# Worksheet: Circular statistics

Joaquin Rapela

February 14, 2024

In this worksheets you will practice:

- accessing electrophysiology data from Dandi,
- bandpass filtering a signal,
- calculating the instantaneous amplitude and phase of a signal with the Hilbert transform,
- computing the circular mean of a set of phases,
- testing for non-uniformity of a set of circular variables with the Rayleigh test,
- detecting travelling waves in local field potentials,
- performing linear regression analysis.

You will quantitatively characterise travelling waves in electro-corticographic recording from humans during the production of consonant vowel syllables, as described in Rapela [2016, 2017, 2018]. A video illustrating these travelling waves can be found here. This video shows the local field potential (LFP) voltages bandpass filtered between 0.4 and 0.8 Hz, around the mean frequency of consonant-vowel syllable production of 0.62 Hz.

## 1 Install the Python packages required to obtain data from Dandi

Do the following installations in the order specified (i.e., first the conda and later the pip installs).

```
# functionality to manipulate data in the NWB format
conda install conda-forge::pynwb

# functionality to access data from Dandi
pip install dandi

# this package contains the function rayleightest
pip install astropy
```

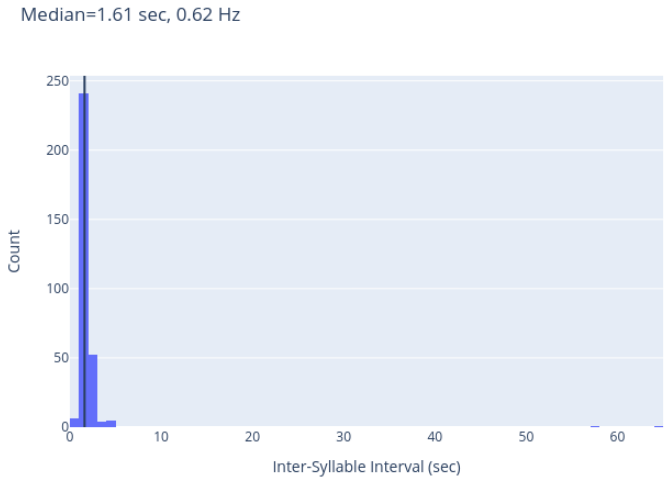



Figure 1: Histogram of inter-syllable intervals

# 2 Download ECoG recordings from Dandi

The script doDownloadData.py illustrates how to use the Dandi Python API to download an ephys data set. The default parameters of this script are set to download ECoG recordings from the subject and session analysed in Rapela [2016, 2017, 2018].

This script saves a Numpy `.npz` file containing the items:

`voltages` : array of size `n_electrodes x n_samples` such that `voltages[i,j]` is the recorded voltage at electrode $i$ and sample point $j$.

`srate` : recording sample rate (Hz).

`electrodes` : electrode numbers saved. The default parameters of the script only save electrodes 135 to 142.

`cvs_transition_times` : transition times between consonant and vowels (i.e., `cvs_transition_times[i]` is the time at which the subject transitioned between the consonant and the vowel of the ith syllable).

# 3 Calculate the mean frequency of consonant-vowel syllable production

Plot a histogram of times between production of consecutive consonant vowel syllables. Calculate the median of these times and add it to the title of the histogram. Refer to Fig. 1.

We called the inverse of this number the mean consonant vowel production frequency. Below we narrow filter the ECoG recordings around this frequency.

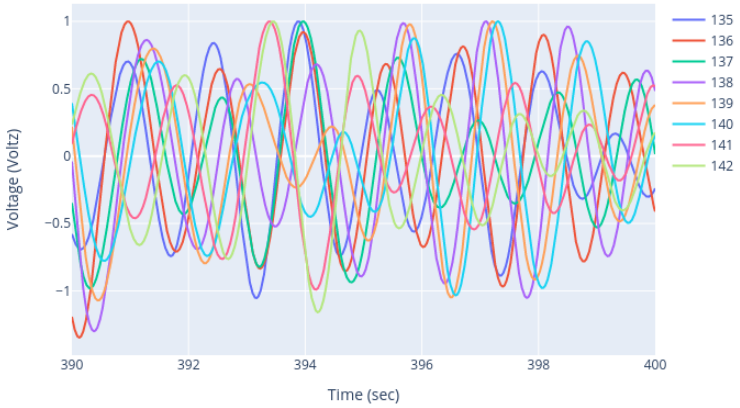



Figure 2: Voltages filtered between 0.4 and 0.8 Hz

# 4 Bandpass filter the raw voltages

The script doBandpassVoltages.py bandpasses the recorded Voltages between 0.4 and 0.8 Hz. It uses a second order butterworth filter (function `scipy.signal.butter`) and forward-backward filtering (function `scipy.signal.filtfilt`) to not introduce phase delays. It saves the filtered voltages as variable `filtered_voltages`.

Use the filtered voltages to reproduce Figure 6 from Rapela [2016] with electrodes 135-142 (Figure 2). Can you see the phase delays between the different electrode waveforms?

# 5 Compute the Hilbert transform of the filtered voltages

The script doHilbertTransform.py calculates the Hilbert transform of the bandpass filtered voltages. It saves the complex time series output of the Hilbert transform in variable `ht_filtered_voltages`. The phase of this time series at time $t$ is the instantaneous phase of the original voltages at time $t$ in the filtered filtered frequency range 0.4-0.8 Hz. We will use these phases below.

# 6 Plot phase histograms, compute the mean resultant vector and test for circular non-uniformity

Reproduce Figure 17 from Rapela [2017].

First extract the phases at the times of consonant-vowel transitions. The following code snippet illustrates one way of doing this.

```
load_res = np.load(ht_filename)
ht_filtered_voltages = load_res["ht_filtered_voltages"]
srate = load_res["srate"]
electrodes = load_res["electrodes"]
cvs_transition_times = load_res["cvs_transition_times"]

```

```
times = np.arange(0, ht_filtered_voltages.shape[1]) / srate
cvs_trans_samples = [np.argmin(np.abs(times-cvs_transition_time))
                     for cvs_transition_time in cvs_transition_times]

phases = np.angle(ht_filtered_voltages)
cvs_transition_phases = phases[:, cvs_trans_samples]
```

Then for each electrode separately calculate the mean resultant vector

$$\bar{\mathbf{R}} = \frac{1}{N}\sum_{i=1}^{N} vec(\theta_i) \tag{1}$$

The following function does this

```

def calculateMeanResultantVector(angles):
    vectors = np.array([np.exp(1j*angle) for angle in angles])
    mean_resultant_vector = vectors.mean()
    return mean_resultant_vector
```

where `angles` represent the phases of from a given electrode.

Next plot a circular histogram of the phases of an electrode. Add to this histogram a vector with radius equal to the absolute value of the mean resultant vector and angle equal to the phase of this vector.

Add to the title of this plot the p-value of a Rayleigh non-uniformity test. To compute this p-value use the function `astropy.stats.rayleightest` (Figure 3).

# 7 Checking for travelling waves events

Reproduce Figure 15 from Rapela [2018].

Select a time of your interest to check if at this time a travelling wave was propagating among electrodes 135-142.

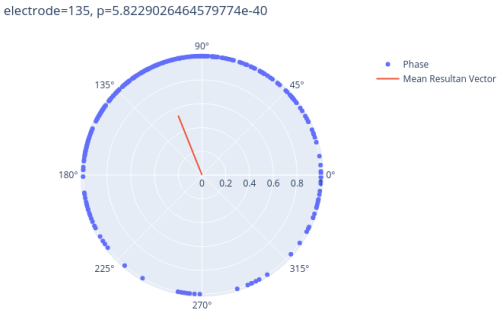
First compute the phase differences of all electrodes with respect to electrode 142. Because phases, and phase differences, are invariant to the addition of multiples of $2\pi$, we will unwrap the phases differences so that they all are in a consistent range, using the function `np.unwrap` that this. The following code does this.

```
    phases = np.angle(ht_filtered_voltages[:, wave_event_sample])
    phase_diffs = np.unwrap(phases - phases[-1])
```

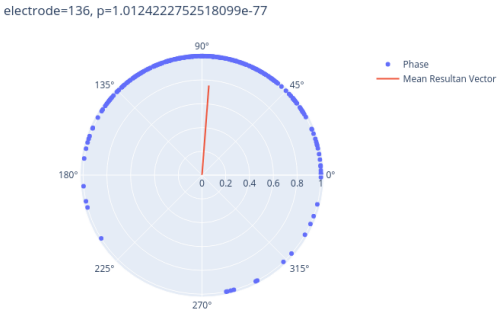
Then calculate the distances between the different electrodes and electrode 142, using an inter-electrode distance of 4 mm. Finally, perform a linear regression analysis between the electrodes phase differences and electrodes separation with respect to electrode 142.

```
electrodes_distances = np.arange(len(electrodes)-1, 0-1, -1) *
   electrodes_separation
lm_res = scipy.stats.linregress(x=electrodes_distances, y=phase_diffs)
```

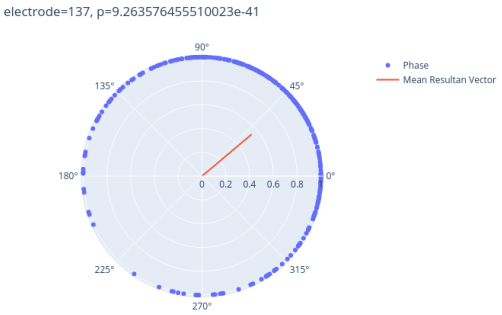
Draw a scatter plot of phase differences versus electrodes separations and superimpose the best fitting line. Add to the the title the slope of this line (i.e., the speed of the travelling wave) and the p-value of the linear regression analysis (Figure 4).



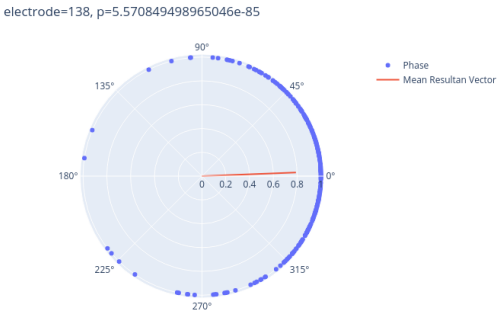

(a) Electrode 135.




(b) Electrode 136.




(c) Electrode 137.




(d) Electrode 138.

Figure 3: Histograms of phases at the times of consonant-vowel transitions at different electrodes along the dorso-ventral axis.

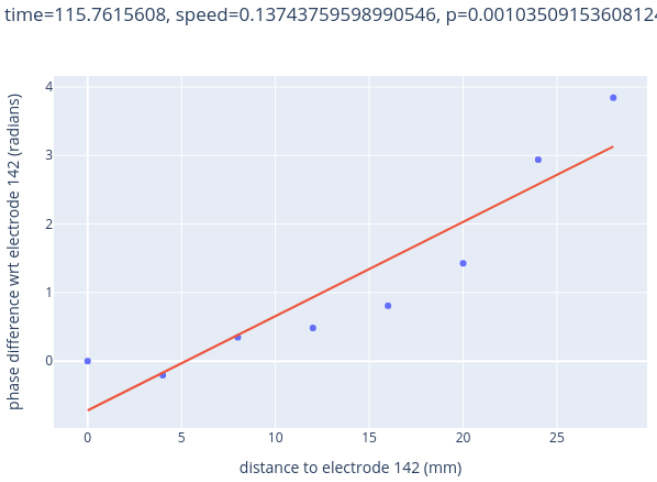



Figure 4: Wave event among electrodes 135 and 142 at time 115.76 secs.

# References

Joaquín Rapela. Entrainment of traveling waves to rhythmic motor acts, 2016. URL http://arxiv.org/abs/1606.02372.

Joaquín Rapela. Rhythmic production of consonant-vowel syllables synchronizes traveling waves in speech-processing brain regions, 2017. URL https://arxiv.org/abs/1705.01615.

Joaquín Rapela. Traveling waves appear and disappear in unison with produced speech, 2018. URL https://arxiv.org/abs/1806.09559.