User-friendly interface for song sorting

Context and purpose

The ability to learn, modify, and produce vocalizations through imitation, also known as vocal learning, is a rare skill in the animal kingdom. Humans possess this remarkable ability, and speech is a vital aspect of our evolution as individuals and as a society. To learn how to speak, children rely on listening to and mimicking the speech of their peers during the early stages of life, and they must do so within a critical period. This time-constrained trial-and-error process allows them to master this incredibly complex sensorimotor skill by building the necessary fine motor coordination of more than 100 muscles required for fluent speech (Kuhl and Meltzoff, 1996).

Just like humans, songbirds can learn and develop their songs, which is an essential skill for successful mating and territorial protection. While certain aspects of birdsong and speech are not directly comparable, such as language's ability for meaning and abstraction, there are remarkable similarities in how sensory experiences are processed and used to shape vocal outputs. These similarities suggest that similar neural mechanisms are at play in both birds and humans (Doupe and Kuhl, 1999). As a result, songbirds are often used as models in neuroscientific research. One of the most commonly used species is *Taeniopygia guttata*, also known as the zebra finch. These birds are easy to maintain in lab conditions and reliably produce a highly stereotyped song with 4-6 syllables (Figure 1) multiple times per day.

As part of the workflow in studying vocal learning in zebra finches, it is necessary to capture the properties of their song. This requires experimentalists to follow a series of steps involving Python scripts to accurately identify the different syllables in a bird's song. This process can be time-consuming, especially for scientists who are not proficient in Python programming. To prevent human errors and standardize the process, the project's objective should be to design a user-friendly interface that combines all the functions used in song processing. Ideally, the interface should allow the user to access spectrograms and other plots already existing in the analysis pipeline while being able to modify the necessary variables.

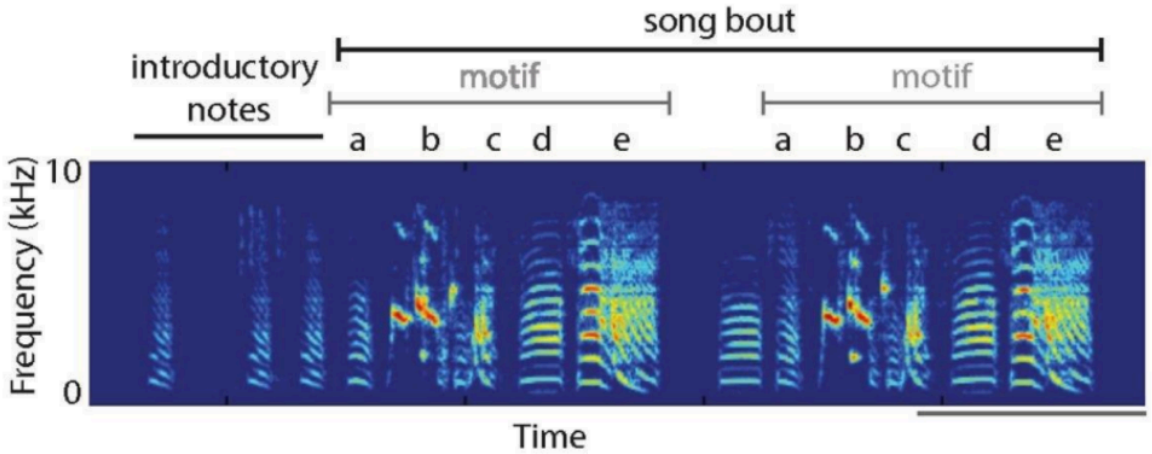

Figure 1. Structure of zebra finches' song. *Song is represented on a spectrogram (frequency over time). Syllables are preceded by introductory notes, which are not part of the motif. A song bout can*

include several motifs played in a row. A motif is defined by a sequence of syllables or notes with a solid pattern of frequencies, that is repeated frequently (scale bar: 500 ms). Source: Chen et al., 2016.

Concrete objectives

1. ***Pre-processing of the sound files:***

From the total number of song recordings, some will be manually annotated (the motifs will be identified with syllables according to the amplitude of the sound and their frequency content, just as in the figure above). The goal is to be able to select an amplitude threshold and manually label the syllables according to the spectrogram, visualizing amplitude and frequencies. To do this, there will be four steps:

a. Selection of an amplitude threshold to detect the song motifs and separate syllables
b. Split the recordings and eliminate the chunks containing silence
c. Recursively label each of the syllables manually by visualizing spectrogram and amplitude plots over the selected dataset (Figure 2)
d. Create a data directory where we include each chunk in .npy format along with its annotation in .csv

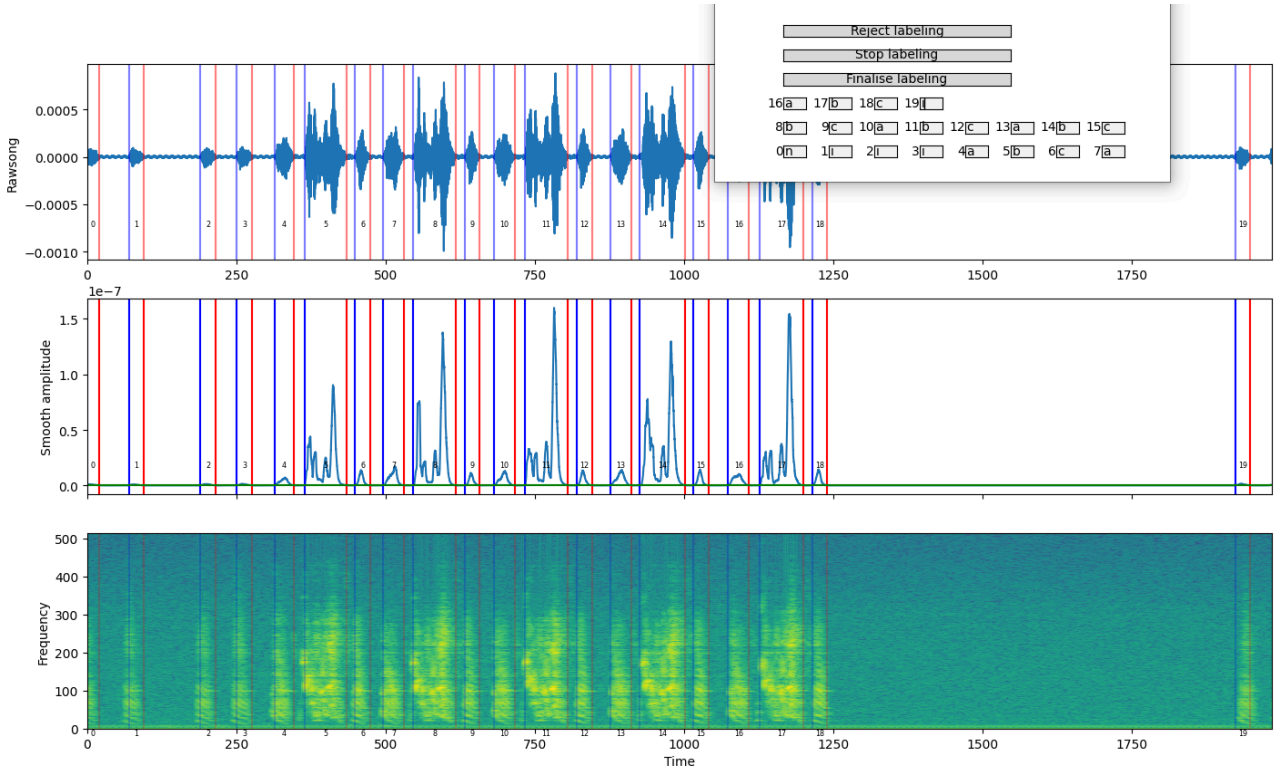

Figure 2. Example of manual labeling visualization*. From top to bottom: raw song given in amplitude of the signal, smooth amplitude of said song, and spectrogram showing the frequency pattern of the song motifs. Blue vertical lines represent syllable onsets and red vertical lines represent its offset.*

2. Post-processing

After having manually labeled part of the given dataset, external software will help us annotate the rest of the recordings. Then, we will have to check that the automatic labeling has been performed correctly and make some corrections otherwise. The set of scripts to do so will also require some visualization tools and the possibility to select thresholds and change data frames. These are:

a. Correct the onset and offset timings of each of the syllables that have been automatically annotated according to an amplitude threshold
b. Detection of outliers based on the syllable duration
c. Syntax-based correction
d. Splitting of syllables in subtones and renaming of the sequence

Important: some of the scripts for these projects are already developed and would have to be adapted and integrated into the workflow