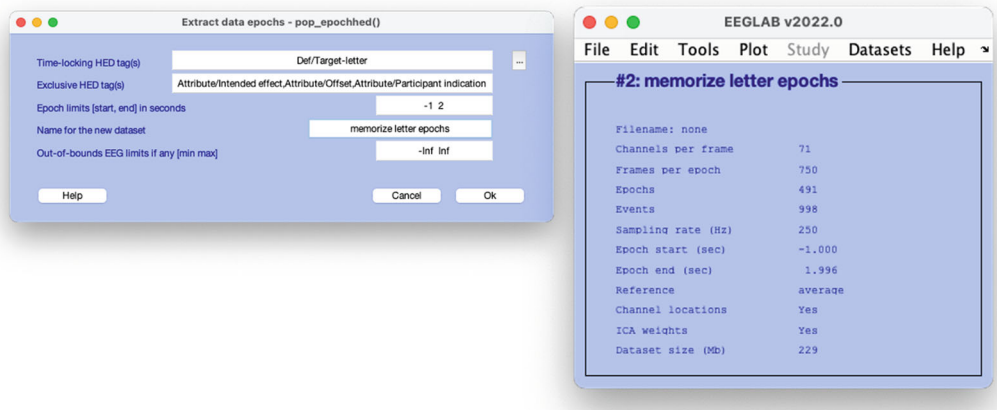

Fig. 10 The *pop_epochhed* function accessed through the Tools menu in EEGLAB

Figure 10 shows the graphic interface for the *pop_epochhed*. Tags can be specified in various ways including in combinations including AND and OR. More complicated search options are also available. The example shows looking for events that have the *Def/Target-letter* tag. Here *Def* indicates a user-defined term *Target-letter*. The definition of this user-defined term, as well as all HED annotations of the dataset can be found in the *events.json* file that accompanies the BIDS dataset (see the excerpt in Fig. 11) and is imported into the EEG.etc. HED field when the BIDS dataset is imported. HED tools look up the tags based on the annotations provided in the JSON sidecar and associate these annotations with the information in *EEG.event*.

Figure 10 shows a search for events containing the user-defined term *Target-letter*. The *Def/* prefix indicates that this term was user-defined using HED tags rather than being drawn directly from the base schema. The epoch start and end times are indicated relative to the position of the specified events.

Figure 11 shows that *Target-letter* is defined, using terms in the base schema, as:

"(Condition-variable/Letter-type, (Target, Memorize), (Letter, Black)))"

This definition includes the *Condition-variable/Letter-type* tag, as do the definitions for *Non-target-letter* and *Probe-letter*. By associating these three terms (*Target-letter*, *Non-target-letter*, and *Probe-letter*) with a common condition variable, the annotator indicates that these terms are three aspects of the same concept.

Most commonly, *Condition-variable* tags are used to group levels of an experiment stimulus or action condition associated with an experimental design. This mechanism allows data authors

```
"task_role": {
   "HED": {
      "bad_trial": "(Invalid, Experimental-trial)",
      "feedback_correct": "(Feedback, Correct-action)",
      "feedback_incorrect": "(Feedback, Incorrect-action)",
      "fixate": "(Task, Fixate)",
      "ignored_correct": "((Recall, Non-target), Correct-action)",
      "ignored_incorrect": "((Recall, Non-target), Incorrect-action)",
      "indicate_ready": "(Appropriate-action, Label/Indicate-ready)",
      "probe_not_shown": "Def/Probe-letter, (Cue, Non-target)",
      "probe_target": "Def/Probe-letter, (Cue, Target)",
      "remembered_correct": "((Recall, Target), Correct-action)",
      "remembered_incorrect": "((Recall, Target), Incorrect-action)",
      "to_ignore": "Def/Non-target-letter",
      "to_remember": "Def/Target-letter",
      "work_memory": "(Cue, Recall)"
   }
},
"curr_memory_load": {
   "Description": "The number of target letters shown up to this point",
   "HED": "(Condition-variable/Working-memory-load, ((Letter, Memorize),
Item-count/#))"
},
"condition_def": {
   "HED": {
      "memorize_cond_def": "(Definition/Target-letter, (Condition-variable/Letter-type,
(Target, Memorize), (Letter, Black)))",
      "ignore_cond_def": "(Definition/Non-target-letter,
(Condition-variable/Letter-type, (Non-target, Ignore), (Letter, Green)))",
      "probe_cond_def": "(Definition/Probe-letter, (Condition-variable/Letter-type,
(Cue, (Press, Mouse-button)), (Letter, Red), Recall))"
   }
}
```


Fig. 11 An excerpt of the top-level events.json sidecar for the Sternberg dataset

to encode the *experiment design matrix* in a standardized format that is consistent with the event structure. HED tools can automatically extract design matrices and also factor vectors associated with the *Condition-variable tags*. For additional information see the tutorial on HED Conditions and Design Matrices at:

https://www.hed-resources.org/en/latest/HedConditionsAndDesignMatrices.html.

Note also that the *Target-letter* definition above might easily be extended to note that the letters presented were in *upper_case*, *non_serif* font in the *Roman alphabet*, and their displayed width (in approximate degrees of viewing angle). These details might be of little or no interest for the analyses planned by the data authors themselves, but after the data were shared publicly might prove to be of real interest to some future studies, for example a study of reading of different character sets. However, when that interest

arose, information about the size and case of the letters presented in this study might no longer be available.

Such considerations might provide data author annotators some incentive to fill in such additional details during initial annotation. In fact, however, the current base HED schema does not contain terms for letter case, font, and alphabet. Therefore, the annotator wishing to record these details would need to either introduce off-schema terms into the annotation or else wait for them to be introduced into HED, most likely in the newly released HED *LANG* (language) library schema for linguistic terms (now in its first release). However, less extensive HED descriptions (such as those in Fig. 11) do indeed represent a real advance beyond the long practice of recording event onset markers only as "*type, 17*" and the like.

3.4 ERP Analysis Using HED-Based Epoch Extraction

The ICA decomposition by AMICA described in Fig. 9 was used to identify the effective brain source components in the data for source-resolved analysis. Using the HED-based epoch extraction demonstrated above, we here extracted epochs time locked to letter presentation onsets of the three types of stimulus conditions used in the experiment (*Def/Target-letter*, *Def/Non-target-letter*, and *Def/Probe-letter*). ERP trial averaging was then performed on the resulting epochs, with the results shown in Fig. 12 as visualized by EEGLAB function *envtopo*. These plots use a pair of thick black traces to plot the outer *envelope* of the bundle of 71 individual scalp channel ERP traces (i.e., the respective maximum and minimum channel values at each ERP trial latency).

The six independent component (IC) scalp maps show the scalp projection patterns learned by ICA decomposition for the six brain-source ICs making the largest contributions to the ERPs (across the 71 scalp channels) within the ERP trial latency window (50–400 ms) indicated by the dotted vertical lines. The top and bottom edges of the blue shaded areas show the envelope (again, the max and min channel values at each latency) of the joint (i.e., summed) projections of these six IC processes. Colored trace pairs show the envelope of the respective IC projections in the ERP data (again, the max- and min-value IC channel projections at each latency).

Plotting the data and channel envelopes in this way allows inspection of the time courses of multiple ICs in the ERP data period in comparison to the whole scalp ERP data. Note that most IC brain sources contribute to more than one ERP peak. Note also that the seeming much larger negative ERP peak near 120 ms (N1) following probe letter presentation onsets (bottom panel) is predominantly accounted for by larger peaks in the projections of the lateral occipital IC6 (with left posterior scalp projection, red traces) and IC9 (projecting predominantly to right posterior scalp, blue traces).

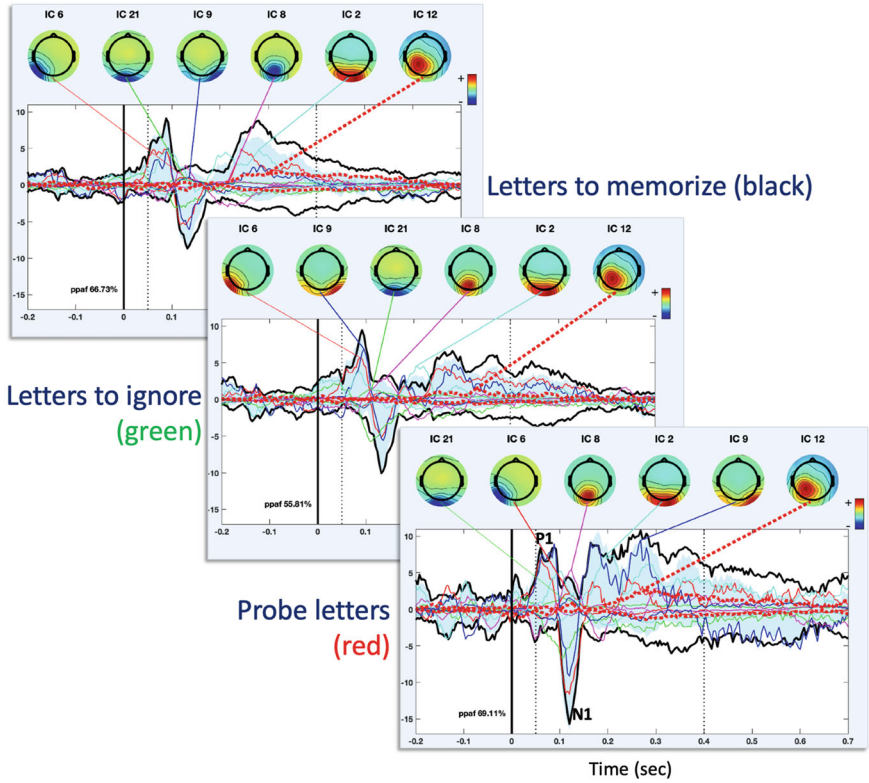

Fig. 12 Brain source resolved ERP trial averages, after removing large IC projections from eye movements, scalp muscle noise, plus small contributions to the data from other non-brain source ICs. The first two ERP peaks are conventionally referred to as P1 and N1. The three averages are time locked to onsets letter presentations of three types: (black) to be memorized letters ($n = 491$), (green) to be ignored letters ($n = 296$), and (red) probe query letters ($n = 99$). See text for further details

3.5 Computing Statistics for an EEGLAB Study

EEGLAB *Studies* (the EEGLAB equivalent of BIDS *datasets*) and EEGLAB *Study designs* provide a convenient way to compute single-participant and group level statistics for various EEG measures. For each design, users can define its independent variables by selecting the column of the event structure as the variable and the column's values as the variable's levels. Once the *Study design* is specified, EEG measures, including ERP, ERSP, power spectrum, and Intertrial coherence can be computed automatically. These measures apply to both the channel space and the independent component source space. Figure 13 shows sample menus for selection of statistics within an EEGLAB Study.

Once EEG measures are computed, users can choose to perform statistical analysis to compare measure values across levels of the specified independent variables. The LIMO-EEG plug-in extends EEGLAB statistical capability by allowing for an arbitrary

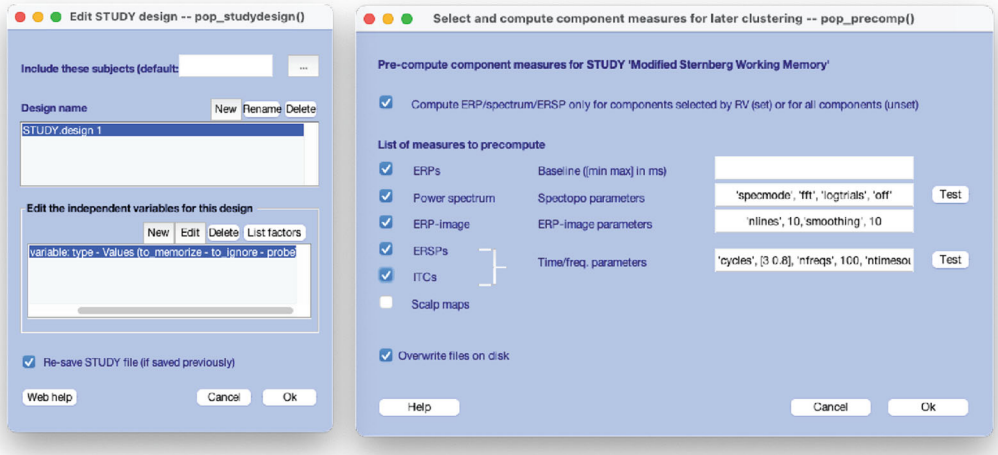

Fig. 13 (Left) The EEGLAB GUI to specify the Study design and its independent variable(s). (Right) The EEGLAB GUI to select the independent component measures to compute for subsequent statistical comparison

number of categorical and continuous variables in trial averages to be contrasted statistically by comparisons at the single-trial level [24].

Work is now underway to integrate *HEDTools* facilities for automated extraction of the experiment design matrix more tightly into the EEGLAB Study infrastructure. These facilities will allow users who import a HED-annotated BIDS dataset to run a chosen preprocessing pipeline and automatically compute all measures needed for statistical comparisons based on the *Study design* extracted from HED condition variable annotations themselves. Thus for a well-formatted HED-annotated BIDS dataset, the steps from importing those datasets into EEGLAB to reproducing statistical results as intended by the original data authors can be made automated. Of course, the primary uses for HED annotation lie in making possible more flexible (as well as larger scale) computing on neuroimaging data. It will create and fit new, more informative models of event-related brain dynamics occurring within a wide array of contexts and purposes (e.g., for basic understanding of brain function, neurological and psychiatric diagnosis, cognitive monitoring).

Many other functions are available within the EEGLAB platform and its associated plug-ins. While executing an analysis through the GUI is useful for exploratory work, it becomes tedious for large-scale analysis. After executing commands using the EEGLAB GUI during exploratory analysis, researchers can create a script for automating the process by editing the *ALLCOM* structure that accumulates the underlying EEGLAB commands that

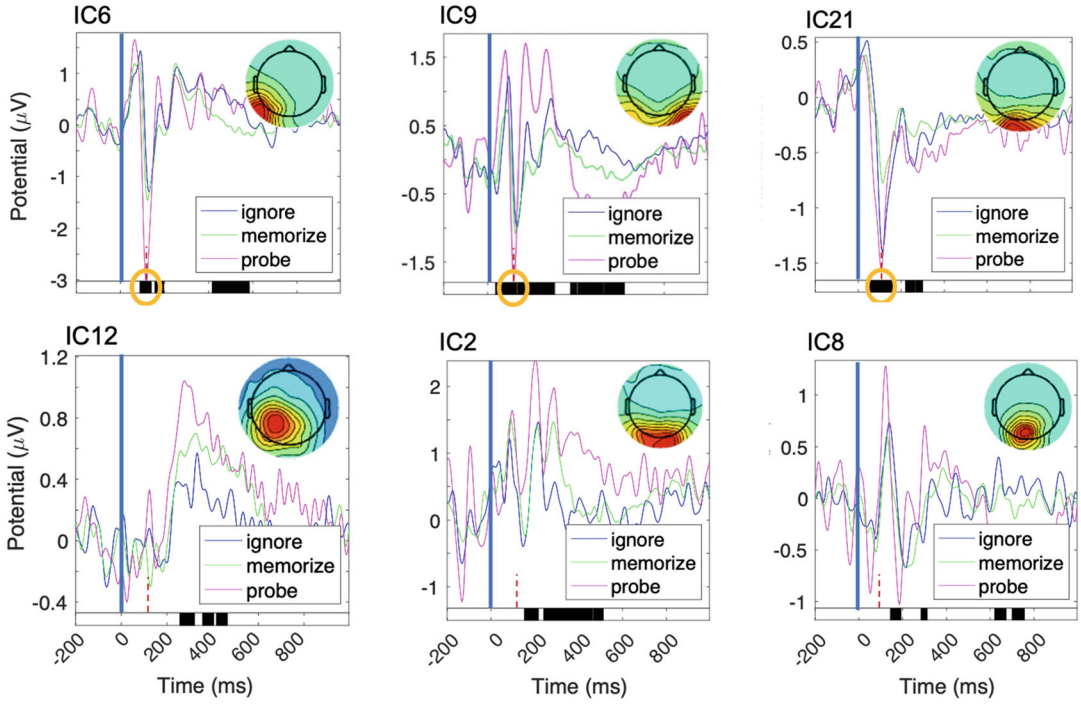

Fig. 14 Independent component (IC) ERPs in the three conditions, with regions of statistical significance for condition differences shown below ($p < 0.01$). Three IC ERPs in the top row (ICs 6, 9, and 21) exhibit pronounced N1 peaks. However, their condition differences differ: for medial occipital IC21 (top right), the N1 peak for letters to be memorized (green traces) is smaller than for the other two letter types, while for the lateral occipital ICs 6 and 9 (top left), the N1 peak in the ERP time locked to probe letters (magenta trace) is stronger than in responses to either the memorize or ignore letters

were executed at each step. This script can then be used for automated analysis exploiting cloud resources as described in the next section.

Figure 14 plots the projections of the six ICs contributing most strongly to the ERPs in the three analysis conditions. Here, each trace represents the trial-averaged IC activations in units of rms microvolts per scalp channel across all channels. (The projection of each IC to each channel ERP is the product of the averaged IC activation time course with the 71 IC scalp map values.)

To perform statistical testing of these results, we made a single-participant EEGLAB Study including the three sets of epoched trial data. We then tested for statistical difference between the three conditions at each ERP latency using methods from the LIMO toolbox [25]. Figure 14 highlights the three IC sources that exhibit a significant condition difference at the "N1" ERP peak (near 120 ms). For two of these ICs (6 and 9), the N1 peak projection is stronger in response to probe letter onsets (magenta). For IC21, the response to target (memorize) letter onsets is weaker than in

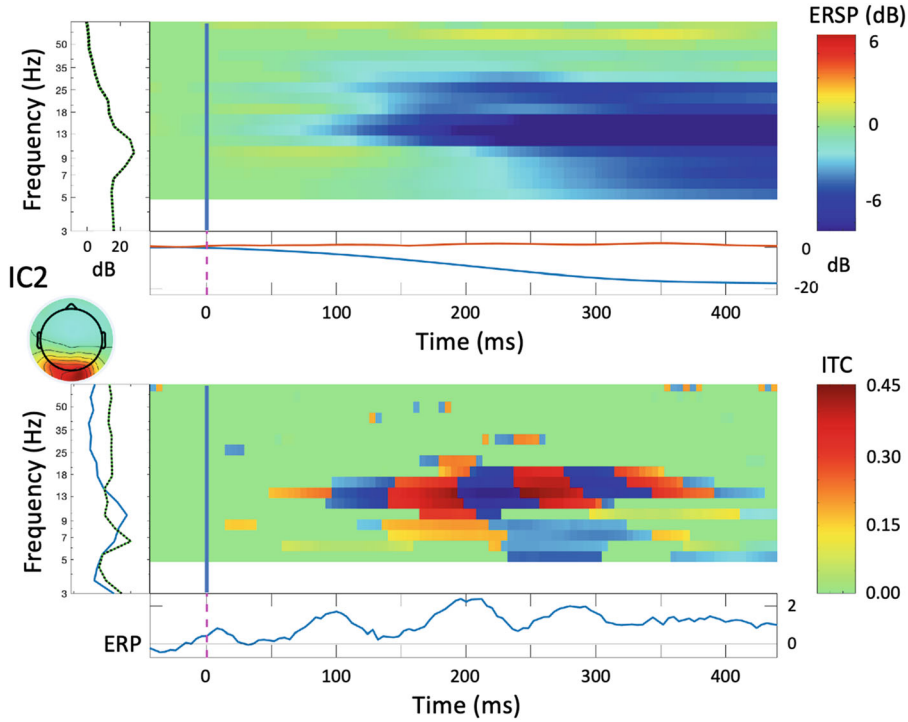

Fig. 15 (Upper panel) Event-related spectral perturbation (ERSP), and (lower panel) Inter-trial coherence (ITC) plots for IC2 epochs time locked to probe letter onsets (at time 0). Figure produced by EEGLAB function *pop_timef*, with later text enhancement for publication. The trial ERP is shown below the ITC plot. See text for further details

the other two conditions. The other three ICs contributing most strongly to the ERPs do not exhibit a peak projection at the N1 peak latency, nor a projection condition effect at that latency.

3.6 Source-Resolved Time/Frequency Analysis

Although ERP measures of event-related brain dynamics dominated EEG research in cognitive psychology and psychiatry for more than 40 years, ERPs alone do not reveal and cannot be used to model every important aspect of event-related EEG brain dynamics. Another set of measures based on time/frequency analysis—measuring mean changes in EEG power and phase spectra time locked to events of interest—reveal information complementary to that revealed by ERPs. Figure 15 shows the trial-mean event-related spectral perturbation (ERSP) [26] and inter-trial coherence (ITC) for IC2 (compatible with a bilateral occipital pole brain source) time locked to probe letter presentation onsets.

The upper panel (ERSP) shows that the upper part (~13 Hz) of the (~10 Hz) alpha peak in the baseline log power spectrum (the end of the memory maintenance period; top panel, left box) is profoundly suppressed (top panel, lower box) during probe stimulus presentation—a nearly 20 dB decrease from the pre-probe baseline period.

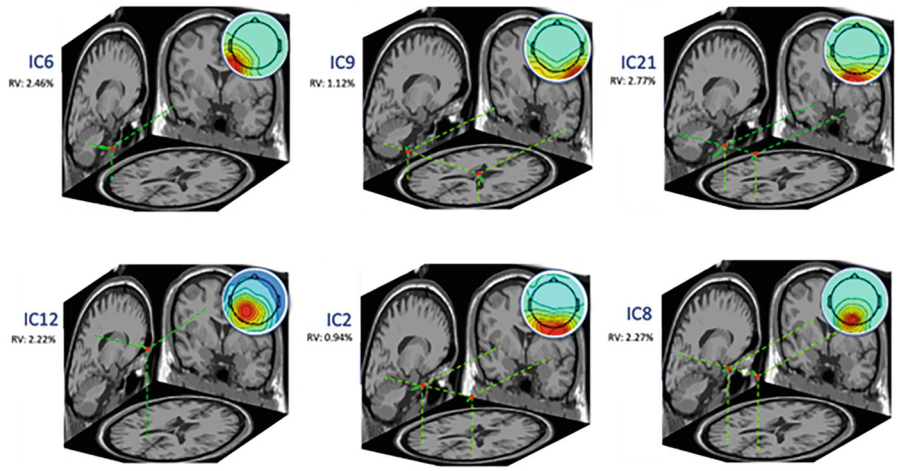

Fig. 16 Equivalent dipole models of source locations for the six sources making largest contributions to the letter presentation ERPs. Here "RV" refers to residual variance in the IC scalp map not explained by the indicated single or bilaterally symmetric dual equivalent dipole model projection to the scalp (Fig. 14). See text for details

The lower panel Fig. 15 shows that while in the pre-stimulus baseline period trial phase with respect to the upcoming stimulus onset is random (green background), phase of remaining activity near 13 Hz (and less strongly so at 7–9 Hz) becomes regularized (blue/red line segments representing negative and positive phase) with respect to stimulus onset. This causes partial failure of phase cancellation across trials, thus producing the trial-mean ERP (bottom panel, lower box) that resembles a ~13-Hz oscillation in the trial-average ERP for this effective brain source component.

The strong low-beta band activity suppression in lateral occipital cortices (cf. Fig. 16) might be tentatively interpreted as ending during and after Probe letter presentations, the suppression of activity in object recognition centers that was encouraged by "alpha flooding" of these areas during task Memory maintenance periods—possibly to enhance object memory retrieval involving these same areas.

3.7 Localizing Sources

The many effective brain sources separated from EEG data by ICA decomposition have a common feature not anticipated nor promoted by the decomposition algorithm itself. They have highly "dipolar" scalp maps, meaning that their scalp maps, learned from the data during decomposition almost exactly resemble the projection of single oriented equivalent dipoles—an imaginary infinitesimal battery located somewhere in the brain volume [19]. In some instances, these "brain IC" scalps may require a dual equivalent dipole model, most often the locations of the two equivalent dipoles (though not their orientations) being nearly bilaterally

symmetric. This is a remarkable observation, given that the ICA algorithm itself is given *no* information about the locations of the electrode channels on the scalp or about the nature of current propagation from the cortex to the scalp through the intervening brain, skull, and skin media. One might put it this way: "ICA algorithms don't know there is a head". Yet many of the "most independent" component sources separated by ICA decomposition are strongly "dipolar".

This can only arise from the neurophysiology of cortical coherence in local field potentials, which can only spread through direct physiological connections in cortical neuropile, which in turn are strongly weighted toward local (<0.1 mm) connections between neurons, in particular inhibitory neurons that play a major role in supporting field dynamics at EEG frequencies. The physiological fact means that synchrony (or near synchrony) in cortical tissue must typically spread out from an origination point (Walter Freeman likened these to "pond ripples"). When field synchrony spreads out from a central field oscillation, its phase near-uniformity across a cortical area gives it far stronger contribution to the electrically distant scalp electrodes than the same area when it is out of synchrony. This is most likely the reason that the projection patterns of effective brain source ICs so often closely resemble the projection patterns of a single equivalent model dipole, e.g., one located at or near the origin of the cortical spreading field pattern and oriented near perpendicularly to the local cortical surface. The origin of the dual-symmetrical dipole models required to fit the scalp maps of other effective brain ICs then likely arises from synchronous field activity patterns arising in two cortical patches that are bidirectionally coupled, either directly by white matter tracts such as *corpus callosum* or possibly indirectly by being driven by strong common input.

Residual variance (RV) in the IC scalp maps learned by ICA decomposition from the data is computed by regressing out the projection of the (best-fitting) equivalent dipole in a (here, template) electrical head model, across all 71 scalp channels and measuring the variance of the residual map values. The ratio of the residual map to the orignal IC scalp map variance is the residual variance (RV). The RVs in Fig. 16 range from 0.94% to 2.27%, values likely no larger than those expected based only on the expected rough fit of the template head model to the participant's actual head (i.e., their exact head geometry and tissue conductance values).

Note that the model equivalent dipole location for IC12 (lower left) is in cortical white matter rather than gray matter. The foremost error in EEG source localization is that created by using a uniform template value for skull conductance, which in fact varies widely across individuals, even across adult individuals. Using an

inaccurate value for skull conductance in the electrical head model used in for source localization will typically drive the implied source location deeper into the head (or less deep) than actual. The SCALE algorithm for using ICA decomposition to estimate the largest source of error in EEG source localization, namely skull conductivity [27], requires an individual MR head image, so could not be used here. The equivalent dipoles for IC9 (top middle) are in the cerebellum; this is likely not their actual locations, as effective sources in the cerebellum of sufficient strength to be captured as an (larger) effective source IC have not so far been demonstrated. Rather, the mislocalization more likely arose here through the projection of the basal posterior bilateral effective source being at the lower edge of the scalp electrode montage and therefore not well enough represented in the IC scalp map, and/or through the use of an incorrect (here) template value in the electrical head model used to compute the equivalent dipole locations.

Note that the *equivalent* dipole (or dual-symmetric dipole) modeling used here does not rest on an assumption that the true sources are infinitesimally small patches of cortex—this would clearly be physiologically impossible. Rather, the attraction of a single (or symmetry-constrained dual) equivalent dipole model is the oft demonstrated fact that the current projecting from an equivalent dipole located in the brain AND of one (or two) cortical patches surrounding the equivalent dipole(s) will be nearly identical. That is, an equivalent dipole is the model dipole whose scalp projection should nearly equal the projection of a cortical patch covering it. EEGLAB now includes a plug-in, the "Neuroelectromagnetic Forward problem head modeling Toolbox" (NFT) [28] to estimate the mean location and cortical surface extent of cortical effective sources learned from the data by ICA decomposition, though these tools could not be applied to these data as they also require availability of an individual MR head image for the participant.

3.8 Processing EEG Data Using High-Performance Computing

We have demonstrated an end-to-end EEGLAB workflow for a BIDS dataset with HED annotation from importing to statistical analysis. Such workflows can be performed locally on a user's own workstation. However, as public sharing of datasets becomes more prevalent and analysis dataset sizes grow, and/or as computational processing applied in EEG brain imaging becomes more intensive, researchers may find their local compute resources too limited for the desired data analyses. Fortunately, online data portals and cloud computing resources are being made publicly available for researchers. The "NeuroElectroMagnetic Archive and compute Resource" (NEMAR) and the Neuroscience Gateway (NSG) are two such facilities being developed and made to function cooperatively to

support the analysis and meta-analysis of human electrophysiology data using publicly available data, tools, and compute resources [*see* Resources].

The joint OpenNeuro/NEMAR/NSG resource recently brought online by a collaboration between the developers of NEMAR, EEGLAB, the Neuroscience Gateway (NSG, http://www.nsgportal.org) [29], and the OpenNeuro neuroimaging data archive (https://openneuro.org) represents, we believe, a new category of open data science facility, a publicly available *integrated data*, *tools*, *and compute resource* (i.e., *datacor*). The NSG compute resource is openly and freely available to researchers working on not-for-profit projects, providing neuroscience community access to multiple software environments that are widely used by neuroscience researchers such as analysis environments (e.g., MATLAB, python, R), and toolsets (e.g., EEGLAB, Open Brain, Freesurfer, TensorFlow). NSG enables users to submit custom analysis scripts running in any of these environments, and making use of users' own – or any NEMAR-hosted toolsets – to direct processing on NSF-supported high-performance computing (aka supercomputer) resources.

Furthermore, NSG provides an easy-to-use web portal-based user environment as well as programmatic access via a REST software interface. Users of the EEGLAB software environment, in particular, may use a set of REST-based EEGLAB tools (*nsgportal*) to launch, monitor, and examine results of NSG jobs directly from the EEGLAB menu window. Using NSG, neuroscientists can thus process and model data, run simulations, and train AI/ML networks on internationally supported high-performance supercomputer networks.

The NEMAR/NSG/OpenNeuro collaboration allows users to bypass the slow and computationally and energywise costly processes of downloading NEMAR datasets of interest, then re-uploading to NSG or elsewhere for analysis. Instead, NSG analysis scripts can directly access data hosted on the NEMAR server, encouraging intensive exploration of NEM datasets that have been made public by their authors on OpenNeuro. NEM data in OpenNeuro are copied to NEMAR where they are further curated and their quality is assessed. These datasets are then made available for user inspection, and visualization within NEMAR and for open-ended analysis via NSG.

The process for users to analyze an NEMAR dataset using NSG is simple: (1) Use the NEMAR.org website data search and visualization tools to identify one or more BIDS-formatted NEM datasets of interest. Then, (2) include the identified dataset control number (s) (example, *ds000123*) in an NSG data processing script. The NSG processing script may use any environment and software tools that NSG supports. (3) When processing is complete, the user will be

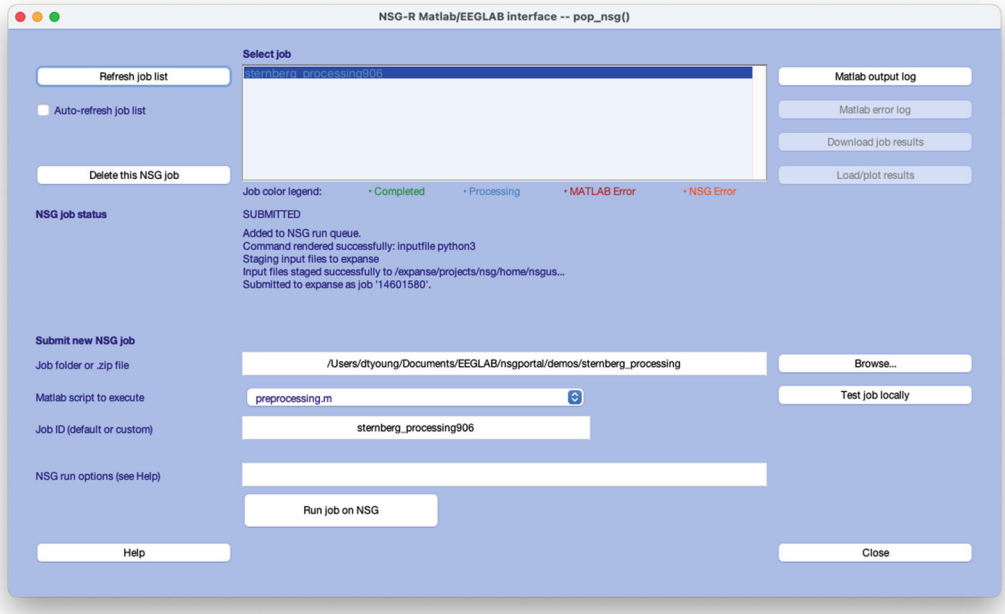

Fig. 17 The nsgportal plug-in GUI in EEGLAB

informed by email that the results specified in the processing script are available by NSG for download. More detailed information can be found at https://nemar.org/nsg_for_nemar.

EEGLAB users can use the *nsgportal* plug-in to directly submit NSG jobs processing NEMAR datasets from their local MATLAB environment [30]. Users place their analysis script in a directory then zip the directory for upload. They can then use the *pop_nsg* GUI shown in Fig. 17 to submit the job zip file, monitor job status in the processing queue, and when completed download results to display or further process locally.

To use the portal, first create a "job directory" containing at least one MATLAB script (the job) and the EEGLAB Study on which the script should be run. In the *Submit new NSG job* section of the GUI browse to that directory. A pull-down menu with the list of potential MATLAB scripts to be run then becomes visible (*preprocessing.m* is shown in Fig. 17). After selecting the script to run and filling in a job name and any desired options, push *Run job on NSG* to submit the job. EEGLAB zips the directory (which can be quite big) if not already zipped and uploads to NSG for execution. The status of the job on NSG is displayed in the top window.

Clearly, uploading an entire BIDS dataset as an EEGLAB Study to run a job is not ideal. The integration of NEMAR with NSG responds to this problem by allowing users, via NSG, to process data originally stored on OpenNeuro and staged on NEMAR, and

```
% Set the NEMAR/OpenNeuro ID of the 24-participant Modified Sternberg dataset
bidsName = 'ds004117';
% Clear the MATLAB workspace
clear

% Open EEGLAB within the NSG session
eeglab;

% Build the NEMAR address of the dataset using the global NEMARPATH variable from NSG MATLAB
filePath = [ getenv('NEMARPATH') bidsName ];   % NEMARPATH points to where NEMAR stores the data

% Open and convert the BIDS dataset to an EEGLAB Study
[STUDY, ALLEEG] = pop_importbids(filePath, 'outputdir', [pwd '/SternbergSTUDY']);

% Output total number of EEG files in the imported Study into an ASCII file
fid = fopen('results.txt', 'w');  % Open a text file for returning dataset information
fprintf(fid,'BIDS dataset %s\n', bidsName);   %  Output the OpenNeuro/NEMAR dataset ID
fprintf(fid,'%d datasets\n', length(EEG));    %  Output the total number of EEG files
fclose(fid);                                  %  Close the file

% ... [Perform some analysis on this Study as desired] ...

% Remove the imported Study directory and its associated files
rmdir('SternbergSTUDY');                      %  To avoid NSG also returning these
```

Fig. 18 Example script for submission to NSG from the EEGLAB nsgportal plug-in to process a BIDS dataset available in NEMAR

to receive and work with results of the processing using local computer resources. When working in MATLAB with a dataset made available on NEMAR, users only need to provide the dataset accession number (an 8-character string beginning with "ds" returned by NEMAR data search tools) and call *getenv("NEMARPATH")* to run their NSG job. The job directory needs only to contain the MATLAB script(s). Figure 18 shows an excerpt of an example of a MATLAB script based on this approach:

Note
The temptation for beginning users of remote resources is to write a processing script and launch a job using it without testing it first on a small dataset. Running scripts by remote execution makes script debugging more difficult than running the same script on the desktop. The MATLAB desktop has excellent facilities for single-stepping through a script or function and observing the data as execution proceeds. It is important to test a script thoroughly on less data before consuming NSG resources to apply it to more data.

4 Conclusions

This chapter covered end-to-end processing of EEG data using standardized BIDS and HED format to organize and describe information about the dataset. We believe that HED will have a crucial role in the use of public data and will be an essential enabler of cross-study and large-scale analysis. It should be emphasized that although BIDS enforces strict formatting standards, which enable tools to run automatically, it currently does not enforce standards for data quality and curation, and also allows some metadata fields to be left empty or with insufficient information. Thus, the researcher using the data must carefully check for inconsistencies and missing data. In the future, we suspect that more investment in careful curation of data to be shared publicly will be required, to better enable automated discovery.

This chapter is supported by extensive online tutorials and documentation. See https://osf.io/8brgv/ for links to the supporting materials and downloadable demo datasets.

Acknowledgments

We would like to acknowledge the seminal vision and contributions of Nima Bigdely-Shamlo, who initially conceived the HED system concept and demonstrated its potential for practical use and importance for data search and analysis within and across studies. We also thank Jonathan Touryan of the Army Research Laboratory and Tony Johnson of DCS Corporation for their work in supporting the development of HED for EEG data sharing. Ian Callanan and Alexander Jones are primary tool developers of the HED supporting infrastructure. This project received support from the Army Research Laboratory under Cooperative Agreement Number W911NF-10-2-0022 (KR) and from NIH projects R01 EB023297-03, R01 NS047293-l4, and R24 MH120037-01 (SM). The Swartz Center for Computational Neuroscience is supported in part by a generous gift from The Swartz Foundation (Old Field, NY).

Appendix: List of Terms Used in This Chapter

AMICA	*Adaptive Mixture Independent Component Analysis*, a powerful ICA algorithm
BIDS	*Brain Imaging Data Structure*—a set of formatting specifications for storing and sharing of neuroimaging data (*bids.neuroimaging.io*)

(continued)

EEGLAB	A software environment for analysis of electrophysiological data running on MATLAB (The Mathworks, Inc.) (eeglab.org)
ERP	*Event-Related Potential*, mean of electrophysiological data trials time locked to events of a specified type
ERSP	*Event-Related Spectral Perturbation*, mean data trial spectrograms, trials time locked to events of a specified type
Event	A process unfolding through time during neuroimaging time series recording, especially any occurring process that may affect the experience and/or behavior of a participant. Also called an *event process*
Event context	The set of ongoing *event processes* at the time point of any *event marker*
Event marker	A pointer from a specified *event process* to a time point on the experiment timeline that marks a critical point, phase transition, or time point of interest within the event process, e.g., marking the event onset or offset. Also called an *event phase marker*
HED	*Hierarchical Event Descriptors*—a system for specifying the nature of events occurring during neuroimaging time series recordings (hedtags.org)
HED Schema	A dictionary of terms for use in *HED tags*, also indicating their allowed syntax
HED Base Schema	The common root *HED schema* of terms in common use across HED annotations (https://www.hedtags.org/display_hed.html)
HED Library Schema	A *HED schema* supplementing the *HED base schema*, adding terms needed to specify the nature of events in some neuroimaging research subfield (for example, language, movement, or clinical diagnosis)
HED String	A comma-separated list of *HED tags* specifying the nature of an event occurring during a neuroimaging experiment
HED Tag	A formatted list of *HED schema terms* giving some fact about the tagged event
IC	*Independent Component* process, identified in data by ICA decomposition
ICA	*Independent Component Analysis*
ITC	*Inter-Trial Coherence*—a measure of the phase angle coherence of a set of trials
NEMAR	*NeuroElectroMagnetic data Archive and compute Resource* (nemar.org)
NSG	*The Neuroscience Gateway*, enabling free research use of a high-performance computing network in neuroimaging research (nsgportal.org)
OpenNeuro	An open neuroimaging data archive using BIDS data formatting (openneuro.org)
RV	*Residual Variance* (percent) variance remaining in data following the removal of some other data
Trial	A task *event process* recurring during a recording, typically encompassing a unit of task performance during which a participant performs some action following one or more sensory and/or action events

References

1. Marek S, Tervo-Clemmens B, Calabro FJ, Montez DF, Kay BP, Hatoum AS, Donohue MR, Foran W, Miller RL, Hendrickson TJ, Malone SM, Kandala S, Feczko E, Miranda-Dominguez O, Graham AM, Earl EA, Perrone AJ, Cordova M, Doyle O, Moore LA, Conan GM, Uriarte J, Snider K, Lynch BJ, Wilgenbusch JC, Pengo T, Tam A, Chen J, Newbold DJ, Zheng A, Seider NA, Van AN, Metoki A, Chauvin RJ, Laumann TO, Greene DJ, Petersen SE, Garavan H, Thompson WK, Nichols TE, Yeo BTT, Barch DM, Luna B, Fair DA, Dosenbach NUF (2022) Reproducible brain-wide association studies require thousands of individuals. Nature 603:654–660. https://doi.org/10.1038/s41586-022-04492-9
2. Gorgolewski KJ, Poldrack RA (2016) A practical guide for improving transparency and reproducibility in neuroimaging research. PLoS Biol 14:e1002506. https://doi.org/10.1371/journal.pbio.1002506
3. Pernet CR, Appelhoff S, Gorgolewski KJ, Flandin G, Phillips C, Delorme A, Oostenveld R (2019) EEG-BIDS, an extension to the brain imaging data structure for electroencephalography. Sci Data 6:103. https://doi.org/10.1038/s41597-019-0104-8
4. Niso G, Gorgolewski KJ, Bock E, Brooks TL, Flandin G, Gramfort A, Henson RN, Jas M, Litvak VT, Moreau J, Oostenveld R, Schoffelen J-M, Tadel F, Wexler J, Baillet S (2018) MEG-BIDS, the brain imaging data structure extended to magnetoencephalography. Sci Data 5:180110. https://doi.org/10.1038/sdata.2018.110
5. Holdgraf C, Appelhoff S, Bickel S, Bouchard K, D'Ambrosio S, David O, Devinsky O, Dichter B, Flinker A, Foster BL, Gorgolewski KJ, Groen I, Groppe D, Gunduz A, Hamilton L, Honey CJ, Jas M, Knight R, Lachaux J-P, Lau JC, Lee-Messer C, Lundstrom BN, Miller KJ, Ojemann JG, Oostenveld R, Petridou N, Piantoni G, Pigorini A, Pouratian N, Ramsey NF, Stolk A, Swann NC, Tadel F, Voytek B, Wandell BA, Winawer J, Whitaker K, Zehl L, Hermes D (2019) iEEG-BIDS, extending the brain imaging data structure specification to human intracranial electrophysiology. Sci Data 6:102. https://doi.org/10.1038/s41597-019-0105-7
6. Bigdely-Shamlo N, Kreutz-Delgado K, Robbins K, Miyakoshi M, Westerfield M, Bel-Bahar T, Kothe C, Hsi J, Makeig S (2013) Hierarchical event descriptor (HED) tags for analysis of event-related EEG studies. In: 2013 IEEE global conference on signal and information processing, pp 1–4
7. Robbins K, Truong D, Appelhoff S, Delorme A, Makeig S (2021) Capturing the nature of events and event context using hierarchical event descriptors (HED). NeuroImage 245:118766. https://doi.org/10.1016/j.neuroimage.2021.118766
8. Robbins K, Truong D, Jones A, Callanan I, Makeig S (2022) Building FAIR functionality: annotating events in time series data using hierarchical event descriptors (HED). Neuroinformatics 20:463–481. https://doi.org/10.1007/s12021-021-09537-4
9. Beniczky S, Aurlien H, Brøgger JC, Hirsch LJ, Schomer DL, Trinka E, Pressler RM, Wennberg R, Visser GH, Eisermann M, Diehl B, Lesser RP, Kaplan PW, Nguyen The Tich S, Lee JW, Martins-da-Silva A, Stefan H, Neufeld M, Rubboli G, Fabricius M, Gardella E, Terney D, Meritam P, Eichele T, Asano E, Cox F, van Emde BW, Mameniskiene R, Marusic P, Zárubová J, Schmitt FC, Rosén I, Fuglsang-Frederiksen A, Ikeda A, MacDonald DB, Terada K, Ugawa Y, Zhou D, Herman ST (2017) Standardized computer-based organized reporting of EEG: SCORE - second version. Clin Neurophysiol 128:2334–2346. https://doi.org/10.1016/j.clinph.2017.07.418
10. Delorme A, Makeig S (2004) EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. J Neurosci Methods 134:9–21. https://doi.org/10.1016/j.jneumeth.2003.10.009
11. Makeig S, Bell A, Jung T-P, Sejnowski TJ (1995) Independent component analysis of electroencephalographic data. In: Advances in neural information processing systems. MIT Press
12. Onton J, Delorme A, Makeig S (2005) Frontal midline EEG dynamics during working memory. NeuroImage 27:341–356. https://doi.org/10.1016/j.neuroimage.2005.04.014
13. Pavlov YG, Adamian N, Appelhoff S, Arvaneh M, Benwell CSY, Beste C, Bland AR, Bradford DE, Bublatzky F, Busch NA, Clayson PE, Cruse D, Czeszumski A, Dreber A, Dumas G, Ehinger B, Ganis G, He X, Hinojosa JA, Huber-Huber C, Inzlicht M, Jack BN, Johannesson M, Jones R, Kalenkovich E, Kaltwasser L, Karimi-Rouzbahani H, Keil A, König P, Kouara L, Kulke L, Ladouceur CD, Langer N, Liesefeld HR, Luque D, MacNamara A, Mudrik L, Muthuraman M,

Neal LB, Nilsonne G, Niso G, Ocklenburg S, Oostenveld R, Pernet CR, Pourtois G, Ruzzoli M, Sass SM, Schaefer A, Senderecka M, Snyder JS, Tamnes CK, Tognoli E, van Vugt MK, Verona E, Vloeberghs R, Welke D, Wessel JR, Zakharov I, Mushtaq F (2021) #EEGManyLabs: investigating the replicability of influential EEG experiments. Cortex J Devoted Study Nerv Syst Behav 144:213–229. https://doi.org/10.1016/j.cortex.2021.03.013

14. Delorme A, Truong D, Martinez-Cancino R, Pernet C, Sivagnanam S, Yoshimoto K, Poldrack R, Majumdar A, Makeig S (2021) Tools for importing and evaluating BIDS-EEG formatted data. In: 2021 10th international IEEE/EMBS conference on neural engineering (NER), pp 210–213
15. Kothe CAE, Jung T-P (2015) Artifact removal techniques with signal reconstruction
16. Chang C-Y, Hsu S-H, Pion-Tonachini L, Jung T-P (2018) Evaluation of artifact subspace reconstruction for automatic EEG artifact removal. Annu Int Conf IEEE Eng Med Biol Soc IEEE Eng Med Biol Soc Annu Int Conf 2018:1242–1245. https://doi.org/10.1109/EMBC.2018.8512547
17. Palmer JA, Makeig S, Kreutz-Delgado K, Rao BD (2008) Newton method for the ICA mixture model. In: 2008 IEEE international conference on acoustics, speech and signal processing, pp 1805–1808
18. Makeig S, Westerfield M, Jung T-P, Enghoff S, Townsend J, Courchesne E, Sejnowski TJ (2002) Dynamic brain sources of visual evoked responses. Science 295:690–694. https://doi.org/10.1126/science.1066168
19. Delorme A, Palmer J, Onton J, Oostenveld R, Makeig S (2012) Independent EEG sources are dipolar. PLoS One 7:e30135. https://doi.org/10.1371/journal.pone.0030135
20. Klug M, Kloosterman NA (2022) Zapline-plus: a Zapline extension for automatic and adaptive removal of frequency-specific noise artifacts in M/EEG. Hum Brain Mapp 43:2743–2758. https://doi.org/10.1002/hbm.25832
21. Mullen T (2012) CleanLine EEGLAB plugin. San Diego CA Neuroimaging Inform Toolsand Resour Clgh NITRC
22. Pion-Tonachini L, Kreutz-Delgado K, Makeig S (2019) ICLabel: an automated electroencephalographic independent component classifier, dataset, and website. NeuroImage 198:181–197. https://doi.org/10.1016/j.neuroimage.2019.05.026
23. Robbins KA, Touryan J, Mullen T, Kothe C, Bigdely-Shamlo N (2020) How sensitive are EEG results to preprocessing methods: a benchmarking study. IEEE Trans Neural Syst Rehabil Eng Publ IEEE Eng Med Biol Soc 28:1081–1090. https://doi.org/10.1109/TNSRE.2020.2980223
24. Pernet CR, Martinez-Cancino R, Truong D, Makeig S, Delorme A (2020) From BIDS-formatted EEG data to sensor-space group results: a fully reproducible workflow with EEGLAB and LIMO EEG. Front Neurosci 14:610388. https://doi.org/10.3389/fnins.2020.610388
25. Pernet CR, Chauveau N, Gaspar C, Rousselet GA (2011) LIMO EEG: a toolbox for hierarchical LInear MOdeling of ElectroEncephaloGraphic data. Comput Intell Neurosci 2011:831409. https://doi.org/10.1155/2011/831409
26. Makeig S, Inlow M (1993) Lapse in alertness: coherence of fluctuations in performance and EEG spectrum. Electroencephalogr Clin Neurophysiol 86:23–35. https://doi.org/10.1016/0013-4694(93)90064-3
27. Akalin Acar Z, Acar CE, Makeig S (2016) Simultaneous head tissue conductivity and EEG source location estimation. NeuroImage 124:168–180. https://doi.org/10.1016/j.neuroimage.2015.08.032
28. Acar ZA, Makeig S (2010) Neuroelectromagnetic forward head modeling toolbox. J Neurosci Methods 190:258–270. https://doi.org/10.1016/j.jneumeth.2010.04.031
29. Sivagnanam S, Yoshimoto K, Carnevale NT, Majumdar A (2018) The neuroscience gateway: enabling large scale modeling and data processing in neuroscience. In: Proceedings of the practice and experience on advanced research computing. Association for Computing Machinery, New York, pp 1–7
30. Martínez-Cancino R, Delorme A, Truong D, Artoni F, Kreutz-Delgado K, Sivagnanam S, Yoshimoto K, Majumdar A, Makeig S (2021) The open EEGLAB portal Interface: high-performance computing with EEGLAB. NeuroImage 224:116778. https://doi.org/10.1016/j.neuroimage.2020.116778