MUSIC/AUDIO ANALYSIS IN PYTHON

Vivek Jayaram

WHY AUDIO SIGNAL PROCESSING?

- My background as a DJ and CS student
- Music is everywhere! So many possibilities
- Many parallels to computer vision

SOME APPLICATIONS

- Shazam - How does it recognize songs?

SOME APPLICATIONS

- Speech to Text – Siri, Android

OTHER APPLICATIONS

- Classify a song into genre
- Find interesting segments of songs
- Recommendations
- Generate Audio Automatically

OVERVIEW

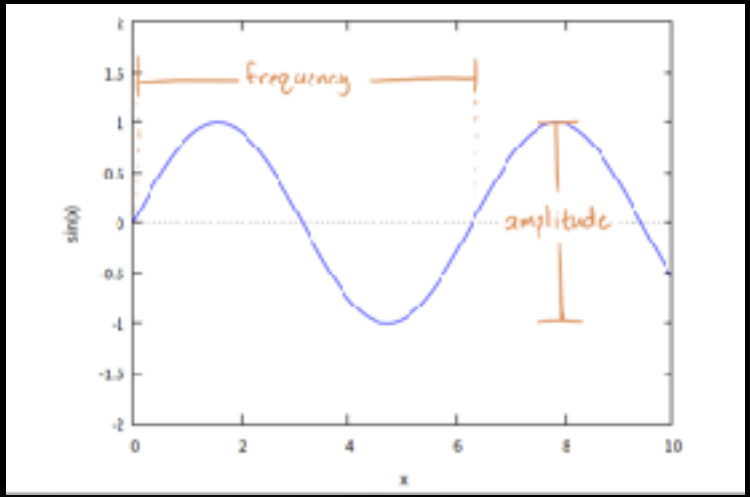
- Basics of Audio
- Sampling and Representation
- Fourier Transformations
- Building an Auto-DJ Software
- Finding Interesting Segments of Songs

BASICS OF AUDIO

Waves and Frequencies

BASICS OF SOUND

- Most basic type of sound is a sine wave

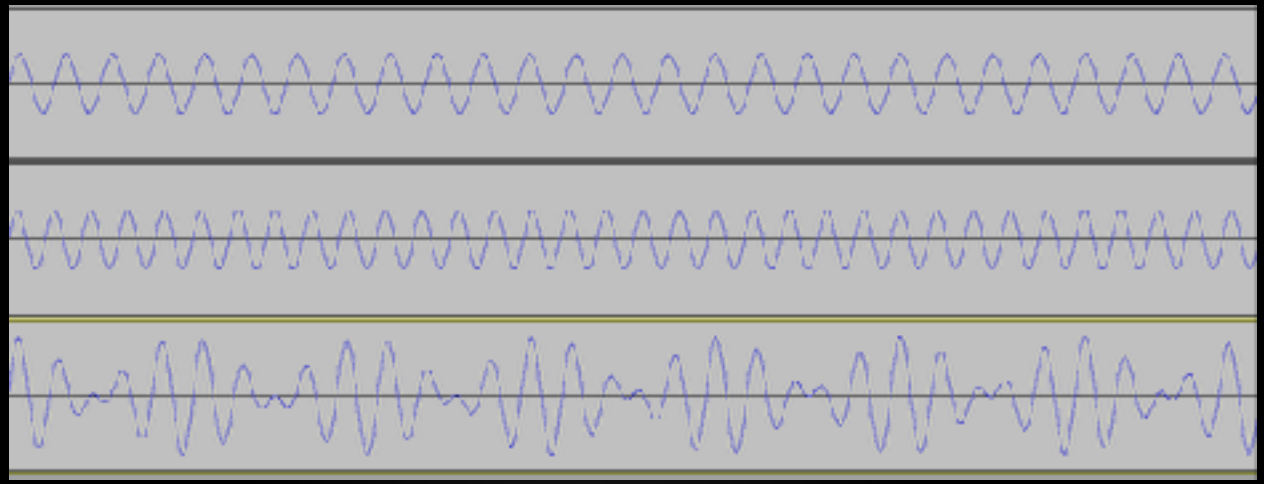


BASICS OF SOUND

- Frequency determines pitch, amplitude determines volume
- Doubling the frequency creates octave (same note)
- “Nice” ratios generally make nice intervals

BASICS OF SOUND

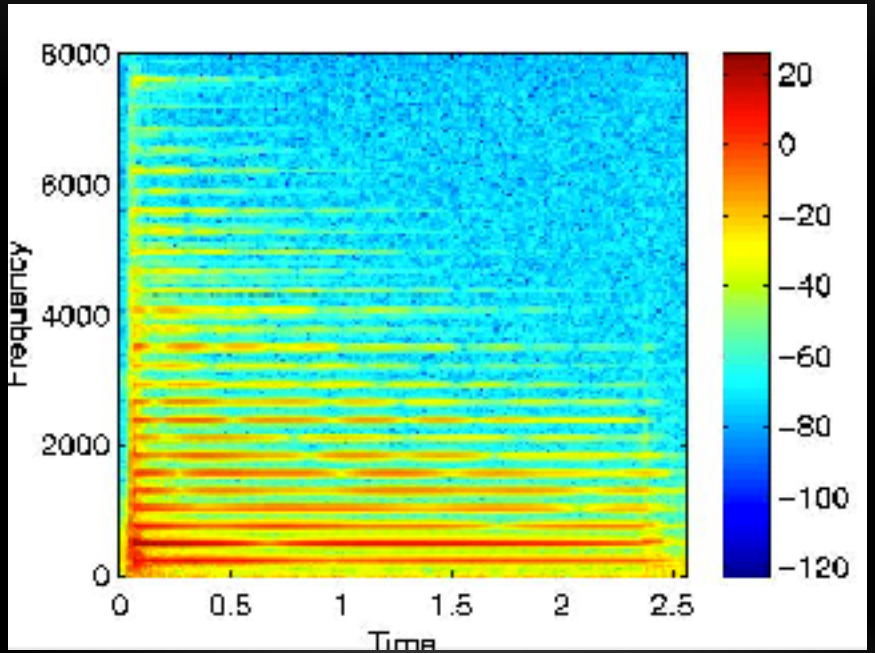
- Can combine sine waves to make intervals (Perfect Fifth Below)



WHAT MAKES A SOUND DISTINCT?

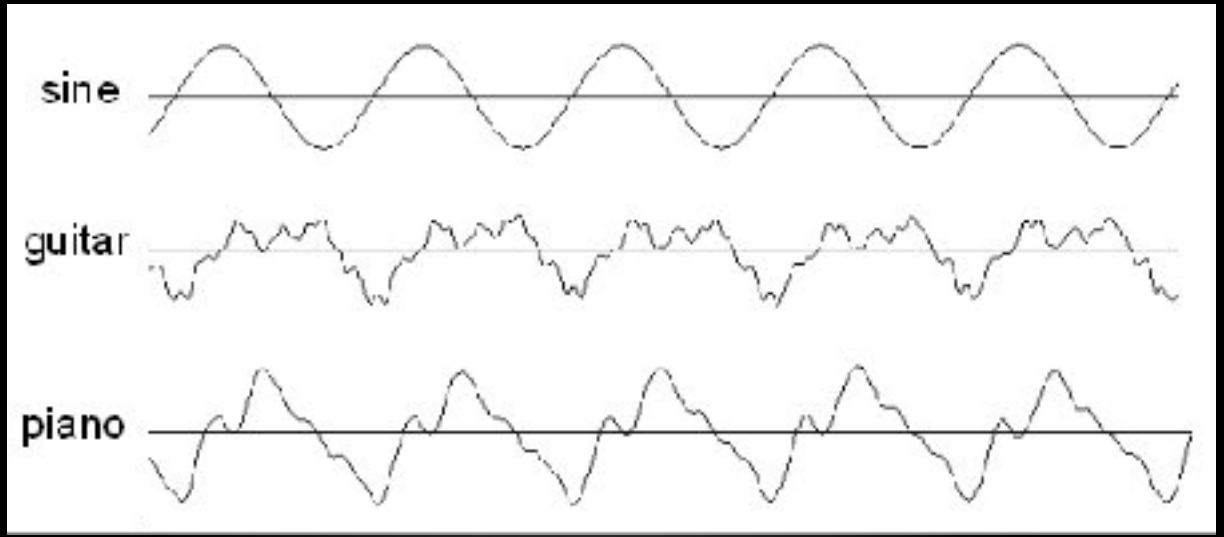
- If a piano and guitar playing A are both 440, why do they sound different?
- Each has different amount of *overtones*
- Frequencies at 440hz, 880hz, 1320hz, 1760hz …
- Ratio of each determines timbre

WHAT MAKES A SOUND DISTINCT?



WHAT MAKES A SOUND DISTINCT

- Add all those sine waves together

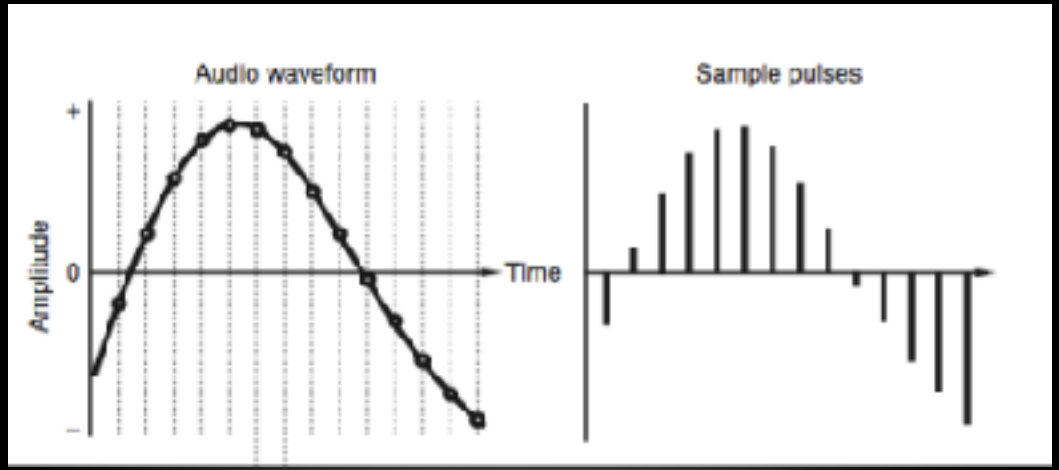


HOW IS AUDIO STORED IN COMPUTERS?

Sampling and Representation

SAMPLING

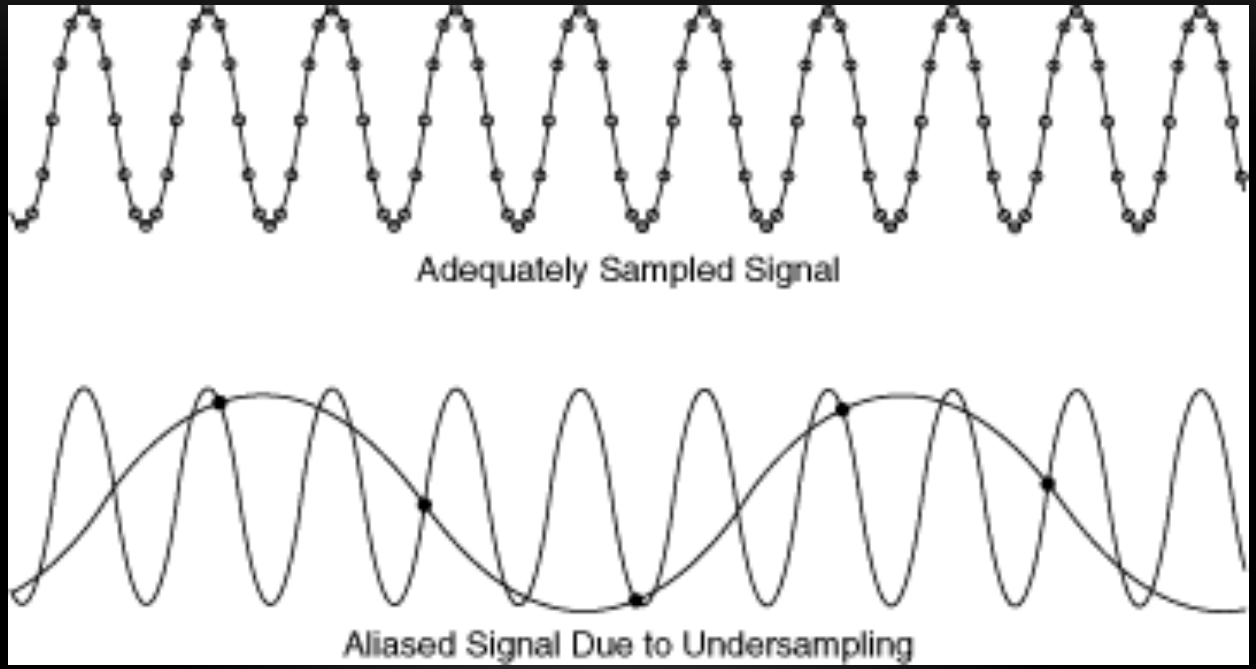
- Sound in the real world is a continuous wave
- Computers are discrete. Need to sample



SAMPLING

- Music is just array of heights sampled at regular intervals
- Music normally sampled at 44khz
- Space vs quality tradeoff
- Issues with high frequencies

SAMPLING

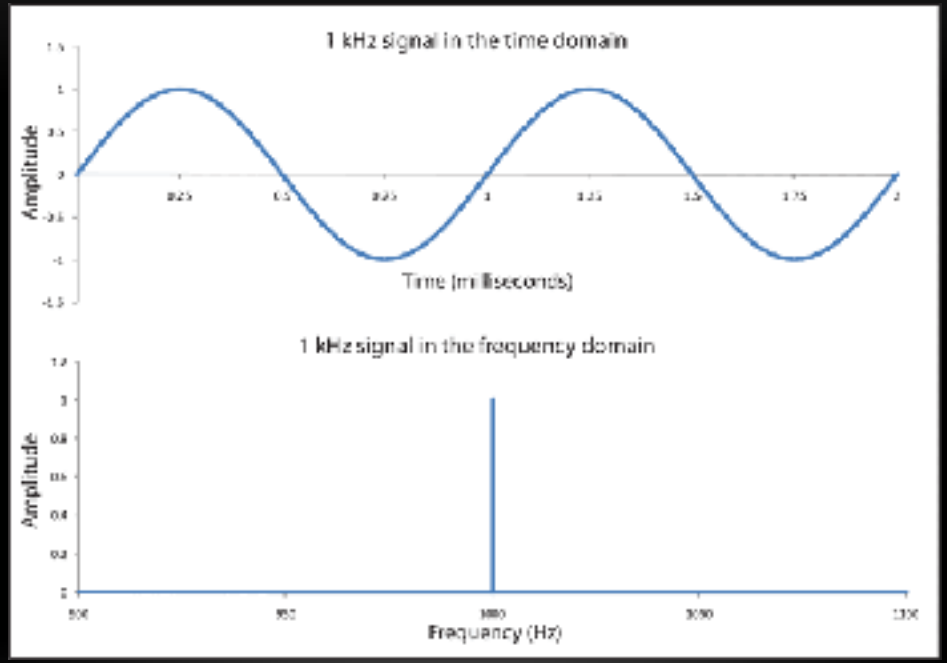

Adequately Sampled Signal
Aliased Signal Due to Undersampling

FOURIER TRANSFORMS

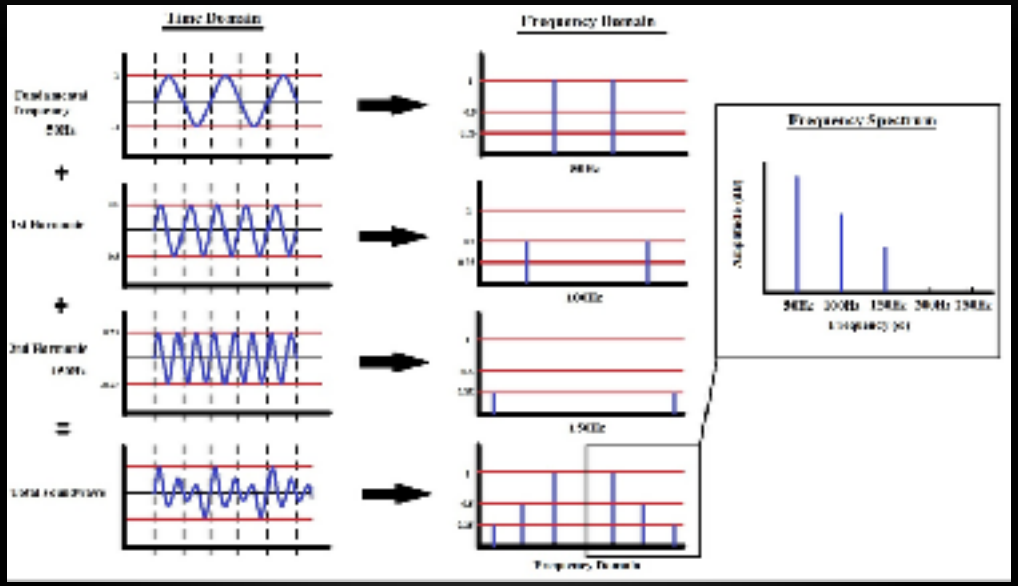
MOTIVATIONS

- Array of numbers doesn’t tell us much about audio
- Want a more representative feature
- Frequency is everything
- Can we get frequencies?

FOURIER TRANSFORM



FOURIER TRANSFORM



FOURIER TRANSFORMS

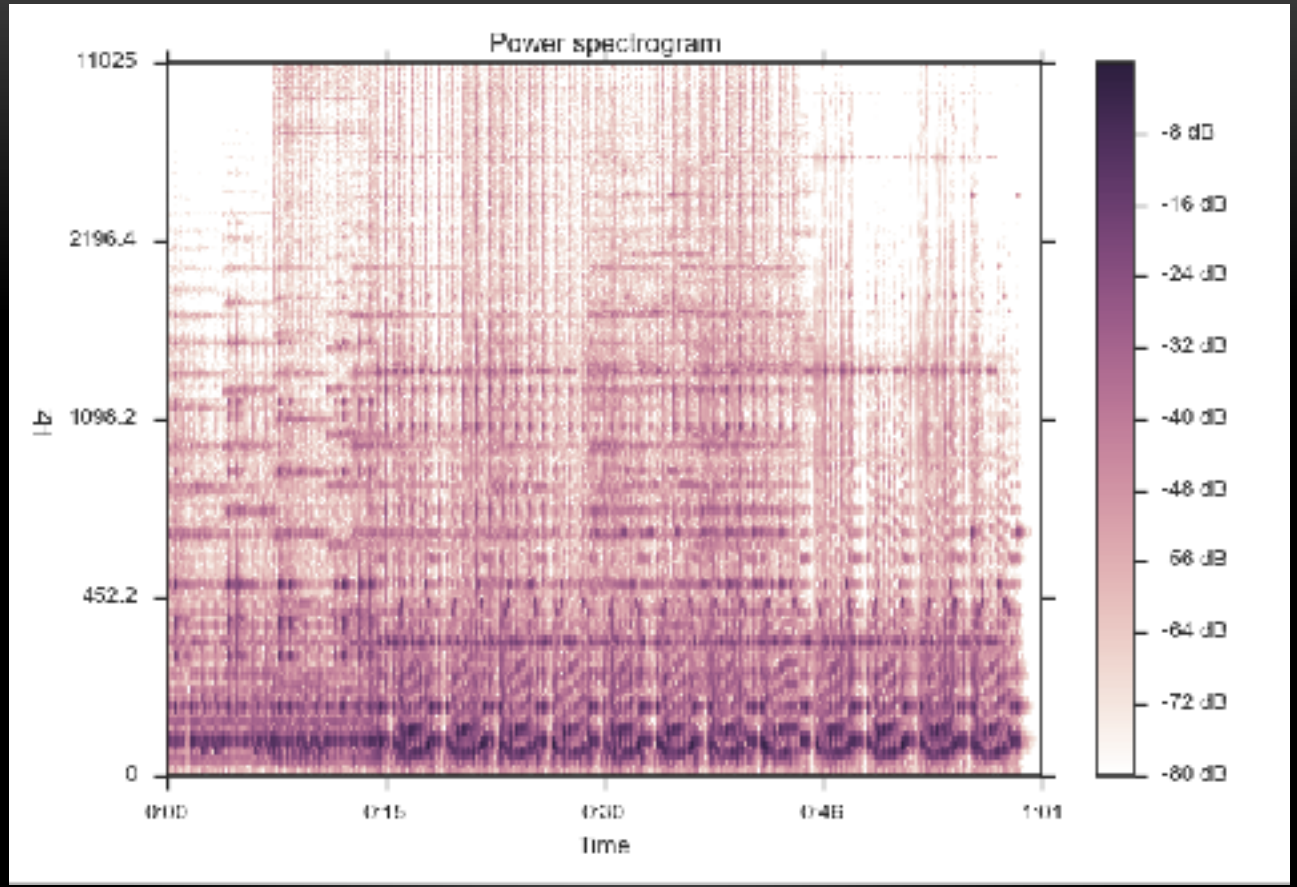
- Decompose any wave into sine frequencies
- Theory is outside scope
- Height is amplitude of that frequency

FOURIER TRANSFORMS IN PYTHON

- FT woks on continuous, infinitely long waves
- Alternative calculates discrete, short time TF
- Take small section of audio (.1 sec), calculate frequencies

LIBROSA

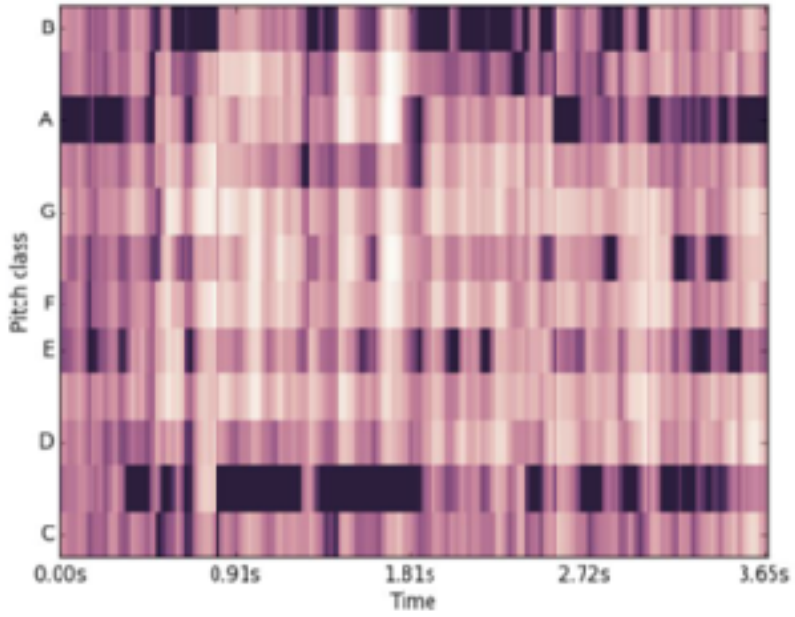
- Don’t reinvent the wheel!
- Can get frequencies in two lines of code
- `y, sr = librosa.load(“song.mp3”)`
- `D = librosa.stft(y)`


Power spectrogram
11025
2196.4
1098.2
452.2
0
f/
-8 dB
-16 dB
-24 dB
-32 dB
-40 dB
-48 dB
-56 dB
-64 dB
-72 dB
-80 dB
Time

GET MUSICAL PITCHES

- Frequencies are nice but can we do more?
- ~440 is an A, so is 880 etc.
- “Bin” frequencies at different octaves to get amount of each note
- Get 12 x num_samples array

```
y, sr = librosa.load(“song.mp3”)
S = np.abs(librosa.stft(y)**2)  # Get magnitude of stft
chroma = librosa.feature.chroma_stft(S=s, sr=sr)
```



Song 1 Chromagram

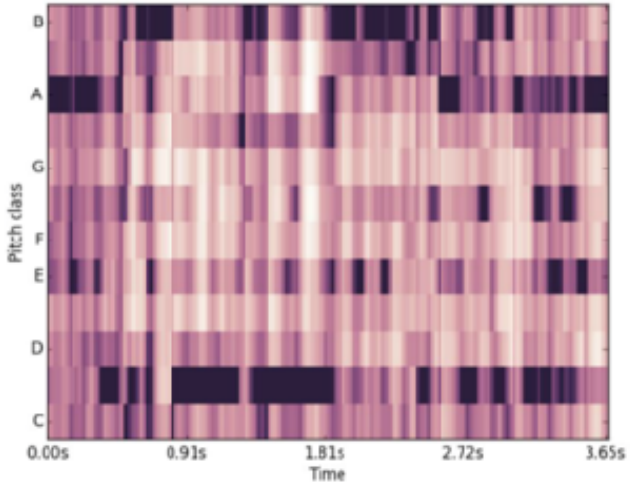
BUILDING AN AUTO-DJ

GOAL

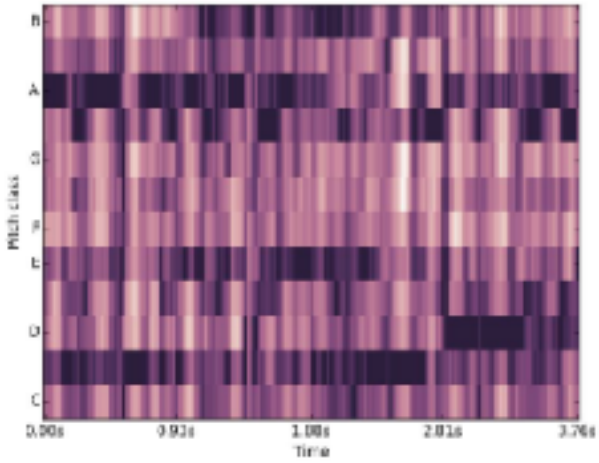
- Create mix-ins and mix-outs that sound good
- Manually select mix-in and mix-out point for many songs of equal lengths
- Try to figure out which songs mash well with each other
- Create the mix and output result

MASHABILITY BASED ON FREQUENCIES

- Want to see how well two songs sound while played over each other
- Compute chromagram for each
- See how similar they are on a frame by frame basis



Song 1 Chromagram



Song 2 Chromagram

RESULT

- Find best mixes
- Synchronize beats using librosa
- Output result using EchoNest for python
- Code at github.com/vivjay30/AutoDJ/

FINDING INTERESTING PARTS OF SONGS

Project with Google

GOAL

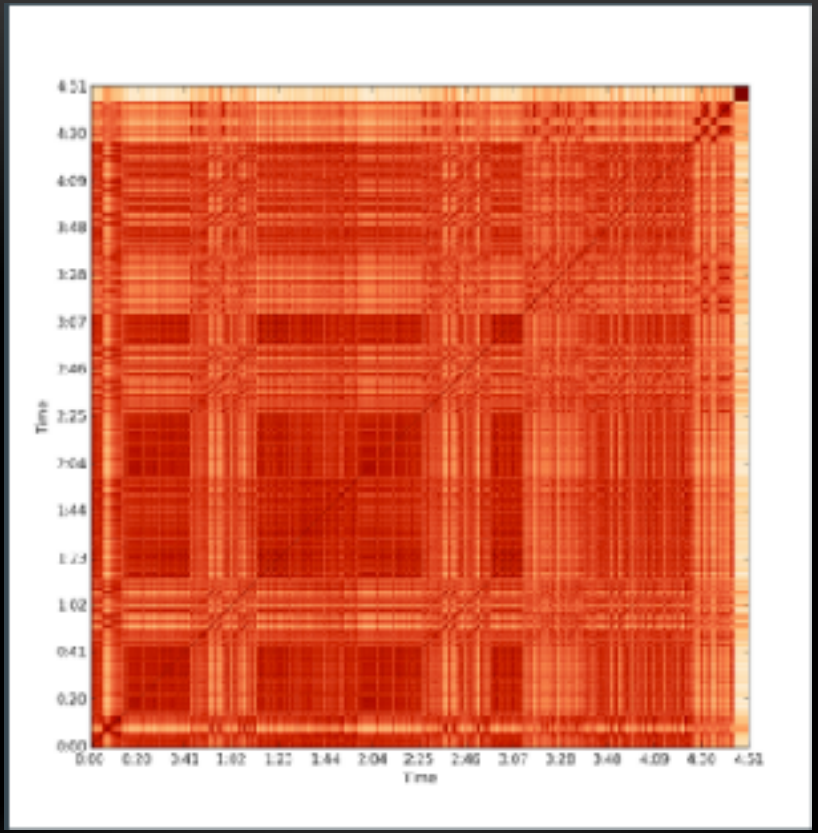
- Wanted 10 second clips for a “Guess the Song Game”
- Random selection won’t suffice
- Need those clips to be interesting/recognizable parts of the song
- Idea: Look for 10s clip that repeats itself the most number of times

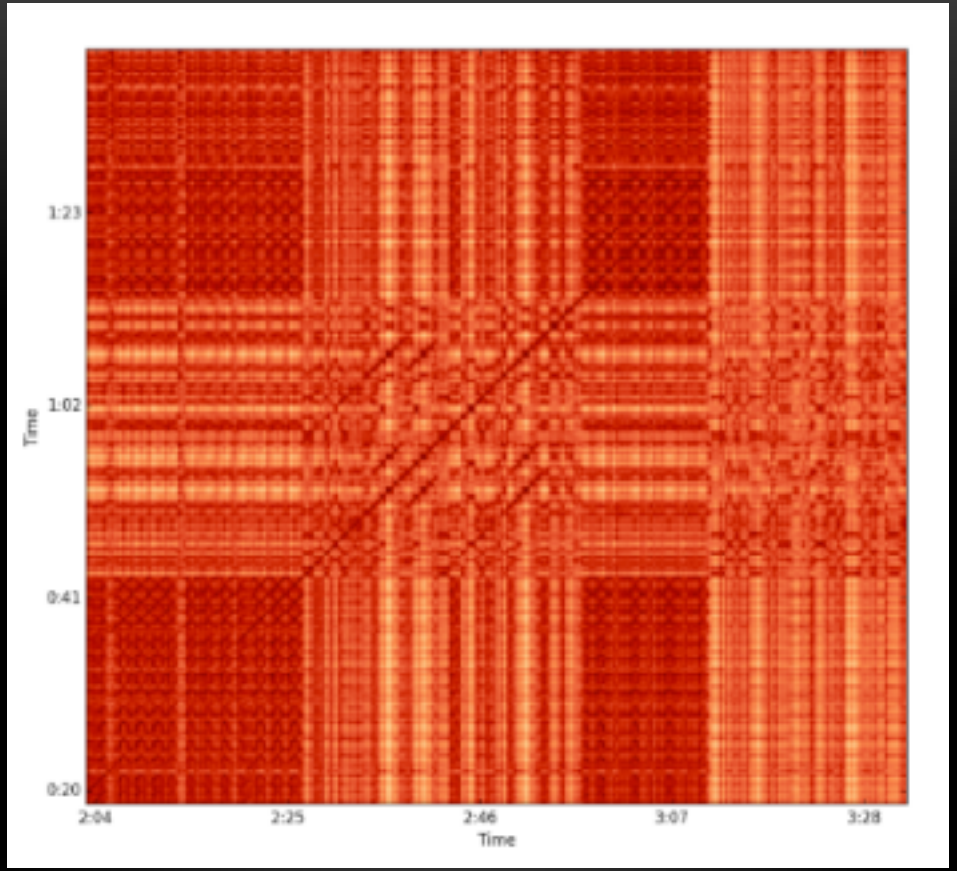
CHROMAGRAMS

- Chromagrams used and worked excellently
- Robust against changes in instrumentation
- Chorus in different octave still looks exactly the same
- Distills piece down to notes

TIME-TIME SIMILARITY MATRIX

- Chromagram is long array, compare each sample to every other sample
- Point (x,y) represents how similar time x and y are
- Example shown for “Scream and Shout”


Time
Time


1:23
1:02
0:41
0:20
Time
2:04
2:25
2:46
3:07
3:28
Time

FINDING SEGMENTS

- Repeated segments show up as diagonal lines
- Look for these diagonal lines and group together to find most repeated segment
- Unfortunately can’t play samples because code belongs to Google

KEY TAKEAWAYS

- Don’t reinvent the wheel. Libraries exist for everything
- Frequencies are important and are an accurate representation of music
- Not always important to understand theory to use application