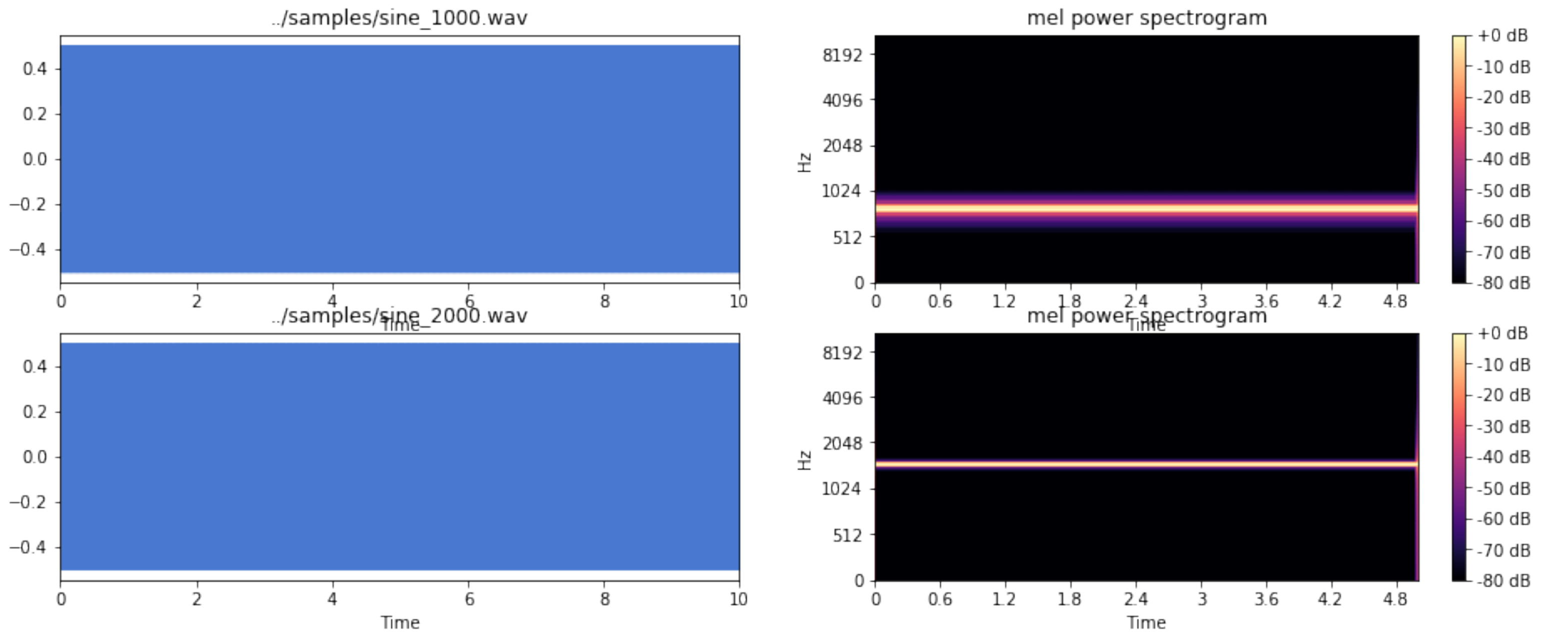
In [377]:

```
# importing libraries
import numpy as np
import numpy.matlib
import matplotlib.pyplot as plt
import matplotlib.style as ms
ms.use('seaborn-muted')
%matplotlib inline
import IPython.display
import librosa
import librosa.display
```

In [378]:

```
track1 = '../samples/sine_1000.wav'
track2 = '../samples/sine_2000.wav'
x1, sr = librosa.load(track1, sr = 44100, mono = 'True')
x2, sr = librosa.load(track2, sr = 44100, mono = 'True')

plt.figure(figsize=(16,6))
plt.subplot(2,2,1)
librosa.display.waveplot(x1)
plt.title(track1)

S1 = librosa.feature.melspectrogram(x1, sr=sr, n_mels=128)
log_S1 = librosa.power_to_db(S1, ref=np.max)
plt.subplot(2,2,2)
librosa.display.specshow(log_S1, sr=sr, x_axis='time', y_axis='mel')
plt.title('mel power spectrogram')
plt.colorbar(format='%+02.0f dB')

plt.subplot(2,2,3)
librosa.display.waveplot(x2)
plt.title(track2)

S2 = librosa.feature.melspectrogram(x2, sr=sr, n_mels=128)
log_S2 = librosa.power_to_db(S2, ref=np.max)
plt.subplot(2,2,4)
librosa.display.specshow(log_S2, sr=sr, x_axis='time', y_axis='mel')
plt.title('mel power spectrogram')
plt.colorbar(format='%+02.0f dB')
```

```
Out[378]:

<matplotlib.colorbar.Colorbar at 0x15a8725d0>
```



../samples/sine_1000.wav

mel power spectrogram

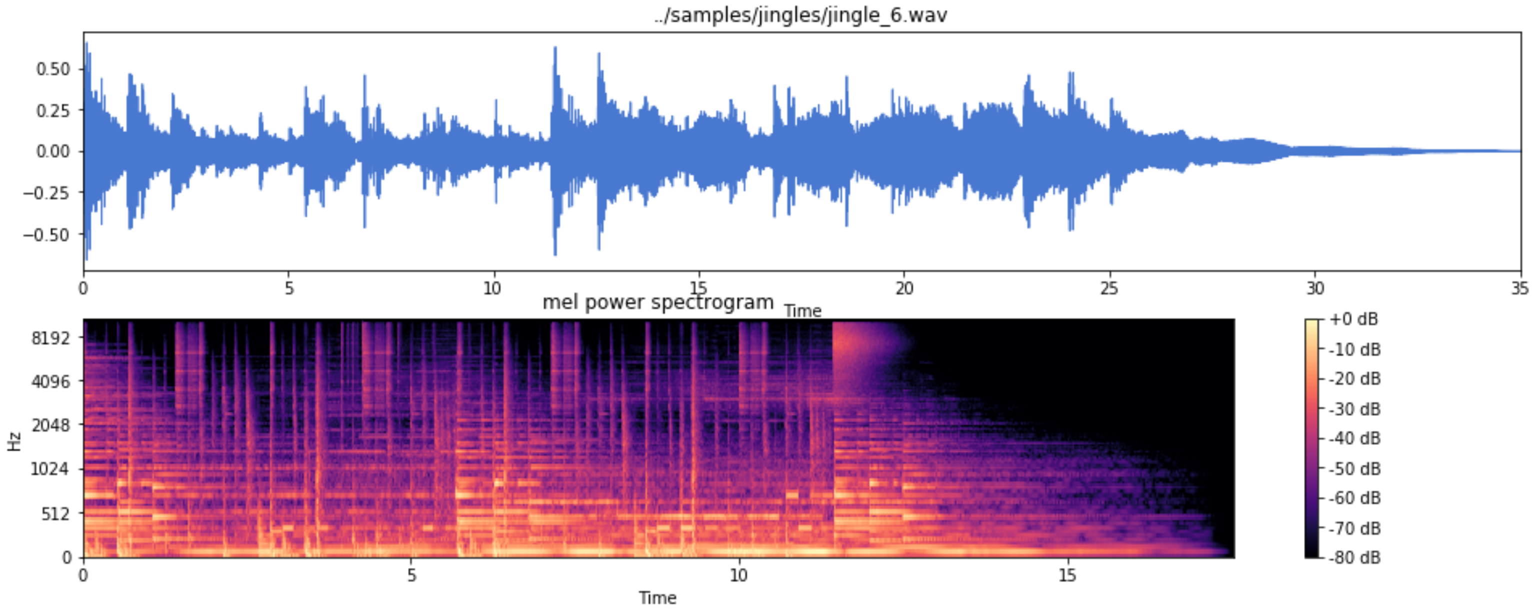
../samples/sine_2000.wav

mel power spectrogram

```
In [379]:
```

```
track = '../samples/jingles/jingle_6.wav'
x, sr = librosa.load(track, sr = 44100, mono = 'True')
plt.figure(figsize=(16,6))
plt.subplot(2,1,1)
librosa.display.waveplot(x)
plt.title(track)

S = librosa.feature.melspectrogram(x, sr=sr, n_mels=128)
log_S = librosa.power_to_db(S, ref=np.max)
plt.subplot(2,1,2)
librosa.display.specshow(log_S, sr=sr, x_axis='time', y_axis='mel')
plt.title('mel power spectrogram')
plt.colorbar(format='%+02.0f dB')
```

```
Out[379]:

<matplotlib.colorbar.Colorbar at 0x15d0d6290>
```



../samples/jingles/jingle_6.wav

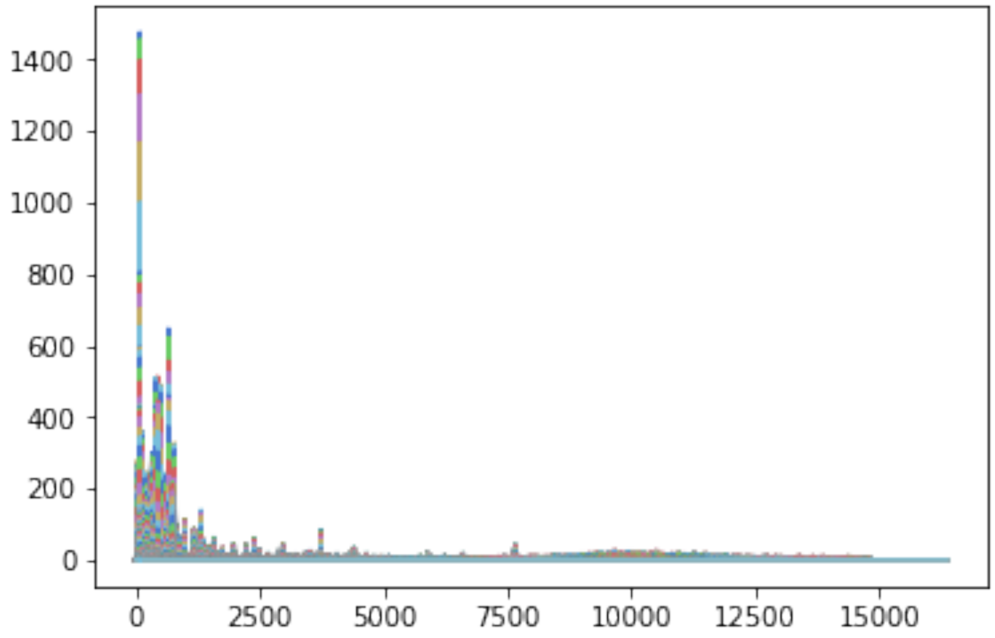
mel power spectrogram

In [380]:

```
# initialising STFT params
N = 2 ** 15 #FFT size
M = N     #window size
H = N/16  #hop size
W = 'hann'
```

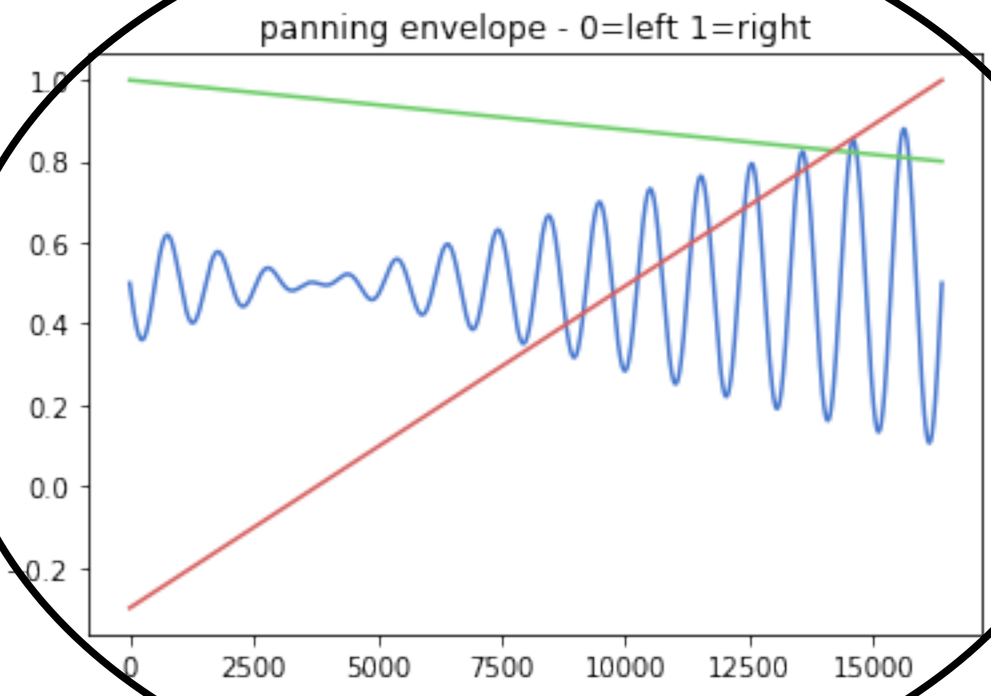
In [381]:

```
stft = librosa.stft(y = x, n_fft = N, hop_length = H, win_length = M, window = W
, center = 'False')
mX = np.copy(stft)
np.shape(mX)
plt.figure()
plt.plot(np.abs(mX))
plt.show()
```



In [395]:

```
n = N/2 + 1
pan_order = 32  #32 is best
alpha = np.pi/(n/pan_order)

start = 0
stop  = n

l1_start = 1.0 #fix
l1_stop = 0.8

l2_start = -0.3
l2_stop = 1.0  #fix

wave = np.zeros(n)
bins = np.arange(0,n)
x_bins = np.arange(0,stop-start)
wave[start:stop] = np.sin((x_bins)*alpha)
line1 = ((bins-stop)*(l1_stop-l1_start))/(stop-start)+l1_stop
line2 = ((bins-stop)*(l2_stop-l2_start))/(stop-start)+l2_stop
pan_filter = (wave*line1*line2 + 1.0)/2.0
plt.figure()
plt.plot(bins,pan_filter)
plt.plot(line1)
plt.plot(line2)
plt.title("panning envelope - 0=left 1=right")
plt.show()
print alpha
```



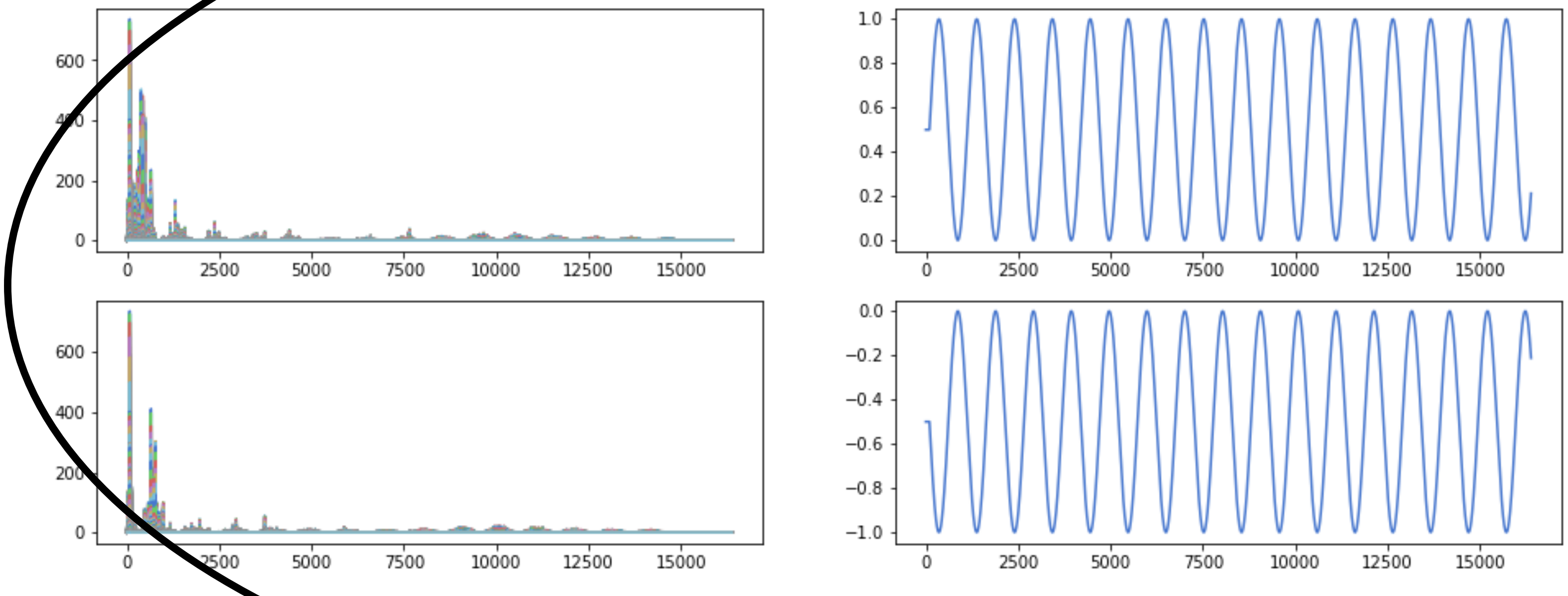
```
0.00613592315154
```

while creating this envelope I have control over:
1. Frequency bin where the envelope should begin
2. Frequency bin where envelope ends
3. order or frequency of the sine wave
4. the 2 lines are to attenuate the panning amount across frequency bins. In this case I wouldnt pan bass frequencies too much, but would be harsh at higher frequency bins

In [383]:

```
dc=2500
arr1=[]
arr2=[]
ons = np.zeros(dc)
ofs = np.ones(dc)
arr1 = np.append(ons,ofs)
arr2 = np.append(ofs,ons)
while ((np.size(arr1)<np.size(stft,axis=0)) and (np.size(arr2)<np.size(stft,axis=0))):
    arr1 = np.append(arr1,arr1)
    arr2 = np.append(arr2,arr2)
sq1 = arr1[0:np.size(stft,axis=0)]
sq2 = arr2[0:np.size(stft,axis=0)]

mY1 = np.copy(mX) #np.zeros(np.shape(mX))
mY1 = np.transpose(np.transpose(mY1) * np.matlib.repmat(pan_filter, np.size(stft,axis=1), 1))

plt.figure(figsize=(16,6))
plt.subplot(2,2,1)
plt.plot(np.abs(mY1))
plt.subplot(2,2,2)
plt.plot(pan_filter)

mY2 = np.copy(mX) #np.zeros(np.shape(mX))
mY2 = np.transpose(np.transpose(mY2) * np.matlib.repmat((-1.0*pan_filter)+1.0 , np.size(stft,axis=1), 1))
plt.subplot(2,2,3)
plt.plot(np.abs(mY2))
plt.subplot(2,2,4)
plt.plot(-1.0*pan_filter)
plt.show()
```

this was a crude choice of envelope(different from the one above) to test if algorithm was wokring, in reality, I would use envelope that would be more subtle to avoid hard panning

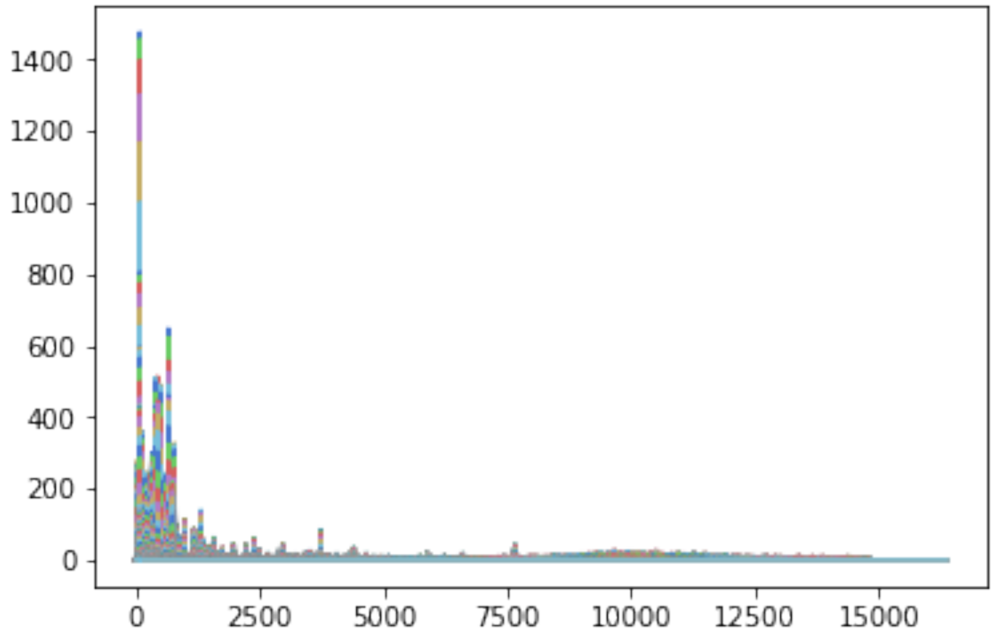


left channel

right channel

In [384]:

```
plt.figure()
plt.plot(np.abs(mX))
plt.show()
```



In [385]:

```
Y1 = librosa.istft(mY1, hop_length = H, win_length = M, window = W, center = 'Fa
lse')
Y2 = librosa.istft(mY2, hop_length = H, win_length = M, window = W, center = 'Fa
lse')
```

Left channel

In [386]:

```
import IPython.display as ipd
ipd.Audio(Y1, rate=sr)
```

Out[386]:

Right channel

In [387]:

```
ipd.Audio(Y2, rate=sr)
```

Out[387]:

Synthesized stereo

In [388]:

```
ipd.Audio([Y1,Y2], rate=sr)
```

Out[388]:

Original

In [389]:

```
ipd.Audio(x, rate=sr)
```

Out[389]:

In [390]:

```
freq = librosa.fft_frequencies(sr=sr, n_fft=N)
print freq, np.shape(freq)

mels = librosa.mel_frequencies(n_mels=N/2, fmin=20.0, fmax=20000.0, htk=False)
print mels, np.shape(mels)
```

```
[0.00000000e+00 1.34582520e+00 2.69165039e+00 ... 2.20473083e+04
 2.20486542e+04 2.20500000e+04] (16385,)
[   20.            20.23712857    20.47425715 ... 19990.22058519
 19995.10969472 20000.        ] (16384,)
```