librosa 常用功能
核心音频处理函数

这部分介绍了最常用的音频处理函数，包括音频读取函数 load( )，重采样函数 resample( )，短时傅里叶变换 stft( )，幅度转换函数 amplitude_to_db( )以及频率转换函数 hz_to_mel( )等。这部分函数很多，详细可参考 librosa 官网 http://librosa.github.io/ librosa/core.html

音频处理

`load` (path[, sr, mono, offset, duration, ...])	Load an audio file as a floating point time series.
`to_mono` (y)	Force an audio signal down to mono.
`resample` (y, orig_sr, target_sr[, res_type, ...])	Resample a time series from orig_sr to target_sr
`get_duration` ([y, sr, S, n_fft, hop_length, ...])	Compute the duration (in seconds) of an audio time series,
`autocorrelate` (y[, max_size, axis])	Bounded auto-correlation
`zero_crossings` (y[, threshold, ...])	Find the zero-crossings of a signal *y*: indices *i* such that *sig*
`clicks` ([times, frames, sr, hop_length, ...])	Returns a signal with the signal `click` placed at each spec

频谱表示

`stft` (y[, n_fft, hop_length, win_length, ...])	Short-time Fourier transform (STFT)
`istft` (stft_matrix[, hop_length, win_length, ...])	Inverse short-time Fourier transform (ISTFT).
`ifgram` (y[, sr, n_fft, hop_length, ...])	Compute the instantaneous frequency (as a proportion
`cqt` (y[, sr, hop_length, fmin, n_bins, ...])	Compute the constant-Q transform of an audio signal.
`icqt` (C[, sr, hop_length, fmin, ...])	Compute the inverse constant-Q transform.
`hybrid_cqt` (y[, sr, hop_length, fmin, ...])	Compute the hybrid constant-Q transform of an audio
`pseudo_cqt` (y[, sr, hop_length, fmin, ...])	Compute the pseudo constant-Q transform of an audio
`iirt` (y[, sr, win_length, hop_length, ...])	Time-frequency representation using IIR filters [R99].
`fmt` (y[, t_min, n_fmt, kind, beta, ...])	The fast Mellin transform (FMT) [R1112] of a uniformly
`interp_harmonics` (x, freqs, h_range[, kind, ...])	Compute the energy at harmonics of time-frequency re
`salience` (S, freqs, h_range[, weights, ...])	Harmonic salience function.
`phase_vocoder` (D, rate[, hop_length])	Phase vocoder.
`magphase` (D[, power])	Separate a complex-valued spectrogram D into its magn

幅度转换

`amplitude_to_db`(S[, ref, amin, top_db])	Convert an amplitude spectrogram to dB-scaled spectr
`db_to_amplitude`(S_db[, ref])	Convert a dB-scaled spectrogram to an amplitude spec
`power_to_db`(S[, ref, amin, top_db])	Convert a power spectrogram (amplitude squared) to d
`db_to_power`(S_db[, ref])	Convert a dB-scale spectrogram to a power spectrogra
`perceptual_weighting`(S, frequencies, **kwargs)	Perceptual weighting of a power spectrogram:
`A_weighting`(frequencies[, min_db])	Compute the A-weighting of a set of frequencies.

时频转换

`frames_to_samples`(frames[, hop_length, n_fft])	Converts frame indices to audio sample indices
`frames_to_time`(frames[, sr, hop_length, n_fft])	Converts frame counts to time (seconds)
`samples_to_frames`(samples[, hop_length, n_fft])	Converts sample indices into STFT frames.
`samples_to_time`(samples[, sr])	Convert sample indices to time (in seconds).
`time_to_frames`(times[, sr, hop_length, n_fft])	Converts time stamps into STFT frames.
`time_to_samples`(times[, sr])	Convert timestamps (in seconds) to sample indices.
`hz_to_note`(frequencies, **kwargs)	Convert one or more frequencies (in Hz) to the nearest
`hz_to_midi`(frequencies)	Get MIDI note number(s) for given frequencies
`midi_to_hz`(notes)	Get the frequency (Hz) of MIDI note(s)
`midi_to_note`(midi[, octave, cents])	Convert one or more MIDI numbers to note strings.
`note_to_hz`(note, **kwargs)	Convert one or more note names to frequency (Hz)
`note_to_midi`(note[, round_midi])	Convert one or more spelled notes to MIDI number(s).
`hz_to_mel`(frequencies[, htk])	Convert Hz to Mels
`hz_to_octs`(frequencies[, A440])	Convert frequencies (Hz) to (fractional) octave number
`mel_to_hz`(mels[, htk])	Convert mel bin numbers to frequencies
`octs_to_hz`(octs[, A440])	Convert octaves numbers to frequencies.
`fft_frequencies`([sr, n_fft])	Alternative implementation of *np.fft.fftfreqs*
`cqt_frequencies`(n_bins, fmin[, ...])	Compute the center frequencies of Constant-Q bins.

特征提取

本部分列举了一些常用的频谱特征的提取方法，包括常见的 Mel Spectrogram、MFCC、CQT 等。函数详细信息可参考 http:// librosa.github.io/librosa/feature.html

`chroma_stft` ([y, sr, S, norm, n_fft, ...])	Compute a chromagram from a waveform or power spectrogram.
`chroma_cqt` ([y, sr, C, hop_length, fmin, ...])	Constant-Q chromagram
`chroma_cens` ([y, sr, C, hop_length, fmin, ...])	Computes the chroma variant "Chroma Energy Normalized" (CENS), following [R3131].
`melspectrogram` ([y, sr, S, n_fft, ...])	Compute a mel-scaled spectrogram.
`mfcc` ([y, sr, S, n_mfcc])	Mel-frequency cepstral coefficients
`rmse` ([y, S, frame_length, hop_length, ...])	Compute root-mean-square (RMS) energy for each frame, either from the audio samples *y* or from a spectrogram *S*.
`spectral_centroid` ([y, sr, S, n_fft, ...])	Compute the spectral centroid.
`spectral_bandwidth` ([y, sr, S, n_fft, ...])	Compute p'th-order spectral bandwidth:
`spectral_contrast` ([y, sr, S, n_fft, ...])	Compute spectral contrast [R3333]
`spectral_flatness` ([y, S, n_fft, hop_length, ...])	Compute spectral flatness
`spectral_rolloff` ([y, sr, S, n_fft, ...])	Compute roll-off frequency
`poly_features` ([y, sr, S, n_fft, hop_length, ...])	Get coefficients of fitting an nth-order polynomial to the columns of a spectrogram.
`tonnetz` ([y, sr, chroma])	Computes the tonal centroid features (tonnetz), following the method of [R3737].
`zero_crossing_rate` (y[, frame_length, ...])	Compute the zero-crossing rate of an audio time series.

绘图显示

包含了常用的频谱显示函数 specshow( ), 波形显示函数 waveplot( )，详细信息请参考 http://librosa.github.io/librosa/display. html

`specshow` (data[, x_coords, y_coords, x_axis, ...])	Display a spectrogram/chromagram/cqt/etc.
`waveplot` (y[, sr, max_points, x_axis, ...])	Plot the amplitude envelope of a waveform.
`cmap` (data[, robust, cmap_seq, cmap_bool, ...])	Get a default colormap from the given data.
`TimeFormatter` ([lag])	A tick formatter for time axes.
`NoteFormatter` ([octave, major])	Ticker formatter for Notes
`LogHzFormatter` ([major])	Ticker formatter for logarithmic frequency
`ChromaFormatter`	A formatter for chroma axes
`TonnetzFormatter`	A formatter for tonnetz axes

三、常用功能代码实现

读取音频

```
>>> import librosa
>>> # Load a wav file
>>> y, sr = librosa.load('./beat.wav')
>>> y
array([  0.00000000e+00,   0.00000000e+00,   0.00000000e+00, ...,
         8.12290182e-06,   1.34394732e-05,   0.00000000e+00], dtype=float32)
>>> sr
22050
```

Librosa 默认的采样率是 22050，如果需要读取原始采样率，需要设定参数 sr=None:

```
>>> import librosa
>>> # Load a wav file
>>> y, sr = librosa.load('./beat.wav', sr=None)
>>> sr
44100
```

可见，'beat.wav'的原始采样率为 44100。如果需要重采样，只需要将采样率参数 sr 设定为你需要的值：

```
>>> import librosa
>>> # Load a wav file
>>> y, sr = librosa.load('./beat.wav', sr=16000)
>>> sr
16000
```

提取特征

提取 Log-Mel Spectrogram 特征

Log-Mel Spectrogram 特征是目前在语音识别和环境声音识别中很常用的一个特征，由于 CNN 在处理图像上展现了强大的能力，使得音频信号的频谱图特征的使用愈加广泛，甚至比 MFCC 使用的更多。在 librosa 中，Log-Mel Spectrogram 特征的提取只需几行代码：

```
>>> import librosa
>>> # Load a wav file
>>> y, sr = librosa.load('./beat.wav', sr=None)
>>> # extract mel spectrogram feature
>>> melspec = librosa.feature.melspectrogram(y, sr, n_fft=1024, hop_length=512, n_mels=128)
>>> # convert to log scale
>>> logmelspec = librosa.power_to_db(melspec)
>>> logmelspec.shape
(128, 194)
```

可见，Log-Mel Spectrogram 特征是二维数组的形式，128 表示 Mel 频率的维度（频域），194 为时间帧长度（时域），所以 Log-Mel Spectrogram 特征是音频信号的时频表示特征。其中，n_fft 指的是窗的大小，这里为 1024；hop_length 表示相邻窗之间的距离，这里为 512，也就是相邻窗之间有 50%的 overlap；n_mels 为 mel bands 的数量，这里设为 128。

提取 MFCC 特征

MFCC 特征是一种在自动语音识别和说话人识别中广泛使用的特征。关于 MFCC 特征的详细信息，有兴趣的可以参考博客 http:// blog.csdn.net/zzc15806/article/details/79246716。在 librosa 中，提取 MFCC 特征只需要一个函数：

```
>>> import librosa
>>> # Load a wav file
>>> y, sr = librosa.load('./beat.wav', sr=None)
>>> # extract mfcc feature
>>> mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=40)
>>> mfccs.shape
(40, 194)
```

关于 mfcc，这里就不在赘述。

Librosa 还有很多其他音频特征的提取方法，比如 CQT 特征、chroma 特征等，在第二部分“librosa 常用功能”给了详细的介绍。

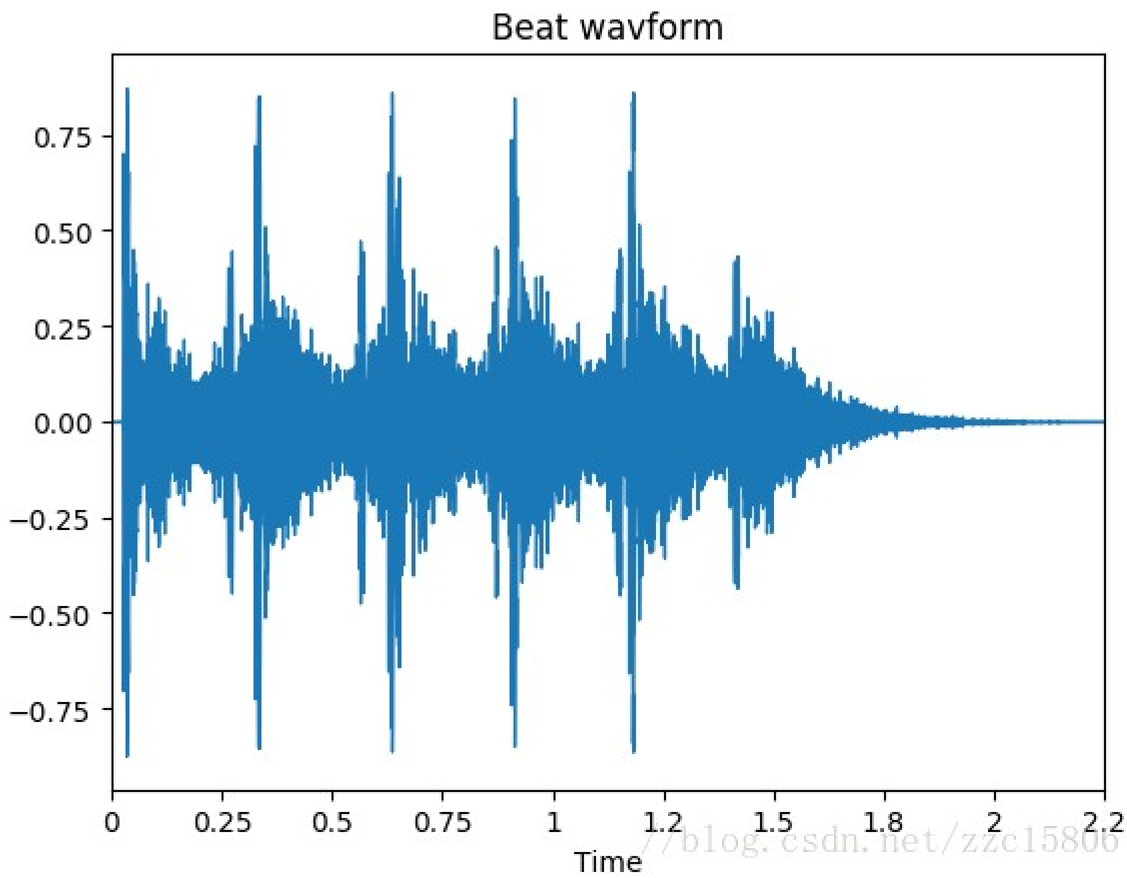
绘图显示

绘制声音波形

Librosa 有显示声音波形函数 waveplot( )：

```
>>> import librosa
>>> import librosa.display
>>> # Load a wav file
>>> y, sr = librosa.load('./beat.wav', sr=None)
>>> # plot a wavform
>>> plt.figure()
>>> librosa.display.waveplot(y, sr)
>>> plt.title('Beat wavform')
>>> plt.show()
```

输出图形为：

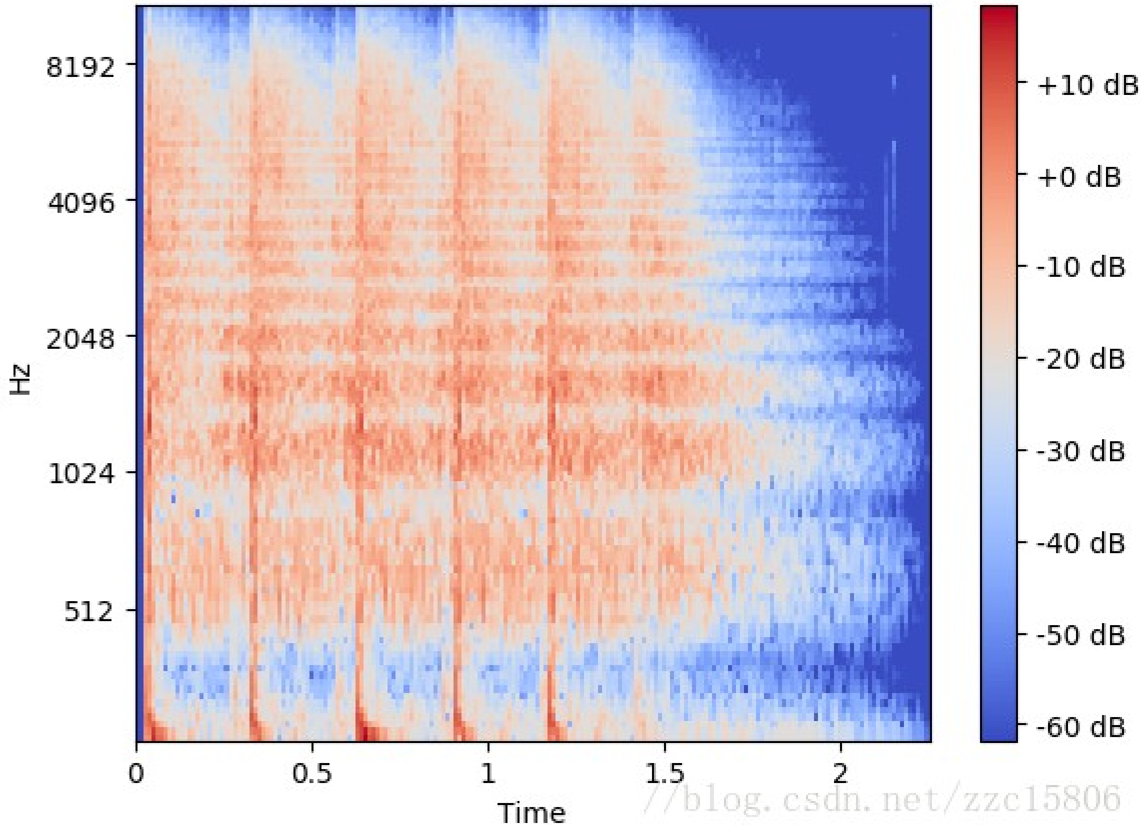


绘制频谱图

Librosa 有显示频谱图波形函数 specshow( ):

```
>>> import librosa
>>> import librosa.display
>>> # Load a wav file
>>> y, sr = librosa.load('./beat.wav', sr=None)
>>> # extract mel spectrogram feature
>>> melspec = librosa.feature.melspectrogram(y, sr, n_fft=1024, hop_length=512, n_mels=128)
>>> # convert to log scale
>>> logmelspec = librosa.power_to_db(melspec)
>>> # plot mel spectrogram
>>> plt.figure()
>>> librosa.display.specshow(logmelspec, sr=sr, x_axis='time', y_axis='mel')
>>> plt.title('Beat wavform')
>>> plt.show()
```

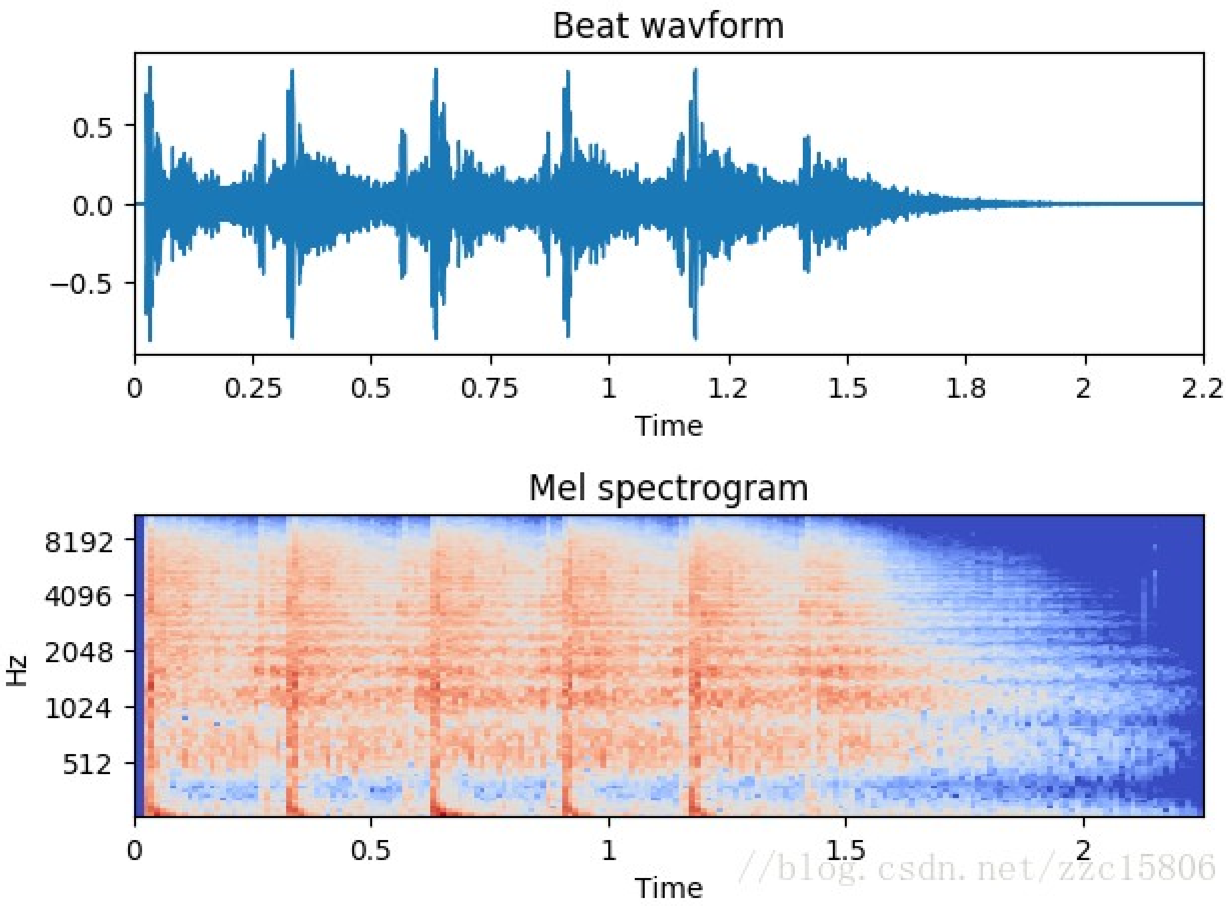
输出结果为：



将声音波形和频谱图绘制在一张图表中：

```
>>> import librosa
>>> import librosa.display
>>> # Load a wav file
>>> y, sr = librosa.load('./beat.wav', sr=None)
>>> # extract mel spectrogram feature
>>> melspec = librosa.feature.melspectrogram(y, sr, n_fft=1024, hop_length=512, n_mels=128)
>>> # convert to log scale
>>> logmelspec = librosa.power_to_db(melspec)
>>> plt.figure()
>>> # plot a wavform
>>> plt.subplot(2, 1, 1)
>>> librosa.display.waveplot(y, sr)
>>> plt.title('Beat wavform')
>>> # plot mel spectrogram
>>> plt.subplot(2, 1, 2)
>>> librosa.display.specshow(logmelspec, sr=sr, x_axis='time', y_axis='mel')
>>> plt.title('Mel spectrogram')
>>> plt.tight_layout() #保证图不重叠
>>> plt.show()
```

输出结果为：



到这里，librosa 的安装和简单使用就介绍完了。事实上，librosa 远不止这些功能，关于 librosa 更多的使用方法还请大家参考 librosa 官网 http://librosa.github.io/librosa/index.html

参考：http://librosa.github.io/librosa/index.html

Librosa 是一个用于音乐和音频分析的 python 包，如果没学过《数字信号处理》需要先了解一下相关的基础知识，傅立叶变换，梅尔频率倒谱

安装：pip install librosa

环境：Python3.6

我们先做个简单的变声

```
import librosa
y,sr = librosa.load("/Users/birenjianmo/Desktop/learn/librosa/mp3/in.wav")
# 通过改变采样率来改变音速，相当于播放速度 X2
librosa.output.write_wav("resample.wav",y,sr*2)

import librosa
y,sr = librosa.load("/Users/birenjianmo/Desktop/learn/librosa/mp3/in.wav")
# 通过移动音调变声 ，14 是上移 14 个半步， 如果是 -14 下移 14 个半步
b = librosa.effects.pitch_shift(y, sr, n_steps=14)
librosa.output.write_wav("pitch_shift.wav",b,sr)
```

复杂的变声

```
import librosa
import matplotlib.pyplot as plt
import numpy as np
y,sr = librosa.load("/Users/birenjianmo/Desktop/learn/librosa/mp3/in.wav")


# stft 短时傅立叶变换
a = librosa.stft(y)
length = len(a)

# 改变或去除某些值，可以改变声音
r_a = a[10:length-10]

# istft 逆短时傅立叶变换，变回去
b = librosa.istft(r_a)

librosa.output.write_wav("stft.wav",b,sr)

# 以下是显示频谱图
fig = plt.figure()
s1 = fig.add_subplot(3,1,1)
s2 = fig.add_subplot(3,1,2)
s3 = fig.add_subplot(3,1,3)

s1.plot(y)
```

```
s2.plot(a)
s3.plot(b)

plt.show()
```

变音的主要算法原理

最简单的是：通过对语音的采样率进行变化，就能改变声音，但是不易用参数进行控制。

别外一种是：提取反应该个性的参数，如，男人、女人；小孩和老人，因声道的长度不一样，导致其基音不一样，进而导致各谐振峰不一样。我们可能通过改变基音和谐振峰的位置来改变声音。

男女声变调必须是进行频谱搬移,在信号处理上通常是乘一个余弦函数
下面是男女声的频谱范围:
男低音:82--330　　　女 175--699
男中音;98--392　　　　220--880
男高音;124--494　　　　262--1047
单位为 hz

上述转自：https://blog.csdn.net/jinbing/article/details/5199605
