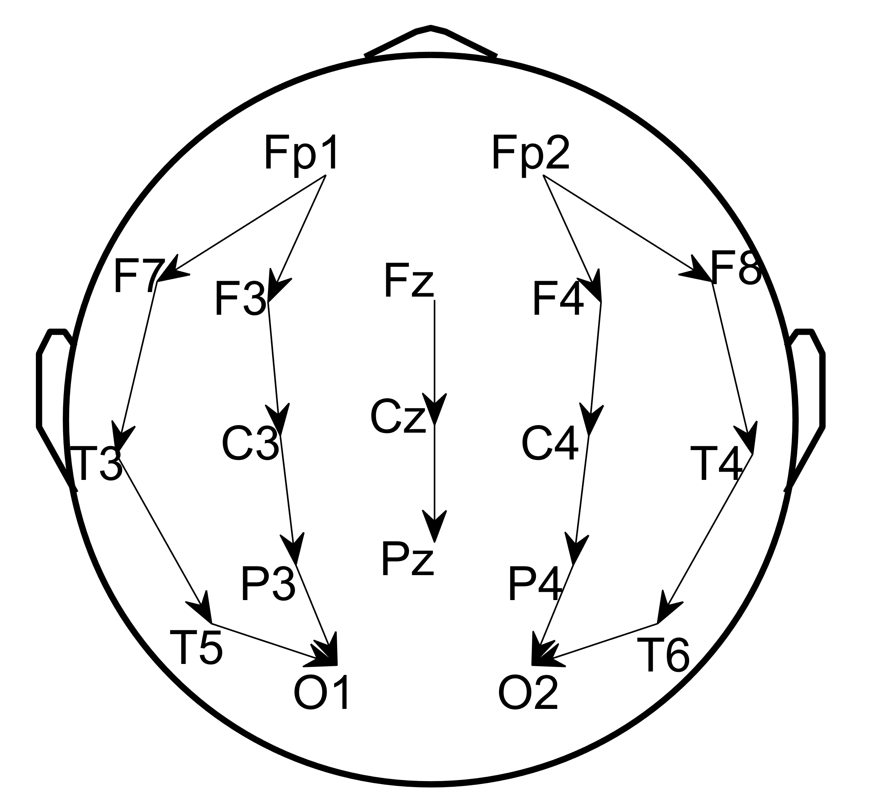
Requirements: MATLAB, EEGLAB (<https://sccn.ucsd.edu/eeglab/index.php>), and Python (Anaconda)

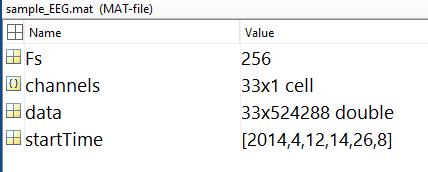
Input data: Raw EDF files inside .\Data\EDF\; scalp monopolar/C2 EEG that contains full set of 19 channels + 1 EKG (optional) as follows:

* Fp1 F3 C3 P3 F7 T3 T5 O1 Fz Cz Pz Fp2 F4 C4 P4 F8 T4 T6 O2 (EKG)

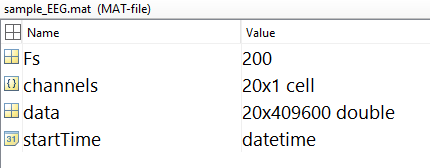


Step1: Read EDF to MAT using EEGLAB toolbox for MATLAB. Run script step1\_readEDF2MAT.m, which converts EDF format to MAT format in .\Data\MAT\ that contains the following variables:

* data: EEG array
* channels: list of channel names in data
* Fs: the sampling rate of data
* startTime: the start time vector of data



Step2: Preprocess MAT to select/rearrange channels, resample to 200Hz , and denoise with [0.5 40Hz] band-pass and 5Hz band-stop centered at the power-line frequency (US: 60Hz UK: 50Hz). Output files are saved in .\Data\processed\.



Step3: run SPaRCNet (Python backend)

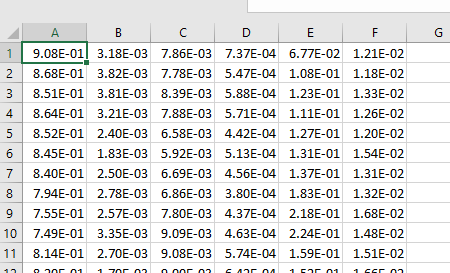
Configure Python

* Install anaconda3 and open a terminal
* $ conda create -n iiic python=3.6
* $ activate iiic
* $ conda install -c conda-forge hdf5storage
* $ pip install mne
* $ pip install torch==1.5.0+cpu torchvision==0.6.0+cpu -f https://download.pytorch.org/whl/torch\_stable.html

Run MATLAB wrapper step3\_runSPaRCNat.m

* CSV score table will be export to .\Data\iiic\
* Each row is the probabilities for 6 classes: Other, Seizure, LPD, GPD, LRDA, and GRDA
* Starting from the 1st 10sec EEG segment and moving at 2sec step in time

Eg. row #1: scores for [0 10sec], row #2: scores for [2 12sec], ...



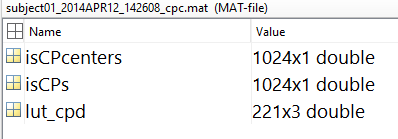
Step4: Run step4\_readCSV.m to read CSV to MAT to make sure every 2sec segment got scores. The output files are saved in .\Data\iiic\model\_prediction\.

Step5: Run step5\_computeSpectrograms to get regional average spectrograms in .\Data\Spectrograms\, which contains the following variables:

* Sdata: 4 regional average spectrograms
* stimes: time coordinates
* sfreqs: frequency coordinates
* params: spectrogram parameters

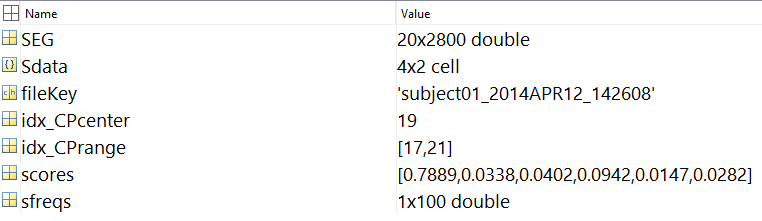
Step6: Run step6\_segementEEG.m to divide EEG into stationary periods with change point (CP) detection, and the output look-up-table are exported to .\Data\CPDs\ which contains the following variables:

* isCPcenters: 0 (not CP center) or 1 (is CP center)
* isCPs: 0 (not CP) pr 1 (is CP)
* lut\_cpd: index of each CP center (column #1) and its range [start (column #2), end (column #3)]



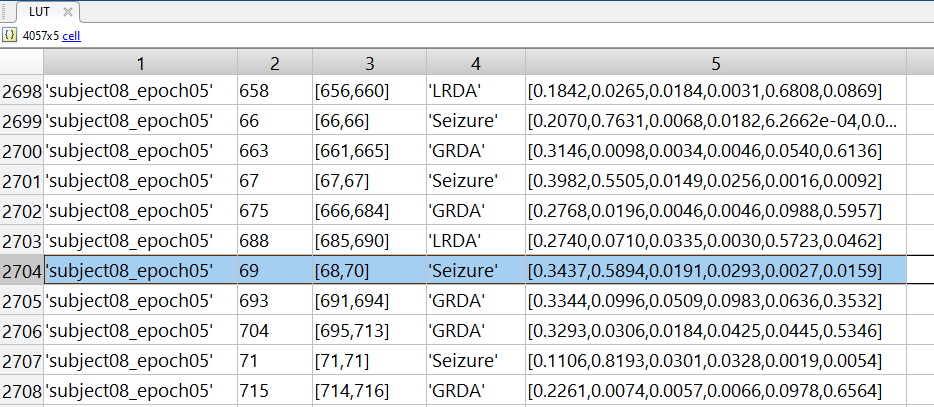
Step 7: Run step7\_parseCPcenters.m to parse data of CP centers for labelling GUI in .\Data\CP\_centers\ each contains the following variables:

* SEG: 14sec EEG
* Sdata: 10min spectrograms
* fileKey: EEG file token
* idx\_CPcenter: the index of CP center (2sec unit) in cEEG
* idx\_CPrange: the range of CP segment (2sec unit) in cEEG
* scores: model predicted probabilities, in order of Other, Seizure, LPD, GPD, LRDA, GRDA
* sfregs: frequency coordinates of spectrograms



Step8: Run step8\_getLUT.m to get the global look-up-table (LUT) for labelling GUI:

* Column #1: EEG name
* Column #2: sample index in 2sec segment (CP center)
* Column #3: CP range
* Column #4: model prediction
* Column #5: model probabilities for Other, Seizure, LPD, GPD, LRDA, GRDA

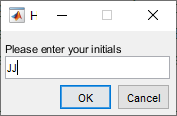


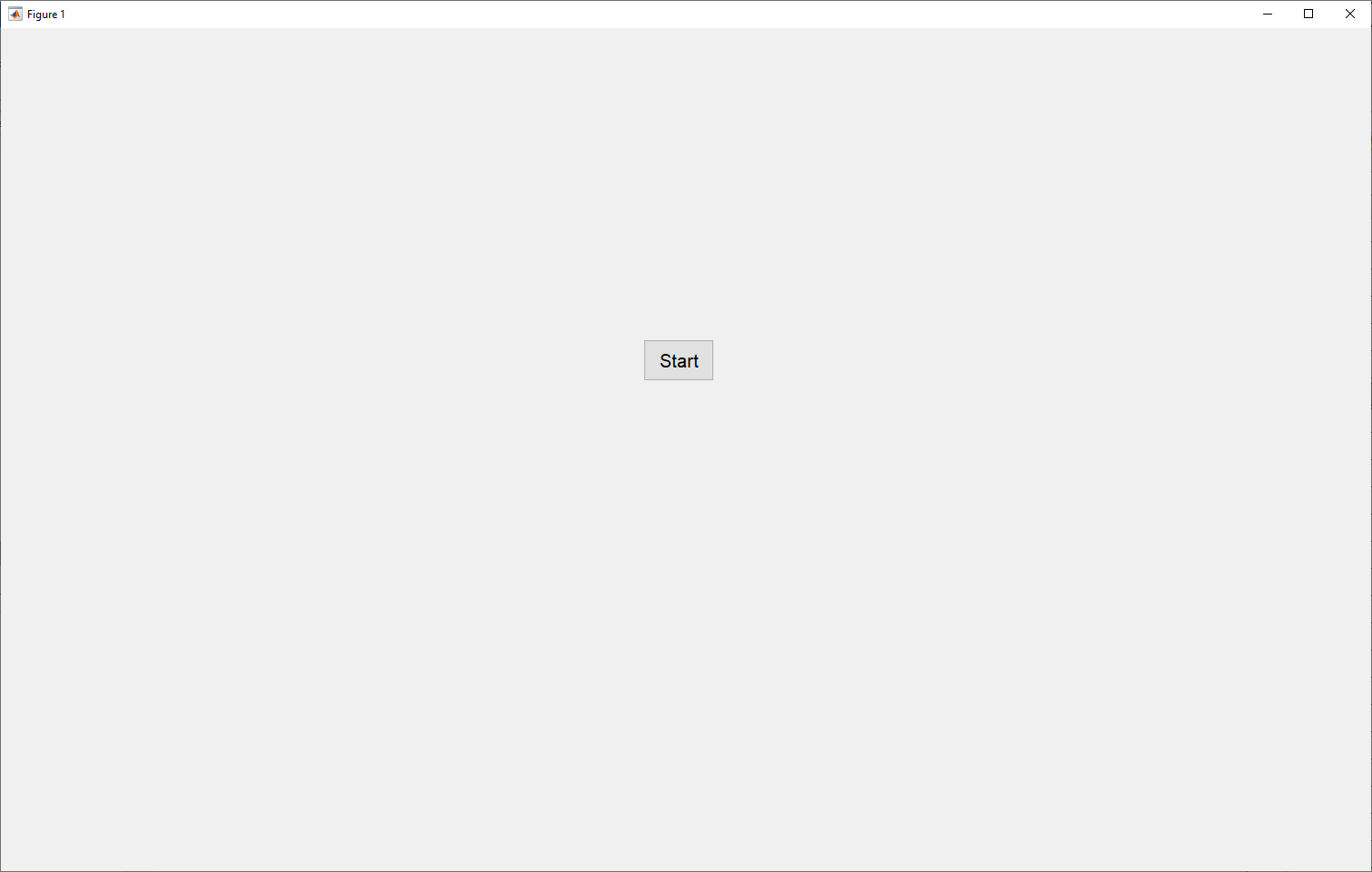
Step9: Run step9\_getBoW.m to get the bag of word (BoW) model using spectrograms with 500 words using K-means clustering method and compute the normalized distribution of words for each sample as BoW feature. This is further used in labelling GUI to do similarity search in chi -square distance.

Step10: Compute embedding (PaCMAP) and wrap all inputs for labelling GUI.

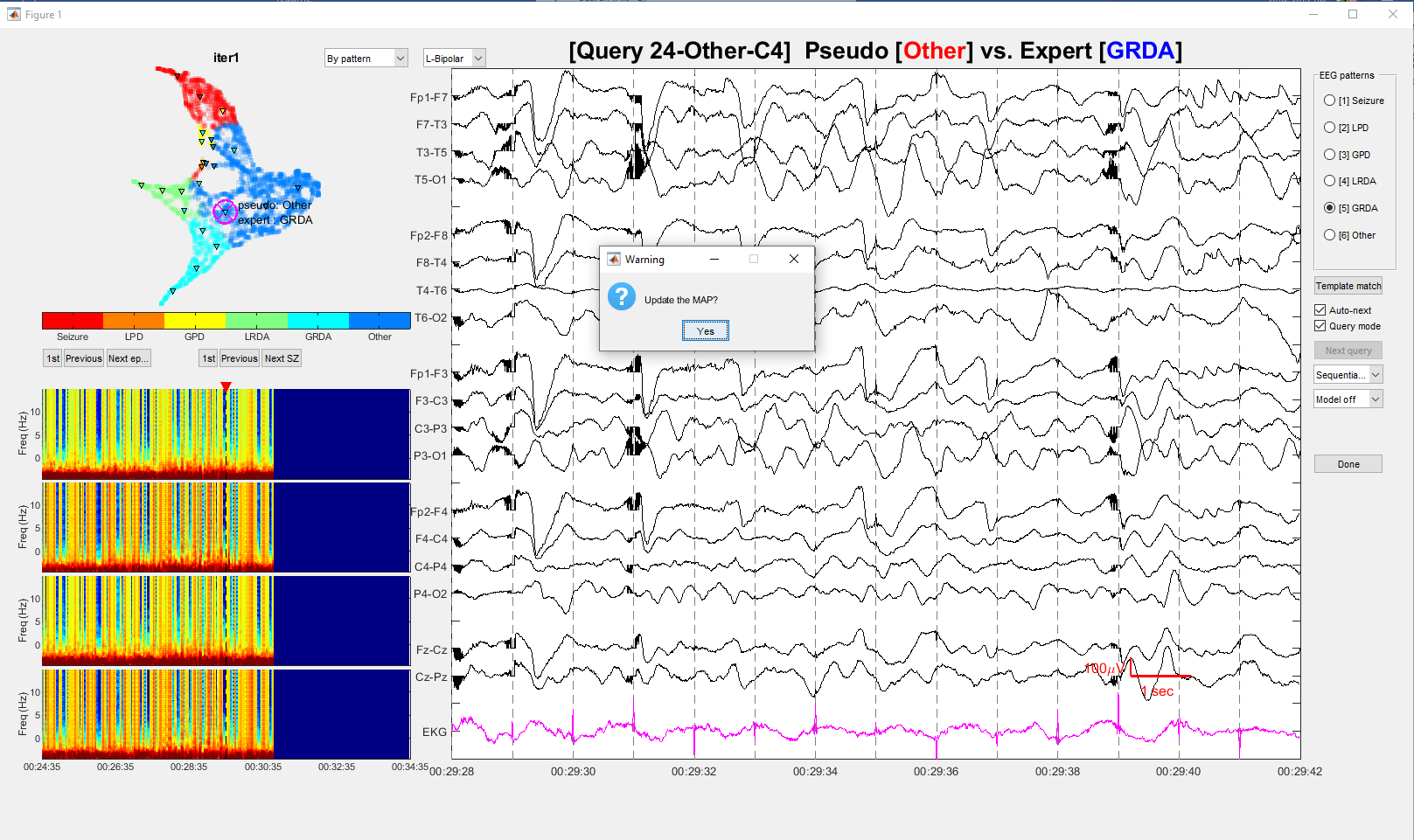
* Configure PaCMAP (Python library: <https://github.com/YingfanWang/PaCMAP>).
* Run step10\_getPaCMap.m to get the global embedding.

Step11: Run labeling GUI step11\_IIICGUI\_mPatients.m

 input rater initials to store scores.

 click Start to continue.

Label 30 to 50 samples selected by GUI per interaction (samples at the class boundaries).



Press Yes to update labels (spreading in PaCMAP by nearest labeled points) and enter next iteration.

Press Done button to seal and export the labels.

