1. **Collect EEG data**
2. **Import into MATLAB**
3. **Import event markers and channel locations**
4. **Re-reference / down sample (potentially)**
5. **High-pass filter**
6. **Examine raw data at this point**
7. **Identify or reject bad channels (bad connections)**
8. **Reject large artefact time points**
9. **Run ICA (independent component analysis) (powerful and widely used) and reject components. (this stage is likely where the various types of pre-processing will take place) (probably)**

Start MATLAB

Add eeglab to the path (don’t add sub path) (or just run it as you have been from the .mat file)

File -> manage plugins -> (add relevant plugins as necessary)

File -> Import data -> (find the format that you’re interested in) (CNT for the original dataset that was supplied) (the events appear to be included – otherwise this would be imported next from: import event info -> from data channel (last channel is usually event channel)

Or …

File -> Import Existing dataset -> (.set format file)

Plot -> Channel data -> Display -> remove DC offset (this wants to be done before filtering so as to reduce the creation of artefacts at the beginning and end of the signal, because the signal timings are offset from each other).

Edit -> channel locations -> plot 2-D or plot 3-D

Tools -> re-reference data -> re-reference data to channels -> save as new copy -> select the two channels

Or …

Tools -> re-reference data -> computer average referencing -> save as new copy

(sum of current is 0) the positive and negative across the entire head should balance and become 0, this depends upon the distribution of electrodes. This is enforces / shown via average referencing.

Resampling the data is an option to reduce the time and space complexities

Tools -> change sampling rate -> it was 1000Hz -> 256? What is best here

Purpose of the high-pass filter is to remove slow, or possibly large amplitude drift

Tools -> filter the data -> Basic FIR filter

High pass is required for ICA (independent component analysis) which is sensitive to low amplitudes (0.1 – 0.5 est (check the relevance)) <https://www.youtube.com/watch?v=nshi47blz7Q> (8 mins in)

If the signal is very unstable, cleanline plugin is recommended

**REGARDING EVENTS**

* type - specifies the type of the event. For example, 'square' in the example above is a stimulus type, 'rt' is a subject button-press (i.e., reaction-time) event, etc... In continuous datasets, EEGLAB may add events of type 'boundary' to specify data boundaries (breaks in the continuous data). The next section on event scripting provides more detailed information about this special event type.
* latency - contains event latencies. The latency information is displayed in seconds for continuous data, or in milliseconds relative to the epoch's time-locking event for epoched data. As we will see in the event scripting section, the latency information is stored internally in data samples (points or EEGLAB 'pnts') relative to the beginning of the continuous data matrix (EEG.data).
* duration - duration of the event. This information is displayed in seconds for continuous data, and in milliseconds for epoched data. Internally, duration is (also) stored in data samples (pnts).
* urevent - contains indices of events in the original ('ur' in German) event structure. The first time events are imported, they are copied into a separate structure called the 'urevent' structure. This field is hidden in the graphic interface (above) since they should not be casually modified.
* epoch - indices of the data epochs (if any) the event falls within. This field is only present for data for which data epochs have been extracted (which is not the case here since the data is continuous).