End-to-end processing of M/EEG data with BIDS and HED in EEGLAB

Release 0.0.1

Dung Truong, Arnaud Delorme, Kay Robbins, Scott Makeig

Jun 14, 2022

OUTLINE:

Abstract Data sharing is on the rise. Community efforts to promote common data standard and public data archive projects are first steps towards the possibility of large-scale M/EEG data analysis. It is thus ever more important that the community converge on methods of data reporting and formatting practice, and on methods of sharing and documenting data processing pipelines. HED and BIDS are two community standards that enable data authors to add extensive metadata into stored and shared M/EEG datasets. The EEGLAB toolbox fully supports these standards. Here we provide practical guides on the I/O workflow between BIDS, HED, and the EEGLAB environment, showing how researchers can import BIDS-formatted, HED-annotated datasets into EEGLAB, and how EEGLAB users can export BIDS-compatible, HED-annotated datasets. We will also provide practical guides on preprocessing pipelines that can be applied across datasets, enabling analysis across as well as within archived datasets.

CHAPTER

ONE

INTRODUCTION

CHAPTER

TWO

DATA IO

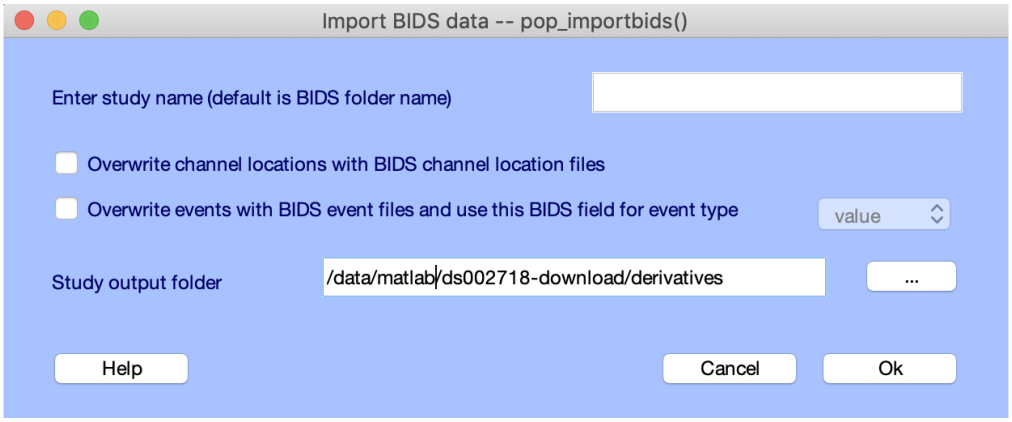
2.1 Getting the data

2.1.1 Getting data from Openneuro/NEMAR

2.1.2 Import data to EEGLAB

Import BIDS dataset using the EEGLAB bids-matlab-tools plug-in

The EEGLAB menu to import a BIDS dataset into an EEGLAB study is fully functional. Screen capture is shown below.



Raw EEG data file often has events. However, BIDS also define events in dedicated event files. Sometimes the BIDS event files contain more information than the raw EEG data file. In that case, users may choose to overwrite raw EEG data events with the event information contained in the BIDS event files.

Similarly, raw EEG data files often define channel labels. However, BIDS also defines channel labels and channel locations in dedicated event files. By pressing the second checkbox, users may choose to use the channel label and location information contained in the BIDS channel definition files.

Finally, users may select an output folder for storing their EEGLAB STUDY. If a folder is not selected, EEGLAB will store STUDY files ‘’in place” which means in the BIDS folder structure - resulting in the BIDS folder becoming non-BIDS compliant and failing to pass BIDS validation because of the additional EEGLAB files.

View HED annotation

HED is fully integrated into EEGLAB via the *HEDTools* plug-in, allowing users to annotate their EEGLAB STUDY and datasets with HED, as well as enabling HED-based data manipulation and processing.

Installing *HEDTools*

HEDTools EEGLAB plug-in can be installed using one of the following ways:

Method 1: EEGLAB Extension Manager:

Launch EEGLAB. From the main GUI select:

File –> Manage EEGLAB extension

The extension manager GUI will pop up.

From this GUI look for and select the plug-in *HEDTools* from the main window, then click into the *Install/Update* button to install the plug-in.

Method 2: Download and unzip

Download the zip file with the content of the plug-in *HEDTools* either from **HED Matlab EEGLAB plugins** or from the EEGLAB **plug-ins summary page**.

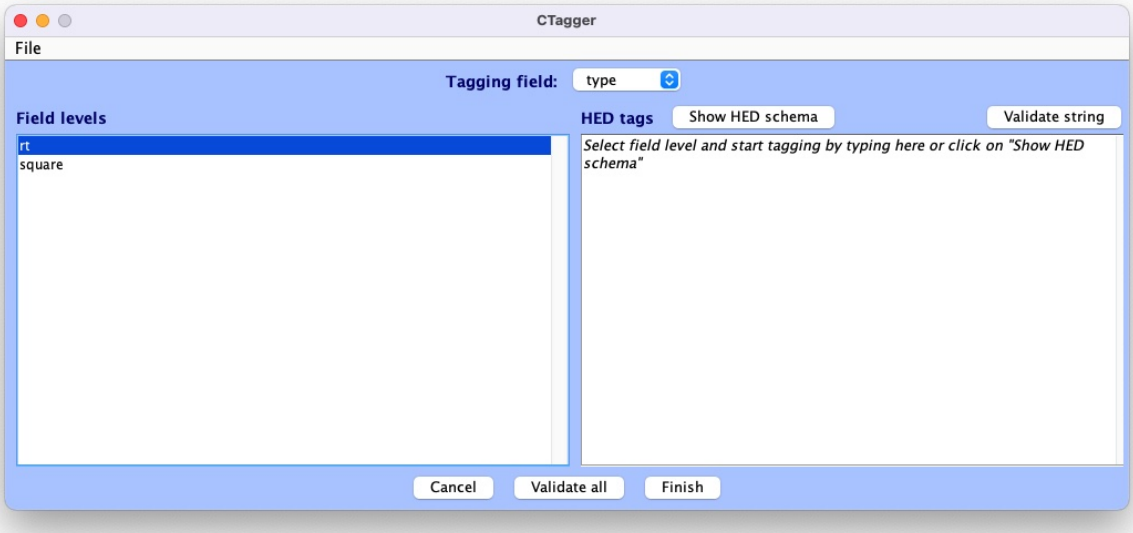
Unzip file into the folder *../eeglab/plugins* and restart the *eeglab* function in a MATLAB session.

Reviewing HED annotations via the EEGLAB HEDTools plug-in

To add and view HED tags for the dataset, from EEGLAB menu, select:

Edit –> Add/Edit event HED tags.

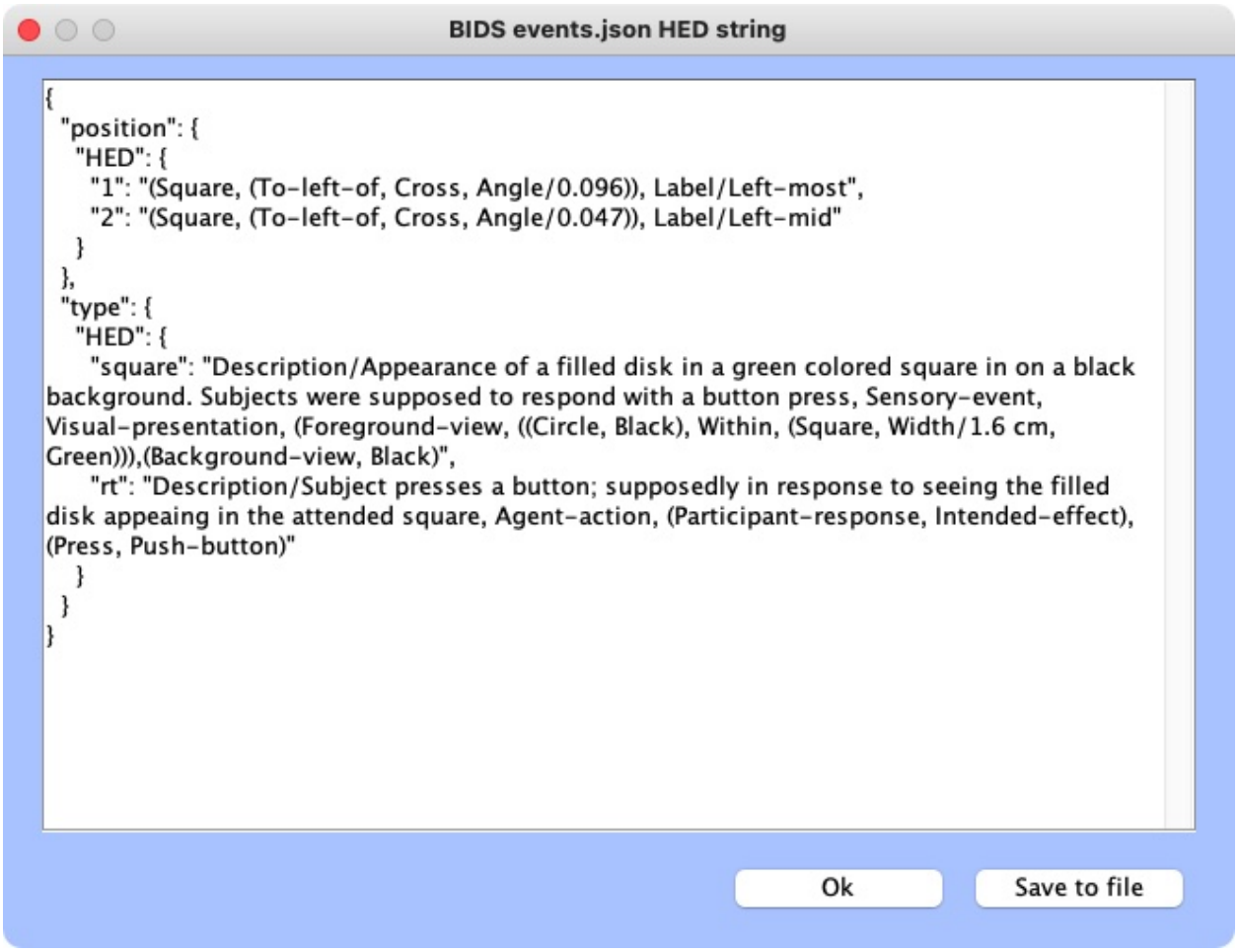
HEDTools will automatically detect HED annotations existed for the dataset imported from BIDS. It will then launch CTagger (for 'Community Tagger'), a graphical user interface (GUI), built to facilitate the process of adding and reviewing HED tags to recorded events in existing datasets.



The CTagger GUI is organized using a split window strategy. The left window shows the items to be tagged, and the right window shows the current HED tags associated with the selected item.

You can now review all the tags via CTagger menu:

File –> Review all tags



View event summary

TODO: Kay's tools?

2.2 From EEGLAB to BIDS

2.2.1 Organize your EEGLAB datasets to STUDY

Creating a STUDY

This part of the tutorial will demonstrate how to create an EEGLAB STUDY and perform simple plotting. An EEGLAB STUDY (or study) contains descriptions of and links to data contained in many epoched or continuous datasets, for example, a set of datasets from a group of subjects in one or more conditions of the same task or performing different tasks in the same or different sessions. We use a *STUDY* to manage and process data recorded from multiple subjects, sessions, and/or conditions of an experimental study.

Creating a new STUDY

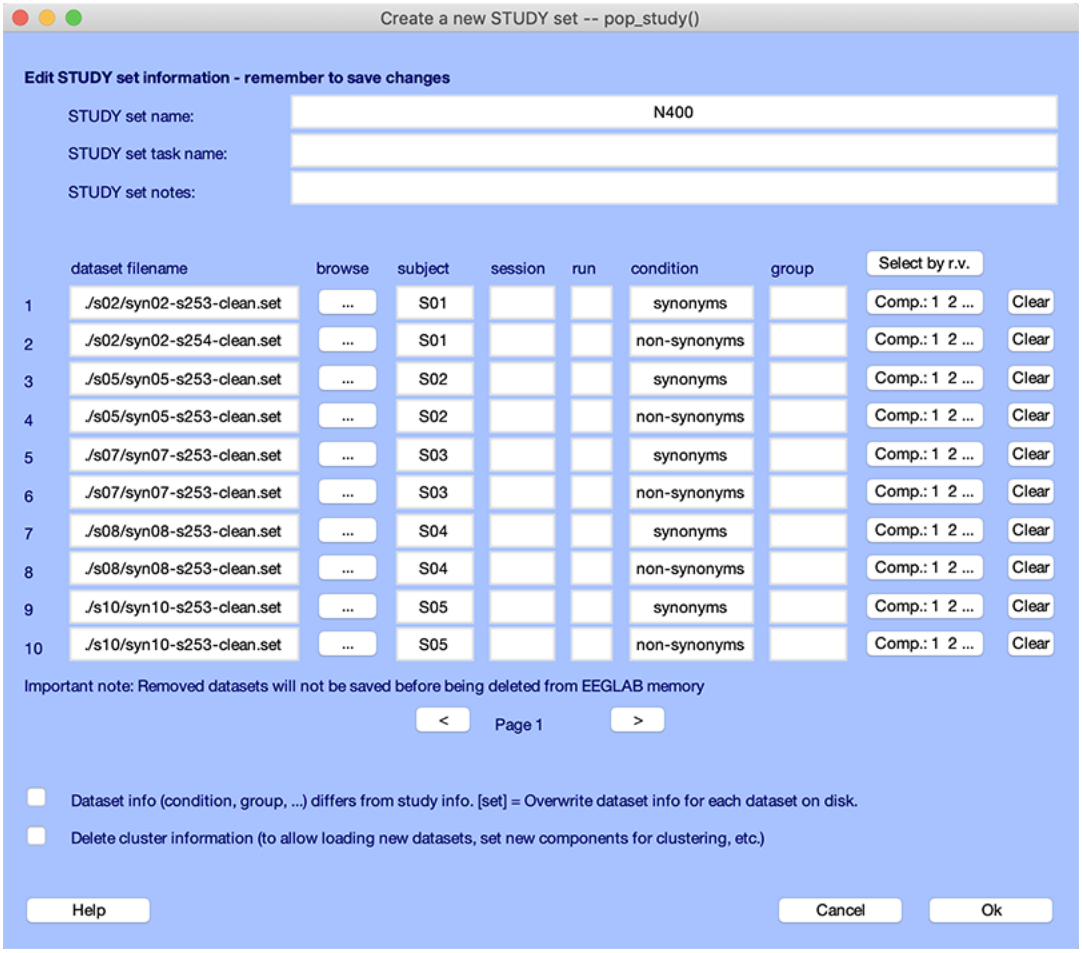
To create a *STUDY*, select the File → Create study → Browse for datasets menu item.

Another option is to load into EEGLAB all the datasets you want to include in the study and select the File → Create study → Using all loaded datasets menu item. A blank interface similar to the one described below will appear. In this window, enter a name for the *STUDY* ('N400'), and a short description of the study ('Auditory task: Synonyms Vs. Non-synonyms, N400').

Here, we do not add notes about the study, but we recommend that you do so for your own studies. The accumulated notes will always be loaded with the study, easing later analyses, and re-analyses. Note that here the fields *Subject* and *Condition* (above) have been filled automatically. This is because the datasets already contained this information. For instance, if you were to load this dataset into EEGLAB by selecting the Edit → Dataset info menu item, you would be able to edit the *subject*, *condition*, *group*, *session*, and *run* for this dataset. You may also edit this information within the study itself. The dataset information and study dataset information may be different to ensure maximum flexibility, although we recommend checking the checkbox *Update dataset info...* to keep them consistent.

Click on the *Browse* button in the first blank location and select a dataset name. Do so for other datasets as well.

The interface window should then look like the following:



Below, we detail what the *STUDY* terms *subject*, *session*, *run*, *condition*, and *group* mean.

- The top of the window contains information about the STUDY, namely its running name, the extended task name for the STUDY, and some notes.

- The next section contains information about the 10 datasets that are part of the STUDY. For each dataset, we have specified a subject code and condition name.
- For each file, you may assign a session and run number. A run is when there are blocks in an experiment, and the data from each block is stored in a separate file. Sessions are used when the data is collected on different days or when there is a break that involves removing the EEG cap. We chose to leave the session and run empty since there are irrelevant for this *STUDY* (there is only one session and one run per subject).
- The *condition* column contains the condition associated with each file. Note that we have two files here per subject. However, it is also possible to have a single file per subject and to define conditions using EEGLAB event trial types. For more information on this topic, read the STUDY design tutorial.
- The *group* column indicates the group a subject belongs to. This is irrelevant for this STUDY since there was only one subject group.
- We will come back later to the *Select by r.v.* (select ICA component by residual variance) and the *Comp...* button when we perform ICA component clustering.
- Pressing the *Clear* button clears the information on a given row.

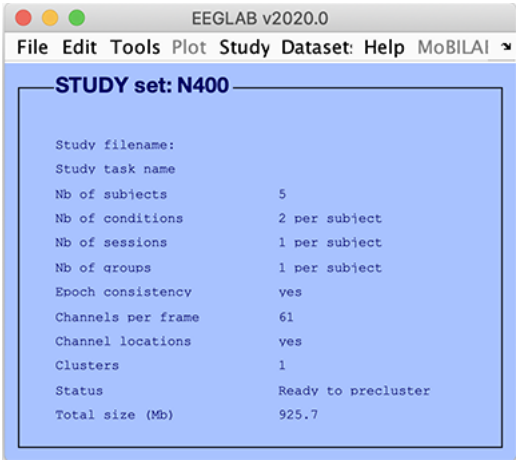
In general, we prefer the dataset information to be consistent with the *STUDY* information – thus, we may check the first checkbox. The second checkbox removes all current cluster information and will be explained when we perform ICA component clustering.

After you have finished adding datasets to the study, press *Ok* in the pop_study.m GUI to import all the datasets.

We strongly recommend that you also save the *STUDY* by selecting the EEGLAB menu item File → Save study as after closing the pop_study.m window.

Loading an existing STUDY

Either use the studyset created in the previous section or load another studyset. To load a studyset, select the File → Load existing study menu item. Select the file *N400.study* in the folder *STUDY5subjects*. After loading or creating a study, the main EEGLAB interface should look like this:



In the EEGLAB GUI (above):

- *Epoch consistency* indicates whether or not the data epochs in all the datasets have the same lengths and limits.
- *Channels per frame* indicates the number of channels in each of the datasets (*It is possible to process datasets with different numbers of channels*).
- *Channel location* indicates whether or not channel locations are present for all datasets.

- *Clusters* indicates the number of component clusters associated with this STUDY. There is always at least one cluster associated with a STUDY. This contains all the pre-selected ICA components from all datasets.
- *Status* indicates the current status of the STUDY. In the case above, this line indicates that the STUDY is ready for pre-clustering.

To list the datasets in the STUDY, use the Study → Edit study info menu item. The interface described in the previous section will pop up.

Editing STUDY datasets

Selecting an individual dataset from the Datasets menu item allows editing individual datasets in a *STUDY*.

Note, however, that creating new datasets or removing datasets will also remove the *STUDY* from memory since the study must remain consistent with datasets loaded in memory (EEGLAB will prompt you to save the *STUDY* before it is deleted).

2.2.2 Add HED annotation using HEDTools plug-in

Event remapping tools

Kay's tools

HED Annotation quickstart

This tutorial takes you through the steps of annotating the events using HED (Hierarchical Event Descriptors). The tutorial focuses on how to make good choices of HED annotations to make your data usable for downstream analysis. The mechanics of putting your selected HED annotations into BIDS (Brain Imaging Data Structure) format is covered in the **BIDS annotation quickstart** guide.

- ***What is HED annotation?***
- ***A recipe for simple annotation***

What is HED annotation?

A HED annotation consists of a comma separated list of tags selected from a HED vocabulary or schema. An important reason for using an agreed-upon vocabulary rather than free-form tagging for annotation is to avoid confusion and ambiguity and to promote data-sharing.

The basic terms are organized into trees for easier access and search. The **Expandable HED vocabulary viewer** allows you to explore these terms.

A recipe for simple annotation

In thinking about how to annotate an event, you should always start by selecting a tag from the *Event* subtree to indicate the general event category. Possible choices are: *Sensory-event*, *Agent-action*, *Data-feature*, *Experiment-control*, *Experiment-procedure*, *Experiment-structure*, and *Measurement-event*. See the **Expandable HED vocabulary viewer** to view the available tags.

Most experiments will only have a few types of distinct events. The simplest way to create a minimal HED annotation for your events is:

1. Select one of the 7 tags from the *Event* subtree to designate the general category of the event.

2. Use the following table to select the appropriate supporting tags given that event type.

Standard HED tag selections for minimal annotation.

Event tag	Support tag type	Example tags	Reason
Sensory-event	*Sensory-presentation*	*Visual-presentationAuditory-presentation*	Which sense?
	Task-event-role	*Experimental-stimulusInstructional*	What task role?
	Task-stimulus-role	*CueTarget*	Stimulus purpose?
	Item	*(Face, Image)Siren*	What is presented?
	Sensory-attribute	*Red*	What modifiers are needed?
Agent-action	*Agent-task-role*	*Experiment-participant*	Who is agent?
	Action	*MovePress*	What action is performed?
	Task-action-type	*Appropriate-actionNear-miss*	What task relationship?
	Item	*ArmMouse-button*	What is action target?
Data-feature	*Data-source-type*	*Expert-annotationComputed-feature*	Where did the feature come from?
	Label	*Label/Blinker_BlinkMax*	Tool name?Feature type?
	Data-value	*Percentage/32.5 Time-interval/1.5 s*	Feature value or type?
Experiment-control	*Agent*	*Controller-Agent*	What is the controller?
	Informational	*Label/Stop-recording*	What did the controller do?
Experiment-procedure	*Task-event-role*	*Task-activity*	What procedure?
Experiment-structure	*Organizational-property*	*Time-blockCondition-variable*	What structural property?
Measurement-event	*Data-source-type*	*Instrument-measurementObservation*	Source of the data.
	Label	*Label/Oximeter_O2Level*	Instrument name?Measurement type?
	Data-value	*Percentage/32.5 Time-interval/1.5 s*	What value or type?

As in BIDS, we assume that the event metadata is given in tabular form. Each table row represents the metadata associated with a single data event marker, as shown in the following excerpt of the `events.tsv` file for a simple Go/No-go experiment. The `onset` column gives the time in seconds of the marker relative to the beginning of the associated data file.

Event file from a simple Go/No-go experiment.

onset	duration	event_type	value	stim_file
5.035	n/a	stimulus	animal_target	105064.jpg
5.370	n/a	response	correct_response	n/a
6.837	n/a	stimulus	animal_distractor	38068.jpg
8.651	n/a	stimulus	animal_target	136095.jpg
8.940	n/a	response	correct_response	n/a
10.801	n/a	stimulus	animal_distractor	38014.jpg
12.684	n/a	stimulus	animal_distractor	82063.jpg
12.943	n/a	response	incorrect_response	n/a

In the Go/No-go experiment, the experimental participant is presented with a series of target and distractor animal images. The participant is instructed to lift a finger off a button when a target animal image appears. Since in this experiment, the `value` column has distinct values for all possible unique event types, the `event_type` column is redundant. In this case, we can choose to assign all the annotations to the `value` column as demonstrated in the following example.

Version 1: Assigning all annotations to the value column.

value	Event category	Supporting tags
animal_target	*Sensory-event*	*Visual-presentation*, *Experimental-stimulus*,*Target*, (*Animal*, *Image*)
animal_distractor	*Sensory-event*	*Visual-presentation*, *Experimental-stimulus*,*Non-target*, *Distractor*, (*Animal*, *Image*)
correct_response	*Agent-action*	*Experiment-participant*, (*Lift*, *Finger*), *Correct-action*
incorrect_response	*Agent-action*	*Experiment-participant*, (*Lift*, *Finger*), *Incorrect-action*

The table above shows the event category and the supporting tags as suggested in the ***Standard hed tags for minimal annotation*** table.

A better format for your annotations is the **4-column spreadsheet format** described in **BIDS annotation quickstart**, since there are online tools to convert this format into a JSON sidecar that can be deployed directly in a BIDS dataset.

4-column spreadsheet format for the previous example.

column_name	column_value	description	HED
value	animal_target	An target animal image waspresented on a screen.	*Sensory-event*, *Visual-presentation*,*Experimental-stimulus*,*Target*, (*Animal*, *Image*)
value	animal_distractor	A non-target animal distractorimage was presentedon a screen.	*Sensory-event*, *Visual-presentation*,*Experimental-stimulus*, *Non-target*,*Distractor*, (*Animal*, *Image*)
value	correct_response	Participant correctlylifted finger off button.	*Agent-action*, *Experiment-participant*,(*Lift*, *Finger*), *Correct-action*
value	incorrect_response	Participant lifted finger offthe button but should not have.	*Agent-action*, *Experiment-participant*,(*Lift*, *Finger*), *Incorrect-action*

HED tools assemble the annotations for each event into a single HED tag string. An exactly equivalent version of the previous example splits the HED tag annotation between the `event_type` and `value` columns as shown in the next

example.

Version 2: Assigning annotations to multiple event file columns.

column_name	column_value	description	HED
event_type	stimulus	An image of an animalwas presented on acomputer screen.	*Sensory-event*, *Visual-presentation*,*experimental-stimulus*
event_type	response	Participant lifted fingeroff button.	*Agent-action*, *Experiment-participant*,(*Lift*, *Finger*)
value	animal_target	A target animal image.	*Target*, (*Animal*, *Image*)
value	animal_distractor	A non-target animal imagemeant as a distractor.	*Non-target*, *Distractor*,(*Animal*, *Image*)
value	correct_response	The previous stimuluswas a target animal.	*Correct-action*
value	incorrect_response	The previous stimuluswas not a target animal.	*Incorrect-action*
stim_file	n/a	Filename of stimulus image.	(*Image*, *Pathname/#*)

In version 2, the annotations that are common to all stimuli and responses are assigned to `event_type`. We have also included the annotation for the `stim_file` column in the last row of this table.

The assembled annotation for the first event (with onset 5.035) in the ***event file excerpt from go/no-go*** above is:

> *Sensory-event*, *Visual-presentation*, *Experimental-stimulus*, *Target*, (*Animal*, *Image*), (*Image*, *Pathname/105064.jpg*)

Mapping annotations and column information across multiple column values often makes the annotation process simpler, especially when annotations become more complex. Multiple column representation also can make analysis easier, particularly if the columns represent information such as design variables.

See **BIDS annotation quick start** for how to create templates to fill in with your annotations using online tools. Once you have completed the annotation and converted it to a sidecar, you simply need to place this sidecar in the root directory of your BIDS dataset.

Add HED Annotation in EEGLAB

HED is fully integrated into EEGLAB via the *HEDTools* plug-in, allowing users to annotate their EEGLAB STUDY and datasets with HED, as well as enabling HED-based data manipulation and processing.

Installing *HEDTools*

HEDTools EEGLAB plug-in can be installed using one of the following ways:

Method 1: EEGLAB Extension Manager:

Launch EEGLAB. From the main GUI select:

File –> Manage EEGLAB extension

The extension manager GUI will pop up.

From this GUI look for and select the plug-in *HEDTools* from the main window, then click into the *Install/Update* button to install the plug-in.

Method 2: Download and unzip

Download the zip file with the content of the plug-in *HEDTools* either from **HED Matlab EEGLAB plugins** or from the EEGLAB **plug-ins summary page**.

Unzip file into the folder *../eeglab/plugins* and restart the *eeglab* function in a MATLAB session.

Annotating datasets

We will start by adding HED annotations to the EEGLAB tutorial dataset.

After installing the *HEDTools* open the EEGLAB main window and load the dataset by selecting the menu item:

File –> Load existing dataset .

Selecting the tutorial dataset under your eeglab installation *eeglab/sample_data/eeglab_data.set*.

Read a description of the dataset and of its included event codes by selecting:

Edit –> About this dataset:

The description gives a general idea of the codes found in the event structure. Yet, inquisitive researchers interested in the nature of the stimuli (e.g., color and exact location of the squares on the screen) would have to look up the referenced paper for details.

Our goal in using HED tags is to describe the experimental events that are recorded in the *EEG.event* data structure in sufficient detail that anyone using the dataset in the future will not need to find and read a separate, detailed description of the dataset or study to understand the recorded experimental events. As demonstrated below, such annotation will allow us to extract epochs using meaningful HED tags instead of the alpha-numeric codes often associated with shared EEG data.

Launching EEGLAB HEDTools

To add and view HED tags for the dataset, from EEGLAB menu, select:

Edit –> Add/Edit event HED tags.

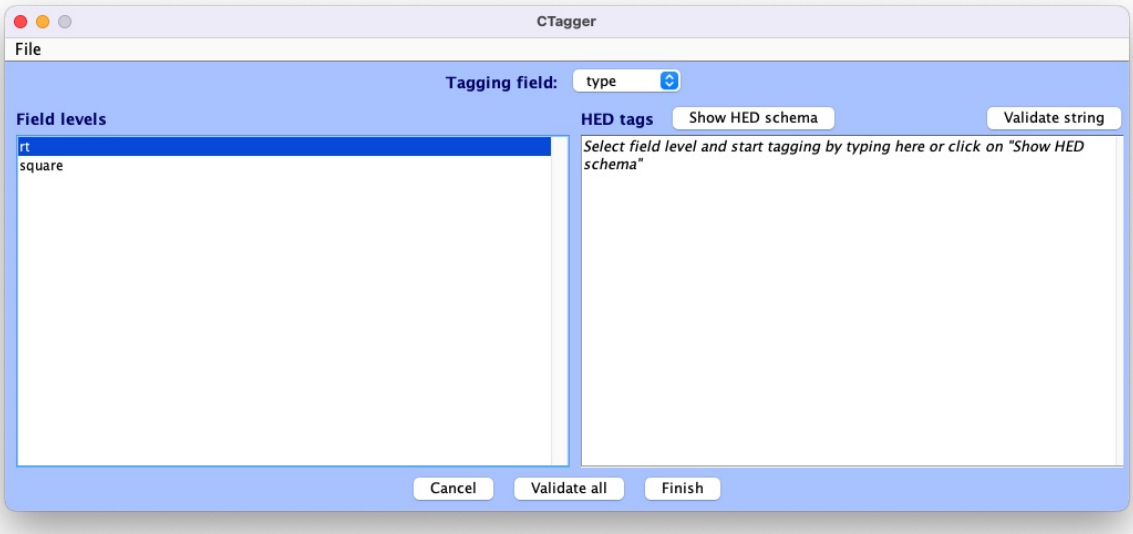
HEDTools will extract information from the *EEG.event* structure, automatically detecting the event structure fields and their unique values.

The *HEDTools* ignore the fields the event structure fields *.latency*, *.epoch*, and *.urevent*.

A window will appear asking you to verify/select categorical fields:

Here both *position* and *type* are categorical fields. *HEDTools* automatically selects fields with less than 20 unique values to be categorical, but the user can modify which values are chosen.

CTagger (for 'Community Tagger') is a graphical user interface (GUI) built to facilitate the process of adding HED tags to recorded events in existing datasets. Clicking *Continue* brings up the *CTagger* interface:



The CTagger GUI is organized using a split window strategy. The left window shows the items to be tagged, and the right window shows the current HED tags associated with the selected item. The *Show HED schema* button brings up a browser for the HED vocabulary.

Through the CTagger GUI, users can explore the HED schema, quickly look up and add tags (or tag groups) to the desired event codes, and use import/export features to reuse tags on from other data recordings in the same study.

The process of tagging is simply choosing tags from the available vocabulary (using the HED schema browser) and associating these tags with each event code.

Once familiar with HED and the vocabulary, most users just type the tags directly in the tag window shown on the right.

CTagger is used as part of the HEDTools plug-in in this tutorial, but it can also be used as a standalone application.

Instructions on downloading and using the standalone version of CTagger, as well as step-by-step guide on how to add HED annotation with CTagger, can be found at in **Tagging with CTagger**.

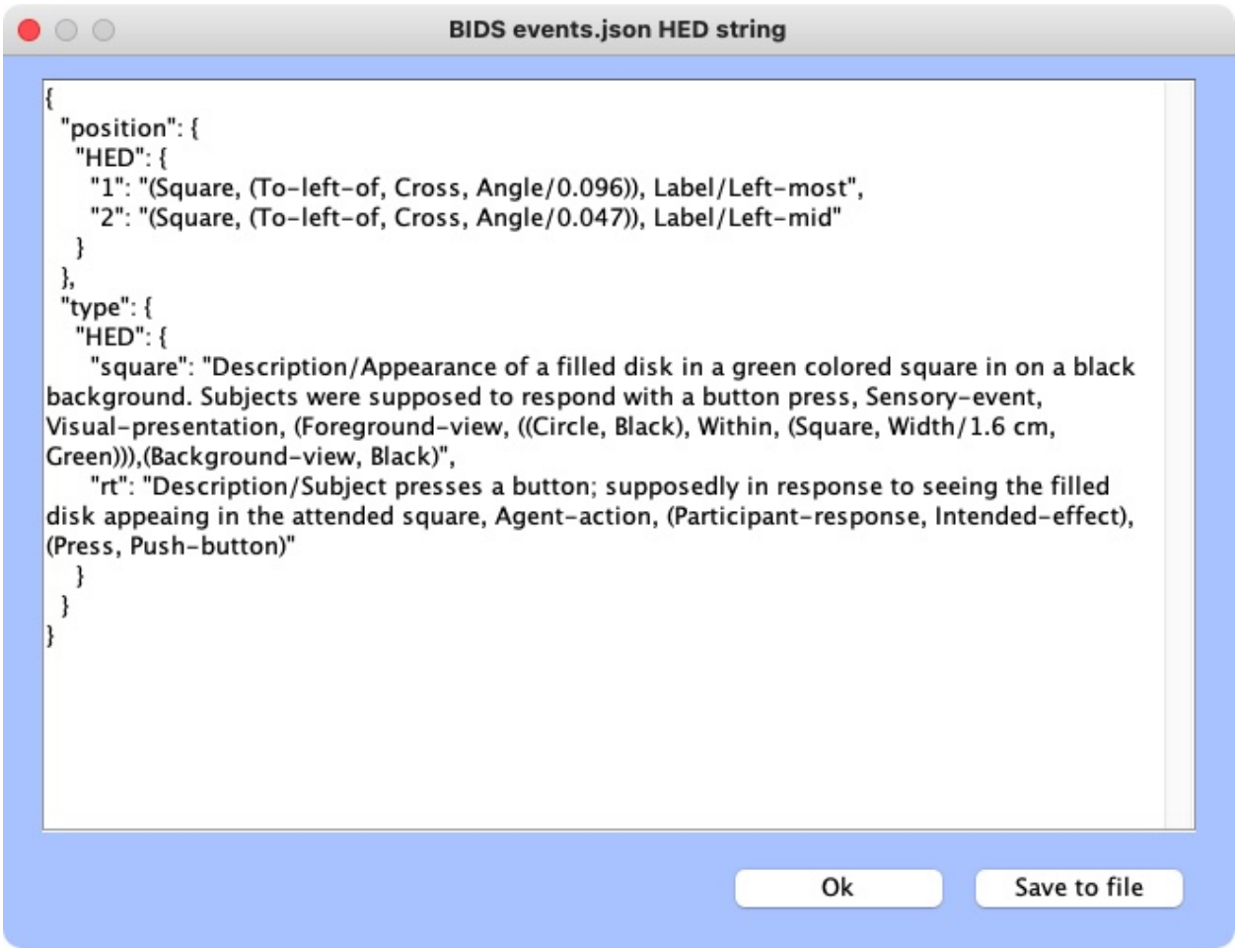
Tagging the events

A brief step-by-step guide to selecting tags can be found at **HED annotation quickstart**. The following shows example annotations using the process suggested in the quickstart. we will import the annotation saved in the _events.json file format. Download the file **eeglab-tutorial_events.json** then select:

File –> Import –> Import BIDS events.json file

to import it to CTagger. You can now review all the tags via:

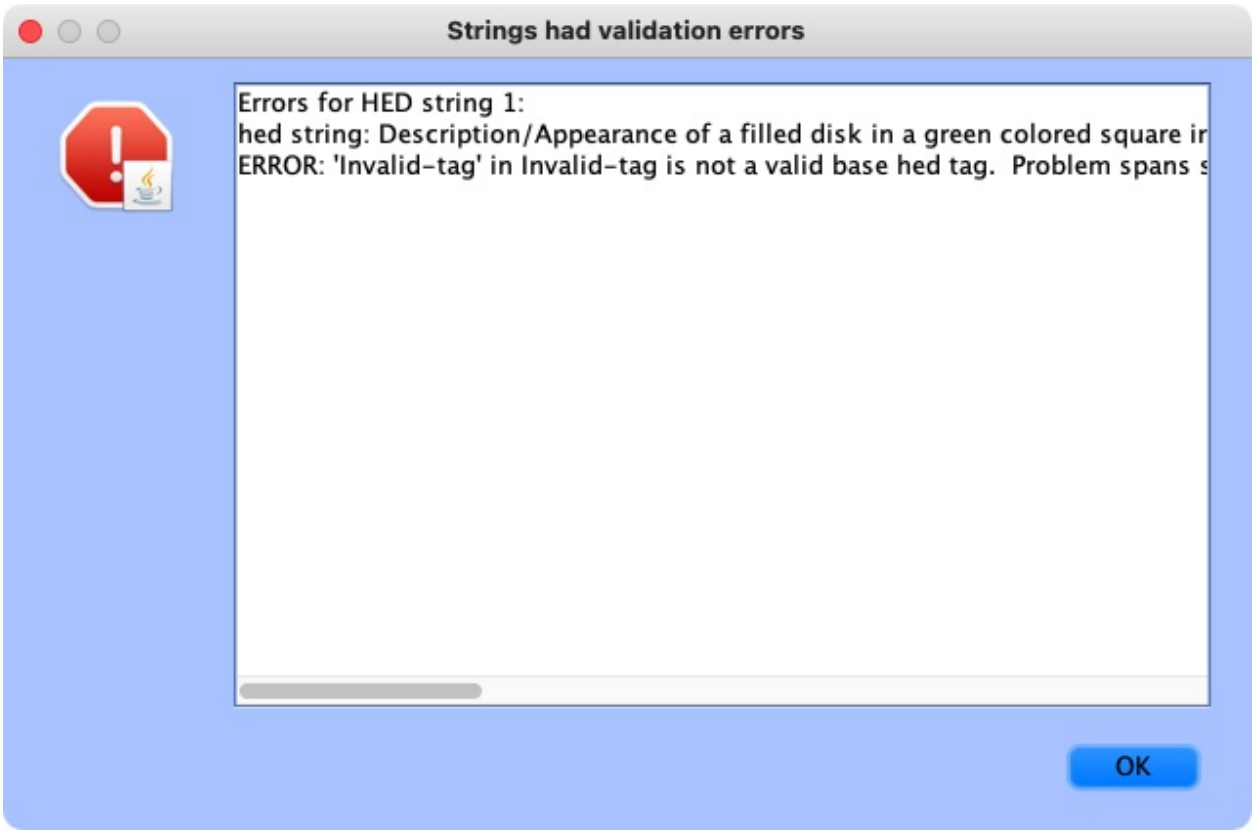
File –> Review all tags



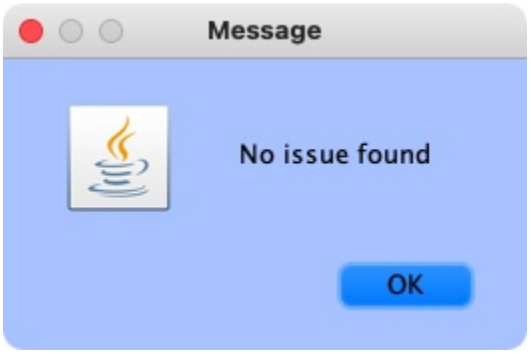
Validation

The last step of the annotation process is to validate the HED annotations. Click on the *Validate all* button at the bottom pane. A window will pop up showing validation results. If there are issues with the annotation, there will be a line for each of the issues found.

Here is an example of validation log file with issues:



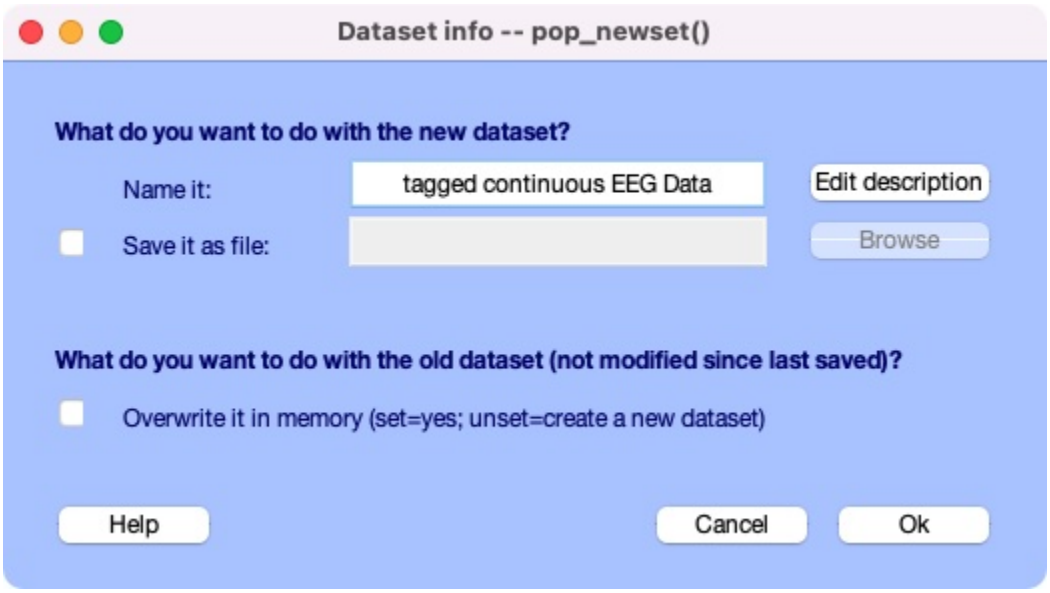
If the annotation was correct, a message will appear confirming the validity:



Click *Finish* on the main CTagger window to end the annotation.

The tag review window will show up again for a final review and the option to save the annotation into an _events.json file for distribution just as with the *eeglab-tutorial_events.json*. Hit *Ok* to continue after that.

A last window will pop up asking what you would like to overwrite the old dataset with the tagged one or save new dataset as a separate file. Click **Ok** when you're done.

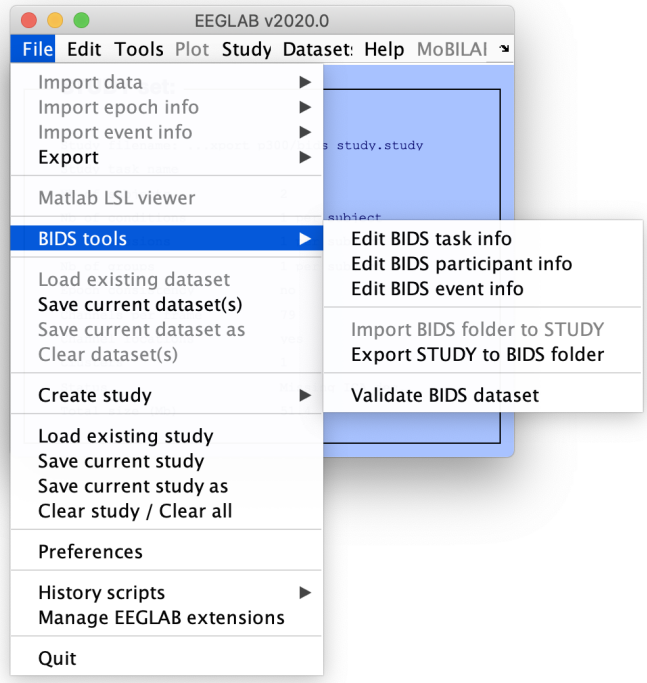


You just finished tagging! *HEDTools* generates the final HED string for each event by concatenating all tags associated with the event values of that event (separated by commas). The final concatenated version is put the string in a new field **HED** in EEG.event.

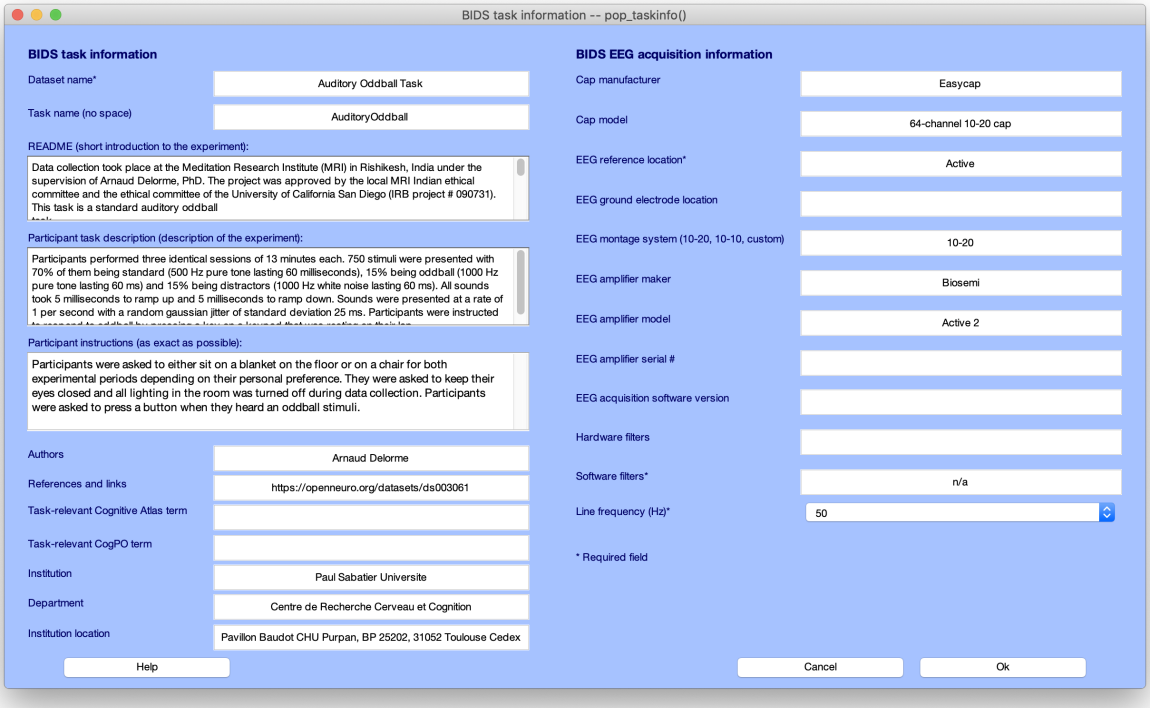
2.2.3 Add BIDS metadata using bids-matlab-tools plug-in and export

Export datasets to BIDS from the graphic interface

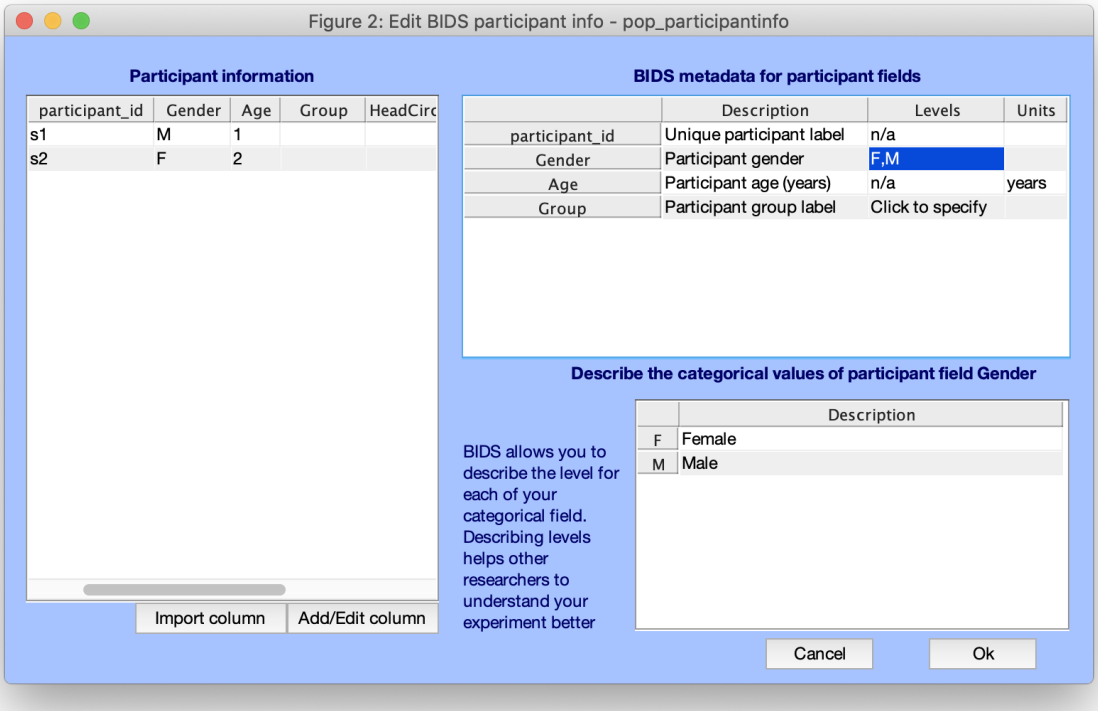
After installing the bids-matlab-tools plugin, the BIDS menu should appear in the EEGLAB File menu item as shown below.



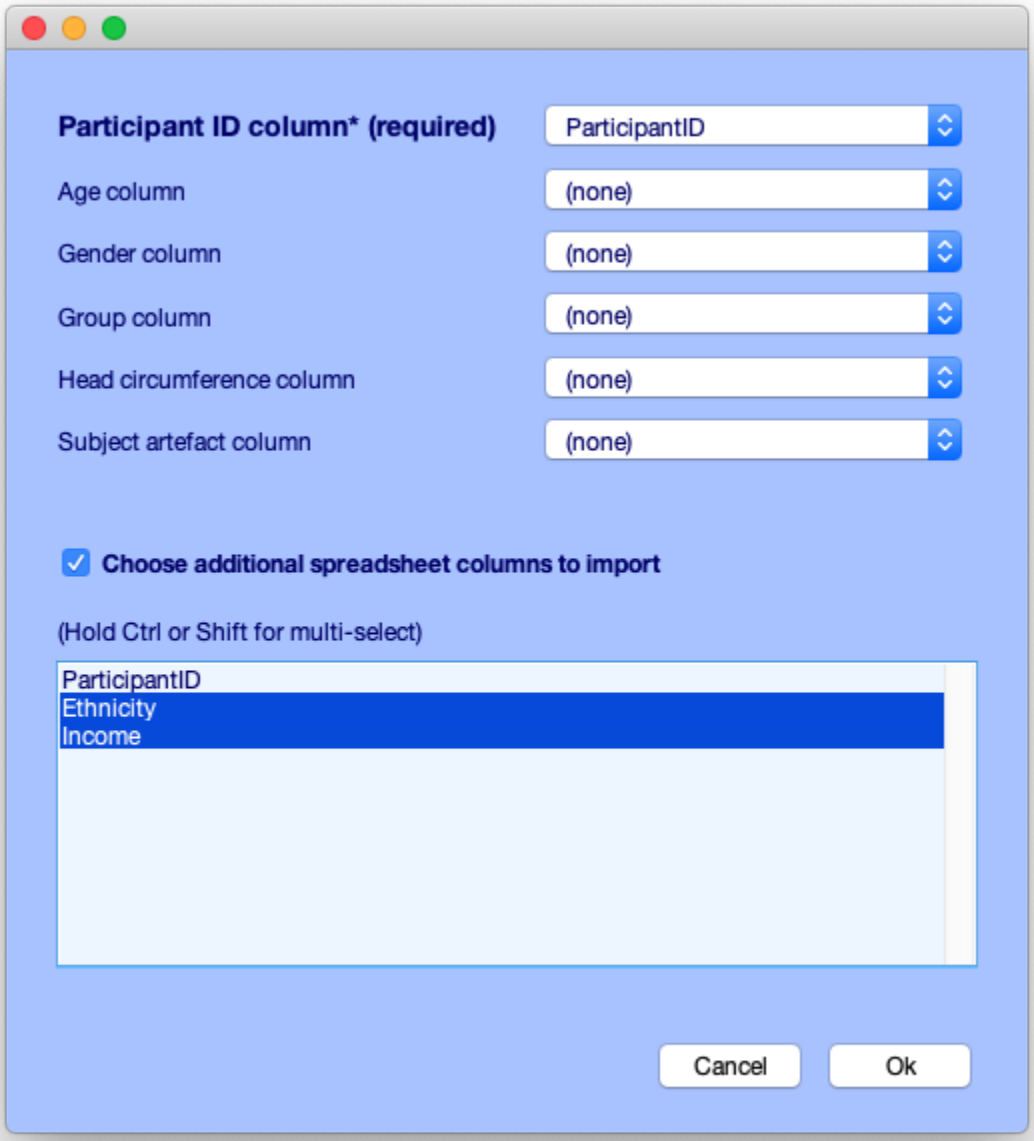
We will go through the menus one by one. First the task menu. This is where you enter all the information about the task and amplifier information.



Once you are done entering task information, you can select the participant menu. There are as many rows as subjects. For each subject, you can specify age, gender, group, etc… Only the participant_id column is mandatory and other columns are optional.

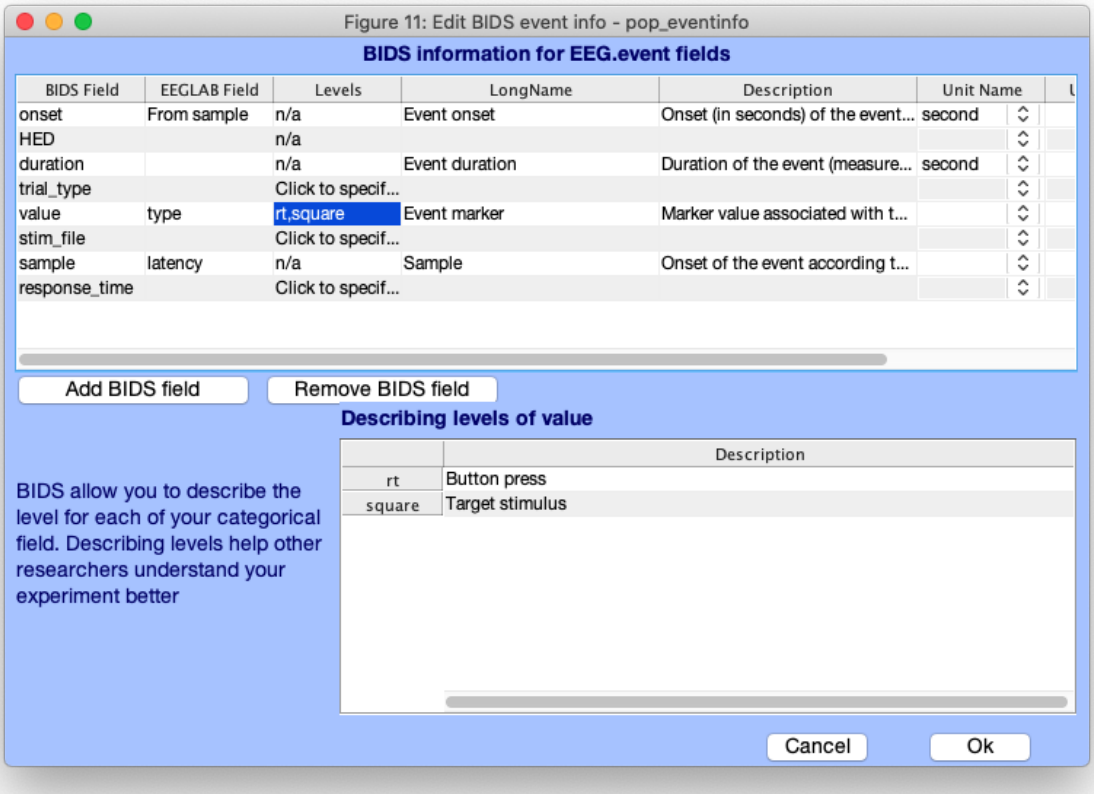


Note that it is also possible to define custom columns. To do this, you can click on the "import column(s)" push button. Upon doing so, you are prompted to select a text or Excel file. Note that the first row must contain column names. After selecting the file, the following interface pops up.



You must indicate which column contains participant ID. You can also select columns for the pre-defined BIDS participant information, such as age, gender, etc… Finally, it is possible to select columns which will be added to the BIDS participant file, in this case "ethnicity" and "income".

Lastly, we fill out information about events. The GUI displays default BIDS event fields to which you can associate the corresponding EEGLAB event field in your data. *onset*, *value*, and *sample* are mapped to their corresponding EEGLAB field by default. *duration* and *HED* will also be automatically mapped if their corresponding EEGLAB field (*duration* and *usertags* respectively) exist. Custom fields can be added via the "Add BIDS field" button. Fields can also be removed via the "Remove BIDS field" button. Note that any BIDS field not mapped to EEGLAB event field will not be saved for later export.



Export datasets to BIDS from the command line

Exporting a collection of datasets to BIDS can also conveniently be done from the command line. A documented example script bids_export_example.m and bids_export_example2.m (more recent) are provided. You may modify these scripts for your own purpose. The help message of the function bids_export.m also contains information on how to export data in BIDS format.

Note for exporting:

- Several tasks may be included in a single BIDS container. In this case, the name of the main task can be "mixed tasks" and multiple EEG datasets for different tasks can be contained in a single subject/eeg/ folder.
- Channel electrodes should not be exported by default if they are template electrode positions. This is because BIDS is about raw data. Electrode positions based on templates (averages) should therefore not be included.

2.2.4 Upload to Openneuro

CHAPTER

THREE

STANDARD PREPROCESSING STEPS/PIPELINES USING EEGLAB

3.1 Standard preprocessing steps/pipelines using EEGLAB

3.2 Think about scaling

3.2.1 Combining multiple BIDS datasets how-to

3.2.2 Compare data across datasets

CHAPTER

FOUR

ANALYSIS

CHAPTER

FIVE

DISCUSSION AND CONCLUSIONS