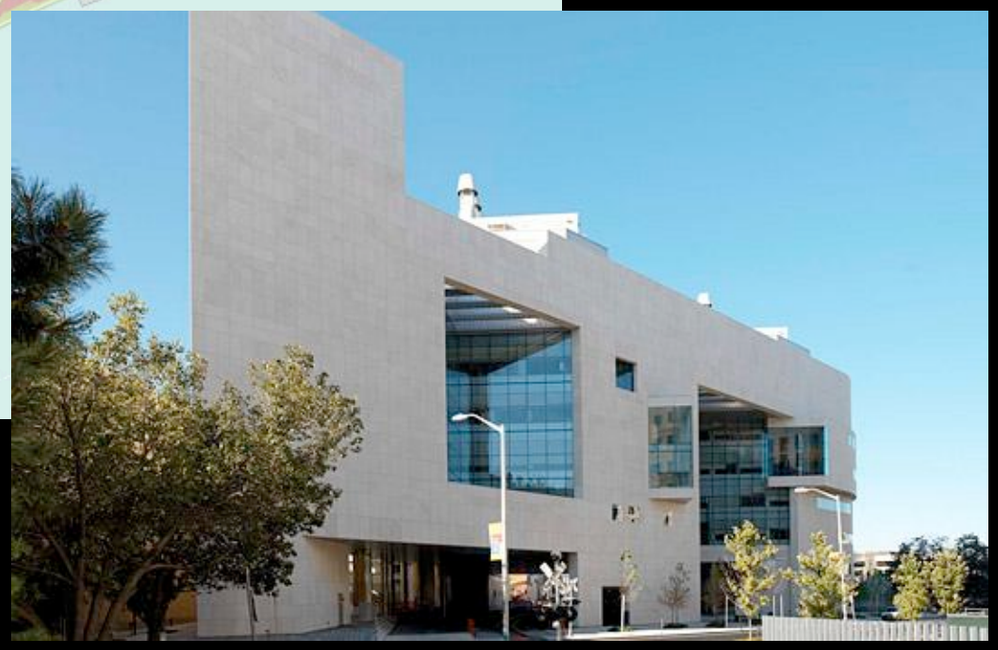
Cochlea.jl

A Julia-based real-time cochleogram visualizer for the Jupyter notebook

Alex Kell

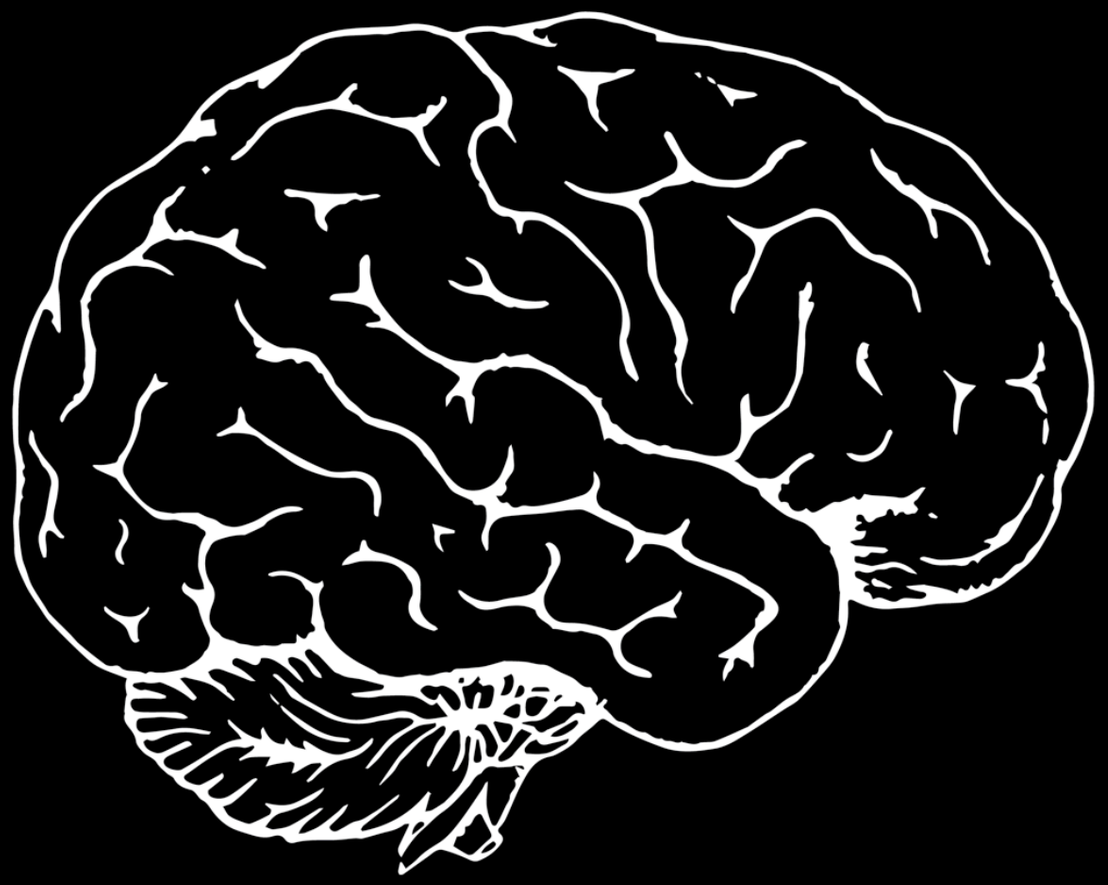
6.338/18.337: Numerical computing w/ Julia

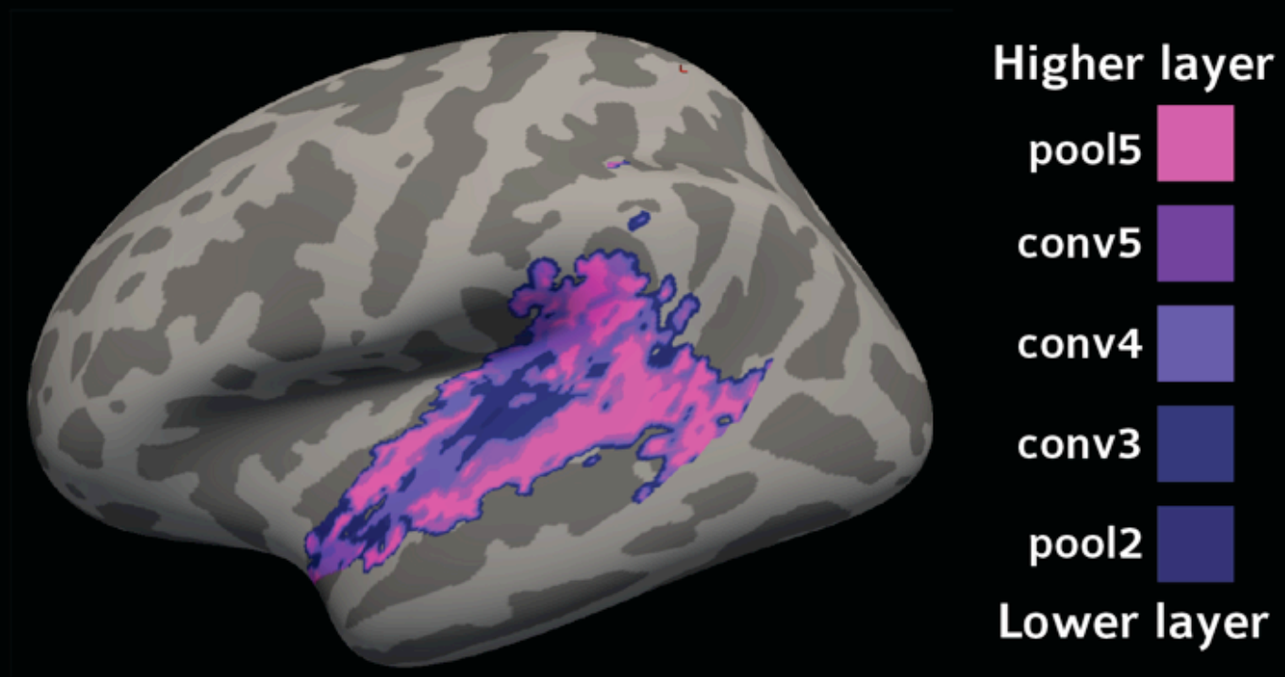
I study how the brain hears

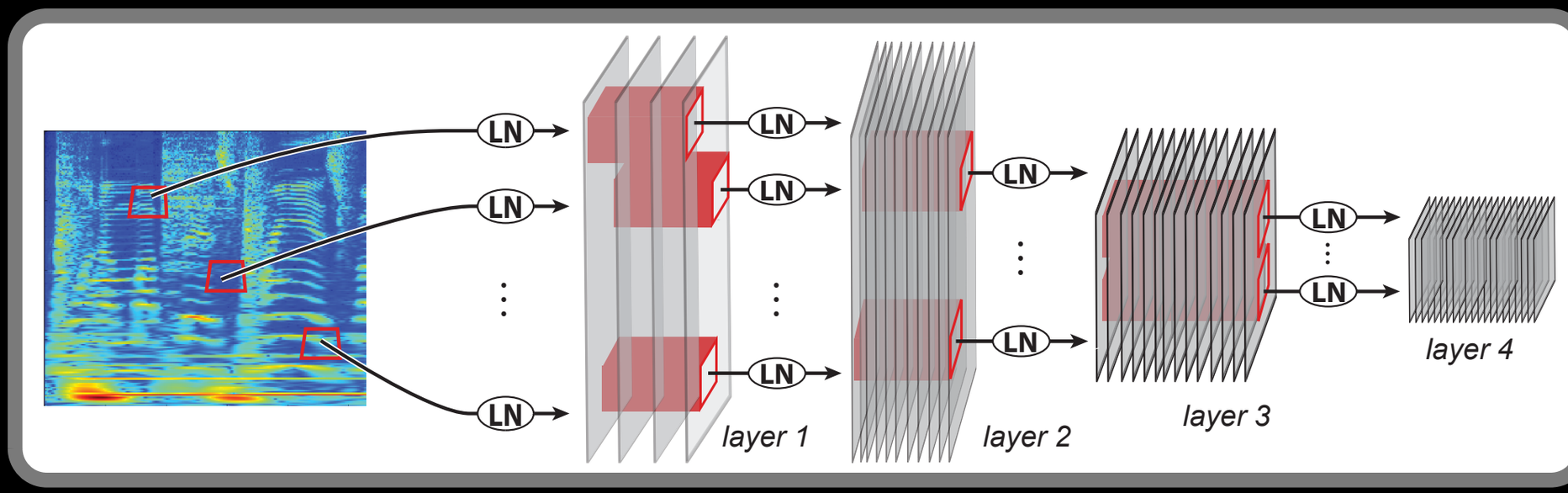


I study how the brain hears

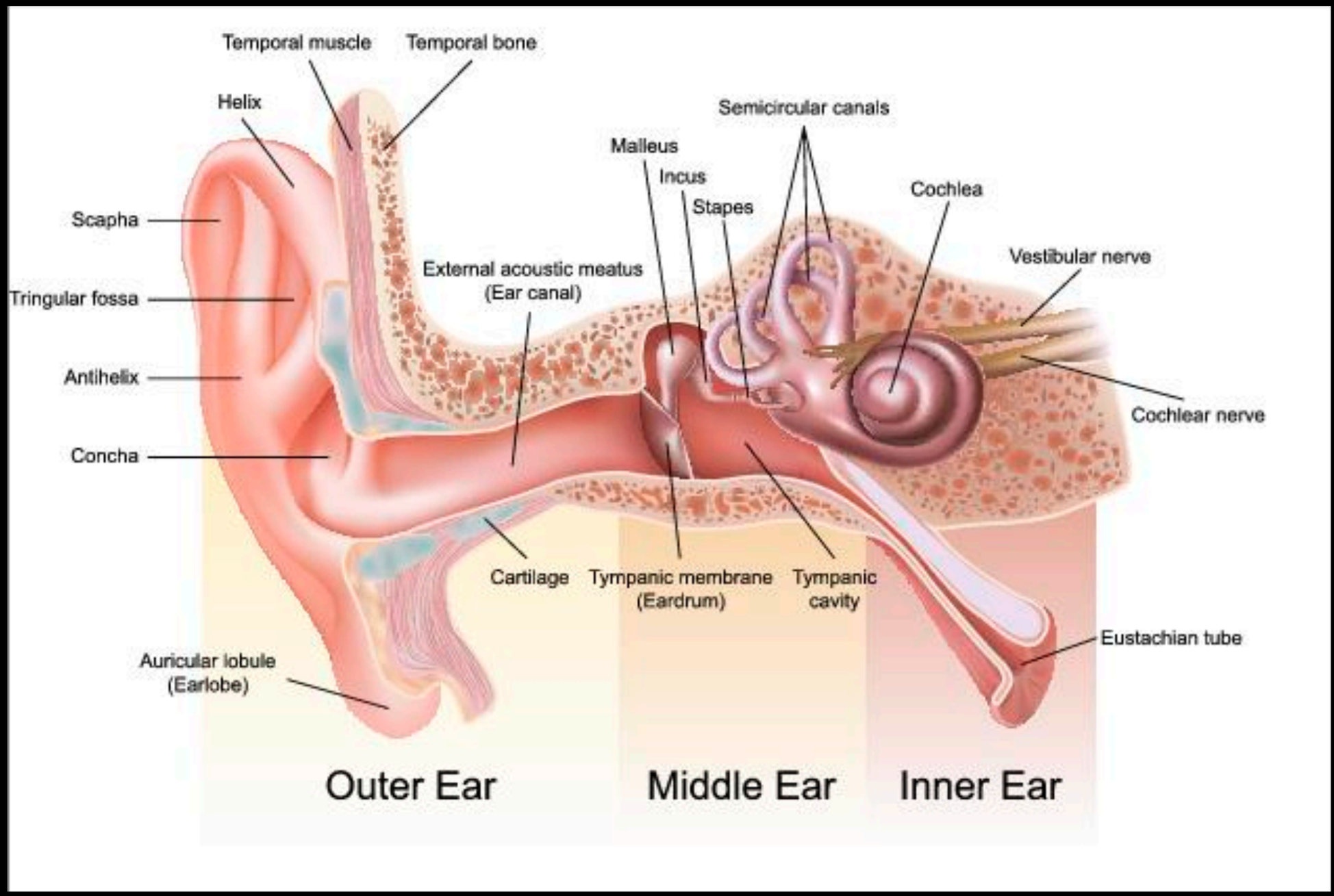
...using machine learning + neuroscience.



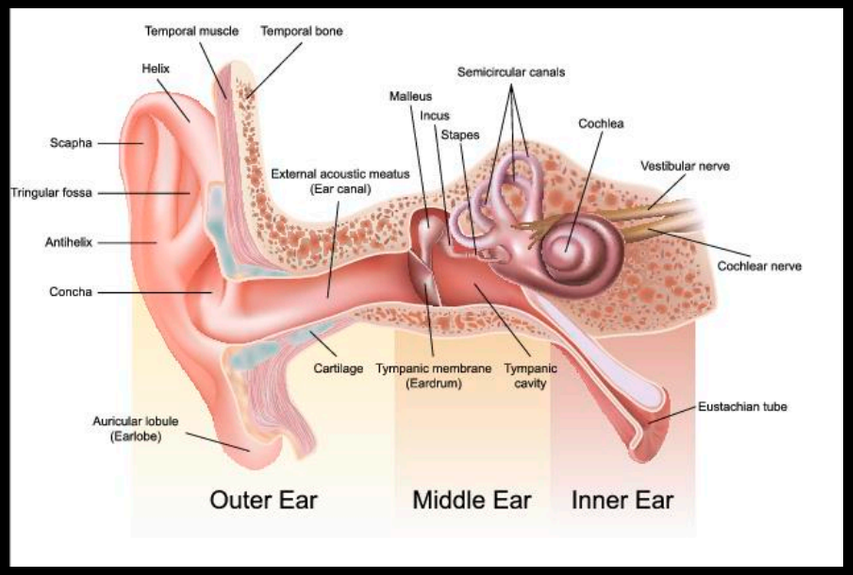


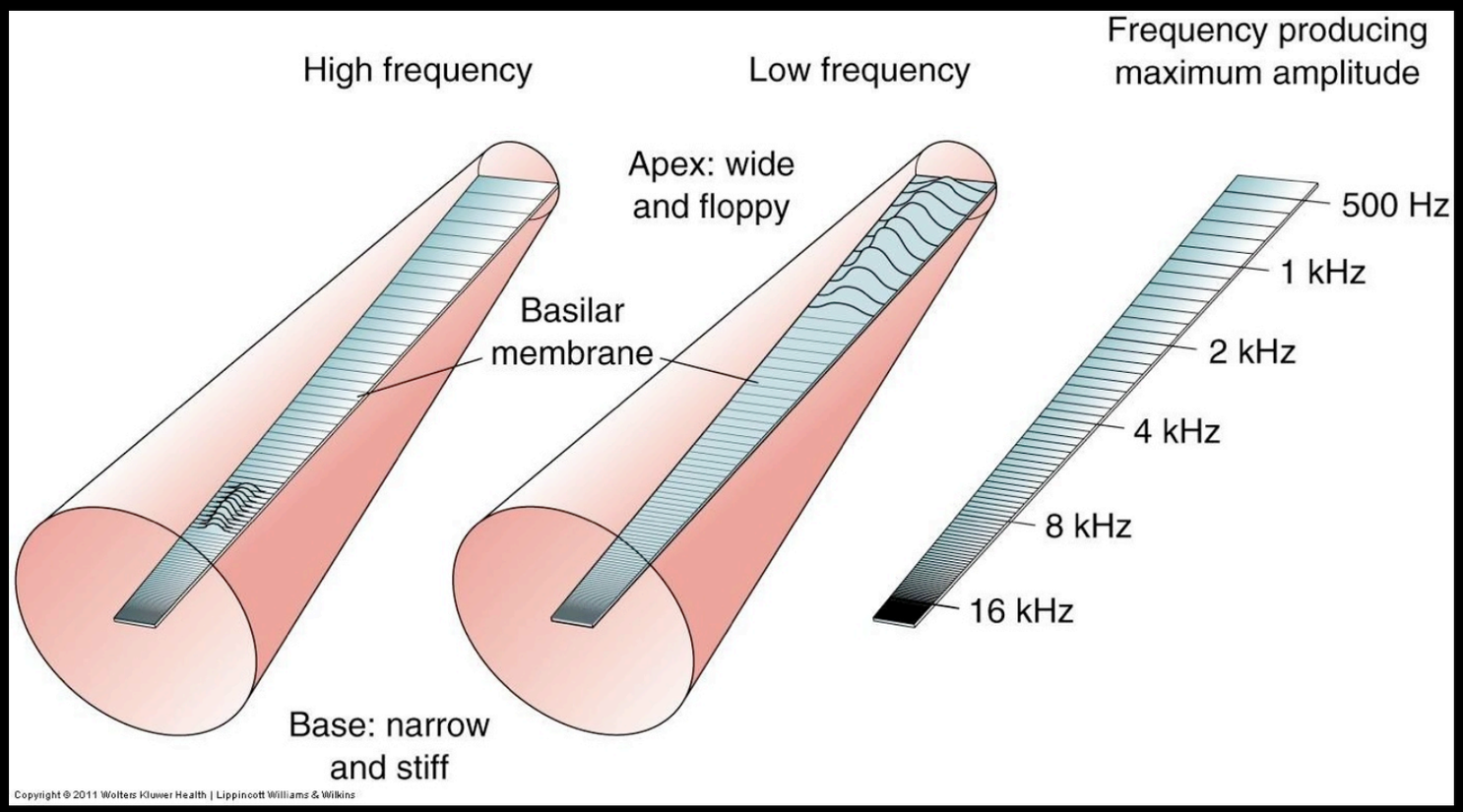


The front end of the auditory brain



The front end of the auditory brain





Physical properties of basilar membrane

↓

Different parts vibrate to different frequencies

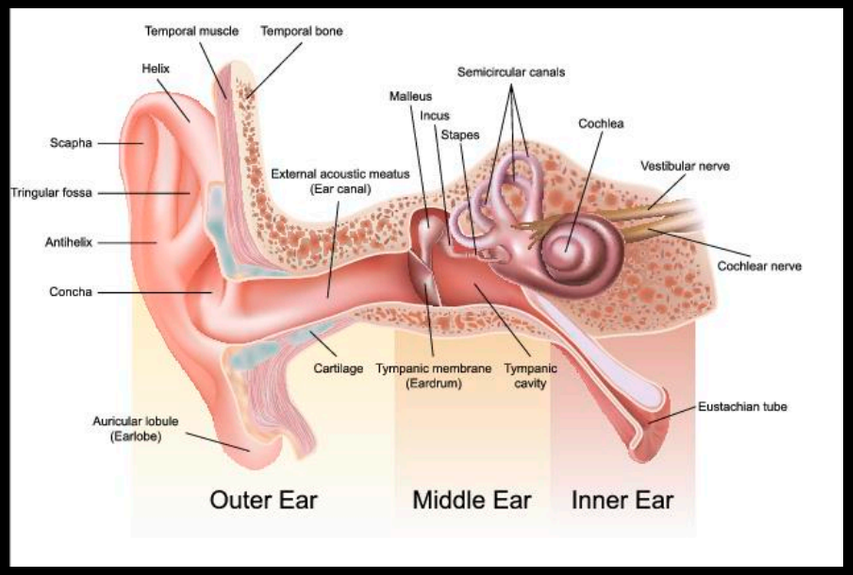
↓

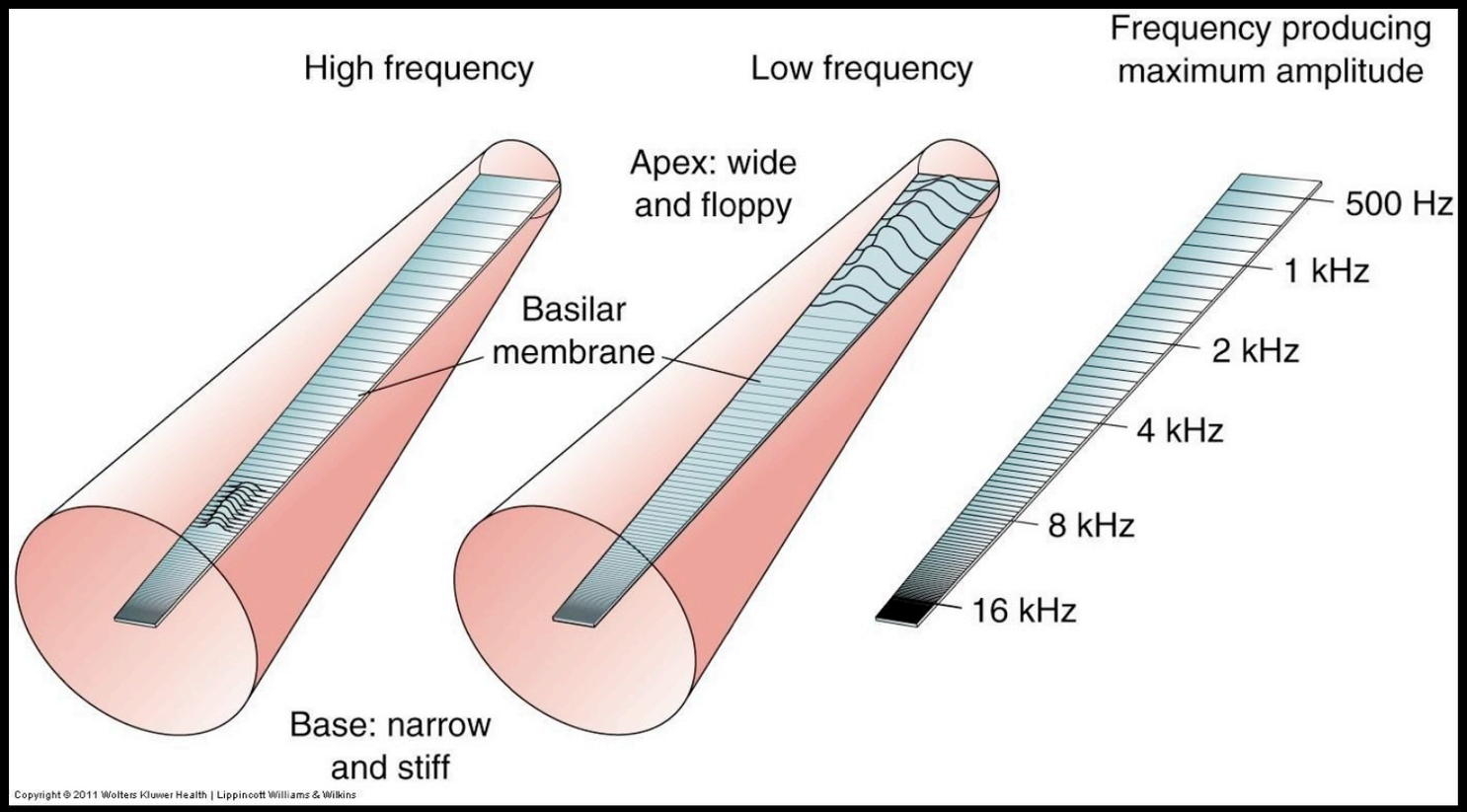
Neurons respond to local vibrations

↓

Frequency-selective neurons

The front end of the auditory brain





Physical properties of basilar membrane

↓

Different parts vibrate to different frequencies

↓

Neurons respond to local vibrations

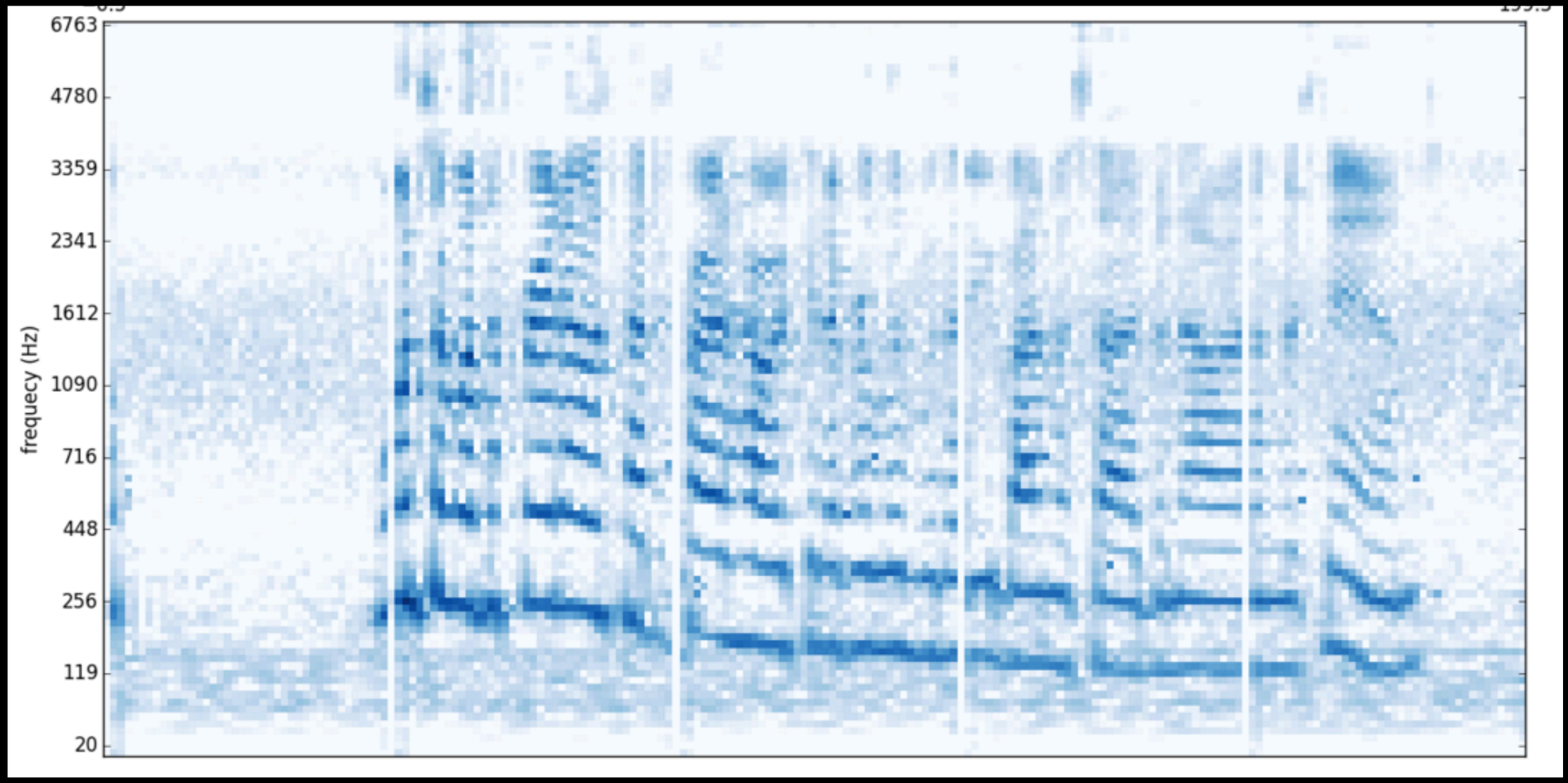
↓

Frequency-selective neurons

The cochlea performs a time-frequency decomposition of incoming sound

The cochleogram:
A coarse model of cochlea processing

Frequency



Time

Like a spectrogram or
short-term Fourier transform
BUT:

1. ~log-spaced frequency axis
2. compressive nonlinearity on amplitude

To build intuitions about sound, would be useful to visualize cochleogram of current real-world sounds.

Cochlea.jl:

-- Implemented cochleagram in Julia

-- Interfaced w/ computer's mic --> piped into Julia

-- Real-time plotting in Jupyter notebooks

DEMO

Interfacing with microphone

PortAudio is an excellent API in C and C++.

- AudioIO.jl wrapped it,
but doesn't seem to work in v0.4.

- Currently: I'm using PyAudio via PyCall.jl.

- Considered invoking PortAudio directly from Julia,
likely a good future direction.

Very basic code optimization for speed:

- Got rid of globals.

- Explicitly had types for inputs to all the functions.

- Profiling:

--> Showed that FFTs were primary cost:
- rfft/rifft instead of vanilla ffts
- less naive signal processing
(e.g., implementing Hilbert transform manually
b/c at that point was already in Fourier domain)

--> Altering dimensionality of input via microphone changes.

--> Could speed up more by dropping PyCall.jl altogether?

Thanks