Professor Rogers' voice x Roger Moore's voice
-> ???

Taylor Swift singing in an Indian fish market?

Lollapalooza in Rockefeller?

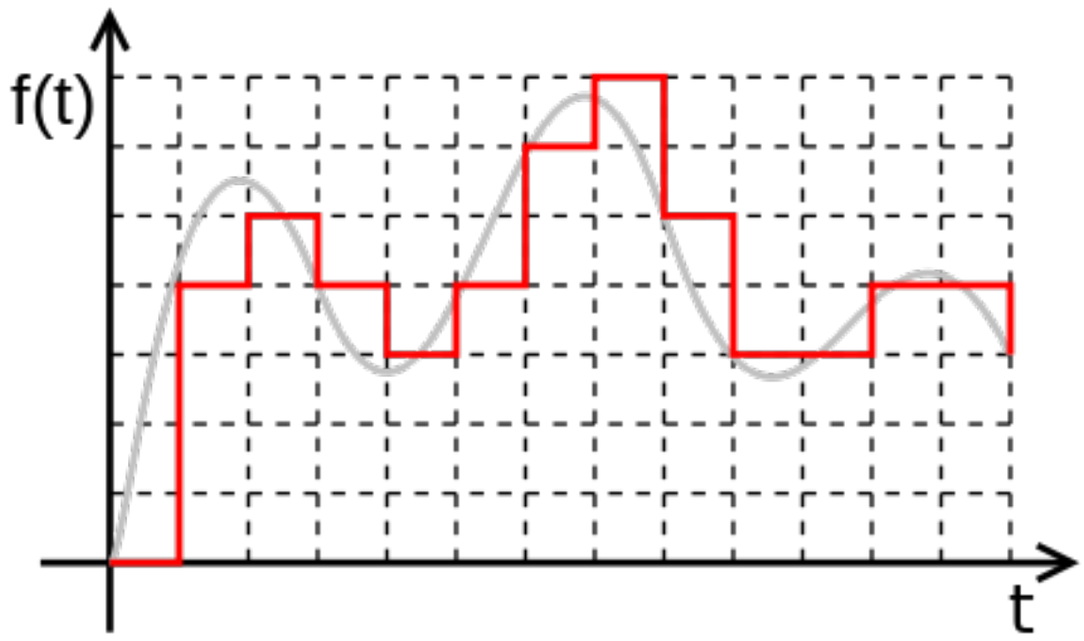
CS 122 Project?

—WIRE—
(Web Interfaced Reverb Engine)

- SQL + signal processing in Python + Django = challenging learning experience
- Original University of Chicago reverb
- Audio curious -> Audio creative! -> Sound limited only by your imagination
- Practical application of scientific concepts
- Limitless data = limitless possibilities
- Customizable and user friendly experience
- Fun, reusable product

Digital signals

- Some time-varying quantity.
- For sound waves, the quantity is air density.
- A microphone “transduces” the variations in air density into variations in voltage.
- Computers can’t store infinite data points - the continuous voltage must be “sampled” aka binned into discrete time and amplitude steps.



Fourier analysis

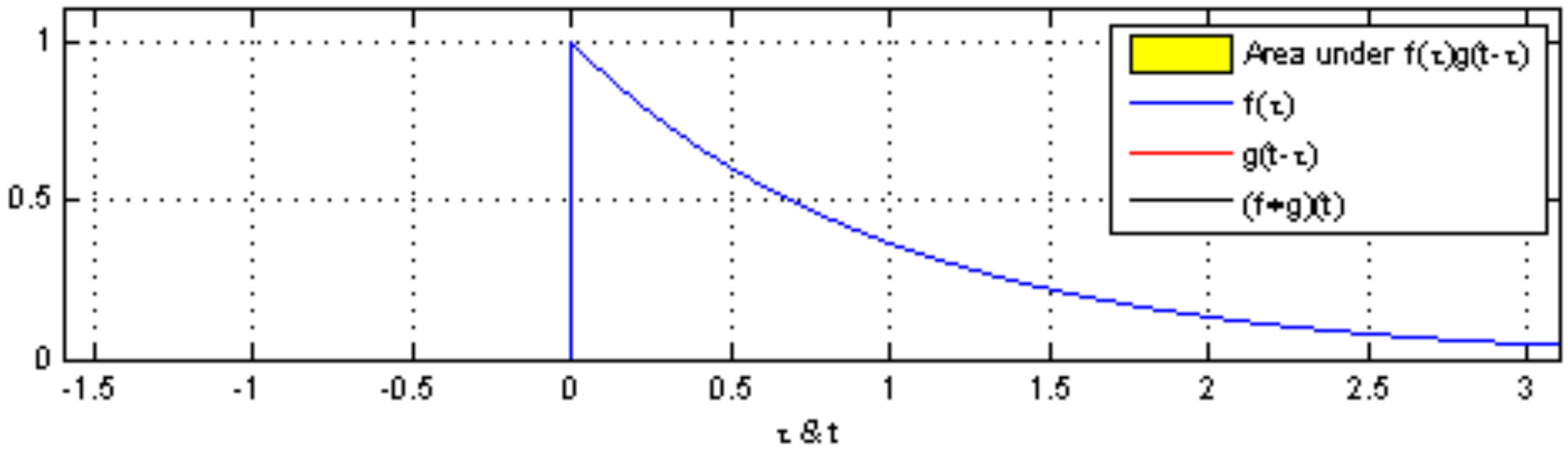
- Continuous (smooth) signals can be decomposed into an infinite series of constituent sine waves via a Fourier transform.

- Discrete-time (stepped) signals are described by a finite number of samples - need to use discrete Fourier transform, which gives a finite series of constituent sine waves.

- This series represents the relative strengths (Fourier coefficients) of the measurable frequencies in a signal.

- Convolution is equivalent to mapping one signal's frequency spectrum onto another's.

Convolution synthesis

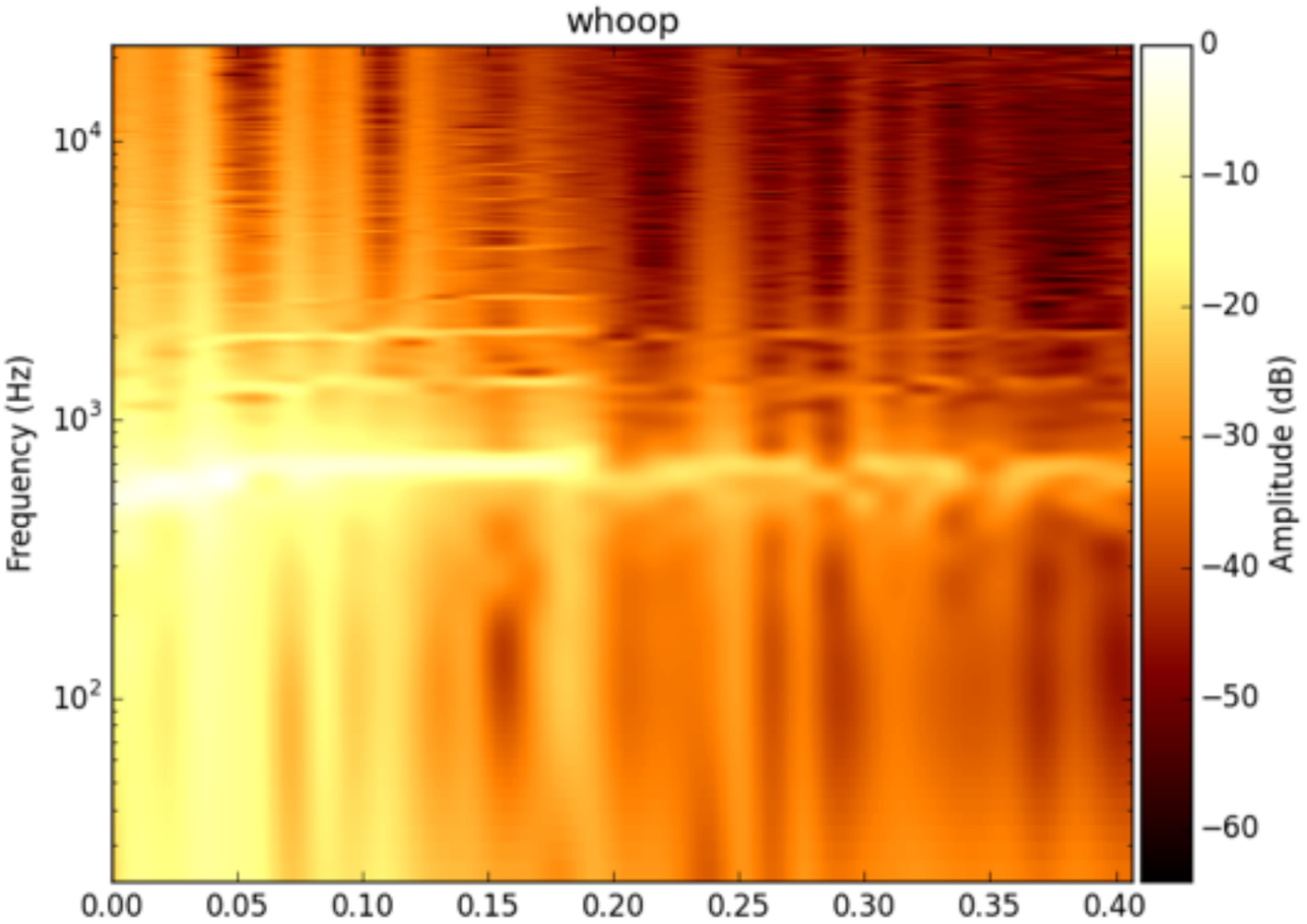


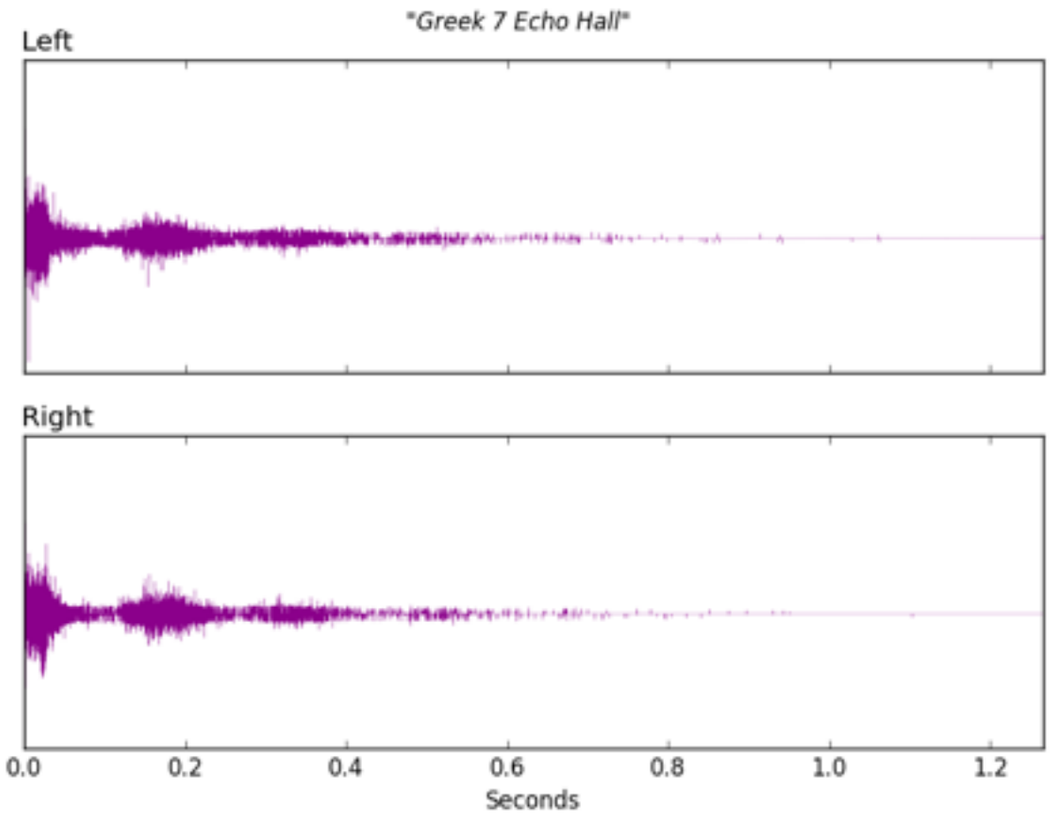
- Take two signals, flip one left-to-right, slide it over the other.

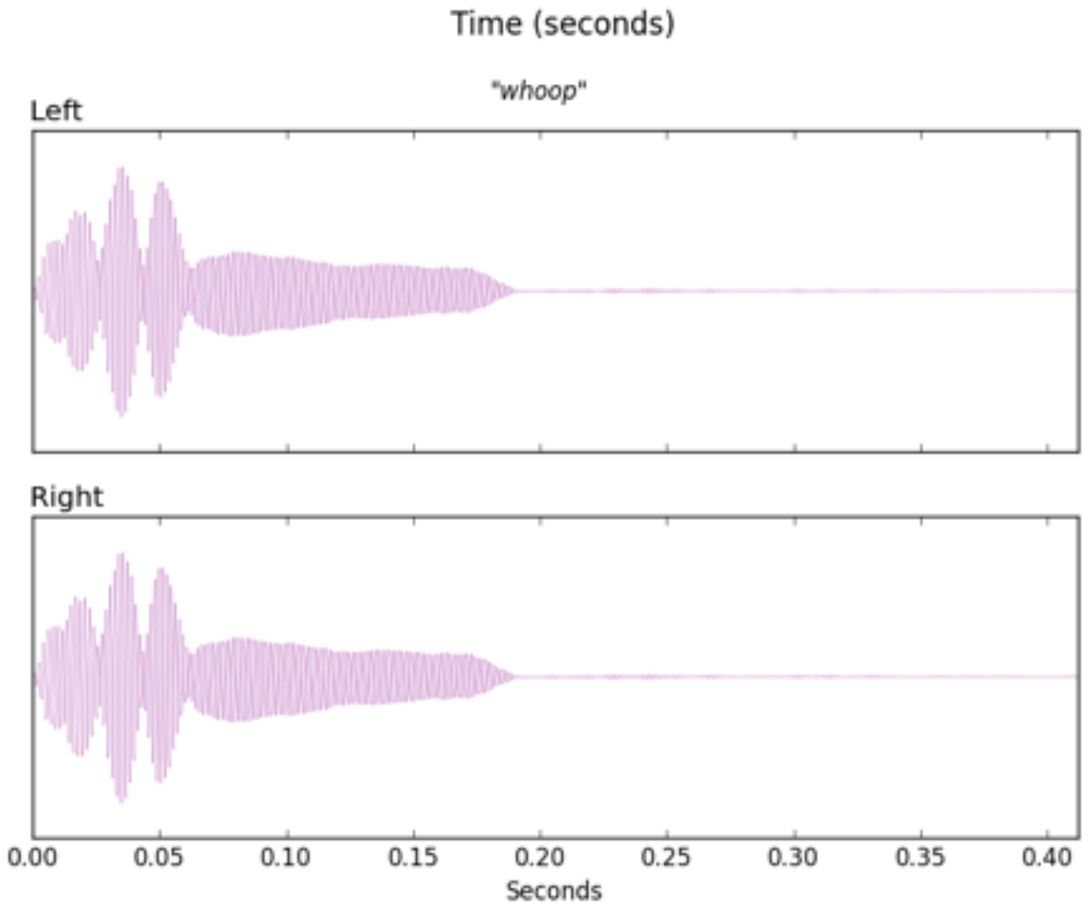
- The convolution is the area under both as you slide.

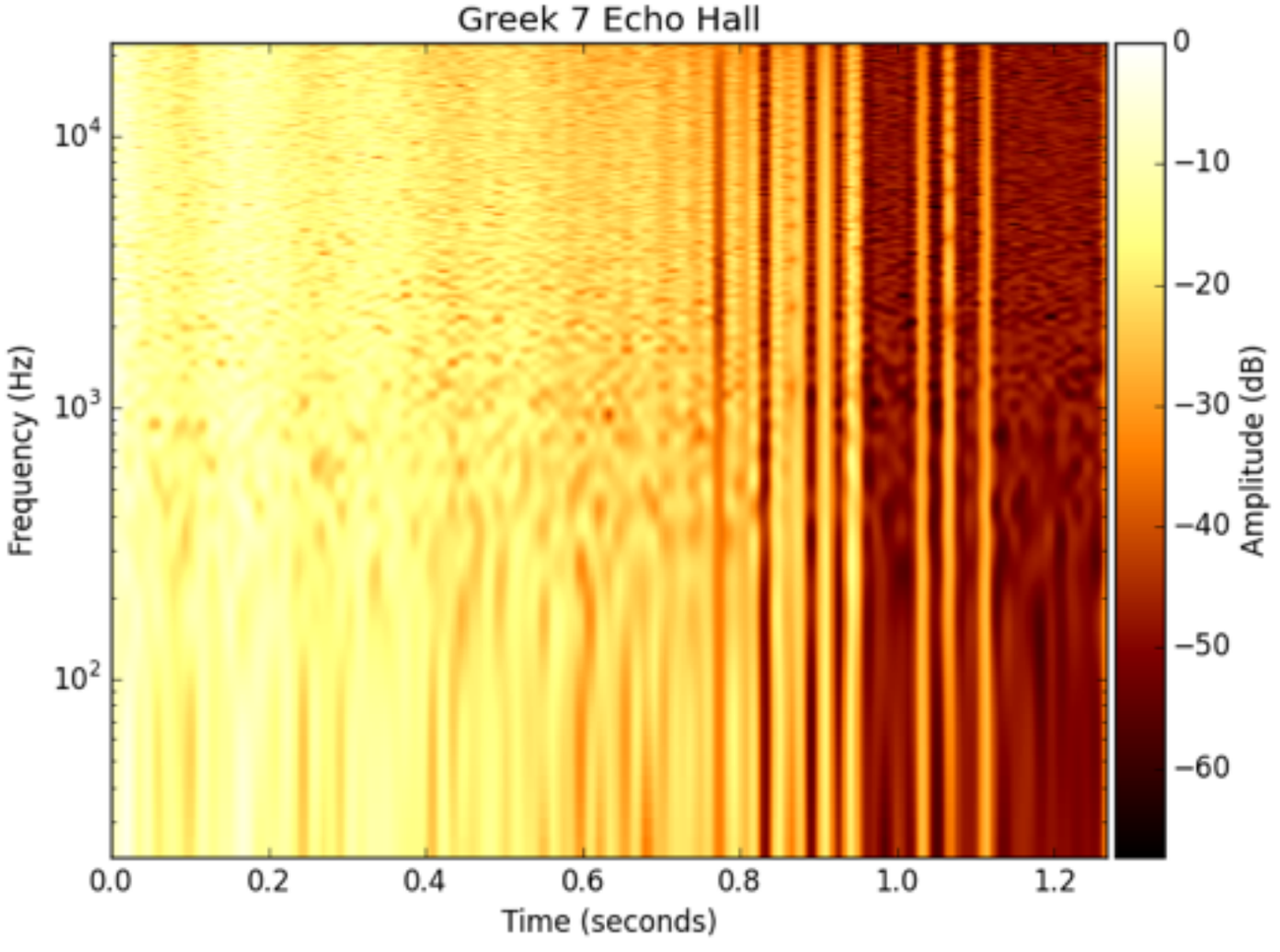
- Convolution in time domain == multiplication in frequency domain.

- We use an algorithm called the “Fast Fourier Transform” to get the Fourier coefficients for each signal, multiply them, then take the inverse FFT of the resulting spectrum to get the convolution of the two signals, $O(N \log N)$.

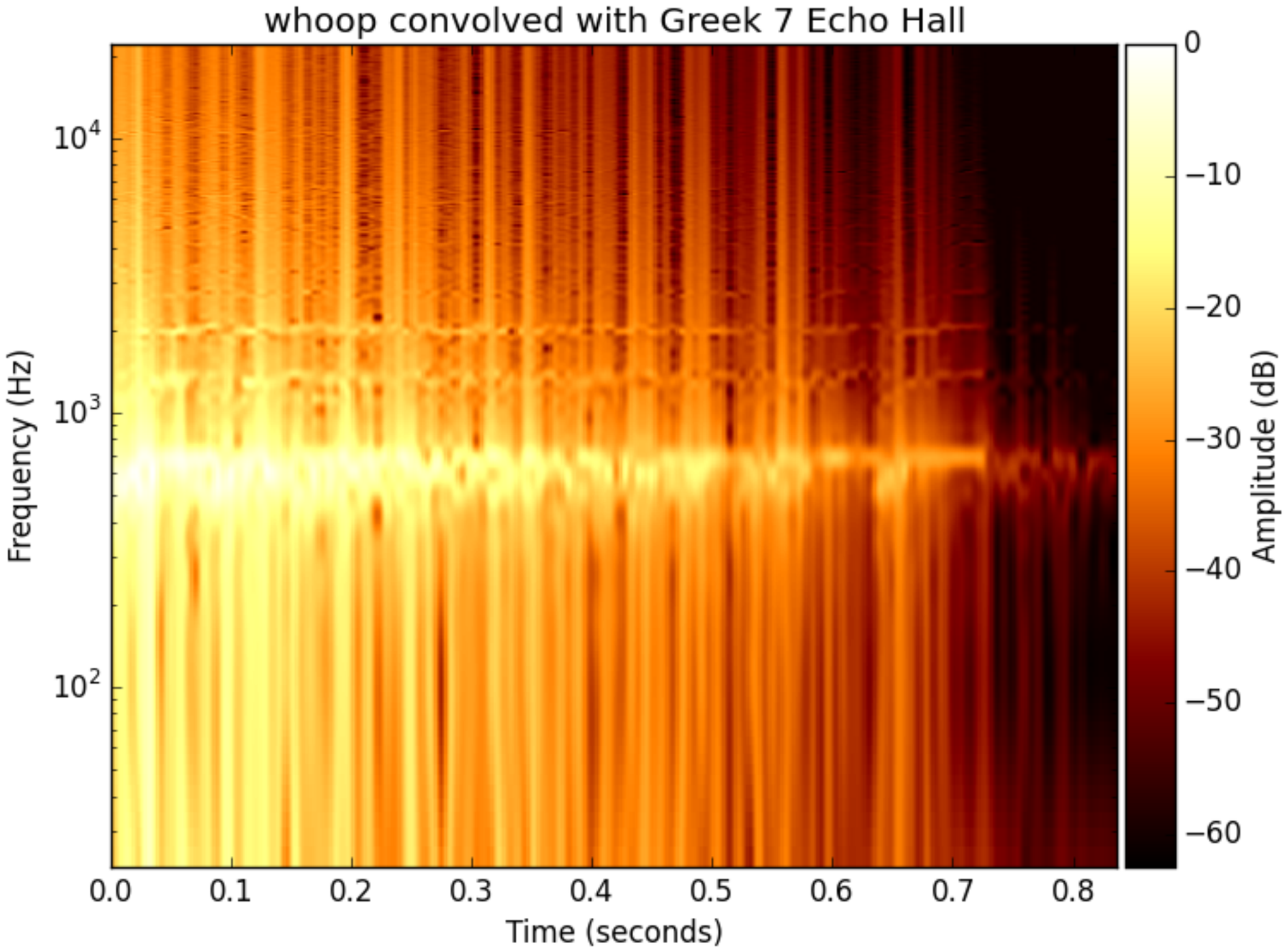

whoop
Frequency (Hz)
10^4
10^3
10^2
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0
−10
−20
−30
−40
−50
−60
Amplitude (dB)


"Greek 7 Echo Hall"
Left
Right
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Seconds


Time (seconds)
"whoop"
Left
Right
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Seconds


Greek 7 Echo Hall
Frequency (Hz)
10^4
10^3
10^2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Time (seconds)
0
−10
−20
−30
−40
−50
−60
Amplitude (dB)

Convolve a “dry” sound with a space’s acoustic resonances and you get the sound in that space.



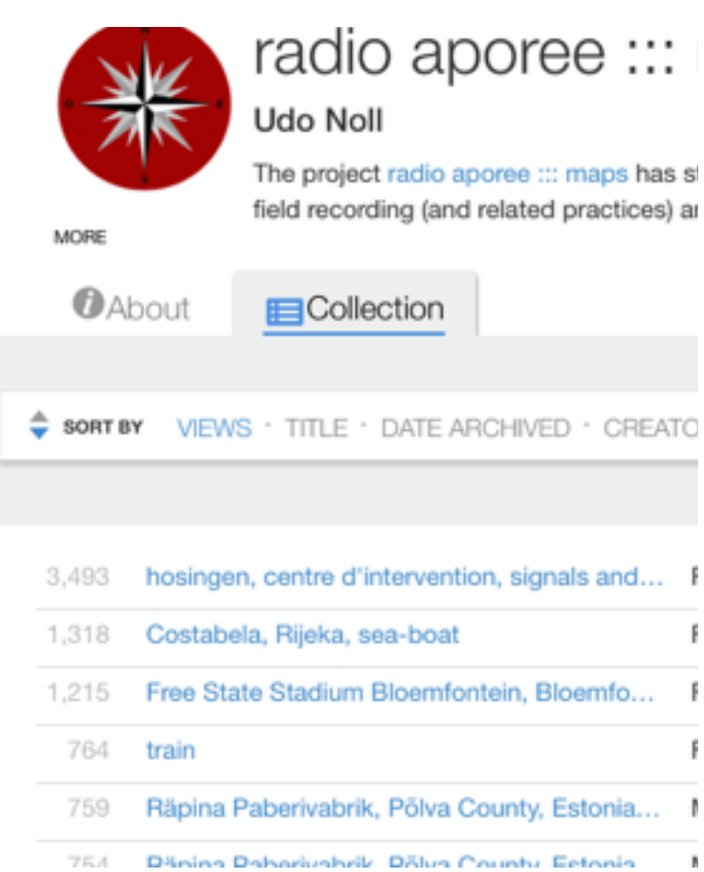
Sources of data

- Recordings of impulse responses from spaces on campus and Chicago.

Sources of data

- *radio aporee* collection of field recordings from around the globe.

A field recording playing from the old town in Ljubljana.



radio aporee collection on archive.org

New data structures

- archive.org API to scrape metadata and download audio files.

- PyDub to read audio files into 16-bit integers (CD-quality bit depth). Convert to numpy arrays of floats, and manipulate with tools from scipy.signal library.

- Django for the front end.

Work Timeline

1. Metadata scraping / Sound recording (Weeks 5–6)
2. Sound engine framework and algorithms (Weeks 4–7)
3. Django framework (Weeks 8–9)
4. Ironing out (Week 10)