Meeting 8

10/27/21

Rolando Martinez

Deliverables

- Apply welch function to all 64 electrodes in each epoch
- Aggregate into frequency bins into brainwaves bands
    - Delta, Theta, Alpha, Beta, Gamma
- Return dictionary
    - Keys = integers corresponding to each epoch
    - Values = list containing time index ranges and pandas dataframe

Methodology and Learnings

How did I do it?

- Nested for loops
- Dictionaries and lists

What did I learn on the way?

- More familiar with dataframes and dictionaries

Results

```
def getEpochbm_dict(eeg_data, epoch_dict):

    df = eeg_data.iloc[:, 0:64]     # create dataframe for 64 signals
    fs = 500                        # Sampling rate of 500 Hz
    epochbm_dict = dict()           # final dictionary

    # Define EEG frequency bands
    bands = {'Delta': (1, 3),
             'Theta': (4, 7),
             'Alpha': (8, 12),
             'Beta': (13, 25),
             'Gamma': (26, 45)}

    # Outer loop iterates through epoch dictionary and creates a dataframe for
    # each epoch based on its time index range and stores it along with the index
    # ranges in the final dictionary
    for e in epoch_dict:
        start_idx = epoch_dict[e][0]    # beginning time index range
        end_idx = epoch_dict[e][1]      # ending time index range

        eeg_list = []                   # list of band dictionaries
        idx_name = []                   # name of electrodes

        # slice dataframe according to epoch edges
        epoch_df = df[start_idx:end_idx]

        # inner loop goes through each column of the sliced dataframe applies the
        # welch function and stores the dictionaries in a list
        for i in epoch_df:
            idx_name.append(i)          # store name of electrode
            eeg_bands = dict()          # dictionary holding EEG bands as keys and freq as value

            # Take one electrode at a time from dataframe
            data_col = epoch_df.loc[:, i]
            eeg_signal = data_col.values
```

Results – cont.

```
        # welch function applied to each electrode using sampling rate
        freq_arr, psd_arr = signal.welch(eeg_signal, fs)

        # find freq bands of all electrodes for given epoch and store in dictionary
        for b in bands:
            # Find frequency match with EEG bands
            freq_ix = np.where((freq_arr >= bands[b][0]) & (freq_arr <= bands[b][1]))

            # Calculate the mean of power spectrum value
            eeg_bands[b] = np.mean(psd_arr[freq_ix])

        # add band dictionary to list of dictionaries
        eeg_list.append(eeg_bands)

    # create new dataframe for freq bands of each electrode for given epoch
    bands_df = pd.DataFrame(eeg_list, columns=['Delta', 'Theta', 'Alpha', 'Beta', 'Gamma'], index=idx_name)

    # add time index ranges and dataframe to dictionary in given epoch
    epochbm_dict[e] = [[start_idx, end_idx], bands_df]

return epochbm_dict
```

Results – cont.

1 : [0, 42663]

	Delta	Theta	Alpha	Beta	Gamma
Fp1	4.889927e-11	8.423252e-12	2.642536e-12	4.359902e-13	1.187204e-13
Fp2	4.390949e-11	7.402872e-12	2.419045e-12	4.088885e-13	1.127960e-13
F3	4.619386e-12	1.046754e-12	1.435916e-12	3.700543e-13	6.363369e-14
F4	3.606577e-12	9.316499e-13	1.307190e-12	4.907973e-13	1.646644e-13
C3	1.533518e-12	5.165224e-13	1.116517e-12	2.897726e-13	1.058815e-13
...	...	...	...	...	...
PO8	1.857688e-11	2.076086e-12	2.389193e-12	7.807219e-13	6.391389e-13
Fpz	3.610387e-11	6.126865e-12	2.290647e-12	3.633384e-13	9.878641e-14
CPz	4.051544e-12	1.029351e-12	1.191260e-12	2.830822e-13	6.620627e-14
POz	4.115669e-12	1.394971e-12	1.448764e-12	3.224239e-13	7.157896e-14
TP10	6.903166e-12	1.633495e-12	1.508446e-12	4.477264e-13	2.529003e-13

[64 rows x 5 columns]

2 : [42663, 91494]

	Delta	Theta	Alpha	Beta	Gamma
Fp1	1.779313e-11	4.163003e-12	2.456524e-12	4.575242e-13	1.391357e-13
Fp2	1.735419e-11	3.567409e-12	2.409759e-12	4.416931e-13	1.254445e-13
F3	3.086889e-12	9.584300e-13	1.509474e-12	3.742806e-13	1.149803e-13
F4	3.546418e-12	7.884749e-13	1.662349e-12	5.552834e-13	1.764324e-13
C3	1.668989e-12	5.598393e-13	1.319197e-12	3.756272e-13	2.212387e-13
...	...	...	...	...	...
PO8	4.876703e-12	1.508113e-12	2.394406e-12	8.061991e-13	6.947308e-13
Fpz	1.415141e-11	3.031471e-12	2.178861e-12	3.832828e-13	1.030079e-13
CPz	4.778353e-12	1.285299e-12	1.542061e-12	3.146615e-13	7.138623e-14
POz	3.436101e-12	1.285270e-12	1.679310e-12	3.305556e-13	7.514902e-14
TP10	4.792582e-12	1.307609e-12	1.566548e-12	5.535721e-13	4.254088e-13

3 : [91494, 100828]

	Delta	Theta	Alpha	Beta	Gamma
Fp1	3.686808e-11	1.342660e-11	3.258992e-12	4.835010e-13	1.180875e-13
Fp2	3.880886e-11	1.192793e-11	3.632454e-12	5.125668e-13	1.116843e-13
F3	3.823898e-12	1.288069e-12	1.772689e-12	4.379029e-13	8.694069e-14
F4	4.266782e-12	8.740953e-13	1.687049e-12	4.627807e-13	7.943096e-14
C3	1.942923e-12	7.721115e-13	1.710549e-12	4.022406e-13	2.026252e-13
...	...	...	...	...	...
PO8	5.689757e-12	1.843980e-12	2.564200e-12	8.724193e-13	6.697649e-13
Fpz	2.894869e-11	9.988427e-12	3.003091e-12	4.496562e-13	1.039248e-13
CPz	3.819268e-12	1.218681e-12	1.507282e-12	3.157477e-13	7.078322e-14
POz	4.109213e-12	1.503513e-12	1.876078e-12	3.411803e-13	7.001991e-14
TP10	6.863945e-12	1.823335e-12	1.594468e-12	5.759919e-13	4.291954e-13

[64 rows x 5 columns]

4 : [100828, 129160]

	Delta	Theta	Alpha	Beta	Gamma
Fp1	1.039539e-10	1.732446e-11	3.774389e-12	4.953543e-13	2.783105e-13
Fp2	1.363205e-10	1.867575e-11	4.003513e-12	4.997233e-13	3.152264e-13
F3	1.458678e-10	3.158588e-12	1.779845e-12	5.886986e-13	2.358607e-13
F4	7.947679e-11	2.148810e-12	1.908817e-12	4.888055e-13	1.774858e-13
C3	3.032301e-11	1.477255e-12	1.411105e-12	5.210646e-13	5.899476e-13
...	...	...	...	...	...
PO8	3.191164e-10	1.375950e-11	4.024865e-12	1.629463e-12	1.539830e-12
Fpz	1.002873e-10	1.368166e-11	3.596202e-12	4.527091e-13	2.282671e-13
CPz	3.403485e-11	1.510730e-12	1.836625e-12	3.005631e-13	9.097610e-14
POz	1.260696e-10	1.259192e-11	3.722210e-12	8.697276e-13	2.145512e-13
TP10	1.126437e-10	4.017120e-12	2.287776e-12	1.054095e-12	1.055179e-12