pysox: Leveraging the audio signal processing power of SoX in Python

Rachel M Bittner*†, Eric Humphrey†, and Juan Pablo Bello*
*Music and Audio Research Laboratory, New York University
†Spotify Inc.

{rachel.bittner, jpbello}@nyu.edu
ejhumphrey@spotify.com

About SoX

- Open source, cross platform library for audio processing
- First released in 1991, in active development for more than 25 years
- Enables users to perform many common audio processing tasks:
  - Format conversion
  - Cutting/splicing
  - Resampling, channel conversion
  - Silence removal
  - Apply effects such as reverb, compression, EQ, etc.
- Useful in MIR workflows, especially dataset creation
- Often no useful default parameters - must read (lengthly) docs
- Clunky to integrate with Python workflows

Transformers

- Applies one or more effects or transformations to an input audio file and writes to a specified output file.
- **Example 1:** File conversion:

```
>>> tfm = sox.Transformer('path/to/
input_audio.wav', 'path/to/output/audio.aiff')
>>> tfm.build()
```

- **Example 2:** Effects chain:

```
# initialize transformer
>>> tfm = sox.Transformer(
    'path/to/input_audio.wav',
    'path/to/output/audio.mp3')

# change sample rate to 8000 Hz
>>> tfm.rate(samplerate=8000)

# normalize to -3 dB
>>> tfm.norm(db_level=-3)

# remove any long silences
>>> tfm.silence()

# trim the audio to the first 30 seconds
>>> tfm.trim(0, 30)

# add reverb with default parameters
>>> tfm.reverb()

# apply a fade in and fade out
>>> tfm.fade(fade_in_len=1.0, fade_out_len=0.5)

# create the output file
>>> tfm.build()
```

- `Transfomer` objects have a `.play()` method that allows users to hear a preview of what the audio sounds like with the current set of effects.

File Information

- Extract basic file information

```
>>> from sox import file_info
>>> file_info.duration('72sec_file.mp3')
72.0
>>> file_info.channels('stereo_file.wav')
2
>>> file_info.sample_encoding('music.ogg')
Vorbis
>>> file_info.silent('silent_file.aiff')
True
```

Combiners

- Used to combine a list of audio files into a single output file.
- Optional transformations can be applied to the combined output.
- **Example 3:** AM Modulation

```
>>> cbn = sox.Combiner(
     ['modulator.aiff', 'carrier.mp3'],
     'AM_modulated.wav', combine_type='multiply')
>>> cbn.build()
```

- **Example 4:** Mixing with effects

```
# Mixes file2.wav at half its original volume.
>>> cbn = sox.Combiner(
    ['file1.wav', 'file2.wav', 'file3.mp3'],
    'output.wav', combine_type='mix',
    input_volumes=[1, 0.5, 1])

# low pass filter the mixed audio
>>> cbn.lowpass(frequency=2000)

# reverse the audio
>>> cbn.reverse()

# create the output file
>>> cbn.build()
```

Download & Resources

- pysox on github:
  - http://www.github.com/rabitt/pysox
- SoX
  - http://sox.sourceforge.net/
- Install - Mac (with Homebrew and pip)

```
$ brew install sox
$ pip install sox
```

- Install - Linux

```
$ apt-get install sox
$ pip install sox
```

NEW YORK UNIVERSITY

ISMIR 2016