SEPARAZIONE DELLA PARTE VOCALE DALLA MUSICA DI SOTTOFONDO

Damiano Valenza mat.3782594

OBIETTIVI:

PRIMA PARTE - ESEMPIO:

```
-Costruire due segnali unidimensionali con la stessa frequenza

-Cercare di separare i due segnali dopo averli uniti
```

SECONDA PARTE - PROGETTO VERO E PROPRIO:

```
-Ripetere la stessa operazione con un file audio importato in locale, cercare di separare ed
isolare la parte vocale
 dalla  traccia strumentale

-Riportare le operazioni riuscite e non riuscite

-Si prova l'operazione con un file audio differente
```

-CONCLUSIONI

Imports

In [1]:

```
from __future__ import print_function
import numpy as np
import matplotlib.pyplot as plt
import librosa
import IPython
import librosa.display
import soundfile as sf
import warnings
warnings.filterwarnings("ignore")
```

Prima parte - Esempio con due segnali monodimensionali

Come esempio ho deciso di creare dei semplici segnali 1-D: uno sinusoidale e uno cosinusoidale. Attraverso la funzione "multiply" della libreria di Numpy li ho uniti in un unico segnale, successivamente ho utilizzato la funzione "divide" per ottenere i due segnali di partenza a partire dall'unione.

CREAZIONE DI ENTRAMBI I SEGNALI:

Creo due segnali con la stessa frequenza e cerco di unirli in un unico segnale

In [2]:

```
f_s = 50.0 #Frequenza di campionamento
T=1.0; #secondi
time = np.arange(0.0, T , 1/f_s) # stiamo campionando alla frequenza f_s
N=time.size
y = 5*np.cos(2*np.pi*T*time)
print(N)
```

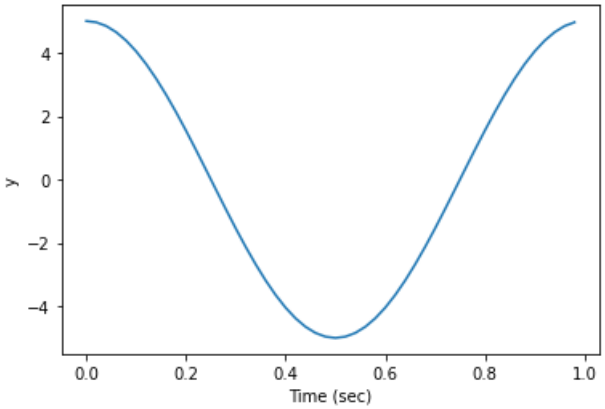
```
50
```

In [3]:

```
plt.plot(time, y)
plt.xlabel("Time (sec)")
plt.ylabel("y")
```

Out[3]:

```
Text(0, 0.5, 'y')
```



In [4]:

```
f_s = 50.0 # Hz
T=1.0 #secondi
time = np.arange(0.0, T, 1/f_s) # stiamo campionando alla frequenza f_s
N=time.size;
y1 = 5*np.sin(2*np.pi*T*time)
print(N)
```

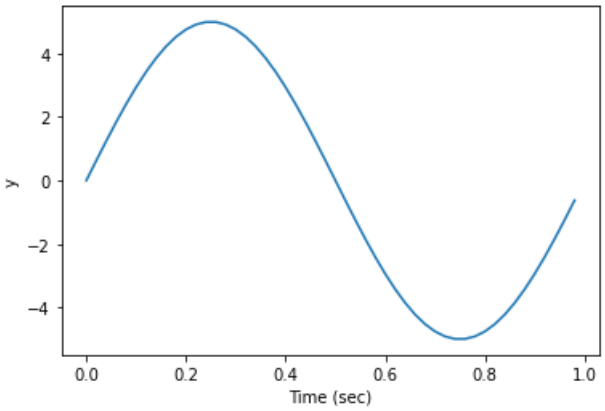
```
50
```

In [5]:

```
plt.plot(time, y1)
plt.xlabel("Time (sec)")
plt.ylabel("y")
```

Out[5]:

```
Text(0, 0.5, 'y')
```



UNIONE DEI DUE SEGNALI:

In [6]:

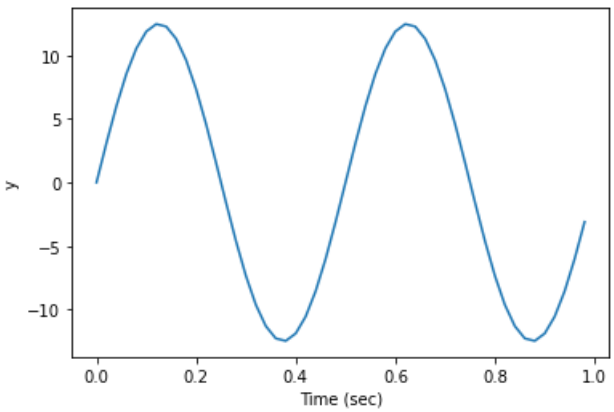
```
unione=np.multiply(y, y1)
```

In [7]:

```
plt.plot(time, unione)
plt.xlabel("Time (sec)")
plt.ylabel("y")
```

Out[7]:

```
Text(0, 0.5, 'y')
```



RIOTTENIMENTO DEI DUE SEGNALI DOPO L'UNIONE:

In [8]:

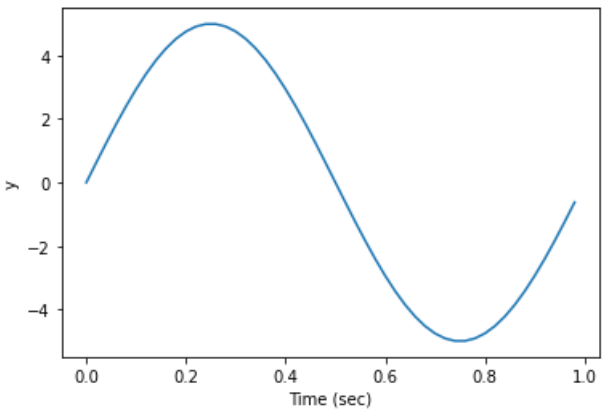
```
y1n=np.divide(unione,y)
```

In [9]:

```
plt.plot(time, y1n)
plt.xlabel("Time (sec)")
plt.ylabel("y")
```

Out[9]:

```
Text(0, 0.5, 'y')
```



In [10]:

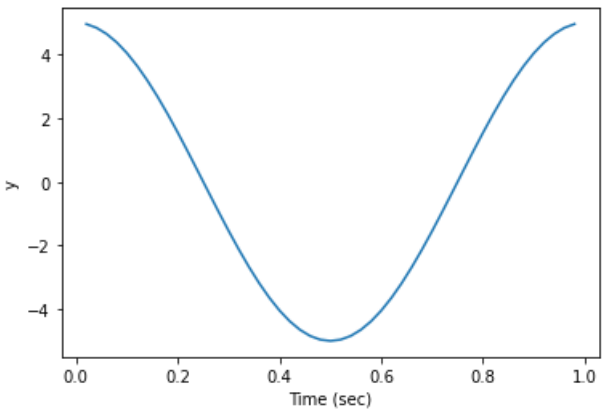
```
yn=np.divide(unione,y1)
```

In [11]:

```
plt.plot(time, yn)
plt.xlabel("Time (sec)")
plt.ylabel("y")
```

Out[11]:

```
Text(0, 0.5, 'y')
```



Seconda Parte - Separazione della traccia vocale e musicale in un file audio

Per questa parte ho deciso di utilizzare un brano abbastanza corto con una parte vocale molto accentuata onde evitare di avere frequenze simili

Caricamento File

In [12]:

```
y, sr = librosa.load('Good.wav', duration=120)

S_full, phase = librosa.magphase(librosa.stft(y)) #ampiezza e fase dello spettrogramma
```

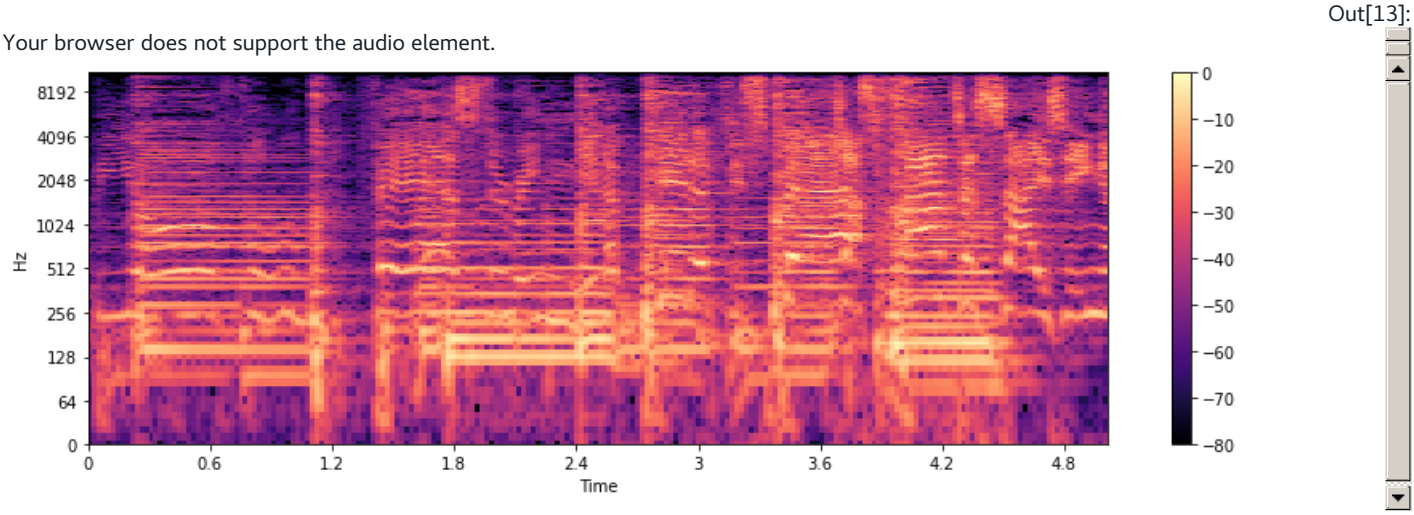
Plot di 5 secondi dello spettrogramma e riproduzione del brano

In [13]:

```
idx = slice(*librosa.time_to_frames([30, 35], sr=sr))
plt.figure(figsize=(12, 4))
librosa.display.specshow(librosa.amplitude_to_db(S_full[:, idx], ref=np.max),
                         y_axis='log', x_axis='time', sr=sr)
plt.colorbar()
plt.tight_layout()
IPython.display.Audio(y, rate=22000)
```

Out[13]:

Your browser does not support the audio element.



Le linee ondulate sono dovute alla componente vocale. L'obiettivo è separarla dalla musica di sottofondo.

All'interno di questa funzione della libreria Librosa le porzioni della STFT simili vengono aggregate e separate da due secondi, attraverso il settaggio del parametro "metric" si utilizza la tecnica della "similarità del coseno" per confrontarle.

Questa tecnica permette di separare le porzioni non-simili dallo spettro, in questo caso le parti vocali.

In [14]:

```
S_filter = librosa.decompose.nn_filter(S_full,
                                       aggregate=np.median, #si aggregano porzioni simili prendendo il lo
                                       metric='cosine',
                                       width=int(librosa.time_to_frames(2, sr=sr)))


# L'output del filtro non dovrebbe essere più grande dell'input

S_filter = np.minimum(S_full, S_filter)
```

L'output del filtro può essere utilizzato come maschera sonora.

Si utilizza un margine per ridurre la perdita tra la maschera vocale e la maschera strumentale, per una corretta separazione della voce e del sottofondo i margini devono essere diversi l'uno dall'altro.

La funzione della libreria Librosa "util.softmask" prende in input due matrici pesate non negative(in questo caso voce e background) e restituisce una maschera attraverso il filtraggio di Wiener.

Una volta ottenute, le maschere vengono moltiplicate con lo spettro in input per separare le componenti.

In [15]:

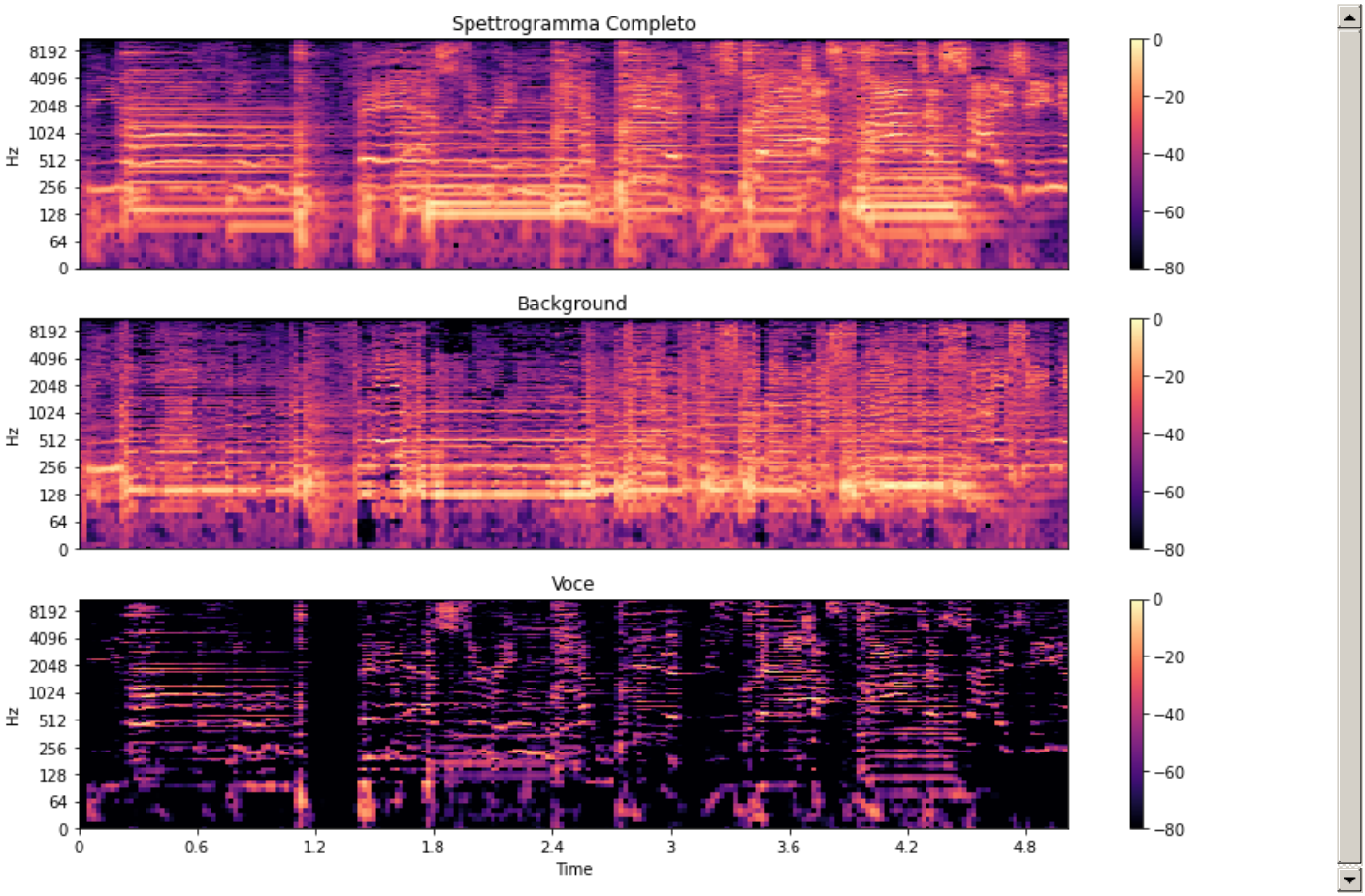
```
margin_i, margin_v = 2, 10
```

```
power = 2

mask_i = librosa.util.softmask(S_filter,              #maschera per il sottofondo che elimina le parti voc
                               margin_i * (S_full - S_filter),
                               power=power)

mask_v = librosa.util.softmask(S_full - S_filter,     #maschera per le parti vocali che elimina il sottofc
                               margin_v * S_filter,
                               power=power)


S_voce = mask_v * S_full
S_background = mask_i * S_full
```

Si plotta la stessa sezione, ma separata in voce e background

In [16]:

```
plt.figure(figsize=(12, 8))
plt.subplot(3, 1, 1)
librosa.display.specshow(librosa.amplitude_to_db(S_full[:, idx], ref=np.max),
                         y_axis='log', sr=sr)
plt.title('Spettrogramma Completo')
plt.colorbar()
plt.subplot(3, 1, 2)
librosa.display.specshow(librosa.amplitude_to_db(S_background[:, idx], ref=np.max),
                         y_axis='log', sr=sr)
plt.title('Background')
plt.colorbar()
plt.subplot(3, 1, 3)
librosa.display.specshow(librosa.amplitude_to_db(S_voce[:, idx], ref=np.max),
                         y_axis='log', x_axis='time', sr=sr)
plt.title('Voce')
plt.colorbar()
plt.tight_layout()
plt.show()
```



Isolamento della voce:

In [23]:

```
new_y = librosa.istft(S_voce*phase)
sf.write("./voce.wav", new_y, sr)
```

In [17]:

```
IPython.display.Audio('voce.wav')
```

Out[17]:

Your browser does not support the audio element.

Da quello che si può sentire l'isolamento della voce è riuscito, ma non del tutto! E' ancora possibile udire(a volume molto basso) la musica di sottofondo.

Isolamento della musica di sottofondo:

In [25]:

```
x = librosa.istft(S_background*phase)
sf.write("./background.wav", x, sr)
```

In [18]:

```
IPython.display.Audio('background.wav')
```

Out[18]:

Your browser does not support the audio element.

L'isolamento della musica di sottofondo è riuscito, ma i suoni non sono ben definiti ed è possibile udire(anche in questo caso a volume molto basso) il suono della parte vocale.

Questa separazione non completa delle due tracce è probabilmente dovuta alla presenza di frequenze simili all'interno del brano che non possono essere elaborate correttamente e quindi difficilmente separabili.

Prova con un file audio differente

Adesso provo la stessa operazione con un altro brano,sempre breve, ma molto diverso dal primo: qui la parte strumentale consta di chitarra elettrica, batteria e basso e la voce è molto meno accentuata.

Caricamento File

In [19]:

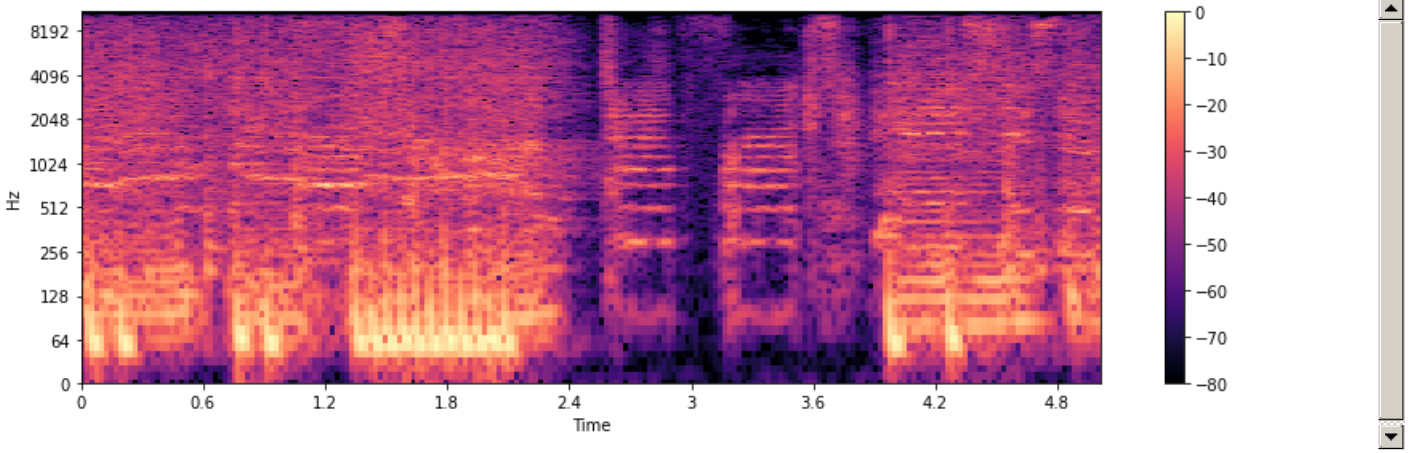
```
y2, sr = librosa.load('Mean_Mr_Mustard.wav', duration=120)

S_full2, phase = librosa.magphase(librosa.stft(y2))
```

In [22]:

```
idx2 = slice(*librosa.time_to_frames([30, 35], sr=sr))
plt.figure(figsize=(12, 4))
librosa.display.specshow(librosa.amplitude_to_db(S_full2[:, idx2], ref=np.max),
                         y_axis='log', x_axis='time', sr=sr)
plt.colorbar()
plt.tight_layout()
IPython.display.Audio(y2, rate=22000)
```

Out[22]:

Your browser does not support the audio element.



In [23]:

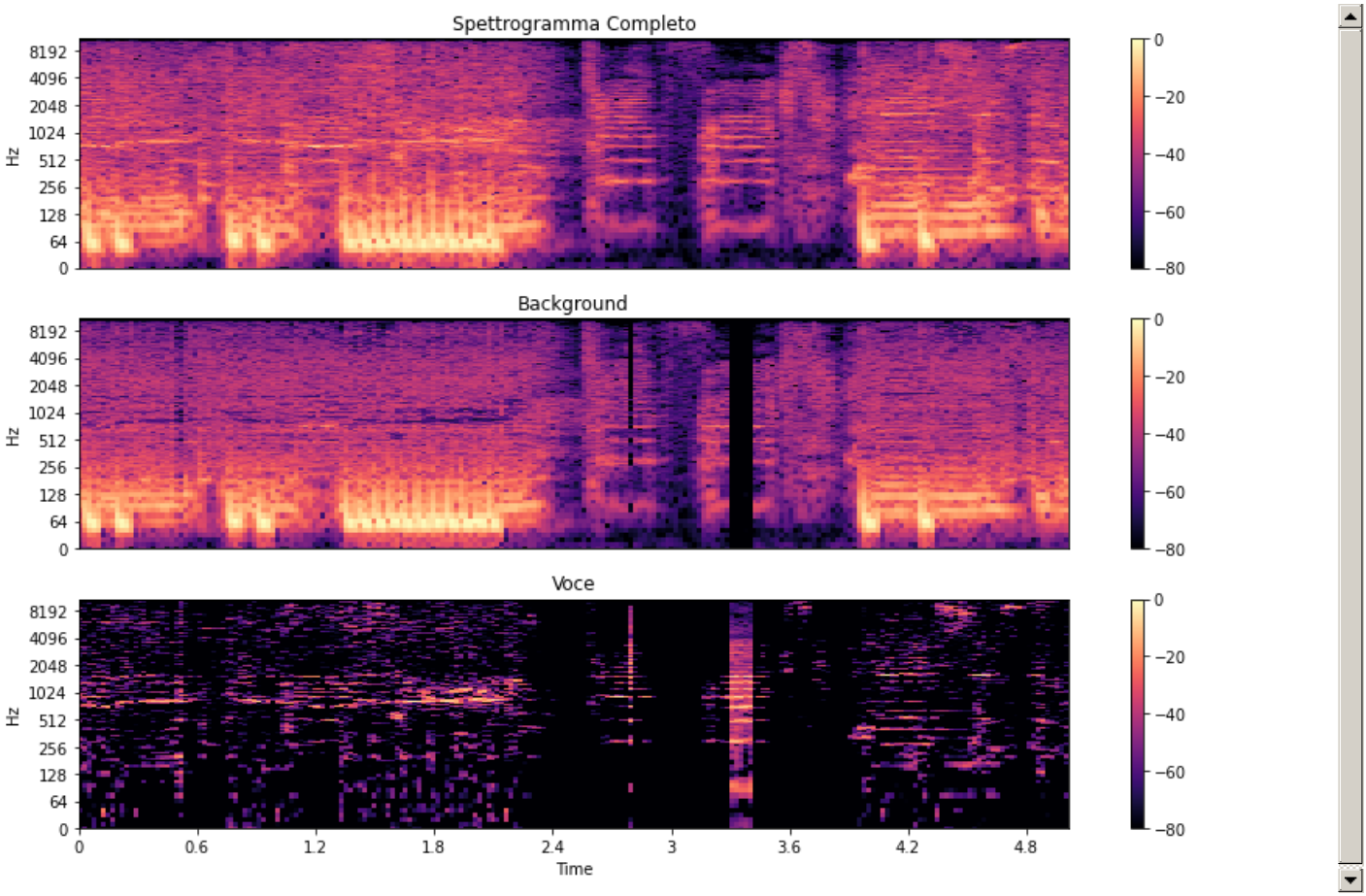
```
S_filter2 = librosa.decompose.nn_filter(S_full2,
                                        aggregate=np.median,
                                        metric='cosine',
                                        width=int(librosa.time_to_frames(2, sr=sr)))

S_filter2 = np.minimum(S_full2, S_filter2)
```

In [24]:

```
m_i, m_v = 2, 10
power = 2

maschera_i = librosa.util.softmask(S_filter2,
                                   m_i * (S_full2 - S_filter2),
                                   power=power)

maschera_v = librosa.util.softmask(S_full2 - S_filter2,
                                   m_v * S_filter2,
                                   power=power)


S_voice = maschera_v * S_full2
S_sottofondo = maschera_i * S_full2
```

In [25]:

```
plt.figure(figsize=(12, 8))
plt.subplot(3, 1, 1)
librosa.display.specshow(librosa.amplitude_to_db(S_full2[:, idx2], ref=np.max),
                         y_axis='log', sr=sr)
plt.title('Spettrogramma Completo')
plt.colorbar()
plt.subplot(3, 1, 2)
librosa.display.specshow(librosa.amplitude_to_db(S_sottofondo[:, idx2], ref=np.max),
                         y_axis='log', sr=sr)
plt.title('Background')
plt.colorbar()
plt.subplot(3, 1, 3)
librosa.display.specshow(librosa.amplitude_to_db(S_voice[:, idx2], ref=np.max),
                         y_axis='log', x_axis='time', sr=sr)
plt.title('Voce')
plt.colorbar()
plt.tight_layout()
plt.show()
```



Isolamento della voce:

In [26]:

```
new_y2 = librosa.istft(S_voice*phase)
```

```
sf.write("./voce2.wav", new_y2, sr)
```

In [27]:

```
IPython.display.Audio('voce2.wav')
```

Out[27]:

Your browser does not support the audio element.

Con mio grande stupore, il risultato è estremamente positivo. Ho pensato che la tonalità bassa della voce e l'alto riverbero potessero dare dei problemi e che quindi non fosse semplice da isolare, ma così non è stato: la musica di sottofondo è quasi totalmente assente.

Isolamento della musica di sottofondo:

In [28]:

```
x2 = librosa.istft(S_sottofondo*phase)
sf.write("./background2.wav", x2, sr)
```

In [29]:

```
IPython.display.Audio('background2.wav')
```

Out[29]:

Your browser does not support the audio element.

Anche l'isolamento della musica di sottofondo è ben riuscito. Qui la voce è presente nella parte centrale del brano nella quale il cantante utilizza un "vocoder" per ottenere una voce robotica, molto probabilmente questo effetto non viene riconosciuto come parte vocale ed è quindi assimilato al background(nell'isolamento della voce si sente appena).

Conclusioni

Il primo è un brano acustico composto da voce e chitarra, spesso la chitarra segue il ritmo e l'andamento della voce.

Il procedimento utilizzato quindi fa molta fatica a distinguere le due parti nonostante la differenza di tonalità tra voce e chitarra sia molto accentuata.

Dato che la separazione non è netta è comunque lecito pensare che ci possano essere frequenze simili tra loro nelle due parti.

Il secondo brano è più completo e ricco di elementi rispetto al primo, la base musicale ha un ritmo diverso rispetto alla voce e da questa particolarità si capisce il perchè la separazione è riuscita in maniera più nitida.

Questo procedimento quindi tende a separare più facilmente la voce da brani con una forte differenza ritmica tra la parte vocale e la musica, se dovesse esistere un brano in cui la voce resta allineata alla base musicale per tutta la durata della traccia esso non verrebbe elaborato correttamente lasciando quindi invariate le due parti.

Attraverso la prima parte del progetto è possibile comprendere che è facile separare due segnali che sono stai uniti in precedenza.

Questo ricorda le sale di registrazione in cui ogni linea di uno strumento musicale viene registrata separatamente per poi assemblarla con le altre per creare il brano completo.

In questo caso è possibile separare le diverse linee attraverso il processo inverso all'assemblaggio.

Suppongo che non sia facile creare un metodo universale per separare correttamente tracce vocali e strumentali da qualsiasi brano poichè ogni brano è diverso dall'altro e non risulta facile avere la piena conoscenza di tutte le tracce musicali esistenti.