librosa (https://nbviewer.jupyter.org/github/librosa/librosa/tree/master)
/ examples (https://nbviewer.jupyter.org/github/librosa/librosa/tree/master/examples)

# Librosa demo

This notebook demonstrates some of the basic functionality of librosa version 0.4.

Following through this example, you'll learn how to:

- Load audio input
- Compute mel spectrogram, MFCC, delta features, chroma
- Locate beat events
- Compute beat-synchronous features
- Display features

In [1]:
```
from __future__ import print_function
```

In [2]:
```
# We'll need numpy for some mathematical operations
import numpy as np

# matplotlib for displaying the output
import matplotlib.pyplot as plt
%matplotlib inline

# and IPython.display for audio output
import IPython.display

# Librosa for audio
import librosa
# And the display module for visualization
import librosa.display
```

In [3]:
```
audio_path = librosa.util.example_audio_file()

# or uncomment the line below and point it at your favorite song:
#
# audio_path = '/path/to/your/favorite/song.mp3'

y, sr = librosa.load(audio_path)
```

By default, librosa will resample the signal to 22050Hz.

You can change this behavior by saying:

```
librosa.load(audio_path, sr=44100)
```

to resample at 44.1KHz, or

```
librosa.load(audio_path, sr=None)
```

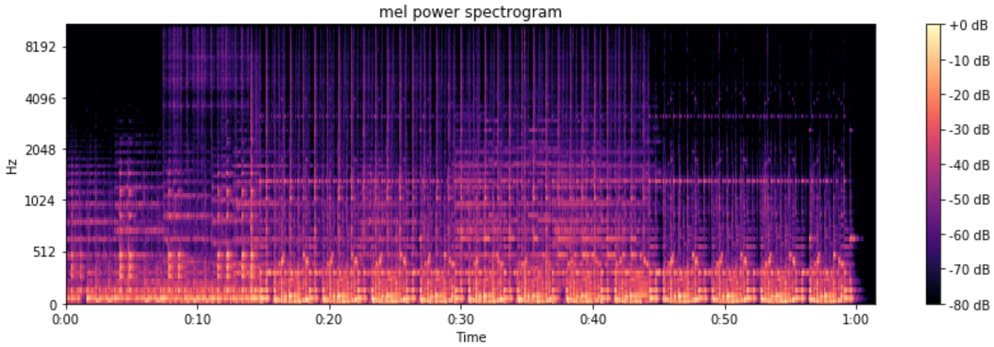
to disable resampling.

File failed to load: file:///home/wb/2019/b/book/my_book/librosa/b_librosa/demo_officialP_Jupyter%20Notebook%20Viewer_files/extensions/MathMenu

# Mel spectrogram

This first step will show how to compute a Mel (https://en.wikipedia.org/wiki/Mel_scale) spectrogram from an audio waveform.

In [4]:
```
# Let's make and display a mel-scaled power (energy-squared) spectrog
S = librosa.feature.melspectrogram(y, sr=sr, n_mels=128)

# Convert to log scale (dB). We'll use the peak power (max) as refere
log_S = librosa.power_to_db(S, ref=np.max)

# Make a new figure
plt.figure(figsize=(12,4))

# Display the spectrogram on a mel scale
# sample rate and hop length parameters are used to render the time a
librosa.display.specshow(log_S, sr=sr, x_axis='time', y_axis='mel')

# Put a descriptive title on the plot
plt.title('mel power spectrogram')

# draw a color bar
plt.colorbar(format='%+02.0f dB')

# Make the figure layout compact
plt.tight_layout()
```




# Harmonic-percussive source separation

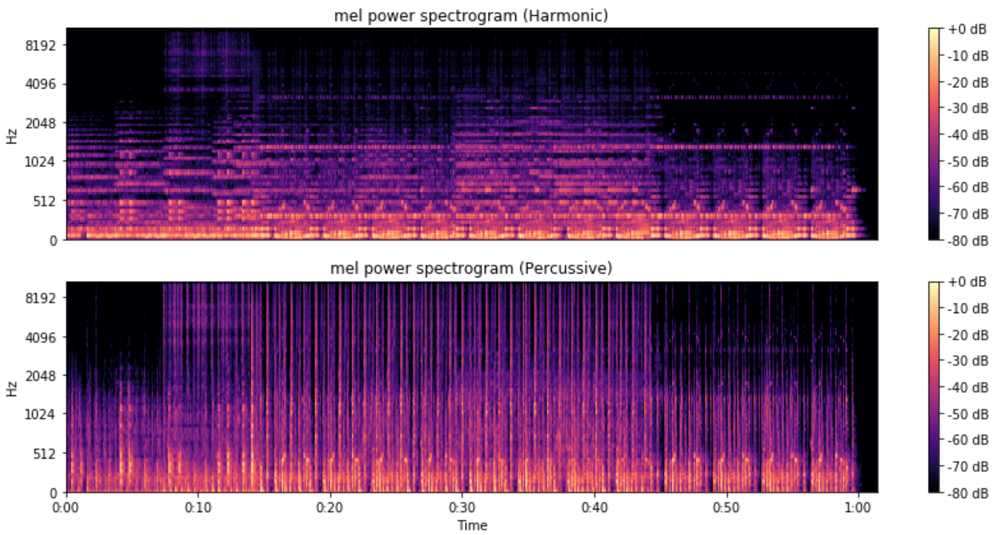
Before doing any signal analysis, let's pull apart the harmonic and percussive components of the audio. This is pretty easy to do with the `effects` module.

In [5]:
```
y_harmonic, y_percussive = librosa.effects.hpss(y)
```

File failed to load: file:///home/wb/2019/b/book/my_book/librosa/b_librosa/demo_officialP_Jupyter%20Notebook%20Viewer_files/extensions/MathMenu

In [6]:

```
# What do the spectrograms look like?
# Let's make and display a mel-scaled power (energy-squared) spectrog
S_harmonic   = librosa.feature.melspectrogram(y_harmonic, sr=sr)
S_percussive = librosa.feature.melspectrogram(y_percussive, sr=sr)

# Convert to log scale (dB). We'll use the peak power as reference.
log_Sh = librosa.power_to_db(S_harmonic, ref=np.max)
log_Sp = librosa.power_to_db(S_percussive, ref=np.max)

# Make a new figure
plt.figure(figsize=(12,6))

plt.subplot(2,1,1)
# Display the spectrogram on a mel scale
librosa.display.specshow(log_Sh, sr=sr, y_axis='mel')

# Put a descriptive title on the plot
plt.title('mel power spectrogram (Harmonic)')

# draw a color bar
plt.colorbar(format='%+02.0f dB')

plt.subplot(2,1,2)
librosa.display.specshow(log_Sp, sr=sr, x_axis='time', y_axis='mel')

# Put a descriptive title on the plot
plt.title('mel power spectrogram (Percussive)')

# draw a color bar
plt.colorbar(format='%+02.0f dB')

# Make the figure layout compact
plt.tight_layout()
```



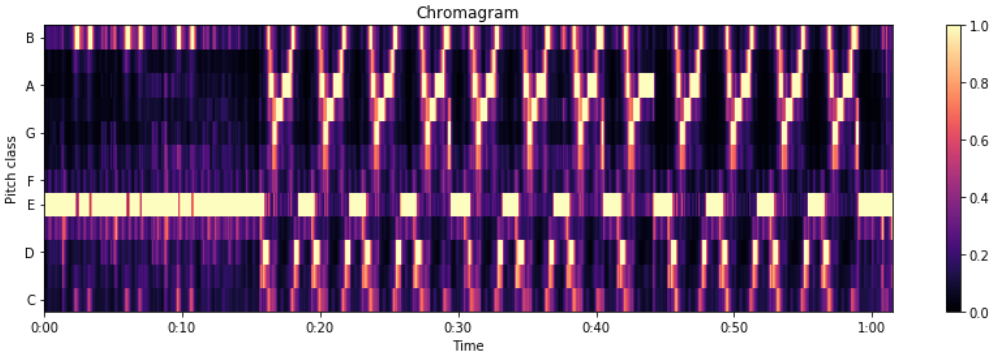

# Chromagram

Next, we'll extract Chroma (http://en.wikipedia.org/wiki/Pitch_class) features to represent pitch class information.

File failed to load: file:///home/wb/2019/b/book/my_book/librosa/b_librosa/demo_officialP_Jupyter%20Notebook%20Viewer_files/extensions/MathMenu

In [8]:

```
# We'll use a CQT-based chromagram with 36 bins-per-octave in the CQT
# We'll use the harmonic component to avoid pollution from transients
C = librosa.feature.chroma_cqt(y=y_harmonic, sr=sr, bins_per_octave=3

# Make a new figure
plt.figure(figsize=(12,4))

# Display the chromagram: the energy in each chromatic pitch class as
# To make sure that the colors span the full range of chroma values,
librosa.display.specshow(C, sr=sr, x_axis='time', y_axis='chroma', vm

plt.title('Chromagram')
plt.colorbar()

plt.tight_layout()
```

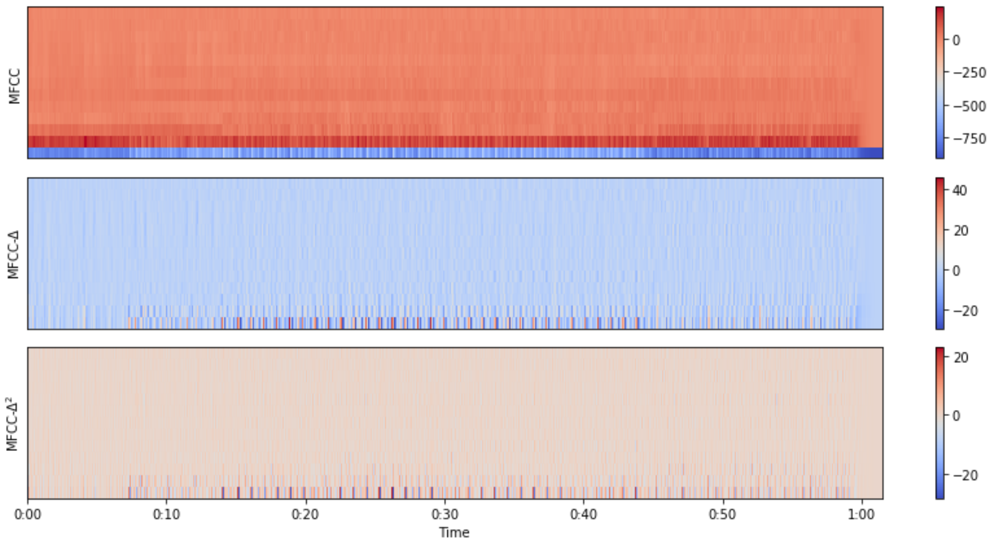



# MFCC

Mel-frequency cepstral coefficients (http://en.wikipedia.org/wiki/Mel-frequency_cepstrum) are commonly used to represent texture or timbre of sound.

File failed to load: file:///home/wb/2019/b/book/my_book/librosa/b_librosa/demo_officialP_Jupyter%20Notebook%20Viewer_files/extensions/MathMenu

In [9]:
```
# Next, we'll extract the top 13 Mel-frequency cepstral coefficients
mfcc        = librosa.feature.mfcc(S=log_S, n_mfcc=13)

# Let's pad on the first and second deltas while we're at it
delta_mfcc  = librosa.feature.delta(mfcc)
delta2_mfcc = librosa.feature.delta(mfcc, order=2)

# How do they look?  We'll show each in its own subplot
plt.figure(figsize=(12, 6))

plt.subplot(3,1,1)
librosa.display.specshow(mfcc)
plt.ylabel('MFCC')
plt.colorbar()

plt.subplot(3,1,2)
librosa.display.specshow(delta_mfcc)
plt.ylabel('MFCC-$\Delta$')
plt.colorbar()

plt.subplot(3,1,3)
librosa.display.specshow(delta2_mfcc, sr=sr, x_axis='time')
plt.ylabel('MFCC-$\Delta^2$')
plt.colorbar()

plt.tight_layout()

# For future use, we'll stack these together into one matrix
M = np.vstack([mfcc, delta_mfcc, delta2_mfcc])
```




## Beat tracking

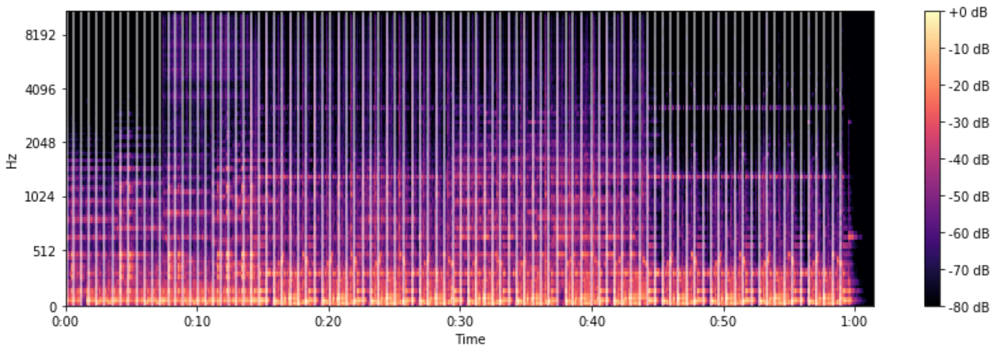
The beat tracker returns an estimate of the tempo (in beats per minute) and frame indices of beat events.

The input can be either an audio time series (as we do below), or an onset strength envelope as calculated by `librosa.onset.onset_strength()` .

File failed to load: file:///home/wb/2019/b/book/my_book/librosa/b_librosa/demo_officialP_Jupyter%20Notebook%20Viewer_files/extensions/MathMenu

In [11]:
```
# Now, let's run the beat tracker.
# We'll use the percussive component for this part
plt.figure(figsize=(12, 6))
tempo, beats = librosa.beat.beat_track(y=y_percussive, sr=sr)

# Let's re-draw the spectrogram, but this time, overlay the detected
plt.figure(figsize=(12,4))
librosa.display.specshow(log_S, sr=sr, x_axis='time', y_axis='mel')

# Let's draw transparent lines over the beat frames
plt.vlines(librosa.frames_to_time(beats),
           1, 0.5 * sr,
           colors='w', linestyles='-', linewidth=2, alpha=0.5)

plt.axis('tight')

plt.colorbar(format='%+02.0f dB')

plt.tight_layout();
```

```
<Figure size 864x432 with 0 Axes>
```




By default, the beat tracker will trim away any leading or trailing beats that don't appear strong enough.

To disable this behavior, call `beat_track()` with `trim=False` .

In [12]:
```
print('Estimated tempo:        %.2f BPM' % tempo)

print('First 5 beat frames:   ', beats[:5])

# Frame numbers are great and all, but when do those beats occur?
print('First 5 beat times:    ', librosa.frames_to_time(beats[:5], sr

# We could also get frame numbers from times by librosa.time_to_frame
```

```
Estimated tempo:        103.36 BPM
First 5 beat frames:    [  4  28  53  77 102]
First 5 beat times:     [0.09287982 0.65015873 1.2306576  1.78793651
```

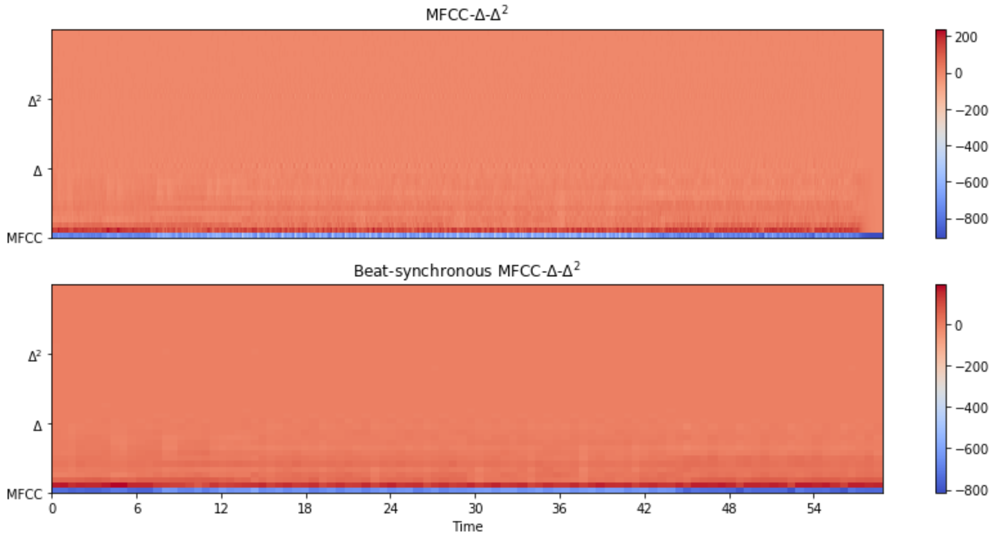
File failed to load: file:///home/wb/2019/b/book/my_book/librosa/b_librosa/demo_officialP_Jupyter%20Notebook%20Viewer_files/extensions/MathMenu

# Beat-synchronous feature aggregation

Once we've located the beat events, we can use them to summarize the feature content of each beat.

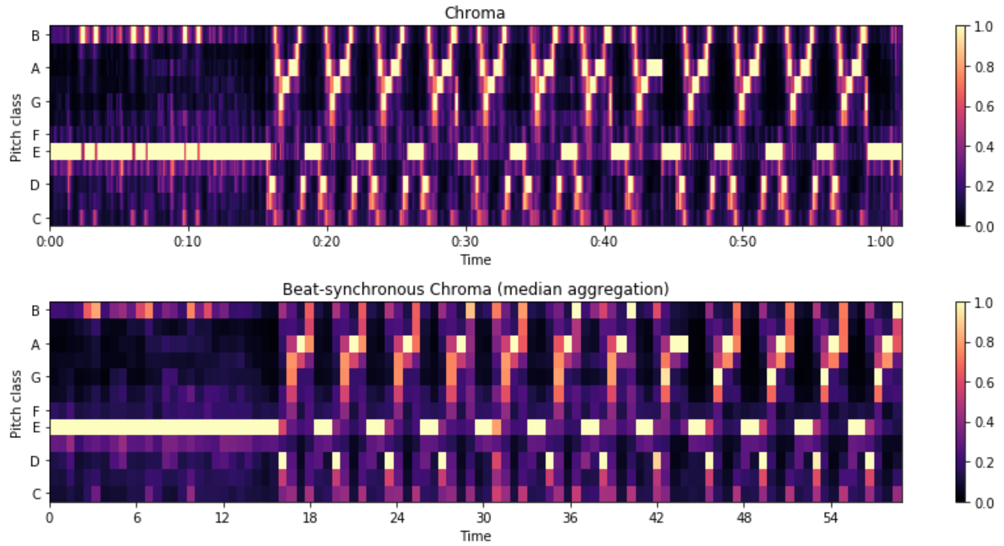
This can be useful for reducing data dimensionality, and removing transient noise from the features.

In [14]:
```
# feature.sync will summarize each beat event by the mean feature vec

M_sync = librosa.util.sync(M, beats)

plt.figure(figsize=(12,6))

# Let's plot the original and beat-synchronous features against each
plt.subplot(2,1,1)
librosa.display.specshow(M)
plt.title('MFCC-$\Delta$-$\Delta^2$')

# We can also use pyplot *ticks directly
# Let's mark off the raw MFCC and the delta features
plt.yticks(np.arange(0, M.shape[0], 13), ['MFCC', '$\Delta$', '$\Delt

plt.colorbar()

plt.subplot(2,1,2)
# librosa can generate axis ticks from arbitrary timestamps and beat
librosa.display.specshow(M_sync, x_axis='time',
                         x_coords=librosa.frames_to_time(librosa.util

plt.yticks(np.arange(0, M_sync.shape[0], 13), ['MFCC', '$\Delta$', '$
plt.title('Beat-synchronous MFCC-$\Delta$-$\Delta^2$')
plt.colorbar()

plt.tight_layout()
```




File failed to load: file:///home/wb/2019/b/book/my_book/librosa/b_librosa/demo_officialP_Jupyter%20Notebook%20Viewer_files/extensions/MathMenu

In [15]:
```
# Beat synchronization is flexible.
# Instead of computing the mean delta-MFCC within each beat, let's dc
# We can replace the mean with any statistical aggregation function,

C_sync = librosa.util.sync(C, beats, aggregate=np.median)

plt.figure(figsize=(12,6))

plt.subplot(2, 1, 1)
librosa.display.specshow(C, sr=sr, y_axis='chroma', vmin=0.0, vmax=1.

plt.title('Chroma')
plt.colorbar()

plt.subplot(2, 1, 2)
librosa.display.specshow(C_sync, y_axis='chroma', vmin=0.0, vmax=1.0,
                         x_coords=librosa.frames_to_time(librosa.util


plt.title('Beat-synchronous Chroma (median aggregation)')

plt.colorbar()
plt.tight_layout()
```




File failed to load: file:///home/wb/2019/b/book/my_book/librosa/b_librosa/demo_officialP_Jupyter%20Notebook%20Viewer_files/extensions/MathMenu

File failed to load: file:///home/wb/2019/b/book/my_book/librosa/b_librosa/demo_officialP_Jupyter%20Notebook%20Viewer_files/extensions/MathMenu