← Back to Index (index.html)

Jupyter Audio Basics

Audio Libraries

We will mainly use two libraries for audio acquisition and playback:

1. librosa

`librosa` (https://librosa.github.io/librosa/index.html) is a Python package for music and audio processing by Brian McFee (https://bmcfee.github.io/). A large portion was ported from Dan Ellis's Matlab audio processing examples (http://www.ee.columbia.edu/%7Edpwe/resources/matlab/).

2. IPython.display.Audio

`IPython.display.Audio` (http://ipython.org/ipython-doc/stable/api/generated/IPython.display.html#IPython.display.Audio) lets you play audio directly in an IPython notebook.

Included Audio Data

In progress.

This GitHub repository includes many short audio excerpts for your convenience.

Here are the files currently in the **`audio`** directory:

In [1]:

```
ls audio
```

```
conga_groove.wav  simple_loop.wav   tone_440.wav
```

Visit https://ccrma.stanford.edu/workshops/mir2014/audio/ (https://ccrma.stanford.edu/workshops/mir2014/audio/) for more audio files.

Reading Audio

Use **librosa.load** (https://librosa.github.io/librosa/generated/librosa.core.load.html#librosa.core.load) to load an audio file into an audio array. Return both the audio array as well as the sample rate:

In [2]:

```
import librosa
x, sr = librosa.load('audio/simple_loop.wav')
```

If you receive an error with **librosa.load**, you may need to install ffmpeg (https://librosa.github.io/librosa/install.html#ffmpeg).

Display the length of the audio array and sample rate:

In [3]:

```
print x.shape
print sr
```

```
(49613,)
22050
```

Visualizing Audio

In order to display plots inside the Jupyter notebook, run the following commands, preferably at the top of your notebook:

In [4]:

```
%matplotlib inline
import seaborn # optional
import matplotlib.pyplot as plt
import librosa.display
```

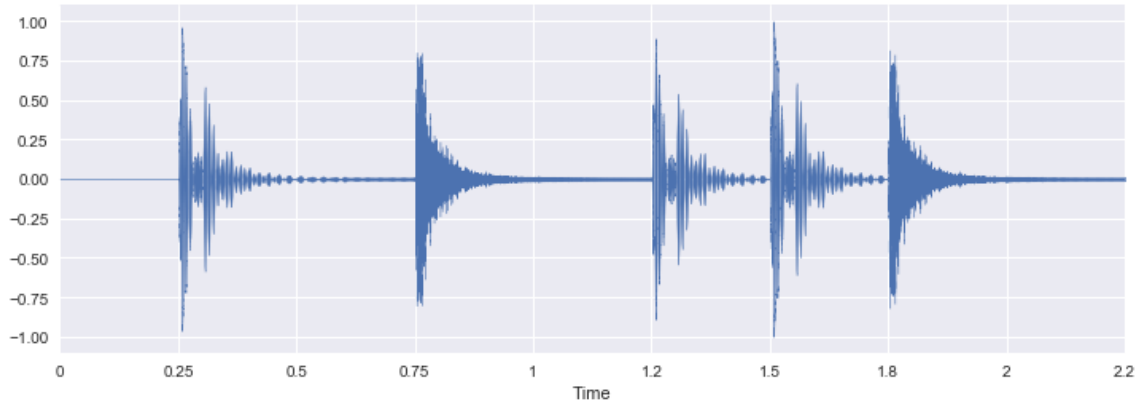
Plot the audio array using **librosa.display.waveplot** (https://librosa.github.io/librosa/generated/librosa.display.waveplot.html#librosa.display.wa

In [5]:

```
plt.figure(figsize=(12, 4))
librosa.display.waveplot(x, sr=sr)
```

Out[5]:

```
<matplotlib.collections.PolyCollection at 0x10dc8f210>
```



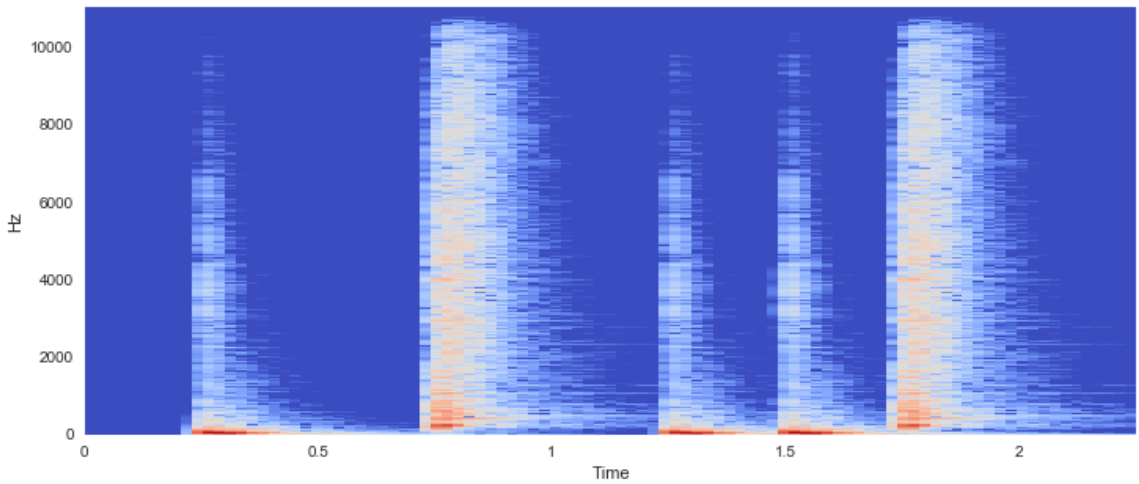
Display a spectrogram using **`librosa.display.specshow`** (https://librosa.github.io/librosa/generated/librosa.display.specshow.html):

In [6]:

```
X = librosa.stft(x)
Xdb = librosa.amplitude_to_db(X)
plt.figure(figsize=(12, 5))
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x119cc60d0>
```



Playing Audio

IPython.display.Audio

Using **`IPython.display.Audio`** (http://ipython.org/ipython-doc/2/api/generated/IPython.lib.display.html#IPython.lib.display.Audio), you can play an audio file:

In [7]:

```
import IPython.display as ipd
ipd.Audio('audio/conga_groove.wav') # load a local WAV file
```

Out[7]:

`Audio` can also accept a NumPy array. Let's synthesize a pure tone at 440 Hz:

In [8]:

```
import numpy
sr = 22050 # sample rate
T = 2.0    # seconds
t = numpy.linspace(0, T, int(T*sr), endpoint=False) # time variable
x = 0.5*numpy.sin(2*numpy.pi*440*t)                 # pure sine wave at 440 Hz
```

Listen to the audio array:

In [9]:

```
ipd.Audio(x, rate=sr) # load a NumPy array
```

Out[9]:

Writing Audio

`librosa.output.write_wav` (https://librosa.github.io/librosa/generated/librosa.output.write_wav.html#librosa.output.wri saves a NumPy array to a WAV file.

In [10]:

```
librosa.output.write_wav('audio/tone_440.wav', x, sr)
```

SoX

SoX (http://sox.sourceforge.net/) is a command-line utility for audio processing, e.g. convert file format, convert sample rate, remix, trim, pad, normalize, etc.

To install, for Mac:

```
brew install sox
```

To play or record audio from the command line:

```
rec test.wav
```

```
play test.wav
```

Resample to 22050 Hz:

```
sox in.wav -r 22050 out.wav
```

Normalize gain, e.g. to avoid clipping:

```
sox in.wav --norm out.wav
```

Down-mix all input channels to mono:

```
sox in.wav out.wav remix -
```

Convert to 16-bit signed integer:

```
sox in.wav -b 16 -e signed-integer out.wav
```

Trim the audio file starting at 1 min 15 sec and ending at 1 min 45 sec:

```
sox in.wav out.wav trim 1:15 =1:45
sox in.wav out.wav trim 1:15 0:30
sox in.wav out.wav trim 75 30
```

Pad 1 second of silence to the beginning and 2 seconds of silence to the end:

```
sox in.wav out.wav pad 1 2
```

← Back to Index (index.html)