Connecting the EEG Muse Headset with PsychoPy Experiments

A Step-by-Step Interactive Manual

July 22, 2025

Contents:

1 Introduction

The Muse EEG captures brain activity using an EEG. An electroencephalogram (EEG) measures and records electrical activity in the brain using sensors, contained in the headband, which are later processed into neural oscillations. The Muse is used for a wide variety of health applications, including scientific and medical, as it serves in psychology for research utilities, in neurology as a clinical tool, and helps researchers understand brain activity and functions in the field of neuroscience, with higher-level equipment. In the context of accessibility, the Muse has the advantage of being low-cost and quick to configure, compared to traditional EEG machines. The simple connection to Bluetooth makes it easy to manipulate and experiment for everyone.

The following is a simple guide to help new users easily set up the Bluetooth Muse EEG headset to use with their PsychoPy experiments. Since this manual is focused on making the setup of the MuseEEG headset easy and accessible to everyone, this manual does not require extensive knowledge of programming.

In the first section, there will be instructions on what to install to conduct the experiment, depending on whether you are working in python or in the psychopy app. Then, in the second section, there will be straightforward guidance regarding the setup of the experiment. Additionally, new users with no experience in programming will acquire the knowledge on how to get the data by streaming, visualizing, and recording using their Anaconda environment. The final section is dedicated to common errors that you might get and how to solve them.

2 Installation of required software for integration in python code, Creation of Anaconda Environment, PsychoPy & Open Python Terminal

IMPORTANT: This section is the installation process for integration of muse EEG in a python code, assuming you're using the psychopy extension library in python. If you are using the psychopy app for computers and building your experiment in the in-app builder, skip this section, and see instead section 3: Installation of Required Software for Integration in Psychopy Builder.

2.1 Anaconda Navigator:

Anaconda is a software package that utilizes the Python language. It manages conda packages and environments without the need for a command line. Head to https://www.anaconda.com/products/navigator and follow the installation instructions. To avoid running into future problems, it is advised that during installation, you respond **'yes'** to 'Do you wish the installer to initialize Anaconda3 by running conda init?'. If you answered 'no', you will have to head to 7.1 or 7.2 in Possible Technical Issues and How to Fix Them.

2.2 Creation of an Anaconda Environment:

Once you have downloaded Anaconda Navigator, go to the left of the screen, and you should see a section named "Environments" (see Figure 1).

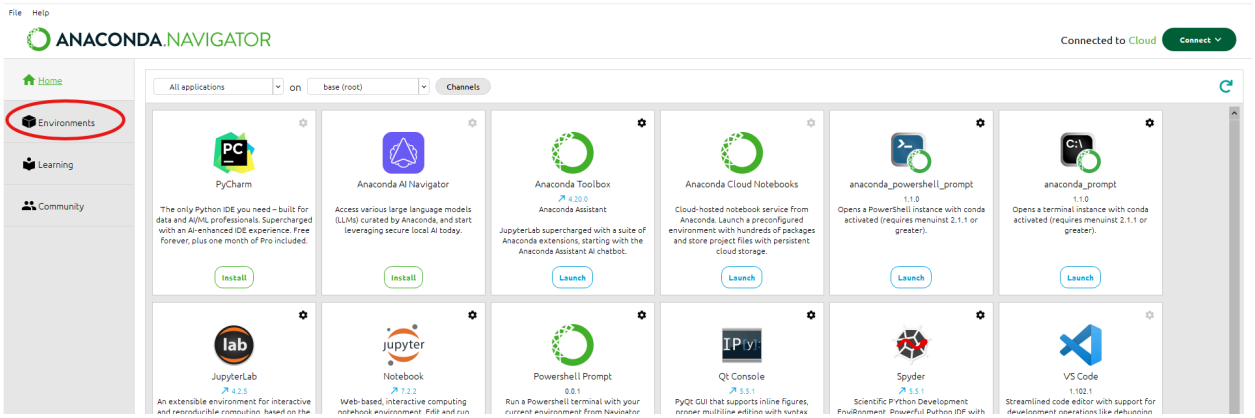

Figure 1

Once you are in the tab, create a new environment (see Figure 2 for reference).

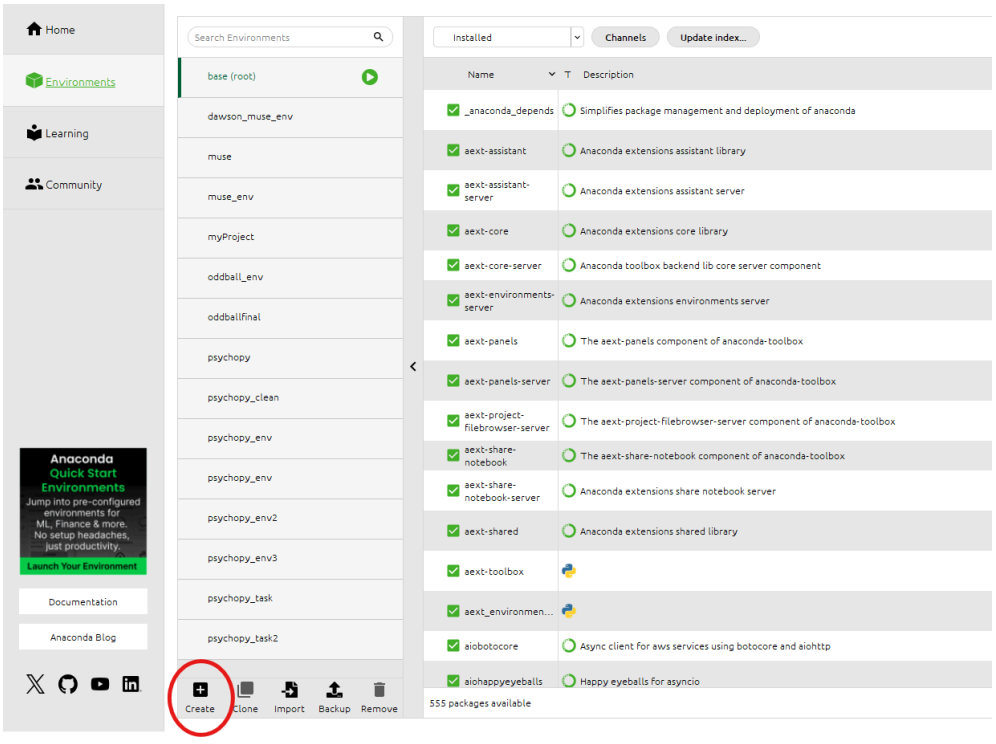

Figure 2

Name your environment, make sure to use version 3.10 of python, and now you are ready to use the MuseLSL package! (step 2.5 will explain how to install MuseLSL into your Anaconda Environment).

Tip: Let your Anaconda Environment open once you create it; you will need it for the following steps.

2.3 Open Python Terminal

The terminal is where we can run commands and install packages. To open a Python terminal using Anaconda, you need to go to Environments. Go to the environment made earlier and click on the green button (see Figure 3).

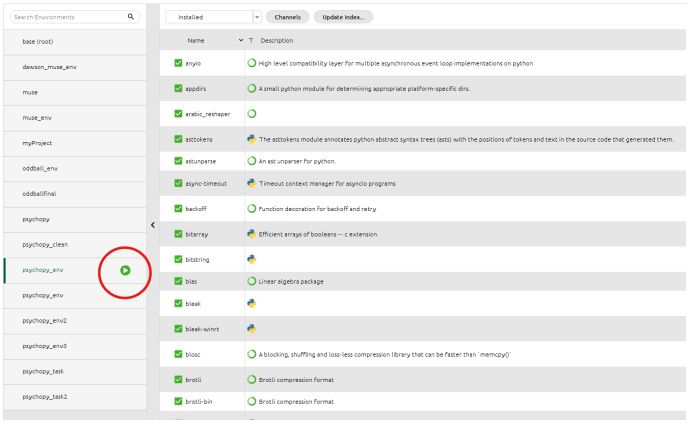

Figure 3

Once you click the button, several options will appear. You will need to choose:

- Open with Python Terminal

* Keep this step in mind, as you will need to open more terminals later.

2.4 PsychoPy:

PsychoPy is known for running neuroscientific experiments. If the program is not installed yet, proceed to https://www.psychopy.org/ and download the latest version. This will be the main focus of the experiment.

Install PsychoPy into your environment as well by running the following in the terminal:

- conda install -c conda-forge psychopy

2.5 MuseLSL:

MuseLSL is a Python package that will be necessary to stream, visualize, and record EEG data from the Muse Lab-Streamins-Layer. To install, you will need to open your previously created Anaconda environment and enter:

- pip install muselsl

This command should install the MuseLSL package into your Anaconda environment.

For additional info on the package, see its GitHub:

https://github.com/alexandrebarachant/muse-lsl

Or it's pypi (Python Package Index): https://pypi.org/project/muselsl/

3 Installation of Required Software for Integration in Psychopy Builder

IMPORTANT: This section is the installation process for integration of muse EEG in the builder in the psychopy app. If you are building a project in python instead, using the psychopy library python extension, skip this section, as the installations you need are the ones found in Section 2: Installation of required software for integration in python code, Creation of Anaconda Environment, PsychoPy & Open Python Terminal.

3.1 Install Psychopy

PsychoPy is a computer application known for running psychology and neuroscience experiments. If the program is not installed yet, proceed to https://www.psychopy.org/ and download the latest version.

3.2 Install Muselsl

Open a terminal on your computer. To do this on mac, go to spotlight search (the little magnifying glass on your toolbar, open the search, and enter "terminal" and the terminal will open for you. To open the terminal on Windows, you can use the Windows key + X shortcut, then select "Windows Terminal". Once your terminal is open, run the command:

pip install muselsl

And wait until the package is fully installed. For additional information on muselsl, view its github: https://github.com/alexandrebarachant/muse-lsl

The setup is now finished, and you are ready to move on to the experiment.

4 Setting Up an Experiment with Muse

To use Muse, you must add markers to your experiment via the Psychopy builder or python code. A marker notes down a timestamp- which is the specific moment down to the millisecond where an event occurs. We can use these markers to sync your experiment to the eeg data during the data analysis phase, allowing the creation of graphs and other representations of the data.

IMPORTANT: This section covers the process for both markers in python code, and in psychopy builder. If you are creating your project in the psychopy builder, follow the instructions in steps 4.1 and 4.2, and skip over 4.3, as that step is for integration in python. If you are instead programming your project fully in python, skip directly to step 4.3, disregarding steps 4.1 and 4.2.

4.1 Pick a Project In PsychoPy

For this section we will be using the default stroop task that comes pre-installed with psychopy as our example for simplicity, but you may select whatever project you want to integrate EEG with and open it. If you are using your own project, go straight to step 4.2.

To use the Stroop task, create a folder on your desktop and copy the default package into a new folder. Then, go to PsychoPy Builder and reselect the CSV of the words used in

the Stroop. To adjust the CSV location, click on the loop (trials) and select the new CSV from the correct folder. (see Figures 4 and 5)

Figure 4

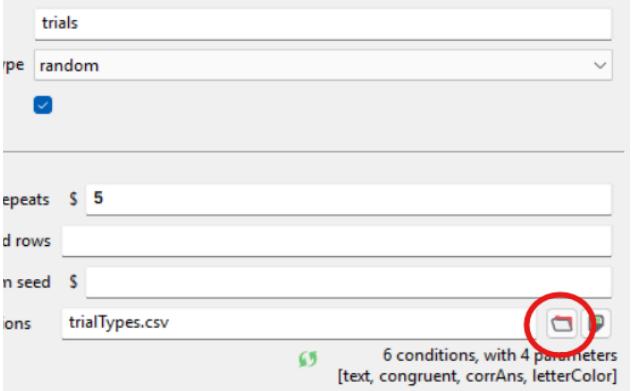

Figure 5

4.2 In-Builder Markers

In this step we will be adding the timestamp markers directly into the psychopy builder. Open the PsychoPy Builder. On the right side, under components, click the grey bar that says "custom", then open "code". Switch "auto→JS" to "py"

The following code will add a timestamp whenever a key is pressed. Copy paste the individual lines of code (without the dashes) into the sections denoted above them in bold (do not copy the bold text as that is just to indicate to you which section to put the parts of code in). Make sure the indentations remain the same:

- **In "begin experiment":**
- import time
- from datetime import datetime

- **In “Begin routine’:**
- timestamp_logged = False
- **In “Each frame”:**
- if resp.keys and not timestamp_logged:
- *# ISO format (easily readable time)*
- iso_timestamp = datetime.now().isoformat()
- thisExp.addData('response_timestamp_iso', iso_timestamp)
-
- *# Unix time (the one we use for plotting)*
- unix_timestamp_ms = time.time()
- thisExp.addData('Marker Timestamp', unix_timestamp_ms)
-
- timestamp_logged = True

Make sure the name of your keyboard response in the builder is “resp,” or this will not work. Alternatively, you can rewrite the line of code that says: if resp.keys and not timestamp_logged: and just replace “resp” with whatever the name of your keyboard responder is called.

Now your timestamps are being taken!

Alternatively, you can add code to take a marker at a certain point of the shown stimulus (image or text) instead of at the user's key response. In that case, your code should read:

- **In “Begin experiment”:**
- import time
- from datetime import datetime

- **In “Begin routine”:**
- stim_timestamp_logged = False
- **In “Each frame”:**
- if targetStim.status == STARTED and not stim_timestamp_logged:
- *# ISO timestamp (easily readable time)*
- display_timestamp_iso = datetime.now().isoformat()
- thisExp.addData('stimulus_timestamp_datetime', display_timestamp_iso)
-
- *# Unix time (the one we use for plotting)*
- display_timestamp_unix = time.time()
- thisExp.addData('Marker Timestamp', display_timestamp_unix)
-
- stim_timestamp_logged = True

Again, make sure your stimulus (image or text) is named targetStim, or if not, rewrite the line of code “if targetStim.status == STARTED and not stim_timestamp_logged:” and replace “targetStim” with the name of your stimulus in the builder.

4.3 In-Code Markers

This section is for placing timestamp markers (in unix time) into a python code. If you are instead integrating timestamps into the psychopy app, which was detailed in steps 4.1 and 4.2, skip this section. For this code, you will need to add it into your existing python file for your psychopy project. Open up your python project, ensuring it’s in the conda environment created in step 2.

At the beginning of your code, write:

```
from pylsl import StreamInfo, StreamOutlet
```

To set up the stream markers:

```
info = StreamInfo(name='Markers', type='Markers', channel_count=1,
            channel_format='string', source_id='marker_stream')
outlet = StreamOutlet(info)
```

To send EEG markers:

```
marker_timestamp = datetime.now().isoformat()
outlet.push_sample([stimulus_type])
```

To save trial data:

```
data_list.append([
   trial + 1,
   stimulus_type,
   response,
   response_time,
   is_correct,
   stimulus_type,
   marker_timestamp
])
```

To save data into a CSV file, where you can change “NAME_OF_EXPERIMENT” to the desired title:

```
with open("NAME_OF_EXPERIMENT", "w", newline="") as file:
    writer = csv.writer(file)
    writer.writerow(["Trial", "Stimulus", "Response", "Response Time", "Is response
correct", "Marker", "Marker Timestamp"])
    writer.writerows(data_list)
```

NOTE: "Trial", "Stimulus", "Response", "Response Time", "Is response correct", "Marker", "Marker Timestamp" are all things your code will likely be taking down if it's a psychopy experiment, but not necessarily. Add or remove these columns to adapt to your actual project. Keep in mind that the only two important for the EEG data are "Stimulus" (which should be taking down what the stimulus is) and "Marker Timestamp" (which is the unix timestamp we use to sync the project and the EEG data).

5 Stream, Visualize, Record

If you are using the method for integration in python, go to your Anaconda environment, once again, and follow the same instructions given at the beginning of 2.4 Open Python Terminal.

If you are integrating into psychopy, you may simply use one of your computer's built in terminals, as was done in step 3.2.

To stream MuseLSL, open a Terminal and type:

- muselsl stream

Make sure to turn on your Muse headset and ensure Bluetooth is working.

To visualize, open a new terminal and type:

- muselsl view

You will see something similar to Figure 6. (This is a muse reading that is just noise as it is not placed on someone's head. If your muse is on a person, you should have 4 colored lines).

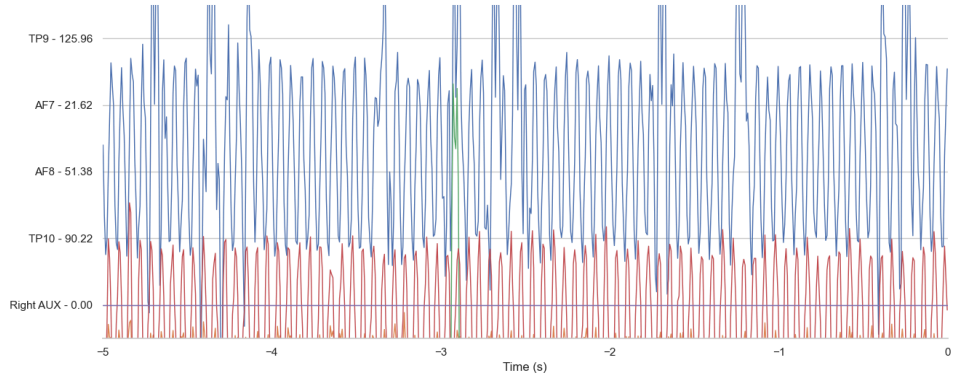

Figure 6

Finally, open a new terminal and type:

- muselsl record --duration (number of seconds you want it to record for)

E.g., muselsl record --duration 300 will record for 300 seconds (5 minutes). Make sure the time you set it to record for is longer than the time that your PsychoPy project will run for. It's better to record too much rather than not enough.

If it has started recording, it will show:

Start recording at time t=(n)

Time correction: (numbers)

Alternatively, for more control over the record time: Make the duration a very large number, like 7200 seconds (2 hours), then just press control+c in the terminal once you've recorded your full task duration. It will automatically cut the recording and save what's been recorded so far as a CSV file. This is the most convenient method.

Once the recording is done, the terminal will show the filepath to where it is saved. Run the following in the terminal:

- open (paste in the given file path)

It will open the data file. Save it under your project folder. The data should look something like Figure 7.

EEG_recording_2025-07-21-01.35.44

timestamps	TP9	AF7	AF8	TP10	Right AUX
1753061750.529	999.512	911.133	114.746	771.484	0.000
1753061750.533	504.883	652.832	-305.176	999.512	0.000
1753061750.537	-1000.000	-878.418	-909.180	-434.082	0.000
1753061750.541	-1000.000	-1000.000	-785.156	-1000.000	0.000
1753061750.545	440.918	163.086	-18.066	100.586	0.000
1753061750.549	999.512	999.512	-46.875	999.512	0.000
1753061750.553	-436.035	-276.367	-831.055	96.680	0.000
1753061750.557	-1000.000	-1000.000	-872.070	-1000.000	0.000
1753061750.561	-105.957	-217.773	-36.133	-250.000	0.000

Figure 7

Make sure to re-save your EEG data as a CSV, as it often saves as a numbers file. By the end of it, you will have two CSV files: PsychoPy with timestamps, and EEG.

6 Data Analysis

Analysis methods 1, 2, and 3 plot the raw data, whereas analysis methods 4 and 5 provide further noise cleaning and analysis to get data that is ready for use for research and projects.

For the data displayed as an example in this section, we used the data from a basic oddball task as well as the eeg readings taken from it to display in the analysis. The oddball task consisted of 20 shapes, 18 of them being common (“circle”) and 2 of them being the oddballs (“square”). So, anywhere in the following plots that says “square” or “circle” will be automatically replaced with the names of the stimulus from your experiment once you plug in your own CSV files into the codes.

The analysis will be broken down into several parts depending on the kind of analysis you will do. Take into consideration that your CSV files will need to follow a specific format. The EEG needs to have the same columns as in the image from Figure 7 (which is the default formatting for muselsl recordings). The PsychoPy CSV needs two specifically named columns that analysis programs will look at. The column with timestamps related to certain events needs to be named “Marker Timestamp.” The column with the event types (the event factors you're comparing) needs to be named "Stimulus”. For example, if you’re using an oddball task that compares reactions to circles (common) vs squares (oddball), your column titled Stimulus would have rows that say “circle” or “square” underneath them, and the programs would compare these two stimuli in the plots made.

If you’re doing an experiment comparing patients in different situations who are taking the same test, and you wish to compare the EEG of each patient, you should paste the different patients’ CSV results into one CSV file. Then, add a column titled “Stimulus” and label each patient depending on which initial group they were part of.

All code required for the analysis methods can be found in this GitHub repository: https://github.com/Jackalope-does-coding/Eeg-analysis-methods

Open the repository on GitHub, click the green “<> Code” button, and click “Download ZIP”. This will download all files from the repository into a folder. For simplicity, save your PsychoPy project and all the CSV data files into this same folder.

You will also need to have software installed on your computer to run Python files. In this manual, Jupyter Notebook was used to run the .ipynb files, and VS Code to run the .py files (Tip: Look at the file names to know which kind of file it is). This is the recommended way to run these programs. However, you could run both types of files from Jupyter Notebook or from VS code, you aren’t obligated to have both.

Here is the link to install VS Code (Visual Studio Code):

https://code.visualstudio.com/download

VS Code has the benefit of being easy to use for beginners and having better automatic debugging tools.

To install jupyter notebooks, see step 2.1 for the installation of Anaconda Navigator. Once Anaconda is installed, open the home tab, then launch jupyter notebooks. Jupyter notebooks has the benefit of being better for data analysis and being web-based.

IMPORTANT: for every analysis method program, you will need to open the code, find the place where the csv files are loaded in, and replace them with the file path to your own data csv files. There will be 2 of these lines in each code - one for each of your csv files. The line of code will look something like this:

```
eeg_df = pd.read_csv("/Users/username/Desktop/folder/eegfilename.csv")  #put the file path to your eeg csv here
```

And will be marked by a comment (a # then some instructions) telling you that this is the line of code where you need to replace the filepath. So for example, your new code might read:

```
eeg_df = pd.read_csv("/Users/sandranitchi/Desktop/MuseEegProject/eegdata.csv")  #put the file path to your eeg csv here
```

But with your name, folder name, file name, etc.

6.1 Analysis Method 1 - Simple Plotting

This displays the data as it is. It shows the entirety of the recorded time and its overlap with the PsychoPy CSV data, without cleaning it. From the GitHub folder, it's the file called "simple_plotting.py". This is a good place to start to make sure your data is visible and

overlapping correctly. Open this Python file, change the file paths to your CSV folders, and you're good to run it. This is what the result should look like (Figure 8):

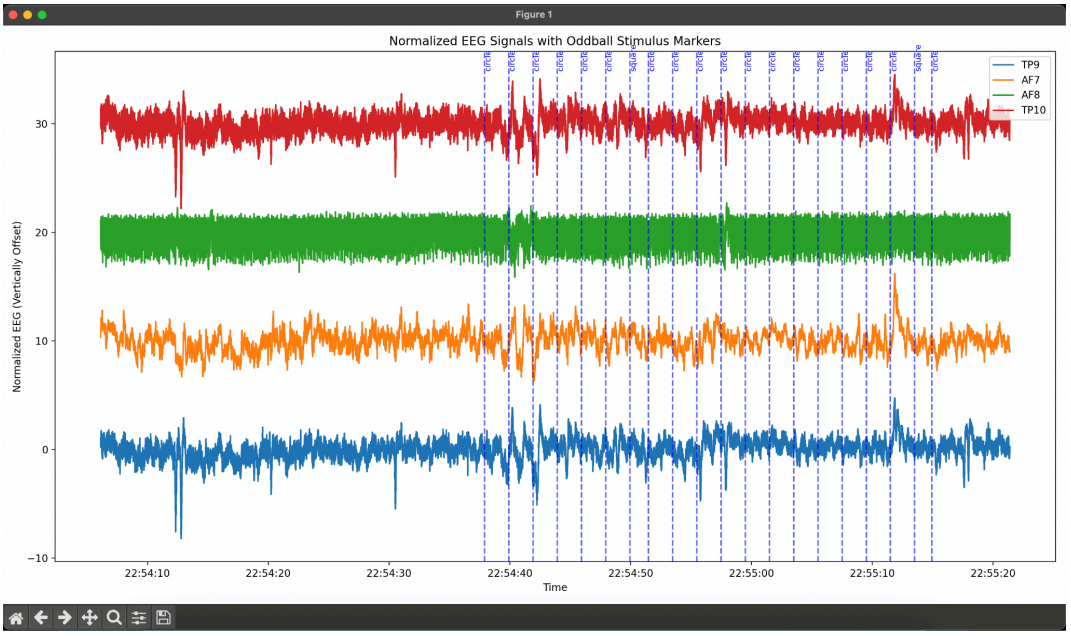

Figure 8

6.2 Analysis Method 2 - Experiment Plotting

This displays the zone where the EEG readings and the PsychoPy experiment overlap, which is the part that interests us. From the GitHub folder, it's the file called "experiment_plotting.py". This is a good code to run to generally visualize how the data looks. Open this Python file, change the file paths to your CSV folders, and run the code. (View Figure 9):

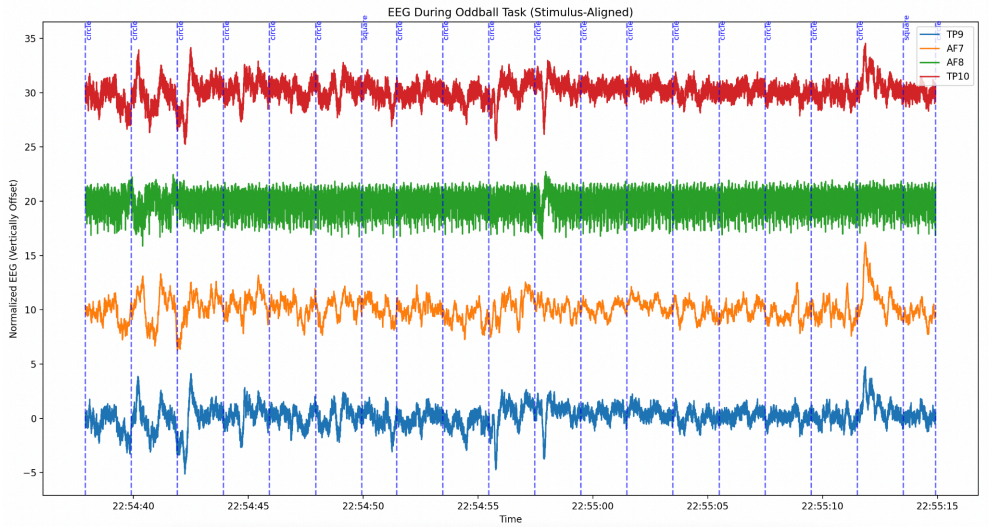

Figure 9

6.3 Analysis Method 3 - Stimulus Slices Plotting

This displays a small slice of the EEG reading around the point where a stimulus appears for every single stimulus shown. From the GitHub folder, it's the file called "stimulus_slices_plotting.py". This is good for observing the expected spikes and general changes in brainwaves when participants are shown a stimulus. Open this Python file, change the file paths to your CSV folders, and you're good to run it. The result should generate something that looks like the upcoming image (Figure 10), except it will generate one of those for every stimulus shown. After closing each one, the next one will come up until you've gone through all of them.

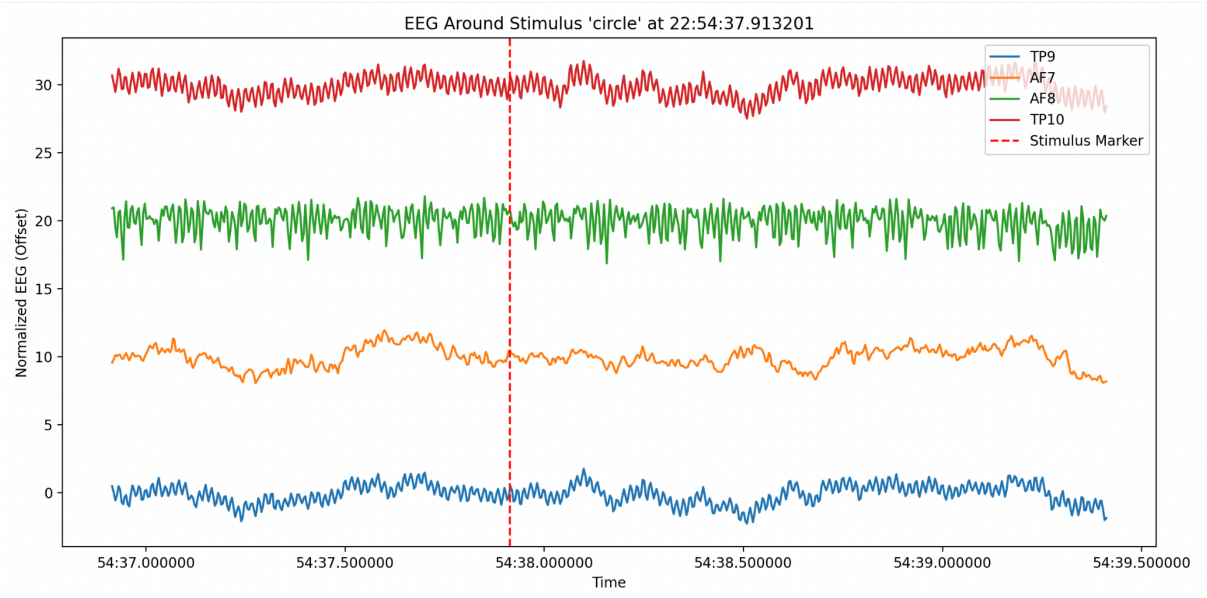

Figure 10

6.4 Analysis Method 4 - ERP Analysis

Now we are moving on to the analysis methods that clean and display data with formatting, which is easier to understand and use for studies and projects.

From the GitHub folder, it's the file called "ERP_cleaning_and_analysis.ipynb." Notice that this filename ends with .ipynb instead of .py. This is because it was made with Jupyter notebooks, so it's running with Python kernels (code that is broken into individual parts

called “cells”). This means that the cells can be run separately, so you can rerun the plotting without having the rerun the importations and installations. Replace the filepaths to your CSV files, and then run each kernel in order from top to bottom. We recommend running .ipynb files from Jupyter notebooks, but they can also be run from VS Code, Google Colab, or most other Python-running softwares. The results displayed will be four plots of cleaned EEG data, one for each electrode. Each plot will have one line displaying the average wave from this electrode for each different stimulus type. This is the Event-related potential, so showing each electrode's average reaction to the stimulus.. The result should be 4 plots that look similar to Figure 11.

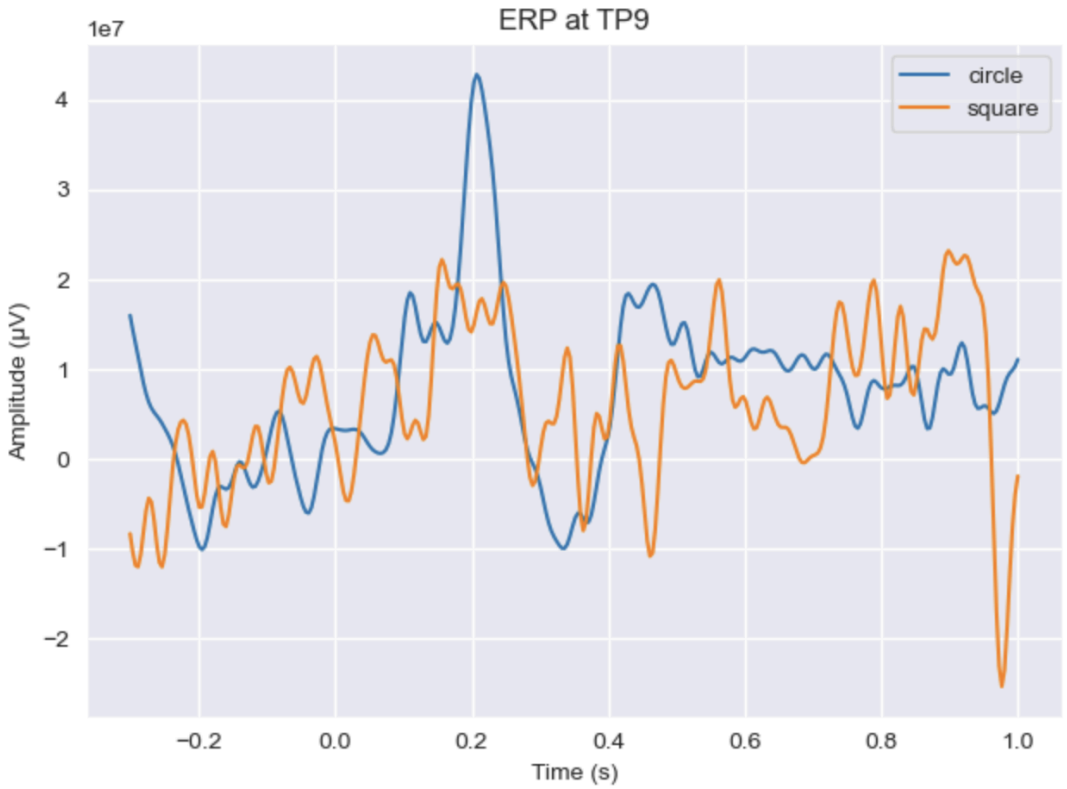

Figure 11

6.5 Analysis Method 5 - Bands Analysis

From the GitHub folder, it’s the file called “general_bands_cleaning_and_analysis.ipynb”. This is another .ipynb file, so the kernels need to be run in order to get the end result. This file displays three interesting results,

coming from running the final three kernels. The first is a display of the cleaning diagnostics, which indicate:

1) How many stimuli were initially inserted (make sure this number is correct so that you know for sure the code is processing your file correctly!)
2) How many of them did the code keep for analysis (trials that don't line up or have too much noise to clean are removed. This can happen if a testee is moving their head a lot or blinking right when the stimulus appears, as this can create large, unrelated spikes of noise that cover the actual data.
3) An average, for the channel of how many clean trials are likely to appear and be usable (you can ignore this part, as it is rarely relevant for research, and is more of a diagnostic matter for the code)

Look at Figure 12 for reference:

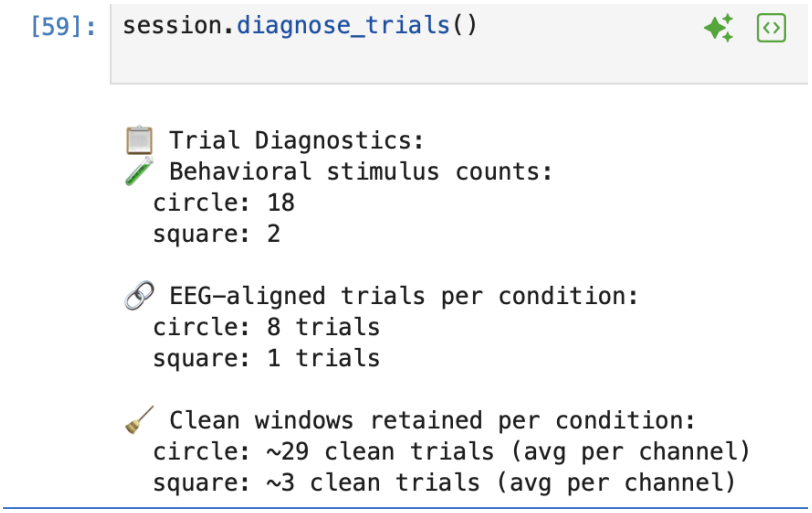

Figure 12

The second thing it displays is the dataframe, which is a dataset composed of the average band power for the four electrodes for each stimuli from the cleaned data! It takes the cleaned usable data and makes a power average to be able to compare the reactions of the different parts of the brain to the different stimuli.(It should be kept in mind that the Muse

EEG only has four electrodes; therefore, it doesn't provide a full complex view of the brain activity.) View Figure 13 to see how the produced dataframe should look:

```
[60]: session.dataframe()
```

[60]:

	Condition	Channel	delta	theta	alpha	beta
0	circle	TP9	0.750799	0.139015	0.057319	0.045783
1	circle	AF7	0.735414	0.143173	0.063864	0.053540
2	circle	AF8	0.047923	0.055226	0.065502	0.821540
3	circle	TP10	0.753173	0.132998	0.054003	0.051342
4	square	TP9	0.769600	0.144748	0.053195	0.047430
5	square	AF7	0.721238	0.161603	0.066864	0.060450
6	square	AF8	0.057969	0.064185	0.075549	0.794115
7	square	TP10	0.770306	0.135791	0.050984	0.053812

Figure 13

Finally, the plotting of the power bands as a Welch periodogram will help with the visual representation of the electrode data, which allows you to see which part of the brain was the most active during each stimulus. This code will create one graph for each of the stimulus types representing the average power bands (see Figure 14).

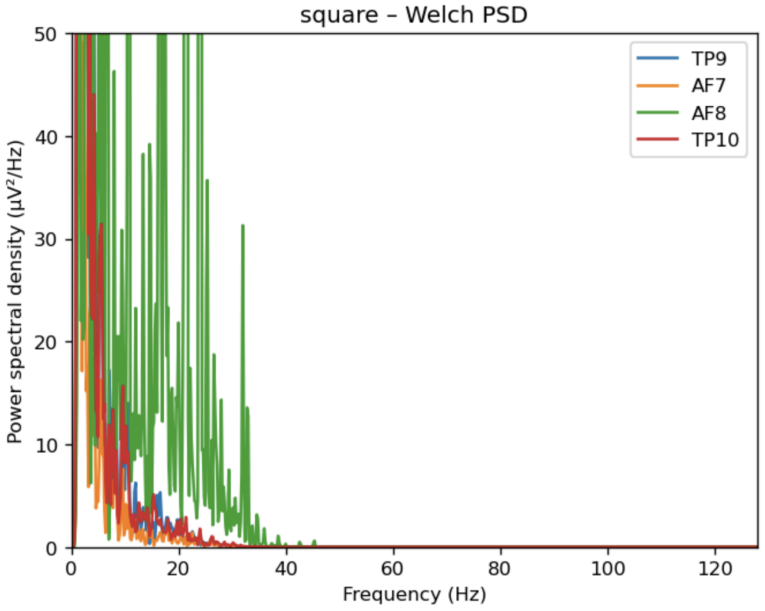

Figure 14

The threshold for which waves are being excluded is currently set to 200, but you should change this as you see fit for your study. Considering the muse is imprecise and very prone to noise, you should ideally lower the threshold as much as possible for the best results. To do that, change the following line of code:

```
self._clean_windows(threshold=200) # CHANGE THE THRESHOLD NUMBER DEPENDING ON HOW EXACT YOU WANT THE RESULTS
```

So that the threshold number (in this case 200) is lower.

7 Some Technical Issues You Might Run Into And How to Fix Them

7.1 Conda not recognized as a command (Windows):

If you encounter an issue saying that conda is not recognized as a command while trying to run the experiment, then it simply means that you need to add Anaconda to your PATH, which most likely means you answered ‘no’ to the question in 2.1 Anaconda Navigator. The following instructions will focus on Windows; for Mac or Linux, go to page 22.

First, you want to press Windows + S simultaneously.

Next, type "variables" (the correct system will show whether your laptop is in English or French)

You will see a System Properties window as shown in Figure 15. Click on the "**Advanced System Settings**" tab (Paramètres systèmes avancés in French)

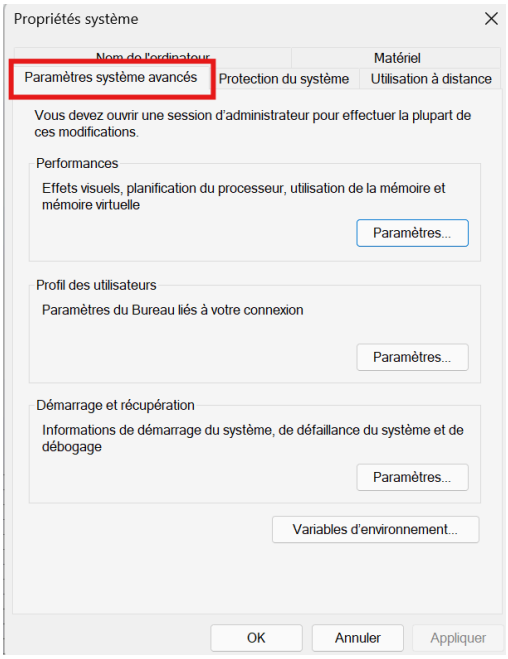

Figure 15

Click on the "**Advanced System Settings**" tab (Paramètres systèmes avancés) (refer to Figure 15 once more)

Head to "**Environment Variables…**" (Variables d'envrionement…) (Figure 16)

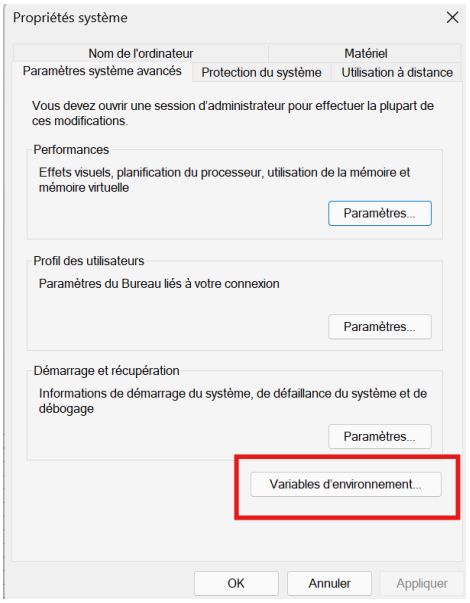

Figure 16

Look under the section titled "System Variables" (Variables système) and find the one called "**Path**" (Figure 17)

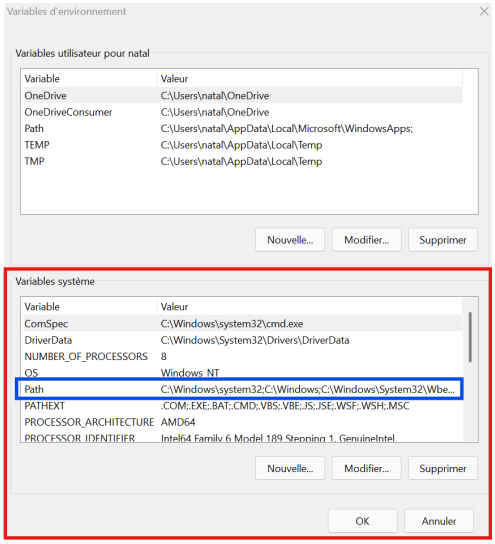

Figure 17

Click on it and go to "new" and paste the following one by one (replace YourUsername with your actual Windows username):

C:\Users\YourUsername\Anaconda3

C:\Users\YourUsername\Anaconda3\Scripts

Select ok, then ok again to exit the Environment Variables.

This should solve the problem.

7.2 Conda not recognized as a command (Linux/macOS):

The following instructions outline how to add Anaconda to your Path on Linux or macOS.

First, open a terminal. If you are unsure of how to proceed, you can click the Launchpad icon in the Dock and then type "Terminal" in the search bar.

Then, open your shell configuration file.

Add:

export PATH="/path/to/anaconda/bin:$PATH"

* Replace /path/to/anaconda/ with your actual Anaconda installation path.

Make sure to save the changes.

8 Credits

This project is a collaboration between Sandra Nitchi, Lisa Zubkova, and Natali Petrosyan

This project's work was overseen by Toky Raharison Ralambomihanta

This project was organized by Prof. Helene Nadeau and Prof. Sylvia Cox

This project was completed at Dawson College, Montreal, summer of 2025

For additional info, feel free to contact any/all of the members of this team:

Sandra Nitchi: sandranitchi@gmail.com

Lisa Zubkova: alisa.zubkova@mail.mcgill.ca

Natali Petrosyan: natalipetrosyan135@gmail.com

Toky Raharison Ralambomihanta: tokiniaina.raharisonralambomihan@mail.mcgill.ca

Dr Helene Nadeau: HNadeau@dawsoncollege.qc.ca

Dr Sylvia Cox: scox@dawsoncollege.qc.ca

This work was supported by Dawson College and the CARE Internship Initiative, made possible by funding from the Tri-Agency College Community Innovation (CCI) program for Mobilize grant.

References

Jackalope-Does-Coding. (n.d.). *Basic-oddball-task-psychopy: These codes are reliant on separately recording EEG data using muselsl, before analyzing it together* [Computer software]. GitHub. https://github.com/Jackalope-does-coding/Basic-oddball-task-psychopy

Kowalej. (n.d.). *BlueMuse: Windows 10 app to stream data from Muse EEG headsets via LSL (Lab Streaming Layer)* [Computer software]. GitHub.https://github.com/kowalej/BlueMuse

Raharison Ralambomihanta, T. (2022, March 17). *Running psychopy experiments with the Muse* [PDF document]. Retrieved from https://drive.google.com/file/d/1mfNe7fA740PfIYRPVO7L4lQCO6P_x_M0/view

Raharison-Toky. (n.d.). *Dawsonmuse: A module for running EEG experiments with Psychopy and a Muse device* [Computer software]. GitHub. https://github.com/raharison-toky/Dawsonmuse