9. Manualna ekstrakcja cech z utworów muzycznych

9.1. Pobierz dane

Dane to 30-sekundowe próbki utworów z dziesięciu kategorii (gatunków). Oryginalny zbiór danych *gitzan* zawierają po 100 fragmentów utworów każdej kategorii. Dane do obróbki podczas zajęć zostały zredukowane - do 10 utworów/kategorię. Plik zajmuje około 120MB.

In [1]:
```
!wget https://dysk.agh.edu.pl/s/Ewqj459KS6eTBd2/download -O gitzan_small.
!unzip gitzan_small.zip
```

```
--2023-05-04 09:18:56--  https://dysk.agh.edu.pl/s/Ewqj459KS6eTBd2/downl
oad
Resolving dysk.agh.edu.pl (dysk.agh.edu.pl)... 149.156.96.4, 2001:6d8:1
0:1060::6004
Connecting to dysk.agh.edu.pl (dysk.agh.edu.pl)|149.156.96.4|:443... con
nected.
HTTP request sent, awaiting response... 200 OK
Length: 128150658 (122M) [application/zip]
Saving to: ‘gitzan_small.zip’

gitzan_small.zip    100%[===================>] 122.21M  21.6MB/s    in
6.7s

2023-05-04 09:19:04 (18.2 MB/s) - ‘gitzan_small.zip’ saved [128150658/12
8150658]

Archive:  gitzan_small.zip
   creating: genres_original/
   creating: genres_original/blues/
  inflating: genres_original/blues/blues.00000.wav
  inflating: genres_original/blues/blues.00001.wav
  inflating: genres_original/blues/blues.00002.wav
  inflating: genres_original/blues/blues.00003.wav
  inflating: genres_original/blues/blues.00004.wav
  inflating: genres_original/blues/blues.00005.wav
  inflating: genres_original/blues/blues.00006.wav
  inflating: genres_original/blues/blues.00007.wav
  inflating: genres_original/blues/blues.00008.wav
  inflating: genres_original/blues/blues.00009.wav
   creating: genres_original/classical/
  inflating: genres_original/classical/classical.00000.wav
  inflating: genres_original/classical/classical.00001.wav
  inflating: genres_original/classical/classical.00002.wav
  inflating: genres_original/classical/classical.00003.wav
  inflating: genres_original/classical/classical.00004.wav
  inflating: genres_original/classical/classical.00005.wav
  inflating: genres_original/classical/classical.00006.wav
  inflating: genres_original/classical/classical.00007.wav
  inflating: genres_original/classical/classical.00008.wav
  inflating: genres_original/classical/classical.00009.wav
   creating: genres_original/country/
  inflating: genres_original/country/country.00000.wav
  inflating: genres_original/country/country.00001.wav
  inflating: genres_original/country/country.00002.wav
  inflating: genres_original/country/country.00003.wav
  inflating: genres_original/country/country.00004.wav
  inflating: genres_original/country/country.00005.wav
  inflating: genres_original/country/country.00006.wav
  inflating: genres_original/country/country.00007.wav
  inflating: genres_original/country/country.00008.wav
  inflating: genres_original/country/country.00009.wav
   creating: genres_original/disco/
  inflating: genres_original/disco/disco.00000.wav
  inflating: genres_original/disco/disco.00001.wav
  inflating: genres_original/disco/disco.00002.wav
  inflating: genres_original/disco/disco.00003.wav
  inflating: genres_original/disco/disco.00004.wav
  inflating: genres_original/disco/disco.00005.wav
  inflating: genres_original/disco/disco.00006.wav
  inflating: genres_original/disco/disco.00007.wav
```

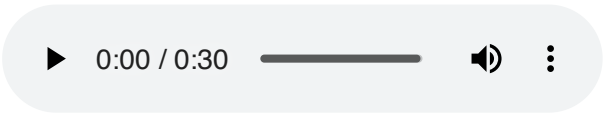
```
inflating: genres_original/disco/disco.00008.wav
inflating: genres_original/disco/disco.00009.wav
 creating: genres_original/hiphop/
inflating: genres_original/hiphop/hiphop.00000.wav
inflating: genres_original/hiphop/hiphop.00001.wav
inflating: genres_original/hiphop/hiphop.00002.wav
inflating: genres_original/hiphop/hiphop.00003.wav
inflating: genres_original/hiphop/hiphop.00004.wav
inflating: genres_original/hiphop/hiphop.00005.wav
inflating: genres_original/hiphop/hiphop.00006.wav
inflating: genres_original/hiphop/hiphop.00007.wav
inflating: genres_original/hiphop/hiphop.00008.wav
inflating: genres_original/hiphop/hiphop.00009.wav
 creating: genres_original/jazz/
inflating: genres_original/jazz/jazz.00000.wav
inflating: genres_original/jazz/jazz.00001.wav
inflating: genres_original/jazz/jazz.00002.wav
inflating: genres_original/jazz/jazz.00003.wav
inflating: genres_original/jazz/jazz.00004.wav
inflating: genres_original/jazz/jazz.00005.wav
inflating: genres_original/jazz/jazz.00006.wav
inflating: genres_original/jazz/jazz.00007.wav
inflating: genres_original/jazz/jazz.00008.wav
inflating: genres_original/jazz/jazz.00009.wav
 creating: genres_original/metal/
inflating: genres_original/metal/metal.00000.wav
inflating: genres_original/metal/metal.00001.wav
inflating: genres_original/metal/metal.00002.wav
inflating: genres_original/metal/metal.00003.wav
inflating: genres_original/metal/metal.00004.wav
inflating: genres_original/metal/metal.00005.wav
inflating: genres_original/metal/metal.00006.wav
inflating: genres_original/metal/metal.00007.wav
inflating: genres_original/metal/metal.00008.wav
inflating: genres_original/metal/metal.00009.wav
 creating: genres_original/pop/
inflating: genres_original/pop/pop.00000.wav
inflating: genres_original/pop/pop.00001.wav
inflating: genres_original/pop/pop.00002.wav
inflating: genres_original/pop/pop.00003.wav
inflating: genres_original/pop/pop.00004.wav
inflating: genres_original/pop/pop.00005.wav
inflating: genres_original/pop/pop.00006.wav
inflating: genres_original/pop/pop.00007.wav
inflating: genres_original/pop/pop.00008.wav
inflating: genres_original/pop/pop.00009.wav
 creating: genres_original/reggae/
inflating: genres_original/reggae/reggae.00000.wav
inflating: genres_original/reggae/reggae.00001.wav
inflating: genres_original/reggae/reggae.00002.wav
inflating: genres_original/reggae/reggae.00003.wav
inflating: genres_original/reggae/reggae.00004.wav
inflating: genres_original/reggae/reggae.00005.wav
inflating: genres_original/reggae/reggae.00006.wav
inflating: genres_original/reggae/reggae.00007.wav
inflating: genres_original/reggae/reggae.00008.wav
inflating: genres_original/reggae/reggae.00009.wav
 creating: genres_original/rock/
inflating: genres_original/rock/rock.00000.wav
inflating: genres_original/rock/rock.00001.wav
```

```
  inflating: genres_original/rock/rock.00002.wav
  inflating: genres_original/rock/rock.00003.wav
  inflating: genres_original/rock/rock.00004.wav
  inflating: genres_original/rock/rock.00005.wav
  inflating: genres_original/rock/rock.00006.wav
  inflating: genres_original/rock/rock.00007.wav
  inflating: genres_original/rock/rock.00008.wav
  inflating: genres_original/rock/rock.00009.wav
  inflating: features_3_sec.csv
  inflating: features_30_sec.csv
```

Możesz odtworzyć wybrany utwór...

In [2]:
```
audio_path = 'genres_original/blues/blues.00008.wav'

import IPython.display as ipd
ipd.Audio(audio_path)
```

Out[2]:



Będziemy używali biblioteki librosa do przetwarzania danych audio.

Zmienna sr to *sampling rate* (czestość próbkowania). Standardem jest 44100, ale librosa przyjmuje defaultową wartość 22050, co odpowiada pasmu przenoszenia do 10kHz.

In [63]:
```
import numpy as np
import librosa

audio_path = 'genres_original/blues/blues.00000.wav'
x , sr = librosa.load(audio_path)
print(x.shape,sr)
print(x.shape[0]/sr)
x , sr = librosa.load(audio_path, sr=44100)
print(x.shape,sr)
print(f'Długość nagrania: {x.shape[0]/sr} sec')
```

```
(661794,) 22050
30.013333333333332
(1323588,) 44100
Długość nagrania: 30.013333333333332 sec
```

9.2. Załaduj wszystkie pliki audio

Wszystkie pliki zostaną umieszczone w Pandas DataFrame. Nazwy folderów (i plików) odpowiadają gatunkom muzycznym.

TODO 9.2.1

1. Utwórz DataFrame przekazując w konstruktorze słownik, którego kluczami sa nazwy kolumn, a wartościami listy. Na przykład postaci

`{'nazwa1':lista1,'nazwa2':lista2}` 2. Nazwij kolumny: 'genre', 'file', 'audio', 'sr'

In [64]:
```
#load all
import numpy as np
import pandas as pd
import librosa

import os

root_folder = 'genres_original'
genres = []
audio_data = []
files=[]
sampling_rates=[]
for genre in os.listdir(root_folder):
    genre_folder=root_folder+'/'+genre
    for audio_file_path in os.listdir(genre_folder):
      fp=genre_folder+'/'+audio_file_path
      x,sr = librosa.load(fp)
      genres.append(genre)
      files.append(audio_file_path)
      audio_data.append(x)
      sampling_rates.append(sr)

import pandas as pd

df = pd.DataFrame({'genre': genres, 'file': files, 'audio': audio_data, '
df.head(10)
```

Out[64]:

	genre	file	audio	sr
0	reggae	reggae.00009.wav	[-0.054840088, -0.088256836, -0.08078003, -0.0...	22050
1	reggae	reggae.00007.wav	[-0.0132751465, -0.02557373, -0.022277832, -0....	22050
2	reggae	reggae.00008.wav	[0.036315918, 0.05316162, 0.048919678, 0.04876...	22050
3	reggae	reggae.00005.wav	[0.23120117, 0.25952148, 0.076690674, 0.069793...	22050
4	reggae	reggae.00002.wav	[-0.05203247, -0.088256836, -0.09857178, -0.12...	22050
5	reggae	reggae.00003.wav	[-0.019836426, -0.0574646, -0.09024048, -0.141...	22050
6	reggae	reggae.00006.wav	[-0.01864624, -0.03515625, -0.0078125, 0.02917...	22050
7	reggae	reggae.00001.wav	[-0.05355835, -0.093811035, -0.09585571, -0.11...	22050
8	reggae	reggae.00000.wav	[0.010375977, 0.007751465, 0.038879395, 0.0934...	22050
9	reggae	reggae.00004.wav	[0.040100098, 0.055999756, 0.03945923, 0.02740...	22050

In [65]:
```
# Sprawdź rozmiary df. Powinno być (100,4)
df.shape
```

Out[65]: (100, 4)

9.2.1 Dostęp do danych

Wybierz jeden z utworów i sprawdź, jak uzyskać dostęp do jego danych.

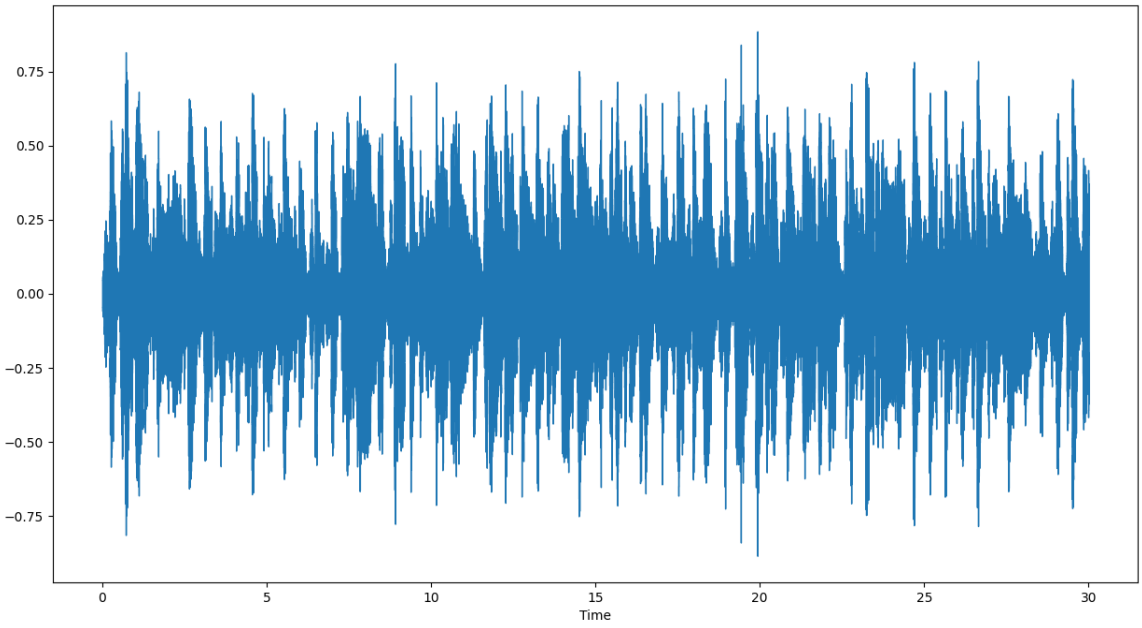
In [66]:
```
# czym jest df.audio[df.file=='blues.00000.wav']
s = df.audio[df.file=='blues.00000.wav']
```

```
print(type(s))
print(s.shape)
print(s)
x = s.iloc[0]
print(x)
sr = df.sr[df.file=='blues.00000.wav'].iloc[0]
print(sr)
```

```
<class 'pandas.core.series.Series'>
(1,)
28    [0.0073242188, 0.016601562, 0.0076293945, -0.0...
Name: audio, dtype: object
[ 0.00732422  0.01660156  0.00762939 ... -0.05560303 -0.06106567
 -0.06417847]
22050
```

Wizualizacja przebiegu

In [67]:
```
import matplotlib.pyplot as plt
import librosa.display
plt.rcParams["figure.figsize"] = (15,8)
librosa.display.waveshow(x, sr=sr);
```



9.2.2 Transformacja FFT

Większość cech utworów muzycznych jest ekstrahowana na podstawie analizy spektrum dźwięku i jego zmian. Do konwersji reprezentacji sygnału w dziedzinie czasu do dziedziny czestotliwości wykorzystywana jest szybka transformacja Fouriera.

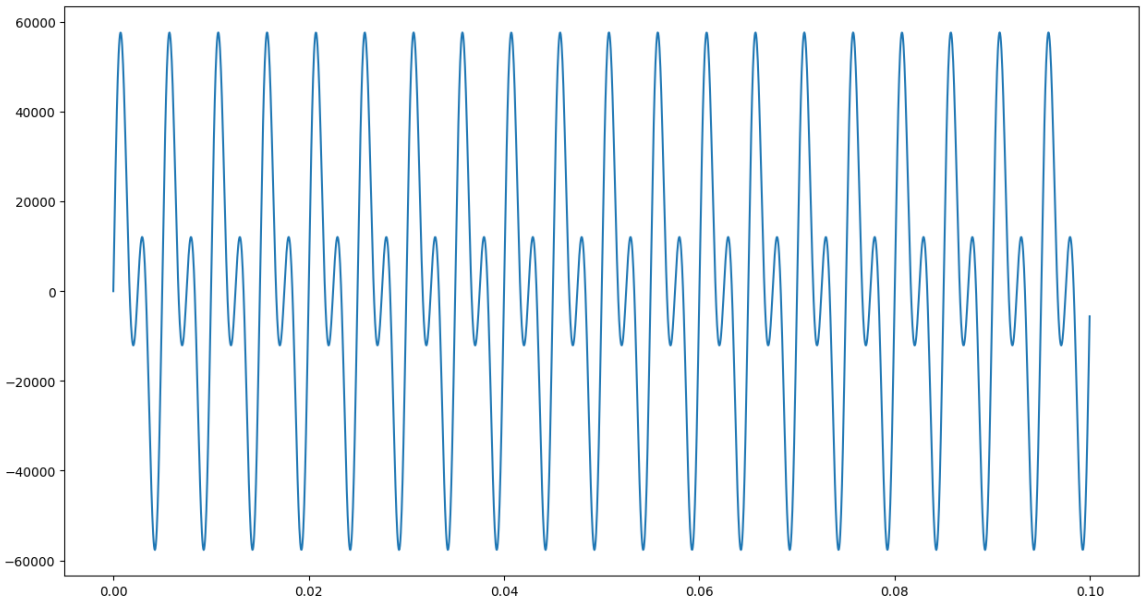
Utworzymy przebieg będący superpozycją dwóch sinusoid.

TODO 9.2.2

- Popraw poniższy kod, aby narysowany został fragment przebiegu pokazujacy sinusoidę....

In [68]:
```
# około 30 sekund przy czasie próbkowania sr
# t=np.linspace(0,30,x.shape[0],endpoint=False)
t=np.linspace(0,30,30*22050,endpoint=False)
print(t.shape)
# sygnał 200 i 400 Hz
signal=np.sin(2*np.pi*200*t) + np.sin(2*np.pi*400*t)
signal=signal*(2**15)
plt.plot(t[0:2205], signal[0:2205])
plt.show()
```

```
(661500,)
```



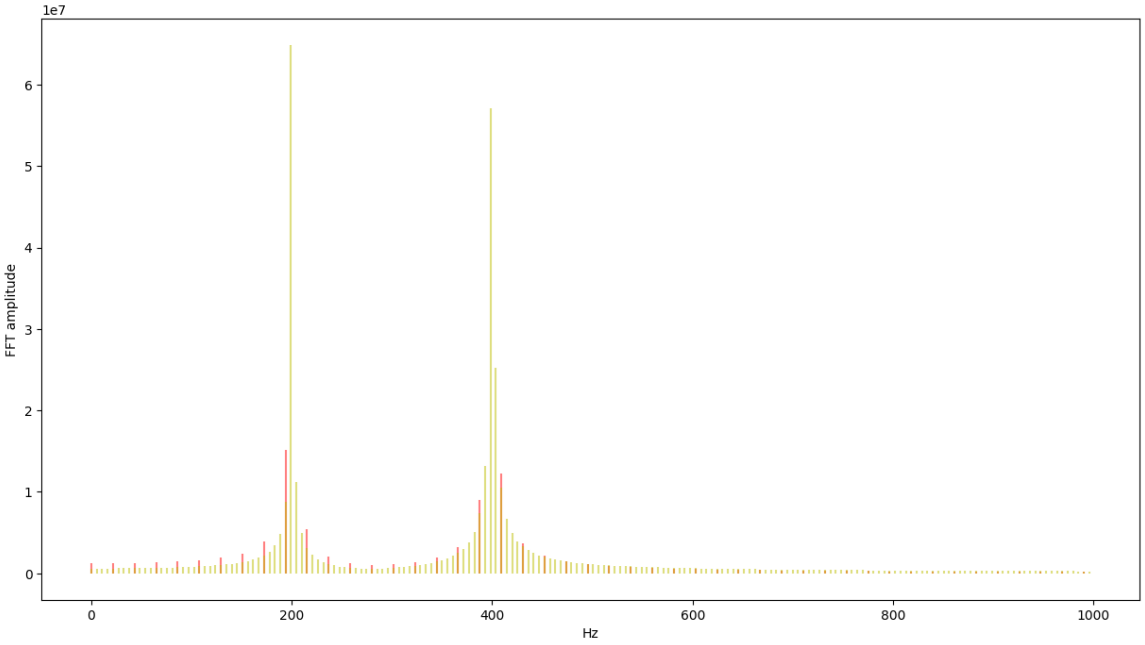
Transformacja Fouriera zamienia sygnał w dziedzinie czasu na reprezentację w dziedzinie czestotliwości.

Narysujmy wynik transformacji.

TODO 9.2.3

- Co się stanie jeżeli zmniejszymy window_length? Np. do 1 sekundy (22050), trochę mniej niż sekundy (22000), około 93ms (2048)

In [69]:
```
from scipy.fftpack import fft, fftfreq
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (15,8)

def plot_fft(signal,start,window_length,max_freq=11025,sr=22050,color='b'
  y=signal[start:start+window_length]
  yf = fft(y)
  xf = fftfreq(window_length, 1 / sr)[:window_length//2]
  limit=xf[xf<=max_freq].shape[0]
  plt.vlines(xf[0:limit], np.zeros(limit),scale*np.abs(yf[0:limit]),color
  plt.grid()
  plt.xlabel('Hz')
  plt.ylabel('FFT amplitude')

# plot_fft(signal,start=0,window_length=signal.shape[0],max_freq=1000)
plot_fft(signal, start=0, window_length=1024, max_freq=1000, color='r')
plot_fft(signal, start=0, window_length=4096, max_freq=1000, color='y')
plt.show()
```

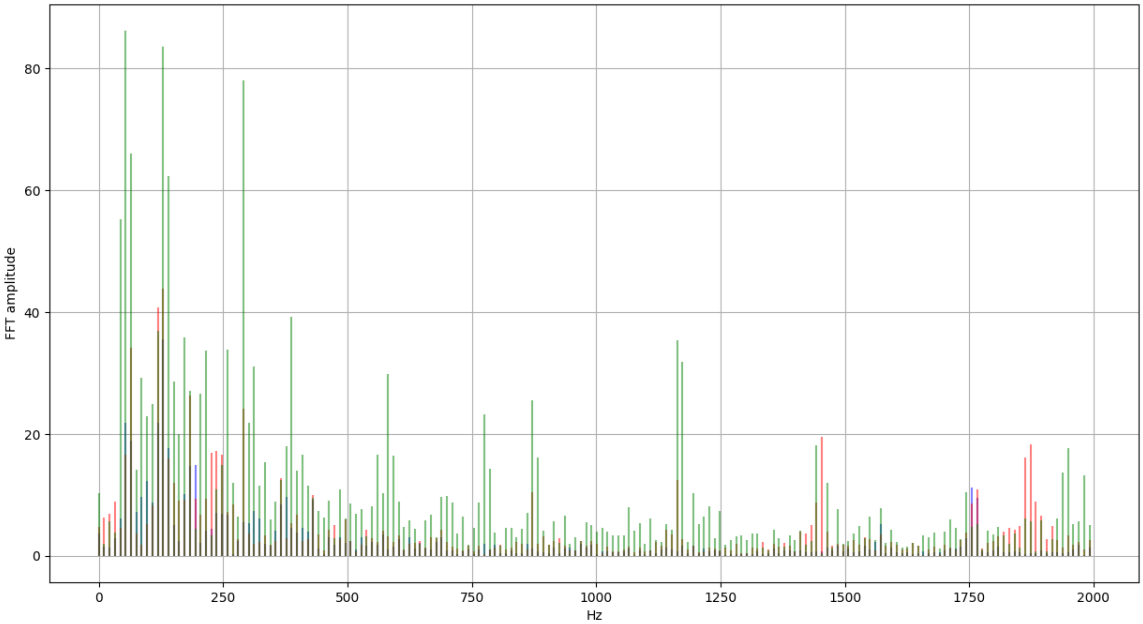


TODO 9.2.4

- Wydrukuj spektrum sygnału dla 3 punktów w czasie (różniacych się o kilkadziesiąt milisekund), używając róznych kolorów ('r', 'g', 'b'). Ile to będzie kilkadziesiąt milisekund?

Dla różnych momentów czasu rozkład czestotliwości powinien się zmieniać.

In [70]:
```
# x = df.audio[df.file=='blues.00000.wav']
# x = df.audio[df.file=='blues.00000.wav'].iloc[0]

# t=np.linspace(0,30,x.shape[0])
# noise=np.sin(2*np.pi*t*200)
# x=x+noise
# noise=np.sin(2*np.pi*t*400)
# x=x+noise

# plot_fft(x,4*22050,22050,1000,'b')
# # plot_fft(x,4*22050+2205,800,1000,'y',scale=1)
# plt.plot(t[0:500],noise[0:500])
# plt.show()

# print(type(x))
# for k in x:
#   print(k)
# plot_fft(x,200,2048,22050)
# print(x.iloc[0])
x = df.audio[df.file=='blues.00000.wav'].iloc[0]
plot_fft(x,start=0,window_length=2048,max_freq=2000,color='b')
plot_fft(x,start=2205,window_length=2048,max_freq=2000,color='r')
plot_fft(x,start=4410,window_length=2048,max_freq=2000,color='g')
```



9.2.3 Spektrogram

Funkcja stft zwraca reprezentację signał w dziedzinie czasu i czestotliwości poprzez wyznaczanie transformacji Fouriera dla kolejnych okien (mogą się nakładać) .

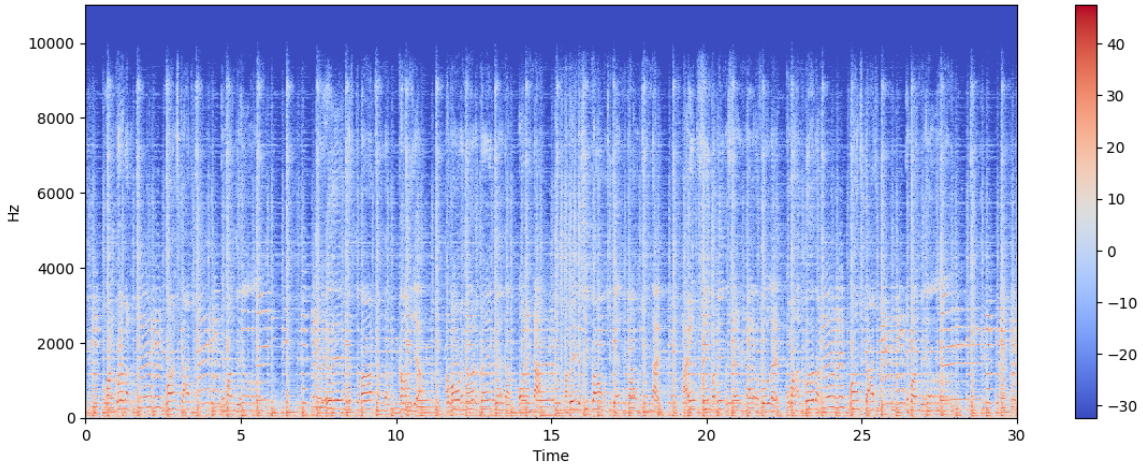
Cytując:

This function returns a complex-valued matrix D such that

- np.abs(D[f, t]) is the magnitude of frequency bin f at frame t, and
- np.angle(D[f, t]) is the phase of frequency bin f at frame t.

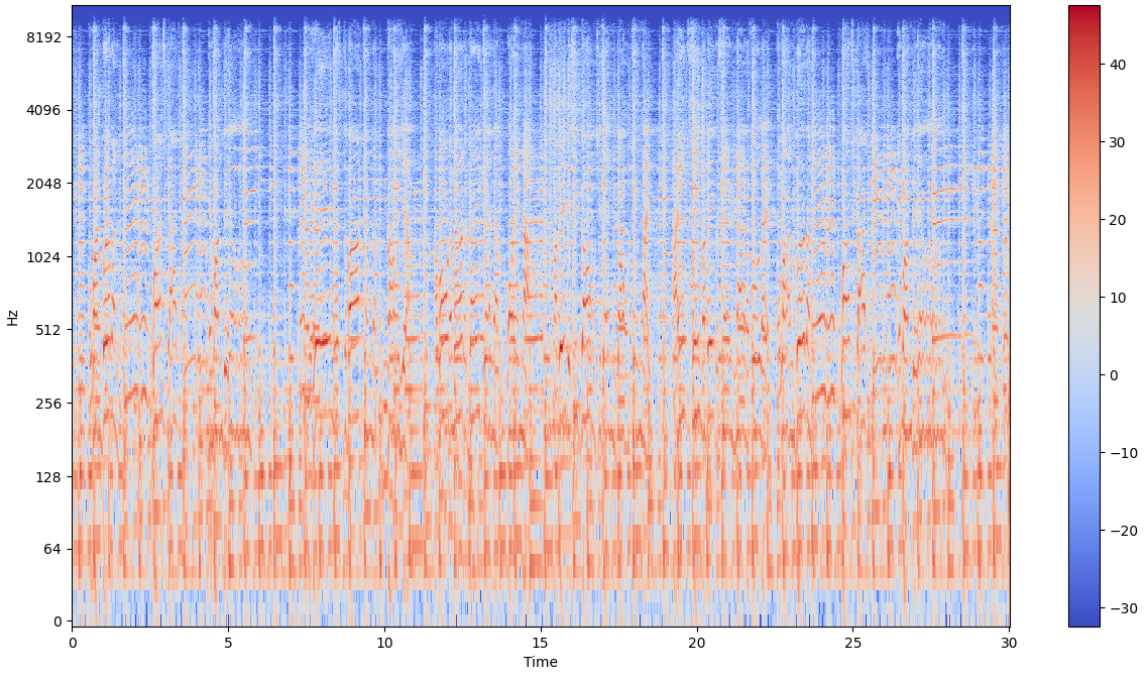
In [71]:
```
X = librosa.stft(x)
Xdb = librosa.amplitude_to_db(abs(X))
plt.figure(figsize=(14, 5))
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar()
print(X.shape)
```

```
(1025, 1293)
```



In [72]:
```
# zmiana na logarytmy
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='log')
```

```
plt.colorbar();
```



Analiza rozmiarów spektrogramu. Ma on 1293 kolumny. Czyli 1293 razy wykonano transformację FFT.

TODO 9.2.5

- Oblicz przesunięcie pomiędzy oknami

In [143…
```
# Skok pomiedzy oknami FFT (hop_length)
#duration = 30sek
print(f'Skok pomiędzy oknami:{30 *sr/1293} (tak naprawdę 512)')
```

```
Skok pomiędzy oknami:511.6009280742459 (tak naprawdę 512)
```

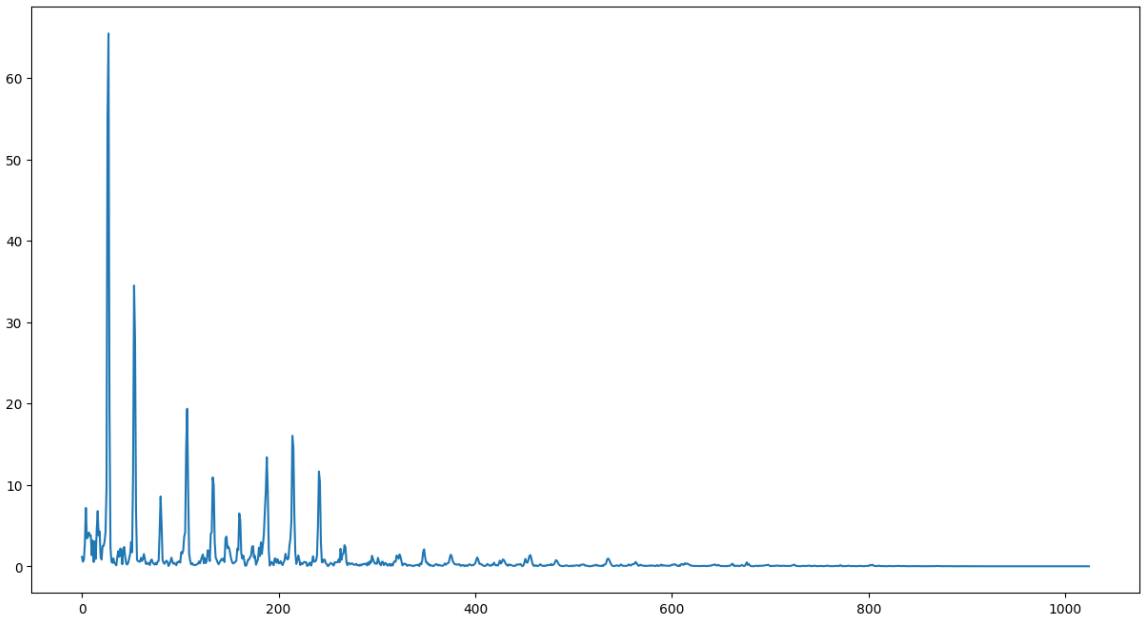
TODO 9.2.6

Narysuj diagram częstotliwości dla czasu t=5sec.

- częstość próbkowania sr=22050
- standardowa wielkośc przesunięcia okna dla funkcji stft 512
- która to będzie kolumna?

In [75]:
```
X = librosa.stft(x)
print(X.shape)
col=100
amp = X[:,col]

amp=np.abs(amp)
print(amp.shape)

plt.plot(np.arange(amp.shape[0]),amp)
plt.show()
```

```
(1025, 1293)
(1025,)
```



9.3. Wyznaczanie cech

Dla każdego nagrania wyznaczymy następujące cechy:

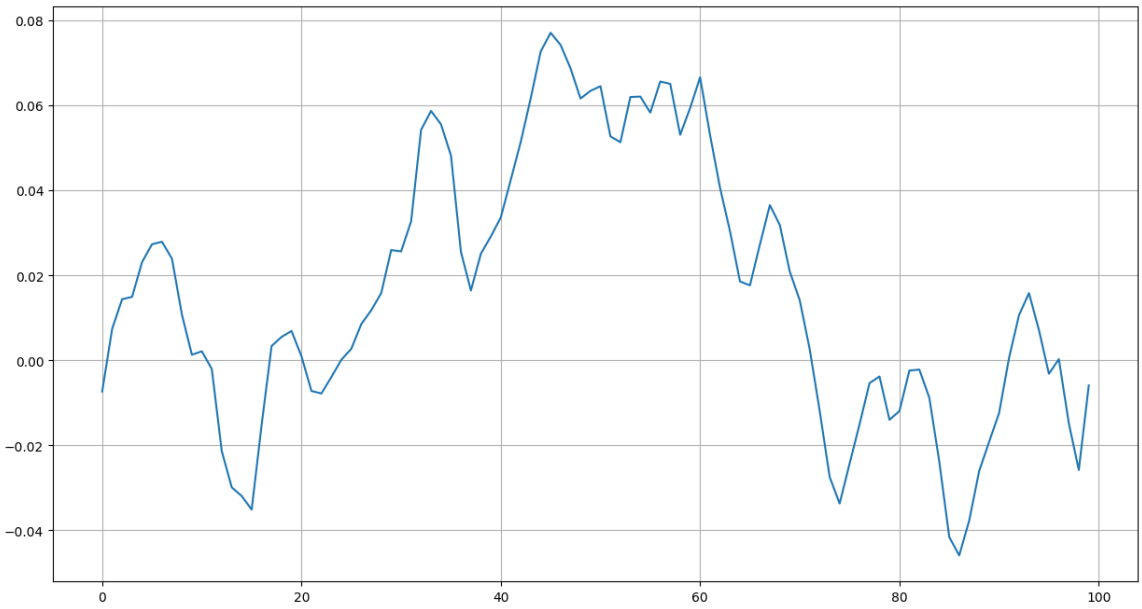
- Zero crossing rate
- Spectral centroid
- RMS
- MEL Frequency Cepstral Coefficients
- Chroma frequenices
- Tempo

Cechy te zostaną dodane jako kolumny do DataFrame. Wyznaczone zostaną wartości średnie i wariancje (w większości przypadków)

9.3.1 Zero crossing rate

Ile razy wykres przechodzi przez zero? Zwykle ma większe wartości dla utworów z dominującą perkusją.

In [76]:
```
plt.plot(x[300:400])
plt.grid()
```



In [77]:
```
zero_crossings = librosa.zero_crossings(x[300:400], pad=False)
print(zero_crossings)
print(sum(zero_crossings))
```

```
[False  True False False False False False False False False False  True
 False False False False False  True False False False  True False False
  True False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
  True False False False False False False False False False False False
 False False False False False False False  True False False False  True
  True  True False False]
10
```

Napiszemy funkcję, która pełny wektor sygnału podziel na pewną liczbę fragmentów (`n_splits`), obliczy dla każdego z nich wartości cech, a następnie wyznaczy wartość średnią i odchylenie standardowe.

In [78]:
```
def calculate_stats(x,compute_feature,n_splits):
  delta = x.shape[0]//n_splits
  ys = np.zeros(n_splits)
  for i in range(n_splits):
    if i<n_splits-1:
      y = compute_feature(x[i*delta:(i+1)*delta])
    else:
      y = compute_feature(x[i*delta:])
    ys[i]=y
  # print(ys)
  return ys.mean(),ys.std()

# print(x)
# x to jedna z wczesniej załadowanych tablic sygnału
calculate_stats(x,lambda z:librosa.zero_crossings(z, pad=False).sum()/z.s
```

Out[78]:
```
(0.08315264585719913, 0.010770225989245166)
```

Do funkcji `calculate_stats` przekazujemy obiekt funkcyjny `compute_feature` odpowiedzialny za obliczanie cechy. Może to być wyrażenie lambda.

- Tworzymy listę wartości średnich i odchyleń standardowych dla wszystkich plików
- Zamieniamy na `ndarray`
- Dodajemy do DataFrame

TODO 9.3.1 Dodaj wariancję, czyli kwadrat std

In [79]:
```
z = [calculate_stats(x,lambda x:librosa.zero_crossings(x, pad=False).sum(
z = np.array(z)
print(z.shape)
df['zero_crossing_rate_mean'] = z[:,0]
df['zero_crossing_rate_var'] = z[:,1]
df.head()
```

```
(100, 2)
```

Out[79]:

	genre	file	audio	sr	zero_crossing_rate_mean	zero_cros
0	reggae	reggae.00009.wav	[-0.054840088, -0.088256836, -0.08078003, -0.0...	22050	0.060647	
1	reggae	reggae.00007.wav	[-0.0132751465, -0.02557373, -0.022277832, -0....	22050	0.073970	
2	reggae	reggae.00008.wav	[0.036315918, 0.05316162, 0.048919678, 0.04876...	22050	0.069513	
3	reggae	reggae.00005.wav	[0.23120117, 0.25952148, 0.076690674, 0.069793...	22050	0.073097	
4	reggae	reggae.00002.wav	[-0.05203247, -0.088256836, -0.09857178, -0.12...	22050	0.075212	

In [80]:
```
print(df.shape)
# powinno być (100,6)
```

```
(100, 6)
```

In [87]:
```
x, sr = librosa.load(audio_path)

print(x)

# Compute spectral centroid
spectral_centroids = librosa.feature.spectral_centroid(y=x, sr=sr)[0]
```

```
[ 0.00732422  0.01660156  0.00762939 ... -0.05560303 -0.06106567
 -0.06417847]
```

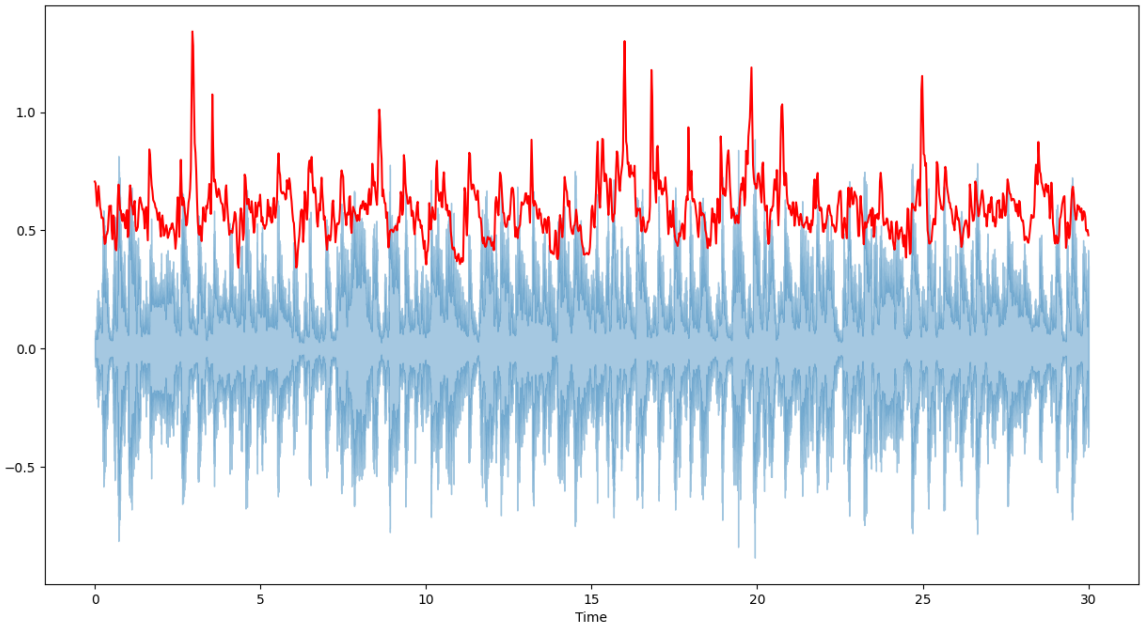
9.3.2 Spectral centroid

Cecha reprezentuje ważoną średnią częstotliwości obecnych w dźwieku. Wagą jest amplituda dla danej czestości.

In [88]:
```
spectral_centroids = librosa.feature.spectral_centroid(y=x, sr=sr)[0]
spectral_centroids.shape
```

Out[88]:
```
(1293,)
```

In [92]:
```
frames = range(len(spectral_centroids))
t = librosa.frames_to_time(frames)
librosa.display.waveshow(y=x, sr=sr, alpha=0.4)
spectral_centroids = spectral_centroids/(spectral_centroids.max()-spectra
plt.plot(t, spectral_centroids, color='r');
```



Jak widać wartości zmieniają się w czasie. Policzymy średnie wartości i wariancje dla każdego utworu. Użyta funkcja calculate_stats standardowo dzieli utwór na 10 fragmentów.

TODO 9.3.2

- Dodaj do DataFrame kolumny 'spectral_centroid_mean' i 'spectral_centroid_var'

In [94]:
```
z = [calculate_stats(x,lambda z:librosa.feature.spectral_centroid(y=z, sr
z = np.array(z)
df['spectral_centroid_mean']=z[:,0]
df['spectral_centroid_var']=z[:,1]
df.head()
```

Out[94]:

	genre	file	audio	sr	zero_crossing_rate_mean	zero_cros
0	reggae	reggae.00009.wav	[-0.054840088, -0.088256836, -0.08078003, -0.0...	22050	0.060647	
1	reggae	reggae.00007.wav	[-0.0132751465, -0.02557373, -0.022277832, -0....	22050	0.073970	
2	reggae	reggae.00008.wav	[0.036315918, 0.05316162, 0.048919678, 0.04876...	22050	0.069513	
3	reggae	reggae.00005.wav	[0.23120117, 0.25952148, 0.076690674, 0.069793...	22050	0.073097	
4	reggae	reggae.00002.wav	[-0.05203247, -0.088256836, -0.09857178, -0.12...	22050	0.075212	

In [95]:
```
# Powinno być (100,8)
df.shape
```

Out[95]:
```
(100, 8)
```

9.3.3.RMS

RMS (root mean square) to pierwiastek ze średniej kwadratów. Może zostać obliczony bezpośrednio z sygnału lub jego spektrogramu.

TODO 9.3.3

- Analogicznie jak w poprzednich punktach oblicz listę wartości wołając `calculate_stats` dla wszystkich ścieżek audio
- Dodaj kolumny 'rms_mean' i 'rms_var'

In [97]:
```
z = [calculate_stats(x, lambda z: librosa.feature.rms(y=z)[0].mean(), 10)
z = np.array(z)
df['rms_mean'] = z[:, 0]
df['rms_var'] = z[:, 1]**2
df.head()
```

Out[97]:

	genre	file	audio	sr	zero_crossing_rate_mean	zero_cros
0	reggae	reggae.00009.wav	[-0.054840088, -0.088256836, -0.08078003, -0.0...	22050	0.060647	
1	reggae	reggae.00007.wav	[-0.0132751465, -0.02557373, -0.022277832, -0....	22050	0.073970	
2	reggae	reggae.00008.wav	[0.036315918, 0.05316162, 0.048919678, 0.04876...	22050	0.069513	
3	reggae	reggae.00005.wav	[0.23120117, 0.25952148, 0.076690674, 0.069793...	22050	0.073097	
4	reggae	reggae.00002.wav	[-0.05203247, -0.088256836, -0.09857178, -0.12...	22050	0.075212	

In [98]:
```
# Powinno być (100,10)
df.shape
```

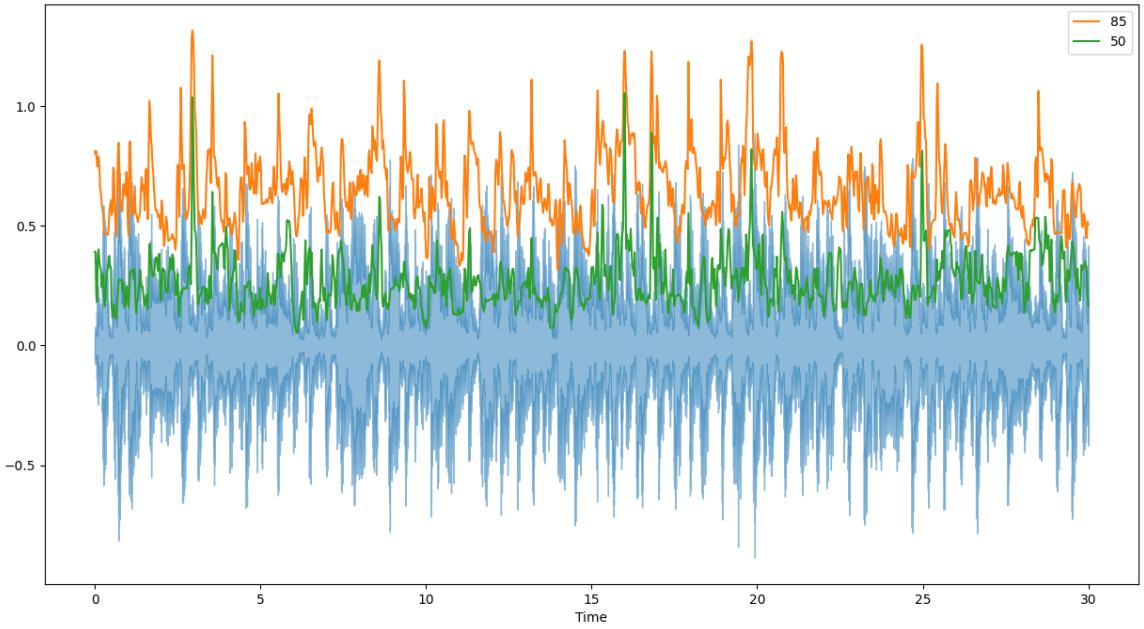
Out[98]:
```
(100, 10)
```

9.3.4 Spectral Rolloff

Cytując: *It is a measure of the shape of the signal. It represents the frequency below which a specified percentage of the total spectral energy, e.g. 85%, lies. librosa.feature.spectral_rolloff computes the rolloff frequency for each frame in a signal.*

In [102…
```
spectral_rolloff85 = librosa.feature.spectral_rolloff(y=x, sr=sr,roll_per
print(spectral_rolloff85.shape)
frames = range(spectral_rolloff85.shape[0])
t = librosa.frames_to_time(frames)

librosa.display.waveshow(x, sr=sr, alpha=0.5)
plt.plot(t, spectral_rolloff85/(spectral_rolloff85.max()-spectral_rolloff
spectral_rolloff50 = librosa.feature.spectral_rolloff(y=x, sr=sr,roll_per
plt.plot(t, spectral_rolloff50/(spectral_rolloff50.max()-spectral_rolloff
plt.legend()
plt.show()
```

```
(1293,)
```



TODO 9.3.4 Analogicznie, jak w w punktach 3.1, 3.2 i 3.3. dodajemy cechy do df

- wołamy calculate_stats dla wszystkich plików przekazując funkcję obliczającą spectral_rolloff (ustawiamy sr=sr, roll_percent=0.85)
- Dodajemy kolumny 'spectral_rolloff_mean' i 'spectral_rolloff_var'

In [103…
```
z = [calculate_stats(x, lambda z: librosa.feature.spectral_rolloff(y=z, s
z = np.array(z)
df['spectral_rolloff_mean'] = z[:, 0]
df['spectral_rolloff_var'] = z[:, 1]**2
df.head()
```

Out[103]:

	genre	file	audio	sr	zero_crossing_rate_mean	zero_cro
0	reggae	reggae.00009.wav	[-0.054840088, -0.088256836, -0.08078003, -0.0...	22050	0.060647	
1	reggae	reggae.00007.wav	[-0.0132751465, -0.02557373, -0.022277832, -0....	22050	0.073970	
2	reggae	reggae.00008.wav	[0.036315918, 0.05316162, 0.048919678, 0.04876...	22050	0.069513	
3	reggae	reggae.00005.wav	[0.23120117, 0.25952148, 0.076690674, 0.069793...	22050	0.073097	
4	reggae	reggae.00002.wav	[-0.05203247, -0.088256836, -0.09857178, -0.12...	22050	0.075212	

In [104…
```
# Powinno być (100,12)
df.shape
```

Out[104]: (100, 12)

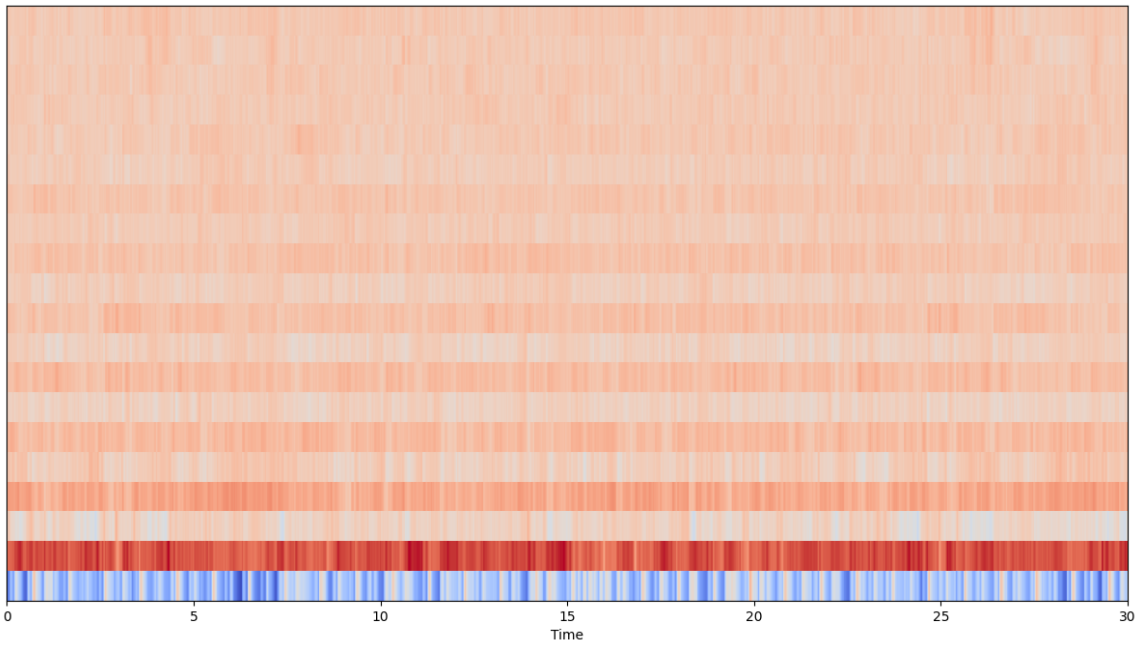
9.3.5 Mel Frequency Cepstral Coefficients

Cechy te powiązane są z nieliniową wrażliwością na słyszane częstotliwości. Patrz: szybkie wprowadzenie na medium.com

Standardowo oblicza się 20 cech...

In [106…
```
mfc = librosa.feature.mfcc(y=x, sr=sr)
print(mfc.shape)
#Displaying  the MFCCs:
librosa.display.specshow(mfc, sr=sr, x_axis='time');
```

```
(20, 1293)
```



In [107…
```
# wartości średnie i wariancje cech
print(mfc.mean(axis=1))
print(mfc.std(axis=1)**2)
```

```
[-113.59882     121.57067     -19.162262     42.36394      -6.362266
   18.621933    -13.699734     15.339802    -12.274304     10.970945
   -8.326061      8.802088     -3.6699414     5.7446756    -5.162783
    0.7517064    -1.687854     -0.40872997   -2.3026767     1.2224671 ]
[2569.3691    295.8471    235.58446   151.03873   167.99287    89.172424
   67.60309    69.001495   82.21985    63.346542   61.76499    51.28075
   41.2159     40.517525   49.784233   52.42454    36.535862   41.60317
   55.053654   46.941353]
```

Czyli dla każdego utworu otrzymamy dwudziestoelementowe wektrory średnich i wariancji. Utworzymy 2 * 20 kolumn DataFrame - mfcc01_mean ... mfcc20_mean i mfcc01_var ... mfcc20_var

In [109…
```
# przetwarzamy wszystkie cechy
def get_stats(mfc):
  return mfc.mean(axis=1),mfc.std(axis=1)**2

mfcs = [get_stats(librosa.feature.mfcc(y=x, sr=sr)) for x in df.audio]
```

Tworzymy słownik, którego kluczami są nazwy cech, a wartościami - listy zawierające elementy wyznaczone dla kolejnych utworów

TODO 9.3.5

- Dodaj zawartość słownika jako kolumny do DataFrame, np. wykonując: `df[klucz]=pd.Series(wartość)`

In [110…
```
dict = {}
for i in range(mfcs[0][0].shape[0]):
  dict[f'mfcc{i+1:02}_mean']=[]
  dict[f'mfcc{i+1:02}_var']=[]
for k in range(len(mfcs)):
  for i in range(mfcs[k][0].shape[0]):
    dict[f'mfcc{i+1:02}_mean'].append(mfcs[k][0][i])
  for i in range(mfcs[k][1].shape[0]):
    dict[f'mfcc{i+1:02}_var'].append(mfcs[k][1][i])

# print(dict)

for k in dict:
  if 'mean' in k:
    df[k] = dict[k]
  elif 'var' in k:
    df[k] = [v**2 for v in dict[k]]
df.head()
```

Out[110]:

	genre	file	audio	sr	zero_crossing_rate_mean	zero_cro
0	reggae	reggae.00009.wav	[-0.054840088, -0.088256836, -0.08078003, -0.0...	22050	0.060647	
1	reggae	reggae.00007.wav	[-0.0132751465, -0.02557373, -0.022277832, -0....	22050	0.073970	
2	reggae	reggae.00008.wav	[0.036315918, 0.05316162, 0.048919678, 0.04876...	22050	0.069513	
3	reggae	reggae.00005.wav	[0.23120117, 0.25952148, 0.076690674, 0.069793...	22050	0.073097	
4	reggae	reggae.00002.wav	[-0.05203247, -0.088256836, -0.09857178, -0.12...	22050	0.075212	

5 rows × 52 columns

In [111…
```
# Powinno być (100,52)
df.shape
```

Out[111]: (100, 52)

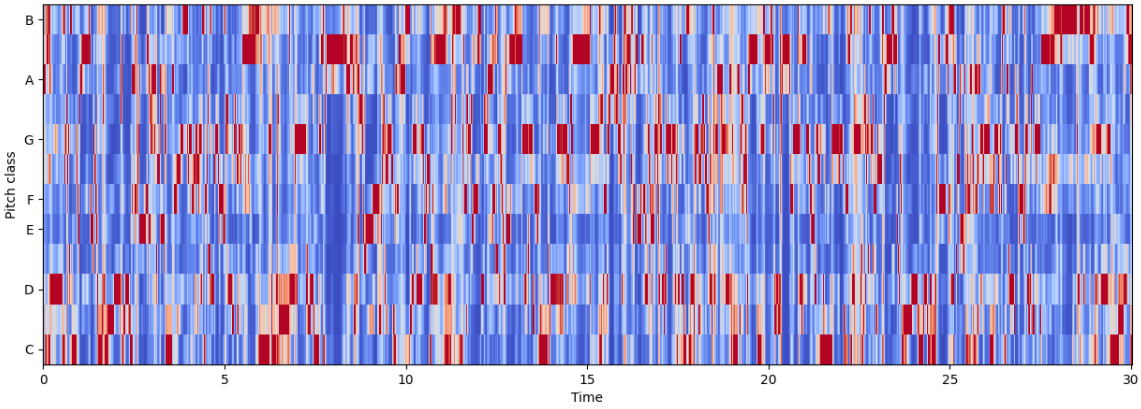
9.3.6 Częstości chromatyczne (Chroma Frequencies)

Reprezentują rozkład w widmie poszczególnych półtonów, np. A: 440HZ,880Hz, 1760Hz, itd.

Jest ich w sumie 12: C, C#, D, D#, E, F, F#, G, G#, A, A#, B

In [113…
```
hop_length=512 #skok pomiędzy oknami
chroma = librosa.feature.chroma_stft(y=x, sr=sr, hop_length=hop_length)
plt.figure(figsize=(15, 5))
librosa.display.specshow(chroma, x_axis='time', y_axis='chroma', hop_leng
print(chroma.shape)
```

```
(12, 1293)
```



Sytuacja jest podobna do MFCC. Dla każdego utworu otrzymujemy tablicę z 12 wierszami (odpowiadającymi półtonom) i kolumnami odpowidajacymi oknom.

TODO 9.3.6

- Dodaj kolumny chroma01_mean, chroma_01_var ... chroma12_mean, chroma_12_var do DataFrame

In [117…

```
chromas = [get_stats(librosa.feature.chroma_stft(y=x, sr=sr)) for x in df

dict = {}
for i in range(chromas[0][0].shape[0]):
  dict[f'chroma{i+1:02}_mean']=[]
  dict[f'chroma{i+1:02}_var']=[]
for k in range(len(chromas)):
  for i in range(chromas[k][0].shape[0]):
    dict[f'chroma{i+1:02}_mean'].append(chromas[k][0][i])
  for i in range(chromas[k][1].shape[0]):
    dict[f'chroma{i+1:02}_var'].append(chromas[k][1][i])

print(dict)

# for k in dict:
#   df[k]=pd.Series(dict[k])
# df.head()
```

{'chroma01_mean': [0.5174866, 0.41249347, 0.35975212, 0.46734732, 0.3842
359, 0.31111136, 0.3386019, 0.39132947, 0.25260174, 0.35418245, 0.089356
27, 0.3864317, 0.30302778, 0.20210567, 0.3541681, 0.22259255, 0.3990940
2, 0.22569439, 0.24007027, 0.44139007, 0.25935704, 0.24824715, 0.3413716
3, 0.31967524, 0.4600403, 0.23832263, 0.1926048, 0.42551336, 0.36224282,
0.27290025, 0.4332075, 0.4509705, 0.6607909, 0.3981771, 0.4608182, 0.317
72602, 0.4207031, 0.5565849, 0.31980312, 0.29642403, 0.2949808, 0.320894
8, 0.36216694, 0.54499733, 0.6355614, 0.37538671, 0.36042982, 0.3433648
3, 0.40850207, 0.50704503, 0.5529875, 0.383527, 0.47582224, 0.4467493,
0.41164145, 0.38014767, 0.18295933, 0.37302968, 0.26229402, 0.36861572,
0.37112907, 0.34603384, 0.5873929, 0.4293007, 0.5317407, 0.55835855, 0.3
5933948, 0.52361506, 0.3994632, 0.5081092, 0.34357986, 0.28090784, 0.474
95064, 0.35568413, 0.38722178, 0.31379935, 0.44006658, 0.30632704, 0.315
5818, 0.26329672, 0.124485895, 0.19733904, 0.33399972, 0.33005592, 0.586
98136, 0.16393, 0.32666084, 0.47982314, 0.18918535, 0.31518847, 0.392250
36, 0.49321443, 0.39989927, 0.43487057, 0.38821188, 0.5714958, 0.622304
4, 0.40285313, 0.3754905, 0.425365], 'chroma01_var': [0.116780676, 0.108
49945, 0.0964409, 0.12336567, 0.075223505, 0.05453974, 0.061499074, 0.08
104652, 0.047480237, 0.066079244, 0.018923752, 0.117432855, 0.11944905,
0.03466077, 0.098806165, 0.05713213, 0.14255388, 0.031528268, 0.1124601
3, 0.119150184, 0.062464725, 0.08754747, 0.06263444, 0.05289205, 0.14257
71, 0.08360195, 0.03813603, 0.13377604, 0.10695024, 0.04471613, 0.08727
6, 0.062738635, 0.089718364, 0.09352807, 0.10144186, 0.049006805, 0.0875
5497, 0.09834737, 0.064601615, 0.046236034, 0.087806016, 0.07351529, 0.0
9129373, 0.10869853, 0.11452909, 0.07384536, 0.06261017, 0.11703084, 0.0
948732, 0.08786588, 0.12942542, 0.09647824, 0.119409785, 0.11408973, 0.0
6021582, 0.08661594, 0.06105624, 0.080514245, 0.038841654, 0.07673774,
0.06464915, 0.06875523, 0.046591967, 0.049149044, 0.08335842, 0.05368842
6, 0.06578409, 0.05856433, 0.07335201, 0.079112574, 0.06776739, 0.041488
15, 0.077365115, 0.06091198, 0.07866874, 0.04179661, 0.08775842, 0.06055
2504, 0.036188208, 0.08810955, 0.014919879, 0.018755667, 0.14040458, 0.1
2264486, 0.1455881, 0.03288633, 0.115528256, 0.15406263, 0.08407132, 0.1
2642941, 0.076427706, 0.054093417, 0.071475714, 0.07422205, 0.078450695,
0.09865456, 0.07675695, 0.06650944, 0.067815065, 0.082778305], 'chroma02
_mean': [0.44544378, 0.35866407, 0.41665047, 0.37569106, 0.3753725, 0.46
482265, 0.50462025, 0.5062697, 0.44541147, 0.4864404, 0.07805213, 0.3635
7102, 0.16451626, 0.3675419, 0.32344395, 0.19598766, 0.32700226, 0.32342
395, 0.2583588, 0.3055276, 0.2641215, 0.27315456, 0.40862364, 0.3044870
2, 0.35018247, 0.30023736, 0.31726655, 0.22324328, 0.367817, 0.19595265,
0.48304316, 0.48139736, 0.45809615, 0.32069436, 0.37290853, 0.39294806,
0.49952257, 0.4800019, 0.3120322, 0.34806708, 0.31911126, 0.44570595, 0.
36138406, 0.31313172, 0.46344292, 0.4899829, 0.47747117, 0.25151965, 0.3
421114, 0.48393878, 0.37999168, 0.32721695, 0.43343723, 0.36252996, 0.54
124093, 0.35780054, 0.28341663, 0.33875045, 0.4336341, 0.38456544, 0.428
69535, 0.35391077, 0.62526494, 0.5540933, 0.5320152, 0.54711306, 0.37136
52, 0.5977527, 0.39673975, 0.42926884, 0.32982484, 0.42642567, 0.4291468
3, 0.36208785, 0.3239737, 0.23277384, 0.4111133, 0.3565917, 0.41513744,
0.3307701, 0.29797456, 0.42375132, 0.13985662, 0.22444354, 0.22405998,
0.24087314, 0.14008462, 0.20529716, 0.1412124, 0.15671287, 0.4013044, 0.
5511518, 0.35146877, 0.4696845, 0.3767389, 0.5505547, 0.5301131, 0.37710
983, 0.35759544, 0.39988568], 'chroma02_var': [0.11213992, 0.056954127,
0.09607618, 0.080362305, 0.08928712, 0.10678462, 0.13491803, 0.11214471,
0.12722386, 0.11596456, 0.016981639, 0.0901703, 0.05502304, 0.1396747,
0.08273562, 0.038229298, 0.09940679, 0.07384662, 0.13241473, 0.05678588
5, 0.0892781, 0.085732296, 0.056245096, 0.05379413, 0.088213086, 0.04594
6997, 0.103559606, 0.061143264, 0.081769824, 0.032730903, 0.096172154,
0.086722225, 0.05984672, 0.06638491, 0.07131179, 0.07464303, 0.09759615,
0.06011382, 0.050388508, 0.07565748, 0.08216899, 0.1184897, 0.08050212,
0.039645914, 0.053792376, 0.078802675, 0.094465986, 0.05466162, 0.066824
764, 0.047799345, 0.06838613, 0.06501442, 0.08230454, 0.079448834, 0.097

60533, 0.0626326, 0.086251296, 0.04724062, 0.13473944, 0.082364224, 0.08
1282936, 0.07699876, 0.047064, 0.09696277, 0.05127857, 0.04275469, 0.060
278364, 0.07121633, 0.069502115, 0.07206063, 0.03981888, 0.08519768, 0.0
9847002, 0.10048256, 0.04663703, 0.039174084, 0.071200475, 0.05781509,
0.06823488, 0.117690995, 0.089453526, 0.12372665, 0.014449136, 0.0285602
2, 0.049014743, 0.08922235, 0.024271639, 0.023895284, 0.03109611, 0.0255
62247, 0.08570204, 0.041234516, 0.059220318, 0.076456495, 0.07082684, 0.
08864443, 0.059903998, 0.07184987, 0.057380896, 0.08375687], 'chroma03_m
ean': [0.356203, 0.47051194, 0.5451907, 0.40673408, 0.44561923, 0.421115
2, 0.43181977, 0.2515375, 0.32408017, 0.33248565, 0.18301772, 0.3536470
8, 0.1663174, 0.26148355, 0.32724616, 0.29486653, 0.22919041, 0.4925756
8, 0.16547205, 0.32529682, 0.3253581, 0.4885124, 0.5243191, 0.36661544,
0.32080683, 0.61718655, 0.39202207, 0.13134763, 0.43983436, 0.31516543,
0.3572281, 0.39704555, 0.35554028, 0.36278656, 0.45280814, 0.50920933,
0.39799485, 0.57228863, 0.371588, 0.2833247, 0.3383049, 0.44266915, 0.35
53603, 0.31472477, 0.44832414, 0.49544105, 0.4031369, 0.24744122, 0.3675
627, 0.7297073, 0.32281658, 0.3898005, 0.4828674, 0.3626063, 0.41286615,
0.42703047, 0.44781056, 0.5640998, 0.26549777, 0.41135353, 0.38583356,
0.33965543, 0.6551111, 0.5102251, 0.6489445, 0.64572334, 0.48659256, 0.6
5344477, 0.4207513, 0.4773874, 0.4920904, 0.3355582, 0.3860817, 0.267298
55, 0.47947618, 0.21850558, 0.49564955, 0.6214756, 0.31909877, 0.2521614
4, 0.4111422, 0.36188352, 0.34869313, 0.42632505, 0.1952545, 0.38838485,
0.24897668, 0.22564027, 0.3422537, 0.25549224, 0.4163287, 0.7130407, 0.3
2045197, 0.48310336, 0.42931476, 0.5816284, 0.54069346, 0.47044718, 0.47
837624, 0.4121594], 'chroma03_var': [0.08803568, 0.10191797, 0.13777846,
0.08574446, 0.10328868, 0.09732322, 0.112467766, 0.037768193, 0.0788779
4, 0.082843326, 0.073676, 0.09942489, 0.05697193, 0.077058144, 0.0755539
6, 0.10260802, 0.07030794, 0.14777613, 0.040850814, 0.07748662, 0.093866
15, 0.15300876, 0.10638914, 0.083559334, 0.09506487, 0.14672934, 0.09741
653, 0.035159744, 0.09995377, 0.073728785, 0.062766045, 0.058123317, 0.0
5413299, 0.086651325, 0.09315501, 0.09170558, 0.06762715, 0.10336361, 0.
08400623, 0.061101694, 0.10480844, 0.07289344, 0.09742458, 0.09437617,
0.12001148, 0.104379185, 0.069886185, 0.059334055, 0.104552515, 0.096447
006, 0.067826316, 0.09730105, 0.10712778, 0.0832261, 0.059778996, 0.1023
47456, 0.15204845, 0.098589905, 0.049161654, 0.06951352, 0.05971451, 0.0
51825963, 0.05739546, 0.05580296, 0.07433627, 0.07765392, 0.09558475, 0.
07835226, 0.078621894, 0.086860314, 0.07385428, 0.06887358, 0.07973832,
0.04406958, 0.08828682, 0.036204293, 0.104733855, 0.1334554, 0.03842919
3, 0.07387993, 0.12276713, 0.10665293, 0.15032214, 0.137986, 0.05212039
5, 0.14265774, 0.11801038, 0.08509081, 0.10170353, 0.06805784, 0.0934953
9, 0.057901412, 0.052229594, 0.08566381, 0.10171085, 0.09475317, 0.06921
549, 0.11447864, 0.09573916, 0.071721725], 'chroma04_mean': [0.4176108,
0.41307116, 0.4196825, 0.48783603, 0.3488734, 0.36252296, 0.32442462, 0.
18557592, 0.24811676, 0.2907431, 0.2726137, 0.3693615, 0.25752336, 0.197
57304, 0.29787716, 0.34999147, 0.18744588, 0.23903397, 0.2599981, 0.4006
5512, 0.21773893, 0.26800498, 0.54701287, 0.42213294, 0.21147157, 0.2357
4118, 0.25582698, 0.15259737, 0.24469176, 0.40798622, 0.34059542, 0.5354
0313, 0.39239988, 0.45692766, 0.38606673, 0.43429136, 0.44637772, 0.4930
9754, 0.37147188, 0.37249914, 0.39390814, 0.51375663, 0.32183206, 0.2259
102, 0.33151257, 0.41691038, 0.36912033, 0.32053697, 0.28992176, 0.39711
782, 0.2577659, 0.32572132, 0.34017327, 0.4185846, 0.43760452, 0.3334311
2, 0.22137366, 0.36755, 0.27774677, 0.6037188, 0.34231097, 0.46343008,
0.59186804, 0.57508606, 0.48088947, 0.50972056, 0.4080366, 0.5890306, 0.
46804538, 0.50555456, 0.28118718, 0.3660879, 0.52029705, 0.32272112, 0.4
100727, 0.36114874, 0.36644205, 0.34400606, 0.44664735, 0.25247994, 0.20
596316, 0.23207875, 0.25355235, 0.4163694, 0.23415685, 0.15326962, 0.125
97182, 0.16869527, 0.39533067, 0.38871375, 0.3756684, 0.8043496, 0.37902
33, 0.4797589, 0.5307582, 0.5056281, 0.59571886, 0.4102829, 0.45156023,
0.48367321], 'chroma04_var': [0.10654058, 0.11003048, 0.107010715, 0.116
142884, 0.06327385, 0.086547874, 0.0716005, 0.037352093, 0.064089596, 0.

06280341, 0.15394558, 0.12770185, 0.10460903, 0.058968488, 0.07267802,
0.11783769, 0.07548941, 0.028893044, 0.1393096, 0.08574077, 0.07137223,
0.059118796, 0.092091426, 0.07555423, 0.052331932, 0.027091702, 0.075362
81, 0.047761384, 0.044646904, 0.07359393, 0.059472486, 0.07763698, 0.074
209906, 0.08294899, 0.055895843, 0.06478219, 0.09340944, 0.07706301, 0.0
66513255, 0.06844311, 0.061425265, 0.11824435, 0.07590824, 0.024118286,
0.055418838, 0.0585161, 0.061674375, 0.06913755, 0.04450714, 0.03255206
7, 0.047847826, 0.060255207, 0.061927363, 0.10491481, 0.071244046, 0.060
820762, 0.058114827, 0.04769303, 0.0997923, 0.110849604, 0.043426365, 0.
09385799, 0.050045036, 0.092758305, 0.054293156, 0.04975386, 0.04739814
6, 0.051949956, 0.0643881, 0.068281814, 0.02757888, 0.059633482, 0.09286
235, 0.06284554, 0.08084023, 0.056169935, 0.06141872, 0.041516654, 0.092
816345, 0.07203609, 0.04201595, 0.040730134, 0.08661027, 0.11821779, 0.0
7633447, 0.03742362, 0.013496372, 0.05603609, 0.13311416, 0.12447003, 0.
09567923, 0.072735675, 0.07969211, 0.086417355, 0.09297962, 0.079492524,
0.08229912, 0.0802281, 0.06783158, 0.08769847], 'chroma05_mean': [0.3041
1273, 0.35306802, 0.28403747, 0.3860547, 0.44407824, 0.5140892, 0.374433
5, 0.22227538, 0.41257972, 0.41026846, 0.13272263, 0.30178288, 0.1829036
5, 0.24138965, 0.36571166, 0.24935727, 0.1293044, 0.2726526, 0.17288487,
0.45663974, 0.21527727, 0.17971845, 0.65707505, 0.52381116, 0.20078844,
0.12398709, 0.1926635, 0.24195808, 0.247862, 0.54682016, 0.44472712, 0.4
8675805, 0.40725023, 0.5894338, 0.49693337, 0.49327213, 0.38258365, 0.38
01826, 0.5163387, 0.42997527, 0.47899717, 0.27510458, 0.28315225, 0.3788
8044, 0.40410286, 0.4765046, 0.33489344, 0.4762034, 0.5177227, 0.2893681
5, 0.27836245, 0.462056, 0.28770345, 0.40150884, 0.42456296, 0.39569744,
0.12728314, 0.27070186, 0.2124858, 0.45576027, 0.4594492, 0.4310474, 0.5
854507, 0.45433915, 0.45208815, 0.39822143, 0.40644, 0.6256999, 0.615144
43, 0.65340954, 0.3551693, 0.36974525, 0.3613886, 0.48402667, 0.4554076
8, 0.5383796, 0.4426751, 0.33014697, 0.41662502, 0.36021203, 0.3399756,
0.39317593, 0.2600847, 0.18138202, 0.17372917, 0.147495, 0.34184572, 0.1
7334239, 0.20491402, 0.18431315, 0.343718, 0.625223, 0.4239903, 0.487819
22, 0.62335074, 0.38393176, 0.63988876, 0.47132, 0.61405116, 0.5282621
4], 'chroma05_var': [0.055636853, 0.07918089, 0.05459117, 0.09025097, 0.
12668498, 0.10954829, 0.11125529, 0.031199664, 0.124633946, 0.0918887,
0.042340253, 0.06281011, 0.060143106, 0.09885125, 0.062282693, 0.0607695
73, 0.016082397, 0.072644666, 0.049429774, 0.10686661, 0.06534606, 0.044
481587, 0.1190286, 0.123187035, 0.043679167, 0.018846786, 0.03210751, 0.
04701489, 0.06566809, 0.12263048, 0.090599656, 0.086392544, 0.04445601,
0.11559323, 0.09117669, 0.12542458, 0.06748535, 0.04613599, 0.10229146,
0.101812154, 0.12694861, 0.047734164, 0.06694422, 0.07406589, 0.0763312
4, 0.11409897, 0.07130204, 0.112690054, 0.107163906, 0.049069088, 0.0497
24538, 0.112560734, 0.05305947, 0.0838653, 0.08044049, 0.096644856, 0.02
827702, 0.039876625, 0.061123412, 0.069851205, 0.068090886, 0.06419442,
0.03513122, 0.05466021, 0.04930544, 0.03600387, 0.092907764, 0.06458604,
0.08114966, 0.10300422, 0.06014455, 0.08829524, 0.069730446, 0.13359295,
0.052278608, 0.12122082, 0.09508586, 0.06995288, 0.092982575, 0.1337606
8, 0.1330817, 0.13505147, 0.089732505, 0.02936571, 0.033541635, 0.064112
13, 0.13607898, 0.044043656, 0.02333243, 0.025142472, 0.08057702, 0.0420
9195, 0.09396344, 0.07219598, 0.10117438, 0.06894776, 0.07908232, 0.0914
0542, 0.10016784, 0.09870515], 'chroma06_mean': [0.2790264, 0.47795433,
0.29334682, 0.34389335, 0.3977612, 0.4248709, 0.25325787, 0.44627023, 0.
37163144, 0.3681207, 0.100789316, 0.3507379, 0.22493047, 0.2031106, 0.55
55827, 0.3461802, 0.2398969, 0.35292754, 0.27947468, 0.33318192, 0.29576
185, 0.26772526, 0.5115236, 0.45111907, 0.3307942, 0.24376832, 0.530220
9, 0.4911694, 0.3356164, 0.32169873, 0.41409266, 0.33895254, 0.43124294,
0.41277862, 0.40072587, 0.48024282, 0.5157536, 0.53433686, 0.50701725,
0.32130235, 0.3133994, 0.22419515, 0.25592095, 0.5477351, 0.1978416, 0.3
3138686, 0.29249614, 0.48080093, 0.5221736, 0.27613235, 0.3521841, 0.480
447, 0.29968783, 0.51254183, 0.37697157, 0.3268187, 0.13074496, 0.321957
35, 0.21989907, 0.40114406, 0.41950804, 0.59057903, 0.6593953, 0.4514892

4, 0.45191303, 0.46589914, 0.36814952, 0.58958757, 0.5249317, 0.5277858
4, 0.43174508, 0.39160556, 0.3855161, 0.38064703, 0.64173454, 0.3554259
5, 0.351655, 0.2327682, 0.45482275, 0.26171848, 0.15979329, 0.1461064,
0.26660758, 0.17938241, 0.27642348, 0.11807121, 0.2416039, 0.32570818,
0.39033648, 0.27619144, 0.37097108, 0.4158607, 0.43623376, 0.5239951, 0.
37439674, 0.34286755, 0.6359424, 0.45537406, 0.47052065, 0.42201653], 'c
hroma06_var': [0.06979692, 0.11149568, 0.0712787, 0.1032277, 0.08497562,
0.093296275, 0.047764245, 0.09854015, 0.081799895, 0.08046316, 0.0226071
72, 0.1080081, 0.1165924, 0.038054135, 0.147674, 0.10561829, 0.05496534,
0.08596695, 0.15465741, 0.099820055, 0.08408145, 0.10779226, 0.09204026,
0.08998871, 0.09667875, 0.07653759, 0.13133419, 0.14840159, 0.08352276,
0.05557937, 0.076031774, 0.046971038, 0.06669434, 0.07782486, 0.072726,
0.089309596, 0.08977729, 0.093252026, 0.08037792, 0.06802225, 0.05073193
5, 0.052475907, 0.05559771, 0.118110016, 0.022252599, 0.08073692, 0.0519
24113, 0.11903312, 0.12319942, 0.05578408, 0.1194542, 0.11392261, 0.0705
1988, 0.1374612, 0.04866032, 0.069072224, 0.042432297, 0.07750155, 0.035
459764, 0.09079185, 0.052638385, 0.095661335, 0.04881722, 0.0730066, 0.0
78750096, 0.054806724, 0.059843395, 0.05582153, 0.06932604, 0.06666751,
0.07854662, 0.055220187, 0.083707124, 0.07925895, 0.089344636, 0.0515453
93, 0.06356534, 0.038870305, 0.05613816, 0.08001665, 0.01761051, 0.01255
9912, 0.12092671, 0.057389382, 0.09779919, 0.032937303, 0.10591687, 0.13
299493, 0.12813297, 0.10526505, 0.082455635, 0.05079833, 0.08221069, 0.0
8515277, 0.063123114, 0.06916885, 0.06641149, 0.09678203, 0.095908396,
0.077368975], 'chroma07_mean': [0.23929925, 0.34712198, 0.33251348, 0.35
323784, 0.47149163, 0.38027573, 0.26838568, 0.39263576, 0.42031556, 0.41
58314, 0.2944042, 0.33848426, 0.1724937, 0.32710817, 0.37338066, 0.37177
765, 0.20703113, 0.441164, 0.109900475, 0.21843949, 0.1900517, 0.2225857
2, 0.35555357, 0.49344274, 0.39737874, 0.27965158, 0.25377938, 0.2455084
6, 0.36461875, 0.30465698, 0.41809464, 0.42672065, 0.35200676, 0.3537508
5, 0.37628847, 0.57519823, 0.4917932, 0.38702035, 0.57351184, 0.4171330
3, 0.27673936, 0.27863634, 0.3339038, 0.6073608, 0.23306726, 0.2833211,
0.37752077, 0.44954658, 0.33342746, 0.29312137, 0.39221734, 0.33977783,
0.32758075, 0.4132656, 0.52365524, 0.34935617, 0.17604223, 0.3861594, 0.
51614064, 0.33865806, 0.56187093, 0.4850164, 0.6037233, 0.46406275, 0.41
55394, 0.5296032, 0.4144415, 0.52757514, 0.43126178, 0.45598888, 0.48092
815, 0.58974, 0.3128998, 0.38942766, 0.42213127, 0.3782779, 0.40360895,
0.36993125, 0.6393338, 0.22777973, 0.33692718, 0.23258813, 0.20028205,
0.23259695, 0.19078134, 0.25473484, 0.122894265, 0.17470205, 0.17472416,
0.16458249, 0.30665493, 0.29882807, 0.48413977, 0.53719974, 0.3315267,
0.33554924, 0.5602837, 0.3936851, 0.45571363, 0.38719505], 'chroma07_va
r': [0.043787822, 0.065916486, 0.06415968, 0.081121646, 0.10967893, 0.09
896089, 0.087715395, 0.11371868, 0.1356883, 0.10332962, 0.16252607, 0.08
035159, 0.04994435, 0.13768722, 0.07164822, 0.14057948, 0.057989504, 0.0
9996941, 0.03415791, 0.038879707, 0.045994606, 0.047405407, 0.05366433,
0.107459255, 0.07982033, 0.065977745, 0.043003816, 0.05963341, 0.0768479
4, 0.058482356, 0.089683376, 0.069012254, 0.03994734, 0.06728331, 0.0497
9356, 0.097563565, 0.09215393, 0.05051103, 0.09933601, 0.11640078, 0.076
153375, 0.073637, 0.05784627, 0.078130186, 0.03836746, 0.057041526, 0.07
9904735, 0.084428705, 0.0641608, 0.04940344, 0.051025912, 0.057570953,
0.07166874, 0.076701514, 0.09532813, 0.077865586, 0.060100194, 0.0533298
2, 0.12616283, 0.08177337, 0.096786164, 0.07688473, 0.03875896, 0.075420
596, 0.050003625, 0.05981553, 0.061224714, 0.0501592, 0.053156823, 0.075
88373, 0.05520007, 0.11913903, 0.047615662, 0.09011611, 0.044690754, 0.0
7661646, 0.076479584, 0.07574672, 0.096573755, 0.077433996, 0.13016729,
0.08390813, 0.057889502, 0.061285663, 0.03941106, 0.1105778, 0.03505398,
0.035233226, 0.024515457, 0.017409872, 0.074531265, 0.05479556, 0.078869
19, 0.07908024, 0.06289802, 0.06139913, 0.057337064, 0.074103996, 0.0865
73616, 0.068627864], 'chroma08_mean': [0.3854239, 0.30324706, 0.4295306
5, 0.48692426, 0.36443385, 0.27617556, 0.22458448, 0.3991012, 0.2717581
4, 0.37915307, 0.19620019, 0.37530917, 0.22814904, 0.27825257, 0.277144

8, 0.20975909, 0.18869376, 0.21440458, 0.11421242, 0.36186188, 0.180749
4, 0.41535112, 0.25161317, 0.35275283, 0.560355, 0.4112927, 0.13912144,
0.16911277, 0.43568376, 0.2884035, 0.3551178, 0.52598065, 0.48512018, 0.
38463858, 0.55245745, 0.4785637, 0.41706282, 0.3859248, 0.43117473, 0.32
01564, 0.30829057, 0.3266592, 0.5450671, 0.53672993, 0.27600208, 0.32555
5, 0.520325, 0.48153147, 0.31746212, 0.29485935, 0.6124786, 0.47913158,
0.41156384, 0.5119891, 0.4097741, 0.39353353, 0.20240055, 0.71050245, 0.
31997252, 0.34271055, 0.5057575, 0.43412542, 0.65086097, 0.40482742, 0.5
485609, 0.49465892, 0.54105777, 0.60000557, 0.524373, 0.39995435, 0.7535
15, 0.39428928, 0.42753583, 0.2803484, 0.4380605, 0.32234073, 0.522922,
0.50710285, 0.42126295, 0.22168617, 0.19643375, 0.14149223, 0.32216415,
0.32187557, 0.29723093, 0.16138996, 0.12996168, 0.2469055, 0.2759339, 0.
3620685, 0.28789714, 0.24181683, 0.5075426, 0.541391, 0.35206988, 0.3340
5113, 0.48081514, 0.46964008, 0.46837425, 0.42968538], 'chroma08_var':
[0.10455458, 0.08445534, 0.12879658, 0.1300229, 0.0739747, 0.054199874,
0.043986715, 0.06276152, 0.054019164, 0.07999416, 0.078912966, 0.0910585
3, 0.07566712, 0.08749958, 0.075400986, 0.056038693, 0.04730485, 0.02439
6017, 0.050465446, 0.07947071, 0.057419874, 0.14531057, 0.039754063, 0.0
65997876, 0.1345516, 0.11315267, 0.031364273, 0.03747183, 0.11980225, 0.
06320127, 0.056267705, 0.06402157, 0.06519262, 0.081583366, 0.091272086,
0.07752286, 0.07547638, 0.06844427, 0.0914636, 0.06300509, 0.070840314,
0.0923578, 0.12731174, 0.09325585, 0.040803727, 0.08631464, 0.10860499,
0.08591869, 0.09637512, 0.053204257, 0.10728814, 0.12980206, 0.10158495
6, 0.11160719, 0.061160937, 0.086514734, 0.07942651, 0.08794949, 0.06098
1497, 0.071374275, 0.088959776, 0.06433455, 0.04731983, 0.04952419, 0.08
792985, 0.061941046, 0.12039862, 0.055455595, 0.07129453, 0.08470449, 0.
086638585, 0.07112137, 0.06673542, 0.038721416, 0.09528217, 0.049620844,
0.10963869, 0.11362237, 0.06077276, 0.053602304, 0.057872023, 0.02127047
6, 0.11120128, 0.0996475, 0.085424595, 0.049756195, 0.041241657, 0.08754
6065, 0.08274161, 0.109294586, 0.065701865, 0.06052448, 0.072776094, 0.0
8901484, 0.0652365, 0.06585535, 0.0669857, 0.09976848, 0.08338401, 0.073
41941], 'chroma09_mean': [0.48208088, 0.28591293, 0.29040015, 0.2578464,
0.24920328, 0.34032732, 0.3566976, 0.57967097, 0.3011969, 0.4877448, 0.1
3535582, 0.3022559, 0.40398523, 0.34814957, 0.23290308, 0.20922892, 0.24
313661, 0.28309312, 0.11190101, 0.3874543, 0.26434094, 0.27268866, 0.246
27055, 0.38013843, 0.38417926, 0.2439858, 0.27642432, 0.28835577, 0.2959
8564, 0.33415222, 0.4727927, 0.65576535, 0.48392802, 0.429819, 0.4851255
7, 0.42939878, 0.57719076, 0.2899447, 0.36462137, 0.45373178, 0.4129998
4, 0.4365419, 0.51203793, 0.29794616, 0.3138011, 0.3884531, 0.476082, 0.
4856616, 0.355702, 0.42637405, 0.37937832, 0.40071708, 0.42162582, 0.459
95763, 0.4324963, 0.42253676, 0.21219686, 0.44133937, 0.3086515, 0.52867
05, 0.46480155, 0.53909224, 0.6981493, 0.4533675, 0.42633212, 0.4822285,
0.4569781, 0.69137526, 0.59776074, 0.30652115, 0.46447614, 0.46357313,
0.5793956, 0.39095148, 0.2958624, 0.44787923, 0.37062052, 0.3788372, 0.3
800397, 0.3736512, 0.17463093, 0.28141227, 0.11452202, 0.2131321, 0.3157
634, 0.090703845, 0.1832983, 0.19397454, 0.30637044, 0.42971838, 0.36184
505, 0.26997653, 0.5721115, 0.5058325, 0.41296372, 0.33598712, 0.4436726
6, 0.528683, 0.46523315, 0.49120003], 'chroma09_var': [0.12684567, 0.065
67376, 0.06422719, 0.045623466, 0.05441358, 0.06934745, 0.0970221, 0.121
378034, 0.07145772, 0.09600464, 0.053869825, 0.059919395, 0.15796676, 0.
10826538, 0.047822613, 0.07624644, 0.07149465, 0.08916633, 0.05247373,
0.096965484, 0.0786708, 0.0700815, 0.046257216, 0.0701336, 0.08719859,
0.055331204, 0.0920575, 0.08771961, 0.06438381, 0.061145034, 0.0672933,
0.08930542, 0.053044252, 0.07159867, 0.06594875, 0.07839331, 0.09821083
4, 0.043924853, 0.057053857, 0.10881401, 0.09916911, 0.10599477, 0.07425
226, 0.06517842, 0.046186406, 0.06693018, 0.054427303, 0.111208394, 0.07
028689, 0.06515482, 0.05161042, 0.07458277, 0.07951146, 0.099049374, 0.0
7652293, 0.08569879, 0.04612462, 0.053638775, 0.10375272, 0.115274765,
0.07573546, 0.09953596, 0.04453303, 0.07319989, 0.042375367, 0.05044642
5, 0.060403973, 0.060511082, 0.08002646, 0.047421694, 0.048248205, 0.106

70466, 0.115207896, 0.104160465, 0.039614305, 0.09774057, 0.059913084,
0.057107285, 0.085434034, 0.09968075, 0.043703817, 0.07628907, 0.0093083
61, 0.057736572, 0.119596094, 0.014727589, 0.07027013, 0.058571182, 0.10
508593, 0.120582074, 0.06450386, 0.07744203, 0.09087908, 0.079037555, 0.
07772714, 0.057218157, 0.062244482, 0.12671661, 0.06938314, 0.08904805],
'chroma10_mean': [0.40864834, 0.34957677, 0.35495663, 0.2097586, 0.23226
468, 0.47948587, 0.413219, 0.3095621, 0.38434148, 0.5609498, 0.21231632,
0.31865048, 0.28505918, 0.25812003, 0.25720868, 0.21515235, 0.3116547,
0.22319578, 0.15232152, 0.32182175, 0.39047354, 0.18117544, 0.31511128,
0.46170583, 0.25532576, 0.2855334, 0.32432652, 0.46184453, 0.3150803, 0.
40181875, 0.5631661, 0.41836998, 0.61469495, 0.46828473, 0.5380487, 0.41
263062, 0.45680454, 0.30312383, 0.44412747, 0.41620374, 0.49694663, 0.32
671455, 0.461979, 0.2860336, 0.48369595, 0.50401485, 0.5345362, 0.441355
74, 0.36598998, 0.6149525, 0.27565488, 0.41143388, 0.5607322, 0.3568102
7, 0.46730247, 0.4765828, 0.49763522, 0.4023943, 0.2406592, 0.3714434,
0.47979236, 0.4192797, 0.71717745, 0.45091358, 0.4366054, 0.53951997, 0.
46651432, 0.609115, 0.6141311, 0.33935633, 0.45571843, 0.32929918, 0.398
90066, 0.29855517, 0.38344565, 0.3798183, 0.41117132, 0.4702666, 0.26149
18, 0.4660133, 0.32131815, 0.28685394, 0.20057935, 0.25459903, 0.1564723
8, 0.17595632, 0.3101378, 0.3135335, 0.25835255, 0.25386262, 0.46989205,
0.33026215, 0.5597266, 0.5443018, 0.48875007, 0.49890006, 0.46527228, 0.
52190363, 0.5391836, 0.56754947], 'chroma10_var': [0.111201435, 0.069496
38, 0.08704917, 0.042854454, 0.056814283, 0.1417806, 0.119408965, 0.0588
8601, 0.11447098, 0.12143844, 0.13272788, 0.06774778, 0.07783793, 0.0764
96415, 0.06827964, 0.06781842, 0.058625385, 0.03082993, 0.045983136, 0.0
6768231, 0.07275321, 0.0471448, 0.08628567, 0.09437403, 0.05437556, 0.07
924659, 0.054561164, 0.12478093, 0.0787414, 0.110964596, 0.10456155, 0.0
6175194, 0.08191678, 0.09753709, 0.103353515, 0.10714129, 0.07780734, 0.
05182778, 0.1011347, 0.11369155, 0.1288795, 0.07016238, 0.1202456, 0.098
89863, 0.14329332, 0.12115716, 0.1236187, 0.1319292, 0.09442032, 0.09197
25, 0.07470738, 0.10066007, 0.09988163, 0.08298077, 0.08243219, 0.092872
22, 0.16049388, 0.090927854, 0.050089825, 0.05573405, 0.083575875, 0.057
98258, 0.049192026, 0.06507775, 0.060860552, 0.057367753, 0.10197221, 0.
054241985, 0.09053668, 0.046589453, 0.06384491, 0.06292786, 0.05560591,
0.072435826, 0.07168848, 0.0831732, 0.098967716, 0.10552918, 0.03583243
5, 0.15644, 0.11068476, 0.08771416, 0.06715888, 0.071749315, 0.03137676
4, 0.07501616, 0.12451651, 0.0907378, 0.03927209, 0.06153574, 0.0859089
2, 0.086332396, 0.11565732, 0.089074016, 0.1011361, 0.095200434, 0.06607
7866, 0.11396788, 0.105503954, 0.12527403], 'chroma11_mean': [0.3855431
7, 0.48064512, 0.38059616, 0.28245488, 0.30479977, 0.36652896, 0.288769
2, 0.27302843, 0.26606017, 0.39088565, 0.15576074, 0.45982927, 0.293305,
0.25895694, 0.23892243, 0.31059855, 0.44209412, 0.36977658, 0.33516994,
0.2842525, 0.6486375, 0.22990079, 0.31657925, 0.3663595, 0.28477612, 0.2
8433675, 0.55512387, 0.3592757, 0.4070083, 0.38438946, 0.4195213, 0.4541
507, 0.69520754, 0.32534432, 0.39453915, 0.367924, 0.45252132, 0.4446343
8, 0.33521265, 0.38951576, 0.3939189, 0.30688277, 0.34720138, 0.1452792
7, 0.32157752, 0.2874396, 0.39421514, 0.30945173, 0.2082015, 0.45172176,
0.2602577, 0.33825043, 0.54671925, 0.36819088, 0.5129991, 0.3589262, 0.3
3310422, 0.3287984, 0.38187972, 0.33800855, 0.38296723, 0.46245602, 0.65
102804, 0.50699025, 0.4704269, 0.54714406, 0.38769683, 0.48623645, 0.446
24975, 0.40984365, 0.39459983, 0.3145764, 0.43052408, 0.26630712, 0.4870
3796, 0.4103556, 0.33256075, 0.27344307, 0.38873166, 0.293119, 0.1465154
3, 0.17807446, 0.16780025, 0.28167403, 0.18855399, 0.21687399, 0.1827246
1, 0.34277746, 0.4685547, 0.25837117, 0.6071159, 0.34773877, 0.5111296,
0.45666584, 0.44899806, 0.5551941, 0.5595348, 0.45731765, 0.4175224, 0.5
249425], 'chroma11_var': [0.09972086, 0.11805496, 0.10156879, 0.07213119
4, 0.05332104, 0.075446755, 0.059190094, 0.091720656, 0.055519406, 0.075
22145, 0.043671254, 0.1162155, 0.12250471, 0.0891107, 0.052508615, 0.107
51565, 0.14593188, 0.08736847, 0.17162216, 0.07305766, 0.12303476, 0.094
13932, 0.07895538, 0.06874424, 0.077855945, 0.08103719, 0.12661976, 0.11

0884756, 0.11086457, 0.063252136, 0.059204254, 0.061863404, 0.08248978,
0.05039492, 0.05680953, 0.06752977, 0.083117306, 0.08088764, 0.05552171,
0.067566, 0.079127684, 0.08395071, 0.07305073, 0.016156964, 0.04697283,
0.05150659, 0.06432433, 0.0752411, 0.04483215, 0.077614926, 0.04680956,
0.07870342, 0.10547554, 0.091523655, 0.07740595, 0.069008514, 0.09590331
5, 0.060679477, 0.071970366, 0.06969323, 0.05887029, 0.08746332, 0.04522
8086, 0.07937051, 0.07471809, 0.05631926, 0.046736676, 0.041964285, 0.06
779818, 0.052807, 0.07263948, 0.061905865, 0.08409242, 0.047575213, 0.12
682255, 0.06345083, 0.050927036, 0.042418838, 0.066829465, 0.06672819,
0.014125492, 0.016682502, 0.039968055, 0.09334052, 0.0735069, 0.05153620
2, 0.055634003, 0.119361244, 0.14724797, 0.08380381, 0.12242474, 0.06997
524, 0.09666904, 0.074086115, 0.086527236, 0.0813734, 0.06539239, 0.0854
2613, 0.07155809, 0.08459076], 'chroma12_mean': [0.37594, 0.37986737, 0.
31940922, 0.32029834, 0.5225845, 0.35595685, 0.3311938, 0.2315236, 0.250
42713, 0.34547764, 0.21216895, 0.36774823, 0.29645246, 0.25924012, 0.332
80793, 0.32220843, 0.30847466, 0.3945492, 0.166403, 0.29697266, 0.436591
1, 0.24054769, 0.38319886, 0.4647148, 0.33409372, 0.36621904, 0.2729307,
0.3071861, 0.38510066, 0.58850825, 0.47741115, 0.5169851, 0.60572886, 0.
4241641, 0.4596858, 0.45552915, 0.3931781, 0.47897375, 0.3970207, 0.5371
542, 0.35725564, 0.23605071, 0.32186845, 0.31572777, 0.41216758, 0.27479
994, 0.3661411, 0.2739369, 0.28072846, 0.47702622, 0.4293575, 0.3377640
8, 0.4258135, 0.34144068, 0.50478256, 0.39097992, 0.20550352, 0.4135603,
0.43975386, 0.2999361, 0.39103666, 0.34267604, 0.591477, 0.46793488, 0.4
5739514, 0.60582787, 0.40468046, 0.484269, 0.41304326, 0.58663803, 0.372
48135, 0.34534782, 0.3507538, 0.465997, 0.3348094, 0.6344665, 0.4441679
7, 0.33295572, 0.50987875, 0.26204762, 0.3131666, 0.45996684, 0.2025061
2, 0.18486089, 0.29773337, 0.46804577, 0.30405736, 0.206462, 0.1720149,
0.17686196, 0.51729506, 0.39502698, 0.49762335, 0.4243017, 0.43483222,
0.6111458, 0.65282315, 0.43349352, 0.43590006, 0.49936062], 'chroma12_va
r': [0.06351311, 0.08077622, 0.069406636, 0.06750983, 0.1326713, 0.08041
8326, 0.08652668, 0.045238853, 0.06726221, 0.07568842, 0.13828354, 0.071
24978, 0.10240879, 0.09624111, 0.101830855, 0.12951396, 0.07408424, 0.10
669987, 0.05514503, 0.06637896, 0.07709392, 0.076015934, 0.113742545, 0.
097662605, 0.07245459, 0.09373134, 0.047706522, 0.07322713, 0.08566732,
0.12460926, 0.10024733, 0.09385727, 0.05394221, 0.08911177, 0.085183345,
0.07862303, 0.0744745, 0.07340898, 0.082700245, 0.116359115, 0.08122763,
0.051975213, 0.057347562, 0.060362317, 0.060057033, 0.05274756, 0.077978
07, 0.055187803, 0.050853174, 0.11558714, 0.075126715, 0.06640764, 0.080
26969, 0.06844422, 0.09159647, 0.09205064, 0.07322055, 0.07806715, 0.103
345655, 0.05554686, 0.08223337, 0.052438054, 0.04382856, 0.07119662, 0.0
53195935, 0.0822905, 0.08767245, 0.051271137, 0.069087684, 0.09601203,
0.049256865, 0.080278724, 0.055087514, 0.10402962, 0.048260886, 0.113536
69, 0.07999903, 0.08127821, 0.114886805, 0.08961972, 0.13536301, 0.15009
892, 0.04962221, 0.027076067, 0.07033279, 0.18270767, 0.101635724, 0.026
349723, 0.027935598, 0.03208747, 0.09036216, 0.048443865, 0.090542704,
0.06814123, 0.08601311, 0.10481898, 0.09788167, 0.07563892, 0.070423424,
0.088266924]}

In [120…

```
for k in dict:
    df[k] = dict[k]
df.head()
```

Out[120]:

	genre	file	audio	sr	zero_crossing_rate_mean	zero_cro
0	reggae	reggae.00009.wav	[-0.054840088, -0.088256836, -0.08078003, -0.0...	22050	0.060647	
1	reggae	reggae.00007.wav	[-0.0132751465, -0.02557373, -0.022277832, -0....	22050	0.073970	
2	reggae	reggae.00008.wav	[0.036315918, 0.05316162, 0.048919678, 0.04876...	22050	0.069513	
3	reggae	reggae.00005.wav	[0.23120117, 0.25952148, 0.076690674, 0.069793...	22050	0.073097	
4	reggae	reggae.00002.wav	[-0.05203247, -0.088256836, -0.09857178, -0.12...	22050	0.075212	

5 rows × 76 columns

In [121…
```
# Powinno być (100,76)
df.shape
```

Out[121]: (100, 76)

9.3.7 Tempo utworu

Tempo utworu jest mierzone w BPM (beats per minute) - czyli liczbie taktów na minutę. Okreslanie tempa opiera się na wykrywaniu początków nut (ang. onset detection), a następnie próbie dopasowania do nich tempa. Więcej informacji

In [123…
```
# Można to zrobić w jednym kroku przyjmujące standardowe wartości
x = df.audio[df.file=='blues.00000.wav'].iloc[0]
tempo = librosa.beat.tempo(y=x)
print(tempo)
```

```
<ipython-input-123-5e763db20671>:3: FutureWarning: librosa.beat.tempo
        This function was moved to 'librosa.feature.rhythm.tempo' in lib
rosa version 0.10.0.
        This alias will be removed in librosa version 1.0.
  tempo = librosa.beat.tempo(y=x)
[123.046875]
```

In [124…
```
# lub dostroić parametry
hop_length = 512 #wielkość okna
oenv = librosa.onset.onset_strength(y=x, sr=sr, hop_length=hop_length)
tempo = librosa.beat.tempo(onset_envelope=oenv, sr=sr,hop_length=hop_leng
print(tempo)
```

```
<ipython-input-124-b67d9c9b2965>:4: FutureWarning: librosa.beat.tempo
        This function was moved to 'librosa.feature.rhythm.tempo' in lib
rosa version 0.10.0.
        This alias will be removed in librosa version 1.0.
  tempo = librosa.beat.tempo(onset_envelope=oenv, sr=sr,hop_length=hop_l
ength)[0]
123.046875
```

Poniżej kod i ilustracje z dokumentacji librosa. Poziome paski na tempogramie to wielokrotności taktów, ale też półnuty, ćwierćnuty, ósemki, itd.

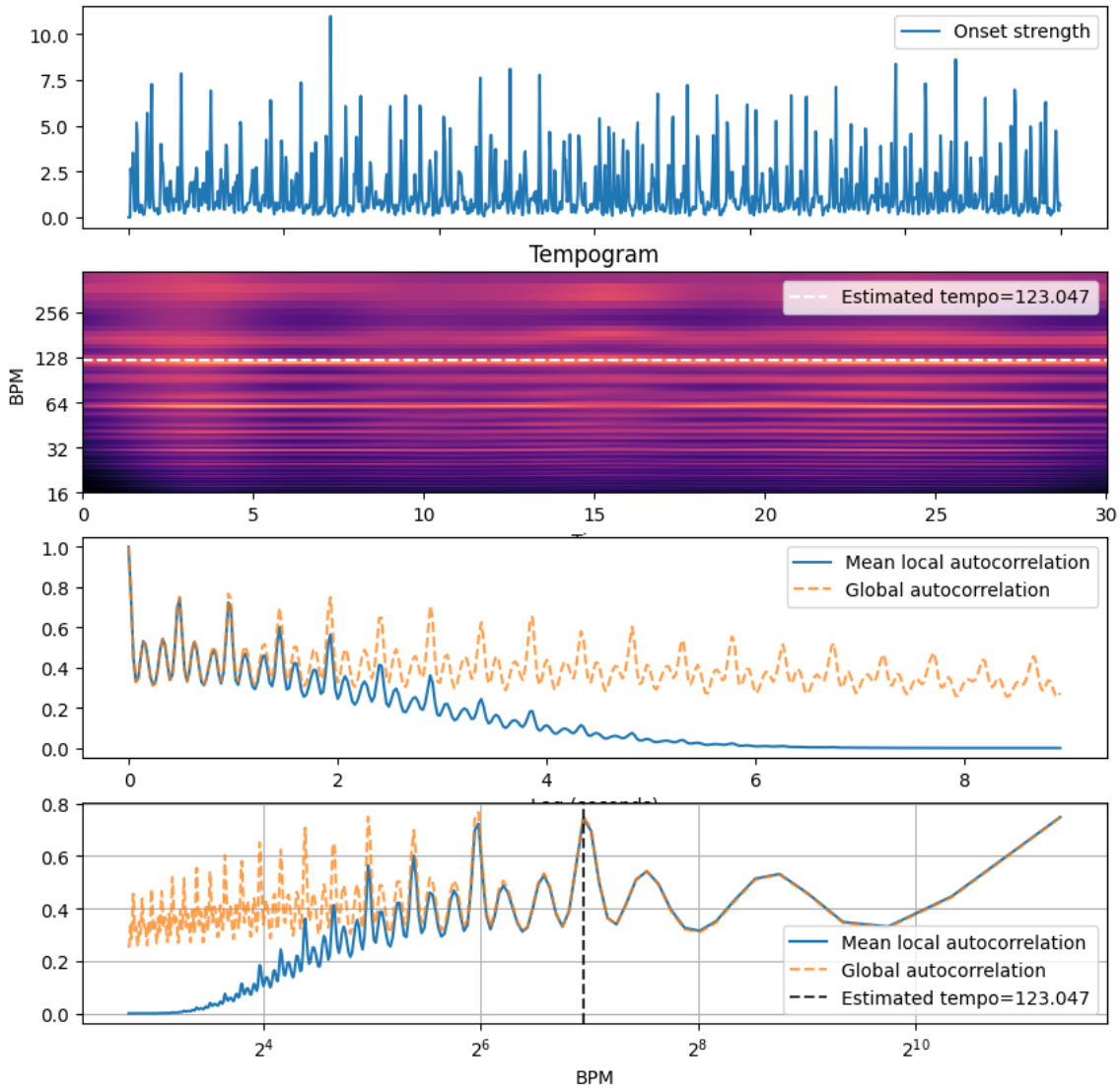
In [125…
```
hop_length = 512
oenv = librosa.onset.onset_strength(y=x, sr=sr, hop_length=hop_length)
tempogram = librosa.feature.tempogram(onset_envelope=oenv, sr=sr,
                                      hop_length=hop_length)
# Compute global onset autocorrelation
ac_global = librosa.autocorrelate(oenv, max_size=tempogram.shape[0])
ac_global = librosa.util.normalize(ac_global)
# Estimate the global tempo for display purposes
tempo = librosa.beat.tempo(onset_envelope=oenv, sr=sr,hop_length=hop_leng

print(tempogram.shape)
print(tempo)
```

```
(384, 1293)
123.046875
```

```
<ipython-input-125-672bdd624ae1>:9: FutureWarning: librosa.beat.tempo
        This function was moved to 'librosa.feature.rhythm.tempo' in lib
rosa version 0.10.0.
        This alias will be removed in librosa version 1.0.
  tempo = librosa.beat.tempo(onset_envelope=oenv, sr=sr,hop_length=hop_l
ength)[0]
```

In [127…
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(nrows=4, figsize=(10, 10))
times = librosa.times_like(oenv, sr=sr, hop_length=hop_length)
ax[0].plot(times, oenv, label='Onset strength')
ax[0].label_outer()
ax[0].legend(frameon=True)
librosa.display.specshow(tempogram, sr=sr, hop_length=hop_length,
                         x_axis='time', y_axis='tempo', cmap='magma',
                         ax=ax[1])
ax[1].axhline(tempo, color='w', linestyle='--', alpha=1,
              label='Estimated tempo={:g}'.format(tempo))
ax[1].legend(loc='upper right')
ax[1].set(title='Tempogram')
x = np.linspace(0, tempogram.shape[0] * float(hop_length) / sr,
                num=tempogram.shape[0])
ax[2].plot(x, np.mean(tempogram, axis=1), label='Mean local autocorrelati
ax[2].plot(x, ac_global, '--', alpha=0.75, label='Global autocorrelation'
ax[2].set(xlabel='Lag (seconds)')
ax[2].legend(frameon=True)
freqs = librosa.tempo_frequencies(tempogram.shape[0], hop_length=hop_leng
ax[3].semilogx(freqs[1:], np.mean(tempogram[1:], axis=1),
               label='Mean local autocorrelation', base=2)
ax[3].semilogx(freqs[1:], ac_global[1:], '--', alpha=0.75,
               label='Global autocorrelation', base=2)
ax[3].axvline(tempo, color='black', linestyle='--', alpha=.8,
              label='Estimated tempo={:g}'.format(tempo))
```

```
ax[3].legend(frameon=True)
ax[3].set(xlabel='BPM')
ax[3].grid(True)
```



TODO 9.3.7

1. Utwórz listę wartości tempa dla wszystkich utworów w df.audio
2. Dodaj kolumnę `tempo`

In [131…

```
tempos = [librosa.beat.tempo(y=x)[0] for x in df.audio]
df['tempo'] = tempos
df.head()
```

```
<ipython-input-131-3189031723d9>:1: FutureWarning: librosa.beat.tempo
        This function was moved to 'librosa.feature.rhythm.tempo' in lib
rosa version 0.10.0.
        This alias will be removed in librosa version 1.0.
  tempos = [librosa.beat.tempo(y=x)[0] for x in df.audio]
```

Out[131]:

	genre	file	audio	sr	zero_crossing_rate_mean	zero_cro
0	reggae	reggae.00009.wav	[-0.054840088, -0.088256836, -0.08078003, -0.0...	22050	0.060647	
1	reggae	reggae.00007.wav	[-0.0132751465, -0.02557373, -0.022277832, -0....	22050	0.073970	
2	reggae	reggae.00008.wav	[0.036315918, 0.05316162, 0.048919678, 0.04876...	22050	0.069513	
3	reggae	reggae.00005.wav	[0.23120117, 0.25952148, 0.076690674, 0.069793...	22050	0.073097	
4	reggae	reggae.00002.wav	[-0.05203247, -0.088256836, -0.09857178, -0.12...	22050	0.075212	

5 rows × 77 columns

TODO 9.3.8

Sprawdź liczbę wierszy i kolumn. Oczekiwany wynik (100,77)

In [132…
```
df.shape
```

Out[132]: (100, 77)

4. Zapisz do pliku

TODO 9.3.9

Zapisz DataFrame do pliku 'gitzan_small_features.csv' i pobierz plik.

In [133…
```
df.to_csv('gitzan_small_features.csv', index=False)

# pobieranie pliku
from google.colab import files
files.download('gitzan_small_features.csv')
```