Batch Electroencephalography Automated Processing Pipeline (BEAPP) User Guide

Version 5.0 Beta

August 2024

Table of Contents

What is BEAPP?

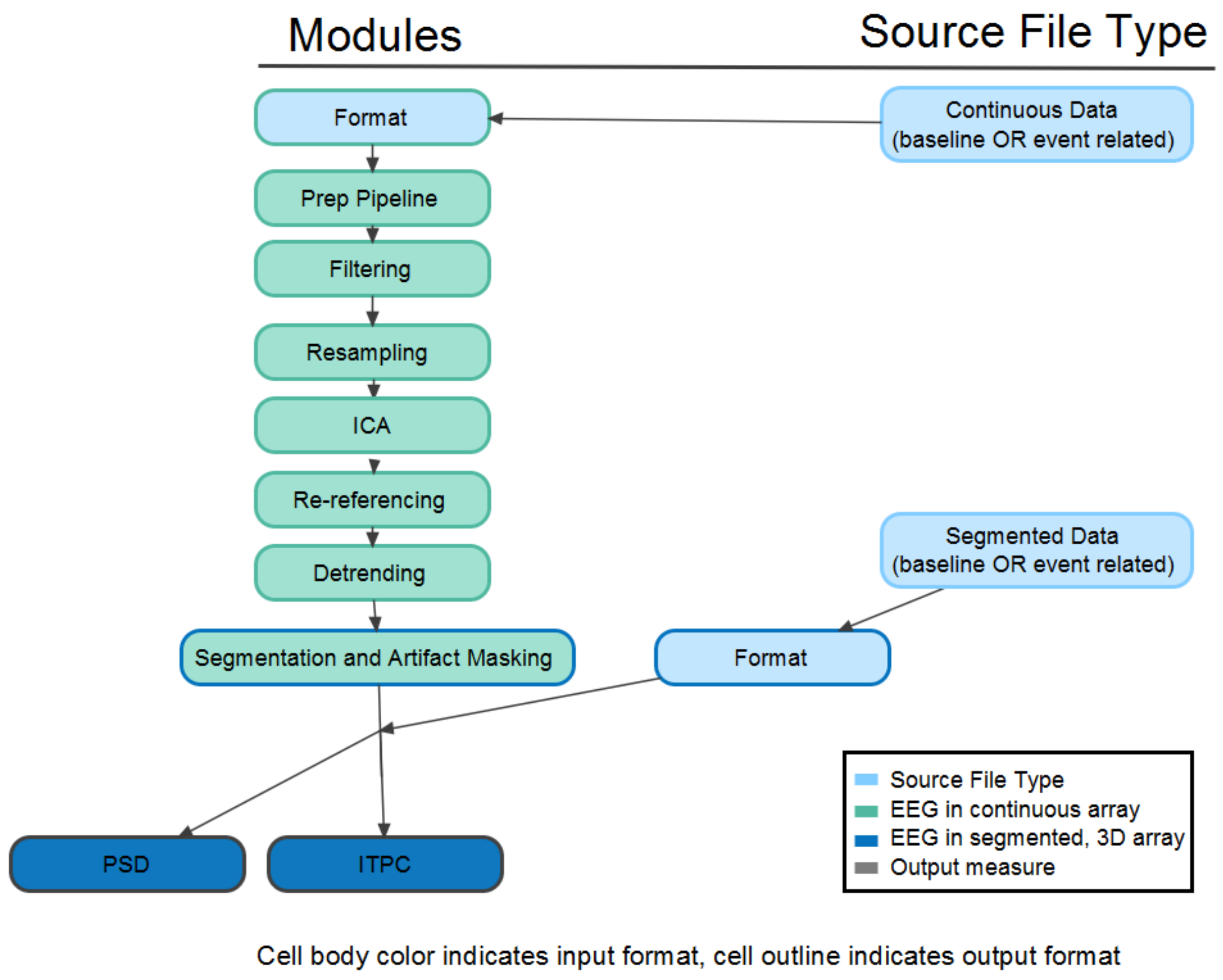
The Batch Electroencephalography Automated Processing Pipeline (BEAPP) is a modular, MATLAB-based software designed to facilitate automated, flexible batch processing of baseline and event-related electroencephalography (EEG) files in datasets with mixed acquisition formats.

Rather than prescribing a specified set of EEG processing steps, BEAPP allows users to choose from a menu of options. Each option can be turned on or off, and options turned "on" can be tailored to fit the user's needs. BEAPP currently provides options for the following user-controlled modules:

1. Format
    a. Convert EEG file to .mat file
2. PREP Pipeline
    a. Line noise removal, interpolation of bad channels, robust average referencing
3. Filtering
    a. Low-pass
    b. High-pass
    c. Notch
    d. CleanLine
4. Resampling
5. Independent Components Analysis (ICA)
    a. ICA only
    b. ICA with Multiple Artifact Rejection Algorithm (MARA)
    c. HAPPE Pipeline
        i. 1-250 Hz band-pass filter
        ii. Select desired channels of interest (e.g. 10-20 channels)
        iii. CleanLine to remove line noise
        iv. Bad Channel Rejection
        v. Wavelet cleaning via wavelet-thresholding ICA (w-ICA)
        vi. ICA with MARA
        vii. Interpolate bad channels
        viii. Average re-reference
6. Re-Referencing
    a. Average re-referencing
    b. Current Source Density Laplacian (CSDLP)
    c. Re-reference to individual or subset of electrodes
    d. Reference Electrode Standardization Technique (REST) Toolbox
7. Detrending
    a. Mean
    b. Linear
    c. Kalman
8. Segmentation
    a. Stimulus-locked (for task-related data)
    b. Non-stimulus-locked (for continuous or "resting" data)
    c. Amplitude-based artifact segment rejection

    d. Join probability segment rejection
9. Power Spectral Decomposition (PSD)
10. Inter-Trial Phase Coherence (ITPC)
11. Fitting Oscillations & One Over F (FOOOF)
12. Phase-Amplitude Coupling (PAC)
13. Bycycle

BEAPP aims to strike a balance between assuming only a basic level of MATLAB and EEG signal processing experience, while also offering a flexible menu of opportunities for more advanced users. At a minimum, no programming experience is required to use BEAPP, but basic familiarity with troubleshooting in MATLAB will likely come in handy.


Modules
Source File Type
Format
Continuous Data (baseline OR event related)
Prep Pipeline
Filtering
Resampling
ICA
Re-referencing
Detrending
Segmented Data (baseline OR event related)
Segmentation and Artifact Masking
Format
PSD
ITPC
Source File Type
EEG in continuous array
EEG in segmented, 3D array
Output measure
Cell body color indicates input format, cell outline indicates output format

Back (User Inputs Overview)
The figure above depicts BEAPP's available modules and the order in which data can run through them. The interior color of each box represents the type of input data for that module module. The border color of each box represents the type of output data for that module. A key is provided in the lower right corner.

Next Steps:
BEAPP is intended to be a dynamic, rather than static, platform for EEG processing. This means that we plan to continue adding additional functionality over time, and we encourage other users to add functionality as well.

What's on Our Wishlist (coming soon):

1. Improved GUI for user inputs
2. Formatted dataset-wide run reporting (general dataset statistics, formatted warnings in a report)
3. Ability to read. bdf/. edf files
4. Coherence
5. Phase lag index
6. Topoplotting outputs with mixed source acquisition layouts/ number of channels
7. Ability to change the order of modules

Start-Up Guide

Back (GUI) Back (GUI Format) Back (SRC EEG Name)

This brief start-up guide provides a basic overview of how to use BEAPP through both the 1) scripting interface and the 2) graphical user interface (GUI), for a beginning user. Details on modules and settings are provided later in the guide.

1. Make sure you have the software necessary for running BEAPP:

- **MATLAB (recommended 2016a or newer)**
  - BEAPP was written in MATLAB 2016a. Older versions of MATLAB may or may not support certain functions used in BEAPP.
- **BEAPP**
  - **Cloning versus downloading?**
    - Cloning is recommended for users familiar with terminal/command line arguments and revision systems, instructions can be found here. Cloning allows changes you make to BEAPP to be tracked by git.
    - Otherwise we recommend that you download the code. Once downloaded, unzip the BEAPP folder in any location you desire. The folder does not need to be in a specific place in order to run BEAPP.
  - BEAPP can be downloaded or cloned from GitHub here.
  - After downloading, open MATLAB, click on Home at the top of the screen. In the Environment section, click on Set Path.
    - Click on Add Folder…
    - Then navigate to where your BEAPP folder is located. Click on the beapp folder and then click Select Folder.
    - Click Save, then Close
    - This will add the BEAPP folder to your path, which will allow you to run BEAPP even if your current folder is not the beapp folder. If you don't do these steps, you'll need to navigate to the beapp folder before running BEAPP
- **Module-specific software dependencies**
  - **ICA** - Open EEGLAB in MATLAB and select the correct settings
  - **PAC/FOOOF/Bycycle** - Requires Python and additional installers provided in beapp\installers

2. Place all the EEG files you plan to process into a single folder on your computer. Do not place them in the BEAPP directory. BEAPP can currently handle the following types of files:

- **.mat files**
  - These may be exported from EGI, Biosemi, ANT, or other EEG data platforms. Each .mat file must contain a variable with the (unsegmented) EEG data, in matrix format. Each row in the matrix should contain the amplitudes of the EEG tracing for a given channel across time. Each column should contain the amplitudes of the EEG tracing for a given time point across channels. The name of this variable can be whatever you typically use in your exports but should be consistent across files. Currently, BEAPP is only able to handle continuous data

from .mat files; if you would like to analyze event-tagged data, you will need your files in .mff or .set format.

- **.mff files**
  - These files are typically exported from NetStation (EGI).
- **.set files**
  - EEGLAB file exports (continuous/unsegmented data only). If these will be used to process events, they will need latency information (see samples).
- For additional information on source file formats, see Running BEAPP with Different Source File Formats.

3. Create a File Info table with the necessary information for each EEG file. Templates for these tables can be found in beapp\user_inputs. Instructions can be found here and example scripts for generating them can be found in beapp\reference_data\example_scripts. The requirements for each file info table will change depending on the user's file type.

- **To run .mat files, the BEAPP file info table:**
  - Must include values for filenames, sampling rates, and net type names.
  - Can include line noise frequencies.
- **To run non-mat files, the BEAPP file info table:**
  - Only needs to include the filenames and the values that vary by file.

4. Choose which interface to use to process your EEG data. There are two ways – the user input script (recommended) or the GUI. The GUI is **not** maintained as frequently as the user input script so it is **not** a recommended tool.

- **If using the BEAPP user inputs script, see** Running BEAPP Using User Input Scripts.
- **If using the BEAPP GUI, see** Running BEAPP Through the GUI.

5. Once the above steps are complete, you're ready to run BEAPP.

- **If you're running BEAPP from the user inputs scripts, run the command:**

```
beapp_main('use_script')
```

- **If you're using the GUI, press the 'Run BEAPP' button on the main** Run Template Panel

6. Enjoy using BEAPP!!!

Important

Please read the following points before running BEAPP for the first time.

- It's **strongly** recommended that you run a few test files through the pipeline before you begin batch processing, especially the first few times you use BEAPP. This will ensure your user settings are correct and you have a chance to adjust without needing to rerun large numbers of files unnecessarily. If you're running .mat files, you can do this by only including a few files in beapp_file_info_table.mat. If you're running .mff files, you can do this in several ways, but the easiest is to only include a few files in your source directory at first.

- Users wishing to convert BEAPP outputs to EEGLAB format should use the beapp2eeglab and batch_beapp2eeglab functions found in beapp\reference_data\example_scripts.

Glossary for Terms Used in BEAPP

Recording Periods/ Epochs: Subdivisions within one source file used to separate experimental paradigms or delineate breaks. In NetStation, these are called epochs.

- Some sections of BEAPP code and documentation still use "epoch" to refer to this, but future versions will use recording period.

Segment: A discrete portion of EEG taken from a recording, to be used for analysis. In EEGLAB, these are called epochs.

Task: Experimental paradigm with its own set of stimuli, e.g., oddball task.

Events/ Event Tags: Markers for the presentation of a stimulus or a manual annotation to a file. Can be added to a file by a user during preprocessing or by EPrime, Presentation, etc.

Condition: Variations of an event or stimulus that can occur as part of a task (for example the 'repeating' vs. 'oddball' stimuli in an oddball task).

Cell Codes: Number codes associated with a given condition.

Cell Array: A MATLAB data type with indexed data containers called cells, where each cell can contain any type of data.

Baseline Data: Continuously collected data not tied to a particular stimulus or time point (in some cases, this may be "resting" data). Event tagging during baseline data will be loaded but ignored during Segmenting.

Event-tagged Data: Data time-locked to a specific stimulus or set of stimuli (marked by tags) that should be used for Segmenting. This can be used for event-related potential (ERP) paradigms, for example.

Conditioned Baseline: Baseline data recorded between a set of event tags. This can be a single recording (for example with start and end tags), or alternating or recurring sections of baseline data. For example, resting data in which the eyes are intermittently opened and closed, sleep stages, or data containing intermittent epileptiform activity would often be of this data type.

General User Inputs Overview

In both the user input script or GUI you can decide:

- Which modules to turn on/run your data through (Ex: Filter, Resample, Re-referencing, etc.)
- How to customize the settings within each selected module (Ex: If the Filter module is turned on, you can choose to apply a high-pass filter, low-pass filter, or a combination)

All files must go through the Format module first before using other modules in BEAPP, but most other modules in the pipeline are optional.

- If you turn a module on, you'll want to make sure the specifications for that module are set to your liking (e.g., if you choose to filter your data, you'll want to edit the user inputs that determine which frequencies to filter).
- If you turn a module off, you can leave the inputs for that module set to the default values without impacting your data. (e.g., if you turn the filtering module off, it doesn't matter how you set your high pass and low pass filter settings – no filtering will occur, regardless of these settings).

Modules in BEAPP are presented in the same order they run (e.g. Format happens before Filter and Filter can not occur before Format). A module may produce one of three types of possible output data:

- Continuous/not yet segmented (pre-processing)
- Segmented
- An output metric (e.g., ITPC)

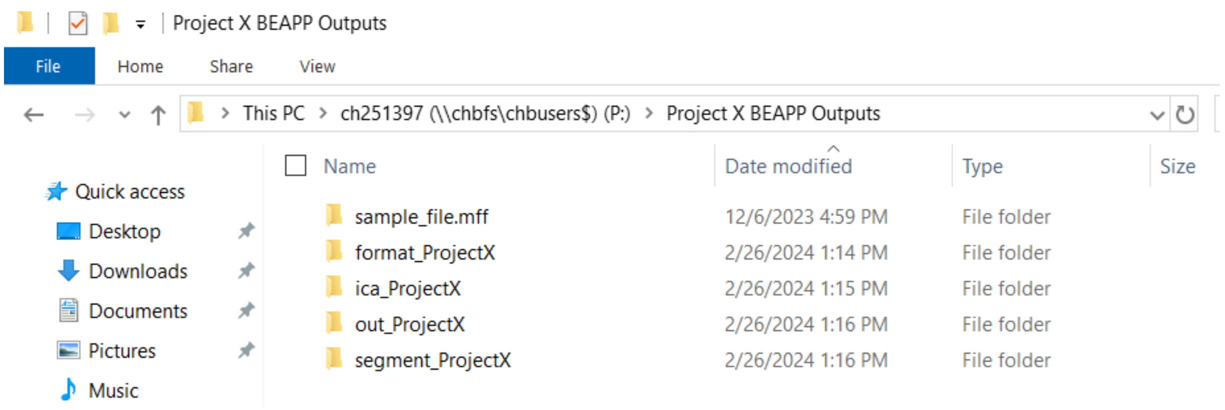
Currently, modules intended for continuous data cannot be applied to segmented data, and output metrics can only be calculated from segmented data. This is important to keep in mind such as if you're reading pre-segmented data into BEAPP as most preprocessing modules will not be an option (see BEAPP outline figure).

Current Run Tags

Back (Enter Run Tag) Back (GUI)

Modules output their processed data to subfolders where your source files are located. You can also add a tag to the subfolder names by specifying this tag in the Current Run Tag field. This is recommended as it will help you keep track of different runs and prevent you from overwriting data you wanted to save.

BEAPP is modular to allow for flexibility in reruns (see section on Rerunning BEAPP Modules for additional information). Users should avoid using the same run tag for runs with different module settings For example, let’s say you ran Filter, ICA, Re-reference, and Segment modules with the run tag ’'Example'. Let’s also say you chose HAPPE as your ICA method. After the run, you might have realized that you shouldn't be running Re-reference after HAPPE[1] and also wanted to change your Filter settings, so you do another run using the same run tag, 'Example', unselect Re-reference, and change your Filter settings. However, if you didn't delete the Re-reference subfolder with 'Example' appended to it, Segment will still use the Re-reference data and not the ICA data. All in all, it is important that you use different run tags to keep your BEAPP outputs organized.



Name	Date modified	Type	Size
sample_file.mff	12/6/2023 4:59 PM	File folder	
format_ProjectX	2/26/2024 1:14 PM	File folder	
ica_ProjectX	2/26/2024 1:15 PM	File folder	
out_ProjectX	2/26/2024 1:16 PM	File folder	
segment_ProjectX	2/26/2024 1:16 PM	File folder	

The figure above shows how run tags appear in BEAPP's output subfolders following a run. In this example, a user ran a sample data file through the Format, ICA, and Segment modules and set their Current Run Tag field to "ProjectX". As a result, each folder created during that run includes that tag in the folder name. The out folder contains general information about the entire run.

[1] You'll learn why this is the case later in the ICA and Re-referencing Module section of this guide

Running BEAPP Using the User Input Script

Back (Start-Up) Back (GUI)
This method is recommended for all users, especially those familiar with MATLAB.

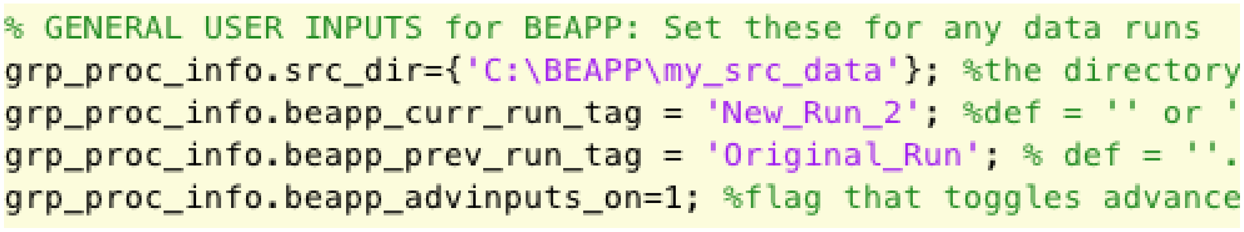
To set your user input script, you will complete the following steps:
1. Set the general user inputs (required for every data run)
2. Choose the modules to run.
3. Choose the details (parameters) for each of the modules you've turned on.
4. **Optional but recommended:** Save your user script with a descriptive name in the folder for your project and tell BEAPP to use this script – see Using Templates and Saved Tables.

Template user input script can be found in beapp/user_inputs/beapp_userinputs.m.

IMPORTANT: The live script beapp/beapp_new_main.mlx. is a hybrid of user input and GUI script. However, it is still under development. Please use the standard user script instead.

Step 1: General User Inputs (Required)

The general user inputs section is required for every data run. It looks like this:



```
% GENERAL USER INPUTS for BEAPP: Set these for any data runs
grp_proc_info.src_dir={'C:\BEAPP\my_src_data'}; %the directory
grp_proc_info.beapp_curr_run_tag = 'New_Run_2'; %def = '' or '
grp_proc_info.beapp_prev_run_tag = 'Original_Run'; % def = ''.
grp_proc_info.beapp_advinputs_on=1; %flag that toggles advance
```

In this section, you will need to specify the following information:

What folder contains the EEG files you plan to process?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.src_dir	The directory for your data files.	= {''}; (default) = {'C:\beapp_beta\my_src_data'};	Users can only input one directory at a time. If empty, BEAPP will throw an error.

How would you like to label ("tag") output directories for this data run?
Back (Gen User Inputs Overview)

- For additional information, see Current Run Tags.
- This is a label that is appended to output directories for this run. Using run tags is always recommended.

Variables/ Inputs	Description	Examples	Notes

grp_proc_info.beapp_curr_run_tag	The tag you would like to append to folder names for this run.	= ''; (default)
		= 'NONE';
		= '2_sec_segs';
		= 'eyes_open';
			If = ''; a timestamp will be added in place of a run tag. To mute this feature, use = 'NONE';
			By default, BEAPP will warn you if you use a previously used run tag.

If you are re-running data (e.g., you've already run data through BEAPP but you want to re-run your data with new filter settings), what was your prior data run tag?

- For additional information on re-running data, see Rerunning BEAPP Modules.

Variables/ Inputs	Description	Examples	Notes

grp_proc_info.beapp_prev_run_tag	Run tag that was appended to directories from a previous BEAPP run that you would now like to use as source data. See Rerun Example.	= ''; (default)
		= '1_sec_windows';
		= 'eyes_open_only';
			This input is only needed during reruns to pull data from directories in a previous BEAPP run. If setting grp_proc_info.beapp_prev_run_tag to a timestamp, be sure the spelling matches exactly.

Will you be setting any advanced inputs?

- For additional information on advanced inputs, see Advanced User Settings.

Variables/ Inputs	Description	Examples	Notes
grp_proc_info.advinputs_on	Toggles using the advanced user settings.	= 0; (off, default) = 1; (on)	Advanced inputs are set in beapp_advinputs.m.

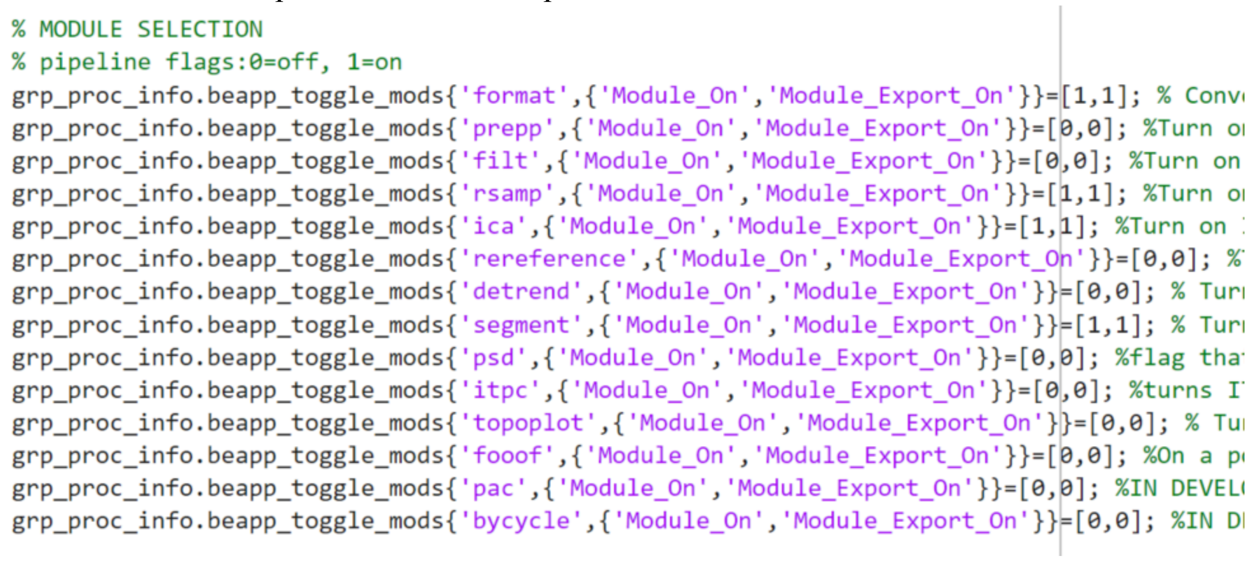
Step 2: Module Selection

Module selection is where users determine:

1. Which modules to run
    - This is determined by the first value in the vector after the = sign (i.e., = [1,1], the highlighted number).
        - 1 means "Run this module"

        ○ 0 means “Don't run this module”
2. Whether to save the output of each module.
    - This is determined by the second value in the vector after the = sign (i.e., = [1,1], the highlighted number).
        ○ 1 means “Save output from this module”
        ○ 0 means “Don't save output from this module”
    - You might choose not to save output from a particular module if that module is an intermediate step in your processing pipeline, the step can quickly be re-run in the future if needed, and/or you need to save space on your computer.

The module selection portion of the user inputs looks like this:



```
% MODULE SELECTION
% pipeline flags:0=off, 1=on
grp_proc_info.beapp_toggle_mods{'format',{'Module_On','Module_Export_On'}}=[1,1]; % Conv
grp_proc_info.beapp_toggle_mods{'prepp',{'Module_On','Module_Export_On'}}=[0,0]; %Turn o
grp_proc_info.beapp_toggle_mods{'filt',{'Module_On','Module_Export_On'}}=[0,0]; %Turn on
grp_proc_info.beapp_toggle_mods{'rsamp',{'Module_On','Module_Export_On'}}=[1,1]; %Turn o
grp_proc_info.beapp_toggle_mods{'ica',{'Module_On','Module_Export_On'}}=[1,1]; %Turn on
grp_proc_info.beapp_toggle_mods{'rereference',{'Module_On','Module_Export_On'}}=[0,0]; %
grp_proc_info.beapp_toggle_mods{'detrend',{'Module_On','Module_Export_On'}}=[0,0]; % Tur
grp_proc_info.beapp_toggle_mods{'segment',{'Module_On','Module_Export_On'}}=[1,1]; % Tur
grp_proc_info.beapp_toggle_mods{'psd',{'Module_On','Module_Export_On'}}=[0,0]; %flag tha
grp_proc_info.beapp_toggle_mods{'itpc',{'Module_On','Module_Export_On'}}=[0,0]; %turns I
grp_proc_info.beapp_toggle_mods{'topoplot',{'Module_On','Module_Export_On'}}=[0,0]; % Tu
grp_proc_info.beapp_toggle_mods{'fooof',{'Module_On','Module_Export_On'}}=[0,0]; %On a p
grp_proc_info.beapp_toggle_mods{'pac',{'Module_On','Module_Export_On'}}=[0,0]; %IN DEVEL
grp_proc_info.beapp_toggle_mods{'bycycle',{'Module_On','Module_Export_On'}}=[0,0]; %IN D
```

In the image above, the user has chosen to run the Format, Resampling, ICA, and Segmentation modules. They are saving outputs for Format, Resampling, ICA, and Segmentation.

For additional information on any of the modules, refer to the Module Parameters section below.

Step 3: Module Parameters

Back (Module Selection)

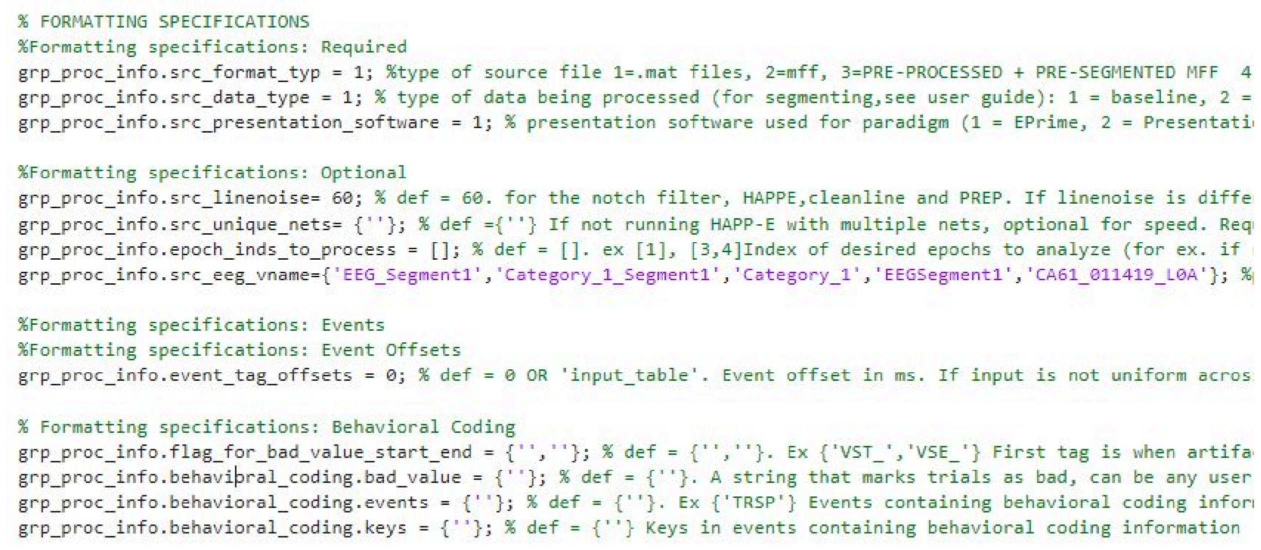
Once you've chosen which modules to run, look further down in the user inputs to edit the parameters for those modules you've turned on. You can ignore the parameters for those modules you've turned off.

FORMATTING SPECIFICATIONS

****Recommended****

The Formatting specifications section gives BEAPP the information it needs to convert your source EEG data into a format that BEAPP can process. Any new dataset to be run through BEAPP will therefore need to be run through the Formatting module.

The section specifying parameters for the Formatting module looks like this:



```
% FORMATTING SPECIFICATIONS
%Formatting specifications: Required
grp_proc_info.src_format_typ = 1; %type of source file 1=.mat files, 2=mff, 3=PRE-PROCESSED + PRE-SEGMENTED MFF  4
grp_proc_info.src_data_type = 1; % type of data being processed (for segmenting,see user guide): 1 = baseline, 2 =
grp_proc_info.src_presentation_software = 1; % presentation software used for paradigm (1 = EPrime, 2 = Presentati

%Formatting specifications: Optional
grp_proc_info.src_linenoise= 60; % def = 60. for the notch filter, HAPPE,cleanline and PREP. If linenoise is diffe
grp_proc_info.src_unique_nets= {''}; % def ={''} If not running HAPP-E with multiple nets, optional for speed. Req
grp_proc_info.epoch_inds_to_process = []; % def = []. ex [1], [3,4]Index of desired epochs to analyze (for ex. if
grp_proc_info.src_eeg_vname={'EEG_Segment1','Category_1_Segment1','Category_1','EEGSegment1','CA61_011419_L0A'}; %

%Formatting specifications: Events
%Formatting specifications: Event Offsets
grp_proc_info.event_tag_offsets = 0; % def = 0 OR 'input_table'. Event offset in ms. If input is not uniform acros

% Formatting specifications: Behavioral Coding
grp_proc_info.flag_for_bad_value_start_end = {'',''}; % def = {'',''}. Ex {'VST_','VSE_'} First tag is when artifa
grp_proc_info.behavioral_coding.bad_value = {''}; % def = {''}. A string that marks trials as bad, can be any user
grp_proc_info.behavioral_coding.events = {''}; % def = {''}. Ex {'TRSP'} Events containing behavioral coding infor
grp_proc_info.behavioral_coding.keys = {''}; % def = {''} Keys in events containing behavioral coding information
```

In this section, you will need to specify the following information:

What is the format of your source files?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.src_format_typ	The format of your raw data. For most users, this will be one of the following: • .mat exports • .mff files with continuous data • .mff files with segmented data	= 1; (default, .mat) = 2; (.mff) = 3; (pre-processed + pre-segmented .mff) = 4; (.set) =5; (pre-processed + pre-segmented .set)	.mat exports require an accompanying table (see Running BEAPP with Different Source File Formats).

What kind of data will you be processing? (e.g., continuous baseline data, or event-related data)			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.src_data_typ	Type of data being processed. For most users, this will be one of the following: 1 = Pure baseline files (pre-extracted) 2 = Event related (or baseline with time-locked	= 1; (default) = 2;	For details and examples, see Running BEAPP with Differently Structured Data.

	information + regular event tags) 3 = Baseline with blocked conditions (e.g., eyes open, eyes closed blocks)		
grp_proc_info.src_presentation_software	Presentation software used for paradigms. 1 = EPrime 2 = Presentation	= 1; (default) = 2;	

What is the frequency of line noise where your data were collected?

Back (Generate File Info Table)

- *You only need to specify this if you will be running a module that removes line noise (e.g., PREP, Notch filter, or HAPPE).* Typically, this is 60 Hz in North and South America, and parts of Asia; it is typically 50 Hz in Europe and other parts of the world. The following website contains information regarding line noise in other countries: https://en.wikipedia.org/wiki/Mains_electricity_by_country.
- If you are running a dataset in which line noise frequency varies by file (e.g., a combined dataset with EEG obtained in the USA and the UK), you will need to specify line noise for each file in the beapp_file_info_table, as described in Generating a BEAPP File Info Table**.**

Variables/ Inputs	Description	Examples	Notes
grp_proc_info.src_linenoise	Frequency of line (electrical) noise in your data set, in Hz.	= 60; (U.S. data, default) = 50; (U.K. data)	This is used for notch filtering, HAPPE, and PREP.

What electrode layouts (or nets) were used to collect your data?

Back (HAPPE Specifications)

- *You are only required to specify this if you will be running the ICA module or re-referencing to specific channels.* However, specifying this variable will also increase the speed of runs (especially reruns). For additional information on managing electrode nets and acquisition layouts in BEAPP, see the section on BEAPP Net Library.

Variables/ Inputs	Description	Examples	Notes
grp_proc_info.src_unique_nets	List of unique nets present in dataset, spelled EXACTLY as in net library.	= {''}; (default) = {'HydroCel GSN 128 1.0', 'Biosemi 32', 'Biosemi 128'};	Optional, unless running HAPPE/ICA/ adding a new net type to the library. During reruns, recommended for speed.

Which data recording periods (epochs) would you like to process?

- *You only need to specify this if your data includes multiple recording periods, and you only want to run a subset of those recording periods.* For example, let's say your continuous EEG included a few minutes of “resting” data, a few minutes of an auditory task, and then a few minutes of a visual task, with each of these tasks in a separate recording period. If you only wanted to run the auditory task, you'd specify that you only wish to run the second epoch, i.e. [2].

Variables/ Inputs	**Description**	**Examples**	**Notes**

grp_proc_info.epoch_inds_to_process	Index of epoch to be processed across files (all indexes in all files).	= []; (default)
		= [1];
		= [3,4];
			If using default, all epochs in all files will be processed.

Which variable(s) contain(s) the EEG data itself?

- *You only need to specify this if your data is in .mat format.* This is the variable with the (unsegmented) EEG data, in matrix format, as described in the start-up guide.

Variables/ Inputs	**Description**	**Examples**	**Notes**

grp_proc_info.src_eeg_vname	Cell array of possible names of the variable with the EEG data.	= {'Category_1_Segment1','Category_1','EEG_Segment1','EEGSegment1'}; (default)
		= {'EEG_Segment1'};
		= {'EEG_Dat','Cat1_Seg'};
			Only required for .mat files.

What is the offset of events tagged in your dataset?

- *You only need to specify this if you will be segmenting events; you do not need this for continuous (non-event-related) data.* If you are running a dataset in which offset varies by file, you will need to specify offset for each file in beapp_file_info_table, as described in Generating a BEAPP File Info Table. If the tagging of your events accounts for any offset, you can set this value to zero.
    - Example: Let's say your EEG has an event tag every time the computer thinks it presents a picture of a face. However, you (as a careful researcher!) have learned that due to a variety of delays in the system wiring, the face in fact appears on the monitor that your experiment's participant can see 18 milliseconds later. You would therefore set your event tag offset to 18.

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.event_tag_offsets	Event tag offsets in source data, in ms.	= 0; (no offset in all files, default)	For information on the format for the offset

	• If offsets are the same across all files, set to one number • If offset differs across files in the dataset, set to 'input_table'	= 2; (offset of 2 ms in all files) = 'input_table'; (differing offsets, specified in file info table)	table, see Running Event Tagged Data.

How should BEAPP recognize time windows or events to be excluded (e.g., events marked in the file as unusable)?

- This is called "behavioral coding" because in many cases segments of EEG are marked for exclusion based on a behavior that an observer notices during the EEG acquisition (e.g., inattention to a stimulus, blinking, etc). *You only need to specify this if your file contains some information about which events or epochs should be excluded. If you do specify this, trials marked "bad" will be excluded.*
- This functionality works only for event-related and conditioned-baseline files. Currently not available for baseline files.
- Read here for more examples.

Variables/ Inputs	Description	Examples	Notes
grp_proc_info.flag_for_bad_value_start_end	Cell array of artifact start tag and artifact end tag.	={'',''}; (default) = {'art+','art-'};	Optional. If filled in, trials that elapse between the artifact start and end tag will be rejected.

grp_proc_info.behavioral_coding.bad_value	A user-defined string that marks behavioral coding as bad.	= {''}; (default)
		= {'badt'};
		= {'1'};
			Must be filled in for grp_proc_info.flag_for_bad_value_start_end to be functional.

grp_proc_info.behavioral_coding.events	Cell array of events that include information about behavioral coding.	= {''}; (default)
		= {'TRSP'};
		= {'TRSP','TRSP2'};
			Optional.

grp_proc_info.behavioral_coding.keys	Cell array of keys with behavioral coding in events.	= {''}; (default)
		= {'badt'};
		= {'badtf','badth'};
			Optional. If filled in, trials with bad values will be rejected.

Format Module Output Variables

Variables/ Outputs	Description	Units	Notes

eeg	Amplitude of continuous EEG data in a 2D matrix (electrode channels x time samples).	Microvolts (µV)	
file_proc_info	Processing information for each data file.	N/A	

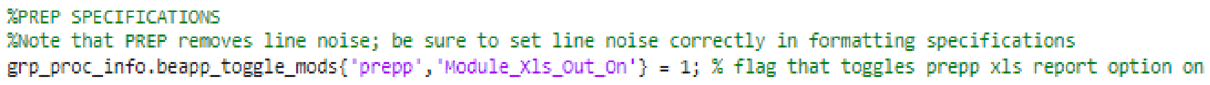
PREP SPECIFICATIONS

****Not Recommended****

If the user has turned on the PREP module, the section on PREP specifications gives BEAPP the information it needs to run the PREP pipeline (Bigdely-Shamlo et al., 2015). PREP is a very early stage EEG preprocessing pipeline that offers the following:

1. Removal of line noise
2. Detection and interpolation of bad channels
3. Robust average referencing

The PREP pipeline is standalone software that has been integrated into BEAPP. The majority of PREP settings are determined by its intrinsic defaults, although the user does have the option to change the line noise frequency as described in the Format module above. Otherwise, the only determination the user needs to make is whether they would like BEAPP to create an Excel output of PREP's findings. Therefore, the section specifying parameters for the PREP module looks like this:



```
%PREP SPECIFICATIONS
%Note that PREP removes line noise; be sure to set line noise correctly in formatting specifications
grp_proc_info.beapp_toggle_mods{'prepp','Module_Xls_Out_On'} = 1; % flag that toggles prepp xls report option on
```

In this section, the user will need to specify only the following information:

Do you want BEAPP to save out a report table with PREP summary information about the files used?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_toggle_mods{'prepp', 'Module_Xls_Out_On'}	Toggles saving a report table with PREP summary information.	= 1; (on → save, default) = 0; (off → don't save)	

PREP Module Output Variables

Variables/ Outputs	**Description**	**Units**	**Notes**

eeg	Amplitude of continuous EEG data in a 2D matrix (electrode channels x time samples).	Microvolts (µV)	
EEG_structs	A MATLAB structure that contains all the information about the current data in EEGLAB format.	N/A	Read more here.
params	A MATLAB structure that contains information about channels referenced, re-referenced, and evaluated, as well as line frequencies.	N/A	
file_proc_info	Processing information for each data file.	N/A	

FILTER SPECIFICATIONS

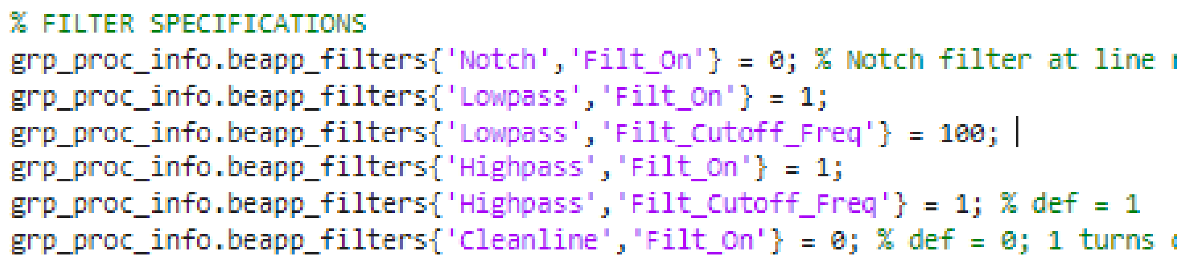
****Recommended****

IMPORTANT: If running ICA with HAPPE, do not turn on cleanline or notch filter.

If the user has turned on the Filtering module, the section on filter specifications determines the filter settings to be run on the dataset. BEAPP offers a Low pass and High pass filter, and offers a Notch or Cleanline filter for line noise removal:

1. Low pass filtering (essentially allowing only oscillations slower than a given frequency to be included)
2. High pass filtering (essentially allowing only oscillations faster than a given frequency to be included)
3. Notch filtering (for line noise removal)
4. Cleanline (for line noise removal)

The section specifying parameters for the Filter module looks like this:



```
% FILTER SPECIFICATIONS
grp_proc_info.beapp_filters{'Notch','Filt_On'} = 0; % Notch filter at line
grp_proc_info.beapp_filters{'Lowpass','Filt_On'} = 1;
grp_proc_info.beapp_filters{'Lowpass','Filt_Cutoff_Freq'} = 100;
grp_proc_info.beapp_filters{'Highpass','Filt_On'} = 1;
grp_proc_info.beapp_filters{'Highpass','Filt_Cutoff_Freq'} = 1; % def = 1
grp_proc_info.beapp_filters{'Cleanline','Filt_On'} = 0; % def = 0; 1 turns
```

In this section, for each filter type the user will need to specify the following information:

Should this filter type be turned on? If so, what is the filter's cutoff frequency?

- 1 means “Turn on this filter type”
- 0 means “Turn off this filter type”
- Cutoff frequency is only specified for low pass and high pass filters. Notch filtering or Cleanline, if turned on, automatically occurs at the line noise frequency specified in the Format module grp_proc_info.src_linenoise.

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_filters{'Notch','Filt_On'}	Toggles notch filter on and off.	= 1; (on, default) = 0; (off)	When on, BEAPP applies a notch filter at the line noise frequency (grp_proc_info.src_linenoise).
grp_proc_info.beapp_filters{'Lowpass','Filt_On'}	Toggles low-pass filter on and off .	= 1; (on, default) = 0; (off)	
grp_proc_info.beapp_filters {'Lowpass','Filt_Cutoff_Freq'}	Cutoff frequency for low-pass filter.	= 100; (default)	
grp_proc_info.beapp_filters{'Highpass','Filt_On'}	Toggles high-pass filter on and off.	= 1; (on, default) = 0; (off)	
grp_proc_info.beapp_filters {'Highpass','Filt_Cutoff_Freq'}	Cutoff frequency for high-pass filter.	= 1; (default)	

grp_proc_info.beapp_filters{'Cleanline','Filt_On'}	Toggles Cleanline on and off.	= 0; (off, default) = 1; (on)	When on, BEAPP applies Cleanline at line noise frequency.

Keep in mind that these settings only matter if the Filtering module is turned on. If the Filtering module is turned off in the "Module Selection" section, no filtering will occur (even if individual filters are set to "1").

Many users will likely choose a low pass filter at the Nyquist frequency (i.e., just under half of the sampling rate). If downsampling, many users will likely choose a low pass filter at the Nyquist frequency of the target sampling rate. For example, if the sampling rate is 250 Hz (or if the sampling rate is 500 Hz but the user will resample to 250 Hz), a user might low pass filter at 100 Hz.

Filter Module Output Variables

Variables/ Outputs	**Description**	**Units**	**Notes**
eeg	Amplitude of filtered, continuous EEG data in a 2D matrix (electrode channels x time samples).	Microvolts (µV)	
file_proc_info	Processing information for each data file.	N/A	

RESAMPLING SPECIFICATIONS

****Recommended****

If the user has turned on the Resampling module, the section on Resampling specifications determines how data will be resampled.

The section specifying parameters for the Resampling module looks like this:

```
%RESAMPLING SPECIFICATIONS
grp_proc_info.beapp_rsamp_srate = 250; %target sampling rate for resampling, if desired
```

In this section, the user will need to specify only the following information:

What is your desired sampling rate?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_rsamp_srate	Desired sampling rate (Hz) after resampling.	= 250; (default)	

Many users will likely choose to resample if they have EEGs in a dataset that are collected at a variety of sampling rates. In many such cases, users will downsample all data to the lowest common sampling rate. For example, if a user has EEG data sampled at 1024 Hz, 1000 Hz, 500 Hz, and 250 Hz, they might choose to downsample all data to 250 Hz.

A user might also downsample to improve the quality of ICA decomposition. Some ICA paradigms work best at lower sampling rates (e.g., 250 Hz).

Resampling Module Output Variables

Variables/ Outputs	**Description**	**Units**	**Notes**
eeg	Amplitude of resampled, continuous EEG data in a 2D matrix (electrode channels x time samples).	Microvolts (μV)	
file_proc_info	Processing information for each data file.	N/A	

ICA/HAPPE SPECIFICATIONS

****Recommended****

IMPORTANT: Before running ICA/HAPPE for the first time, users need to follow the instructions provided in ICA Module/HAPPE Preparations.

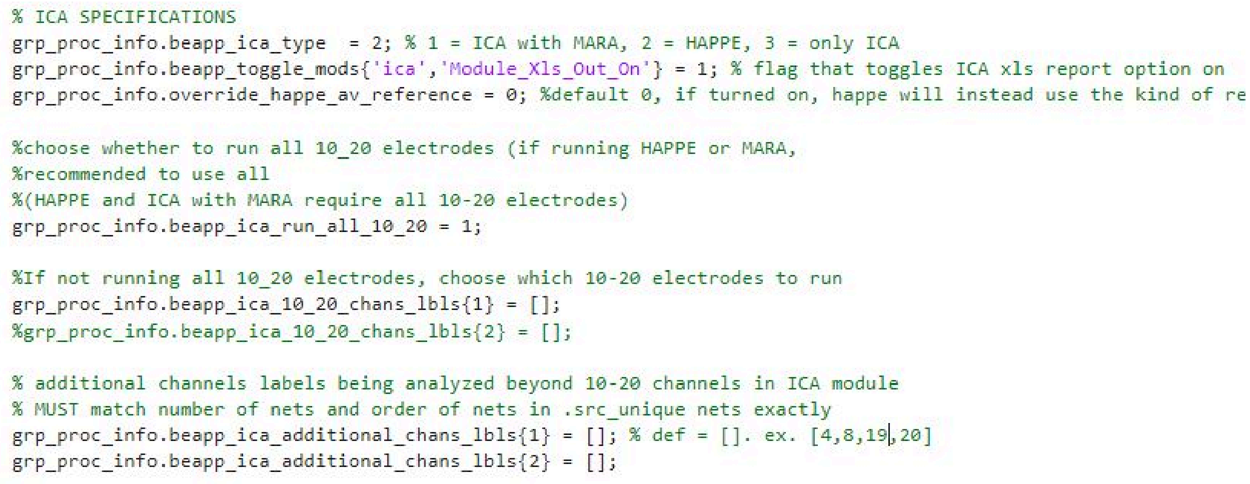
In this module, users have the option to run ICA, ICA+MARA, or HAPPE, a specific EEG preprocessing pipeline targeted towards artifact removal for infant EEG data.

HAPPE integrated into BEAPP includes the following steps:

1. 1-250 Hz band-pass filter
2. Select desired channels of interest (e.g. 10-20 channels)
3. CleanLine to remove line noise
4. Bad Channel Rejection
5. Wavelet-Thresholding ICA (W-ICA) cleaning
6. ICA with MARA
7. Interpolate bad channels
8. Average re-reference

An in-depth explanation for each of the processing steps used in HAPPE can be found in the link in the paragraph above.

The section specifying parameters for the ICA module looks like this:



```
% ICA SPECIFICATIONS
grp_proc_info.beapp_ica_type  = 2; % 1 = ICA with MARA, 2 = HAPPE, 3 = only ICA
grp_proc_info.beapp_toggle_mods{'ica','Module_Xls_Out_On'} = 1; % flag that toggles ICA xls report option on
grp_proc_info.override_happe_av_reference = 0; %default 0, if turned on, happe will instead use the kind of re

%choose whether to run all 10_20 electrodes (if running HAPPE or MARA,
%recommended to use all
%(HAPPE and ICA with MARA require all 10-20 electrodes)
grp_proc_info.beapp_ica_run_all_10_20 = 1;

%If not running all 10_20 electrodes, choose which 10-20 electrodes to run
grp_proc_info.beapp_ica_10_20_chans_lbls{1} = [];
%grp_proc_info.beapp_ica_10_20_chans_lbls{2} = [];

% additional channels labels being analyzed beyond 10-20 channels in ICA module
% MUST match number of nets and order of nets in .src_unique nets exactly
grp_proc_info.beapp_ica_additional_chans_lbls{1} = []; % def = []. ex. [4,8,19,20]
grp_proc_info.beapp_ica_additional_chans_lbls{2} = [];
```

In this section, the user will need to specify the following information:

Which ICA process would you like to run?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_ica_type	Desired type of ICA process.	= 1; (ICA with MARA, default) = 2; (HAPPE) = 3; (only ICA)	If you choose to run ICA or ICA+ MARA, the other steps of HAPPE described above (i.e., line noise removal, crude bad channel detection, wavelet cleaning, interpolation of bad channels, average reference) are not included.

Do you want BEAPP to save out a report table with summary information about the files used?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_toggle_mods{'ica','Module_Xls_Out_On'}	Toggles ICA summary output table on and off.	= 1; (on, default) = 0; (off)	When on, BEAPP saves a report table in the "out" folder with summary ICA metrics.

Do you want to override the default average re-referencing in HAPPE and replace it with specific electrode re-referencing?			
Variables/ Inputs	**Description**	**Examples**	**Notes**

grp_proc_info.override_happe_av_reference	Toggles HAPPE average re-referencing on and off	= 0; (off, default) = 1; (on)	When on, BEAPP changes the re-reference type conducted in HAPPE to the type set through grp_proc_info.reref_typ. Currently, only specific electrodes re-referencing is supported through this method. Remember to also specify desired re-reference channel(s) in grp_proc_info.beapp_reref_chan_inds

If you don't want to run all 10-20 electrodes, which 10-20 electrodes would you like to run?			
Variables/ Inputs	**Description**	**Examples**	**Notes**

grp_proc_info.beapp_ica_10_20_chans_lbls{1}
 	
grp_proc_info.beapp_ica_10_20_chans_lbls{2}
	Labels associated with specific 10-20 channels in ICA. Each row applies to the corresponding net from grp_proc_info.src_unique_nets	= []; (run all 10-20 electrodes, default)	

In addition to the 10-20 channels, would you like ICA to include any other channels in its analyses?

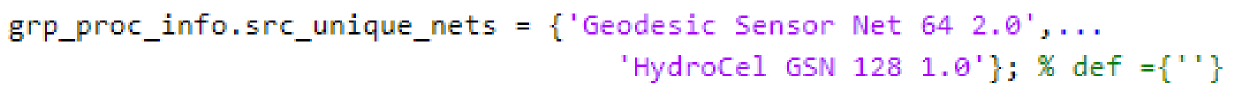
- The ICA module will automatically use any electrodes listed as equivalent to 10-20 electrodes for a given net in the net library (as MARA uses 10-20 locations for component evaluation). Users have the option to add additional electrodes of interest for each net using grp_proc_info.happe_additional_chans_lbls.
- Channels should be listed using the index (i.e. the rows in eeg corresponding to the desired channel). Each net's cell number in grp_proc_info.ica_additional_chans_lbls (e.g. grp_proc_info.ica_additional_chans_lbls{1}) should correspond to the net's position in grp_proc_info.src_unique_nets. The number of channels listed for each net can be different. For now this is applied to ICA without MARA as well.

Variables/ Inputs	Description	Examples	Notes

grp_proc_info.beapp_ica_additional_chans_lbls{1}
 	
grp_proc_info.beapp_ica_additional_chans_lbls{2}
etc.
	Labels associated with additional (non-10-20 configuration) channels in ICA. Each row applies to the corresponding net from grp_proc_info.src_unique_nets.	= []; (no additional channels, default)
		= [119]; (include channel 119)
		= [36,42,8]; (include channels 36, 42, 8)	

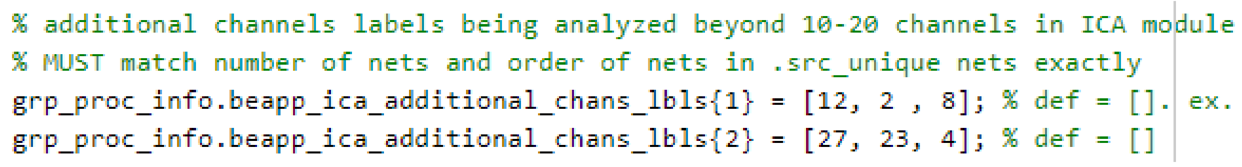
An example of using the parameter above (grp_proc_info.additional_chans_lbls{}):

- Let's say a user is running ICA for 2 nets, using different electrodes for each net. In Formatting specifications, they specified nets as follows:



```
grp_proc_info.src_unique_nets = {'Geodesic Sensor Net 64 2.0',...
                                  'HydroCel GSN 128 1.0'}; % def ={''}
```

- If the user wishes to include additional channels (not just the 10-20 channels) in the ICA decomposition, they could therefore specify additional channels as follows:



```
% additional channels labels being analyzed beyond 10-20 channels in ICA module
% MUST match number of nets and order of nets in .src_unique nets exactly
grp_proc_info.beapp_ica_additional_chans_lbls{1} = [12, 2 , 8]; % def = []. ex.
grp_proc_info.beapp_ica_additional_chans_lbls{2} = [27, 23, 4]; % def = []
```

- Note that the first set specifies channels to be included for EEG data from a Geodesic Sensor Net 64 2.0 net, and the second set specifies channels to be included for EEG data from a HydroCel GSN 128 1.0 net, i.e. channels 12, 2, and 8 will also be included for ICA in Geodesic Sensor Net 64 2.0 nets and channels 27, 23, and 4 will be included for HydroCel GSN 128 1.0 nets. The order of specification is important here.

Notes regarding channel inclusion:

- ICA is the only module in BEAPP that automatically restructures data to include only the user selected channels (i.e. 10-20 channels along with any additional channels that the user explicitly instructs BEAPP to include).
- All other modules in BEAPP will include all input channels (i.e., all 64 channels in a 64-channel net, and all 128 channels in a 128-channel net).
- Choosing how many channels to run in the ICA module is somewhat subjective, and users can run the module with visualizations turned on to check how well the ICA decomposition is working when there are different numbers of channels as input. However, generally, the number of channels that can be added will be limited by the input file's length (time). It is recommended that the number of samples in your EEG recording be equal to 20 * (the number of channels)[2].

    - Example: an EEG acquired with a 128-channel net and sampling rate of 500 Hz (500 samples/ second) would need at least 327,680 samples ($128^2 * 20$ samples), that is, 655.36 seconds of recording (327,680 samples at 500 Hz) reliably decompose all channels with ICA (see HAPPE for further discussion).
    - i.e. $Channels = \sqrt{\frac{SamplingRate \times Seconds}{20}}$
- Outside of ICA/HAPPE, the only exception to the rule that BEAPP will include all channels is if the user specifies (in advanced user inputs) that they would like bad channels removed. In this case, any channels that PREP (or another module) identifies as bad will be replaced with NaN. Otherwise, BEAPP defaults to interpolating channels that PREP (or another module) identifies as bad, but then keeps these interpolated channels for downstream analysis.

ICA/HAPPE Module Output Variables

Variables/ Outputs	**Description**	**Units**	**Notes**
ICA Report Table Back (Segment)	A .csv report of descriptive statistics and data metrics for each EEG file in the batch that evaluates pipeline performance and data quality, stored in the "out_[run_tag]" folder.	N/A	Find more information on the data metrics here.
eeg	Amplitude of artifact detected/corrected continuous EEG data in a 2D matrix (electrode channels x time samples).	Microvolts (µV)	
file_proc_info	Processing information for each data file.	N/A	

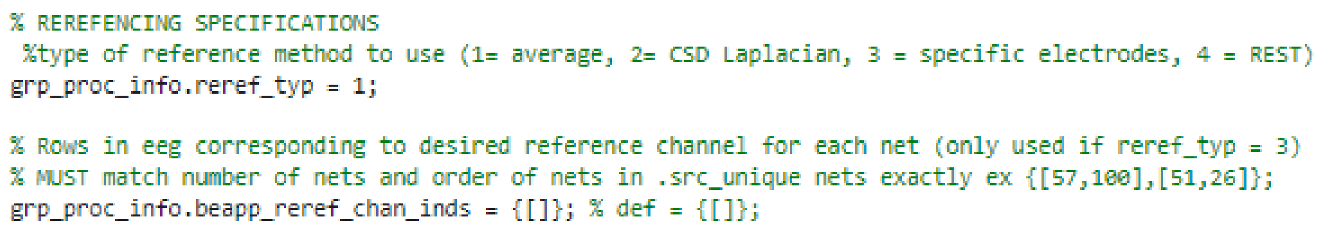
RE-REFERENCING SPECIFICATIONS

****Recommended****

IMPORTANT: If ICA was run with HAPPE, do not run re-referencing.

If the user has turned on the re-referencing module, the section on re-referencing specifications determines the type of re-referencing that will occur.

The section specifying parameters for the Re-Referencing module looks like this:



```
% REREFENCING SPECIFICATIONS
 %type of reference method to use (1= average, 2= CSD Laplacian, 3 = specific electrodes, 4 = REST)
grp_proc_info.reref_typ = 1;

% Rows in eeg corresponding to desired reference channel for each net (only used if reref_typ = 3)
% MUST match number of nets and order of nets in .src_unique nets exactly ex {[57,100],[51,26]};
grp_proc_info.beapp_reref_chan_inds = {[]}; % def = {[]};
```

In this section, the user will need to specify only the following information:

What type of re-referencing would you like applied to your data?			
Variables/ Inputs	**Description**	**Examples**	**Notes**

grp_proc_info.reref_typ	Type of re-referencing method applied:
	1 = Average
	2 = CSD Laplacian
	3 = Specific electrode(s)
	4 = REST
		= 1; (default)
		= 2;
		= 3;
		= 4;
			CSD uses the CSD Toolbox.

If re-referencing to a specific electrode or electrodes, which electrode(s) would you like to use in each net? ● The order of the electrode lists must line up exactly with the order of the nets they refer to in grp_proc_info.src_unique_nets, specified in the Formatting module (see example below) ● Channels used for reref_chan_inds must be one of the channels included for the ICA module.			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_reref_chan_inds	Rows in EEG corresponding to the desired reference channel for each net (only used if reref_typ = 3).	= {[]}; (default) = {[57,100],[51,26]};	Number of []s must match the number of nets and order of nets in .src_unique_nets exactly.

Re-referencing Module Output Variables

Variables/ Outputs	**Description**	**Units**	**Notes**
eeg	Amplitude of re-referenced, continuous EEG data	Microvolts (µV)	

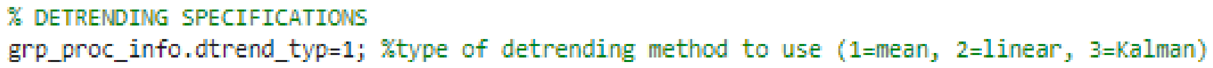
	in a 2D matrix (electrode channels x time samples).		
file_proc_info	Processing information for each data file.	N/A	

DETRENDING SPECIFICATIONS

****Optional****

If the user has turned on the Detrending module, the section on Detrending specifications determines the type of detrending that will occur.

The section specifying parameters for the Detrending module looks like this:



```
% DETRENDING SPECIFICATIONS
grp_proc_info.dtrend_typ=1; %type of detrending method to use (1=mean, 2=linear, 3=Kalman)
```

In this section, the user will need to specify only the following information:

What type of detrending would you like applied to your data?			
Variables/ Inputs	**Description**	**Examples**	**Notes**

grp_proc_info.dtrend_typ	Detrending method applied in module:
	1 = Mean
	2 = Linear
	3 = Kalman
		= 1; (default)
		= 2;
		= 3;	

Note on Kalman filtering (Back Advanced Inputs Kalman Filter)

- Please use Kalman filtering with caution as it is generally not recommended. It is used only in specific circumstances (e.g., to attenuate the impact of TMS artifact or certain epileptiform activity). To determine the b-value for the Kalman filter, please consider the following implications of a low b-value:
    - Lower b-values will improve fitting, and therefore decrease the impact of high-amplitude artifact.
    - For users excluding segments using amplitude cutoffs, lower b-values may increase the number of usable segments.
    - Users should be cautious about the potential impact of low b-values on the power spectrum. That is, low b-values may excessively attenuate low (e.g., delta, theta) and even mid (e.g., alpha, beta) frequency oscillations. The extent of this effect may also depend on the sampling rate. Therefore, if users plan to use a Kalman filter and then run the PSD (power spectral density) module, we encourage them to plot power spectra using several different b-values, to determine potential impact of the chosen value on the power spectrum.

Detrending Module Output Variables

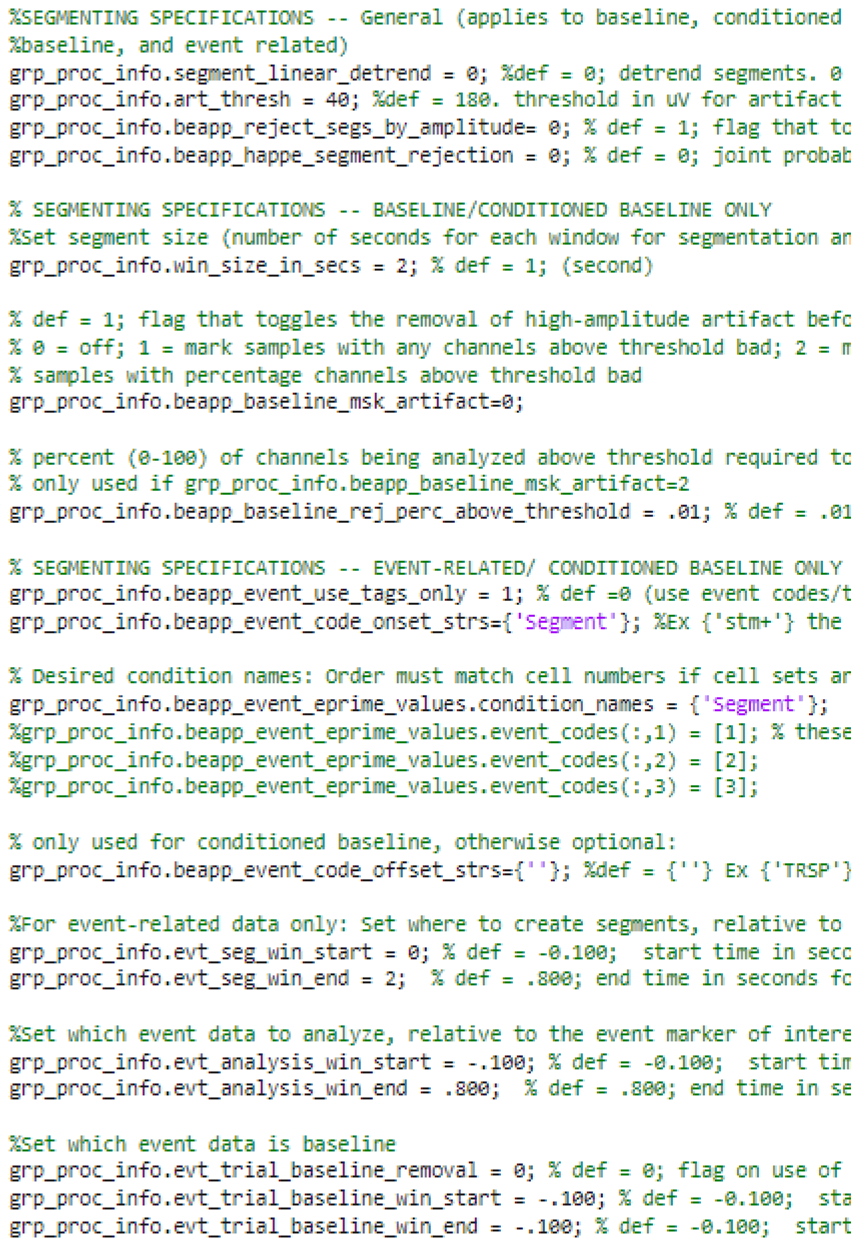
Variables/ Outputs	**Description**	**Units**	**Notes**
eeg	Amplitude of detrended, continuous EEG data in a 2D matrix (electrode channels x time samples).	Microvolts (µV)	
file_proc_info	Processing information for each data file.	N/A	

SEGMENTING SPECIFICATIONS

****Recommended****

If the user has turned on the Segmentation module, the section on Segmentation specifications determines the type of segmentation that will occur.

The section specifying parameters for the Segmenting module looks like this:



```
%SEGMENTING SPECIFICATIONS -- General (applies to baseline, conditioned
%baseline, and event related)
grp_proc_info.segment_linear_detrend = 0; %def = 0; detrend segments. 0
grp_proc_info.art_thresh = 40; %def = 180. threshold in uV for artifact
grp_proc_info.beapp_reject_segs_by_amplitude= 0; % def = 1; flag that tc
grp_proc_info.beapp_happe_segment_rejection = 0; % def = 0; joint probat

% SEGMENTING SPECIFICATIONS -- BASELINE/CONDITIONED BASELINE ONLY
%Set segment size (number of seconds for each window for segmentation ar
grp_proc_info.win_size_in_secs = 2; % def = 1; (second)

% def = 1; flag that toggles the removal of high-amplitude artifact befc
% 0 = off; 1 = mark samples with any channels above threshold bad; 2 = m
% samples with percentage channels above threshold bad
grp_proc_info.beapp_baseline_msk_artifact=0;

% percent (0-100) of channels being analyzed above threshold required tc
% only used if grp_proc_info.beapp_baseline_msk_artifact=2
grp_proc_info.beapp_baseline_rej_perc_above_threshold = .01; % def = .01

% SEGMENTING SPECIFICATIONS -- EVENT-RELATED/ CONDITIONED BASELINE ONLY
grp_proc_info.beapp_event_use_tags_only = 1; % def =0 (use event codes/t
grp_proc_info.beapp_event_code_onset_strs={'Segment'}; %Ex {'stm+'} the

% Desired condition names: Order must match cell numbers if cell sets ar
grp_proc_info.beapp_event_eprime_values.condition_names = {'Segment'};
%grp_proc_info.beapp_event_eprime_values.event_codes(:,1) = [1]; % these
%grp_proc_info.beapp_event_eprime_values.event_codes(:,2) = [2];
%grp_proc_info.beapp_event_eprime_values.event_codes(:,3) = [3];

% only used for conditioned baseline, otherwise optional:
grp_proc_info.beapp_event_code_offset_strs={''}; %def = {''} Ex {'TRSP'}

%For event-related data only: Set where to create segments, relative to
grp_proc_info.evt_seg_win_start = 0; % def = -0.100;  start time in secc
grp_proc_info.evt_seg_win_end = 2;  % def = .800; end time in seconds fc

%Set which event data to analyze, relative to the event marker of intere
grp_proc_info.evt_analysis_win_start = -.100; % def = -0.100;  start tim
grp_proc_info.evt_analysis_win_end = .800;  % def = .800; end time in se

%Set which event data is baseline
grp_proc_info.evt_trial_baseline_removal = 0; % def = 0; flag on use of
grp_proc_info.evt_trial_baseline_win_start = -.100; % def = -0.100;  sta
grp_proc_info.evt_trial_baseline_win_end = -.100; % def = -0.100;  start
```

In this section, the user will need to specify the following information:

Would you like to apply detrending to your segments?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.segment_linear_detrend	Toggles detrending each channel in each segment after Segmentation.	= 0; (off, default) = 1; (on, linear detrend)	This variable is different from the Detrending Module in that it will detrend

		= 2; (on, mean detrend)	each segment individually rather than detrend the whole signal un-segmented.

What is the amplitude threshold for artifact removal?

- The default amplitude threshold for artifact removal is 100µV. However, users should note that certain types of data (e.g., infant data, or EEG recorded after craniotomy) may require higher amplitude thresholds (e.g., 150). Other types of data (e.g., data that has been run through HAPPE, or data that has undergone a Laplacian transform) may require changes to amplitude thresholds. For HAPPE, an amplitude threshold of 40 or 50 is typically recommended.

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.art_thresh	Threshold (in µV) for BEAPP artifact removal software.	= 150; (default) = 100;	Will need to be scaled appropriately if using CSDLP (e.g. 3000) or ICA with HAPPE (e.g 40 or 50). Only relevant if the user toggles on removing segments based on amplitude or HAPPE segment rejection.

Should segments with high amplitude artifacts be removed?

- If this parameter is toggled on, BEAPP will remove any segments in which amplitude exceeds the specified threshold (grp_proc_info.art_thresh) in any channel.

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_reject_segs_by_amplitude	Toggles removing segments with high amplitude artifacts.	= 1; (on, default) = 0; (off)	

Do you want to use the HAPPE segment rejection code?

- This rejects segments based on amplitude (same as grp_proc_info.beapp_reject_segs_by_amplitude) and the joint probability of samples (EEGLAB jointprob function).
- Some users might choose not to use this code if they wish to keep all segments, or reject segments using another set of parameters (e.g., amplitude-based rejection criteria from the Segmenting module only).

Variables/ Inputs	Description	Examples	Notes
grp_proc_info.beapp_happe_segment_rejection	Toggles HAPPE method for segment rejection on and off.	= 0; (off, default) = 1; (on)	

If processing baseline or conditioned baseline files, what is the desired segment size?
Back (PSD) Back (GUI Event Seg Gen) Back (GUI Base Seg Gen) Back (GUI PSD)

- This only needs to be specified for **continuous (non-event-related data) baseline/conditioned-baseline data**. For event-related data, segment size will be determined by segment start and end times as they relate to the event marker of interest (grp_proc_info.evt_seg_win_start & grp_proc_info.evt_seg_win_end).
- This is often set to either 1 or 2 seconds.
  - For infant data, which is often highly contaminated by artifact, 1-second segments may be more feasible.
  - If the user will be running the PSD module with a multitaper, the minimum meaningful number of tapers (3) requires segment length of at least 2 seconds in order to get a frequency resolution of 1 Hz. For most users looking at low (delta, theta) or mid (alpha, beta) frequencies, segments of at least 2 seconds will therefore be necessary for a multitaper. However, for users looking at higher (gamma) frequencies, where low frequency resolution is less of a concern, a 1-second window may still be feasible with 3 tapers.
  - Users interested in looking at low frequencies (e.g., delta and below) may wish to use segments longer than 2 seconds.

Variables/ Inputs	Description	Examples	Notes
grp_proc_info.win_size_in_secs	Window size (in seconds) for segment creation and windowed analyses.	= 1; (default) = 2;	Used in PSD, etc.

Will high amplitude artifacts be removed prior to Segmentation (in baseline/resting data)?
Back (GUI Base Seg) Back (GUI Base Seg Gen)

- If selected, BEAPP will create a mask of all unusable data prior to Segmentation. To do so, BEAPP will scan the data in all channels for any data point that is above threshold. Upon identifying these suprathreshold data points, BEAPP will determine the nearest zero-crossing before and after that data point, for that channel. Above-threshold segments are then defined as beginning and ending at the nearest zero-crossings, rather than only including the narrower windows of time where data is suprathreshold. Segments will be created only from the usable data. Users have the option to specify whether timepoints should be marked bad if any channel has a supra-threshold value, or if a specific percentage of channels have values above threshold.

Variables/ Inputs	Description	Examples	Notes

grp_proc_info.beapp_baseline_msk_artifact	Toggles BEAPP amplitude-based artifact removal software before Segmentation.	= 1; (reject samples with any bad channels, default) = 0; (off) = 2; (reject samples with more than a % bad channels)	BEAPP artifact detection/removal occurs before Segmenting the data. This input will generate the output variable eeg_msk.
grp_proc_info.beapp_baseline_rej_perc_above_threshold	Percentage of bad channels required to mark sample bad.	= .01; (.01%, with 1000 channels, would reject sample if at least one channel is bad, default)	Only used if grp_proc_info.beapp_baseline_msk_artifact = 2.

How would you like to set event-related/conditioned baseline specifications in Segmentation?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_event_use_tags_only	Toggles the use of event codes/tags/strings only for Segmenting.	= 0; (use event codes/tags/strings and condition/cell information, default) = 1; (use event codes/tags/strings only for Segmenting (usually for .set source files))	

grp_proc_info.beapp_event_code_onset_strs	The event codes assigned during data collection to signify the onset of the stimulus.	= {''}; (default)
		= {'stm+'};
		= {'face', 'door'};
			Must be four characters.
			Read here for more examples.

grp_proc_info.beapp_event_eprime_values.condition_names	Desired condition names.	= {''}; (default) = {'Standard', 'Close', 'Far'};	Order must match cell numbers if cell sets are being used (grp_proc_info.beapp_event_use_tags_only = 0), or event tags if only event tags are being used (grp_proc_info.beapp

			_event_use_tags_only = 1). Read here for more examples.
grp_proc_info.beapp_event_eprime_values.event_codes(:,1) grp_proc_info.beapp_event_eprime_values.event_codes(:,2)	E-Prime Cell Numbers that map to grp_proc_info.beapp_event_eprime_values.condition_names.	= [1];	Only used if grp_proc_info.beapp_event_use_tags_only = 0. Read here for more examples.
grp_proc_info.beapp_event_code_offset_strs	The event codes assigned during data collection to signify the offset of the stimulus (should match onset strings).	= {''};(default) = {'stm-'}; = {'TRSP'};	Must be four characters.

Where will segments start and end in relation to the event marker of interest?

- This only needs to be specified for **event-related data**.
- This specifies how the data will be segmented, which may be different from how it will be analyzed. A segment should include all the data to be analyzed, but the analysis may include only a portion of the segment. For example, if a user wishes for the analysis window (e.g., 100 to 800 ms post-stimulus) to be compared to a baseline window (e.g.,-200 to -100ms), then the segment should start at -200 ms (or earlier), and the segment should end at 800 ms (or later). These inputs determine the size of segments saved after this module.
    - If you plan to run the ITPC module check the important note in relation to this parameter

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.evt_seg_win_start	Segment start (in seconds) for event-related segments, relative to event of interest.	= -0.1; (100 ms before the onset of event, default)	
grp_proc_info.evt_seg_win_end	Segment end (in seconds) for event-related segments, relative to event of interest.	= 0.8; (800 ms after the onset of event, default)	

Where will the analysis window start and end in relation to the event marker of interest?

- This only needs to be specified for **<u>event-related data</u>**. - If you plan to run the ITPC module check the important note in relation to these parameters			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.evt_analysis_win_start	Analysis window start (in seconds) for event-related segments, relative to event of interest.	= -0.1; (100 ms before the onset of event, default)	Must be greater than (or equal to) segment start time specified above (grp_proc_info.evt_seg_win_start).
grp_proc_info.evt_analysis_win_end	Analysis window end (in seconds) for event-related segments, relative to event of interest.	= 0.8; (800 ms after the onset of event, default)	Must be less than (or equal to) segment end time specified above (grp_proc_info.evt_seg_win_end).

Will data be baseline corrected? - If so, where will the baseline window start and end in relation to the event marker of interest? - All 3 parameters below only need to be specified for **<u>event-related data</u>**, and only if the user wishes to apply baseline correction.			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.evt_trial_baseline_removal	Toggles baseline removal for individual segments.	= 0; (off, default) = 1; (on)	
grp_proc_info.evt_trial_baseline_win_start	Baseline window start (in seconds) for event-related segments, relative to event of interest.	= -0.1; (100 ms before the onset of event, default)	Must be greater than (or equal to) segment start time specified above (grp_proc_info.evt_seg_win_start).
grp_proc_info.evt_trial_baseline_win_end	Baseline window end (in seconds) for event-related segments, relative to event of interest.	= 0; (at the onset of event, default)	Must be less than (or equal to) segment end time specified above (grp_proc_info.evt_seg_win_end).

Segmenting Module Output Variables

IMPORTANT: File won't be saved if all segments are rejected.

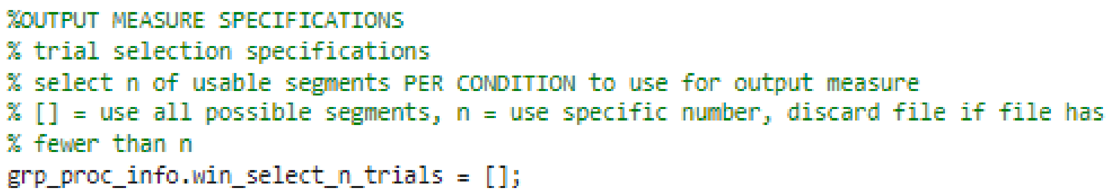
Variables/ Outputs	**Description**	**Units**	**Notes**
ICA Report Table	By running the Segmenting module, two additional variables get added to the ICA Report Table: num_segs_pre_rej and num_segs_post_rej.	N/A	
eeg_w	● n x 1 cell, where n = number of conditions. ● In each cell, you will find the amplitude of segmented EEG data in 3D matrices (electrode channels x time samples x number of segments).	Microvolts (μV)	
eeg_msk	● 1 x n cell, where n = number of epochs. ● Each cell contains an 1 x samples array of 0s and 1s. 1s represent a sample that has been flagged as an artifact.	No units	This variable is only used for baseline files. If grp_proc_info.beapp_baseline_msk_artifact = 0 (off), this variable will generate an array of zeros for each epoch.
file_proc_info	Processing information for each data file.	N/A	

OUTPUT MODULES SPECIFICATIONS

****Optional****

If the user has turned on any output modules (e.g., PSD, ITPC, or FOOOF), the following section on output module specifications determines the parameters of these output measures.

The section specifying general parameters for the output modules looks like this:



```
%OUTPUT MEASURE SPECIFICATIONS
% trial selection specifications
% select n of usable segments PER CONDITION to use for output measure
% [] = use all possible segments, n = use specific number, discard file if file has
% fewer than n
grp_proc_info.win_select_n_trials = [];
```

In this section, the user will need to specify the following information:

How many trials do you want to extract from your segmented data to run through the output modules?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.win_select_n_trials	Input number of usable segments (trials) per condition to use for output measures. Will randomly select from usable segments. Currently not supported.	= []; (use all trials, default) = 45; (select 45 trials)	The index of selected trials will be saved to file_proc_info.selected_segs. If there are not enough trials, it will save an empty array []. Currently not supported.

What are the bandwidths of interest? ● *This is only necessary if the user wants .csv reports for the PSD or ITPC modules (or other output modules in the future).* The user can specify as many (or as few) frequency bands of interest as they wish. The user is also asked to specify which frequencies they would like to include in measures of total power (used to normalize power in reports).			
Variables/ Inputs	**Description**	**Examples**	**Notes**

grp_proc_info.bw(1,1:2)	Start and end frequencies for each bandwidth you would like to analyze. You can have as many or few as you want.	grp_proc_info.bw(1,1:2) = [2,4]; (default)
		
		grp_proc_info.bw(2,1:2) = [4,6];
		
		grp_proc_info.bw(3,1:2) = [6,9];
		
		grp_proc_info.bw(4,1:2) = [9,13];
		
		grp_proc_info.bw(5,1:2) = [13,30];
		
		grp_proc_info.bw(6,1:2) = [30,50];
			Total (all frequencies between the start of the lowest bandwidth to the end of the highest) is included automatically.

grp_proc_info.bw_name(1)	Name associated with each frequency bandwidth created.	grp_proc_info.bw_name(1) = {'Delta'}; (default)	Every specified bandwidth in the script should have a name (i.e. the number of grp_proc_info.bw

		grp_proc_info.bw_name(2) = {'Theta'};
		
		grp_proc_info.bw_name(3) = {'LowAlpha'};
		
		grp_proc_info.bw_name(4) = {'HighAlpha'};
		
		grp_proc_info.bw_name(5) = {'Beta'};
		
		grp_proc_info.bw_name(6) = {'Gamma'};
			variables should be equivalent to the number of grp_proc_info.bw_name variables). Make sure the numbers between .bw and .bw_name align.

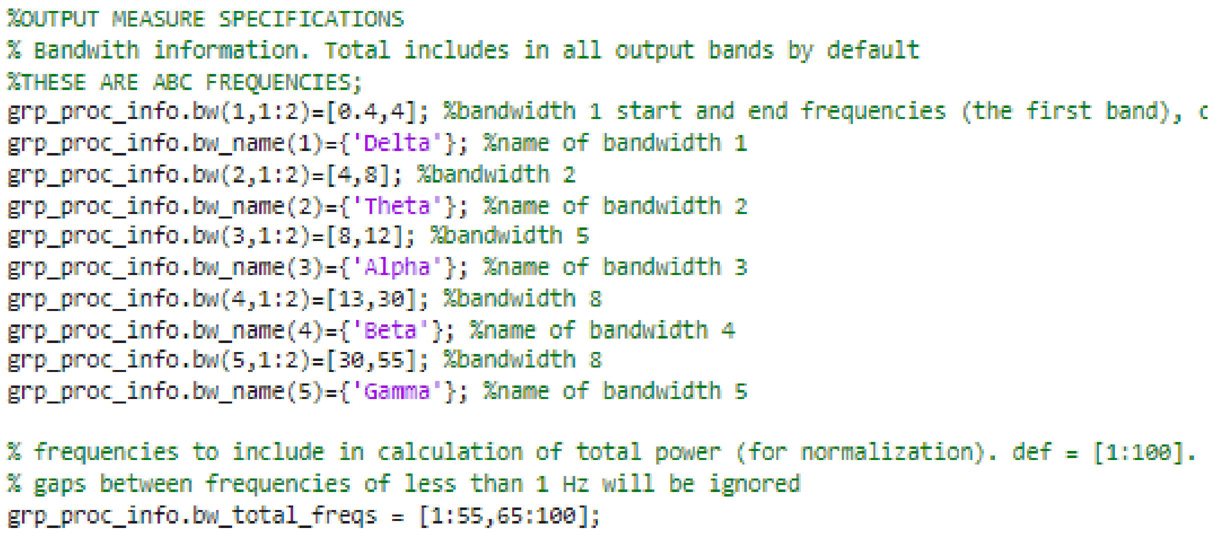
grp_proc_info.bw_total_freqs	Frequencies to include in the calculation of total power.	= [1:100]; (default) = [1.5:57,63:110];	Gaps of less than 1 Hz are ignored.

SPECIFIC OUTPUT MODULES

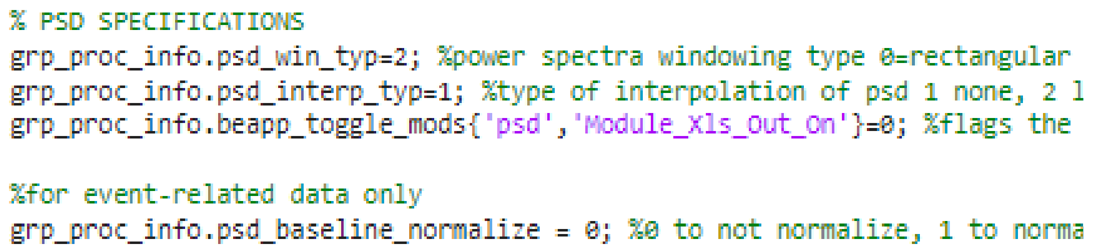
POWER SPECTRUM SPECIFICATIONS

****Optional****

The section for specifying parameters for the PSD module looks like this:



```
%OUTPUT MEASURE SPECIFICATIONS
% Bandwith information. Total includes in all output bands by default
%THESE ARE ABC FREQUENCIES;
grp_proc_info.bw(1,1:2)=[0.4,4]; %bandwidth 1 start and end frequencies (the first band), c
grp_proc_info.bw_name(1)={'Delta'}; %name of bandwidth 1
grp_proc_info.bw(2,1:2)=[4,8]; %bandwidth 2
grp_proc_info.bw_name(2)={'Theta'}; %name of bandwidth 2
grp_proc_info.bw(3,1:2)=[8,12]; %bandwidth 5
grp_proc_info.bw_name(3)={'Alpha'}; %name of bandwidth 3
grp_proc_info.bw(4,1:2)=[13,30]; %bandwidth 8
grp_proc_info.bw_name(4)={'Beta'}; %name of bandwidth 4
grp_proc_info.bw(5,1:2)=[30,55]; %bandwidth 8
grp_proc_info.bw_name(5)={'Gamma'}; %name of bandwidth 5

% frequencies to include in calculation of total power (for normalization). def = [1:100].
% gaps between frequencies of less than 1 Hz will be ignored
grp_proc_info.bw_total_freqs = [1:55,65:100];
```



```
% PSD SPECIFICATIONS
grp_proc_info.psd_win_typ=2; %power spectra windowing type 0=rectangular
grp_proc_info.psd_interp_typ=1; %type of interpolation of psd 1 none, 2 l
grp_proc_info.beapp_toggle_mods{'psd','Module_Xls_Out_On'}=0; %flags the

%for event-related data only
grp_proc_info.psd_baseline_normalize = 0; %0 to not normalize, 1 to norma
```

If the user has the power spectrum module selected, the following additional specifications are necessary:

What type of windowing should be applied to calculate the power spectrum? ● Most signal processors recommend a multitaper, if possible. This smooths the power spectrum, and mitigates edge effects. However, as described in the section on choosing segment length, some users may find that their segment lengths are too short, and frequency resolution needs too high, to allow for a multitaper to be feasible.			
Variables/ Inputs	**Description**	**Examples**	**Notes**

● When using a Hanning window, overlapping windows are not currently used. We hope to add this soon!			
grp_proc_info.psd_win_typ	Window type for power spectra: 0 = Rectangular 1 = Hanning 2 = Multitaper	= 1; (Hanning window, default) = 2; (multitaper) = 0; (rectangular window)	2+ second window length recommended for multitaper, to allow frequency resolution of 1 Hz.

Do you want BEAPP to interpolate the power spectrum?

- Most users will not choose any interpolation. Note that interpolation does not change the frequency resolution of the PSD; it simply interpolates between values output by the PSD.
- In some cases, however, users may wish to interpolate the frequency axis if they are concerned about the effects of "edge" values in binned power calculations. For example, by default, 5 bandwidths are analyzed (delta, theta, alpha, beta, and gamma) as shown here. By using interpolation, power spectrum results may be better differentiated between the 5 bandwidths.

Variables/ Inputs	Description	Examples	Notes
grp_proc_info.psd_interp_typ	Type of interpolation for the PSD: 1 = None 2 = Linear 3 = Nearest neighbor 4 = Piecewise cubic spline	= 1; (none, default) = 2; (linear) = 3; (nearest neighbor) = 4; (piecewise cubic spline)	This interpolates the power spectrum between output frequencies, but does not alter frequency resolution of the power spectrum.

Do you want BEAPP to save out a report table with PSD output summary information?

- This table would include power values in each frequency band, in each channel, for each file. Information reported can be selected by the user in the advanced user settings; see this portion of the user guide for details on settings and outputs.

Variables/ Inputs	Description	Examples	Notes
grp_proc_info.beapp_toggle_mods{'psd','Module_Xls_Out_On'}	Toggles PSD summary output table on and off.	= 1; (on, default) = 0; (off)	When on, BEAPP saves out a report table with PSD summary information.

Would you like to apply PSD baseline normalization?

Variables/ Inputs	Description	Examples	Notes

grp_proc_info.psd_baseline_normalize	Toggles PSD baseline normalization on and off.	= 0; (off, default) = 1; (normalize using decibel conversion) = 2; (normalize with percent change)	

PSD Module Output Variables

Variables/ Outputs	**Description**	**Units**	**Notes**
mean/med log or ln pwr per Hz Report	● One .csv report for each condition ● Reports the mean/median log10 or ln power per Hertz (Hz) in the delta, theta, alpha, beta, gamma, and total frequencies at each electrode channel	No units (log transform)	The name of the report begins with the name of the condition and is followed by 'mean' or 'med.'
mean/med log or ln RelativePwr Report	● One .csv report for each condition ● Reports the mean/median log10 or ln relative power in the delta, theta, alpha, beta, gamma, and total frequencies at each electrode channel.	No units (log transform) $log(\frac{Microvolts(\mu V)^2}{Hertz(Hz)})$ $ln(\frac{Microvolts(\mu V)^2}{Hertz(Hz)})$	The name of the report begins with the name of the condition and is followed by 'mean' or 'med.'
mean/med pwr per Hz Report	● One .csv report for each condition. ● Reports the mean/median power per Hertz (Hz) in the delta, theta, alpha, beta, gamma, and total frequencies at each electrode channel.	If grp_proc_info.psd_baseline_normalize = 0, the unit would be $\frac{Microvolts(\mu V)^2}{Hertz(Hz)}$. If grp_proc_info.psd_baseline_normalize = 1 or 2, the unit would be decibel (dB) or % change, respectively.	The name of the report begins with the name of the condition and is followed by 'mean' or 'med.'
mean/med RelativePwr Report	● One .csv report for each condition.	No units (units cancel out)	The name of the report begins

	• Reports the mean/median relative power in the delta, theta, alpha, beta, gamma, and total frequencies at each electrode channel.	$\frac{\frac{Microvolts(\mu V)^2}{Hertz(Hz)}}{\frac{Microvolts(\mu V)^2}{Hertz(Hz)}}$	with the name of the condition and is followed by ‘mean’ or ‘med.’
STD Raw Power	• One .csv report for each condition. • Reports the standard deviation of raw power in the delta, theta, alpha, beta, gamma, and total frequencies at each electrode channel.	Units of standard deviation	The name of the report begins with the name of the condition and is followed by ‘sd’
STD Normalized Power	• One .csv report for each condition. • Reports the standard deviation of normalized power in the delta, theta, alpha, beta, gamma, and total frequencies at each electrode channel.	Units of standard deviation	The name of the report begins with the name of the condition and is followed by ‘sd’
STD log or ln Raw Power	• One .csv report for each condition. • Reports the standard deviation of the log10 or ln raw power in the delta, theta, alpha, beta, gamma, and total frequencies at each electrode channel.	Units of standard deviation	The name of the report begins with the name of the condition and is followed by ‘sd’
STD log or ln normalized power	• One .csv report for each condition. • Reports the standard deviation of log10 or ln normalized power in the delta, theta, alpha, beta, gamma, and total frequencies at each electrode channel.	Units of standard deviation	The name of the report begins with the name of the condition and is followed by ‘sd’
eeg_wfp	• n x 1 cell, where n = number of conditions.	If grp_proc_info.psd_baseline_normalize	

	• In each cell, you will find the power of segmented EEG data in a 3D matrix (electrode channels x number of frequencies x number of segments).	= 0, the unit would be $\frac{Microvolts(\mu V)^2}{Hertz(Hz)}$. If grp_proc_info.psd_baseline_normalize = 1 or 2, the unit would be decibel (dB) or % change respectively.	
f	• n x 1 cell, where n = number of conditions. • In each cell, you will find a 2D matrix (number of frequencies x 1) with a list of frequencies that correspond to eeg_wfp.	Hertz (Hz)	
file_proc_info	Processing information for each data file.	N/A	

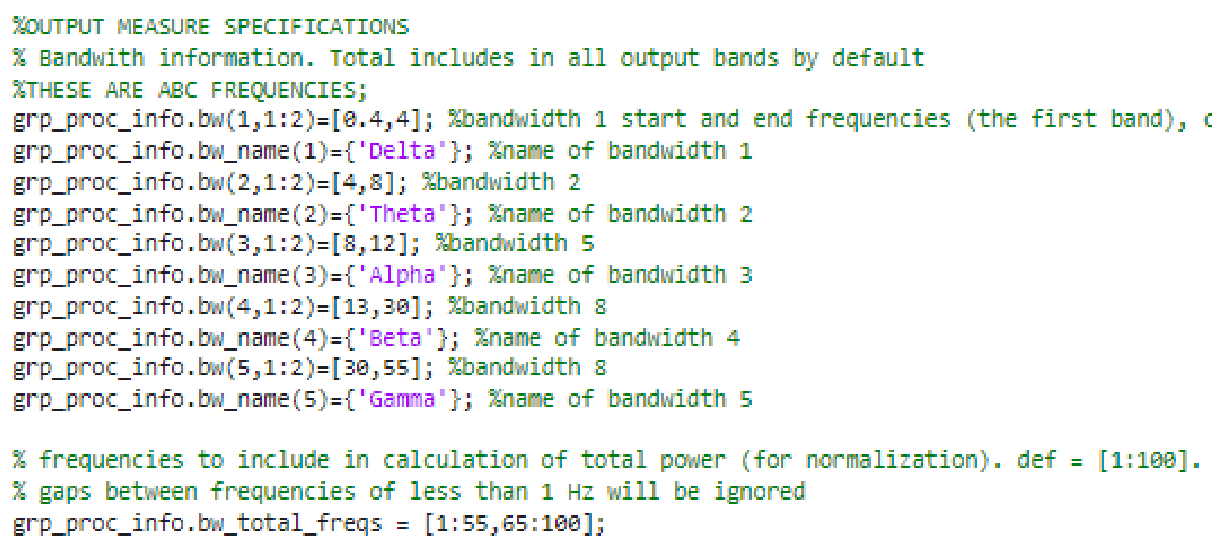
INTER-TRIAL PHASE COHERENCE (ITPC) / EVENT-RELATED SPECTRAL PERTURBATION (ERSP)

****Optional****

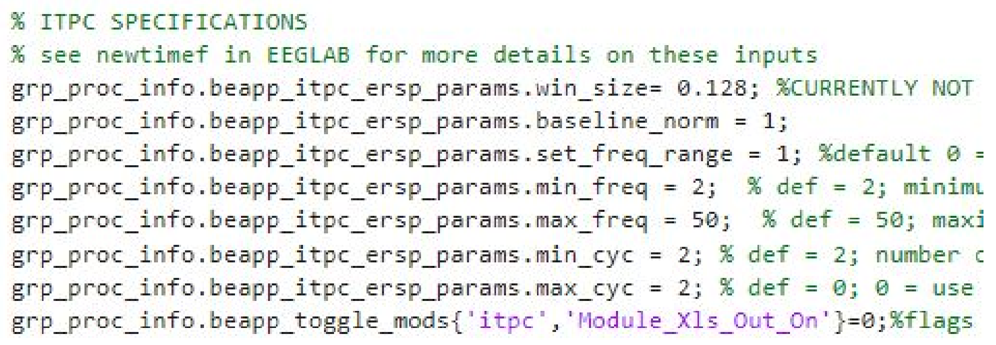
Back (Segment Analysis Window)

IMPORTANT: ITPC will analyze the window set by grp_proc_info.evt_analysis_win_start and grp_proc_info.evt_analysis_win_end. Make sure this analysis window is the same or smaller than the segmentation window (grp_proc_info.evt_seg_win_start and grp_proc_info.evt_seg_win_end)**.** The analysis window also needs to be long enough to resolve frequencies of interest, see here for more information.

The section for specifying parameters for the ITPC module looks like this:



```
%OUTPUT MEASURE SPECIFICATIONS
% Bandwith information. Total includes in all output bands by default
%THESE ARE ABC FREQUENCIES;
grp_proc_info.bw(1,1:2)=[0.4,4]; %bandwidth 1 start and end frequencies (the first band), c
grp_proc_info.bw_name(1)={'Delta'}; %name of bandwidth 1
grp_proc_info.bw(2,1:2)=[4,8]; %bandwidth 2
grp_proc_info.bw_name(2)={'Theta'}; %name of bandwidth 2
grp_proc_info.bw(3,1:2)=[8,12]; %bandwidth 5
grp_proc_info.bw_name(3)={'Alpha'}; %name of bandwidth 3
grp_proc_info.bw(4,1:2)=[13,30]; %bandwidth 8
grp_proc_info.bw_name(4)={'Beta'}; %name of bandwidth 4
grp_proc_info.bw(5,1:2)=[30,55]; %bandwidth 8
grp_proc_info.bw_name(5)={'Gamma'}; %name of bandwidth 5

% frequencies to include in calculation of total power (for normalization). def = [1:100].
% gaps between frequencies of less than 1 Hz will be ignored
grp_proc_info.bw_total_freqs = [1:55,65:100];
```



```
% ITPC SPECIFICATIONS
% see newtimef in EEGLAB for more details on these inputs
grp_proc_info.beapp_itpc_ersp_params.win_size= 0.128; %CURRENTLY NOT
grp_proc_info.beapp_itpc_ersp_params.baseline_norm = 1;
grp_proc_info.beapp_itpc_ersp_params.set_freq_range = 1; %default 0 =
grp_proc_info.beapp_itpc_ersp_params.min_freq = 2;  % def = 2; minimu
grp_proc_info.beapp_itpc_ersp_params.max_freq = 50;  % def = 50; maxi
grp_proc_info.beapp_itpc_ersp_params.min_cyc = 2; % def = 2; number c
grp_proc_info.beapp_itpc_ersp_params.max_cyc = 2; % def = 0; 0 = use
grp_proc_info.beapp_toggle_mods{'itpc','Module_Xls_Out_On'}=0;%flags
```

If the user wishes to run the Inter-Trial Phase Coherence(ITPC)/Event-Related (log) Spectral Perturbation (ERSP) module, the following specifications are necessary:

Would you like to apply baseline normalization to the estimation of ERSP?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_itpc_ersp_params.baseline_norm	Toggles baseline normalization on and off.	= 1; (on, default) = 0; (off)	

For what frequency ranges do you want to compute ITPC and ERSP?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_itpc_ersp_params.set_freq_range	Toggles frequency range on and off.	= 0; (off, default) = 1; (on)	

grp_proc_info.beapp_itpc_ersp_params.min_freq Back (ITPC)	Minimum frequency in Hz.	= 2; (default)	The minimum frequency you can resolve may be limited by segment length. We recommend making sure you have a segment length long enough to observe at least 3-4 cycles of your minimum frequency (e.g for a minimum frequency of 2 hz, you would want a 3-4 second long segment).
grp_proc_info.beapp_itpc_ersp_params.max_freq	Maximum frequency in Hz.	= 80; (default)	

How would you like to set the cycle specifications for the time-frequency decomposition?

- Before ITPC and ERSP can be computed, data goes through time-frequency decomposition using Morlet wavelets. The parameters below specify the parameters of the Morlet wavelet used for the decomposition.

Variables/ Inputs	Description	Examples	Notes
grp_proc_info.beapp_itpc_ersp_params.min_cyc	The number of cycles in each Morlet wavelet.	= 2; (default)	
grp_proc_info.beapp_itpc_ersp_params.max_cyc	How the number of cycles changes as frequency increases.	= 0; (use same window size for all frequencies, default) = 1; (cycles do not increase) = > 0 and < 1; (cycles increase linearly with frequency) = > 1; (cycles set to maximum: the number of cycles used at the maximum frequency. Used under the	

		assumption that you would like the number of cycles to increase as frequency increases)	

Do you want BEAPP to save out a report table with ITPC output summary information?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_toggle_mods{'itpc','Module_Xls_Out_On'}	Toggles ITPC summary output table on and off.	= 1; (on, default) = 0; (off)	When on, BEAPP saves a report table with ITPC summary information.

What window size would you like to calculate ERSP and ITPC? **IMPORTANT:** This feature is currently unavailable in BEAPP.			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_itpc_ersp_params.win_size	Window size (in seconds) to calculate ERSP and ITPC from the ERPs of the composed dataset.	= 0.128; (default)	This feature is currently unavailable in BEAPP. ITPC uses a Morlet wavelet which is set by the min_cyc and max_cyc, instead of a Hanning window.

ITPC/ERSP Module Output Variables

Variables/ Outputs	**Description**	**Units**	**Notes**
av_AbsITPC Report	• One .csv report for each condition. • Reports the average absolute ITPC in the delta, theta, alpha, beta, gamma, and total frequencies at each electrode channel.	No units	

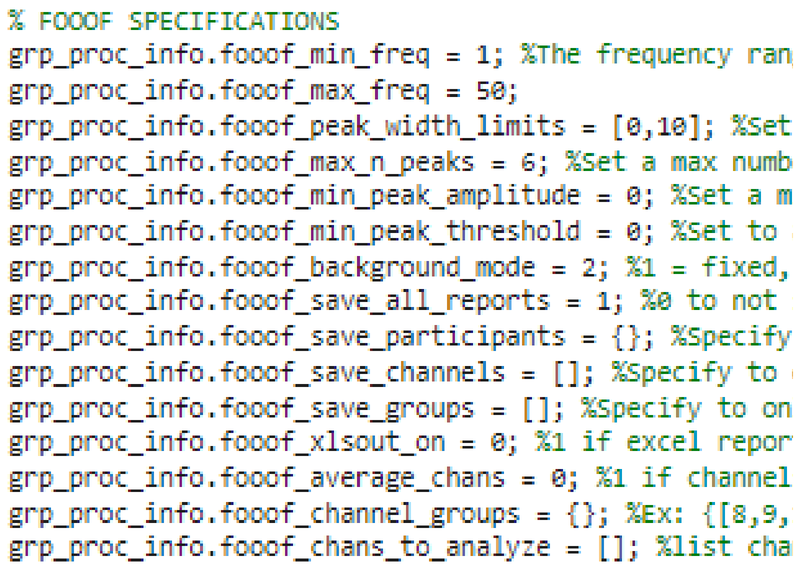
max_AbsITPC Report	<ul><li>One .csv report for each condition.</li><li>Reports the maximum absolute ITPC in the delta, theta, alpha, beta, gamma, and total frequencies at each electrode channel.</li></ul>	No units	
eeg_itc	<ul><li>n x 1 cell, where n = number of conditions.</li><li>In each cell, you will find a 3D matrix of complex inter-trial coherencies (electrode channels x frequency bins x time bins).</li></ul>	No units	Time bins reflect the spectral time window centers.
ERSP	<ul><li>n x 1 cell, where n = number of conditions.</li><li>In each cell, you will find a 3D matrix of log spectral diffs from baseline (electrode channels x frequency bins x time bins).</li></ul>	Decibels (dB)	Time bins reflect the spectral time window centers.
f	<ul><li>n x 1 cell, where n = number of conditions.</li><li>In each cell, you will find a vector of frequency bin centers.</li></ul>	Hertz (Hz)	
f_powbase	A vector of frequency bin centers	Hertz (Hz)	
powbase	<ul><li>n x 1 cell, where n = number of conditions.</li><li>In each cell, you will find a 2D matrix of baseline power spectrum (electrode channels x frequency bins).</li></ul>	$\frac{Microvolts(\mu V)^2}{Hertz(Hz)}$	
t	<ul><li>n x 1 cell, where n = number of conditions.</li><li>In each cell, you will find a vector of output times (spectral time window centers).</li></ul>	Milliseconds (ms)	
file_proc_info	Processing information for each data file.	N/A	

FOOOF SPECIFICATIONS

****Optional****

IMPORTANT: In order to run FOOOF, users must run the PSD module first. More background information on running FOOOF, can be found here.

The section for specifying parameters for the FOOOF module looks like this:



```
% FOOOF SPECIFICATIONS
grp_proc_info.fooof_min_freq = 1; %The frequency ran
grp_proc_info.fooof_max_freq = 50;
grp_proc_info.fooof_peak_width_limits = [0,10]; %Set
grp_proc_info.fooof_max_n_peaks = 6; %Set a max numb
grp_proc_info.fooof_min_peak_amplitude = 0; %Set a m
grp_proc_info.fooof_min_peak_threshold = 0; %Set to
grp_proc_info.fooof_background_mode = 2; %1 = fixed,
grp_proc_info.fooof_save_all_reports = 1; %0 to not
grp_proc_info.fooof_save_participants = {}; %Specify
grp_proc_info.fooof_save_channels = []; %Specify to
grp_proc_info.fooof_save_groups = []; %Specify to on
grp_proc_info.fooof_xlsout_on = 0; %1 if excel repor
grp_proc_info.fooof_average_chans = 0; %1 if channel
grp_proc_info.fooof_channel_groups = {}; %Ex: {[8,9,
grp_proc_info.fooof_chans_to_analyze = []; %list cha
```

If the user wishes to obtain data on FOOOF, the following specifications are necessary:

What frequency range of the power spectrum do you want FOOOF to run on?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.fooof_min_freq	The minimum of the frequency range of the power spectrum FOOOF will run on.	= 1; (default)	
grp_proc_info.fooof_max_freq	The maximum of the frequency range of the power spectrum FOOOF will run on.	= 50; (default)	

What are the lower and upper bound bandwidth limits for the peaks FOOOF finds?

- Use to prevent overfitting – A lower bound limit that is too low may lead to overfitting noise as small bandwidth peaks. To prevent that from happening, we recommend a lower bound of approximately 2x the frequency resolution. A lower bound limit < the frequency resolution of the power spectrum will not have any effect. More information can be found here.

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.fooof_peak_width_limits	Peak bandwidth limits to prevent overfitting.	= [0,10]; (default)	The first number is the lower bound limit.

			The second number is the upper bound limit. Needs to be > frequency resolution.

What is the maximum number of peaks FOOOF should find? ● Use to prevent overfitting.			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.fooof_ max_n_peaks	Maximum number of peaks FOOOF should find.	= 6; (default)	Some maximum **must** be set.

What is the minimum amplitude for the peaks FOOOF must find? ● Use to prevent overfitting.			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.fooof_ min_peak_amplitude	Minimum peak amplitude for FOOOF to find.	= 0; (default)	Set to 0 to not set a minimum.

What is the minimum threshold for the peaks FOOOF must find? ● This parameter evaluates whether a data point may be considered a peak. It is measured in units of standard deviation above the noise of the flattened spectrum. In order for a data point to be considered a peak, it must pass the threshold. Recommended to be set to 0 when no peaks are expected to prevent overfitting.			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.fooof_ min_peak_threshold	Minimum peak threshold (usually > 0).	= 0; (default)	Recommended to set if PSD has peaks; otherwise, keep 0 to not set a threshold.

What background method would you like FOOOF to use? ● To fit with just an offset and a slope (equivalent to a linear fit in log-log space), use 1 (fixed). To fit with a bend in log-log space, select 2 (knee). Note, fitting with a knee leads to a 3rd background parameter output (called 'knee') in addition to the offset and slope outputs.			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.fooof_ background_mode	Background method: 1 = fixed	= 2; (default) = 1;	If freq range is ~40Hz or below,

	2 = knee		recommended to use 'fixed'; otherwise, use 'knee'.

What reports should be saved?

- Reports are figures showing your original power spectrum, FOOOF's background fit, and FOOOF's overall fit. Since FOOOF is run on many channels and many participants, saving all reports can be overwhelming, so BEAPP has options to save reports for only certain participants or only certain channels or channel groups. If fooof_save_all_reports = 1, all reports will be saved regardless of the other specifications. If fooof_save_all_reports = 0, and certain participants, channels, or channel groups are specified, only those corresponding reports will be saved. To not save any reports, keep fooof_save_all_reports as 0 and do not specify any participants, channels, or channel groups.

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.fooof_save_all_reports	Toggles saving all reports or not.	= 1; (on → save all, default) = 0; (off)	
grp_proc_info.fooof_save_participants	Specify which participants' reports should be saved.	= {}; (default) = {'baselineEEG01.mat'};	To not specify participants, = {};
grp_proc_info.fooof_save_channels	Specify to only save reports for some channel #'s.	= []; (default) = [1,2];	To not specify channels, = [];
grp_proc_info.fooof_save_groups	Specify to only save reports for group #'s specified.	= []; (default) = [1,3];	To not specify groups, = [];

Should an excel output be generated and saved?

- Excel outputs will contain the same output information as .mat, but may be easier to use for some users.

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.fooof_xlsout_on	Toggles generating an excel output.	= 0; (off, default) = 1; (on)	

Should channels be averaged all together, in different groups, or not at all?

- *To average all channels together*, set fooof_average_chans = 1 and fooof_channel_groups = {}. *To average in groups*, set fooof_average_chans = 1 and list the channels to group together in the same [].

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.fooof_average_chans	Toggles averaging all channels together or running separately.	= 0; (off → separate, default) = 1; (on → average)	
grp_proc_info.fooof_channel_groups Back (Save Groups)	If averaging is on (.fooof_average_chans = 1), specify which channels should be averaged in separate groups.	= {}; (default) = {[8,9,10],[15,16,17,18,19,20], [25,26,27]};	To average all channels together, = {};

Should only certain channels be analyzed?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.fooof_chans_to_analyze	List channels to analyze if only some should be selected.	= []; (default) = [19,31,37];	To analyze all channels, = [];

FOOOF Module Outputs

Variables/ Outputs	**Description**	**Units**	**Notes**
Channel #s' Images	Figures of fitted power spectrums for selected channels.	N/A	PNG images
Excel FOOOF Report	Combination of the MATLAB fooof_results' variables in an Excel file. Cells hold values of model fit parameters.	N/A	To generate, set grp_proc_info.fooof_save_all_reports =1; OR specify at least one of the these variables: • grp_proc_info.fooof_save_participants • grp_proc_info.fooof_save_channels • grp_proc_info.fooof_save_groups
fooof_results.mat			
column_labels	Categories of FOOOF measurements saved/extracted.	N/A	Each title corresponds to a matching column of values in

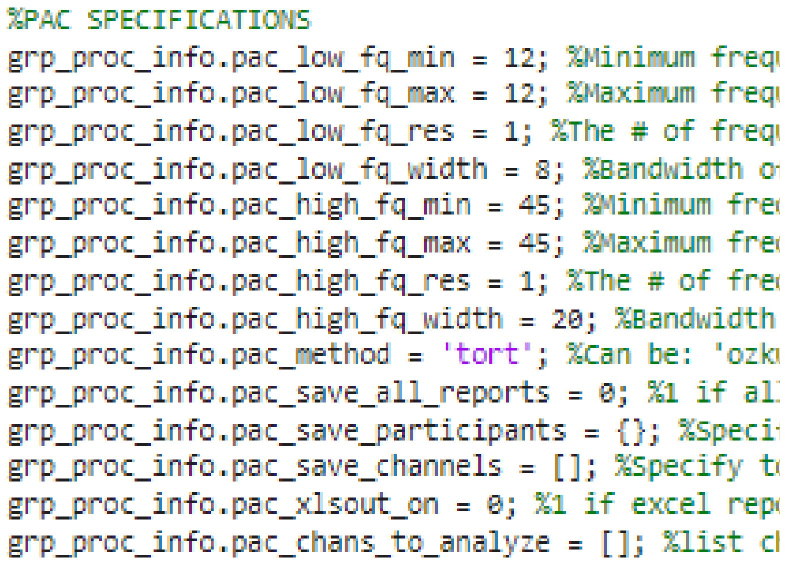
			fooof_results.
fooof_results	Values of peak parameters and measurements of validity of model fit across peaks.	N/A	The columns and rows for each cell corresponds with the column_labels and row_labels variables.
row_labels	Complete list of channels.	N/A	Each channel corresponds to a matching row of values (or NaNs) in the fooof_results variable.
third_dimension_label s	Name of source data file.	N/A	Will be included in the title for all figures and used to differentiate between files in FOOOF Reports.

PHASE-AMPLITUDE COUPLING SPECIFICATIONS

****Optional****

The Phase Amplitude Coupling (PAC) module uses the comodulogram function within pactools (https://github.com/pactools/pactools). It generates a correlation matrix between a series of lower frequencies and a series of higher frequencies. This is defined as exploratory phase-amplitude coupling, which is useful when you have no reason to select either frequency band, or you want to confirm the specificity of a known correlation between two frequencies.

The section for specifying parameters for the PAC module in the user_script looks like this:



```
%PAC SPECIFICATIONS
grp_proc_info.pac_low_fq_min = 12; %Minimum freq
grp_proc_info.pac_low_fq_max = 12; %Maximum freq
grp_proc_info.pac_low_fq_res = 1; %The # of freq
grp_proc_info.pac_low_fq_width = 8; %Bandwidth o
grp_proc_info.pac_high_fq_min = 45; %Minimum fre
grp_proc_info.pac_high_fq_max = 45; %Maximum fre
grp_proc_info.pac_high_fq_res = 1; %The # of fre
grp_proc_info.pac_high_fq_width = 20; %Bandwidth
grp_proc_info.pac_method = 'tort'; %Can be: 'ozk
grp_proc_info.pac_save_all_reports = 0; %1 if al
grp_proc_info.pac_save_participants = {}; %Speci
grp_proc_info.pac_save_channels = []; %Specify t
grp_proc_info.pac_xlsout_on = 0; %1 if excel rep
grp_proc_info.pac_chans_to_analyze = []; %list c
```

If the user wishes to obtain data on PAC, the following specifications are necessary:

What is the minimum, maximum, and resolution of the list of lower frequencies you want to analyze in the comodulogram?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.pac_low_fq_min	The minimum of the lower frequency range to calculate the comodulogram on.	= 1; (default)	
grp_proc_info.pac_low_fq_max	The maximum of the lower frequency range to calculate the comodulogram on.	= 10; (default)	
grp_proc_info.pac_low_fq_res	The # of frequencies to calculate between the min and max of the lower frequency range: in other words, the resolution of the lower frequency range.	= 10; (default)	To calculate PAC for frequencies 1-10, set min to 1, max to 10, and res to 10. Depending on MATLAB's memory settings, users may have set the resolution to a smaller value.
grp_proc_info.low_fq_width	Bandwidth of the bandpass filter for	= 2.0; (default)	

	each frequency in the low frequency range.		

What is the minimum, maximum, and resolution of the list of higher frequencies to analyze in the comodulogram?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.pac_high_fq_min	The minimum of the higher frequency range to calculate the comodulogram on.	= 10; (default)	
grp_proc_info.pac_high_fq_max	The maximum of the higher frequency range to calculate the comodulogram on.	= 125; (default)	
grp_proc_info.pac_high_fq_res	The # of frequencies to calculate between the min and max of the higher frequency range; in other words, the resolution of the higher frequency range.	= 10; (default)	Depending on MATLAB's memory settings, users may have set the resolution to a smaller value.
grp_proc_info.pac_high_fq_width	Bandwidth of the bandpass filter for each frequency in the high frequency range.	= 20; (default)	

Which method would you like to use to generate the correlations between each frequency?

IMPORTANT: 'tort' is the only method supported to run in BEAPP.

- 'ozkurt', 'canolty', 'tort', 'penny' = methods based on filtering and using a Hilbert transform.
- 'vanwijk' = method for a joint AAC and PAC estimation based on filtering and using the Hilbert transform.
- 'sigl', 'nagashima', 'hagihira', 'bispectrum' = methods for a PAC estimation based on bicoherence.
- 'colgin', 'jiang' = PAC / PAC directionality estimation (respectively) based on filtering and computing coherence.
- 'deprelatour' = PAC estimation based on a driven autoregressive model.
- More information on methods can be found here.

Variables/ Inputs	**Description**	**Examples**	**Notes**

grp_proc_info.pac_method	The method you would like to use to generate the comodulogram.	= 'tort'; (default)	Needs to be in string format.

What reports should be saved?

- Reports are figures showing a heatmap of the correlation matrix between all frequencies within the lower and higher frequency range. Since PAC is run on many channels and many participants, saving all reports can be overwhelming, so BEAPP has options to save reports for only certain participants or only certain channels. If pac_save_all_reports = 1, all reports will be saved regardless of the other specifications. If pac_save_all_reports = 0, and certain participants, channels, or channel groups are specified, only those corresponding reports will be saved. To not save any reports, keep pac_save_all_reports as 0 and do not specify any participants or channels to save.

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.pac_save_all_reports	Toggles saving all PAC reports or not.	= 1; (on → save, default) = 0; (off)	
grp_proc_info.pac_save_participants	Specify which participants to save.	= {}; (default) = {'baselineEEG01.mat'};	To save all participants, = {};
grp_proc_info.pac_save_channels	Specify which channels to save.	= []; (default) = [1,2];	To save all channels, = [];

Should an excel output be generated and saved?

- Excel outputs will contain the same output information as .mat, but may be easier to use for some users.

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.pac_xlsout_on	Toggles generating an excel output for PAC measurements.	= 0; (off, default) = 1; (on)	

If you only want to analyze specific channels, what channels should be analyzed?

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.pac_chans_to_analyze	List channels to analyze if only some	= []; (default) = [1,2,6];	To analyze all channels, = [];

	channels should be selected.		

PAC Module Outputs

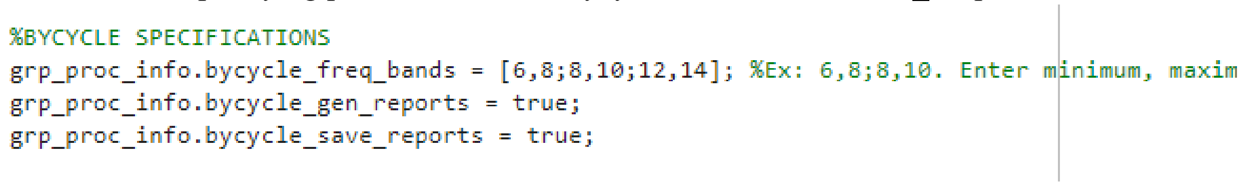
Variables/ Outputs	Description	Units	Notes
Image Outputs Folder	Folder of comodulograms (coupling maps/heatmaps) for each channel selected.	N/A	To generate this, set grp_proc_info.pac_save_all_reports=1; PNG images
Excel PAC Report	An Excel file of the frequency labels for each column/row across all comodulograms.	Hertz (Hz)	To generate this, set grp_proc_info.pac_xlsout_on = 1; Row 1 holds Phase Frequency values (x-axis). Column A holds Amplitude Frequency values (y-axis).
Excel Channel #s' PAC Report	An Excel file of the coupling strength between frequencies, where each value corresponds to a matching square in the channel's comodulogram.	No units	To generate this, set grp_proc_info.pac_xlsout_on = 1; There should be an Excel file for each channel selected in the user inputs.
pac_results.mat			
comodulogram	4D matrix of estimated PAC metrics between select frequencies for all channels and segments.	N/A	4 dimensions: high frequency resolution x low frequency resolution x channels x segments.
comodulogram_column_headers	The low frequency range values for each comodulogram (phase frequencies).	Hertz (Hz)	All values should be in row 1. Each value corresponds to a vertical column in the comodulograms.

comodulogram_row_headers	The high frequency range values for each comodulogram (amplitude frequencies).	Hertz (Hz)	All the values should be in column 1. Each value corresponds to a horizontal row in the comodulograms.
comodulogram_third_dim_headers	List of channels analyzed in string format.	N/A	These values will be attached as a tag in the file name for each PAC report generated.
file_proc_info	Processing information for each data file.	N/A	
phase_bias_comod	Phase angle coupling measurements between frequencies.	No units	3 dimensions: high frequency resolution x low frequency resolution x channels.
rawmi_comod	Modulation indices/ coupling strength between frequencies.	No units	3 dimensions: high frequency resolution x low frequency resolution x channels.
zscore_comod	Zscore of PAC normed modulation index.	No units	3 dimensions: high frequency resolution x low frequency resolution x channels. This variable is filled with NaNs unless grp_proc_info.pac_calc_zscore = 1 (see PAC Advanced settings).

BYCYCLE SPECIFICATIONS

****Optional****

The section for specifying parameters for the bycycle module in the user_script looks like this:



```
%BYCYCLE SPECIFICATIONS
grp_proc_info.bycycle_freq_bands = [6,8;8,10;12,14]; %Ex: 6,8;8,10. Enter minimum, maxim
grp_proc_info.bycycle_gen_reports = true;
grp_proc_info.bycycle_save_reports = true;
```

If the user wishes to run the bycycle module, the following specifications are necessary:

What frequency bands do you want to analyze?

Variables/ Inputs	Description	Examples	Notes
grp_proc_info.bycycle_freq_bands	Enter minimum, maximum of each frequency range to analyze.	= [12,14]; (default) = [6,8;8,10];	Separate ranges with a semicolon.

Do you want to generate and save figures?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.bycycle_gen_reports	Generate figures including bycycle metrics over the input signal and histograms of bycycle metrics.	= 1; (default)	
grp_proc_info.bycycle_save_reports	Toggles saving figures generated by grp_proc_info.bycycle_gen_reports	= 1; (default) = 0;	

BYCYCLE Module Outputs

The output from the Bycycle module is the results cell array, where each row corresponds to each condition analyzed, and each column corresponds to each frequency band analyzed (see frequencies). Within each cell, the Bycycle metrics are stored in a 2D matrix of cycles x segments. Bycycle outputs several metrics on a per cycle basis. Each cycle's metrics are stored as a row. Each column corresponds to each metric – see var_names for more details. All segments' cycle statistics are appended together into one 2D array (stored as a cell in the results cell array).

Variables/ Outputs	Description	Units	Notes
Image Outputs Folder	Folder of histogram and time series figures for each data segment across all selected channels.		To generate this folder, set grp_proc_info.bycycle_gen_reports = true; **AND** grp_proc_info.bycycle_save_reports = true; The title of each figure will include the data's file name, channel number,

			segment number, and histogram/time series label.
frequencies	The frequency bands selected for analysis from grp_proc_info.bycycle_freq_bands	Hertz (Hz)	
results	Cycle statistics separated by frequency band. Each column corresponds to a value in the ‘frequencies’ variable.		Within each cell is a 2D matrix of cycle metrics x segments. Each column corresponds to a value from the ‘var_names’ variable.
var_names	A row of Bycycle metric labels associated with the columns of the ‘results’ variable.		

Advanced User Settings

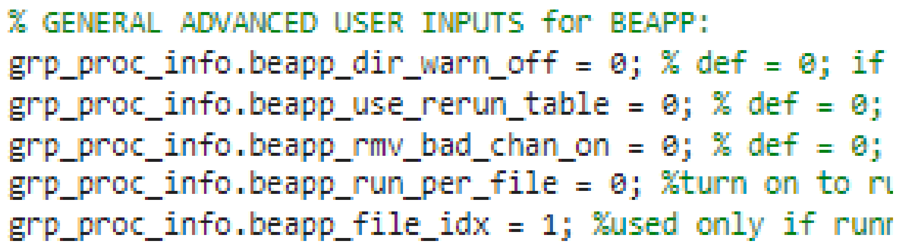
****Optional****

Back (Adv Input Toggle)

Advanced user inputs can be found in beapp\user_inputs\beapp_advinputs.m. These will only be applied if grp_proc_info.adv_inputs = 1 in the standard user inputs script. Otherwise, the defaults will be used for these values. Advanced users not wanting to use more than one or two advanced inputs can also copy variables to the standard user inputs script if they want to set certain values without turning the advanced inputs on.

Advanced General Variables/Inputs

The section for specifying the General Advanced User Inputs settings in the advanced input script looks like this:



```
% GENERAL ADVANCED USER INPUTS for BEAPP:
grp_proc_info.beapp_dir_warn_off = 0; % def = 0; if
grp_proc_info.beapp_use_rerun_table = 0; % def = 0;
grp_proc_info.beapp_rmv_bad_chan_on = 0; % def = 0;
grp_proc_info.beapp_run_per_file = 0; %turn on to ru
grp_proc_info.beapp_file_idx = 1; %used only if runr
```

If the user wishes to run BEAPP using the advanced user script, the following specifications are necessary:

Mute directory warnings?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_dir_warn_off	Toggles muting BEAPP directory warnings that require user input.	= 0; (off → no mute, default) = 1; (on → mute)	If this variable is on, BEAPP will not alert the user when there is a risk of data being overridden.

Only use a subset of files previously run during a rerun? ● By default, this uses the rerun_fselect_table.mat in user_inputs to determine which files to rerun. See Rerunning BEAPP Modules for additional information.			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_use_rerun_table	Toggles using a file selection table to rerun a subset of files.	= 0; (off, default) = 1; (on)	Optional, not needed for normal reruns.

Replace bad channels with NaN instead of interpolating?

- Users may choose to do this if they would like to run analyses without interpolated channels.

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_rmv_bad_chan_on	Toggles removing channels flagged as bad in PREP or HAPPE.	= 0; (off, default) = 1; (on)	Only applied in PREP or HAPPE. If on, will replace bad channels with NaN instead of interpolating.

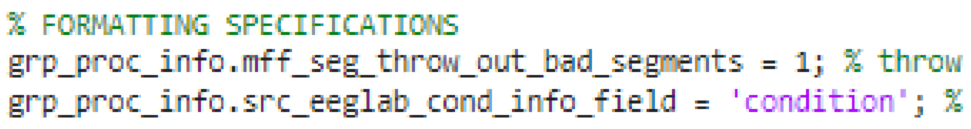
Should BEAPP run only 1 file?

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_run_per_file	Toggles running only 1 file in beapp_file_info_table.	= 0; (off, default) = 1; (on)	Intended for computer cluster use.
grp_proc_info.beapp_file_idx	The index for the file in beapp_file_info_table that you would like to run through BEAPP.	= 1; (the first file in beapp_file_info_table)	Used only if grp_proc_info.beapp_run_per_file = 1.

Advanced Inputs by Modules

Formatting Specifications

The section for specifying the Advanced Formatting settings in the advanced input script looks like this:



```
% FORMATTING SPECIFICATIONS
grp_proc_info.mff_seg_throw_out_bad_segments = 1; % throw
grp_proc_info.src_eeglab_cond_info_field = 'condition'; %
```

In this section, users can choose to specify the following parameters:

Throw out segments previously marked bad (during hand editing, for example) during import?

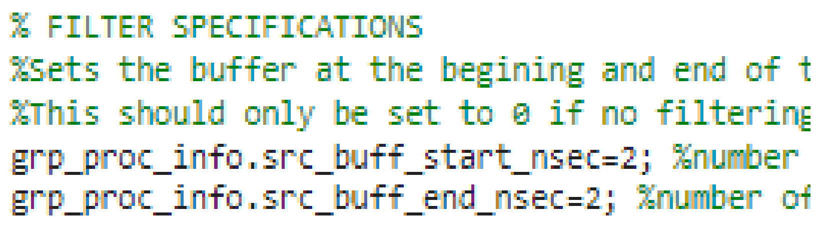
Variables/ Inputs	**Description**	**Examples**	**Notes**

grp_proc_info.mff_seg_throw_out_bad_segments	Toggle throwing out segments previously marked bad during import.	= 1; (on, default) = 0; (off)	Only used during Format for pre-segmented .mff's.

If using EEGLAB .set files, what is the field name with condition information?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.src_eeglab_cond_info_field	Name of field with condition information for EEGLAB .set files.	= 'condition'; (default) = 'cel_type';	The examples correspond to the field names .condition and .cel_type.

Filter Specifications

The section for specifying the Advanced Filter settings in the advanced input script looks like this:



```
% FILTER SPECIFICATIONS
%Sets the buffer at the begining and end of t
%This should only be set to 0 if no filtering
grp_proc_info.src_buff_start_nsec=2; %number
grp_proc_info.src_buff_end_nsec=2; %number of
```

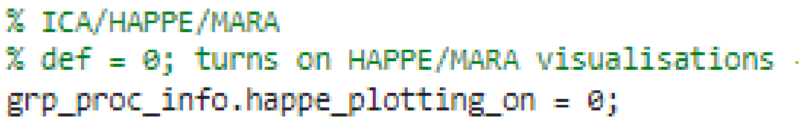
In this section, users can choose to specify the following parameters:

When making segments, how long do you want to set the buffer at the beginning and end of the source files?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.src_buff_start_nsec	Number of seconds to buffer at the start of EEG recording to be excluded after filtering and artifact removal.	= 2; (default)	Recommended value ≥ 2.
grp_proc_info.src_buff_end_nsec	Number of seconds to buffer at the end of the EEG recording to be excluded after	= 2; (default)	Recommended value ≥ 2.

	filtering and artifact removal.		

ICA/HAPPE/MARA Specifications

The section for specifying the Advanced ICA/HAPPE/MARA settings in the advanced input script looks like this:



```
% ICA/HAPPE/MARA
% def = 0; turns on HAPPE/MARA visualisations ·
grp_proc_info.happe_plotting_on = 0;
```

In this section, users can choose to specify the following parameters:

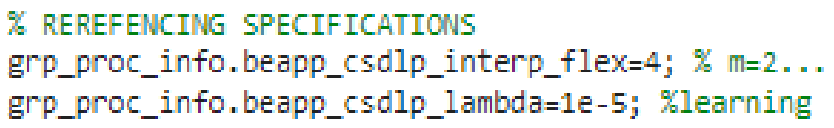
Would you like to visualize each processing step in HAPPE?

- If advanced users choose to toggle on the visualizations provided by MARA within HAPPE, users must manually confirm or adjust ICA components to reject.

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.happe_ plotting_on	Toggles HAPPE visualizations on or off.	= 0; (off, default) = 1; (on)	Allows visualization of each processing step in HAPPE. If this is on, each file will require user input to confirm or modify the rejection of ICA components during HAPPE.

Re-referencing Specifications

The section for specifying the Advanced Re-referencing settings in the advanced input script looks like this:



```
% REREFENCING SPECIFICATIONS
grp_proc_info.beapp_csdlp_interp_flex=4; % m=2...
grp_proc_info.beapp_csdlp_lambda=1e-5; %learning
```

In this section, users can choose to specify the following parameters:

If using the laplacian referencing method, what spline flexibility and smoothing constant would you like to apply to your dataset?

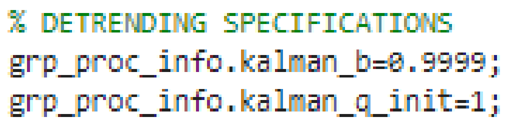
- The following variables are only applicable if grp_proc_info.reref_typ = 3.
- These variables will only be used in the CSD toolbox.

- For more background information on the CSD toolbox and the following input parameters go here.

Variables/ Inputs	Description	Examples	Notes
grp_proc_info.beapp_csdlp_interp_flex	The spline interpolation constant, which determines spline flexibility.	= 4; (default)	Valid values range from 2 (most flexible) to 10 (most rigid). Changing this value will affect the spatial resolution of the surface Laplacians.
grp_proc_info.beapp_csdlp_lambda	Smoothing constant.	= 1e-5; (default)	Changing this value will affect the spatial resolution of the surface Laplacians.

Detrending Specifications

The section for specifying the Advanced Detrending settings in the advanced input script looks like this:



```
% DETRENDING SPECIFICATIONS
grp_proc_info.kalman_b=0.9999;
grp_proc_info.kalman_q_init=1;
```

In this section, users can choose to specify the following parameters:

How would you like to set Kalman filter specifications?

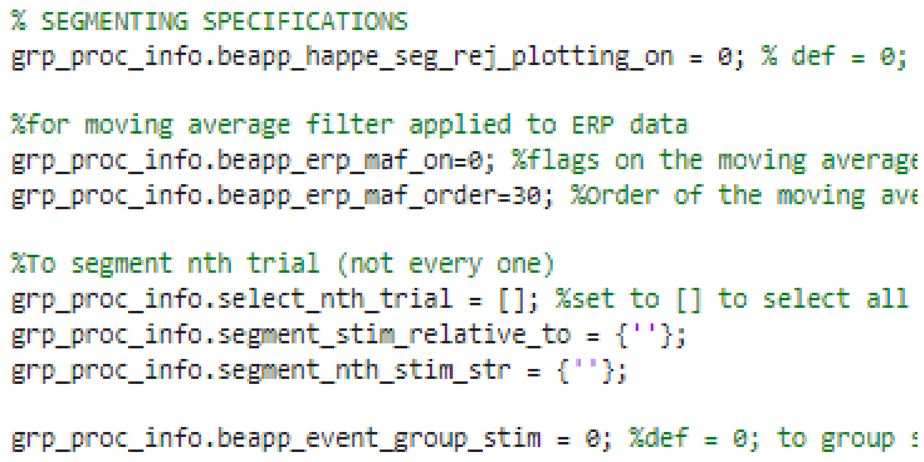
- The following parameters allow users to adjust the settings for Kalman filter applied in the Detrending module see more information on Kalman filtering here

Variables/ Inputs	Description	Examples	Notes
grp_proc_info.kalman_b	b-value for the Kalman filter detrend option.	= 0.9999; (default)	Value chosen will depend on the sampling rate. Use caution; lower b-values will improve fitting (and improve number of usable segments) but may excessively attenuate low and even mid-frequency oscillations.

grp_proc_info.q_init	Determines smoothing in the Kalman filter.	= 1; (default)	

Segmenting Specifications

The section for specifying the Advanced Segmenting settings in the advanced input script looks like this:



```
% SEGMENTING SPECIFICATIONS
grp_proc_info.beapp_happe_seg_rej_plotting_on = 0; % def = 0;

%for moving average filter applied to ERP data
grp_proc_info.beapp_erp_maf_on=0; %flags on the moving average
grp_proc_info.beapp_erp_maf_order=30; %Order of the moving ave

%To segment nth trial (not every one)
grp_proc_info.select_nth_trial = []; %set to [] to select all
grp_proc_info.segment_stim_relative_to = {''};
grp_proc_info.segment_nth_stim_str = {''};

grp_proc_info.beapp_event_group_stim = 0; %def = 0; to group s
```

In this section, users can choose to specify the following parameters:

How would you like to apply additional methods to the Segmenting module? ● The following parameters allow users to apply detrending and/or moving average filter in the Segmenting module			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_happe_seg_rej_plotting_on	Toggles showing jointprob visualizations if HAPPE segment rejection is on.	= 0; (off, default) = 1; (on)	
grp_proc_info.beapp_erp_maf_on	Toggles the moving average filter when the ERP events are generated.	= 0; (off, default) = 1; (on)	
grp_proc_info.beapp_erp_maf_order	Order of the moving average filter.	= 30; (default)	
grp_proc_info.beapp_event_group_stim	Toggles group sequences of stimuli (and removes all if one is rejected in sequence).	= 0; (off, default) = 1; (on)	

If you would like to segment only specific trials, how would you like to do so?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.segment_stim_relative_to	Set the label for the trial you will segment relative to.	= {''}; (default) = {'Vbeg'}; (label signaling paradigm start)	Example
grp_proc_info.segment_nth_stim_str		= {''}; (default) = {'stim+'}; (label for the trials you would like to segment)	
grp_proc_info.select_nth_trial	Set to the specific trial(s) you would like to segment.	= []; (to segment all relevant trials, default) = n; (to segment the nth trial) = [1 2 3 4 5]; (refer to the Example here)	

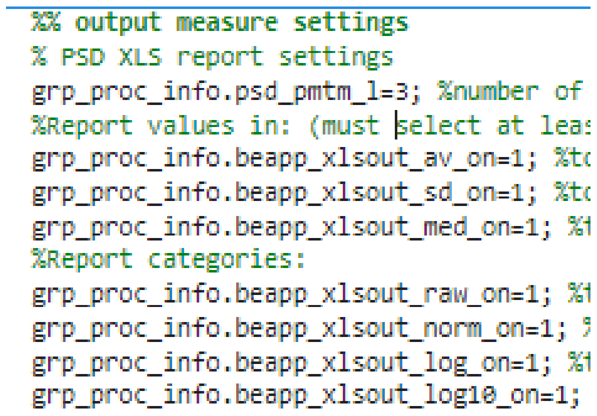
Example:
Back (Select Nth Trial)
If you would like to only segment the first 5 trials following the onset of the paradigm that has the event code of 'stim+', you would set:

- grp_proc_info.segment_stim_relative_to={'Vbeg'};
- grp_proc_info.segment_nth_stim_str={'stim+'};
- grp_proc_info.select_nth_trial=[1 2 3 4 5];

Advanced User Inputs for BEAPP Output Modules (PSD, ITPC, etc.)

PSD Specifications

Back (PSD)
The section for specifying the Advanced PSD settings in the advanced input script looks like this:



```
%% output measure settings
% PSD XLS report settings
grp_proc_info.psd_pmtm_l=3; %number of
%Report values in: (must select at leas
grp_proc_info.beapp_xlsout_av_on=1; %tc
grp_proc_info.beapp_xlsout_sd_on=1; %tc
grp_proc_info.beapp_xlsout_med_on=1; %1
%Report categories:
grp_proc_info.beapp_xlsout_raw_on=1; %1
grp_proc_info.beapp_xlsout_norm_on=1; %
grp_proc_info.beapp_xlsout_log_on=1; %1
grp_proc_info.beapp_xlsout_log10_on=1;
```

In this section, users can choose to specify the following parameters:

If using a multitaper window, how many tapers would you want to use?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.psd_pmtm_l	Number of tapers to use if using the multitaper window.	= 3;	Should be integer ≥ 3.

How would you like to arrange your PSD report(s)? ● The parameters below allow users to specify what .csv PSD reports to generate.			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_xlsout_av_on	Toggle on mean power .csv report(s).	= 1; (on) = 0; (off)	Only generated if beap_xlsout_raw_on OR beapp_xlsout_norm_on is on. If both are on, then two reports are created.
grp_proc_info.beapp_xlsout_sd_on	Toggle on standard deviation .csv report(s).	= 1; (on) = 0; (off)	Only generated if beap_xlsout_raw_on OR beapp_xlsout_norm_on is on. If both are on, then two reports are created.
grp_proc_info.beapp_xlsout_med_on	Toggle on median .csv report(s).	= 1; (on) = 0; (off)	Only generated if beap_xlsout_raw_on OR beapp_xlsout_norm_o

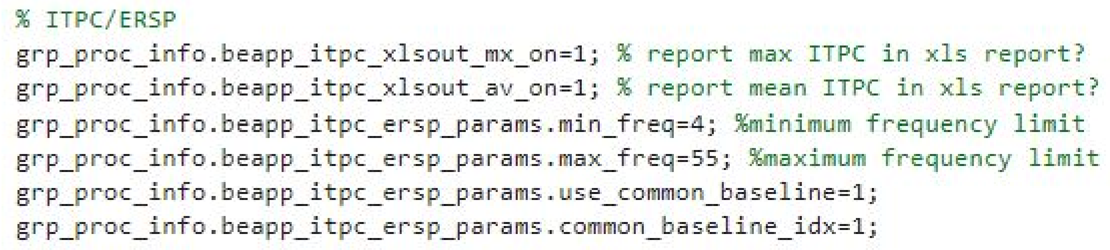
			n is on. If both are on, then two reports are created.
grp_proc_info.beapp_xlsout_raw_on	Toggle on absolute power .csv report.	= 1; (on) = 0; (off)	See notes below on how this is calculated.
grp_proc_info.beapp_xlsout_norm_on	Toggle on relative power .csv report.	= 1; (on) = 0; (off)	See notes below on how this is calculated.
grp_proc_info.beapp_xlsout_log_on	Toggle computing the natural log transformation of the outputs in the PSD report in a separate report that contains "_log_" in the file name.	= 1; (on) = 0; (off)	
grp_proc_info.beapp_xlsout_log10_on	Toggle computing the log10 transformation of the outputs in the PSD report in a separate report that contains "_log10_" in the file name.	= 1; (on) = 0; (off)	

Notes on absolute power and relative power calculation:

- Reports can calculate power as one or both *absolute* and *relative*.
- *Absolute* power will be calculated per frequency ($\frac{total\ power\ at\ each\ frequency\ in\ frequency\ band}{number\ of\ frequencies\ in\ frequency\ band}$)
- *Relative* power will be calculated as: $\frac{the\ total\ power\ in\ frequency\ band}{total\ power\ at\ all\ frequencies}$

ITPC Specifications

The section for specifying the Advanced ITPC settings in the advanced input script looks like this:



```
% ITPC/ERSP
grp_proc_info.beapp_itpc_xlsout_mx_on=1; % report max ITPC in xls report?
grp_proc_info.beapp_itpc_xlsout_av_on=1; % report mean ITPC in xls report?
grp_proc_info.beapp_itpc_ersp_params.min_freq=4; %minimum frequency limit
grp_proc_info.beapp_itpc_ersp_params.max_freq=55; %maximum frequency limit
grp_proc_info.beapp_itpc_ersp_params.use_common_baseline=1;
grp_proc_info.beapp_itpc_ersp_params.common_baseline_idx=1;
```

In this section, users can choose to specify the following parameters:

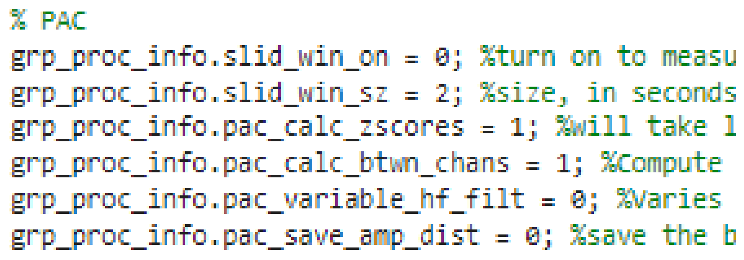
Would you like max and mean ITPC to be reported?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_itpc_xlsout_mx_on	Toggles reporting mean ITPC in .csv exports for the ITPC module.	= 1; (on, default) = 0; (off)	
grp_proc_info.beapp_itpc_xlsout_av_on	Toggles reporting standard deviation of ITPC in .csv exports for the ITPC module.	= 1; (on, default) = 0; (off)	

How would you like to set baseline normalization specifications for the estimation of ERSP?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_itpc_ersp_params.use_common_baseline	Toggles common baseline when comparing two conditions.	= 0; (off, default) = 1; (on)	This parameter will only be accessed if grp_proc_info.beapp_itpc_params.baseline_norm is set to 1.
grp_proc_info.beapp_itpc_ersp_params.common_baseline_idx	If you choose to use a common baseline, enter the index in eeg_w for the condition used to compute the mean baseline spectrum.	= 1; (the first condition listed in eeg_w, default)	This parameter will only be accessed if both grp_proc_info.beapp_itpc_params.baseline_norm and grp_proc_info.beapp_itpc_params.use_common_baseline are set to 1.

PAC Specifications

Back (Z-Score)

The section for specifying the Advanced PAC settings in the advanced input script looks like this:



```
% PAC
grp_proc_info.slid_win_on = 0; %turn on to measu
grp_proc_info.slid_win_sz = 2; %size, in seconds
grp_proc_info.pac_calc_zscores = 1; %will take l
grp_proc_info.pac_calc_btwn_chans = 1; %Compute
grp_proc_info.pac_variable_hf_filt = 0; %Varies
grp_proc_info.pac_save_amp_dist = 0; %save the b
```

In this section, users can choose to specify the following parameters:

What PAC settings would you like to apply to your data, and what values would you like to save?			
Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.slid_win_on	Toggles measuring PAC across time.	= 0; (off, default) = 1; (on)	This feature is currently unavailable in BEAPP.
grp_proc_info.slid_win_sz	Size, in seconds, of the sliding window.	= 1; (default) = 2;	This feature is currently unavailable in BEAPP.
grp_proc_info.pac_calc_zscores	Toggles calculating Modulation Index (MI) on 200 time-shifted surrogate phase-amplitude distributions.	= 0; (off, default) = 1; (on)	If on, PAC will take longer to run. Requires a larger window size (longer segment window). This feature uses a lot of memory in MATLAB so users may have to decrease their resolution values (grp_proc_info.pac_low_fq_res and grp_proc_info.pac_high_fq_res) and/or update MATLAB's memory settings to avoid running out of RAM.
grp_proc_info.pac_calc_btwn_chans	Toggles computing PAC between 2 channels, instead of within each channel.	= 0; (off, default) = 1; (on)	Not recommended (BETA in development).

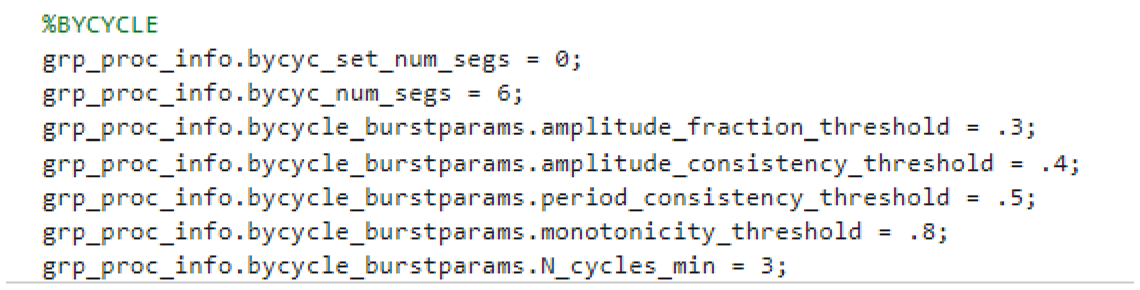
grp_proc_info.pac_variable_hf_filt	Toggles varying the high frequency filter width to prevent overlap with low frequency filter.	= 0; (off, default) = 1; (on)	Not recommended; see Berman et. al, 2015.
grp_proc_info.pac_save_amp_dist	Toggles saving the binned high frequency amplitude distribution variable.	= 0; (off, default) = 1; (on)	

Advanced PAC Module Outputs

Variable Outputs	**Description**	**Units**	**Notes**
grp_proc_info.pac_save_amp_dist	The binned high frequency amplitude distribution.		To generate, set grp_proc_info.pac_save_amp_dist = 1; This variable has 4 dimensions: high frequency x low frequency x phase bins x channels (if z-scores are computed (grp_proc_info.pac_calc_zscores=1), surrogates are added as a fifth dimension).

Bycycle Specifications

The section for specifying the Advanced Bycycle settings in the advanced input script looks like this:



```
%BYCYCLE
grp_proc_info.bycyc_set_num_segs = 0;
grp_proc_info.bycyc_num_segs = 6;
grp_proc_info.bycycle_burstparams.amplitude_fraction_threshold = .3;
grp_proc_info.bycycle_burstparams.amplitude_consistency_threshold = .4;
grp_proc_info.bycycle_burstparams.period_consistency_threshold = .5;
grp_proc_info.bycycle_burstparams.monotonicity_threshold = .8;
grp_proc_info.bycycle_burstparams.N_cycles_min = 3;
```

In this section, users can choose to specify the following parameters:

Do you want to set the number of segments to analyze through bycycle?			
Variables/ Inputs	**Description**	**Examples**	**Notes**

grp_proc_info.bycyc_set_num_segs	Toggles selecting the number of segments to analyze.	= 0; (off, default) = 1; (on)	
grp_proc_info.bycyc_num_segs	Number of trials to analyze.	= 6; (default) = 0;	To use, set grp_proc_info.bycyc_set_num_segs = 1;

Parameters for burst detection – what thresholds does a cycle need to surpass to be considered a burst?			
Variables/ Inputs	**Description**	**Examples**	**Notes**

• For more information, see https://bycycle-tools.github.io/bycycle/auto_tutorials/plot_2_bycycle_algorithm.html#sphx-glr-auto-tutorials-plot-2-bycycle-algorithm-py (some descriptions copied here).			
grp_proc_info.bycycle_burstparams.amplitude_fraction_threshold	Amplitude threshold to be considered a burst.	= 0.3; (default)	
grp_proc_info.bycycle_burstparams.amplitude_consistency_threshold	Amplitude consistency to be considered a burst– consecutive rises and decays should be comparable in magnitude.	= 0.4; (default)	The amplitude consistency of a cycle is equal to the maximum relative difference between rises and decay amplitudes across all pairs of adjacent rises and decays that include one of the flanks in the cycle (3 pairs). For example, if a rise is 10mV and a decay is 7mV, then its amplitude consistency is 0.7.
grp_proc_info.bycycle_burstparams.period_consistency_threshold	Period consistency to be considered a burst– consecutive cycles should be comparable in duration.	= 0.5; (default)	The period consistency is equal to the maximum relative difference between all pairs of adjacent periods that include the cycle of interest (2 pairs: current + previous cycles and current + next cycles). For example, if the previous, current, and next cycles have periods 60ms, 100ms, and 120ms, respectively, then the period consistency is $\min(\frac{60}{100}, \frac{100}{120}) = 0.6$.
grp_proc_info.bycycle_burstparams.monotonicity_threshold	Monotonicity needed to be considered a burst– the rise and	= 0.8; (default)	The monotonicity is the fraction of samples that the

	decay flanks of the cycle should be mostly monotonic.		instantaneous derivative (numpy.diff) is consistent with the direction of the flank. For example, if in the rise, the instantaneous derivative is 90% positive, and in the decay, the instantaneous derivative is 80% negative, then the monotonicity of the cycle would be 0.85 = $(\frac{0.9+0.8}{2})$.
grp_proc_info.bycycle_burstparams.N_cycles_min	Minimum number of cycles needed to be considered a burst.	= 3; (default)	

Running BEAPP Through the GUI

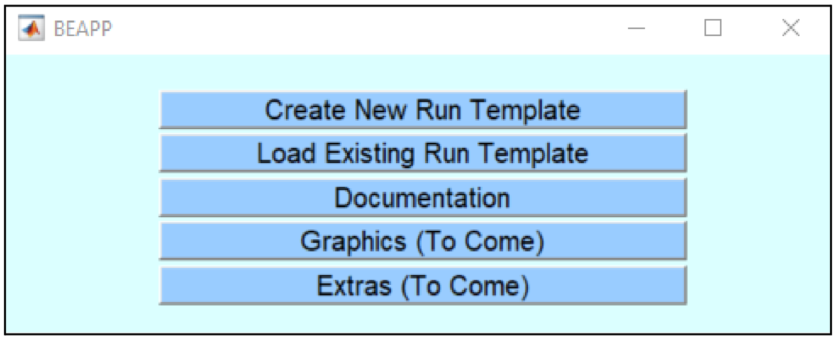
Back (Start-Up)

IMPORTANT: The guide to the GUI section has not been updated in this latest version. GUI functionality receives less regular support compared to the user input script and therefore is not recommended.

If using the GUI, navigate to your BEAPP directory in MATLAB, and on the command line type:

```
beapp_gui
```

The primary BEAPP menu will pop up, and you will be asked to 'Create New Run Template' or 'Load Existing Template'. For your first run, select 'Create New Run Template' to select your settings. On future runs, select 'Load Existing Run Template' to load and modify templates from previous runs. When your template is complete and you have saved it, select 'Run BEAPP' to run.



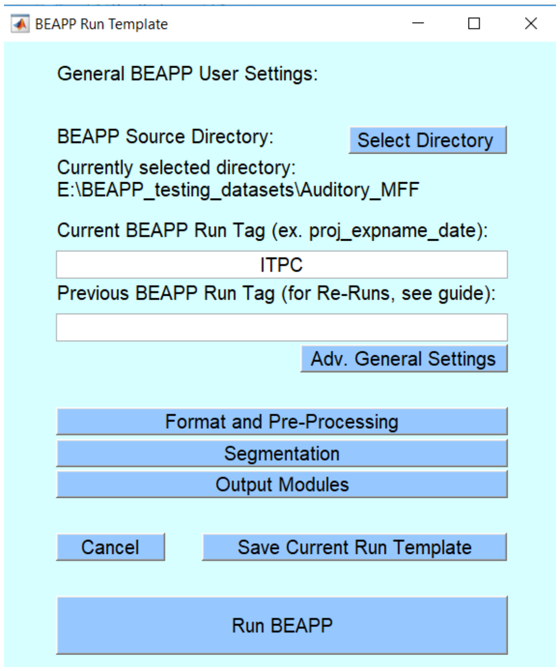
Steps for Creating a Run Template in the GUI

1. Set the general user inputs (required for every data run, found in the General BEAPP User Settings panel)
2. Choose the modules to run for each "step" in BEAPP (Formatting and Pre-Processing, Segmentation, and Output Modules)
3. Choose the details (parameters) of what will occur in the modules you've turned on.

Step 1: General User Inputs Panel

Back (Start-Up)

The general user inputs section is required for every data run. It looks like this:


BEAPP Run Template

General BEAPP User Settings:

BEAPP Source Directory:

Currently selected directory:
E:\BEAPP_testing_datasets\Auditory_MFF

Current BEAPP Run Tag (ex. proj_expname_date):

ITPC

Previous BEAPP Run Tag (for Re-Runs, see guide):

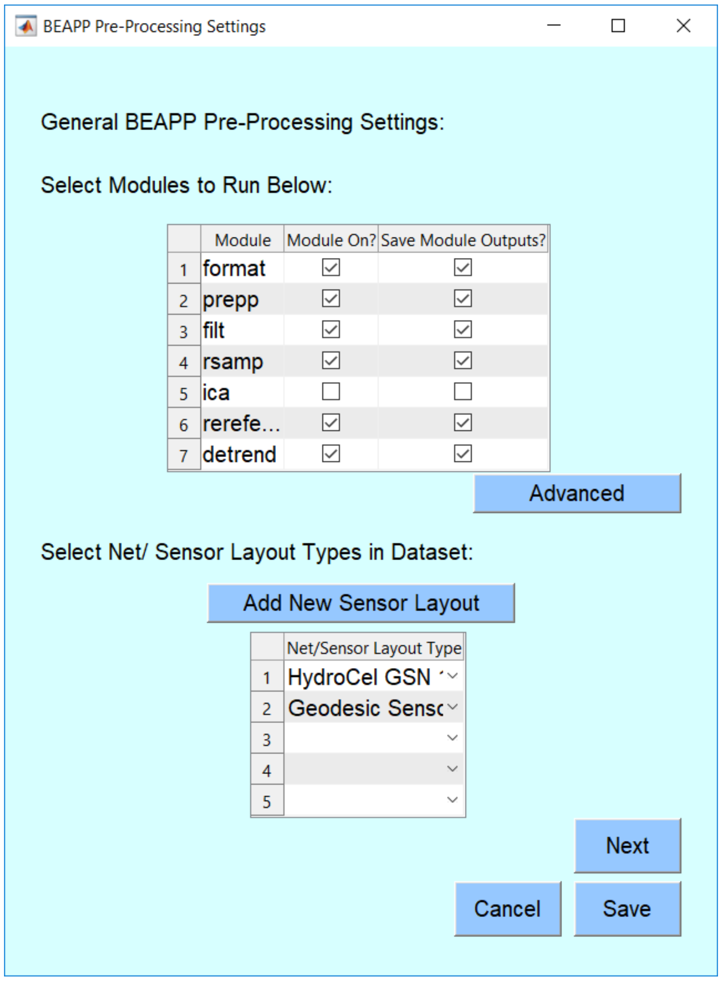
In this section:

- Choose the source directory containing your raw EEG files using the 'Select Directory' button
- Enter the Current Run Tag (see section on Run Tags above). This is a label that is appended to output directories for this run. Using run tags is always recommended.
    - During reruns, if this field is left empty, output directories will have a timestamp appended. Enter 'NONE' if you would like to mute this feature.
- If you are re-running data (e.g., you've already run data through BEAPP but you want to re-run your data with new Filter settings), how was your prior data run labeled ("tagged")?
    - This input is only needed during reruns, to select directories from a previous run that you would like rerun modules to pull data from.

For additional information on re-running data, see section below on Rerunning BEAPP Modules.

Step 2: Format and Pre-Processing Settings (Click 'Format and Pre-Processing')

General Pre-Processing Specifications Panel



BEAPP Pre-Processing Settings

General BEAPP Pre-Processing Settings:

Select Modules to Run Below:

	Module	Module On?	Save Module Outputs?
1	format	✓	✓
2	prepp	✓	✓
3	filt	✓	✓
4	rsamp	✓	✓
5	ica	☐	☐
6	rerefe...	✓	✓
7	detrend	✓	✓

Select Net/ Sensor Layout Types in Dataset:

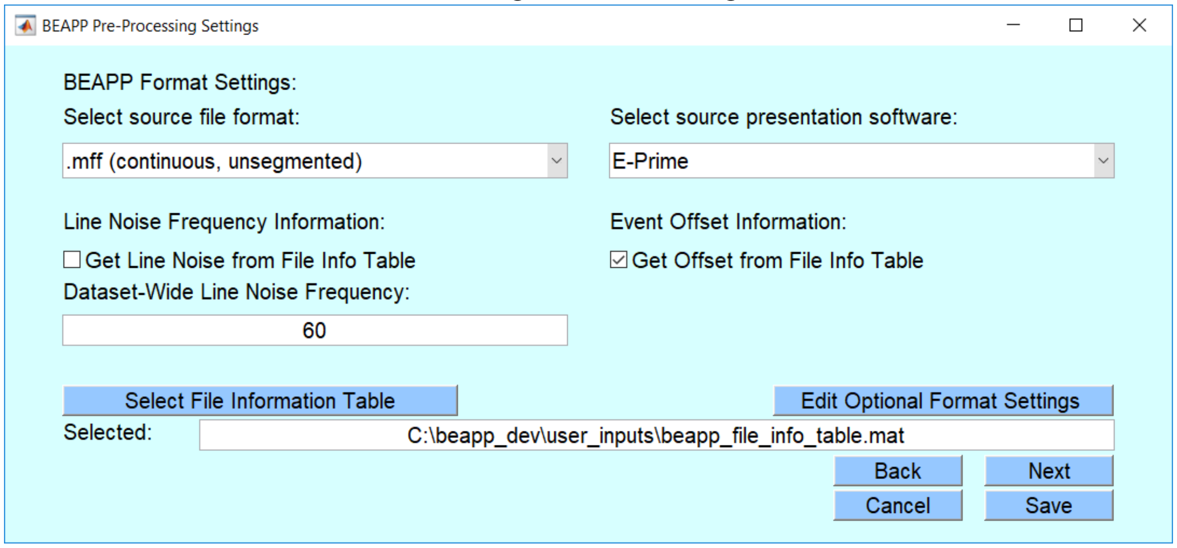
	Net/Sensor Layout Type
1	HydroCel GSN '
2	Geodesic Sensc
3	
4	
5	

- Select which modules to run, and whether to save the outputs from a module using the checkboxes in the module table.
  - You might choose not to save output from a particular module if that module is an intermediate step in your processing pipeline, the step can quickly be re-run in the future if needed, or if you need to save space on your computer.
- Select all possible sensor layout types ("net types") used to collect your data using the dropdown menu. Click 'Add New Sensor Layout' if a layout is not listed (see Adding New Sensor Layouts to the BEAPP Net Library).
- Select Advanced General Settings to:
  - Use a rerun table to run only a subset of files that have been previously run during reruns.
  - Mute warnings generated when BEAPP is writing data into a previously existing folder (during re-use of a "current run tag"). This setting is not recommended for new users, as BEAPP will not alert the user when there is a risk of data being overridden if this is on.
- Click 'Next' when ready to edit parameters for each module selected. You will only be prompted for information relevant to modules turned on in this panel.

Format Specifications Panel

Back (Cond Base Files) Back (Net Library)

The section on Formatting specifications gives BEAPP the information it needs to convert your source EEG data into a format that BEAPP can process. Any new dataset to be run through BEAPP will therefore need to be run through the Formatting module once.


BEAPP Pre-Processing Settings
Select File Information Table
Edit Optional Format Settings
Back
Next
Cancel
Save

BEAPP Format Settings:

Select source file format: .mff (continuous, unsegmented)

Select source presentation software: E-Prime

Line Noise Frequency Information:

☐ Get Line Noise from File Info Table

Dataset-Wide Line Noise Frequency: 60

Event Offset Information:

☑ Get Offset from File Info Table

Selected: C:\beapp_dev\user_inputs\beapp_file_info_table.mat

- Select source file format and presentation software used to collect your source files
- Specify the offset of events tagged in your dataset.
    - *You only need to specify this if you will be segmenting events; you do not need this for continuous (non-event-related) data.* If you are running a dataset in which offset varies by file, you will need to specify offset for each file in a beapp_file_info_table, as described in the start-up guide. If the tagging of your events accounts for any offset, you can set this value to zero.
    - Example: Let's say your EEG has an event tag every time the computer thinks it presents a picture of a face. However, you (as a careful researcher!) have learned that due to a variety of delays in the system wiring, the face in fact appears on the monitor that your experiment's participant can see 18 milliseconds later. You would therefore set your event offset to 18.

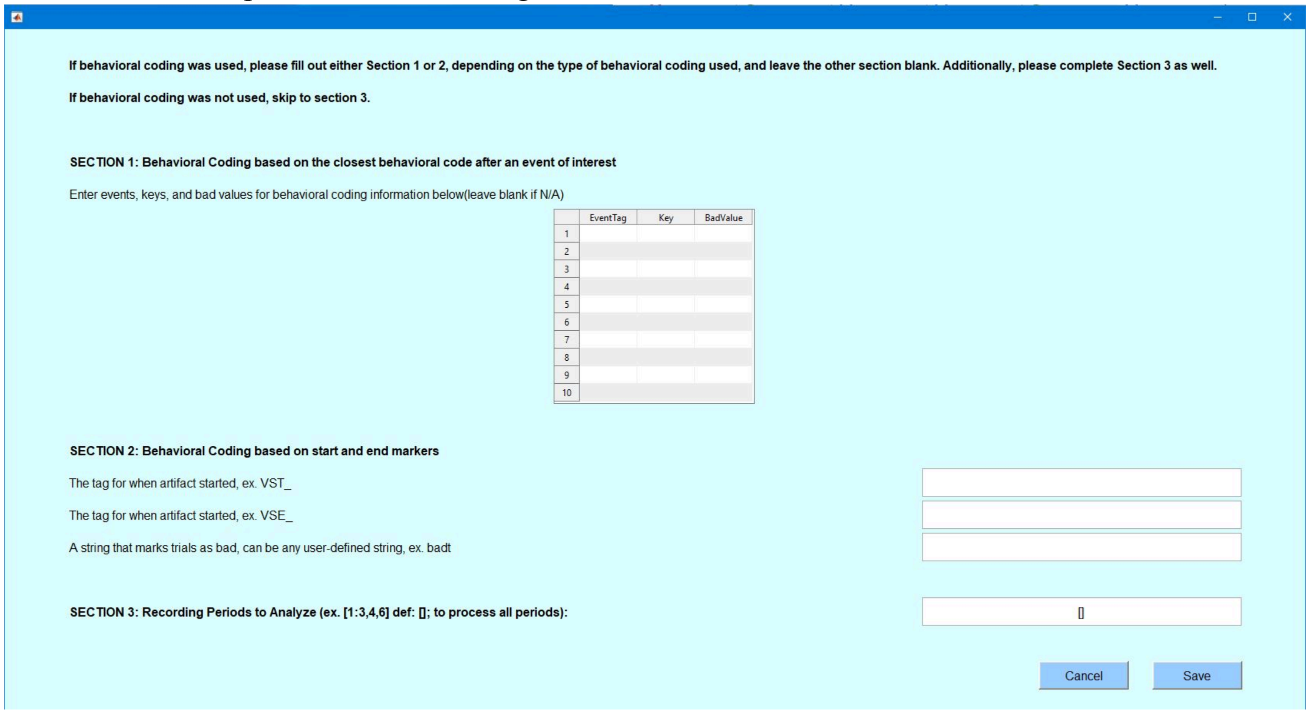
- Specify the frequency of line noise where your data were collected.
    - *You only need to specify this if you will be running a module that removes line noise (e.g., PREP, Notch filter, or HAPPE).* Typically this is 60 Hz in North and South America, and parts of Asia, and 50 Hz in Europe and other parts of the world. Information on line noise in other countries can be found here.
    - If you are running a dataset in which line noise frequency varies by file (e.g., a combined dataset with EEG obtained in the USA and the UK), you will need to specify line noise for each file in a beapp_file_info_table.mat as described in the Start-Up Guide.

- Specify the location of the beapp_file_info_table.mat if source files are a combination of .mats and .sets or if the line noise frequencies or event offsets vary across files for non-mat files. Select appropriate checkbox if offset or line noise frequency vary by file and need to be pulled from a table.

- **IMPORTANT:** If running .mat files, click 'Optional Format Settings' and confirm that the variable name for your EEG data is included in the table that appears:
  - This is the variable(s) that contain(s) the (unsegmented) EEG data, in matrix format, as described in the start-up guide. For example, 'EEG_Segment1' or 'Category_1'.

- If running non-mat files, click 'Optional Format Settings' if you'd like to add behavioral coding information or limit processing to specific recording periods.

  Non-Mat Optional Format Settings:



If behavioral coding was used, please fill out either Section 1 or 2, depending on the type of behavioral coding used, and leave the other section blank. Additionally, please complete Section 3 as well.

If behavioral coding was not used, skip to section 3.

SECTION 1: Behavioral Coding based on the closest behavioral code after an event of interest

Enter events, keys, and bad values for behavioral coding information below(leave blank if N/A)

	EventTag	Key	BadValue
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

SECTION 2: Behavioral Coding based on start and end markers

The tag for when artifact started, ex. VST_

The tag for when artifact started, ex. VSE_

A string that marks trials as bad, can be any user-defined string, ex. badt

SECTION 3: Recording Periods to Analyze (ex. [1:3,4,6] def: []; to process all periods): []

Here, you can specify:

- In Section 1 or 2: How should BEAPP recognize time windows or events to be excluded (e.g., events marked in the file as unusable)?
  - This is called "behavioral coding" because in many cases segments of EEG are marked for exclusion based on a behavior that an observer notices during the EEG acquisition (e.g., inattention to a stimulus, blinking, etc.). *You only need to specify this if your file contains some information about which events or epochs should be excluded. If you do specify this, trials marked "bad" will be excluded during the Segmentation step.*
  - There are two kinds of behavioral coding:
    1. Behavioral coding based on the closest behavioral code after an event of interest (i.e. reject the closest trial next to a user-marked up behavioral code)

        - If this kind of behavioral coding is used, please fill out Section 1:
            - Each row contains a set of identifiers for a kind of behavioral coding information. These identifiers are:
                - EventTag: The name of the event tag in the recording that contains the relevant information (e.g., TRSP).
                - Key: The name of the related key ("sub-tag") that contains the relevant information (e.g., 'badt').
                - BadValue: The value that the Key in the EventTag will take on if a trial has been marked for exclusion (e.g., '1').
            - If multiple pieces of information are contained in the same identifier (e.g., the EventTag has 3 Keys related to behavioral coding, or the Key has 4 possible bad values), list each case separately in its own row in the table as shown above.
    2. Behavioral coding based on start and end markers (i.e. reject trials that elapse between a pair of start-end markers that marks the duration of bad trials.
        - If this kind of behavioral coding is used, please fill out Section 2
- In Section 3: Which data recording periods (epochs) would you like to process?
    - *You only need to specify this if your data included multiple recording periods, and you only want to run a subset of those recording periods.* For example, let's say your continuous EEG included a few minutes of "resting" data, a few minutes of an auditory task, and then a few minutes of a visual task, with each of these tasks in a separate recording period. If you only wanted to run the baseline data, you'd specify that you only wish to run the first recording period.

PREP Specifications Panel

If the user has turned on the PREP module, the section on PREP specifications gives BEAPP the information it needs to run the PREP pipeline (Bigdely-Shamlo et al., 2015). PREP is a very early-stage EEG preprocessing pipeline that offers the following:

1. Removal of line noise
2. Detection and interpolation of bad channels
3. Robust average referencing

The PREP pipeline is standalone software that has been integrated into BEAPP. Most PREP settings are determined by its intrinsic defaults, although the user does have the option to change the line noise frequency as described in the Format module above. Otherwise, the only determination the user needs to make is whether they would like BEAPP to create an Excel output of PREP's findings.

- Therefore, the only selection the user must make for this section is whether you would like Excel output reports after the PREP module (default is on).

Filtering Module Settings Panel

If the user has turned on the Filtering module, the section on filter specifications determines filter settings to be run on the dataset. BEAPP offers three types of filtering and offers CleanLine for line noise removal:

- Low pass filtering (essentially allowing only oscillations slower than a given frequency to be included) using eegfiltnew
- High pass filtering (essentially allowing only oscillations faster than a given frequency to be included) using eegfiltnew
- Notch filtering (for line noise removal)
- CleanLine (line noise removal)

- Select filtering or CleanLine options to apply and enter appropriate cutoffs.
    - Notch and CleanLine cutoffs should be set to NaN, as the line noise frequency can vary by file and is set during Format.
- Click 'Advanced' to set the number of seconds (default = 2) to buffer at the start of and the end of the EEG recording to be excluded after filtering and artifact removal. This exclusion happens during the Segmentation module.

Resampling Specifications Panel

If the user has turned on the Resampling module, the section on Resampling specifications determines how data will be resampled.

- Here, select the resampling frequency
    - Many users will likely choose to resample if they have EEGs in a dataset that are collected at a variety of sampling rates. In many such cases, users will downsample all data to the lowest common sampling rate. For example, if a user has EEG data sampled at 1024 Hz, 100 Hz, 500 Hz, and 250 Hz, they might choose to downsample all data to 100 Hz.
    - A user might also downsample to improve the quality of ICA decomposition. Some ICA paradigms work best at lower sampling rates (e.g., 250 Hz).

ICA/HAPPE Specifications Panel

Users have the option to run ICA, ICA+MARA, or the Harvard Automated Preprocessing Pipeline for EEG (HAPPE), a specific EEG preprocessing pipeline targeted towards artifact removal for infant EEG data.

HAPPE includes the following steps:

1. Identify which channels in the 10-20 electrode system (for EEG recorded from higher-density systems), as well as any additional electrodes, you would like to include in your analysis.
    - This step takes place during the Formatting module, described above. For additional information on selecting the channels to be included in the HAPPE analysis, see the section below.
2. 1 Hz high pass filtering, and 250 Hz low pass filtering

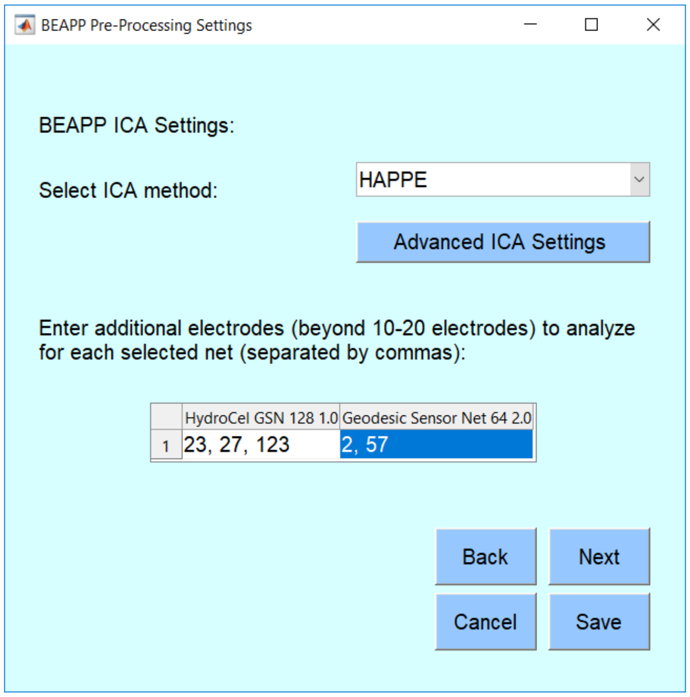
        - If a user wishes to run HAPPE, the filtering module can be turned on if users wish to run an initial band-pass prior to HAPPE, but the cleanline filter and the notch filter should be turned **off** as HAPPE includes a cleanline filter and repeating line noise removal is not recommended.
3. Line noise removal using the *cleanline* function
    - Line noise frequency is set in the Formatting module, described above. However, unless a user runs PREP or a notch filter (neither of which is recommended if a user will be running HAPPE), line noise removal does not take place until the HAPPE module runs.
4. Crude bad channel detection
5. Wavelet cleaning
6. ICA with MARA
7. Interpolation of bad channels
8. Reference to average
9. Removal of bad segments of data

Instructions for running the ICA module in BEAPP are included on this page and the next. Additional information required to run HAPPE for the first time is included here.

The ICA module will automatically use any electrodes listed as equivalent to 10-20 electrodes for a given net in the net library (as MARA uses 10-20 locations for component evaluation). Users have the option to add additional electrodes of interest for each net in the fields provided in the panel.

To run ICA/ICA+MARA/HAPPE module in BEAPP, select the following:

- Which ICA process would you like to run?
    - If you choose to run ICA+ MARA, the other steps of HAPPE described above (i.e., line noise removal, crude bad channel detection, wavelet cleaning, interpolation of bad channels, average reference) are not included.



BEAPP ICA Settings:

Select ICA method: HAPPE

Enter additional electrodes (beyond 10-20 electrodes) to analyze for each selected net (separated by commas):

	HydroCel GSN 128 1.0	Geodesic Sensor Net 64 2.0
1	23, 27, 123	2, 57

- In addition to the 10-20 channels specified when your sensor layouts were added to the library, would you like ICA to include any additional channels in its analyses?
  - For each net, users should create a new entry to include the additional channels for analysis. The additional channels should be entered as the row number (index) in the eeg variable of each corresponding net. Because each net has its own entry, the number of additional channels listed for each net can be different. For now, this is applied to ICA without MARA as well.
  - Choosing how many additional channels to run in the ICA module is somewhat subjective, and users can run the module with visualizations on to check how well the ICA decomposition is working when there are different numbers of channels as input. However, generally, the number of channels that can be added will be limited by the input file's length (time). It is recommended that the number of samples in your EEG recording be equal to $20 * (the\ number\ of\ channels)^2$.
    - For example, an EEG acquired with a 128-channel net and sampling rate of 500 Hz (500 samples/second) would need at least 327,680 samples ($128^2 * 20$ samples), that is, 655.36 seconds of recording (327,680 samples at 500 Hz) to reliably decompose all channels with ICA (see HAPPE for further discussion).

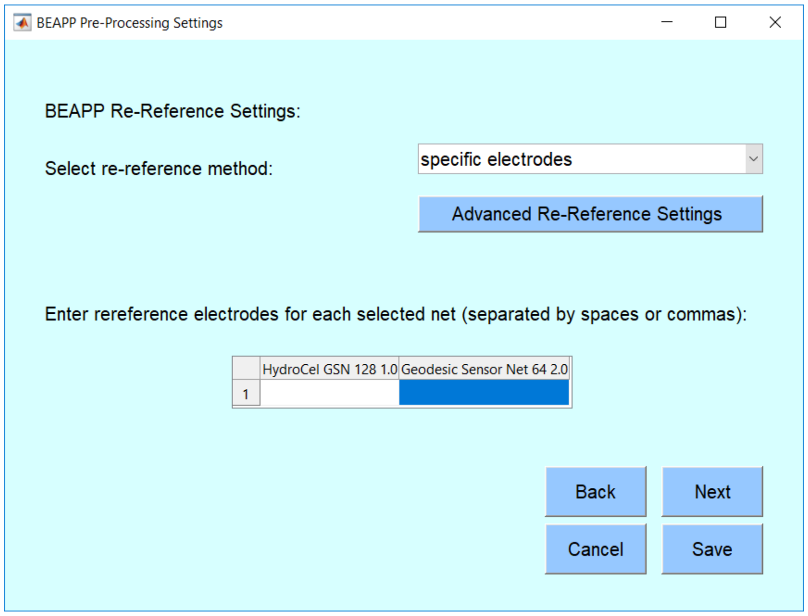
Notes regarding channel inclusion:

- ICA is the only module in BEAPP that automatically restructures data to include only the 10-20 channels (along with any additional channels that the user explicitly instructs BEAPP to include).
- All other modules in BEAPP will include all input channels (i.e., all 64 channels in a 64-channel net, and all 128 channels in a 128-channel net).
- Outside of HAPPE, the only exception to the rule that BEAPP will include all channels is if the user specifies (in advanced general user inputs) that they would like

bad channels removed. In this case, any channels that PREP (or another module) identifies as bad will be replaced with NaN. Otherwise, BEAPP defaults to interpolating channels that PREP (or another module) identifies as bad, but then keeps these interpolated channels for downstream analysis.

Re-Referencing Specifications Panel

If the user has turned on the Re-Referencing module, the section on re-referencing specifications determines the type of re-referencing that will occur.


BEAPP Pre-Processing Settings

BEAPP Re-Reference Settings:

Select re-reference method: specific electrodes

Enter rereference electrodes for each selected net (separated by spaces or commas):

	HydroCel GSN 128 1.0	Geodesic Sensor Net 64 2.0
1		

- Select whether you'd like to apply an average, Laplacian, REST, or a specific electrode re-reference. More information on the different types of re-referencing can be found here.
  - If re-referencing to specific electrodes, enter the rows in eeg corresponding to desired reference channel for each net (the index) for each sensor layout in the table.
- Select 'Advanced Re-Reference Settings' to change the CSD smoothing constant (λ) and the CSD spline flexibility parameter (m).

Note on Re-Referencing: If ICA with HAPPE is used, the Re-Referencing module should be turned off because ICA with HAPPE will automatically use average re-reference.

Detrending Specifications Panel

If the user has turned on the Detrending module, the section on Detrending specifications determines the type of detrending that will occur. It is important to note that Detrending is an optional module. Users are encouraged to visualize data without detrending first and re-process data with detrending as needed. In this section, the user will need to specify only the following information:

- What type of detrending would you like applied to your data?
  - Mean

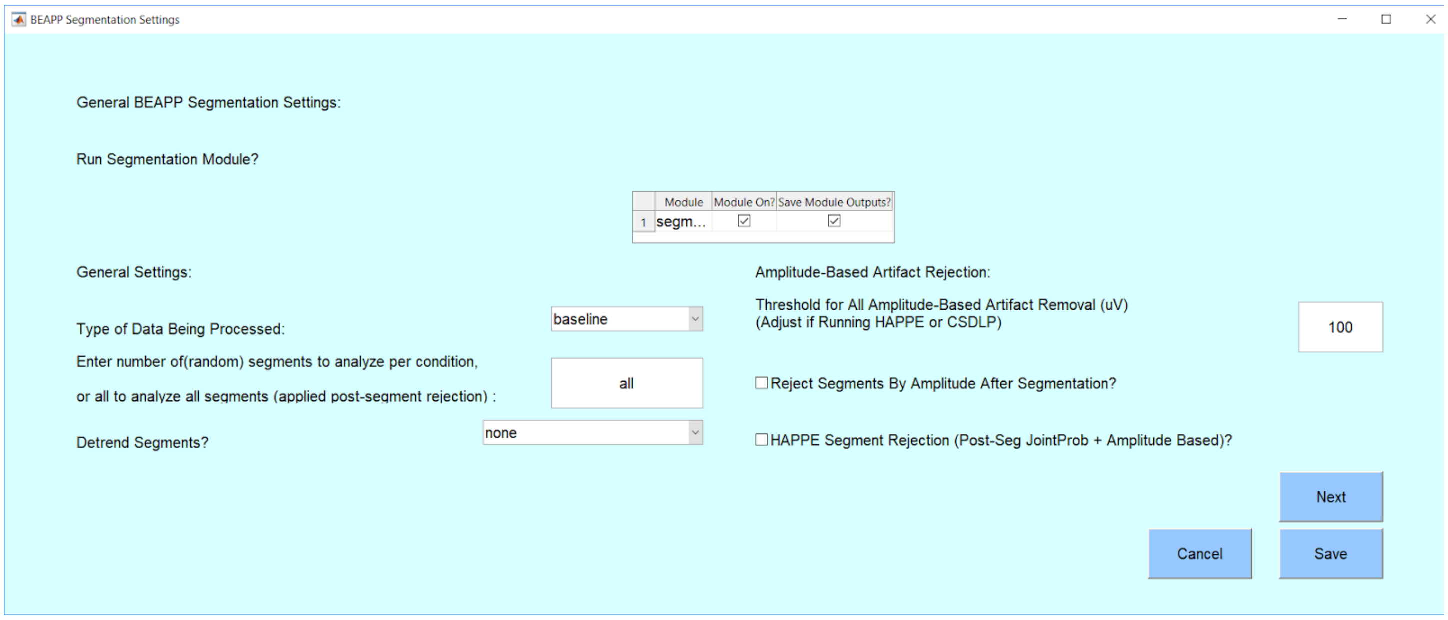
        - Linear
        - Kalman
- If desired, select advanced detrending settings to change:
    - The b value for the Kalman filter (def: .9999).
        - This value will depend on the sampling rate. Use caution; lower b-values will improve fitting (and improve number of usable segments) but may excessively attenuate low and even mid-frequency oscillations.
    - Q init, the smoothing parameter for the Kalman filter.

Note on Kalman filtering: This will likely be used only in specific circumstances (e.g., to attenuate the impact of TMS artifact or certain epileptiform activity). Lower b-values will improve fitting, and therefore decrease the impact of high-amplitude artifact. For users excluding segments using amplitude cutoffs, lower b-values may therefore increase the number of usable segments. However, users should be cautious about the potential impact of low b-values on the power spectrum; low b-values may excessively attenuate low (e.g., delta, theta) and even mid (e.g., alpha, beta) frequency oscillations. The extent of this effect may also depend on the sampling rate. Therefore, if users plan to use a Kalman filter and then run the PSD (power spectral density) module, we encourage them to plot power spectra using several different b-values, to determine potential impact of the chosen value on the power spectrum.

Step 3: Segmentation Settings (Click 'Segmentation' on the General Panel)

If the user has turned on the Segmentation module, the section on Segmentation specifications determines the type of segmentation that will occur.

General Segmentation Specifications Panel



General BEAPP Segmentation Settings:

Run Segmentation Module?

	Module	Module On?	Save Module Outputs?
1	segm...	☑	☑

General Settings:

Type of Data Being Processed: baseline

Enter number of(random) segments to analyze per condition, or all to analyze all segments (applied post-segment rejection) : all

Detrend Segments? none

Amplitude-Based Artifact Rejection:

Threshold for All Amplitude-Based Artifact Removal (uV) (Adjust if Running HAPPE or CSDLP) 100

☐ Reject Segments By Amplitude After Segmentation?

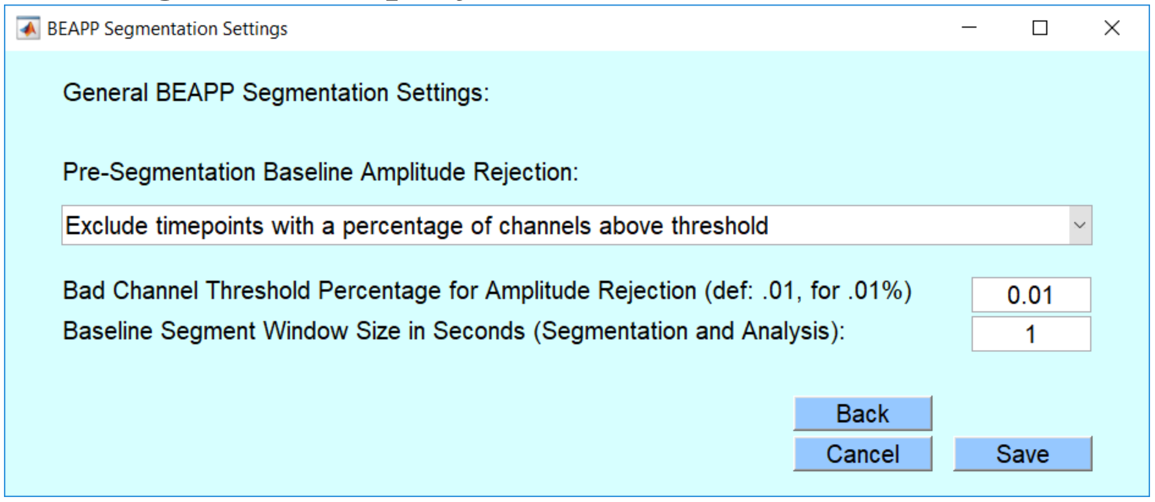
☐ HAPPE Segment Rejection (Post-Seg JointProb + Amplitude Based)?

In this section, the user will need to specify the following information:

- Is the data you would like to segment baseline, event-related, or conditioned baseline?
- Would you like to analyze all segments that survive your selected artifact rejection, or a subset?
    - To analyze all segments, enter all. Otherwise enter the (random) number of segments to sub-select for each condition.
- Would you like to detrend data after Segmentation (within-segment detrending)?
- What is the amplitude threshold for amplitude- based artifact removal?
    - The default value is set to 100. However, users should note that certain types of data (e.g., infant data, or EEG recorded after craniotomy) may require higher amplitude thresholds (e.g., 150). Other types of data (e.g., data that has been run through HAPPE, or data that has undergone a Laplacian transform) may also require changes to amplitude thresholds. For HAPPE, an amplitude threshold of 40 or 50 is typically recommended.
- Will high amplitude artifact be removed after Segmentation?
    - If this is turned on, BEAPP will first create segments as specified below. It will then remove any segments in which amplitude is above the specified threshold in any channel.
- Do you want to use the HAPPE segment rejection code? This variable rejects based on amplitude after Segmentation, and the joint probability of samples (EEGLAB jointprob function).
    - Some users might choose not to use this code if they wish to keep all segments or reject segments using another set of parameters (e.g., amplitude-based rejection criteria from the Segmenting module only).

Additional settings in the Segmentation step are dependent on whether data is baseline, event-tagged, or conditioned baseline.

Baseline Segmentation Specifications Panel


BEAPP Segmentation Settings

General BEAPP Segmentation Settings:

Pre-Segmentation Baseline Amplitude Rejection:

Exclude timepoints with a percentage of channels above threshold

Bad Channel Threshold Percentage for Amplitude Rejection (def: .01, for .01%) 0.01

Baseline Segment Window Size in Seconds (Segmentation and Analysis): 1

If baseline data is being segmented, the user will need to specify the following:

- Will high amplitude artifact be removed prior to Segmentation (in baseline/resting data) using the BEAPP amplitude-based artifact masking? (Additional information on the BEAPP amplitude-based artifact masking can be found here) If yes,
    - BEAPP will create a mask of all unusable data prior to Segmentation. To do so, BEAPP will scan the data in all channels for any data point that is above threshold. Upon identifying these suprathreshold data points, BEAPP will determine the nearest zero-crossing before and after that data point, for that channel. Above-threshold segments are then defined as beginning and ending at the nearest zero-crossings, rather than only including the narrower windows of time where data is suprathreshold. Segments will be created only from the usable data. Users have the option to specify whether timepoints should be marked bad if any channel has a supra-threshold value, or if a specific percentage of channels have values above threshold.
    - Exclude timepoints with any channels above threshold: if an above-threshold data point is marked in any channel, that time point is determined to be unusable.
    - Exclude timepoints with a percentage of channels above threshold: if above-threshold segments are marked in more than the user-set percent of channels (EX: 5 for 5%, .01 for .01%), that time point is determined to be unusable.
- What will be the segment length?
    - This is often set to either 1 or 2 seconds. Additional recommendations for baseline segment length can be found here.

Event-Tagged Segmentation General Specifications Panel

If event-tagged data is being segmented, the user will need to specify the following:

- What is the event code that signifies onset of a stimulus (ex. stm+, stim, pres, etc.)? See Running Event-Tagged Data for more information.
    - List each relevant tag in its own space in the table.
- Where will segments start and end in relation to the event marker of interest (in seconds)?
    - This specifies how the data will be segmented, which may be different from how it will be analyzed. A segment should include all the data to be analyzed, but the analysis may include only a portion of the segment. For example, if a user wishes for the analysis window (e.g. 100 to 800 ms post-stimulus) to be compared to a baseline window (e.g., -200 to -100 ms), then the segment should start at -200 ms (or earlier), and the segment should end at 800 ms (or later). The segment start and end times determine the size of segments saved after this module.

- Will data be baseline corrected? If so, where will the baseline window start and end in relation to the event marker of interest?

- Are segment conditions determined by event tags and cell codes (default) or by the event tags alone (typically for EEGLAB files)?

Event-Tagged Segmentation Condition Specifications Panel

IMPORTANT: We recommend users read the section on Notes for Event Processing in BEAPP prior to running event-related data.

If segment conditions are determined by cell codes, the following will appear:

- Within a given event, what conditions might exist? See section on Condition Options in BEAPP for an explanation of the impetus for requiring this information and examples of usage.
    - First, determine the number and names of conditions you would like to analyze:
        1. Conditions: Variations of an event or stimulus that can occur as part of a task, for example the 'repeating' vs. 'oddball' stimuli in an oddball task.
        2. Condition names do not have to match information provided in the source file.
        3. The example above is an auditory paradigm with “Native”, “Non-Native” and “Standard” conditions.
    - Adjust and name the number of conditions accordingly:
        1. To add conditions: Click “Add Condition” and list new condition names to add in the table that appears. Spaces should not be used in condition names.
        2. To rename conditions: Click “Rename Condition” and update any condition names necessary in the right column.
        3. To delete conditions: Click “Delete condition” and select conditions to delete.

- What are all possible cell codes that correspond to the conditions you've listed?
    - Each column represents a list of possible cell codes that might be associated with a given condition. Most users will only have one cell code value per condition. See Condition Options in BEAPP for more information.
    - The example above has 3 possible sets of conditions that may apply to files in the same dataset (BEAPP will use the provided condition set that has the best fit for the events in the file.).

If segment conditions are marked using event tags alone, the following panel will appear:

- What is the name of the condition that corresponds to each relevant tag?
    - If there are multiple options for tags for a given condition, list the condition name next to each relevant tag and BEAPP will group these appropriately.

Conditioned Baseline Segmentation General Specifications Panel

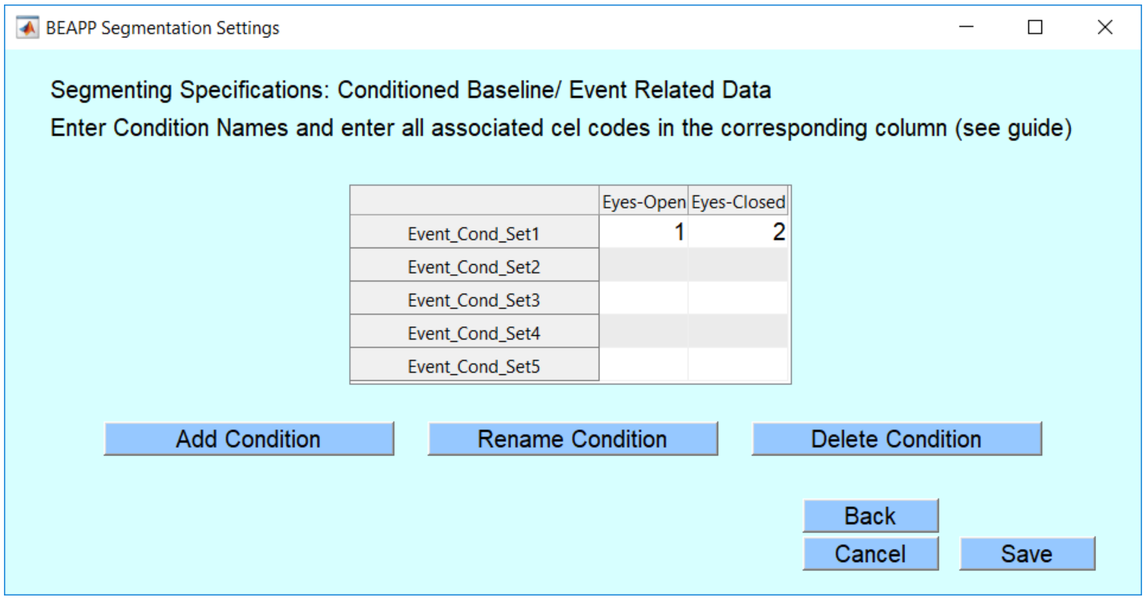
- What are the event codes that signify onset of a conditioned baseline period (EX: stm+, stim, eyeo, etc.). See Running Event-Tagged Data for more information.
    - List each relevant tag in its own space in the table.
- What are the event codes that signify offset of a conditioned baseline period (EX: stm-, eyec etc)

    - Any period between an onset and offset tag pairing (listed in the same row of the table) will be used to extract baseline segments.
- Will high amplitude artifact be removed prior to Segmentation (in baseline/resting data) using the BEAPP amplitude-based artifact masking? Additional information on the BEAPP amplitude-based artifact masking can be found here. If yes,
    - Exclude timepoints with any channels above threshold: if an above-threshold data point is marked in any channel, that time point is determined to be unusable.
    - Exclude timepoints with a percentage of channels above threshold: if above-threshold time points are marked in more than the user-set percent of channels (ex: 5 for 5%, .01 for .01%), that time point is determined to be unusable.
- What will be the segment length?
    - This is often set to either 1 or 2 seconds. Additional recommendations for baseline segment length can be found here.

Conditioned Baseline Condition Selection Specifications Panel

We recommend users read the section on Notes for Event Processing in BEAPP prior to running event-related data.

If segment conditions are determined by cell codes, the following will appear:



Segmenting Specifications: Conditioned Baseline/ Event Related Data

Enter Condition Names and enter all associated cel codes in the corresponding column (see guide)

	Eyes-Open	Eyes-Closed
Event_Cond_Set1	1	2
Event_Cond_Set2		
Event_Cond_Set3		
Event_Cond_Set4		
Event_Cond_Set5		

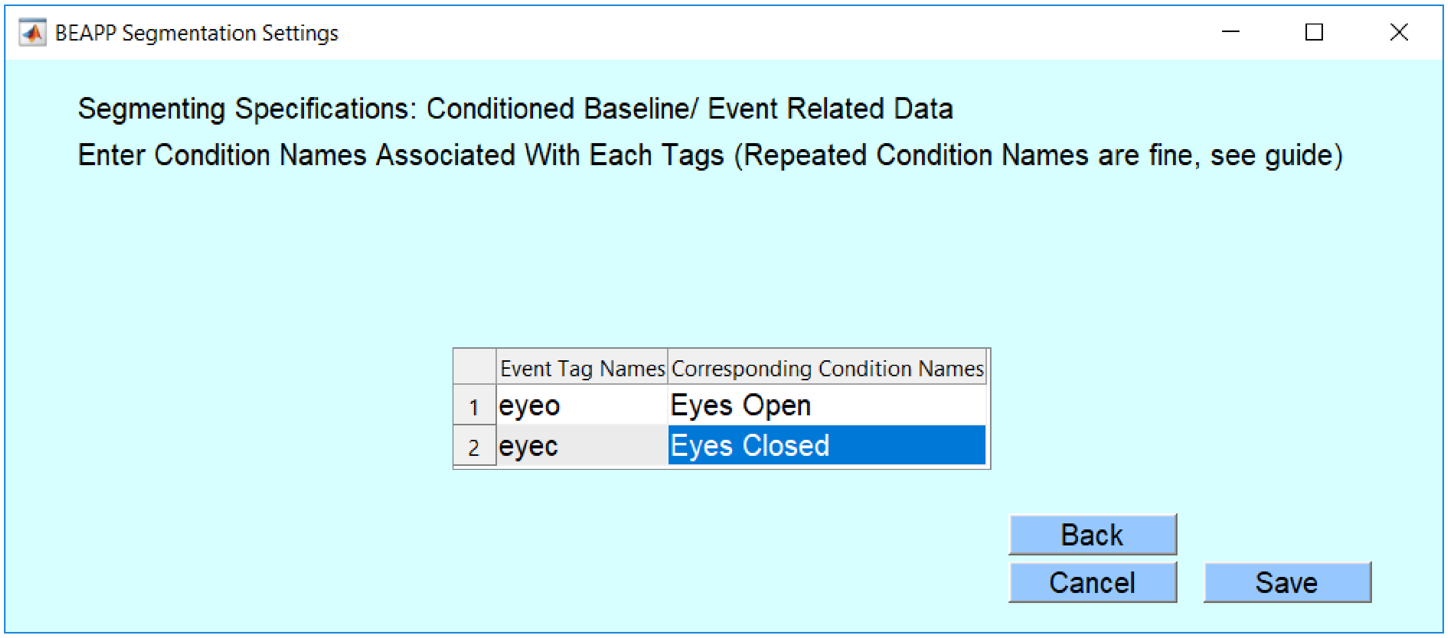
As with event-tagged data, conditioned baseline segments will need to be extracted using event tags that may have different conditions (eyes open and eyes closed, for example). The user will need to specify:

- Within a given conditioned baseline period, what conditions might exist? See section on Condition Options in BEAPP for an explanation of the impetus for requiring this information and examples of usage.
    - First, determine the number and names of conditions you would like to analyze:

        1. Conditions (for baseline): Variations of an event or stimulus that can occur as part of a paradigm, for example, eyes open and eyes closed data.
        2. Condition names do not have to match information provided in the source file.
        3. The example above is a baseline paradigm with Eyes-Open and Eyes-Closed Conditions.
    - Adjust and name the number of conditions accordingly:
        1. To Add Conditions: Click “Add Condition” and list new condition names to add in the table that appears. Spaces should not be used in condition names.
        2. To rename conditions: Click “Rename Condition” and update any condition names necessary in the right column.
        3. Delete conditions: Click “Delete condition” and select conditions to delete.

- What are all possible cell codes that correspond to the conditions you've listed?
    - Each column represents a list of possible cell codes that might be associated with a given condition. Most users will only have one cell code value per condition. See Condition Options in BEAPP for more information.
    - The example above has 3 possible sets of conditions that may apply to files in the same dataset (BEAPP will use the provided condition set that has the best fit for the events in the file).

If segment conditions are marked using event tags alone, the following panel will appear:


BEAPP Segmentation Settings

Segmenting Specifications: Conditioned Baseline/ Event Related Data

Enter Condition Names Associated With Each Tags (Repeated Condition Names are fine, see guide)

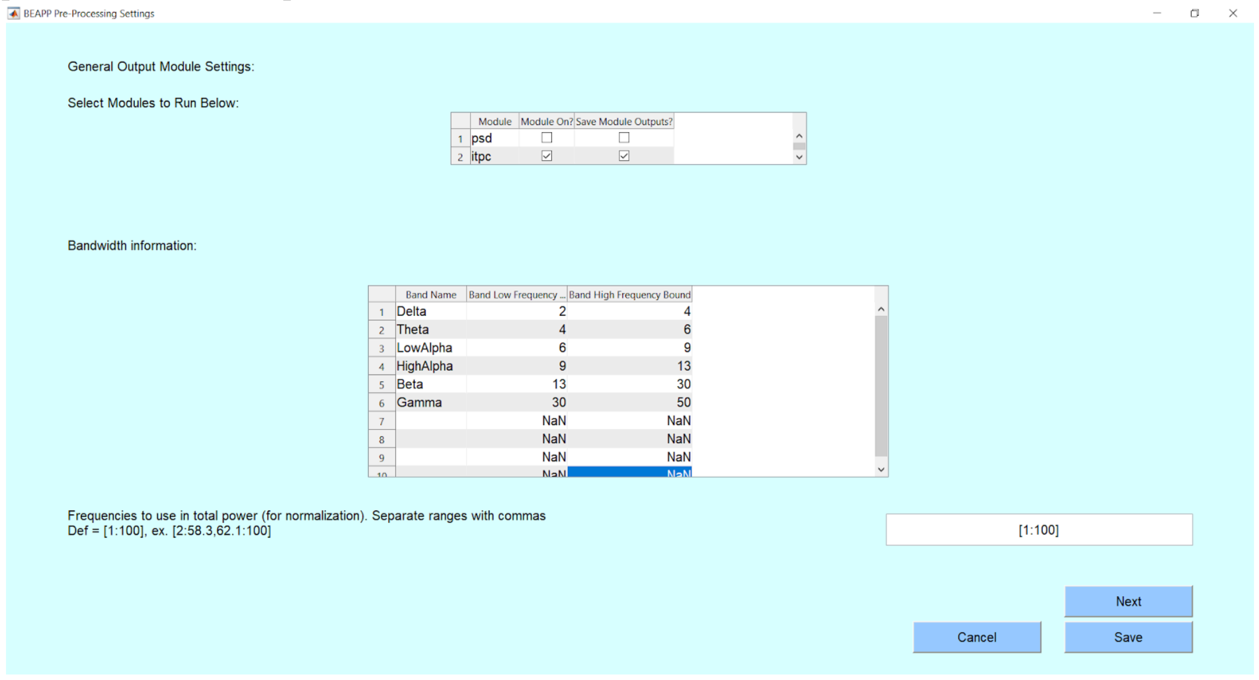
	Event Tag Names	Corresponding Condition Names
1	eyeo	Eyes Open
2	eyec	Eyes Closed

- What is the name of the condition that corresponds to each relevant tag?
    - If there are multiple options for tags for a given condition, list the condition name next to each relevant tag and BEAPP will group these appropriately.

Output Measure Settings (Click 'Output Modules' on the General Panel)

Output Module General Specifications Panel

If the user has turned on any output modules (e.g., PSD or ITPC), the section on output module specifications determines the parameters of these output measures. The section specifying parameters for the output measures looks like this:



General Output Module Settings:

Select Modules to Run Below:

	Module	Module On?	Save Module Outputs?
1	psd	☐	☐
2	itpc	✓	✓

Bandwidth information:

	Band Name	Band Low Frequency ...	Band High Frequency Bound
1	Delta	2	4
2	Theta	4	6
3	LowAlpha	6	9
4	HighAlpha	9	13
5	Beta	13	30
6	Gamma	30	50
7		NaN	NaN
8		NaN	NaN
9		NaN	NaN
10		NaN	NaN

Frequencies to use in total power (for normalization). Separate ranges with commas
Def = [1:100], ex. [2:58.3,62.1:100]

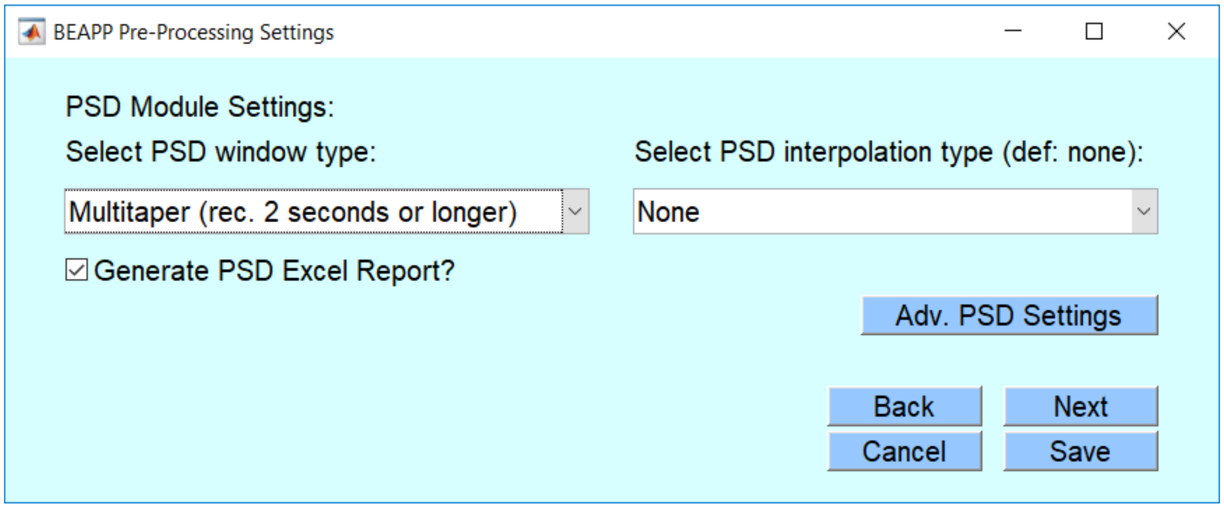
[1:100]

In this section, the user will need to specify the following information:

- Which of the existing output modules would you like to run?
- What are the bandwidths of interest? *This is only necessary if the user wants csv reports for the PSD or ITPC modules (or other output modules in the future).* The user can specify as many (or as few) frequency bands of interest as they wish.
- The user is also asked to specify which frequencies they would like to include in measures of total power (used to normalize power in reports).
- If data is set as event-related in the Segmentation module, users will also be prompted to specify when the analysis window starts and ends in relation to the event marker of interest.

PSD Module Specifications Panel

If the user wishes to obtain data on the POWER SPECTRUM, the following specifications are necessary:



PSD Module Settings:

Select PSD window type: Multitaper (rec. 2 seconds or longer)

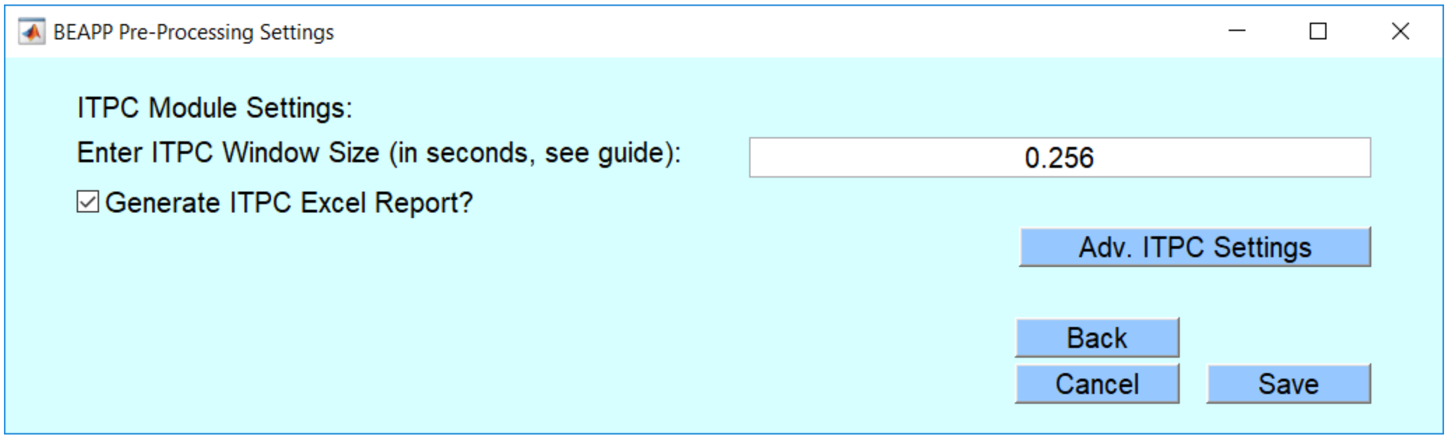
Select PSD interpolation type (def: none): None

☑ Generate PSD Excel Report?

- What type of windowing should be applied to calculate the power spectrum?
  - Most signal processors recommend a multitaper, if possible. This smooths the power spectrum and mitigates edge effects. However, as described in the section above on choosing segment length, some users may find that their segment lengths are too short (e.g. < 2 seconds), and frequency resolution needs too high, to allow for a multitaper to be feasible.
    - When using a Hanning window, overlapping windows are not currently used, as is often recommended (we hope to add this soon!).
- Do you want BEAPP to interpolate the power spectrum?
  - Most users will not choose any interpolation. Note, interpolation does not change the frequency resolution of the PSD; it simply interpolates between values output by the PSD. In some cases, however, users may wish to interpolate the frequency axis if they are concerned about the effects of "edge" values in binned power calculations. For example, if power values are output at 2, 4, and 6 Hz in the example above, interpolation might help a user better differentiate between power in the delta vs. theta vs. low alpha bands.
- Do you want BEAPP to save out a report table with PSD output summary information? This table would include power values in each frequency band, in each channel, for each file.
- To add or change the types of values given in the report tables (e.g., median, mean, standard deviation, log, etc.), click Advanced PSD Settings and select the appropriate options. See the advanced user inputs section of this guide for details on these outputs.

ITPC Module Specifications Panel

If the user wishes to obtain data on INTER-TRIAL PHASE COHERENCE (ITPC), the following specifications are necessary:


BEAPP Pre-Processing Settings

ITPC Module Settings:

Enter ITPC Window Size (in seconds, see guide): 0.256

☑ Generate ITPC Excel Report?

- What size sub-window should be applied to calculate ITPC?
  - When calculating ITPC, the analysis window is typically divided into multiple overlapping sub-windows, each of the size specified here. Thus ITPC is not output as a single value, but instead as a vector of values specifying the ITPC for the midpoint of each of these sub-windows. This allows the user to assess how ITPC changes over a window of time in relation to a repeated event.

- Do you want BEAPP to save out a report table with ITPC output summary information?

FOOOF Module Specifications Panel

If the user wishes to obtain data from the FOOOF module, the following specifications are necessary:

- Over what frequency range do you want FOOOF to run? Specify the minimum and maximum frequencies.
- Parameters used to prevent overfitting:
  - Peak width limit (min, max): What is the minimum and maximum width for which a peak should be detectable?
  - Max number of peaks: What is the maximum number of peaks FOOOF should find? A maximum must be set. Anything > 6 will rarely have an effect.
  - Min peak amplitude: What is the minimum amplitude for which a peak should be detectable?
  - Min peak threshold: For this to have an effect, set to be > 0. Should be set if the data may not have any peaks.
  - Background mode: What method do you want FOOOF to use to find the background of the power spectra? Recommended to use 'knee' when the frequency range > 40.
    - 'knee' will result in a third parameter for the background (called 'knee').
- Do you want BEAPP to save an excel report with the FOOOF results?
- Do you want to save all FOOOF-generated figures? Do not check if you either do not want any figures or you want figures for only specific channel #'s, files, or channel groups.

- Do you want channels to be averaged together for FOOOF analysis, or should channels be run separately?
- Specify which channels should be run. If all channels should be run, leave blank.
- If you want to average channels in groups, specify which channels should be averaged together.
    - Channels to average together should be separated by a comma only, and groups should be separated by a space only.
- Specify which participants / channels / groups to save reports for.
    - Leave blank to not specify, and therefore save all.
    - Separate participants / channels / groups with a space (not a comma).
      To specify groups, do so by group number. Group number is determined by the order the groups were input.

PAC Module Specifications Panel

- Which method would you like to use to generate the correlations between each frequency?
    - 'ozkurt', 'canolty', 'tort', 'penny' = methods based on filtering and using a Hilbert transform.
    - 'vanwijk' = method for a joint AAC and PAC estimation based on filtering and using the Hilbert transform.
    - 'sigl', 'nagashima', 'hagihira', 'bispectrum' = methods for a PAC estimation based on bicoherence.
    - 'colgin', 'jiang' = PAC / PAC directionality estimation (respectively) based on filtering and computing coherence.
    - 'deprelatour' = PAC estimation based on a driven autoregressive model
    - More information on methods can be found here: https://github.com/pactools/pactools.
- What frequency range should PAC run on, for the lower range of frequencies?
- What resolution should be used for the lower frequency range?
    - PAC will run on a subdivision of frequencies starting on the minimum and ending at the maximum; resolution will determine the number of frequencies run in the frequency range.
- What bandwidth should be used for the bandpass filter on the lower frequency?
- ^You will repeat the parameters corresponding to the previous 3 bullets for the higher frequency.
- Do you want BEAPP to save an excel report with the PAC results?
- Do you want to save all PAC-generated figures? Do not check if you either do not want any figures or you want figures for only specific channel #'s or files.
- Specify which channels should be run. If all channels should be run, leave blank.
- Specify which participants / channels to save reports for.
    - Leave blank to not specify, and therefore save all.
    - Separate participants / channels with a space (not a comma).

Supplemental Guides

Rerunning BEAPP Modules

Back (Current Run Tags) Back (Prev Run Tag) Back (Adv Gen) Back (GUI)

If you have run BEAPP on a dataset and would like to rerun some section of the pipeline with new parameters or files, without rerunning all the modules upstream, you can run BEAPP as normal with only the desired modules selected if you fill in the prev_run_tag field. For example, you may want to rerun the Segmentation module with different window sizes and have already run Format, Filter, Resample, and ICA, so you only need to turn on Segmentation and not the others.

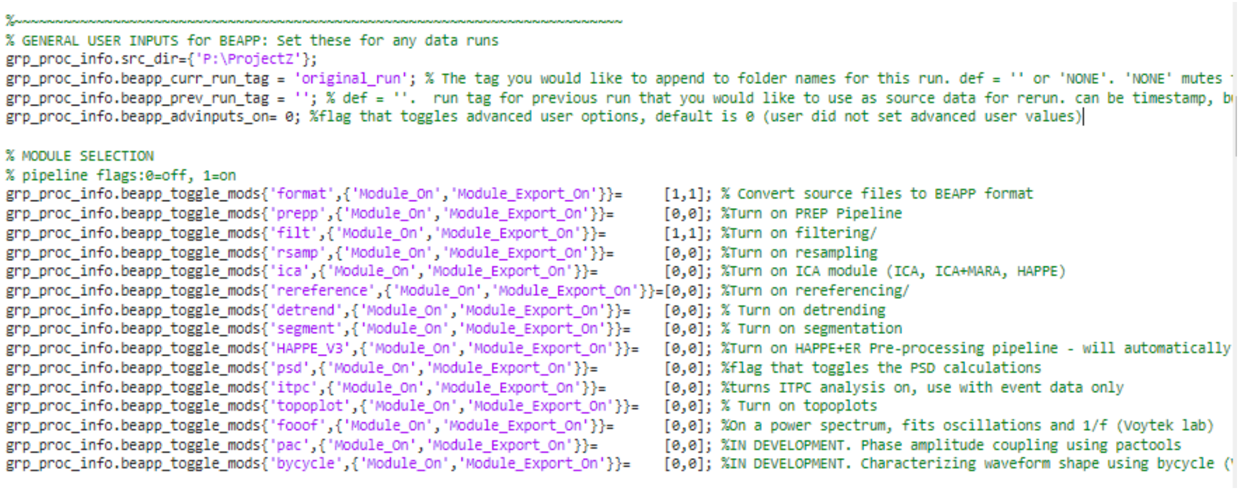
BEAPP automatically appends new directories created during reruns with a timestamp. You can overwrite the automatic timestamp tag by using the current run tag input, or mute the appended tag by setting the current run tag to 'NONE'. If you'd like to rerun modules from directories that have a tag appended to them (directories from runs where the current run tag was not empty, or where timestamps were appended), you can specify this as the previous run tag.

During reruns, it's recommended that you specify the list of net types present in your dataset (exact names as they appear in your net library) in grp_proc_info.src_unique_nets. While specifying these values is not required for BEAPP to run, it will make your rerun much faster, especially for large datasets. It's fine to include this information during initial runs, but it will not impact run speed.

Rerun Example

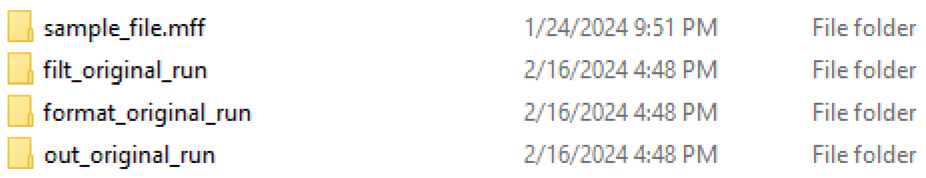
Back (Gen Inputs)

In this example, the original run turns on two modules: Formatting and Filtering. This means that the source files are first going to be Formatted then Filtered. In the rerun, only Filtering is turned on. This means that we are asking BEAPP to run Filtering on the Formatted file from the original run. This would be useful when we explore different Filtering settings.
Below is how the original run user inputs looks:



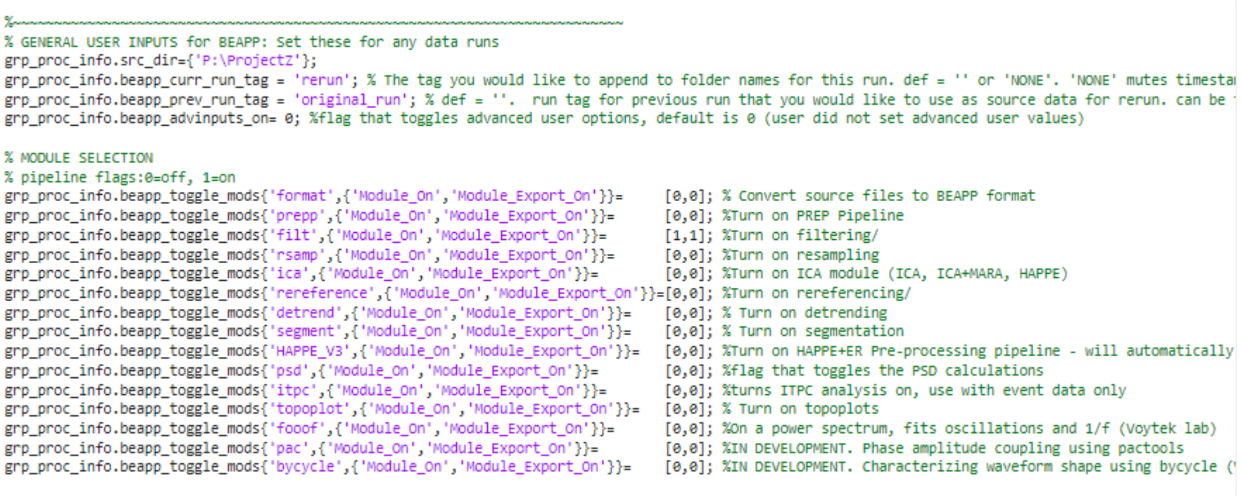
```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% GENERAL USER INPUTS for BEAPP: Set these for any data runs
grp_proc_info.src_dir={'P:\ProjectZ'};
grp_proc_info.beapp_curr_run_tag = 'original_run'; % The tag you would like to append to folder names for this run. def = '' or 'NONE'. 'NONE' mutes
grp_proc_info.beapp_prev_run_tag = ''; % def = ''.  run tag for previous run that you would like to use as source data for rerun. can be timestamp, b
grp_proc_info.beapp_advinputs_on= 0; %flag that toggles advanced user options, default is 0 (user did not set advanced user values)

% MODULE SELECTION
% pipeline flags:0=off, 1=on
grp_proc_info.beapp_toggle_mods{'format',{'Module_On','Module_Export_On'}}=     [1,1]; % Convert source files to BEAPP format
grp_proc_info.beapp_toggle_mods{'prepp',{'Module_On','Module_Export_On'}}=      [0,0]; %Turn on PREP Pipeline
grp_proc_info.beapp_toggle_mods{'filt',{'Module_On','Module_Export_On'}}=       [1,1]; %Turn on filtering/
grp_proc_info.beapp_toggle_mods{'rsamp',{'Module_On','Module_Export_On'}}=      [0,0]; %Turn on resampling
grp_proc_info.beapp_toggle_mods{'ica',{'Module_On','Module_Export_On'}}=        [0,0]; %Turn on ICA module (ICA, ICA+MARA, HAPPE)
grp_proc_info.beapp_toggle_mods{'rereference',{'Module_On','Module_Export_On'}}=[0,0]; %Turn on rereferencing/
grp_proc_info.beapp_toggle_mods{'detrend',{'Module_On','Module_Export_On'}}=    [0,0]; % Turn on detrending
grp_proc_info.beapp_toggle_mods{'segment',{'Module_On','Module_Export_On'}}=    [0,0]; % Turn on segmentation
grp_proc_info.beapp_toggle_mods{'HAPPE_V3',{'Module_On','Module_Export_On'}}=   [0,0]; %Turn on HAPPE+ER Pre-processing pipeline - will automatically
grp_proc_info.beapp_toggle_mods{'psd',{'Module_On','Module_Export_On'}}=        [0,0]; %flag that toggles the PSD calculations
grp_proc_info.beapp_toggle_mods{'itpc',{'Module_On','Module_Export_On'}}=       [0,0]; %turns ITPC analysis on, use with event data only
grp_proc_info.beapp_toggle_mods{'topoplot',{'Module_On','Module_Export_On'}}=   [0,0]; % Turn on topoplots
grp_proc_info.beapp_toggle_mods{'fooof',{'Module_On','Module_Export_On'}}=      [0,0]; %On a power spectrum, fits oscillations and 1/f (Voytek lab)
grp_proc_info.beapp_toggle_mods{'pac',{'Module_On','Module_Export_On'}}=        [0,0]; %IN DEVELOPMENT. Phase amplitude coupling using pactools
grp_proc_info.beapp_toggle_mods{'bycycle',{'Module_On','Module_Export_On'}}=    [0,0]; %IN DEVELOPMENT. Characterizing waveform shape using bycycle (
```

Below is how the original run output will look:



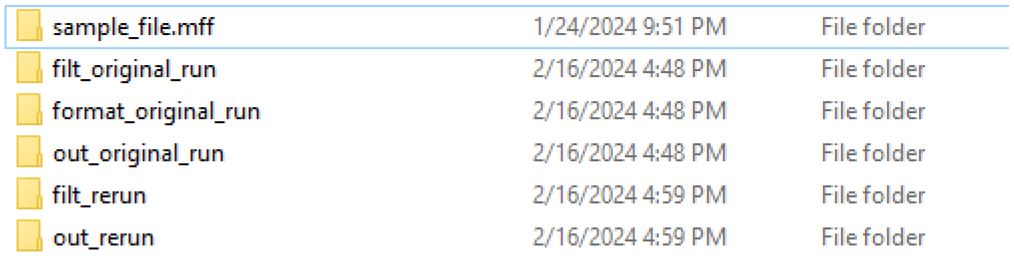
```
sample_file.mff          1/24/2024 9:51 PM     File folder
filt_original_run        2/16/2024 4:48 PM     File folder
format_original_run      2/16/2024 4:48 PM     File folder
out_original_run         2/16/2024 4:48 PM     File folder
```

Below is how the re-run user inputs looks:



```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% GENERAL USER INPUTS for BEAPP: Set these for any data runs
grp_proc_info.src_dir={'P:\ProjectZ'};
grp_proc_info.beapp_curr_run_tag = 'rerun'; % The tag you would like to append to folder names for this run. def = '' or 'NONE'. 'NONE' mutes timesta
grp_proc_info.beapp_prev_run_tag = 'original_run'; % def = ''.  run tag for previous run that you would like to use as source data for rerun. can be
grp_proc_info.beapp_advinputs_on= 0; %flag that toggles advanced user options, default is 0 (user did not set advanced user values)

% MODULE SELECTION
% pipeline flags:0=off, 1=on
grp_proc_info.beapp_toggle_mods{'format',{'Module_On','Module_Export_On'}}=     [0,0]; % Convert source files to BEAPP format
grp_proc_info.beapp_toggle_mods{'prepp',{'Module_On','Module_Export_On'}}=      [0,0]; %Turn on PREP Pipeline
grp_proc_info.beapp_toggle_mods{'filt',{'Module_On','Module_Export_On'}}=       [1,1]; %Turn on filtering/
grp_proc_info.beapp_toggle_mods{'rsamp',{'Module_On','Module_Export_On'}}=      [0,0]; %Turn on resampling
grp_proc_info.beapp_toggle_mods{'ica',{'Module_On','Module_Export_On'}}=        [0,0]; %Turn on ICA module (ICA, ICA+MARA, HAPPE)
grp_proc_info.beapp_toggle_mods{'rereference',{'Module_On','Module_Export_On'}}=[0,0]; %Turn on rereferencing/
grp_proc_info.beapp_toggle_mods{'detrend',{'Module_On','Module_Export_On'}}=    [0,0]; % Turn on detrending
grp_proc_info.beapp_toggle_mods{'segment',{'Module_On','Module_Export_On'}}=    [0,0]; % Turn on segmentation
grp_proc_info.beapp_toggle_mods{'HAPPE_V3',{'Module_On','Module_Export_On'}}=   [0,0]; %Turn on HAPPE+ER Pre-processing pipeline - will automatically
grp_proc_info.beapp_toggle_mods{'psd',{'Module_On','Module_Export_On'}}=        [0,0]; %flag that toggles the PSD calculations
grp_proc_info.beapp_toggle_mods{'itpc',{'Module_On','Module_Export_On'}}=       [0,0]; %turns ITPC analysis on, use with event data only
grp_proc_info.beapp_toggle_mods{'topoplot',{'Module_On','Module_Export_On'}}=   [0,0]; % Turn on topoplots
grp_proc_info.beapp_toggle_mods{'fooof',{'Module_On','Module_Export_On'}}=      [0,0]; %On a power spectrum, fits oscillations and 1/f (Voytek lab)
grp_proc_info.beapp_toggle_mods{'pac',{'Module_On','Module_Export_On'}}=        [0,0]; %IN DEVELOPMENT. Phase amplitude coupling using pactools
grp_proc_info.beapp_toggle_mods{'bycycle',{'Module_On','Module_Export_On'}}=    [0,0]; %IN DEVELOPMENT. Characterizing waveform shape using bycycle (
```

Below is how the re-run output will look:



sample_file.mff	1/24/2024 9:51 PM	File folder
filt_original_run	2/16/2024 4:48 PM	File folder
format_original_run	2/16/2024 4:48 PM	File folder
out_original_run	2/16/2024 4:48 PM	File folder
filt_rerun	2/16/2024 4:59 PM	File folder
out_rerun	2/16/2024 4:59 PM	File folder

Rerun Subset of Files

Finally, if you would only like to rerun the selected modules for a subset of the files in the most recent source directory, you may choose to set the advanced input grp_proc_info.beapp_use_rerun_table = 1. By default, this will use the files listed in the rerun_fselect_table.mat in user_inputs (and optionally, net types, sampling rates, and linenoise frequencies). If you would like to use your own rerun_fselect_table.mat you can set the location for it in beapp_set_input_file_locations. For information on generating this table, see Generating a BEAPP File Info Table.

Saving/Reusing User Input Templates or File Information Tables

Saving Templates

Some users may choose to save their user settings for a dataset to allow them to switch between setting configurations quickly without having to re-enter information. Users wishing to do this should enter their settings in beapp_userinputs.m and beapp_advinputs.m as usual, and then save each file with a new name using 'Save As' in MATLAB (found in the 'Save' menu in the Editor panel).

Saving File Information Tables

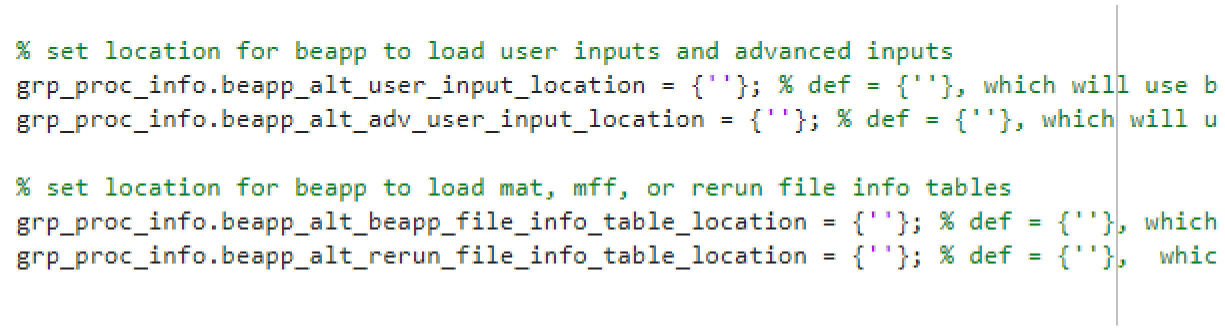
Some users may choose to save the beapp_file_info_tables used for different datasets as well. Users wishing to do this should create the tables as normal, and rename the .mat file with the desired table name (as described in Generating a BEAPP_File_Info_Table). **Make sure not to change the MATLAB variable name.**

Using Templates and Saved Tables

Back (Rerun) Back (User Script Guide)

If you'd like to change between multiple user input or advanced input templates, or change which table is used as the beapp_file_info_table you may specify the file paths for your templates in beapp\user_inputs\beapp_set_input_file_locations.m. If any of these are left blank, BEAPP will use the default BEAPP files and tables located in beapp\user_inputs.

The code where you specify parameters for the beapp_set_input_file_locations function looks like this:



```
% set location for beapp to load user inputs and advanced inputs
grp_proc_info.beapp_alt_user_input_location = {''}; % def = {''}, which will use b
grp_proc_info.beapp_alt_adv_user_input_location = {''}; % def = {''}, which will u

% set location for beapp to load mat, mff, or rerun file info tables
grp_proc_info.beapp_alt_beapp_file_info_table_location = {''}; % def = {''}, which
grp_proc_info.beapp_alt_rerun_file_info_table_location = {''}; % def = {''},  whic
```

IMPORTANT: If you don't want to use a different user input template or file info table, make sure the following values are set to default before your run.

Variables/ Inputs	**Description**	**Examples**	**Notes**
grp_proc_info.beapp_alt_user_input_location	File path for alternative BEAPP user inputs templates	= {"}; (default) = {'C:\example_dir\user_inputs\dataset2_settings.m'};	To load the beapp_userinputs.m template, = {"};
grp_proc_info.beapp_alt_adv_user_input_location	File path for alternative BEAPP advanced inputs templates	= {"}; (default) = {'C:\example_dir\user_inputs\dataset2_adv_settings.m'};	To load the beapp_advinputs.m template, = {"};
grp_proc_info.beapp_alt_beapp_file_info_table_location	File path for alternative BEAPP file info tables	= {"}; (default) = {'C:\example_dir\user_inputs\dataset2_beapp_file_info_table.mat'};	To load the beapp_file_info_table.mat table, = {"};
grp_proc_info.beapp_alt_rerun_fselect_table_location	File path for alternative BEAPP rerun_fselect_table	= {"}; (default) = {'C:\example_dir\user_inputs\dataset2_rerun_fselect_table.mat'};	To load the rerun_fselect_table.mat table, = {"};

Running BEAPP with Different Source File Formats

Back (Start-Up) Back (src_format_typ)

All source files for a given run should be placed in the source directory listed in the user inputs.

Unsegmented .mff (EGI) Files (Baseline or Event Related, 1 or More NetStation Epochs)

To run unsegmented .mff files you'll need to enter the appropriate source directory and user settings and set grp_proc_info.src_format_typ = 2 (or select continuous .mff in the GUI). mff files with multiple NetStation epochs can be read by BEAPP using this setting. If your files have different event tag offsets, you'll need to indicate individual file names and file offsets in the beapp_file_info_table in the user_inputs folder, and change grp_proc_info.event_tag_offsets = 'input_table' (or select the corresponding checkbox in the GUI). You can also enter the exact names of possible net types into grp_proc_info.src_unique_nets for speed, although it's optional if you're not running HAPPE. In the GUI, entering all relevant net types is required.

Segmented .mff (EGI) Files (Baseline or Event Related)

To run segmented .mff files you'll need to enter the appropriate source directory and user settings, and set grp_proc_info.src_format_typ = 3 (or select segmented .mffs in the GUI). If your files have different event tag offsets, you'll need to indicate individual file names and file offsets in the beapp_file_info_table in the user_inputs folder, and change grp_proc_info.event_tag_offsets = 'input_table' (or select the corresponding checkbox in the GUI). Make sure you've done all the preprocessing necessary for your files before they're read into BEAPP – you'll only be able to use BEAPP output metrics, since the files are already segmented.

.mat exports (from EGI, Biosemi, ANT, etc.)

BEAPP can run MATLAB file exports from a variety of acquisition setups. Each .mat file must contain a variable with the (unsegmented) EEG data. The name of this variable can be whatever you typically use in your exports but should be consistent across files (even if your dataset contains files from mixed acquisition setups). You should specify the name(s) of this variable in grp_proc_info.src_eeg_vname. In addition, for all files you would like to run, you must add a line with the file name, sampling rate, and EEG net type to the beapp_file_info_table in the user_inputs folder and re-save the table (and no other variables listed under your Workspace). Files that do not exist in BOTH the source directory and the beapp_file_info_table will NOT be run through the pipeline.

.set (EEGLAB) Files

BEAPP can run EEGLAB files from a variety of acquisition setups, provided they have not been previously segmented (at least in the beta version). For all files you would like to run, you must add a line with the file name and EEG net type to the beapp_file_info_table in the user_inputs folder (or another table you've selected in the user_inputs folder) and re-save the table (and no other variables listed under your Workspace). Files that do not exist in BOTH the source directory and the beapp_file_info_table will NOT be run through the pipeline. If your files have different event tag offsets, you'll need to indicate individual file names and file offsets in the beapp_file_info_table in the user_inputs folder, and change grp_proc_info.event_tag_offsets = 'input_table' (or select the corresponding checkbox in the GUI).

Generating a BEAPP File Info Table

Back (Start-Up) Back (Src Line Noise) Back (Offsets) Back (Rerun Modules) Back (Save File Info Table) Back (File Event Tags Offsets)

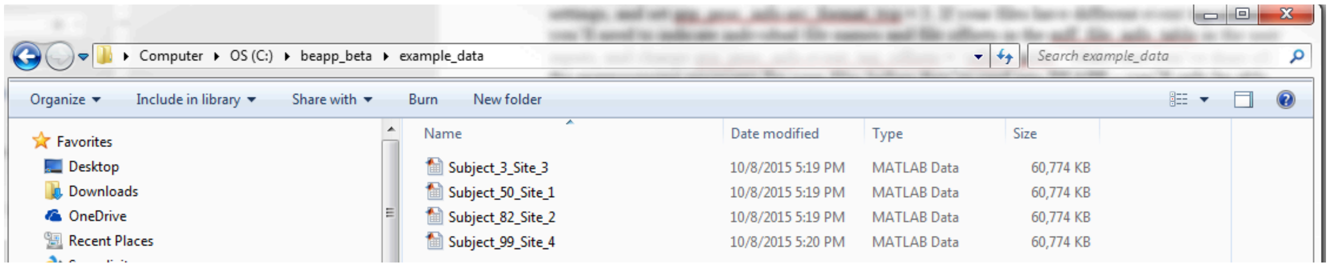
Users should generate BEAPP file info tables when running .mat files or when running files with line noise or offset information that varies within the dataset. Current table options include file names(FileName), sampling rates(SamplingRate), net types(NetType), event offsets(FileOffset), and line noise frequencies(Line_Noise_Freq). The table must always include these columns, but information that is not required can be left as NaNs or '' if not applicable. The requirements for each source file are provided below:

- **.mat files (Required)**
  - Set up beapp_file_info_table.mat with the following information for each file:
    - File Name
    - Sampling rate
    - Net Type (See section on BEAPP Net Library)
    - Line noise frequency (Typically 50Hz or 60Hz, depending on country. See section on Line Noise Frequency in Formatting Specifications.)

- **.mff files (Optional unless line noise/offsets vary between files within the dataset)**
  - You have the option to set up a 'beapp_file_info_table.mat' with information about line noise frequency and/or event offsets for each file. If you (1) do not have event offsets (e.g. if event offsets are already accounted for in your .mff file) or (2) do not have line noise or event offsets that vary between files within the dataset, you do not need to create this table.

- **.set files (Required)**
  - Set up beapp_file_info_table.mat with the following information for each file:
    - File Name
    - Net Type (See section on BEAPP Net Library)
    - If you have event offsets that are not already accounted for in your .set file, you can include event offset
    - Line noise frequency (Typically 50Hz or 60Hz, depending on country. See section on Line Noise Frequency in Formatting Specifications.)

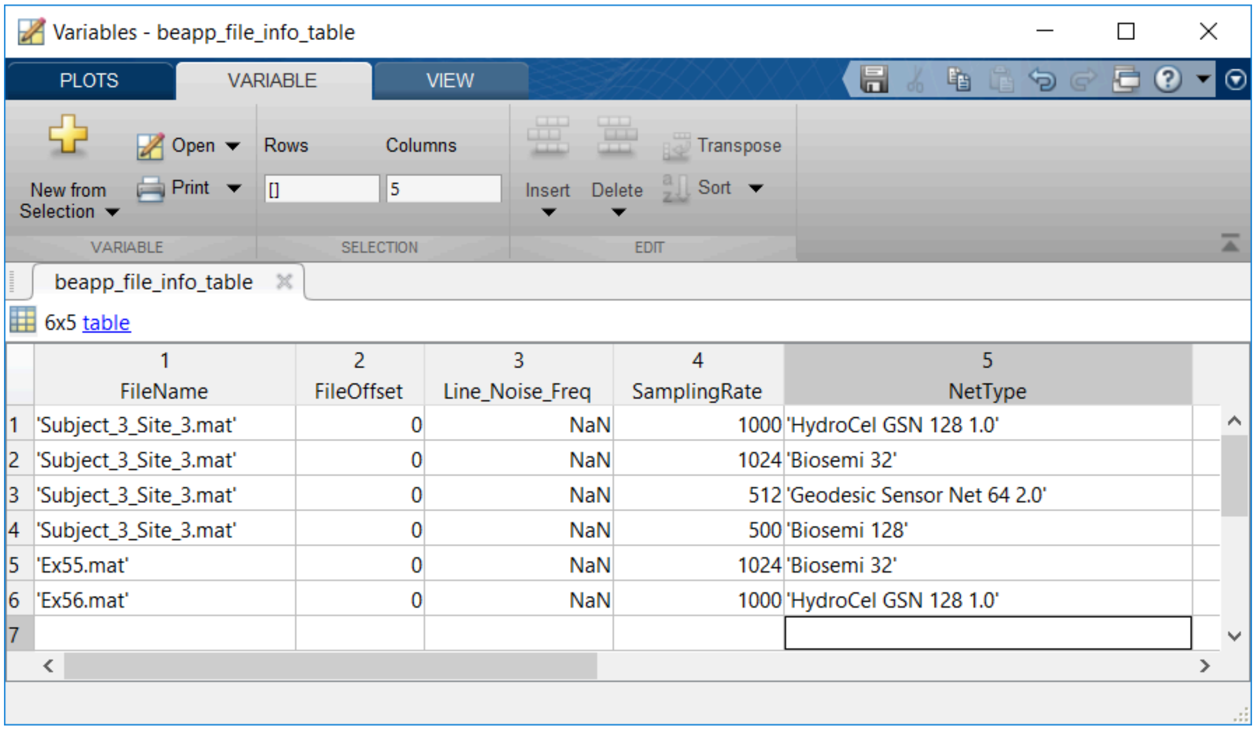
In the example below, the user has 4 source files exported to MATLAB, all from different sites and collected with different nets. The user has placed them all in the source directory example_data:



Name	Date modified	Type	Size
Subject_3_Site_3	10/8/2015 5:19 PM	MATLAB Data	60,774 KB
Subject_50_Site_1	10/8/2015 5:19 PM	MATLAB Data	60,774 KB
Subject_82_Site_2	10/8/2015 5:19 PM	MATLAB Data	60,774 KB
Subject_99_Site_4	10/8/2015 5:20 PM	MATLAB Data	60,774 KB

In MATLAB the user has then navigated to beapp\user_inputs\beapp_file_info_table and opened the file by double clicking, before opening the beapp_file_info_table variable from the MATLAB workspace.

The user has then entered the file name, the sampling rate, and the net type for each of these files. Each net name should match the full name of one of the options in the table under reference_data\net_library_options.mat exactly unless a new net type is being introduced. If a new net type is being used, the user can use a consistent, new net name for all the relevant files, as they will be prompted to add the net later. Note that having additional file listings in the table is fine – only files listed in the table that are also present in the current source directory will be run.



Variables - beapp_file_info_table

6x5 table

	1 FileName	2 FileOffset	3 Line_Noise_Freq	4 SamplingRate	5 NetType
1	'Subject_3_Site_3.mat'	0	NaN	1000	'HydroCel GSN 128 1.0'
2	'Subject_3_Site_3.mat'	0	NaN	1024	'Biosemi 32'
3	'Subject_3_Site_3.mat'	0	NaN	512	'Geodesic Sensor Net 64 2.0'
4	'Subject_3_Site_3.mat'	0	NaN	500	'Biosemi 128'
5	'Ex55.mat'	0	NaN	1024	'Biosemi 32'
6	'Ex56.mat'	0	NaN	1000	'HydroCel GSN 128 1.0'
7					

Once the table has been updated, the user should save just the beapp_file_info_table variable into beapp_file_info_table.mat in the user_inputs folder by navigating to the user_inputs directory in MATLAB, and then entering the following into the MATLAB command window:

```
save('beapp_file_info_table.mat','beapp_file_info_table');
```

Users wishing to save a copy of the table with a **different name** (for reuse when switching between datasets) should instead enter the command:

```
save('desired_table_name.mat','beapp_file_info_table');
```

Users wishing to save a copy of the table in an **alternative location** (for reuse when switching between datasets) should instead enter the command:

```
save('desired_location\desired_table_name.mat','beapp_file_info_table');
```

IMPORTANT: The variable name must be 'beapp_file_info_table', but the file name/file location can be anything you want. Remember you must specify the file name/file location in beapp_set_input_file_location.m

An example script for generating this table automatically can be found at \reference_data\example_scripts\gen_beapp_file_info_table_basic_case.m. Users are encouraged to modify this script to generate this table for their own data. While this script assumes all the files have the same net type, sampling rate, and line noise frequency, an example for generating the beapp_file_info_table automatically for a more complex case can be found under \reference_data\example_script\ISP_gen_beapp_file_info_table.

Users wishing to generate a rerun_fselect_table can use the same process, selecting the rerun_fselect_table.mat file in the user_inputs, entering appropriate file information, and saving the table as a variable, rerun_fselect_table, inside rerun_fselect_table.mat. As with the beapp_file_info_table, users may also save the table in an alternative location.

Running BEAPP With Differently Structured Data

Back (Src Data Type)

Baseline

Running Standard Baseline Files

To run baseline files that contain only baseline EEG data (e.g., files that have had relevant sections pulled out in NetStation), set grp_proc_info.src_ data_type =1 in the user settings. If you have a .mff file where some epochs are entirely baseline and some are event related, you can also run the baseline epochs by setting grp_proc_info.src_data_type = 1 and selecting the desired epochs in grp_proc_info.epoch_inds_to_process.

Running Conditioned Baseline Files

Conditioned Baseline files include baseline data between event tags, including alternating or recurring sections of baseline data (e.g. eyes open, eyes closed). To run conditioned baseline files, set grp_proc_info.src_data_type = 3 in the user input script, or select this option in the 'Format' panel of the GUI.

Running Baseline Files With Time-Locked Information

Data can be run as event-tagged, despite being baseline. This might apply to baseline data with trial-based behavioral coding. See section on running event-tagged data below.

Running Event-Tagged Data

Back (Offsets) Back (GUI Event Seg Gen) Back (GUI Base Seg Gen) Back (DIN Info)

To run files with event-tagged data, you will need to set stimulus onset tag, condition name, and condition cell number information for the events you'd like to analyze. Segments are grouped by condition, so the events don't need to be from the same epoch, provided they're in the same file. Additionally, this event related information is first used in the Segmentation module, so if you'd like to change the subset of events you'd like to analyze, you can alter these settings and then rerun BEAPP starting at the Segmentation module. The events selected at Segmentation are automatically the ones used in the output modules.

The standard event related inputs are:

- List of strings of event tags to be analyzed (e.g., stm+, stim, evt+ etc)
    - This list should include all possible stimulus onset tags for the conditions selected, but the tags and conditions do not need to match one-to-one
        - For example, event markers for the Cat and Dog conditions can both be stm+, provided they have cell numbers that differ. The event tag list for Cat and Dog would therefore only include stm+.
- Start and end times for segments (relative to a given event tag)
- List of the conditions being analyzed, and corresponding cell numbers from the presentation software (not required)

    - This list of condition names does not need to match whatever name was given in the E-Prime file. If the cell numbers used for each condition have changed or are different at different sites, users can add additional rows of cell numbers that correspond to the appropriate conditions. **BEAPP will choose the set of cell numbers that has the most overlap** with the contents of a file as the assumed condition information.
        - **Note** - While BEAPP is designed to allow for some flexibility in condition cell number naming across sites, if different sites have assigned the same cell numbers for different conditions, users should create separate input scripts and run these files in separate batches.
    - Users also have the option to identify conditions using only the event tags.

Condition Options in BEAPP

Back (GUI Event Seg Condition) Back (GUI Event Seg Condition)

- The impetus for creating the option to manage conditions is that in some cases, the same event code (e.g., stm+) can be used to signify a variety of events. For example, let's say a participant is having EEG recorded during an oddball task in which they hear three different sounds: A standard syllable, a native syllable, and a nonnative syllable. Onset of all three syllables might be indicated by a stm+ tag, but within that tag there may be event "cell codes" that indicate exactly which of the 3 syllables was delivered. Here, BEAPP provides an opportunity to differentiate among these syllables (based on their "cell codes.")
- Additionally, in some datasets, different cell codes might be used to indicate different event conditions. For example, one site may have indicated Standard, Native, and Non-Native syllables as 1, 2, and 3, respectively. Another site may have indicated these as 10, 11, and 12 respectively. The options here allow BEAPP to handle such an occurrence.

File Event Tag Offsets

Back (Generate File Info Table - .mff) Back (Generate File Info Table - .set)

- In order to adjust for timing offsets in event tagging between presentation software (e.g. EPrime) and acquisition software (e.g. NetStation), users have two options.
    - If all files/events in the dataset have the same offset, the user may set this value for the dataset. This offset will usually be positive (for example, EPrime offsets typically cause event markers to appear before a participant is presented with a stimulus due to delays caused by presentation hardware), but in some cases may be negative.
    - If files have different offsets, users will need to provide file-specific information in a BEAPP File Info Table (see section on generating a BEAPP File Info Table).
        - An example script for generating this table automatically can be found in \reference_data\example_scripts under generate_beapp_file_info_table.m. Users are encouraged to modify this script to generate this table for their own data.

Example of File Event Tag Offsets

Let's say your EEG has an event tag every time the computer thinks it presents a picture of a face. However, you (as a careful researcher!) have learned that due to a variety of delays in the system wiring, the face in fact appears on the monitor that your experiment's participant can see 18 milliseconds later. You would therefore set your event offset to 18.

Notes for Event Processing in BEAPP

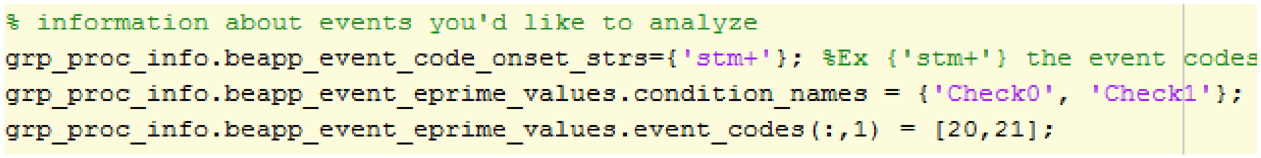
Back (GUI Event Seg Cond) Back (GUI Base Seg Cond)

1. The event offset is consistent within a file and is applied to all events in Format. BEAPP stores when events occur as both time in milliseconds and as samples relative to the start of the recording (evt_times_samp_rel). If Resampling is conducted, evt_times_samp_rel is recomputed to incorporate the event offset with the new sampling rate.
2. BEAPP selects the cell code/condition set provided by the user that has the best fit for the events in the file. BEAPP will skip files if the events present don't overlap with the provided cell codes. If 2 sets of cell codes have an equal amount of overlap with the events present in a file, the first set is always selected and BEAPP will display a warning.
3. Running all conditions in a file at once can be convenient, but because BEAPP will store segments (and later output metrics) for all selected conditions in each file, running too many conditions at once can dramatically increase the size of Segmentation output files and of output files.
4. Information about the events within a file can be found in file_proc_info.evt_info in the appropriate epoch. Sample numbers will correspond to the current sampling rate for a file, and will change during the Resampling module to match the new sampling rate.
5. By default, segments with the same event tag are collapsed across epochs.
6. Beta: At present, BEAPP can only run event related data from EGI .mff files and EEGLAB .set files

Examples of Event Related User Settings

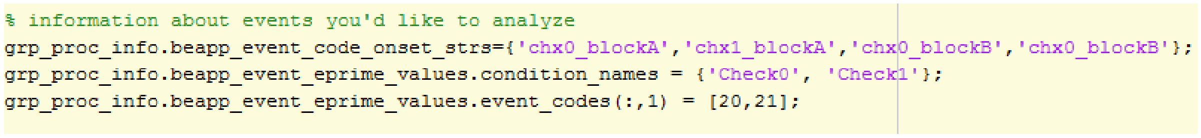
Back (Segment beapp_event_code_onset_strs), Back (Segment - beapp_event_eprime_values.condition_names), Back (Segment - beapp_event_eprime_values.event_codes)

Example 1: User is analyzing 2 conditions for one task. Stimulus onset was marked using the same tag for both conditions.



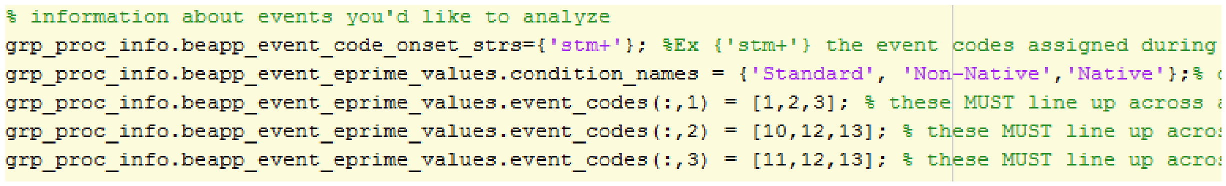
```
% information about events you'd like to analyze
grp_proc_info.beapp_event_code_onset_strs={'stm+'}; %Ex {'stm+'} the event codes
grp_proc_info.beapp_event_eprime_values.condition_names = {'Check0', 'Check1'};
grp_proc_info.beapp_event_eprime_values.event_codes(:,1) = [20,21];
```

Example 2: User is analyzing 2 conditions for one task. Stimulus onset was marked using different tags depending on the condition and block, but all events with the same condition have the same cell number. The user would like to analyze all Check0 trials together independent of block, and the same for Check1.



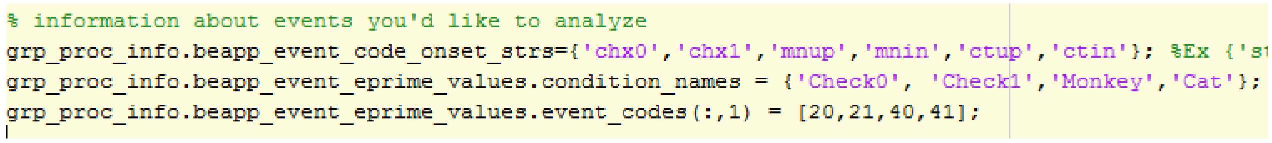
```
% information about events you'd like to analyze
grp_proc_info.beapp_event_code_onset_strs={'chx0_blockA','chx1_blockA','chx0_blockB','chx0_blockB'};
grp_proc_info.beapp_event_eprime_values.condition_names = {'Check0', 'Check1'};
grp_proc_info.beapp_event_eprime_values.event_codes(:,1) = [20,21];
```

Example 3: User is analyzing 3 conditions for one task. The stimulus onset for all of the conditions was marked using the same stm+ tag, but the cell numbers used to represent each condition were changed during the course of the study.



```
% information about events you'd like to analyze
grp_proc_info.beapp_event_code_onset_strs={'stm+'}; %Ex {'stm+'} the event codes assigned during
grp_proc_info.beapp_event_eprime_values.condition_names = {'Standard', 'Non-Native','Native'};%
grp_proc_info.beapp_event_eprime_values.event_codes(:,1) = [1,2,3]; % these MUST line up across
grp_proc_info.beapp_event_eprime_values.event_codes(:,2) = [10,12,13]; % these MUST line up acro
grp_proc_info.beapp_event_eprime_values.event_codes(:,3) = [11,12,13]; % these MUST line up acro
```

Example 4: User is analyzing 2 conditions for a checkerboard task and 2 conditions for a task that displays monkeys and cats (here, mn and ct, either in inverted or upright position). Each condition only has one associated cell number.



```
% information about events you'd like to analyze
grp_proc_info.beapp_event_code_onset_strs={'chx0','chx1','mnup','mnin','ctup','ctin'}; %Ex {'s
grp_proc_info.beapp_event_eprime_values.condition_names = {'Check0', 'Check1','Monkey','Cat'};
grp_proc_info.beapp_event_eprime_values.event_codes(:,1) = [20,21,40,41];
```

Behavioral Coding Information

Beta: Behavioral coding information can be read in from files and can be seen in file_proc_info.evt_info, but excluding segments based on behavioral coding is not yet an option. Instructions for future usage below.

If behavioral coding information is encoded in the event tags, users have the option to add the event tags, event keys, and key values (strings or numbers) that contain information about trial quality. These should line up exactly across the three input cell arrays. If any of the relevant keys is marked "bad" for a trial, that trial will be excluded.

Variables/ Inputs	**Description**	**Examples**	**Notes**

grp_proc_info.behavioral_coding.events	List of events with behavioral coding.	= {''}; (default)
		= {'TRSP'};
		= {'TRSP','TRSP2'};
			Optional

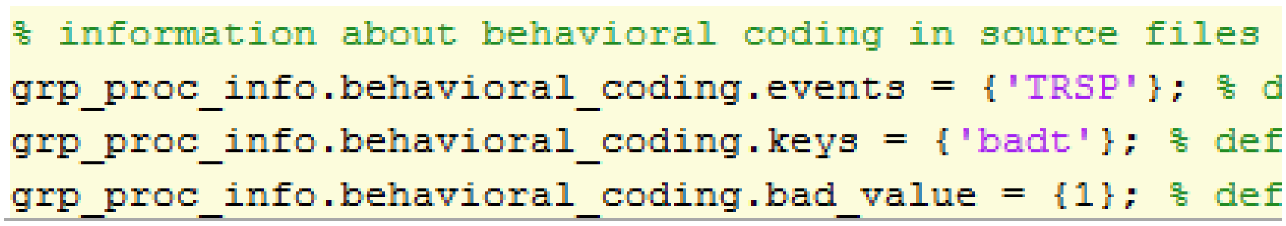
grp_proc_info.behavioral_coding.keys	List of keys with behavioral coding in events.	= {'badt'};
		= {'badtf','badth'};
			Optional

grp_proc_info.behavioral_coding.bad_value	Value that marks behavioral coding as bad	= {''};
		= {1};
			Optional

Examples of Behavioral Coding Information

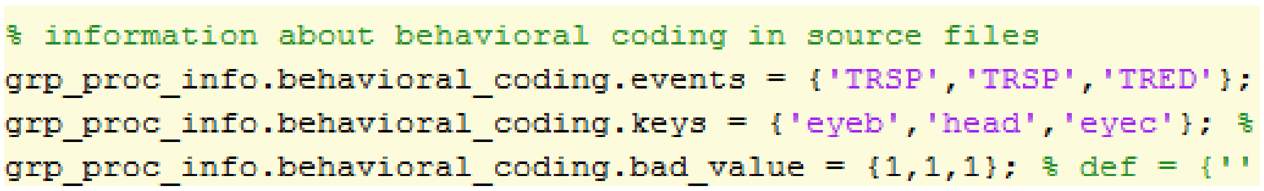
Back (Behav Code Marking)

Example 1: Bad trial information contained in the TRSP tag under the badt key, with bad value 1.



```
% information about behavioral coding in source files
grp_proc_info.behavioral_coding.events = {'TRSP'}; % d
grp_proc_info.behavioral_coding.keys = {'badt'}; % def
grp_proc_info.behavioral_coding.bad_value = {1}; % def
```

Example 2: Bad trial information contained in the TRSP and TRED tags under the eyeb, head, and eyec keys, with bad value 1. If any of the relevant keys is marked "bad" for a trial, that trial will be marked bad.



```
% information about behavioral coding in source files
grp_proc_info.behavioral_coding.events = {'TRSP','TRSP','TRED'};
grp_proc_info.behavioral_coding.keys = {'eyeb','head','eyec'}; %
grp_proc_info.behavioral_coding.bad_value = {1,1,1}; % def = {''
```

DIN Tag Information

DIN tags that have been pre-matched can be treated as regular events, following the instructions above. Beta: DIN matching functionality and instructions to come.

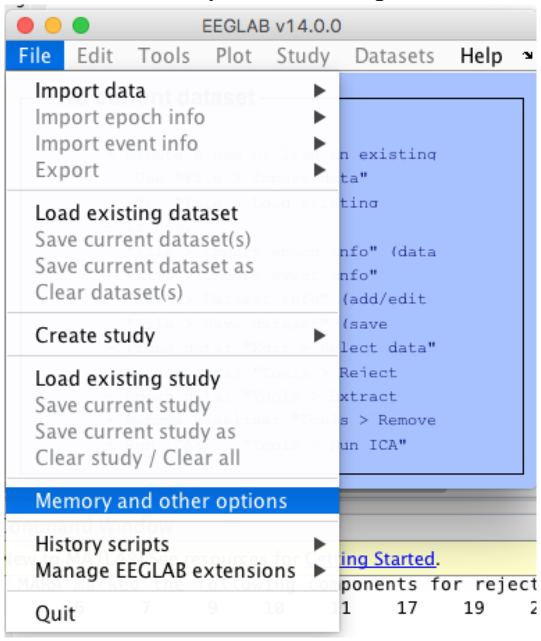
ICA Module/HAPPE Preparations

Back (Start-Up) Back (ICA/HAPPE) Back (GUI ICA/HAPPE)

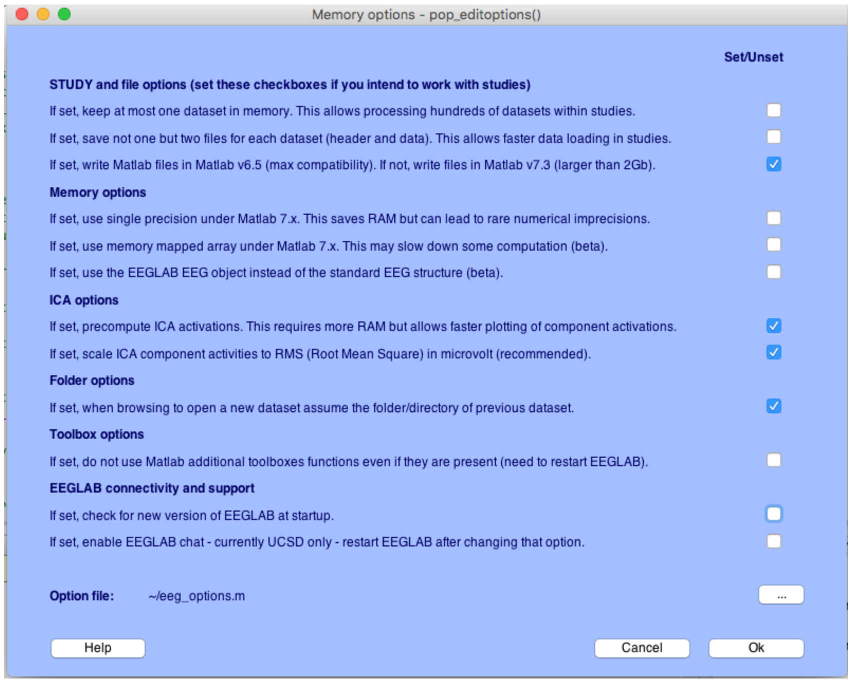
Before running the ICA module for the first time, follow the instructions below. The page thereafter includes information on HAPPE settings in BEAPP's user inputs.

To run the MARA algorithm with ICA, MATLAB needs the ‘Optimization Toolbox’ and the ‘Statistics and Machine Learning Toolbox’ installed. Information on how to add Toolboxes to MATLAB can be found here.

Before running HAPPE for the first time, users will need to configure EEGLAB memory settings. In MATLAB, navigate to your eeglab folder (found in beapp\Packages\eeglab14_1_2b). After opening the EEGLAB GUI (type **eeglab** in the MATLAB command window to open the GUI), users should navigate to File>Memory and other options.



Settings for this panel should match the image below. Click 'Ok' before quitting EEGLAB:


Memory options - pop_editoptions()

Set/Unset

STUDY and file options (set these checkboxes if you intend to work with studies)

If set, keep at most one dataset in memory. This allows processing hundreds of datasets within studies. ☐

If set, save not one but two files for each dataset (header and data). This allows faster data loading in studies. ☐

If set, write Matlab files in Matlab v6.5 (max compatibility). If not, write files in Matlab v7.3 (larger than 2Gb). ✓

Memory options

If set, use single precision under Matlab 7.x. This saves RAM but can lead to rare numerical imprecisions. ☐

If set, use memory mapped array under Matlab 7.x. This may slow down some computation (beta). ☐

If set, use the EEGLAB EEG object instead of the standard EEG structure (beta). ☐

ICA options

If set, precompute ICA activations. This requires more RAM but allows faster plotting of component activations. ✓

If set, scale ICA component activities to RMS (Root Mean Square) in microvolt (recommended). ✓

Folder options

If set, when browsing to open a new dataset assume the folder/directory of previous dataset. ✓

Toolbox options

If set, do not use Matlab additional toolboxes functions even if they are present (need to restart EEGLAB). ☐

EEGLAB connectivity and support

If set, check for new version of EEGLAB at startup. ☐

If set, enable EEGLAB chat - currently UCSD only - restart EEGLAB after changing that option. ☐

Option file: ~/eeg_options.m

Once you've set up EEGLAB for HAPPE, you're ready to return to your user inputs.

IMPORTANT: Several of the BEAPP modules outside of HAPPE will impact how HAPPE is run. The creators of HAPPE have the following recommendations:

Formatting module

Users running HAPPE or ICA with MARA in BEAPP will need to have included accurate 10-20 electrode mappings when adding the appropriate nets to the net library (see the section below titled BEAPP Net Library). Additionally, if HAPPE is being run on a dataset with more than one EEG acquisition layout or net, you will need to set the following inputs accordingly.

The line noise removal portion of HAPPE will also refer to the line noise frequency specified in the input module.

PREP module

Because PREP interpolates bad channels, and because previously interpolated channels alter the integrity of ICA-MARA, it is recommended that the PREP module be turned off if a user will be running HAPPE.

Filtering module

If users turn on the Filtering module, HAPPE recommends turning off notch and cleanline.

Resampling module

HAPPE recommends resampling to 250.

Segmenting module

HAPPE recommends a threshold of 40 or 50 for grp_proc_info.art_thresh.

PAC, FOOOF, & Bycycle Module Preparations

Back (Start-Up)

When running FOOOF, PAC, and/or Bycycle for the first time in BEAPP, users must perform additional steps to make MATLAB compatible with the modules' functions. Please refer to the instructions below for more information.

Check Python Installation

FOOOF and PAC require a version of Python 3.5 or newer to be installed on the same computer as BEAPP. To determine if your computer already has the correct version of Python installed, do the following:

Windows Users

1. Type 'cmd' in the search bar of your computer and hit enter.
    a. This will open your command prompt.
2. In the command prompt, type 'python --version' and hit enter.

Mac Users

1. Open your terminal.
2. In the terminal, type 'python –version' and hit enter.

If Python is already on your computer, then the command prompt/terminal will return the version of Python installed (ex: “Python 3.10.0”). Here, you can check if your current version of Python is ≥ 3.5. If it is older than 3.5, then you will need to install a newer version.

If Python is not installed on your computer, then you will receive the message “'python' is not recognized as an internal or external command, operable program or batch file.”

Check Python + MATLAB Compatibility

Depending on the MATLAB version you use, only certain versions of Python are capable of interfacing with MATLAB. Please refer to this table to determine which versions of Python are compatible to use with your release of MATLAB.

For example, if you use the 2023a release of MATLAB, then you would need to have the 3.8, 3.9, or 3.10 version of Python installed.

Install Python

Within your BEAPP directory, we have included the installer for Python 3.7.0. If Python 3.7.0 is not compatible with your version of MATLAB, you will need to download a different version from online.

If Python 3.7.0 is compatible with your version of MATLAB, then follow the steps below:

1. Navigate to the “Installers” folder in your BEAPP directory.

This folder holds all the applications and files necessary to install Python 3.7.0 and download the FOOOF, PAC, and BYCYCLE toolboxes.

Windows Users

1. Double click 'python-3.7.0-amd64_windows.exe'
    a. This will open a window that says "Install Python 3.7.0 (64-bit)"

Mac Users

1. Double click 'python-3.7.0-macosx10.9.pkg

2. Check the box at the bottom of the pop-up window that says, "Add Python 3.7 to PATH."
3. Click the "Install Now" link.

If Python 3.7.0 is not compatible with your version of MATLAB, use the link from the previous section ('Check Python + MATLAB Compatibility') to determine which versions of Python are supported. Based on the table, perform a Google search for a downloadable link to install one of the compatible versions of Python. Once downloaded, execute the program and do steps 2 and 3 from above.

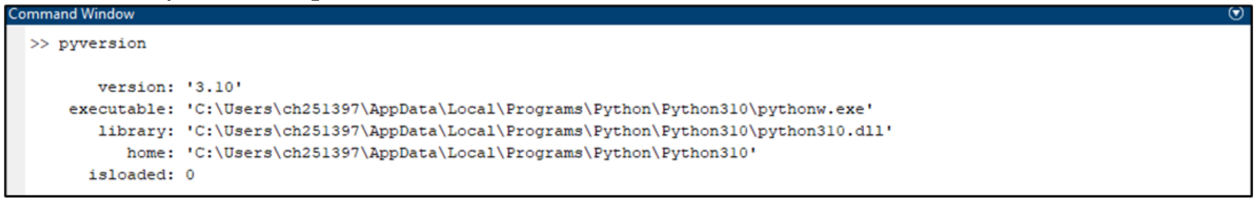
IMPORTANT: MATLAB does not support anaconda Python, so be sure to install Python a different way!

Load Python into MATLAB

Now that you have the correct version of Python installed on your computer, you will need to load it into MATLAB.
To determine if Python is already loaded into MATLAB, (and if so, what version of Python):

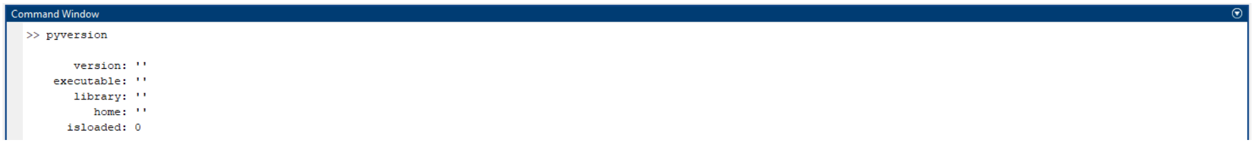
1. Type 'pyversion' in MATLAB's Command Window and hit enter.

If MATLAB has loaded Python, you should receive an output message like the figure shown below. If so, you can skip to the next section.



```
>> pyversion

      version: '3.10'
   executable: 'C:\Users\ch251397\AppData\Local\Programs\Python\Python310\pythonw.exe'
      library: 'C:\Users\ch251397\AppData\Local\Programs\Python\Python310\python310.dll'
         home: 'C:\Users\ch251397\AppData\Local\Programs\Python\Python310'
     isloaded: 0
```

If MATLAB has not loaded Python, the output message you receive will show the same parameters(version, executable, library, etc.) all set to empty values ("), as seen in the figure below. Depending on your computer system, refer to the different methods below to connect Python to MATLAB.



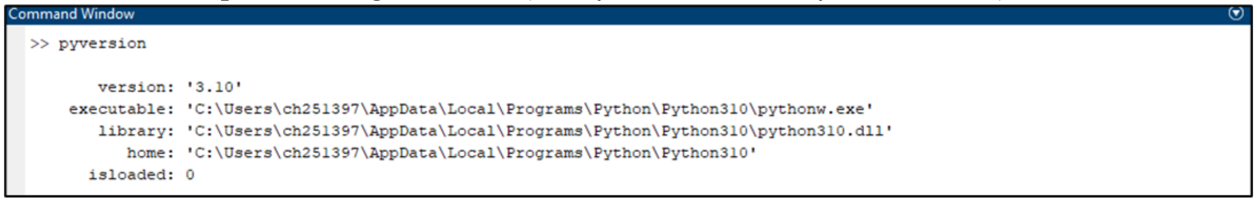
```
>> pyversion

      version: ''
   executable: ''
      library: ''
         home: ''
     isloaded: 0
```

Run 'pyversion'

Only Windows users can use this command to immediately connect Python to MATLAB.

Windows Users Only

1. Open MATLAB's Command Window and type 'pyversion 3.7' (or the version of Python you have installed) and hit enter. If Python is connected, you will see a message that looks like the output in the figure below (with your version of Python filled in).



```
>> pyversion

       version: '3.10'
    executable: 'C:\Users\ch251397\AppData\Local\Programs\Python\Python310\pythonw.exe'
       library: 'C:\Users\ch251397\AppData\Local\Programs\Python\Python310\python310.dll'
          home: 'C:\Users\ch251397\AppData\Local\Programs\Python\Python310'
      isloaded: 0
```

Find Python's Location

In order to load Python into MATLAB, Mac users need to find Python's location within their computer. There are two methods Mac users can use, which are both provided below. Windows users **do not** need to follow these instructions but can use them as an alternative method if the above steps were not successful.

Method 1: Mac Users & Windows Users

These instructions can be used for both Mac and Windows Systems.

1. Open the terminal (Mac) or command prompt (Windows).
2. Type 'python ' (do not hit enter).
3. Drag the file 'beapp\Installers\findmypython.py' into the terminal (Mac) or command prompt (Windows).
4. Hit enter.
   - This command will output Python's location in your computer.
5. Highlight and copy the output.
6. On MATLAB, run the command 'pyversion('copy_location_here')'.
7. Type ‘pyversion’ in the MATLAB command window again to see if it worked.

IMPORTANT: Make sure you run the command outside of the python interpreter – i.e., make sure there isn't “>>>” before your command line. If there is, type 'exit()' and press enter to exit python and return to command prompt (Windows) or terminal (Mac).

Method 2: Mac Users Only

Use this method if the instructions above did not work or if you prefer to select from multiple Python paths.

1. Open the terminal (Mac) or command prompt (Windows)
2. Type 'type -a python3' and hit enter.
   - This will output all file paths of python3 on your computer.
3. Copy Python's location
4. Restart MATLAB and run the command 'pyversion('*copy_location_here*')'

TIP: To confirm Python is connected, run the command 'pyversion' in MATLAB and compare the resulting message to the figure provided above.

Install FOOOF, Bycycle, and pactools Dependencies

AFTER you have installed Python, you still need to run the dependences installer. This file can be found within the Installers folder in the BEAPP directory. Depending on which computer system you use, refer to the appropriate instructions below.

Windows Users

1. Navigate to the "Installers" folder in your BEAPP directory through File Explorer.
2. Double click the file 'dependences_fooof_pactools_installer_win.bat' and hit 'Run'.
3. **You're all done! Enjoy BEAPP!!!**

Mac Users

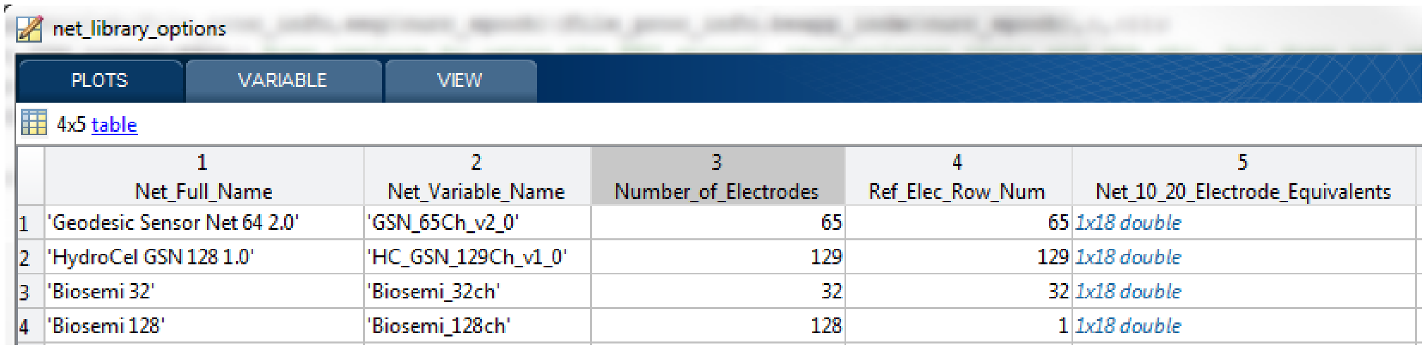
1. Open terminal.
2. Type 'chmod +x' (do not hit enter).
3. Drag the depedences_fooof_pactools_installer_mac into the terminal and hit enter.
    - This will allow you to run the dependences installer.
4. Type 'cd' (do not hit enter).
5. Drag the "Installers" file into the terminal and hit enter.
6. Type '.dependences_fooof_pactools_installer_mac.command' and hit enter.
7. **You're all done! Enjoy BEAPP!!!**

After following the instructions in this section, if you are still experiencing errors running FOOOF, PAC, or Bycycle modules in BEAPP, you may need to check that all the dependencies/toolboxes have been successfully imported.

BEAPP Net Library

Back (Src Unique Nets) Back (Rerun Modules) Back (Generate File Info Table) Back (Generate File Info Table) Back (ICA HAPPE Format)

The BEAPP net library contains .mat files with the coordinate information for each of the nets that have previously been used to run BEAPP on a given computer. You can see which nets are currently in the net library in the Formatting panel of the GUI or by opening the catalog. The latter is done by double clicking the net_library_options_table.mat file in the reference_data folder from the MATLAB viewer and clicking finish.



	1 Net_Full_Name	2 Net_Variable_Name	3 Number_of_Electrodes	4 Ref_Elec_Row_Num	5 Net_10_20_Electrode_Equivalents
1	'Geodesic Sensor Net 64 2.0'	'GSN_65Ch_v2_0'	65	65	1x18 double
2	'HydroCel GSN 128 1.0'	'HC_GSN_129Ch_v1_0'	129	129	1x18 double
3	'Biosemi 32'	'Biosemi_32ch'	32	32	1x18 double
4	'Biosemi 128'	'Biosemi_128ch'	128	1	1x18 double

In the net library catalog, the Net_Full_Name is the exact name of the net from the source file. For .mff files, this can be found in Netstation in Edit>File Info >Sensor Layout. If you are using beapp_file_info_table.mat file to run BEAPP, each distinct input you use for the variable NetType in beapp_file_info_table.mat should match exactly with one of the options in Net_Full_Name in net_library_options_table.mat file. The Net_Variable_Name is a user-defined abbreviation. If for some reason the same net can have different names in your source files, you can add multiple listings for the net with different Net_Full_Names, but the same Net_Variable_Name.

Adding New Nets to the BEAPP Library

Back (GUI)

If the data being run was collected with a net type that has not been used to run BEAPP on your computer before, you will need to add that net to the net library. This can be done either before running BEAPP or during a run when BEAPP detects files with the new net type. The latter will require the user to pay attention during the run, as BEAPP will prompt the user and pause until the net is added. You only need to add a given net once on each new computer, and users can transfer their net libraries and net library catalogs to new computers to avoid reentering the information.

Net Addition Instructions

Add New Nets to the Library Programmatically

1. Add New Nets to Library Before a Run:
   Recommended

a. To add a new net before a run, you will need the exact names of the nets you would like to add as they are listed in the source files. Enter the list of nets in the dataset (all, whether they are in the library or not) into the grp_proc_info.src_unique_nets variable in the user inputs. For example, if you'd like to add an EGI HCGSN 129 channel net and an EGI GSN 65 channel net to your library, but you already have 2 Biosemi nets in your dataset, you would set grp_proc_info.src_unique_nets = {'HydroCel GSN 128 1.0', 'Biosemi 32', 'Biosemi 128', 'Geodesic Sensor Net 64 2.0'}; in the user inputs.

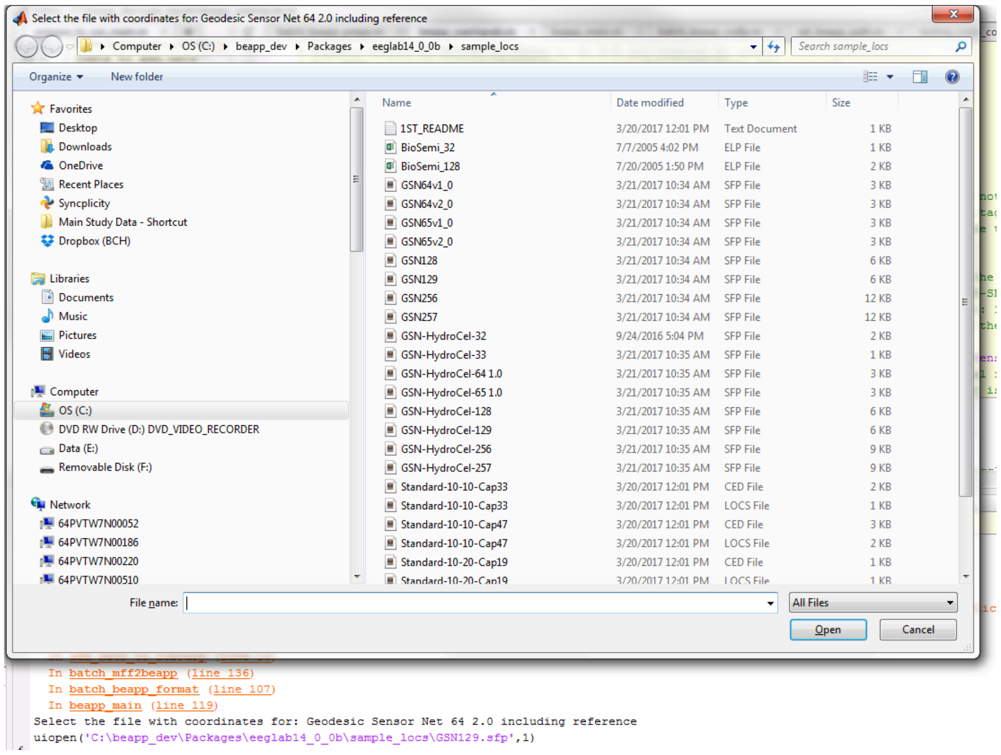
2. Add New Nets to Library During a Run:
   Not recommended

   a. Adding nets during a run is only recommended if you do not know the exact names of your nets as they exist in the file. If BEAPP detects a new net during a run, the user will be prompted to select the file containing the coordinates for that net type. Nets are then added as shown below.

Add New Nets to the Library Using the GUI

1. On the primary format and pre-processing panel, select "Add New Sensor Layout."
   b.

BEAPP will automatically prompt you to select the file with the relevant coordinate information for the new nets, as shown below:



Select the file with coordinates for: Geodesic Sensor Net 64 2.0 including reference

Computer ▸ OS (C:) ▸ beapp_dev ▸ Packages ▸ eeglab14_0_0b ▸ sample_locs

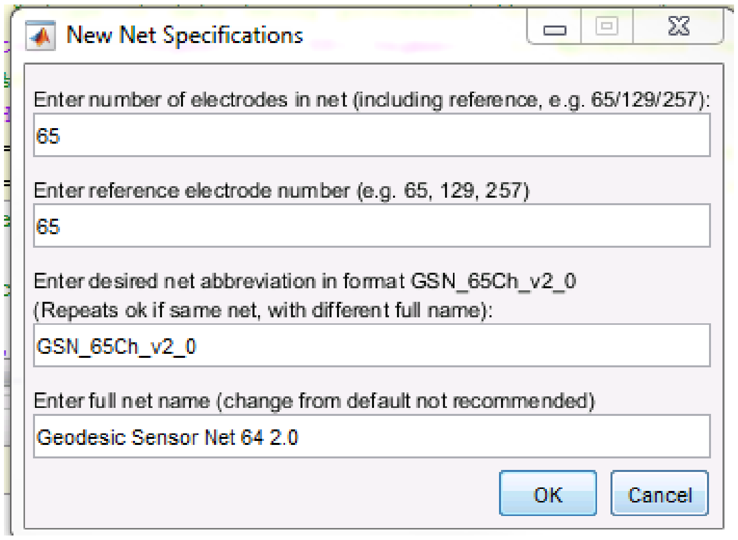
Name	Date modified	Type	Size
1ST_README	3/20/2017 12:01 PM	Text Document	1 KB
BioSemi_32	7/7/2005 4:02 PM	ELP File	1 KB
BioSemi_128	7/20/2005 1:50 PM	ELP File	2 KB
GSN64v1_0	3/21/2017 10:34 AM	SFP File	3 KB
GSN64v2_0	3/21/2017 10:34 AM	SFP File	3 KB
GSN65v1_0	3/21/2017 10:34 AM	SFP File	3 KB
GSN65v2_0	3/21/2017 10:34 AM	SFP File	3 KB
GSN128	3/21/2017 10:34 AM	SFP File	6 KB
GSN129	3/21/2017 10:34 AM	SFP File	6 KB
GSN256	3/21/2017 10:34 AM	SFP File	12 KB
GSN257	3/21/2017 10:34 AM	SFP File	12 KB
GSN-HydroCel-32	9/24/2016 5:04 PM	SFP File	2 KB
GSN-HydroCel-33	3/21/2017 10:35 AM	SFP File	1 KB
GSN-HydroCel-64 1.0	3/21/2017 10:35 AM	SFP File	3 KB
GSN-HydroCel-65 1.0	3/21/2017 10:35 AM	SFP File	3 KB
GSN-HydroCel-128	3/21/2017 10:35 AM	SFP File	6 KB
GSN-HydroCel-129	3/21/2017 10:35 AM	SFP File	6 KB
GSN-HydroCel-256	3/21/2017 10:35 AM	SFP File	9 KB
GSN-HydroCel-257	3/21/2017 10:35 AM	SFP File	9 KB
Standard-10-10-Cap33	3/20/2017 12:01 PM	CED File	2 KB
Standard-10-10-Cap33	3/20/2017 12:01 PM	LOCS File	1 KB
Standard-10-10-Cap47	3/20/2017 12:01 PM	CED File	3 KB
Standard-10-10-Cap47	3/20/2017 12:01 PM	LOCS File	2 KB
Standard-10-20-Cap19	3/20/2017 12:01 PM	CED File	1 KB
Standard-10-20-Cap19	3/20/2017 12:01 PM	LOCS File	1 KB

File name: | All Files

```
  In batch_mff2beapp (line 136)
  In batch_beapp_format (line 107)
  In beapp_main (line 119)
Select the file with coordinates for: Geodesic Sensor Net 64 2.0 including reference
uiopen('C:\beapp_dev\Packages\eeglab14_0_0b\sample_locs\GSN129.sfp',1)
```

By default, BEAPP will open to the EEGLAB sample_locs directory, but you can navigate anywhere to select your relevant channel locations files. BEAPP uses the eeglab pop_readlocs function to read channel positions, so channel positions can be in any format supported by EEGLAB.

- Select the appropriate file (including reference) and click open. You will then be prompted to enter some additional information about the net:


New Net Specifications

Enter number of electrodes in net (including reference, e.g. 65/129/257):
65

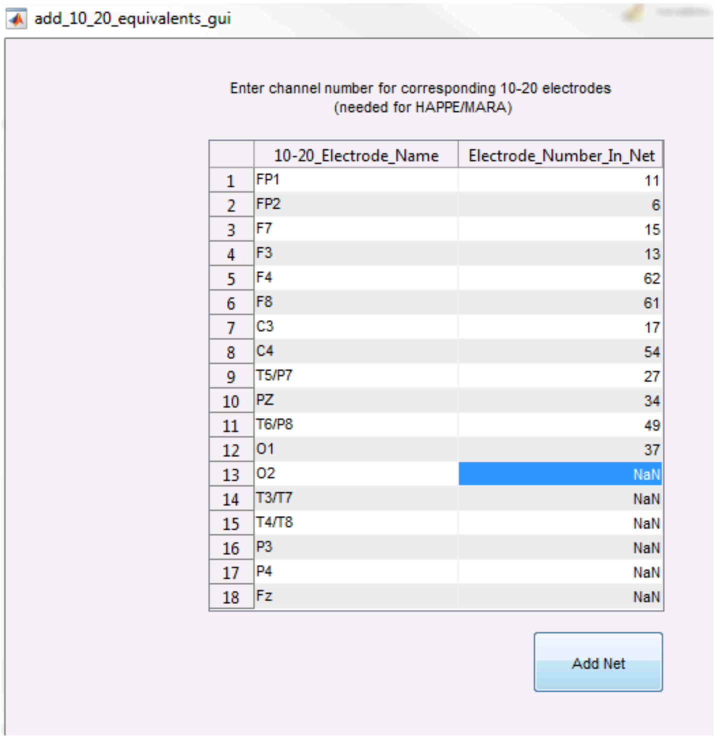
Enter reference electrode number (e.g. 65, 129, 257)
65

Enter desired net abbreviation in format GSN_65Ch_v2_0
(Repeats ok if same net, with different full name):
GSN_65Ch_v2_0

Enter full net name (change from default not recommended)
Geodesic Sensor Net 64 2.0

Enter the information and click 'OK'. Changing the full net name is not recommended. Instead, we recommend entering a net abbreviation name, as shown in the image above. Click 'OK' when you are finished.

- Finally, you will be prompted to enter which channel numbers (rows in the data) correspond to each of the 10 20 electrodes. Users not intending to use HAPPE can leave these as NaNs and select 'OK', but it's recommended to enter these correctly, as you will need this information if you ever decide you would like to run HAPPE or another 10-20 based analysis using this net type.


add_10_20_equivalents_gui

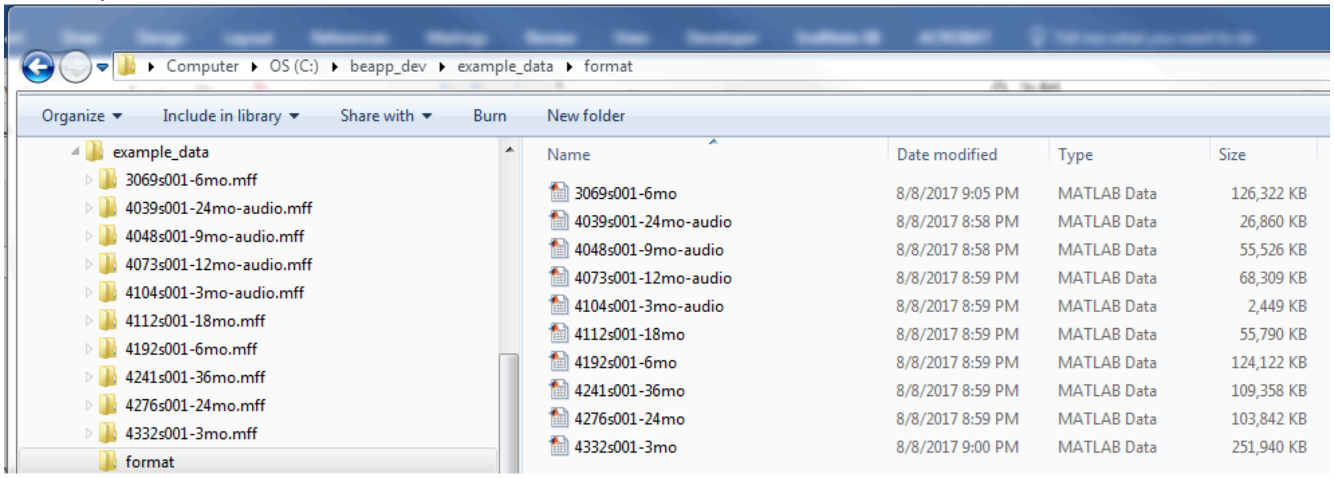
Enter channel number for corresponding 10-20 electrodes
(needed for HAPPE/MARA)

	10-20_Electrode_Name	Electrode_Number_In_Net
1	FP1	11
2	FP2	6
3	F7	15
4	F3	13
5	F4	62
6	F8	61
7	C3	17
8	C4	54
9	T5/P7	27
10	PZ	34
11	T6/P8	49
12	O1	37
13	O2	NaN
14	T3/T7	NaN
15	T4/T8	NaN
16	P3	NaN
17	P4	NaN
18	Fz	NaN

BEAPP Outputs and Reporting

Outputs

Outputs for each module in the pipeline will be stored as .mat files in the corresponding directory named for the module (see example below for Format module outputs). Each file will contain a file_proc_info structure variable within it with processing information about that file and a second variable containing the eeg data at the current stage of the pipeline. Files with no data left at the end of a module or with invalid source data will not be saved in a module's output directory.



example_data
- 3069s001-6mo.mff
- 4039s001-24mo-audio.mff
- 4048s001-9mo-audio.mff
- 4073s001-12mo-audio.mff
- 4104s001-3mo-audio.mff
- 4112s001-18mo.mff
- 4192s001-6mo.mff
- 4241s001-36mo.mff
- 4276s001-24mo.mff
- 4332s001-3mo.mff
- format

Name	Date modified	Type	Size
3069s001-6mo	8/8/2017 9:05 PM	MATLAB Data	126,322 KB
4039s001-24mo-audio	8/8/2017 8:58 PM	MATLAB Data	26,860 KB
4048s001-9mo-audio	8/8/2017 8:58 PM	MATLAB Data	55,526 KB
4073s001-12mo-audio	8/8/2017 8:59 PM	MATLAB Data	68,309 KB
4104s001-3mo-audio	8/8/2017 8:59 PM	MATLAB Data	2,449 KB
4112s001-18mo	8/8/2017 8:59 PM	MATLAB Data	55,790 KB
4192s001-6mo	8/8/2017 8:59 PM	MATLAB Data	124,122 KB
4241s001-36mo	8/8/2017 8:59 PM	MATLAB Data	109,358 KB
4276s001-24mo	8/8/2017 8:59 PM	MATLAB Data	103,842 KB
4332s001-3mo	8/8/2017 9:00 PM	MATLAB Data	251,940 KB

File Processing Information

Back (Format) Back (PREP) Back (Filter) Back (Resamp) Back (ICA/HAPPE) Back (Re-ref) Back (Detrend) Back (Segment) Back (PSD) Back (ITPC) Back (PAC)

File processing information for each BEAPP output file can be found in the file_proc_info variable. To access this information, double click on the data file in the appropriate module folder to load it into Matlab, and then double click on the file_proc_info variable in the workspace. Each variable stored in file_proc_info should have a prefix related to its function. The list below is not exhaustive of all variables in file_proc_info. Generally:

Prefix	**Description**	**File_proc_info Field Example**
.beapp	Current file information	.beapp_srate (the current file srate)
.src	Information related to the source file format	.src_srate (original srate)
.evt	Information related to file events	.evt_info (struct containing event tag information for a file)

.hist	File/run history information	.hist_run_table (information about modules applied to file)
.epoch	Information about recording periods (this will change to .rec_period in future versions)	.src_epoch_end_times (recording period end times in .mff files)
.net	Information about nets used	.net_typ (file net name)
.ref	Information about reference data/resources	.ref_net_library_dir (reference net library)
.seg	Information about segments created outside BEAPP	.seg_info (struct with segment information generated in pre-segmented .mff files)

Depending on the processing stage and the source file type, you will see some or all of the variables below (recording periods are listed as epochs for the time being):

.beapp Variables Current file information ● Here, epochs are recording periods.		
File_proc_info Variable Name	**Description**	**Notes**
file_proc_info.beapp_bad_chans	List of bad channels detected in BEAPP for each epoch analyzed from source file	List within each cell corresponds to each epoch. Initialized as an empty variable during Format Module.
file_proc_info.beapp_fname	BEAPP filename for this file	Set during Format Module.
file_proc_info.beapp_indx	List of channels BEAPP is using for analysis at the current stage	Will change depending on interpolation/removal/selection of channels. Set during Format Module.
file_proc_info.beapp_nchans_used	Number of channels being used by BEAPP in each epoch analyzed	Will change depending on interpolation/removal/selection of channels. Set during Format Module.
file_proc_info.beapp_num_epochs	The number of epochs from the source file that BEAPP is analyzing	Impacted by user selection of epochs in inputs. Set during Format Module.

file_proc_info.beapp_srate	The current sampling rate for the file	Will differ from src_srate if file has been resampled. Set during Format Module.
file_proc_info.beapp_win_size_in_samps	Window size in samples (calculated using beapp_rsamp_srate)	Only output for baseline files. Set during Segmentation Module.
file_proc_info.beapp_filt_max_freq	Maximum Frequency used to filter signal	Set during Filter Module.
file_proc_info.selected_segs	Indices for a subset of trials to use for each file/condition	Only set if grp_proc_info.win_select_n_trials is set to >=0.

.epoch Variables
Epoch/recording period information

File_proc_info Variable Name	**Description**	**Notes**
file_proc_info.epoch_inds_to_process	Indices of epochs in source file that were chosen for processing in this file	Set during Format module.

.evt Variables
Information related to file events
- All fields set during Segmentation Module.

File_proc_info Variable Name	**Description**	**Notes**
File_proc_info.evt_conditions_being_analyzed	Table with the event codes, user given condition name, and native file condition name for each condition analyzed in that file	May be a subset of events in source file – depends on user event selection.
File_proc_info.evt_header_tag_information	Information extracted from file event track header (.mff files)	May be empty.
File_proc_info.evt_info	Information about all event tags present in file: time, sample number, epoch, label, cell number, bad trial coding	Separated by source file epoch (each cell contains event information from a different epoch).

File_proc_info.evt_seg_win_evt_ind	Index of event tag within original segmentation window	Used to find edges of analysis window.

.grp Variables Group wide information that cannot change		
File_proc_info Variable Name	**Description**	**Notes**
File_proc_info.grp_wide_possible_cond_names_at_segmentation	Ordered condition names at Segmentation, determines order of segments in eeg_w in all files produced in that run	Set during the Segmentation module.

.ica Variables ICA module stats		
File_proc_info Variable Name	**Description**	**Notes**
File_proc_info.ica_stats	Contains channel and ICA information for this file if ICA or HAPPE was run (the same information can be found in the HAPPE reports)	Only present if ICA has been run. Set during ICA module.

.hist Variables File history		
File_proc_info Variable Name	**Description**	**Notes**
File_proc_info.hist_run_table	Contains information on which modules have been applied to this file, the start timestamp for each module, how long it took to process each file in each module, and what tag name was used	Set for every BEAPP run.
File_proc_info.hist_run_tag	The start timestamp for when the file is created	Set for every BEAPP run.

.net Variables File net information

All fields set during Format Module.		
File_proc_info Variable Name	**Description**	**Notes**
File_proc_info.net_10_20_elecs	Electrodes corresponding to 10_20 electrodes for this net	Information set and accessible in the net library catalog.
File_proc_info.net_happe_additional_channel_lbls	List of additional channels (beyond 10-20s) analyzed using HAPPE for this net	Set by user.
file_proc_info.net_typ	File net type name	
file_proc_info.net_vstruct	Net channel positions for this file	
File_proc_info.ref_elec_rnum	Index of reference electrode for this net	Information set and accessible in the net library catalog.

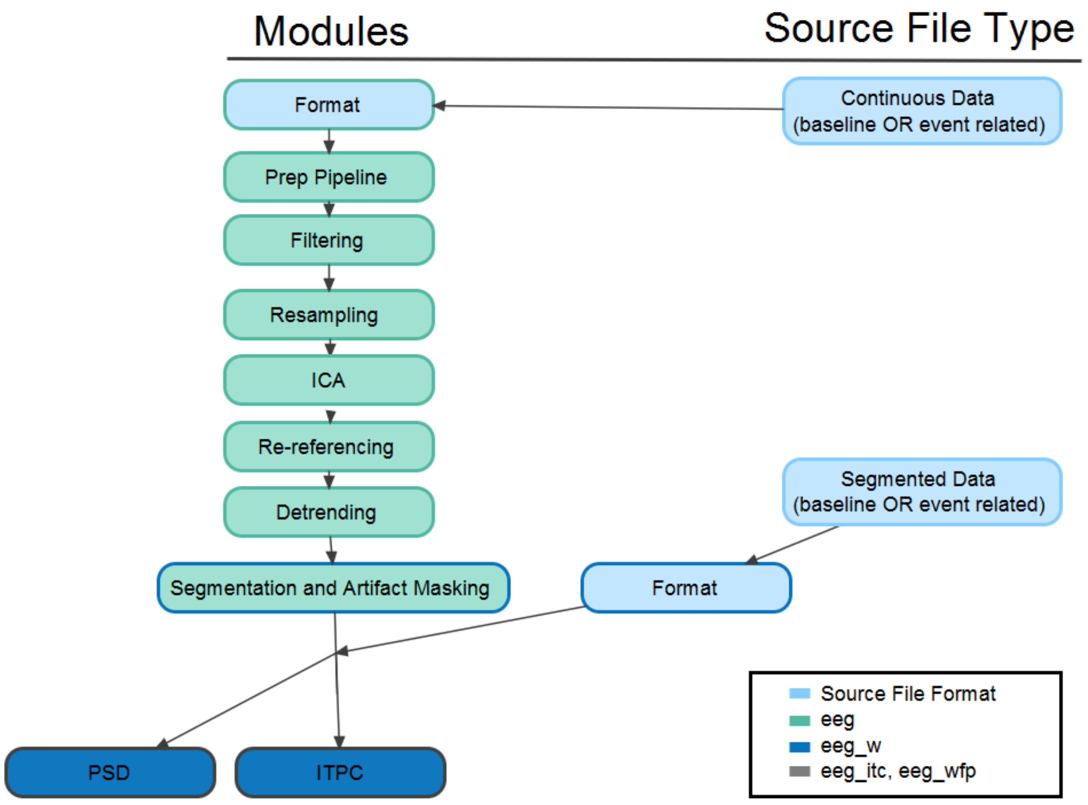
.seg Variables File pre-generated segment information ● This may appear in files with large "segments" used to pull paradigms out of a recording in NetStation, even if the files were not pre-segmented (.src_format_typ = 3) .mffs. ● Fields set during Format Module.		
File_proc_info Variable Name	**Description**	**Notes**
File_proc_info.seg_info	Information about all the segments present in file: time, label, cell number, bad trial coding, hand editing information	Only present in files read into BEAPP after Segmentation
File_proc_info.seg_tasks	Tasks present in pre-generated segments	

.src Variables Source file information ● Some of the variables unlikely to be useful to users have been excluded from this list. ● Fields set during Format module.		
File_proc_info Variable Name	**Description**	**Notes**
file_proc_info.src_amp_serial	Amplifier serial number	

file_proc_info.src_amp_type	Amplifier type	
file_proc_info.src_epoch_nsamps	Number of samples (using source srate) in each epoch in original file	
file_proc_info.src_file_offset_in_ms	File offset in ms (user input)	
file_proc_info.src_fname	Source filename for this file	
File_proc_info.src_linenoise	Source linenoise frequency (user input)	
File_proc_info.src_nchan	Original number of channels for this file	
File_proc_info.src_num_epochs	Number of recording periods in source file	Different from beapp_num_epochs depending on user selection of epochs in inputs
File_proc_info.src_record_start_day	Date file recorded	
File_proc_info.src_record_start_time	Time file recorded	
File_proc_info.src_srate	Original source file sampling rate	Will differ from beapp_srate if file has been resampled
File_proc_info.src_subject_id	Subject ID, if provided in source file	

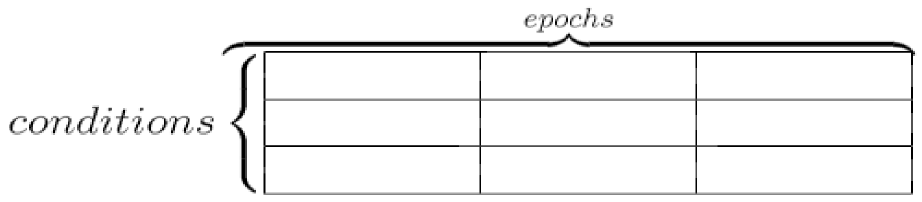
Individual EEG Data / EEG Analysis Outputs

All BEAPP EEG outputs are Matlab cell arrays with variable names that are dependent on the stage of the pipeline. The variable name for the eeg output is **eeg** (all lowercase) during the continuous data stage and is **eeg_w** after Segmentation. Output names from the analysis/output metric stage of the pipeline vary for each module (ex: eeg_itc, eeg_wfp).



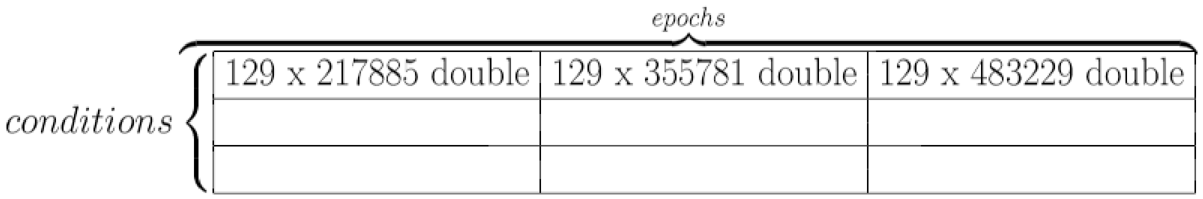
Cell body color indicates input format, cell outline indicates output format

Each cell in the cell array contains a numerical 2D or 3D array with the EEG data, depending on the pipeline stage. In general, BEAPP EEG output cell arrays are organized by splitting data horizontally by epoch before Segmentation occurs, and then vertically by condition after Segmentation, as shown below:



Continuous / Not Yet Segmented EEG Outputs (eeg)

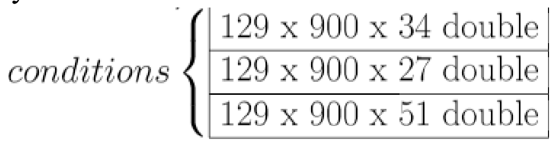
All individual output files from modules that produce continuous data (all modules before Segmentation) will contain the file_proc_info variable and a second variable, eeg. The eeg variable is a cell array, with each cell representing an epoch from the original file and containing a 2D channel x sample array with data from the epoch. In the example below, 3 experiments were run during the participant's EEG session, each with its own epoch. Data collected from each of these epochs is placed in its own cell. Users wishing to access data from the first epoch, for example, should load the relevant output file, and access eeg{1}.


epochs
conditions

129 x 217885 double	129 x 355781 double	129 x 483229 double

Segmented EEG Outputs

All individual output files from modules that produce segmented data will contain the file_proc_info variable and a second variable, eeg_w. Each cell in the eeg_w array will contain outputs for a given condition in the form of a 3D channel x sample x trial array. By default, segments with the same condition are collapsed across epochs. The order of the conditions used to store the cell array can be found in file_proc_info.evt_conditions_being_analyzed. In the example below, the user has selected 3 conditions to analyze, each 900 samples long, with 34, 27, and 51 trials, respectively.


conditions

129 x 900 x 34 double
129 x 900 x 27 double
129 x 900 x 51 double

Analyses / Output Metric EEG Outputs

The contents of the output files for analysis modules will depend on the analysis selected, but they all follow the same format as the segmented outputs. Each cell in the eeg_out array will contain outputs for a given condition in the form of a 3D array (e.g. channel x frequency x time for ITPC).

Reports

Run Reports

In the current version of BEAPP, a very rudimentary report of command line outputs (file specific warnings, for example) can be found in the out directory for a run named BEAPP_Run_Reporting_[run_tag].txt . Additionally, a MATLAB structure Run_Report_Variables_and_Settings.mat that contains all the user settings selected for that run can be found in the out directory. Formatted reporting will be added in future BEAPP versions.

Analysis / Module Reports

The PREP module will automatically generate a list of files that failed in PREP. Users have the option to generate reports in HAPPE, PREP, and all output modules (ITPC, PSD, etc.) using the user inputs. These reports are tables containing basic information for each file, along with the relevant outputs for each file in the dataset. A different table will be generated for each event type/condition, and for each kind of output metric selected (e.g. raw power vs. log power). Files that did not survive the pipeline (because of a lack of usable segments, for example) will have NaNs listed in place of an output metric. For further information about the HAPPE outputs, please see the companion paper detailing HAPPE by Gabard-Durnam and colleagues in this issue.

The output metrics for a run are also saved in the out folder, which may be useful for users who would like to continue processing the data in Matlab. For each output module, if grp_proc_info.beapp_toggle_mods{'[module]', 'Module_Xls_Out_On'} (ex: grp_proc_info.beapp_toggle_mods{'psd', 'Module_Xls_Out_On'}) is turned on, a .mat file named [module]_report_values_[beapp tag].mat (ex: psd_report_values_tag.mat) is generated, in which a cell array named report_values (dimension: condition x epoch) can be found. Each cell in report_values contains a 3D array (subject x observation x output metric). In addition to report_values, the [module]_report_values_[beapp tag].mat file also contains variables hdr_out (a list of the observation names) and tabnames (a list of the output metric names).

Running BEAPP through a Computer Cluster

Transfer BEAPP to your computer cluster

With a computer cluster, BEAPP has the capability to run each file in parallel.
The file cc_beapp_script(file_number) takes a given file number and uses it as an index into beapp_file_info_table. As a result, the user can write a script that runs each file as an individual job on the computer cluster. This is shown by beapp_parallel_jobs.sh (copied here):

```
#!/bin/bash
#SBATCH -p short
#SBATCH  -t 0-00:05
#SBATCH --mem=1G
#SBATCH -c 1
#SBATCH -n 1
#SBATCH --constraint="scratch2"

module load matlab/2017a
matlab -nodesktop -r "cc_beapp_script(${SLURM_ARRAY_TASK_ID})"
```

You will want to change the settings depending on your run and the commands depending on your computer cluster.

And here's cc_beapp_script:

```
try
   cd beapp_dev
   grp_proc_info = beapp_configure_settings;
   grp_proc_info.beapp_file_idx = file;
   grp_proc_info.beapp_run_per_file = 1;
   grp_proc_info.beapp_dir_warn_off = 1;
   beapp_main(grp_proc_info);
catch my_error
    my_error
    exit(1)
end
exit
```

This works if you're running cc_beapp_script from your home directory and if your BEAPP folder is named "beapp_dev" and is in your home directory. If that's not the case, either change the line "cd beapp_dev" to cd into your BEAPP or move /rename BEAPP.

To run this as a batch job, with each file queued in parallel, run
 sbatch --array=1-sizeofyourbeapp_file_input_table beapp_parallel_jobs.sh
Using a SLURM scheduler, this functionally works as a for loop, queueing the files in beapp_file_info_table.

IMPORTANT: If you can, we recommend running BEAPP interactively (on one or two of your files) before running as a batch script.

Cloning BEAPP Using Git

Back (Start-Up)

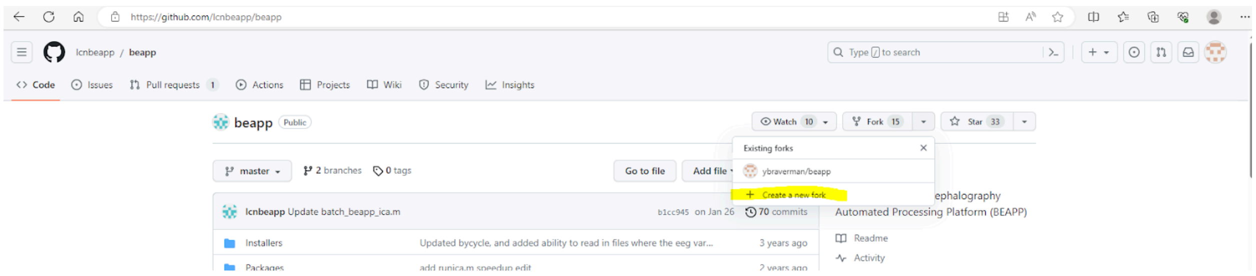
To clone BEAPP using Git you'll need to:

1. Have a github account
2. Download git revision software onto the computer you'll store BEAPP on.
3. Create a copy of BEAPP (aka a "Fork") onto your github account
4. Clone BEAPP from your remote github repository to your local machine

Forking

Back (Using Git)

1. Navigate to the set of code you wish to fork (ex: BEAPP) on a web browser
2. Click the "Fork" dropdown and select "+ Create a new fork"
3. Fill out the information in the next window and then hit "Create Fork"



lcnbeapp Update batch_beapp_ica.m b1cc945 on Jan 26 70 commits

Installers Updated bycycle, and added ability to read in files where the eeg var... 3 years ago

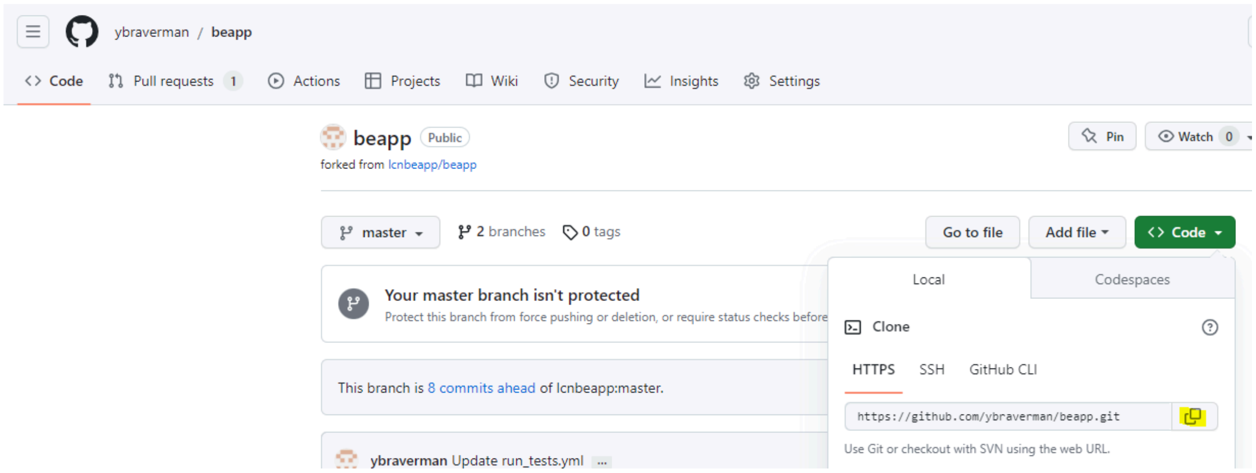
Packages add runica.m speedup edit 2 years ago

ephalography Automated Processing Platform (BEAPP)

Cloning

Back (Using Git)

1. Open Git
2. Navigate to the folder you want BEAPP to live in using the "cd" command
3. In your browser, navigate to your personal fork of BEAPP (ex: https://github.com/ybraverman/beapp) and copy the linked address under the "Code" button



forked from lcnbeapp/beapp

Your master branch isn't protected

Protect this branch from force pushing or deletion, or require status checks before

This branch is 8 commits ahead of lcnbeapp:master.

ybraverman Update run_tests.yml

https://github.com/ybraverman/beapp.git

Use Git or checkout with SVN using the web URL.

4. Back in your Git Bash window, type git clone [paste the url here] and hit enter. You'll now see BEAPP (or other code) downloaded in your desired folder and ready to use in MATLAB/other software!

Additional Information

BEAPP integrates the following preexisting software packages. These are provided in Packages in the BEAPP repository.

- EEGLab 14.0.0b
- EEGLAB plugins:
    - Dipfit 2.3
    - Firfilt1.6.2
    - MARA
    - PrepPipeline v0.52
    - Cleanline
    - Fieldtrip (only if using topoplots)
- CSD Toolbox
- REST Toolbox
- HAPPE
- Bycycle
- FOOOF
- PAC Tools