Getting to Know the Mel Spectrogram

Dalya Gartzman Follow
Aug 19, 2019 · 5 min read

Read this short post if you want to be like Neo and know all about the Mel Spectrogram! (Ho maybe not all, but at least a little)

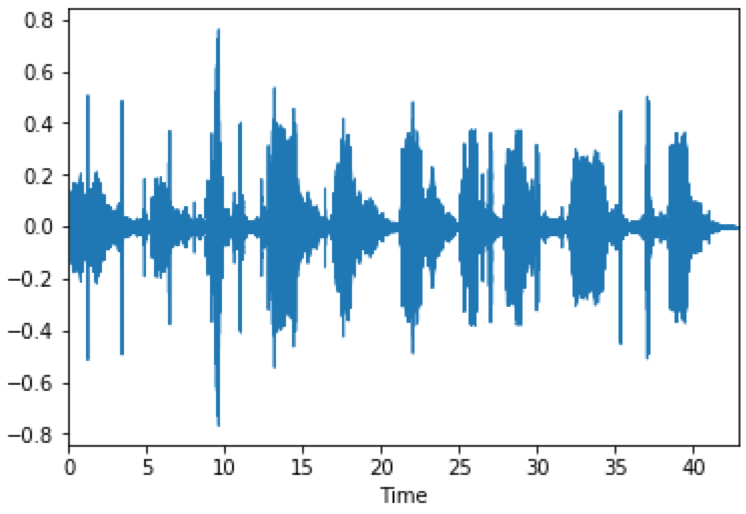
> *For the tl;dr and full code, go here.*

A Real Conversation That Happened in My Head a Few Days Ago

Me: Hi Mel Spectrogram, may I call you Mel?
Mel: Sure.

```python
import librosa
import librosa.display

filename = 'Haunting_song_of_humpback_whales.wav'
y, sr = librosa.load(filename)
# trim silent edges
whale_song, _ = librosa.effects.trim(y)
librosa.display.waveplot(whale_song, sr=sr);
```

display_waveplot_librosa.py hosted with ♡ by GitHub view raw

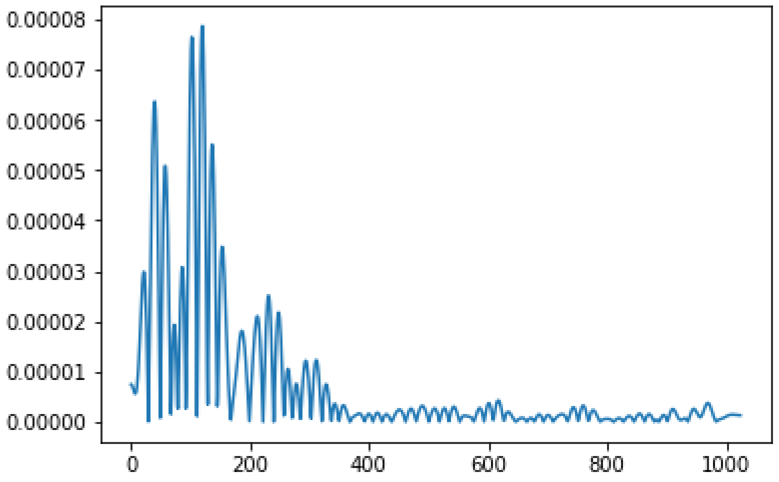


But this is just a two dimensional representation of this complex and rich whale song! Another mathematical representation of sound is the Fourier Transform. Without going into too many details (watch this educational video for a comprehensible explanation), Fourier Transform is a function that gets a signal in the time domain as input, and outputs its decomposition into frequencies.

Let’s take for example one short time window and see what we get from applying the Fourier Transform.

```python
import numpy as np
import matplotlib.pyplot as plt

n_fft = 2048
D = np.abs(librosa.stft(whale_song[:n_fft], n_fft=n_fft, hop_length=n_fft+1))
plt.plot(D);
```

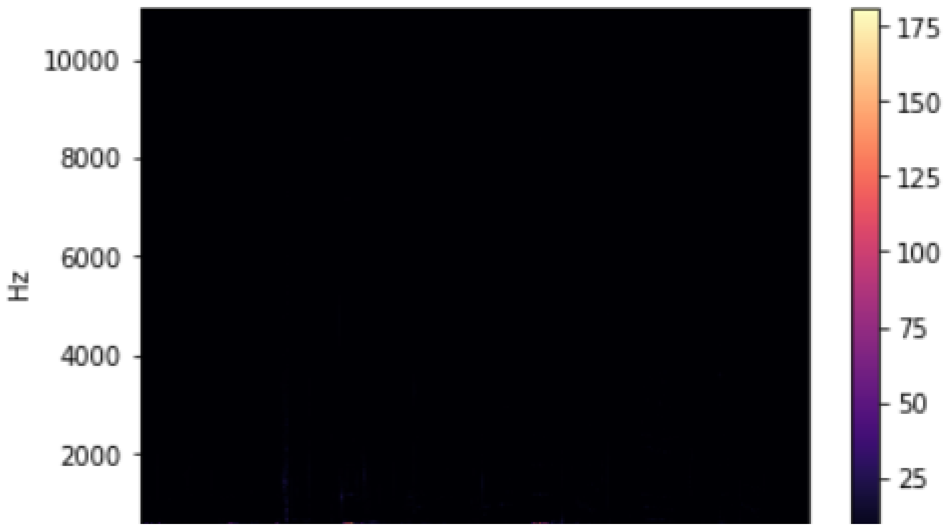
plot_fourier_transform_librosa.py hosted with ♡ by GitHub view raw



Now let's take the complete whale song, separate it to time windows, and apply the Fourier Transform on each time window.

```python
hop_length = 512
D = np.abs(librosa.stft(whale_song, n_fft=n_fft,  hop_length=hop_length))
librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='linear');
plt.colorbar();
```

plot_fourier_transform_full_song.py hosted with ♡ by GitHub view raw



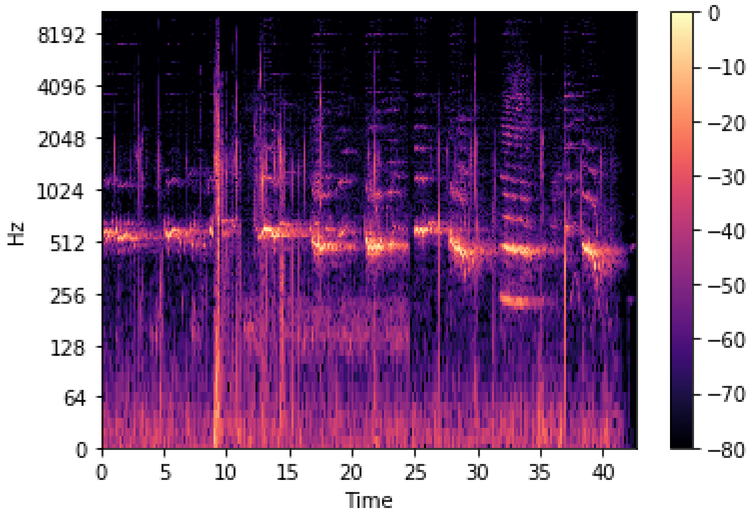
Wow can’t see much here can we? It’s because most sounds humans hear are concentrated in very small frequency and amplitude ranges.

My favorite frequency is 50,000 Hz
You've probably never heard it before
quickmeme.com

Let's make another small adjustment - transform both the y-axis (frequency) to log scale, and the "color" axis (amplitude) to Decibels, which is kinda the log scale of amplitudes.

```python
DB = librosa.amplitude_to_db(D, ref=np.max)
librosa.display.specshow(DB, sr=sr, hop_length=hop_length, x_axis='time', y_axis='log');
plt.colorbar(format='%+2.0f dB');
```

plot_spectrogram.py hosted with ♡ by GitHub view raw



Now this is what we call a Spectrogram!

The Mel Scale

Let's forget for a moment about all these lovely visualization and talk math. The Mel Scale, mathematically speaking, is the result of some non-linear transformation of the frequency scale. This Mel Scale is constructed such that sounds of equal distance from each other on the Mel Scale, also "sound" to humans as they are equal in distance from one another.
In contrast to Hz scale, where the difference between 500 and 1000 Hz is obvious, whereas the difference between 7500 and 8000 Hz is barely noticeable.

Luckily, someone computed this non-linear transformation for us, and all we need to do to apply it is use the appropriate command from librosa.

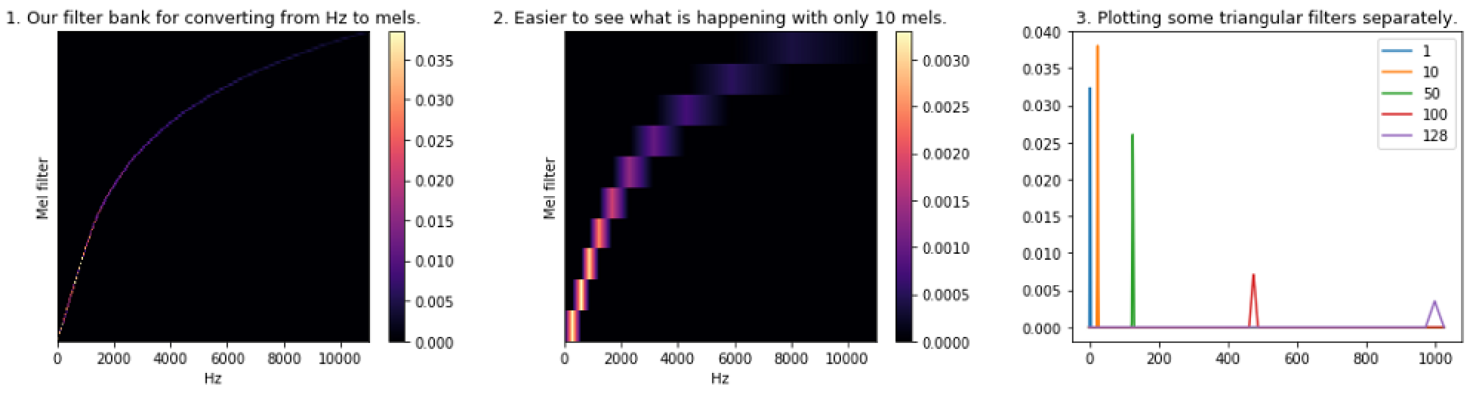
```python
n_mels = 128
mel = librosa.filters.mel(sr=sr, n_fft=n_fft, n_mels=n_mels)
```

define_mel.py hosted with ♡ by GitHub view raw

Yup. That's it.
But what does this give us?
It partitions the Hz scale into bins, and transforms each bin into a corresponding bin in the Mel Scale, using overlapping triangular filters.

```python
plt.figure(figsize=(15, 4));

plt.subplot(1, 3, 1);
librosa.display.specshow(mel, sr=sr, hop_length=hop_length, x_axis='linear');
plt.ylabel('Mel filter');
plt.colorbar();
plt.title('1. Our filter bank for converting from Hz to mels.');

plt.subplot(1, 3, 2);
mel_10 = librosa.filters.mel(sr=sr, n_fft=n_fft, n_mels=10)
librosa.display.specshow(mel_10, sr=sr, hop_length=hop_length, x_axis='linear');
plt.ylabel('Mel filter');
plt.colorbar();
plt.title('2. Easier to see what is happening with only 10 mels.');

plt.subplot(1, 3, 3);
idxs_to_plot = [0, 9, 49, 99, 127]
for i in idxs_to_plot:
    plt.plot(mel[i]);
plt.legend(labels=[f'{i+1}' for i in idxs_to_plot]);
plt.title('3. Plotting some triangular filters separately.');

plt.tight_layout();
```

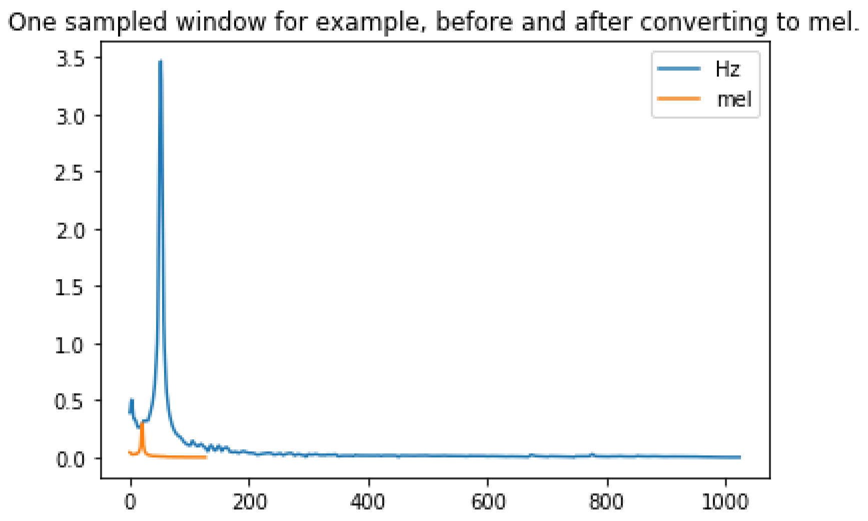
plot_mel_scale.py hosted with ♡ by GitHub view raw



Now what does *this* give us?
Now we can take the amplitude of one time window, compute the dot product with `mel` to perform the transformation, and get a visualization of the sound in this new frequency scale.

```python
plt.plot(D[:, 1]);
plt.plot(mel.dot(D[:, 1]));
plt.legend(labels=['Hz', 'mel']);
plt.title('One sampled window for example, before and after converting to mel.');
```

before_and_after_mel.py hosted with ♡ by GitHub   view raw



Mmmm, still doesn't mean much, he?
OK let's wrap it all up and see what we get.

The Mel Spectrogram

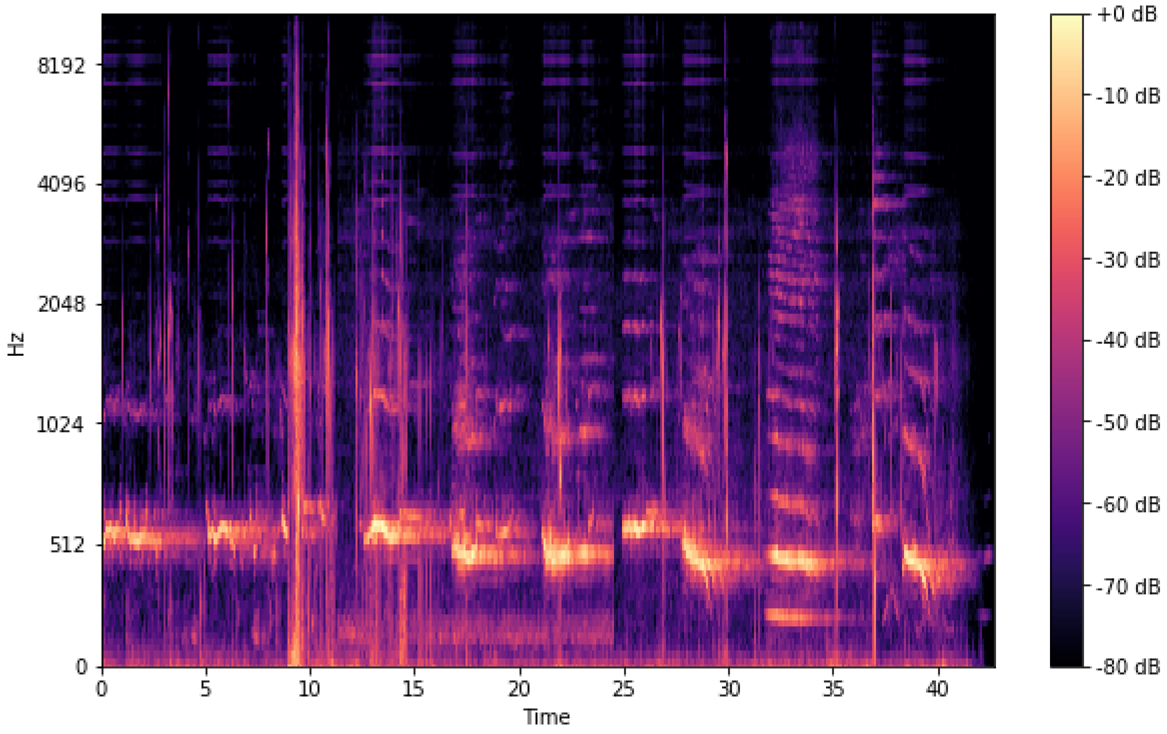
We know now what is a Spectrogram, and also what is the Mel Scale, so the Mel Spectrogram, is, rather surprisingly, a Spectrogram with the Mel Scale as its *y* axis.

And this is how you generate a Mel Spectrogram with one line of code, and display it nicely using just 3 more:

```python
S = librosa.feature.melspectrogram(whale_song, sr=sr, n_fft=n_fft, hop_length=hop_length, n_mels=
S_DB = librosa.power_to_db(S, ref=np.max)
librosa.display.specshow(S_DB, sr=sr, hop_length=hop_length, x_axis='time', y_axis='mel');
```

```python
plt.colorbar(format='%+2.0f dB');
```

plot_mel_spectrogram.py hosted with ♡ by GitHub view raw



Recap

The Mel Spectrogram is the result of the following pipeline:

1. **Separate to windows**: Sample the input with windows of size `n_fft=2048`, making hops of size `hop_length=512` each time to sample the next window.

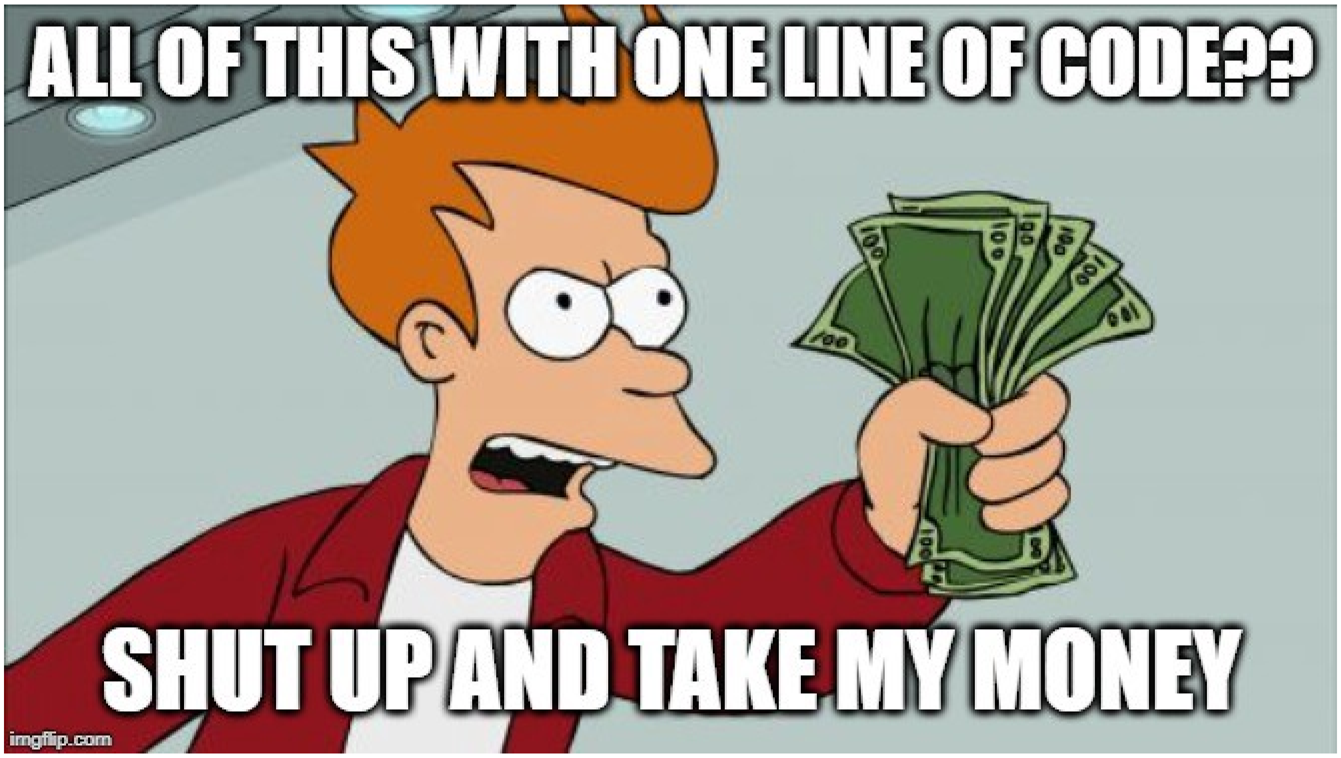
2. **Compute FFT** (Fast Fourier Transform) for each window to transform from time domain to frequency domain.

3. **Generate a Mel scale**: Take the entire frequency spectrum, and separate it into `n_mels=128` evenly spaced frequencies.
   And what do we mean by evenly spaced? not by distance on the frequency dimension, but distance as it is heard by the human ear.

4. **Generate Spectrogram**: For each window, decompose the magnitude of the signal into its components, corresponding to the frequencies in the mel scale.

```python
# Sanity check that indeed we understood the underlying pipeline
S = librosa.feature.melspectrogram(whale_song, sr=sr, n_fft=n_fft, hop_length=hop_length, n_mels=
fft_windows = librosa.stft(whale_song, n_fft=n_fft, hop_length=hop_length)
magnitude = np.abs(fft_windows)**2
mel = librosa.filters.mel(sr=sr, n_fft=n_fft, n_mels=n_mels)

assert (mel.dot(magnitude) == S).all()
```

sanity_check_for_mel_spectrogram.py hosted with ♡ by GitHub view raw


ALL OF THIS WITH ONE LINE OF CODE??
SHUT UP AND TAKE MY MONEY
imgflip.com

To Be Continued…

Now that we know Mel Spectrogram as well as Neo, what are we going to do with it?

Well, that’s for another post…

In the meanwhile, got any wild projects you are working on using the Mel Spectrogram? Please share in the comments!

Sign up for The Daily Pick

By Towards Data Science

Hands-on real-world examples, research, tutorials, and cutting-edge techniques delivered Monday to Thursday. Make learning your daily ritual. Take a look

Your email

Get this newsletter

By signing up, you will create a Medium account if you don't already have one.

Python Programming Music Audio Data Science

Medium

About Help Legal

Get the Medium app

Download on the App Store

GET IT ON Google Play

Me: Thanks. So Mel, when we first met, you were quite the enigma to me.
Mel: Really? How's that?

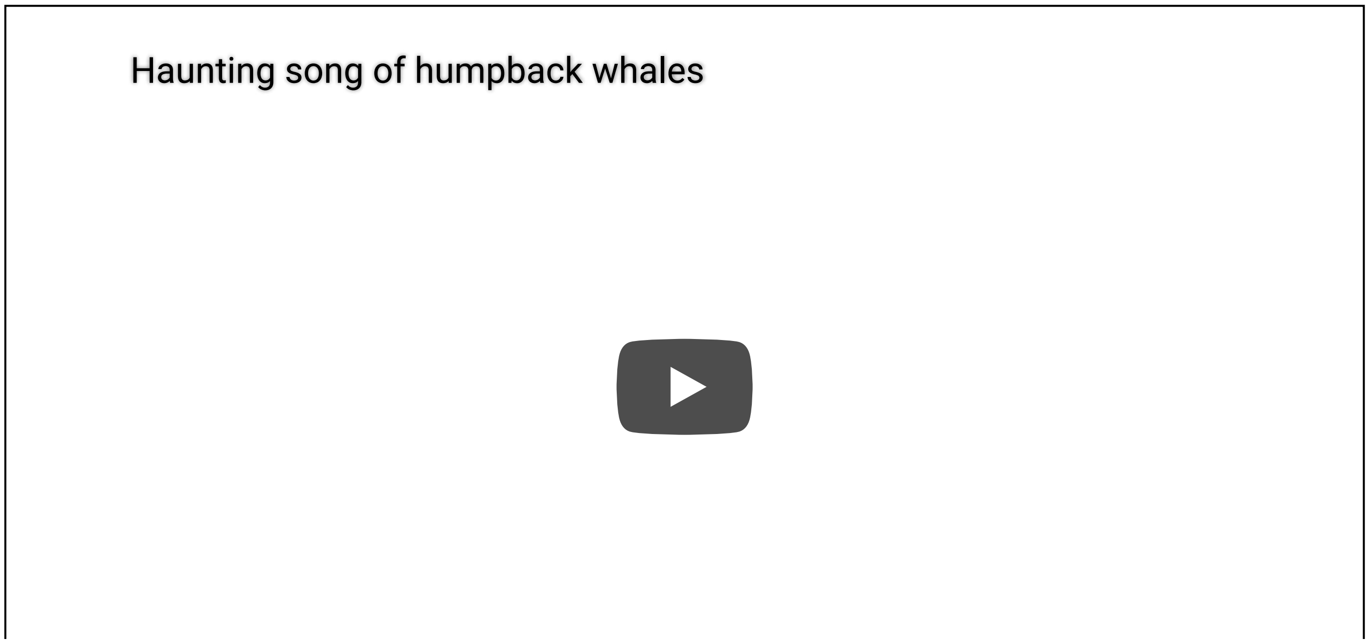
Me: You are composed of two concepts that their whole purpose is to make abstract notions accessible to humans - the **Mel Scale** and **Spectrogram** - yet you yourself were quite difficult for me, a human, to understand.
Mel: Is there a point to this one-sided speech?

Me: And do you know what bothered me even more? I heard through the grapevine that you are quite the *buzzz* in DSP *(Digital Signal Processing)*, yet I found very little intuitive information about you online.
Mel: Should I feel bad for you?

Me: So anyway, I didn't want to let you be misunderstood, so I decided to write about you.
Mel: Gee. That's actually kinda nice. Hope more people will get me now.

Me: With pleasure my friend. I think we can talk about what are your core elements, and then show some nice tricks using the **librosa** package on python.
Mel: Oooh that's great! I love librosa! It can generate me with one line of code!

Me: Wonderful! And let's use this beautiful whale song as our toy example throughout this post! What do you think?
Mel: You know you're talking to yourself right?



The Spectrogram

Visualizing sound is kind of a trippy concept. There are some mesmerizing ways to do that, and also more mathematical ones, which we will explore in this post.

Photo credit: Chelsea Davis. See more of this beautiful artwork here.

When we talk about *sound*, we generally talk about a sequence of vibrations in varying pressure strengths, so to visualize sound kinda means to visualize airwaves.