SPIKE

COMPUTATIONAL NEUROSCIENCE COMMUNITY

Spike Academy

Course: EEG Processing: An Entry to the World of Brain Waves

Home Work 5:

Topic: ERP and Brain Topomaps

Spring 2025

Prepared By:

Mohammadreza Shahsavari

1. Homework Overview: Event-Related EEG Data Analysis

In this homework, we'll guide you through the practical steps of working with real-world, event-related EEG data using the powerful MNE-Python library. Step-by-step, you'll begin by loading your EEG dataset into Python. Then, we'll dive into basic preprocessing techniques to clean up the data. This involves removing non-EEG channels, applying filters to eliminate non-brain activity, and resampling the data if necessary to optimize computations.

Once your data has undergone initial preprocessing, you'll learn how to calculate and visualize the brain's response to specific events. This involves computing and plotting Event-Related Potentials (ERPs) and creating corresponding brain topomaps. Critically examining these plots is the next step; you'll assess the quality of your EEG data by looking for clear patterns and identifying any obvious signs of noise or artifacts that might suggest the data isn't clean enough yet.

It's quite common for the initial results to need further refinement. If needed, you have to get back to the preprocessing stage to apply more advanced techniques. This might include interpolating data for bad channels or removing specific trials (epochs) that are too noisy. This iterative cycle of preprocessing, plotting, and assessing is fundamental. You'll continue this process until you successfully obtain clean EEG data of acceptable quality, evidenced by clear, interpretable ERPs and reasonable topomaps. This hands-on experience will build your skills in achieving reliable EEG analysis results.

2. Data

The EEG data that you are going to use in this home work are these files:

1. `901Goalie_Spring2024.eeg`
2. `901Goalie_Spring2024.vhdr`
3. `901Goalie_Spring2024.vmrk`
4. `EEG Event Markers.xlsx`
5. `TrialInfo.csv`

You can find and download these data from this link.

The three files with names “`901Goalie_Spring2024`” are the primary EEG data obtained from a real-world EEG project. Permission has been granted for these files to be used specifically for this homework assignment, with certain considerations.

The naming convention of these EEG files contains important details about the project and recording session. The project is titled “Goalie EEG,” and the number “901” serves as the unique subject identifier. Within this identifier, the first digit (9) refers to the operator ID—the person who recorded this session. The remaining numbers (01 in this example) indicate the subject ID assigned to the individual being recorded by this specific operator. Together, these numbers form the overall subject ID: 901. This format includes both the subject’s unique identifier and the operator’s information.

The “Spring2024” tag in the filename specifies the season and year during which the recording was performed.

This naming convention is used by the lab to systematically document their data. While labs may use different conventions to name EEG files, it's a common practice to include information that conveys the subject ID, operator ID, session details, and recording timeline.

Here are some more information about each of these three main EEG data:

- `901Goalie_Spring2024.vhdr`:
  - The header file containing metadata like electrode names, sampling frequency, and other parameters
  - This is the primary file you must introduce to MNE-Python to load the entire dataset
- `901Goalie_Spring2024.eeg`:
  - The largest file in terms of storage size
  - Contains the actual EEG signal array (raw recordings)
- `901Goalie_Spring2024.vmrk`:
  - Contains the marker (trigger) data
  - Stores event timestamps and experimental condition codes

Note: When using MNE-Python, you only need to specify the path to the `.vhdr` file, and it will automatically locate and load the associated `.eeg` and `.vmrk` files.

In addition to the three main files mentioned above there are two other meta data files which although do not contain the EEG dataitse lf, they include some useful information. More specifically:

- `EEG Event Markers.xlsx :`
  - includes the trigger codes and their meanings.
  - By this file we can know what each trigger code means and therefore we can define our trials.
- `TrialInfo.csv:`
  - includes trial type labels and bad trial labels.
  - Using trial type labels we can create ERPs for each trial type

- Using bad trial labels we can exclude some trials which has been previously detected as noise.

We will learn more about triggers and trials in the rest of this home work.

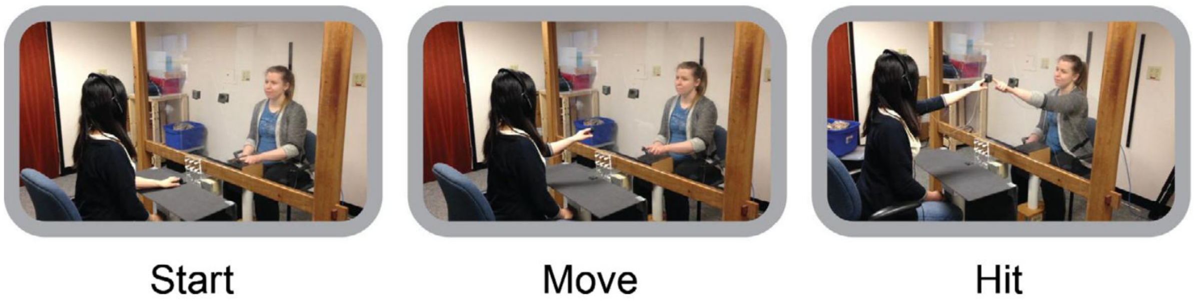
Data Explanation

This EEG data was collected from a subject participating in a two-player game designed to study social interaction and action prediction. The two players in the game are the Attacker and the Blocker, separated by a glass window with two buttons attached to it—one on the left and one on the right. The Attacker's goal is to choose and press one of the buttons, while the Blocker's task is to observe the Attacker and press the same button as quickly as possible after the Attacker makes their move.

During the game, EEG recordings were taken from the Blocker to analyze their brain activity. The primary goal of the study is to investigate whether the Blocker's brain signals contain predictive information about the Attacker's upcoming move—potentially even before the Attacker begins to move their finger toward one of the buttons. This analysis could provide insights into the human ability to predict the actions of others in real-time social interactions.

Understanding action prediction is crucial for explaining how humans navigate complex social environments. For instance, when walking down a crowded street, people rarely collide because they seamlessly predict others' movements and adjust their paths accordingly. Similarly, predictions are made when a barista hands you a coffee or when you shake hands with

a colleague. These abilities to anticipate actions are fundamental to smooth and efficient social interactions, and this study aims to explore the neural mechanisms behind them.
The image below illustrates the experimental setup of the game.



The primary objective of this homework assignment is to help you develop essential skills in EEG data analysis that will serve as the building blocks for more advanced work later in the project.
In this first step, you will familiarize yourself with the EEG dataset collected from the Blocker during the action prediction game, and learn the fundamentals of data preprocessing using Python. This involves cleaning the raw EEG data by applying standard techniques like filtering and artifact removal—crucial steps that ensure your data is reliable and ready for meaningful analysis.
Once your data is preprocessed, you will move on to visualizing neural activity by plotting ERPs and brain topomaps. These methods are not only widely used in EEG research to interpret brain activity, but they are also useful tools for identifying patterns and potential sources of noise in your recordings. Although the main focus of ERPs and topomaps is to visualize neural responses, they also serve as valuable diagnostic tools for assessing data quality and guiding further cleaning if necessary.

Most importantly, the skills you acquire in this homework—such as effective data cleaning, basic EEG signal processing, and visualization—will be directly applicable and immensely valuable in subsequent stages of the project. As you progress to more complex analyses, such as decoding the Blocker's decision to move from their neural signals, a solid foundation in these basic techniques will be indispensable. Mastering these initial steps will enable you to approach EEG decoding for action prediction with confidence, ensuring that your analyses are both accurate and insightful.

Load EEG

First, you need to load and inspect your EEG data to understand its basic properties. When working with EEG data, two key pieces of information are essential: 1- `raw.info`, which provides metadata about the recording (such as channel names and sampling rate), and 2 - event markers with their meanings, which are crucial for analyzing event-related (not resting-state) data. Both are fundamental for accurate EEG analysis.

`raw.info`

The `raw.info` attribute in MNE contains metadata about the recording and the acquisition system. It includes details such as:

1. Sampling Frequency (`sfreq`):
   - The rate at which data was recorded (in Hz).
   - Example: 500 Hz means 500 samples per second.
2. Number of Channels (`nchan`):
   - Total number of sensors/channels in the dataset.
3. Channel Types (`chs`):
   - Information about individual channels (e.g., EEG, MEG, EOG, EMG).

4. Measurement Date (`meas_date`):
    - The date and time of the recording.
5. Bad Channels (`bads`):
    - List of channels marked as "bad" (e.g., due to noise or artifacts).
6. High-Pass and Low-Pass Filters (`highpass, lowpass`):
    - Frequency filtering applied during acquisition.
7. Coordinate System (`dig`):
    - Information about the spatial positions of sensors (if available).

Here is an example:

```python
import mne

# Load BrainVision data (.vhdr file)
raw = mne.io.read_raw_brainvision('901Goalie_Spring2024.vhdr', preload=True)

# Access raw.info metadata
info = raw.info

# Example structure of `info`:
# info = {
#   'sfreq': 2000.0,
#   'nchan': 32,
#   'chs': [
#       {'ch_name': 'EEG 001', 'kind': 2, 'unit': 'V', ...},
#       {'ch_name': 'EOG 002', 'kind': 202, 'unit': 'V', ...}
#   ],
#   'meas_date': datetime.datetime(2025, 4, 23, 14, 0),
#   'bads': ['EEG 001'],
#   'highpass': 0.1,
#   'lowpass': 100.0,
#   ...
# }
```

When you plot the EEG with Event marker you will find each event named is a code (like 226, 227 , etc). Each code has a meaning and related to some real event occurred during the game being played.

You can find the event code meanings in `‘EEG Event Markers.xlsx’`. for example here we have these events with their respective event codes.

Event	Event Codes
fixation Cross Started	226
fixation Cross Repeated (500 ms)	227
New Game Round Starts	230
First Movement	235
Screen Touch	238
Finger Rest Initial	244
Finger in rest position	245

Here is a brief explanation of each event and its respective event code:

Fixation Cross Started (226): This event marks the beginning of the display of a fixation cross on the screen, which helps the participant focus before the game round begins.

Fixation Cross Repeated (500 ms) (227): This event indicates that the fixation cross is displayed again for a duration of 500 ms, reinforcing focus before the start of the game.

New Game Round Starts (230): This event signals the start of a new round in the game, where the Attacker and Blocker prepare for the next interaction.

First Movement (235): This event marks the initiation of the first movement by the Blocker towards one of the buttons.

Screen Touch (238): This event captures the moment the Blocker touches the screen (button), completing the action.

Finger Rest Initial (244): This event indicates that Blocker hands is returning to the rest position.

Finger in Rest Position (245): This event indicates the Blocker's finger returning to the rest position after completing an action, preparing for the next round.

The rows highlighted in yellow and green represent a complete game round, which constitutes a full trial. These rows are particularly important for calculating the ERP in the subsequent steps.

Use `mne.events_from_annotations(raw)` to extract the event codes and the corresponding time points at which they occurred.

When using `mne.events_from_annotations(raw)`, you'll get:

- An events array with shape (n_events, 3) where:
  - Column 0: Sample index (time point) where the event occurred
  - Column 1: Unused. You can ignore this colum (typically 0)
  - Column 2: Event ID (numeric code for the event type)
- A dictionary mapping event names to their numeric IDs

```
# Extract events and event IDs from annotations
events, event_id = mne.events_from_annotations(raw)


# Output structures:
# 'events' is a NumPy array with shape (n_events, 3):
#    Column 0: Sample index (time point) where the event occurred
#    Column 1: Unused (typically 0)
#    Column 2: Event ID (numeric code for the event type)

# Example structure:
# events = array([[  1000,     0,     1],
#                   [  2000,      0,      2],
#                   [  3000,      0,      3]])

# 'event_id' is a dictionary mapping event names to numeric IDs:
# Example structure:
# event_id = {'Stimulus/S1': 1, 'Stimulus/S2': 2, 'Stimulus/S3': 3}
```

Tasks:

1. Use `mne.io.read_raw_brainvision()` to load the BrainVision EEG data (`901Goalie_Spring2024.vhdr`)
2. Print basic information about the data (sampling rate, number of channels, duration) and report it.
3. Plot a 20 seconds segment of the raw EEG data to visually inspect it.
4. Load trial information from '`TrialInfo.csv`'
5. Extract events from the EEG data using `mne.events_from_annotations(raw)`
6. Plot the events to understand their distribution throughout the recording

Basic Preprocessing

EEG data typically requires several preprocessing steps to remove artifacts and noise before analysis.

Tasks:

1. Plot the Power Spectral Density (PSD) of the raw data to identify frequency characteristics
2. Remove non-EEG channels ('Corr', 'Left Ear')

3. Apply a bandpass filter between 0.1 and 100 Hz
4. Examine the PSD after bandpass filtering to identify electrical line noise frequency (typically 50 or 60 Hz)
5. Apply a notch filter at the identified line noise frequency
6. Plot the PSD after all filtering steps to verify improvements
7. Determine an appropriate sampling rate for downsampling and explain your choice
8. Resample the data to your chosen sampling rate
9. Plot a segment of the preprocessed data to compare with the raw data

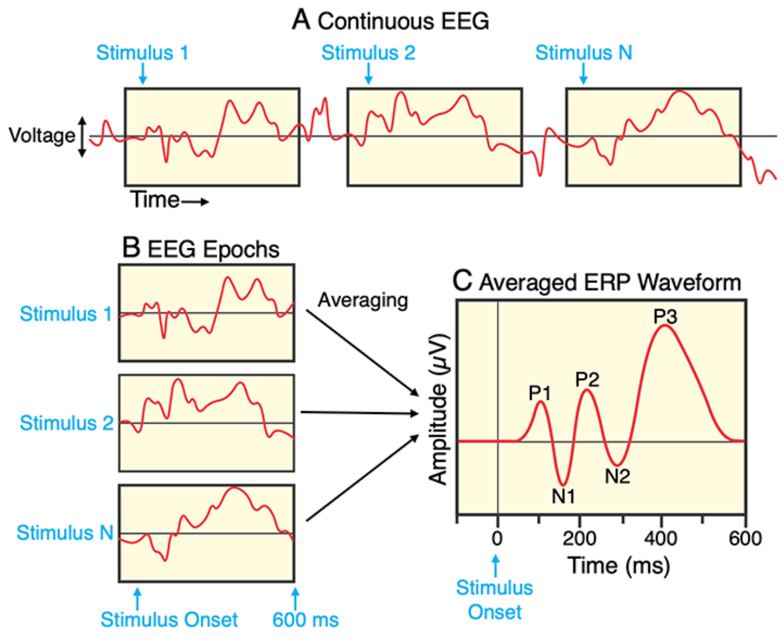
Understanding Event-Related Potentials (ERPs)

Event-Related Potentials (ERPs) are small voltage changes in the EEG signal that occur in response to specific events or stimuli. They represent the brain's electrical activity that is time-locked to sensory, cognitive, or motor events.

ERPs are important because:

1. They provide high temporal resolution (millisecond precision) of neural processes
2. They can reveal cognitive processing that may not be evident in behavior
3. They allow us to track the time course of information processing in the brain
4. They can be used to study various cognitive functions like attention, memory, and language processing

However, individual ERP responses are typically too small to be visible in the raw EEG, which contains ongoing brain activity unrelated to the event of interest. To extract the ERP signal, we average multiple trials time-locked to the same event type. This averaging process enhances the signal-to-noise ratio by reducing random background activity while preserving the consistent event-related response.
This diagram illustrates how trials from a continuous EEG are extracted and averaged to obtain denoised ERP.



Let's break down each panel of this image:

A. Continuous EEG

- This is the raw EEG signal recorded over time while a subject is repeatedly presented with the same stimulus (e.g., a the move of the attacker as a visual stimuli in our case).
- Each stimulus presentation (Stimulus 1, Stimulus 2, ... Stimulus N) is marked with a blue arrow.

- The yellow-highlighted sections represent *trials*, or short segments of EEG data, that are time-locked to the onset of each stimulus.

B. EEG Trials

- Here, the EEG segments from panel A have been extracted around each stimulus onset, forming individual trials.
- Each trial is typically time-locked to the stimulus onset and has a fixed duration (in this example, 600 ms).
- These segments still contain both signal (brain responses to the stimulus) and noise (random brain activity unrelated to the stimulus).

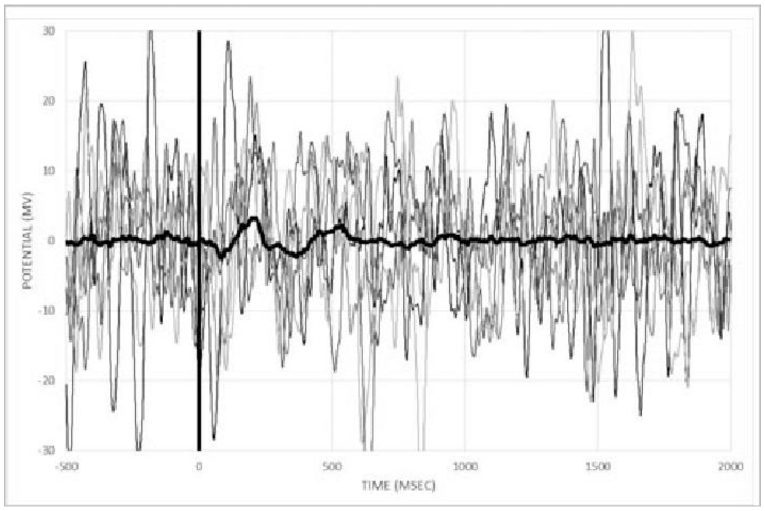
Averaging Process

- All the extracted EEG epochs are averaged together point-by-point.
- The idea is that the random noise varies from trial to trial and will cancel out during averaging, while the brain's response to the stimulus—if consistent—will remain and become clearer.

C. Averaged ERP Waveform

- The result of averaging is a cleaner waveform, called an Event-Related Potential (ERP).
- This ERP waveform reveals consistent patterns in brain activity that are time-locked to the stimulus.
- Common ERP components are labeled:
  - **P1, P2, P3**: Positive peaks (P stands for "positive")
  - **N1, N2**: Negative peaks (N stands for "negative")
- The time (in milliseconds) on the x-axis shows how long after stimulus onset each component occurs.
- Interpretation of these waveforms depend of the nature of stimuli and is not our interest in this homework assignment.

The following image illustrates the ho the denoising through averaging works:



Calculating ERPs from EEG Data

Now you'll calculate ERPs by manually extracting and averaging trials between event codes 230 (start) and 245 (end).

Tasks:

1. Define event codes for trial extraction (230 for start, 245 for end)
2. Find all occurrences of these events in your data
3. For each start event (code 230):
   - Find the first corresponding end event (code 245) coming after.
   - Extract the EEG data between these events, including a peristimulus period. (use `peristimulus = 0.5s`)
   - Categorize the trial as *left* or *right* based on `TrialInfo.csv`
   - Only include trials marked as "good" (`GoodTrial = 1`)
4. Since trials may have different lengths, pad all trials to the maximum length using NaN values

5. Calculate the ERP for left and right trials by averaging across trials using `np.nanmean()`
6. Calculate the standard error of mean (SEM) for each time point
7. Plot the ERPs with error bars for left and right trials

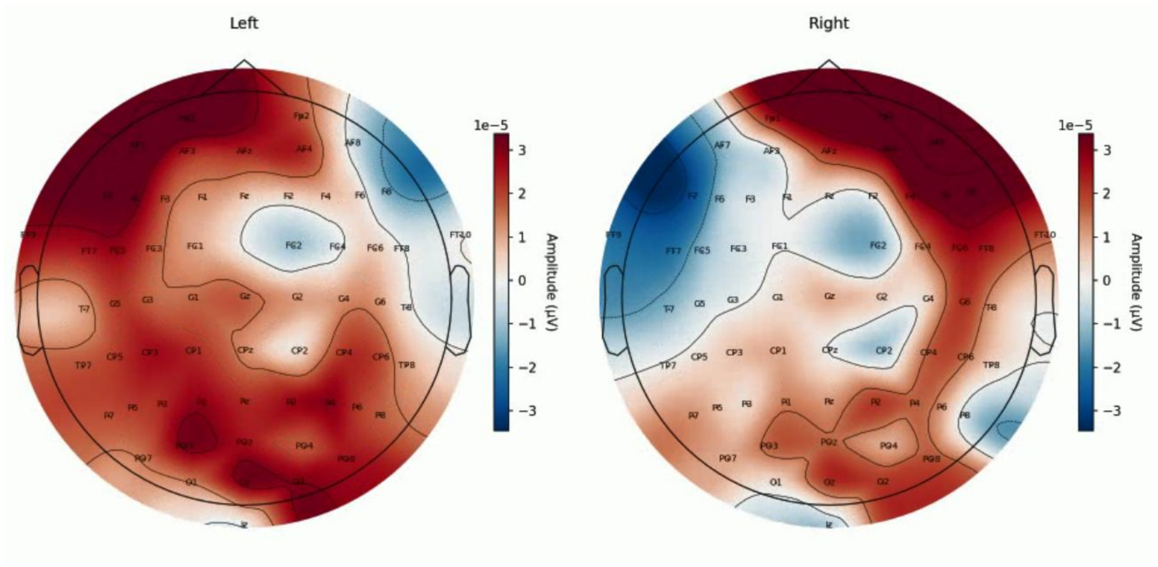
Questions to address in your report:

- How many good trials did you extract for left and right conditions?
- What does the peristimulus period represent and why is it important to include?
- What differences do you observe between left and right trial ERPs?
- Are the error bars narrow or wide, and what does this tell you about the data quality?

Understanding Topographic Maps of the Brain

Topographic maps (topomaps) are graphical representations of EEG activity distributed across the scalp. They provide spatial information about brain activity by mapping the electrical potentials measured at various electrode locations to the corresponding positions on the scalp. This visualization helps researchers and clinicians understand how different brain regions contribute to specific processes or tasks.

Below is an image of two topomaps generated from the EEG data in this homework, showing that when the attacker hits different buttons (Left vs. Right), different hemispheres of the blocker's brain are activated.



Why Are Topographic Maps Important?

1. **Spatial Insight**: Topomaps allow you to analyze activity patterns across different regions of the brain, revealing localized changes in neural dynamics.
2. **Time-Evolving Activity**: By generating topomaps at successive time points, you can observe how brain activity evolves in response to stimuli or events, making it easier to identify critical moments of activation.
3. **Comparison Across Conditions**: Topomaps enable comparisons between experimental conditions (e.g., "Left" vs. "Right" trials) or groups (e.g., healthy subjects vs. patients).
4. **Artifact Detection**: Anomalies in topomaps can indicate the presence of bad channels or artifacts, which need to be addressed during preprocessing.

How Are Topographic Maps Created?

Topomaps are generated by interpolating EEG activity measured at electrode positions across the scalp. The interpolation is based on the

spatial layout of electrodes, typically defined by the international 10-20 system. Each electrode contributes to the visualization based on its recorded signal amplitude.
Key steps to create a topomap:

1. **Input EEG Data**: Provide EEG data averaged across trials (e.g., ERP data) for a specific time point.
2. **Electrode Positions (Channel Locations)**: Use the electrode layout (montage) to define spatial positions on the scalp.
3. **Interpolation**: Perform spatial interpolation to estimate activity between electrodes.
4. **Color Mapping**: Map the interpolated values to a color scale for easy visualization of activity intensity.

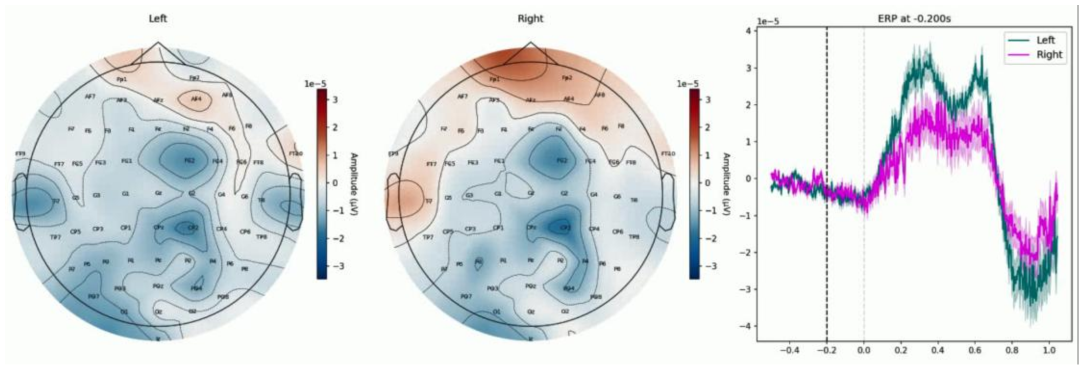
Time-Evolving Topomaps
Creating a sequence of topomaps at different time points allows you to visualize how brain activity changes over time. This is particularly useful for event-related EEG studies, where activity is analyzed relative to a stimulus or event. By overlaying these maps onto a timeline, you can see when and where specific brain regions activate, providing insights into neural processes such as attention, perception, or motor control.

Creating Topographic Maps and Video

Tasks:

1. Apply a standard 10-20 montage to your EEG data for topomap visualization
2. Define time points for topomaps based on a specified time resolution
3. For each time point:
    - Create a figure with three panels:
        - Left panel: Topomap for left trial's ERPs at the current time point
        - Middle panel: Topomap for right trial's ERPs at the current time point
        - Right panel: ERP plot with a vertical line indicating the current time point. Average all the channels of ERP and plot it in this pannel along with error bar.
    - NOTE: use a global min and max value in `vlim` parameter of `plot_topomap` function to be consistent about colors meaning during the time.
    - Save each figure as a frame
4. Compile all frames into a video showing the evolution of brain activity over time

Here is how a single from should look like at `t=-0.2`:



The general structure of the three panels in your output should exactly look like the figure above, but the ERP and brain activity distribution over brain region may be different from this image. Your results should contain irregular electrode activity and very wide error bar which is an indication of presence of noise or bad channel.

Identifying and Handling Bad Channels

After examining your topographic maps and ERP plots, you may notice channels that show unusual activity patterns. These could be "bad channels" that need to be addressed.

Signs of bad channels include:

1. Extreme voltage values compared to neighboring channels
2. Flat lines (no signal)
3. Excessive noise or spikes
4. Unusual patterns that don't match surrounding channels

Tasks:

1. Examine your topomaps and ERP plots to identify any potential bad channels
2. If you find bad channels, mark them in your data using `raw.info['bads']`
3. Interpolate bad channels using data from surrounding channels
4. Recalculate ERPs and regenerate topomaps with the interpolated data
5. Compare before and after interpolation to assess improvements

If you see spikes in the output ERP that saturate the figure, you should check the change intervals of each trial (channel by channel):

1. Calculate the change interval (max - min amplitude) for each channel in each trial
2. Determine normal bounds for these intervals using median and IQR
3. Identify trials with abnormal change intervals
4. Replace anomalous channels with NaN values
5. Recalculate ERPs with the cleaned data

Questions to address in your report:

- Did you identify any bad channels in your data? If so, which ones and why did you classify them as bad?
- Did you detect any trials with anomalous change intervals? How many channels/trials were affected?
- How did interpolation or anomaly handling affect your ERPs and topomaps?
- What improvements did you observe after addressing data quality issues?

Here are two topomap videos for a different subject, recorded before and after bad channel detection and interpolation. Note that this is not subject 901, so the activity patterns may differ (Link to the Topomap Videos).