# Sound and music description, revisited

Tomás Bilal Iaquinta

February 2020

---

## Question 1

As a first step 10 instruments were selected among the suggested ones, those were: **acoustic guitar, violin, trumpet, naobo, cello, snare drum, transverse flute, bassoon, clarinet, xiaoluo**. These represent different families of instruments: percussive, stringed and wind.

The code was modified for having high quality downloads as suggested. After reviewing the *Freesound.org* web, a dictionary in Python was created to include each instrument as a key, along with tags for each one, that point to one note/stroke sounds. The tags used were mostly **'single-note'** and **'1-shot'**. For the snare drum the query text **'snare_pearl'** was used since it successfully yielded only single strokes. The selection of the transverse flute was also helpful for obtaining single notes.

Please refer to the uploaded archives for the specific code used.

## Question 2

Next step was to try different pair of parameters and watching the scatter plot for all sounds, after a several tries certain parameters were extracted for the k-means algorithm.

Again, after trying different combinations a **68.5%** which will be our **baseline clustering performance**. The parameters used were:

- **Low-level dissonance mean**
- **Log attack time mean**
- **Low-level MFCC mean** (3rd coefficient)

## Question 3

For this question we need to first update the **essentia** package, which was done via the terminal by:

```
python -m pip install --upgrade essentia
```

Then, a function called `computeEnergy()` was created, in order to compute the **energy** and **STFT** of the input audio file for each frame of length $H$. Plots were computed for the energy of some instruments, in order to look for a threshold candidate to not consider non-desirable pieces of the audio (i.e. silence). Then, an energy `threshold = 0.002` was selected. Another function called `plotSampleEnergy()` was created to plot sample sounds energies along with the threshold was also created. Yielding for example what is seen in Figure 1.

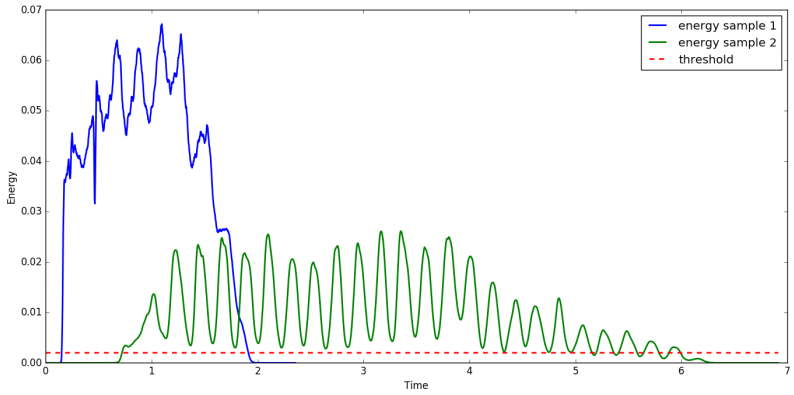


Figure 1: Energy of sample sounds and selected energy threshold.

So, to calculate the parameters the code is modified to compute only the part of the audio in between the first and the last frames where the energy is superior to the energy threshold. This is done with the function `frameEnergyThreshold()` which calls the previously introduced one and returns frame by frame the input audio where the condition stated above is true.

Then, the `LowLevelSpectralExtractor()` function was used to extract a multitude of parameters for each input audio frame. These parameters are related to several spectral characteristics, which are seen as adequate to improve the clustering efficiency. Only some of the extracted parameters will be used, while other are discarded.

All selected parameters are computed with the `calculateParameters()` function, and then by running the `addParameters()` function, all the `.json` files downloaded from `Freesound.org` are edited, to include the new parameters calculated. The `soundAnalysis.py` is updated for their clustering

analysis.

Repeating what was done on Question 2, different scatter plots are viewed, and the clustering performance for some combinations is computed.

Finally, with the parameters indicated below a **75% clustering performance** is obtained, showing an improvement on the previously obtained one. These are:

- **Low-level MFCC mean** (2nd coefficient)
- **Low-level MFCC mean** (3rd coefficient)
- **Spectral flatness** [dB]

Other shortlisted parameters, which yielded similar accuracy, where: -**Log attack time**, -**Low-level spectral contrast** (mean 0, 4, 5), -**Bark bands skewness**, -**Pitch**, -**Pitch salience**, -**Spectral complexity**, -**Spectral energy band**(middle), and -**Spectral roll-off**.

Concluding, by adding one of the new parameters in the analysis, and combining with the ones downloaded from the page, the accuracy is improved. For improving the obtained representation one possibility would be to calculate everything again using the `frameEnergyThreshold()` function.

## Final remarks

- In the uploaded files you can find all relevant code used for the assignment. If you insert you API key in the `assignment.py` file, you can run all the code as was performed.
- Be aware that the execution of the mentioned file, and of the function `addParameters()` takes a considerable amount of time, since it makes a very long calculation.
- By including more than 3 parameters a better performance will be obtained, however since so many parameters are available, obtaining a good performance with only this quantity was the objective.