Quick tutorial on importing, processing and visualising Wada data

Sebastian Halder

June 2021

Reading and visualising Wada data

The goal of this program is to read the data, pre-process it using a bandpass filter and re-referencing, extract the feature we are interested in (signal diversity) and visualise it.

Importing required Python modules

We will be using MNE for EEG analysis, numpy and pandas for data structures, seaborn for visualisation and code provided by Michael Schartner [1] for calculating our signal diversity measure (Lempel-Ziv Complexity).

```
import mne
import numpy as np
import pandas as pd
import seaborn as sns
import LZ_Spectral
```

Importing and pre-processing

Reading the data

The data is stored as an EEGLAB set which can be read using a method provided with MNE. The boundary events are due to artifact removal.

```
raw = mne.io.read_raw_eeglab('patient1.set', preload = True)
```

```
Reading patient1.fdt
Reading 0 ... 872749  =      0.000 ...  3490.996 secs...
```

Bandpass filtering

This is an optional step. The band could also be modified to focus on a particular frequency range. E.g. alpha, beta, etc.

```
raw = raw.filter(l_freq=0.1, h_freq=45)
```

```
Filtering raw data in 1 contiguous segment
Setting up band-pass filter from 0.1 - 45 Hz

FIR filter parameters
---------------------
Designing a one-pass, zero-phase, non-causal bandpass filter:
- Windowed time-domain design (firwin) method
- Hamming window with 0.0194 passband ripple and 53 dB stopband attenuation
- Lower passband edge: 0.10
- Lower transition bandwidth: 0.10 Hz (-6 dB cutoff frequency: 0.05 Hz)
- Upper passband edge: 45.00 Hz
- Upper transition bandwidth: 11.25 Hz (-6 dB cutoff frequency: 50.62 Hz)
- Filter length: 8251 samples (33.004 sec)
```

Re-referencing

Originally the data were recorded with references behind the ears. Here we decide to re-ference to electrodes near the midline to minimise volume conduction effects. The electrodes on the left hemisphere (odd numbers) will be re-referenced to A1, the electrodes on the right hemisphere (even numbers) to A2. We will also discard all electrodes on the midline (Fz, Cz, etc.).

```
lh_labels = ['Fp1', 'F7', 'T7', 'P7', 'O1', 'F3', 'C3', 'P3', 'A1']
raw_lh = raw.copy();
raw_lh.pick_channels(lh_labels)
raw_lh.set_eeg_reference(ref_channels=['A1'])
lh_labels = ['Fp1', 'F7', 'T7', 'P7', 'O1', 'F3', 'C3', 'P3']
raw_lh.pick_channels(lh_labels)

rh_labels = ['Fp2', 'F8', 'T8', 'P8', 'O2', 'F4', 'C4', 'P4', 'A2']
raw_rh = raw.copy()
raw_rh.pick_channels(rh_labels)
raw_rh.set_eeg_reference(ref_channels=['A2'])
rh_labels = ['Fp2', 'F8', 'T8', 'P8', 'O2', 'F4', 'C4', 'P4']
raw_rh.pick_channels(rh_labels)
```

```
EEG channel type selected for re-referencing
Applying a custom EEG reference.
EEG channel type selected for re-referencing
Applying a custom EEG reference.
```

Selecting the segment of interest and epoching

The raw data is almost an hour long. We are primarily interested in the time before the injection of the drug and the first two minutes after the effects of the drug start (about 30s after injection; see Figure 1A in [2]). The injection time point (829s into the recording) is contained in the text file provided with the data. The file also contains the time at which the clinician determined all the effects of the drug had stopped and the timepoints of a second injection. We will use only the data from the first injection. We will also assign an ID to the participant and to the frequency band we filtered to. These values are not strictly necessary in the current analysis but would be useful if several participants and bands are analysed. The segment length could also be modified, we will use 6s (see [3] for a discussion of segment length).

We will first crop the data to the time around the injection we are interested in. Then we will generate fixed length events at intervals of half the segment length for 50% overlap. Finally we epoch the data.

```
myTimeSegment = 120
myTimeToExclude = 30
mySegmentLength = 6
myWadaTimepoint = 829
myParticipantNumber = 1
myBandName = 'broad'

# 1. Cropping to segment of interest

leftRest = raw_lh.copy().crop(myWadaTimepoint
  - myTimeSegment, myWadaTimepoint)
leftWada = raw_lh.copy().crop(myWadaTimepoint
  + myTimeToExclude, myWadaTimepoint + myTimeToExclude + myTimeSegment)

rightRest = raw_rh.copy().crop(myWadaTimepoint
  - myTimeSegment, myWadaTimepoint)
rightWada = raw_rh.copy().crop(myWadaTimepoint
  + myTimeToExclude, myWadaTimepoint + myTimeToExclude + myTimeSegment)

# 2. Generating the fixed length events at intervals of half the length
# we want to extract (for 50% overlap)
```

```
myEventsLR = mne.make_fixed_length_events(leftRest, 1, duration=mySegmentLength / 2)
myEventsLW = mne.make_fixed_length_events(leftWada, 1, duration=mySegmentLength / 2)

myEventsRR = mne.make_fixed_length_events(rightRest, 1, duration=mySegmentLength / 2)
myEventsRW = mne.make_fixed_length_events(rightWada, 1, duration=mySegmentLength / 2)

# 3. Epoching of the data based on the events

leftRestEpochs = mne.Epochs(leftRest, events=myEventsLR, tmin=0,
 tmax=mySegmentLength, baseline=None)
leftWadaEpochs = mne.Epochs(leftWada, events=myEventsLW, tmin=0,
 tmax=mySegmentLength, baseline=None)

rightRestEpochs = mne.Epochs(rightRest, events=myEventsRR, tmin=0,
 tmax=mySegmentLength, baseline=None)
rightWadaEpochs = mne.Epochs(rightWada, events=myEventsRW, tmin=0,
 tmax=mySegmentLength, baseline=None)
```

```
40 matching events found
No baseline correction applied
Not setting metadata
0 projection items activated
40 matching events found
No baseline correction applied
Not setting metadata
0 projection items activated
40 matching events found
No baseline correction applied
Not setting metadata
0 projection items activated
40 matching events found
No baseline correction applied
Not setting metadata
0 projection items activated
```

Feature extraction

Computing the features

In this example we will only compute signal diversity on a channel-wise basis. We will create several lists that contain information about each segment. If we were to compute several different measures (e.g. other signal diversity measures, spectral or connectivity) we could use this to identify them. The most important value in the current example bis the condition and the hemisphere. Our goal is to compare the hemispheres before and after the injection of the drug and between hemispheres during the injection of the drug. There are many other ways to approach this step so feel free to adapt it to your preferences.

```
myCompVal = []
myCompType = []
myParticipant = []
myCondition = []
myHemisphere = []
myBand = []
myEpoch = []
mySegLength = []
myChannelName = []

for epNum, ep in enumerate(leftRestEpochs[:]):
    for chNum, ch in enumerate(ep[:]):
        myCompVal.append(LZ_Spectral.LZs(ch))
        myChannelName.append(leftRestEpochs.ch_names[chNum])
```

```
        myCompType.append('LZs')
        myParticipant.append(str(myParticipantNumber))
        myHemisphere.append('L')
        myCondition.append('R')
        myBand.append(myBandName)
        mySegLength.append(str(mySegmentLength))
        myEpoch.append(str(epNum))

for epNum, ep in enumerate(leftWadaEpochs[:]):
    for chNum, ch in enumerate(ep[:]):
        myCompVal.append(LZ_Spectral.LZs(ch))
        myChannelName.append(leftWadaEpochs.ch_names[chNum])
        myCompType.append('LZs')
        myParticipant.append(str(myParticipantNumber))
        myHemisphere.append('L')
        myCondition.append('W')
        myBand.append(myBandName)
        mySegLength.append(str(mySegmentLength))
        myEpoch.append(str(epNum))

for epNum, ep in enumerate(rightRestEpochs[:]):
    for chNum, ch in enumerate(ep[:]):
        myCompVal.append(LZ_Spectral.LZs(ch))
        myChannelName.append(rightRestEpochs.ch_names[chNum])
        myCompType.append('LZs')
        myParticipant.append(str(myParticipantNumber))
        myHemisphere.append('R')
        myCondition.append('R')
        myBand.append(myBandName)
        mySegLength.append(str(mySegmentLength))
        myEpoch.append(str(epNum))

for epNum, ep in enumerate(rightWadaEpochs[:]):
    for chNum, ch in enumerate(ep[:]):
        myCompVal.append(LZ_Spectral.LZs(ch))
        myChannelName.append(rightWadaEpochs.ch_names[chNum])
        myCompType.append('LZs')
        myParticipant.append(str(myParticipantNumber))
        myHemisphere.append('R')
        myCondition.append('W')
        myBand.append(myBandName)
        mySegLength.append(str(mySegmentLength))
        myEpoch.append(str(epNum))
```

Conversion to Pandas data frame

This is also a step that may be accomplished in very many ways based on your preference. I chose to convert the data to a Pandas data frame.

```
myAll = np.concatenate([myCompVal, myChannelName, myCompType,
  myParticipant, myHemisphere, myCondition, myBand, mySegLength, myEpoch])

myAll = myAll.reshape((9, len(myCompVal)))
myAll = np.transpose(myAll)

df = pd.DataFrame(myAll, columns=['Signal Diversity', 'Channel', 'Method',
  'Participant', 'Hemisphere', 'Condition', 'Band', 'Segment','Epoch'])
df["Signal Diversity"] = df["Signal Diversity"].astype('float64')
df["Channel"] = df["Channel"].astype('category')
df["Method"] = df["Method"].astype('category')
df["Participant"] = df["Participant"].astype('category')
```

```
df["Hemisphere"] = df["Hemisphere"].astype('category')
df["Condition"] = df["Condition"].astype('category')
df["Band"] = df["Band"].astype('category')
df["Segment"] = df["Segment"].astype('category')
df["Epoch"] = df["Epoch"].astype('category')
```

Below is a summary of the content.

```
df.head
```

```
Signal Diversity Channel Method Participant Hemisphere Condition   Band  \
0              0.684982     Fp1    LZs           1          L         R  broad
1              0.693727      F7    LZs           1          L         R  broad
2              0.723247      T7    LZs           1          L         R  broad
3              0.704797      P7    LZs           1          L         R  broad
4              0.663004      O1    LZs           1          L         R  broad
...                 ...     ...    ...         ...        ...       ...    ...
1243           0.681159      P8    LZs           1          R         W  broad
1244           0.650735      O2    LZs           1          R         W  broad
1245           0.580882      F4    LZs           1          R         W  broad
1246           0.639706      C4    LZs           1          R         W  broad
1247           0.622711      P4    LZs           1          R         W  broad

     Segment Epoch
0          6     0
1          6     0
2          6     0
3          6     0
4          6     0
...      ...   ...
1243       6    38
1244       6    38
1245       6    38
1246       6    38
1247       6    38
```

Visualisation

Visualising the data as a topography

We want to inspect the data as a scalp topography. We can use MNE to make a standard montage for the channels that we selected. We then iterate over the channels to extract the data in the sequence they are contained in the montage. This step is required because the order of the channels in the data frame and the montage differs. So we need to make sure we put the correct values in the correct locations. I chose to use the median of the signal diversity values over the epochs we extracted. The colormap and value range can be adapted.

The left hemisphere was injected in this case and we can observe a decrease in signal diversity as expected.

```
montage = mne.channels.make_standard_montage('standard_1005')
info = mne.create_info(df['Channel'].unique().tolist(), 1, 'eeg').set_montage(montage)

myTopoRest = np.zeros((16))
myTopoWada = np.zeros((16))

for chNum,ch in enumerate(info['ch_names']):
    myTopoRest[chNum] = df.query("Condition == 'R' and Channel == '"
      + ch + "'").median()['Signal Diversity']
    myTopoWada[chNum] = df.query("Condition == 'W' and Channel == '"
      + ch + "'").median()['Signal Diversity']
```

```
mne.viz.plot_topomap(myTopoRest, pos=info,  cmap='Spectral_r', vmin=0.5, vmax=0.75)
mne.viz.plot_topomap(myTopoWada, pos=info,  cmap='Spectral_r', vmin=0.5, vmax=0.75)
```

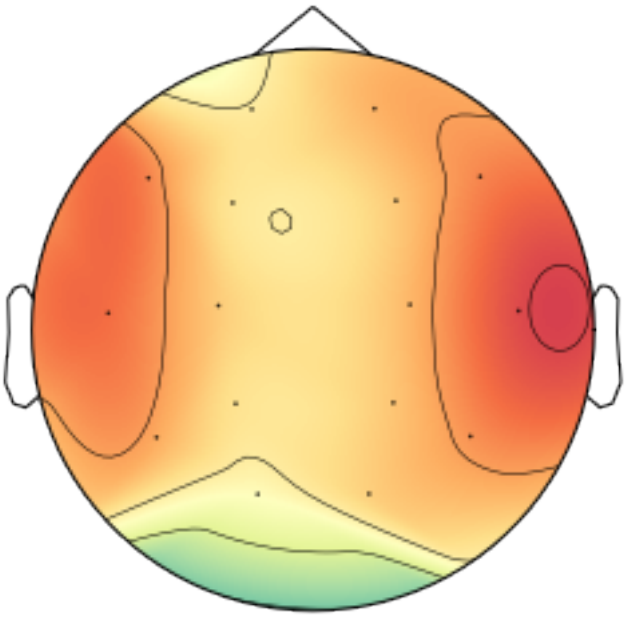

Figure 1: Topography during normal state before injection of drug. Signal diversity values similar on both hemispheres.

Visualisation as boxplot

Again there are very many options to do this. I am a fan of the seaborn module and use this implementation of a boxplot. It works nicely with the pandas data frame. The despining is optional and can be omitted.

Again we can observe that the values drop on the left hemisphere (injected) and less so on the right hemisphere (normal).

```
sns.boxplot(x='Hemisphere', y='Signal Diversity', hue='Condition', data=df, fliersize=0)
sns.despine(offset=10, trim=True);
```

Conclusions

This analysis could be augmented in many ways but that is beyond the scope of this workshop. Have a look at [2] for the analysis that we did.

The data can be accessed via this link:

https://osf.io/fhk6y/?view_only=a05e1d9b593c4beb927976f5d701e996

[1] Schartner, M. M., Carhart-Harris, R. L., Barrett, A. B., Seth, A. K., & Muthukumaraswamy, S. D. (2017). Increased spontaneous MEG signal diversity for psychoactive doses of ketamine, LSD and psilocybin. Scientific reports, 7(1), 1-12.

[2] Halder, S., Juel, B. E., Nilsen, A. S., Raghavan, L. V., & Storm, J. F. (2021). Changes in measures of consciousness during anaesthesia of one hemisphere (Wada test). NeuroImage, 226, 117566.

[3] Schartner, M., Seth, A., Noirhomme, Q., Boly, M., Bruno, M. A., Laureys, S., & Barrett, A. (2015). Complexity of multi-dimensional spontaneous EEG decreases during propofol induced general anaesthesia. PloS one, 10(8), e0133532.

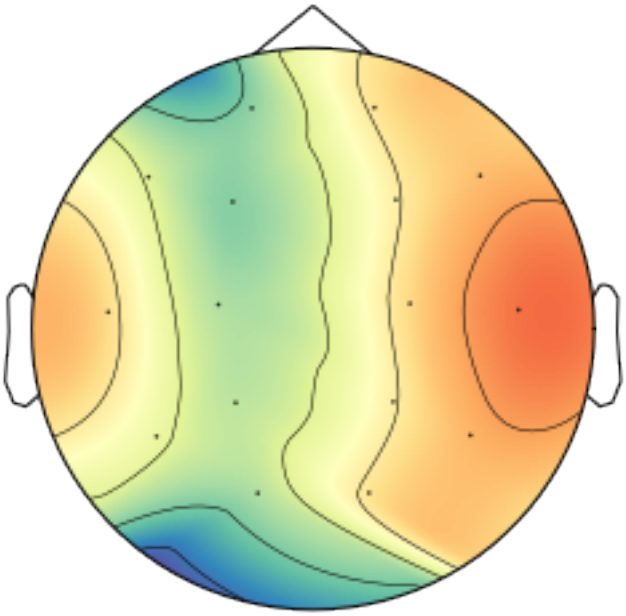

Figure 2: Topography after the injection of the drug. Signal diversity values reduced on left hemisphere (injected).

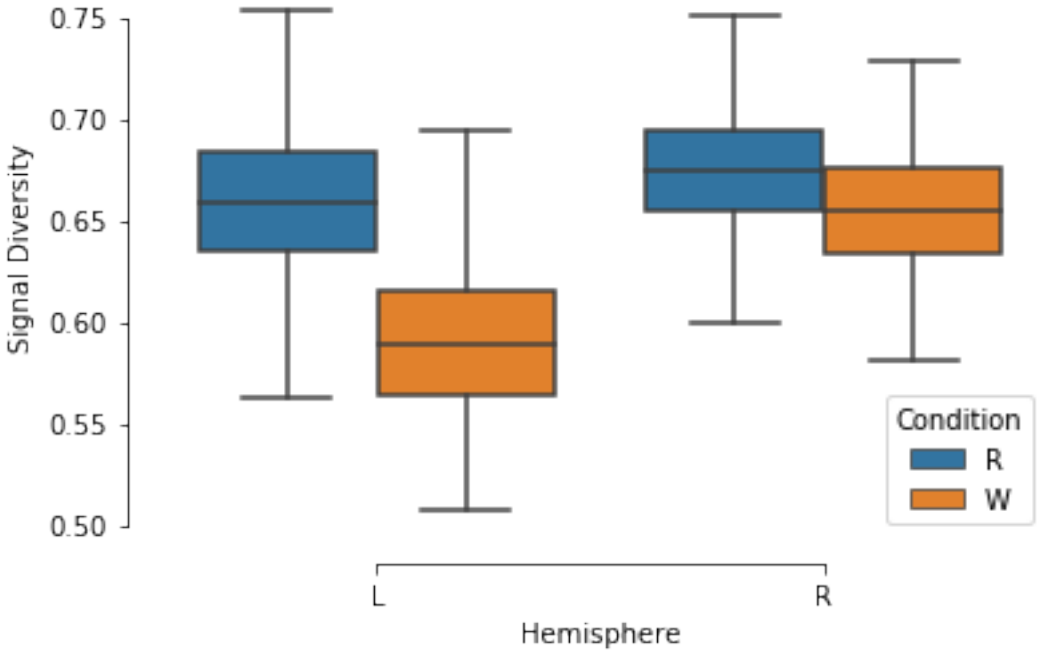

Figure 3: Boxplot of signal diversity values. Values reduced during Wada (W) on left hemisphere (injected) and similar during normal state (label R for rest).