Sonic and Visual Representation of Data (in python)

Ben Holtzman, benh@ldeo.columbia.edu,
http://www.ldeo.columbia.edu/~benh/courses/CMCclass_spr18/

January 30, 2020

Contents

Chapter 1

The Idea

Course Description:

> The working title of this course is "Sonic and Visual Representation of Data (using python)". Humans are currently producing enormous amounts of complex data representing complex phenomena, but are lagging in our ability to perceive and understand patterns in the data. Our auditory and visual perception systems are complementary whenever possible, but are optimized for different kinds of spatial and temporal patterns. We will develop Python programming in the context of data exploration, filtering, processing, including an introduction to machine learning. Then we will build up a wide range of approaches of generating sounds and visual animations from the same data and combine them. Questions of how to design and tune these representations to bring out patterns in the data, based on the nature of human perception, will be kept in mind throughout, with and without regard for differences between scientific and artistic intents. The first half of the semester will focus on methods. Students will select datasets they want to study early in the course, and will develop and build these projects in the second half. Some programming experience is necessary, preferably some comfort level with python. Musical interest is necessary but no background is required.

I also would like to call this book and course "Sonic and Visual Representation of *Natural* Data" to emphasize that we are just another species that happens to be a little bit more quantitative in its self-awareness than others (and quite out of control!). All the data that we record reflecting our own behavior is just as much about a natural phenomenon as those of ants or starlings (or tides or solar flares).

1.1 Philosophy and motivation for this experiment

Paul Klee's course at the Bauhaus, 1921-1922:

> *I must insist even more on the fact that the most exact scientific knowledge of nature, plants, animals, the Earth and its history, the stars, is of no use to us if we are not skilled in their representation. We can have the most complete conception of the*

combined action of these objects in the universe, but this is useless towards our aim if we are not equipped with the necessary tools. [1]

This drawing from Klee (Fig. 1.1) foreshadowed one of the essential questions of this course: How do you represent motion through some data-space using sound? He shows a 3D graph of a process that could be broken into several stages (subjectively), and each of those can be discretized in different ways, that somehow reflect the physical character or meaning of the graph. These discretizations could reflect to how you would map the information to different tones. How do you bring that static image to life?

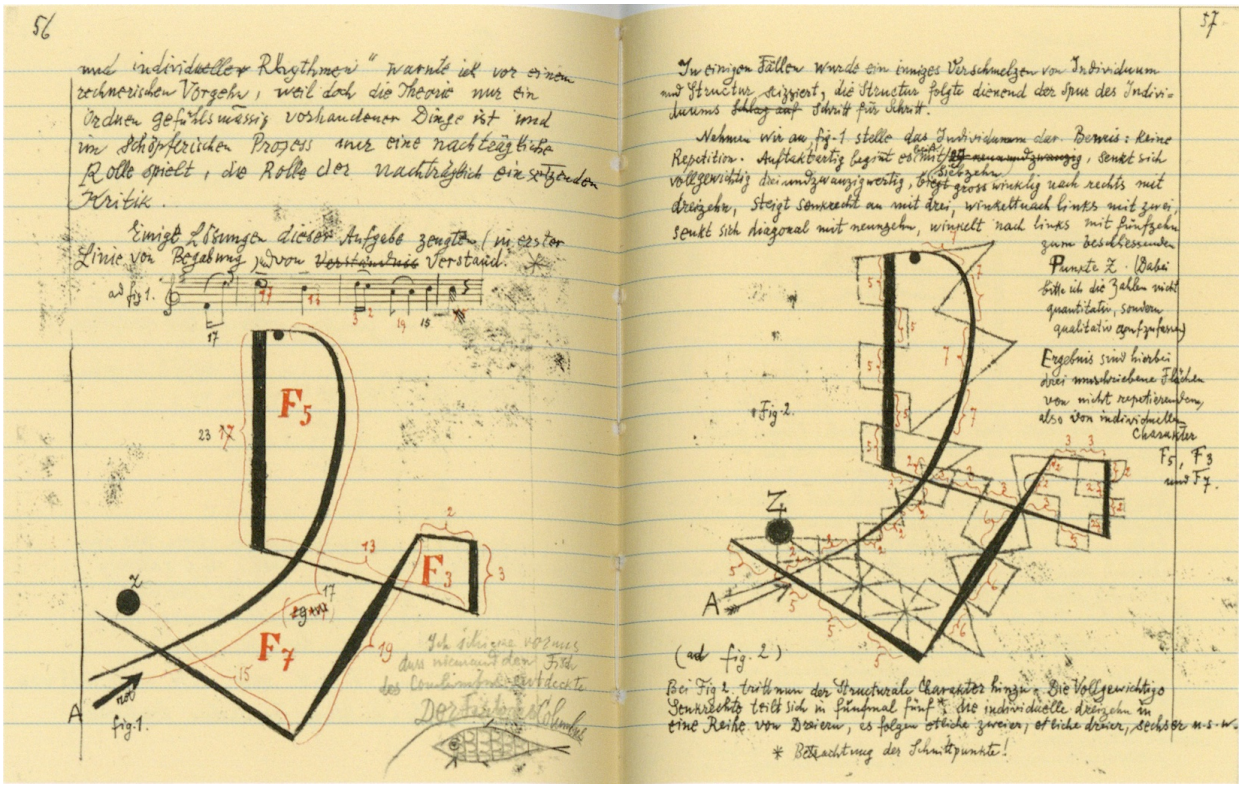

Figure 1.1: From Paul Klee's course at the Bauhaus, lessons (see text).

Frank Oppenheimer, founder of the Exploratorium:

Art and science are very different, but they both spring from cultivated perceptual sensitivity. They both rest on a base of acute pattern recognition... [A]rtists and physicists [...] use perceived patterns to create additional patterns that are not directly derived from sensory perception... Eyes and ears enable us to absorb and store the patterns of shape and time that are embodied in our experience. A higher level of perception becomes aware of patterns among these stored patterns... It is such patterns of patterns that reveal new insights. [2]

[1] Paul Klee, Cours de Bauhaus (1921-1922). Editions des Museées de Strasbourg, 2004, translated by BKH and googletranslate

[2] Oppenheimer, F. (1979). Aesthetics and the right answer. The Humanist, 39(2):18-21

John Cage:

> *We have eyes as well as ears, and it is our business while we are alive to use them.*

Questions:

- How do we perceive patterns in phenomena around us? What kinds of patterns is our visual system optimized to perceive? And our auditory system? How are they different? How do they complement each other and how do we use them together both consciously and subconsciously?
- How do we build machines and methods for extending our range of senses (i.e. microscopes, satellite telescopes, any kind of sensors?). How do we build methods for seeking, revealing and describing patterns in complex data (with or without computers!) ? And when these phenomena are far beyond our ranges of direct perception, can we take advantage of understanding our sonic and visual systems to bring these patterns into our range of senses? In doing so, what can we learn?
- Is there a meaningful and useful distinction for those first questions between the intents of science and of art? (What are the intents and aims of science and art? Are there meaningful and useful distinctions between science and art? Or should we pass on that question in this practical situation !?) Why and when does the distinction between science and art matter? And **not** matter?

1.2 Some representations of natural patterns

1.2.1 Physical phenomena that we *can* perceive

There are endless phenomena for which we have strong sonic and visual associations: animals making sounds, moving things, breaking things, colliding things, turbulence in air, water. Our perception of the size and velocity of an object through sound...

Test your perception of physics with your eyes closed: breaking glass, hammering on wood, wind in branches with no leaves, waterfall, etc.. Why do we know the sound of turbulence? Turbulence is danger!

1.2.2 Physical phenomena that we *cannot* perceive

There are far more examples of physical phenomena that we cannot perceive directly, and therefore have no sonic or visual associations, because our range of perception is very small compared to the range of length and time scales in nature ! In physics, we seek analogies and scaling laws..

- Ocean Circulation : `http://pong.tamu.edu/~kthyng/movies/txla_river_drifters/2007-05-30/movie.mp4`
  `http://kristenthyng.com/gallery/txla_vorticity.html`
  from `http://kristenthyng.com/`
  (Also see her discussions on use of color in display, in python: `https://www.youtube.com/watch?time_continue=2&v=XjHzLUnHeM0`.)

- Climate and weather data:
  `http://www.metoffice.gov.uk/hadobs/hadcrut4/`

- Hurricane tracks:
  `https://vimeo.com/294892588`

- Sea level change:
  A great example of a very broadly multi-scale system (in time and space!); wind-driven waves (seconds), tides (hours to weeks), seasonal melting/freezing (annual), sea level change (decadal +) ;
  Global scale: `https://podaac.jpl.nasa.gov/AnimationsImages/Animations`
  Global scale: `https://svs.gsfc.nasa.gov/11241`
  Local analog mappings ! (Tide organs in San Francisco)
  Local scale: `https://www.youtube.com/watch?v=Gg3EuI5vSTs`
  Local scale: `https://www.youtube.com/watch?v=n86pF-wQKrw`

- Climate change spiral:
  Some abstraction– `http://www.climate-lab-book.ac.uk/spirals/`

Earthquakes:

- Earthquakes in time and space
  As examples of earthquake catalogs (lists of when and where earthquakes occurred), compare two representations of similar data sets, both intended to convey the dramatic rise of seismicity rate in Oklahoma due to injection of wastewater from fracking, into rock formations that used to be oil reservoirs. Oklahoma (BKH and Douglas Repetto):
  `https://vimeo.com/186029718`

- Seismic Wavefields
  ("Ground Motion Visualizations") (by Martin Pratt):
  `https://vimeo.com/187740441`
  Simulations, with sonified seismic data `https://vimeo.com/153300296`

- Free Oscillations of the Earth:
  `https://vimeo.com/245658903`

- The Lone Star Geyser:
  Three different kinds of data, each generating very different sounds, to convey coupled, causal processes: `https://vimeo.com/244587127`

- Higgs Boson:
  Lily Asquith: Sonification as a scientific tool, by someone who hates having to give a TED talk ! `https://www.youtube.com/watch?v=iQiPytKHEwY`

- Protein Structure:
  What do we learn from this one ? `https://www.nytimes.com/2016/10/22/science/proteins-music.html?hpw&rref=science&action=click&pgtype=Homepage&module=well-region&region=bottom-well&WT.nav=bottom-well&_r=0`

1.2.3 *Human* phenomena that we *cannot* perceive

And human activity (not "nature" vs "human": we are natural organisms, behave just like other creatures!) : Traffic (pedestrian, vehicular), voting patterns...

- Two trains:
  `https://datadrivendj.com/tracks/subway`
  One parameter (average wealth per area) mapped to emotional character of the music, embodied in multiple musical aspects (rhythm/tempo, key, chord depth, complexity).

- Beijing Air Quality:
  `https://datadrivendj.com/tracks/smog`

- Nuclear Bomb testing:
  `https://www.youtube.com/watch?v=I9lquok4Pdk`
  A classic! Minimal emotional content mapped to the sound, but lots projected by the listener! Like the Oklahoma earthquake movie. Is this more "effective" (depending on what your aim is) ?

1.3 Data Art and Data Artists

In the domain of art, how are people approaching these questions? What are they doing? What are differences in their aims and intents, and in their methods? (The grant that funds my teaching of this class also supports the "Artists Using Data" lecture series organized at the CMC.)
2018:

- Allison Parrish `https://www.decontextualize.com/` .
- Luke DuBois `http://lukedubois.com/`.
- Gene Kogan `http://genekogan.com`. (Machine learning for artists: `https://ml4a.github.io/`)
- James Hoff `http://www.callicoonfinearts.com/artists/james-hoff/`.

2019:

- Brian House `https://brianhouse.net/`.
- The wind people
- The cell phone number on a card to whistlers...

1.3.1 Algorithmic composition

So many threads in this history of music, that I know little about, but could easily start with Bach– The minimalists (Terry Riley, Steve Reich, John Adams)... simple structures that can be described geometrically and modified by systematic operations (substitutions, transpositions, transformations, etc). Also, Laurie Spiegel ! See below in discussion on patterns:
`http://retiary.org/ls/` and
`http://retiary.org/ls/writings/musical_manip.html`
Dan Tepfer ?

1.3.2 Data-driven composition

Charles Dodge, solar wind.. etc...

1.4 The nature of patterns

What do we mean by patterns anyway? Spatial, temporal ?

Worlds of ways of describing patterns.. interesting patterns have certain properties in between predictable and random..

1.4.1 Randomness

Here is an example of one of the codes we will give you, where each dot moves randomly through space and time, and the sound is driven by the position. `https://vimeo.com/210714480` Most processes are not truly random– they are bounded, and the next event depends somehow on the previous one, and there are interactions among particles. Sometimes it is good to compare something that seems random to something that is truly random, for hints at buried patterns.

1.4.2 Deterministic patterns

Newtonian mechanics: $F = ma$, momentum conservation.. pool balls. More on this in the Primer on Sound (Chapter **??**).

1.4.3 Chaotic patterns

1.4.4 Information and entropy

Expectation, in music... etc..

From Laurie Spiegel[3]

> If man has evolved music as a communications medium in a noisy world, it must embody reflections of other structures, including cognitive processes and natural sounds, and have built deeply into it's nature, among the many things that make it work, compensations for natural phenomena which would otherwise diminish its effectiveness. These must also be phenomena that follow natural laws, in this case Shannon's[2,3].
>
> We should also consider the nature of the noise affecting our signal. Not all noise is created equal nor does it exist in a vacuum as a Platonic Gaussian ideal. Much of the "noise" that affects our ability to communicate is generated by something that makes sense in some other context. I consider randomness a relativistic phenomenon: any signal, no matter how internally consistent or meaningful it is within its own context, may be perceived as random noise relative to some other coherent signal.

(Put in equations of information entropy! refer to Musimathics Markov process sections.)

[3]from Laurie Spiegel,"An Information Theory Based Compositional Model" *Leonardo Music Journal*, Volume 7, January, 1998, MIT Press`http://retiary.org/ls/writings/info_theory_music.html`

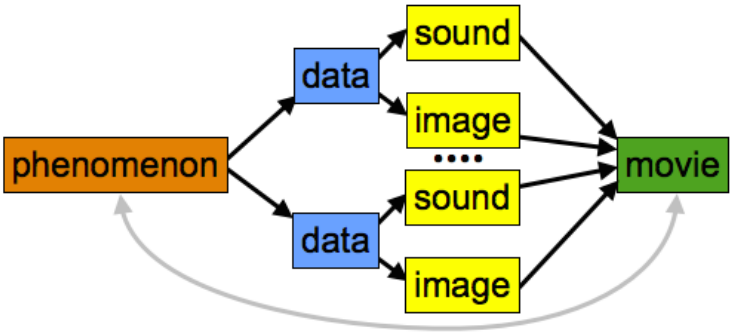

Figure 1.2: To what extent does the movie represent the phenomenon?

1.5 The Pure Phenomenon and the Measurement

What is the pure thing, that is unperceived, that is happening regardless of whether or not we perceive and measure it? How do we measure it? What do we want to measure that best captures the nature of the thing and its motions in time and space?

1.5.1 The Breathalyzer

Breathalyzer demo:
`https://vimeo.com/200771098`

Why is this NOT what we want to do!?

1.6 Thinking about your data

1.6.1 The Flowchart

The flowchart that we will frequently revisit, to ask: "what is the relationship between the movie and the actual thing?"

What is the meaning of the data that you want to bring to life? What patterns are in there but difficult to visualize or perceive in "normal boring science plots" ? What are the patterns that can be illuminated with sound? Especially via *comparison* of two or more that differ in one aspect...

(A cynical and grumpy criticism of sonification of data, but makes a few useful points that we will make over and over:
`https://www.youtube.com/watch?v=Ocq3NeudsVk&feature=youtu.be`)

- Do not overstate or misrepresent your case, the meaning of the sounds, or the value of sonification ! It is not new anymore!
- Be honest and transparent about your mapping, your process, your aesthetic choices: make a key or legend or "explainer movie" that lays out the rules for the representation, the mapping of data to visual and sonic elements. Only then there will be potential as a pattern-seeking tool !

- Make the key and the data movie iteratively, in parallel, so that they are constantly being checked against each other.
- Wherever possible, design a comparative representation– one example against another that helps illustrate the patterns of interest by their presence and absence. This approach does not need to be limited to scientific intents, though is perhaps more important to be explicit in that comparison.

"concept explainer" and "movie explainer"

1.6.2 Storytelling: the use of comparison to illuminate meaning in data

California earthquakes vs Oklahoma earthquakes
California earthquakes are "tectonic" earthquakes– the stress that drives them is always there in the plate boundary... the randomness conveys a constant driving force. In contrast, before the fluid injection began, there was no significant driving force for earthquakes in Oklahoma. `http://www.seismicsoundlab.org/?page_id=336` The sound is what conveys this difference in a visceral way; if these movies were silent, they would be far less effective.

1.6.3 Projects !!

This is a PROJECT CLASS. I encourage people to form groups formed by common INTEREST, not by skills ! (if someone in the arts wants to work with someone in science, fine! but let it emerge by interest in that problem and question, not by programming skills or lack thereof.

I also encourage collaborative coding, within and across groups. This is not a competition ! Everyone's experience is different! If you have no coding experience, take one piece of the problem and pick it apart 'til you get it. If different groups need similar methods, share them!

All programming is trial and error and error and error and error and then a small success. Repeat.

Chapter 2

A micro-primer on the physics and perception of sound

2.1 Overview

Before talking about perception (next chapter), its worth considering what sound is. Sound waves travel through the air and are picked up by our ears. Waves propagate from a source. That source can be an oscillation (vibrating vocal chords, wings flapping, a speaker cone), or it can be an impulse (a clap, an earthquake, a pluck of a string), or any combination. But no matter what the nature of the source was, the information propagates through a medium (air, usually) and is perceived by us through our ears (more on them later). However, the sources of data that we represent with sound may be from many different kinds of processes, some with close analogies to sound in air, and some that are completely non-physical. So it may help to have a sense for what those physics are in a range of materials (including the Earth) to think about physical analogies.

- What do we mean by "information"? Another word for information in this context is "energy"! Wind is the movement or flow of air masses; the motion is "irreversible". Sound waves are NOT wind; they are just the movement of pressure waves, which are the carrier of information in this case.
- Why/how does this information propagate as waves? Because of "elasticity", the capacity of a material to store mechanical energy and transfer it as a wave or return it to its surroundings or another object.
- What is elasticity at the atomic or molecular scale ? What properties control the speed of a wave propagating through a material? And how do those waves propagate ?

We will move through these questions for gases, liquids and solids.

2.1.1 What are sound waves?

These waves travel by elastic collisions between air molecules. To a very good approximation, as far as our perception goes, the speed of a wave does NOT depend on frequency, so we can use this

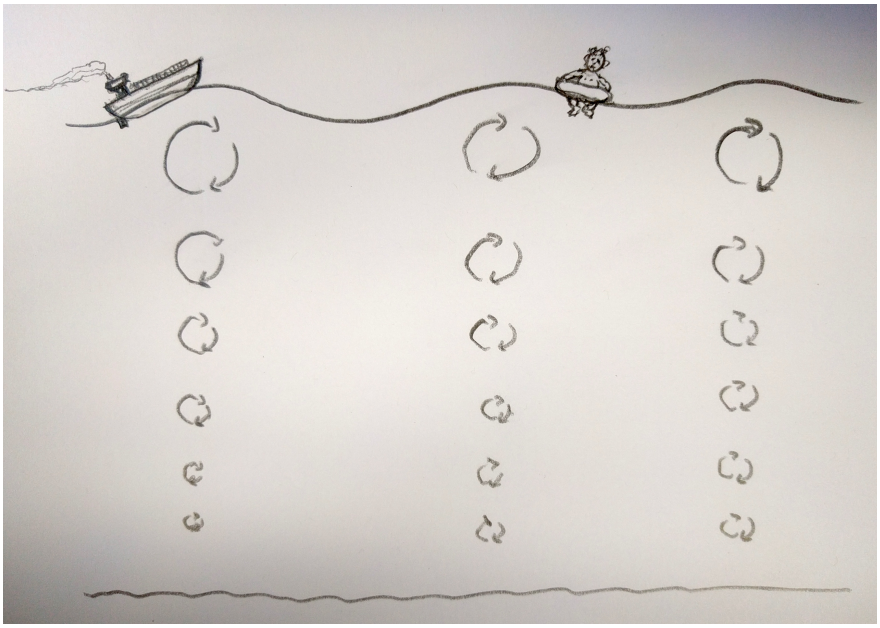

Figure 2.1: Ocean waves (at all scales!): no net motion of mass, only of "information", in this case, kinetic energy of water molecules... And these are "surface waves".

equation for the speed of sound $c = \lambda f$, where λ is the wavelength and f is the frequency of a sound wave.

What is the source of a sound? (In seismology, the earthquake is called the "source"). We have a rich vocabulary for sounds, that all describe the source: scratching, banging, hissing, thudding, pounding... The word invokes the physics of the process.

Think about the process you are trying to represent with sound, the process that is not actually producing a sound, or even waves of any kind. Is there a physical, sonic or musical analogy you can find, that gives the meaning you want? How far do we want to take that approach ? Might this approach be a good way of designing synthetic sounds?

2.2 A very simple description of matter

First, what is the difference between a solid, liquid and gas, using the concept of interatomic potential ?

- That previous model assumes that molecules are little elastic balls, but real matter is different electron clouds `http://en.wikipedia.org/wiki/Atomic_orbital`
- types of bonds: van der wals (weak, distant bonds); covalent (stronger); ionic (weaker) representation as potential wells.
- statistical ensemble link– nice movie of ensemble in potential well `http://en.wikipedia.org/wiki/Statistical_ensemble_(mathematical_physics)` Temperature determines how deep in the well the atoms are sitting.

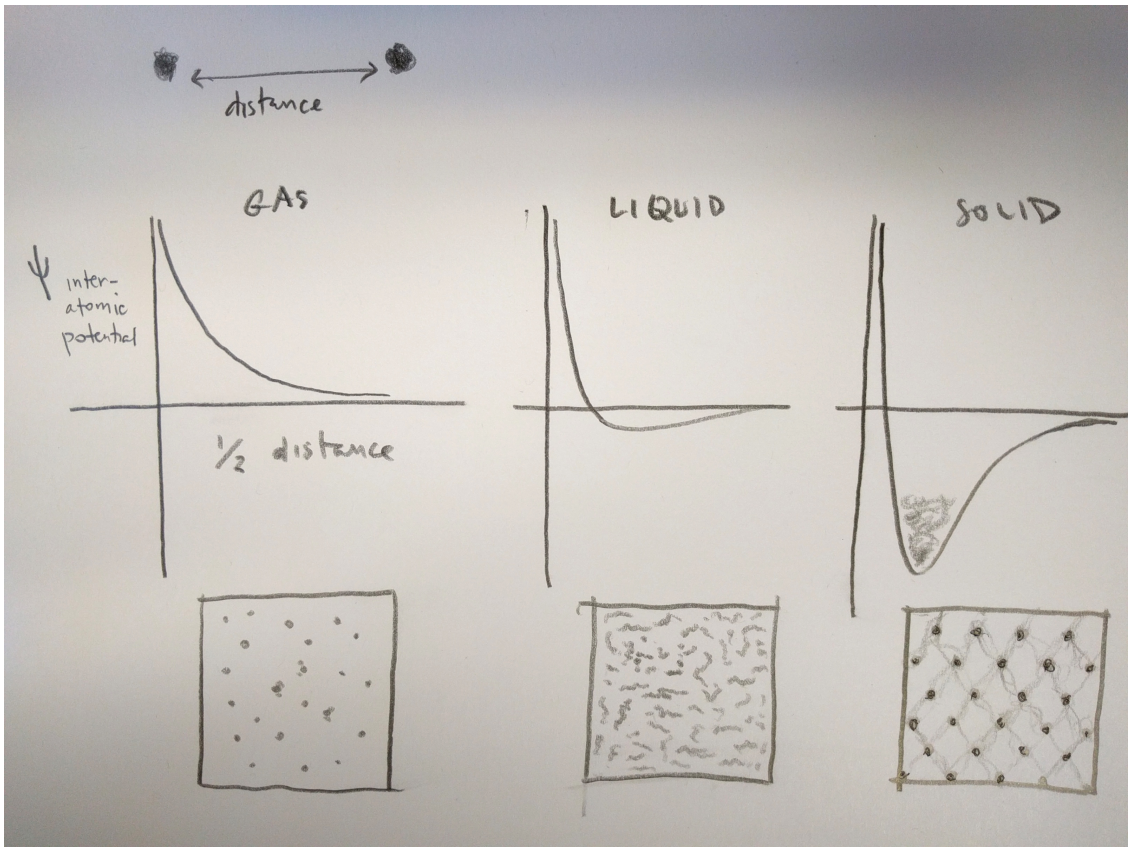

Figure 2.2: The "interatomic potential"– a simple description of interactions among atoms and the differences between gases, liquids and solids.

- The emergence of macroscopic properties can be predicted from interatomic potential (mechanical strength, thermal expansion).

  The restoring potential gives strength to solids in tension and allows the existence of shear waves, and gives

2.2.1 Kinetic theory of gases

molecular velocity and temperature relationship:

$$v_{RMS} = \sqrt{(3k_B T)} \tag{2.1}$$

k_B is relation between thermal and mechanical energy, at the particle level (Boltzmann hypothesized atoms but did not know anything about them!) $k_B = 1.38e - 23$ J/K

`https://www.youtube.com/watch?v=05uSAhcxMbE`

2.2.2 Zooming out to Thermodynamics

Now to zoom out, and consider a more removed kind of view.. we define average properties. Pressure arises from all those masses hitting each other and transferring their force– the average force transferred is the "pressure". So, pressure has no meaning for a single particle– only for a large enough ensemble !

- Density is the mass per unit volume, $\rho = Nm_a/l^3$ $[kg/m^3]$. (N here is the number of particles– atoms or molecules; m_a is the mass per atom or molecule; l is the length of one edge of a cubic volume) [indicates the units].
- Force is defined as mass times acceleration, $F = ma$ [kg/m^2/s^2= N] (where N here is Newtons, the unit of force!)
- Pressure is Force per unit area: $P = F/A$ [N/m^2].
- Energy is Force x Distance: [Nm]. In the case of a sound wave, energy can be thought of as Pressure x Volume: $(N/m^2) * m^3$!

Ideal Gas Law: the macroscopic $PV = Nk_B T$ (per atom, where N is the number of atoms), or $PV = nRT$ (per mole, where n is the number of moles.). "Ideal" means that the particles are not interacting with each other other than by perfectly elastic collisions. It says "You cannot change the Pressure or Volume without changing the Temperature".

2.3 The wave equation

The general mathematical description of a wave is much more complex, from which all kinds of wave behavior can emerge.. `https://en.wikipedia.org/wiki/Acoustic_wave_equation`

A good visualization of the standing wave: `http://www.acs.psu.edu/drussell/Demos/StandingWaves/StandingWaves.html` .

Figure 2.3: An exhibit at the Exploratorium, San Francisco, showing the ideal gas law with rubber balls in a cylinder.

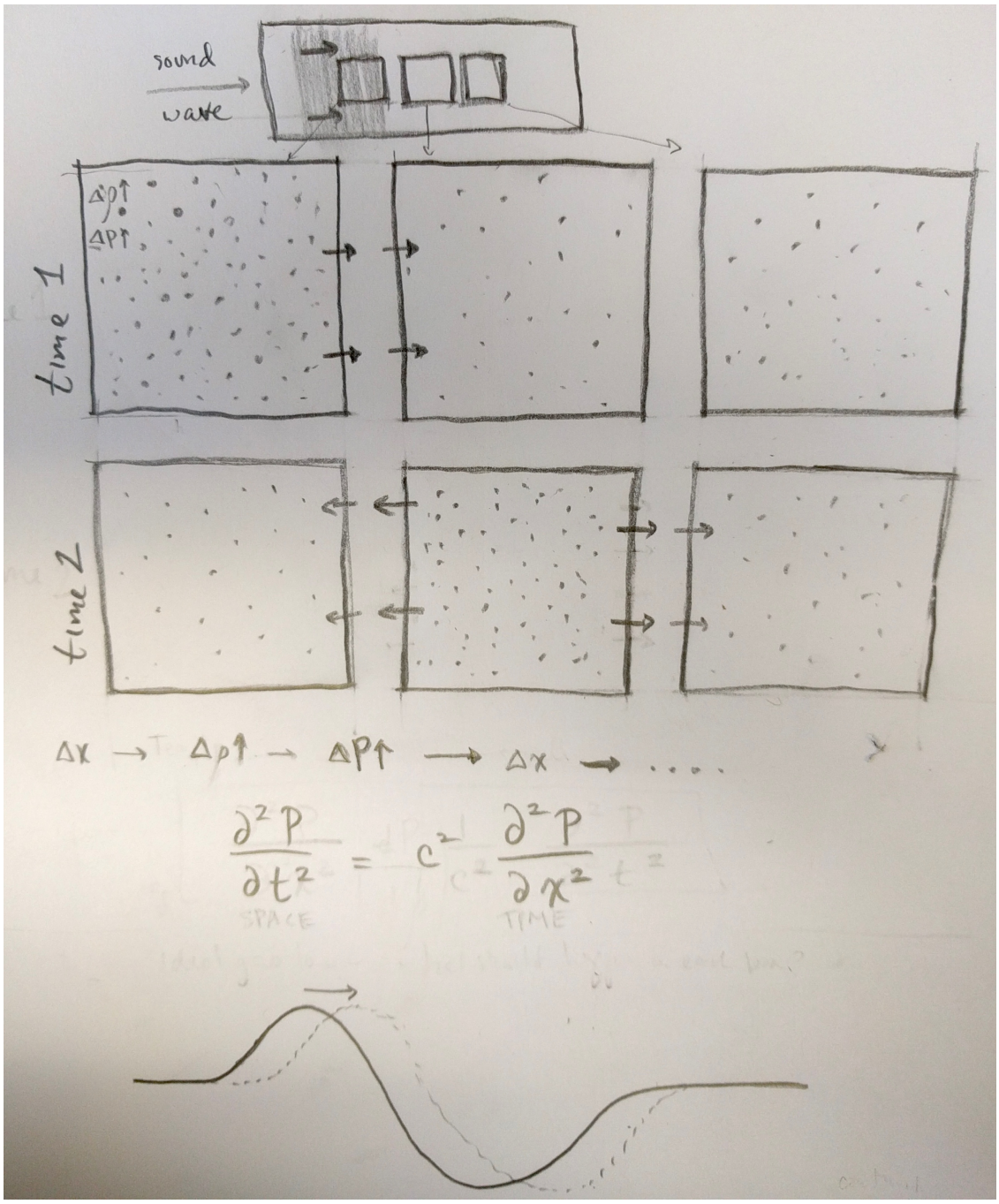

Figure 2.4: Sound waves !

2.3.1 Attenuation

Real materials... Though it is often a good approximation, rarely is something purely elastic.. There is loss of energy during wave propagation, conversion of wave energy to heat. The wave speed may not change, but the energy is lost.. a little bit every cycle.

Our hearing is very sensitive to attenuation:

Example: Loudness of Airplanes in the Winter!

Sounds are LOUDER in cold weather. This is because less information is lost during travel. Is the sound moving faster or slower?

Example: Arthur's earthquake listening experiment

2.3.2 Relation to $c = \lambda f$

Dave and Douglas's favorite equation: wave speed c equals the wavelength λ [m] times the frequency f [1/s]:

$$c = \lambda f \tag{2.2}$$

This equation that is constantly used in musical discussions: $c = \lambda f$ is a very reduced solution to the above equation. First, it is only strictly true for non-dispersive wave, i.e. the wave speed does not depend on frequency or wavelength.

Exercise: Setup Standing Waves in the Hallway

Setup a speaker that can play a single sine wave with controlled frequency:

For example:

$\lambda = c/f$

f = 118 Hz; c = 343.2 m/s;

λ = 2.9 meters

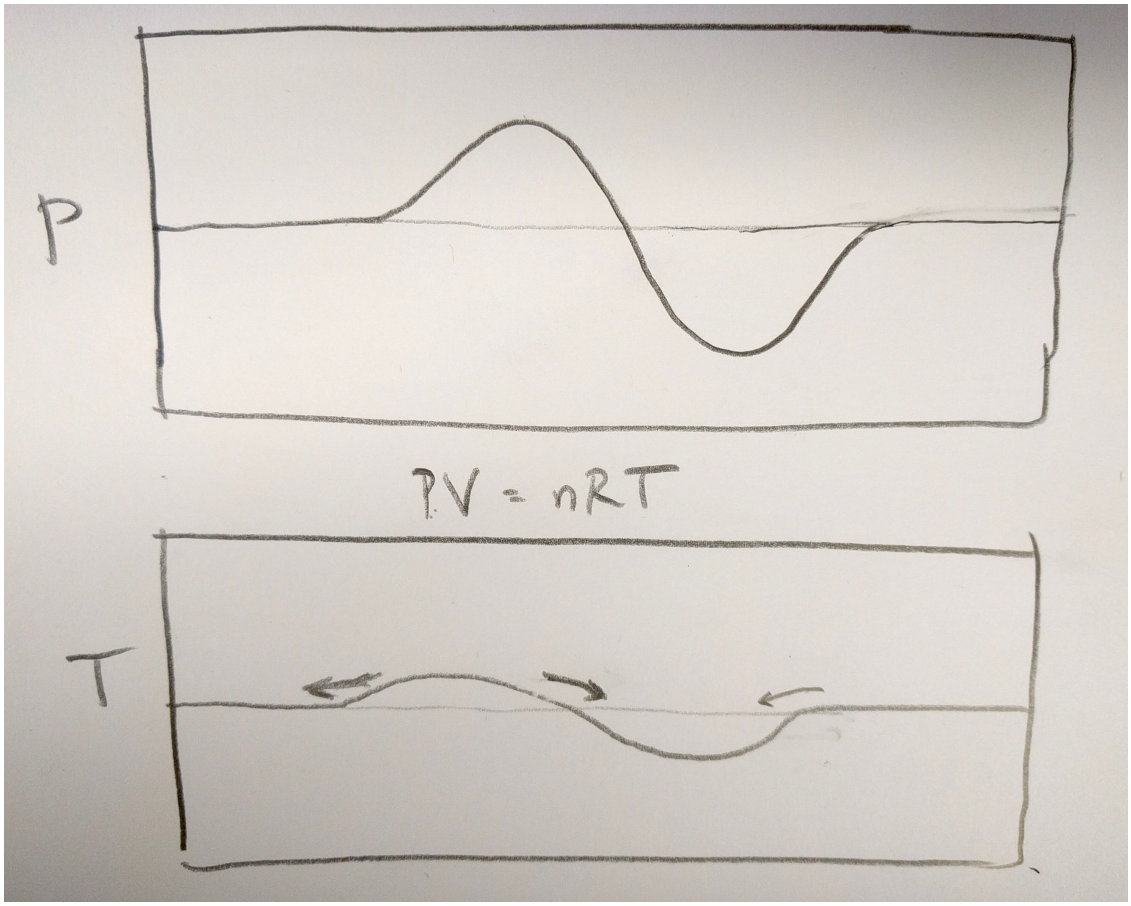

Figure 2.5: The ideal gas law says that if you change the pressure in a parcel of air, you have to change the temperature (at constant volume). Newton assumed the temperature equilibrated quickly; Laplace assumed it did not. Laplace was correct.

2.4 Sound Speed in Air

Isaac Newton (Principia, 1687) was close but wrong. He derived

$$v_s = \sqrt{\frac{P}{\rho}}, \tag{2.3}$$

assuming that a pressure wave in air was isothermal, meaning that the temperature could equilibrate between both sides of the wave, or that the temperature was not important. Recall the ideal gas law !

About a century later, **Pierre-Simon Laplace** solved it, as the understanding of thermodynamics had developed. A sound wave is close to "adiabatic"– heat cannot be transferred quickly enough from one side of the wave to the other to accommodate the changes in pressure by moving heat. This lead to the need to incorporate the heat capacity into the physics of the speed of sound, and leads to a prediction of the temperature dependence:

The bulk modulus K can be written as a function of the ratio of the specific heat capacities

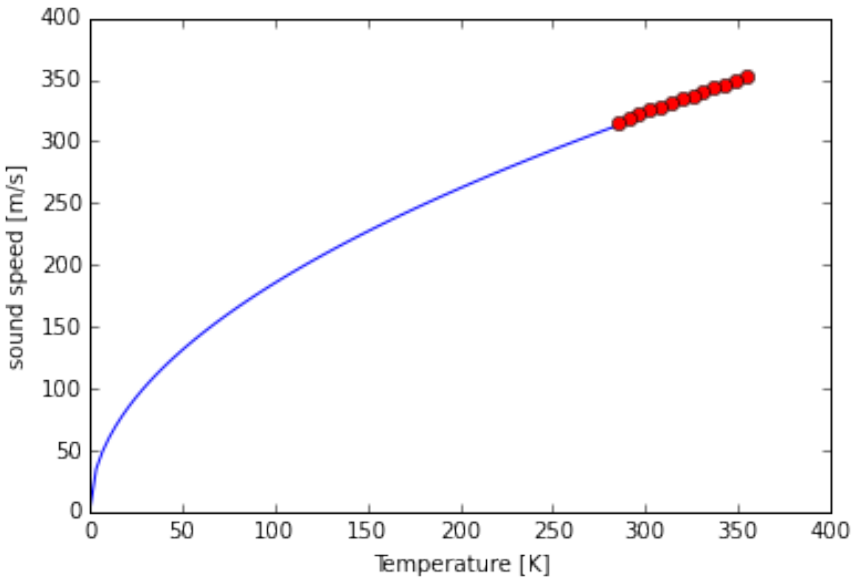

Figure 2.6: Plot of the speed of sound in air as a function of temperature. The red dots are experimental data.

at constant volume (c_v) and constant pressure, c_p, $\gamma = \frac{c_p}{c_v}$, a measure of the "squishiness" of a gas. For an ideal gas, they are the same; For a non-ideal gas, this ratio is greater than one, indicating that thermal energy is distributed (partitioned) differently when pressure or volume are held constant (Loy says: "a non-ideal gas stored energy in translational velocity as well as rotational and vibrational velocity–" I am not sure how to extend that to the mechanical properties). The bulk modulus can be written $K = \gamma P$ (wikipedia). The Blundell's book had a great explanation but I do not have it with me. So, Laplace showed that γ was the missing factor, and arrived at

$$v_s = \sqrt{\gamma \frac{P}{\rho}} = \sqrt{\frac{K}{\rho}} \tag{2.4}$$

(REFS: Blundell & Blundell; Loy ; Wikipedia)

2.4.1 Derived through the molecular velocity

(from Blundell's) The mean particle velocity as a function of temperature is

$$v_{rms} = 3k_B T/m_a \tag{2.5}$$

and the speed of sound is related directly to that particle velocity.

$$v_s = \sqrt{\frac{1}{3}\gamma v_{rms}} \tag{2.6}$$

so the speed of sound in air as a function of temperature is:
$v_s = \sqrt{\gamma k_B T/m_a}$

2.4.2 Derived through the ideal gas law

from Musimathics (Loy)

2.5 Waves in Solids

Because of the depth of the potential wells (conceptually), a solid can support more than compressional waves. It can also support shear waves because there is a restoring force pulling atoms back into their energy minimizing positions in the solid. So the wave equation becomes much more complex, full of tensors !

Ballistic shear waves propagate at

$$V_s = \sqrt{\frac{\mu}{\rho}} \tag{2.7}$$

Ballistic P-waves propagate at

$$V_p = \sqrt{\frac{K + 4\mu/3}{\rho}} \tag{2.8}$$

about twice the velocity of S-waves.

Every solid has a somewhat different wave speed depending on its structure, that determine both the density and the effective elastic moduli. There are some apparent universal or near-universal scaling relations, such as a scaling of velocity to density. Here is a beautiful summary, comparing rocks and cheese:

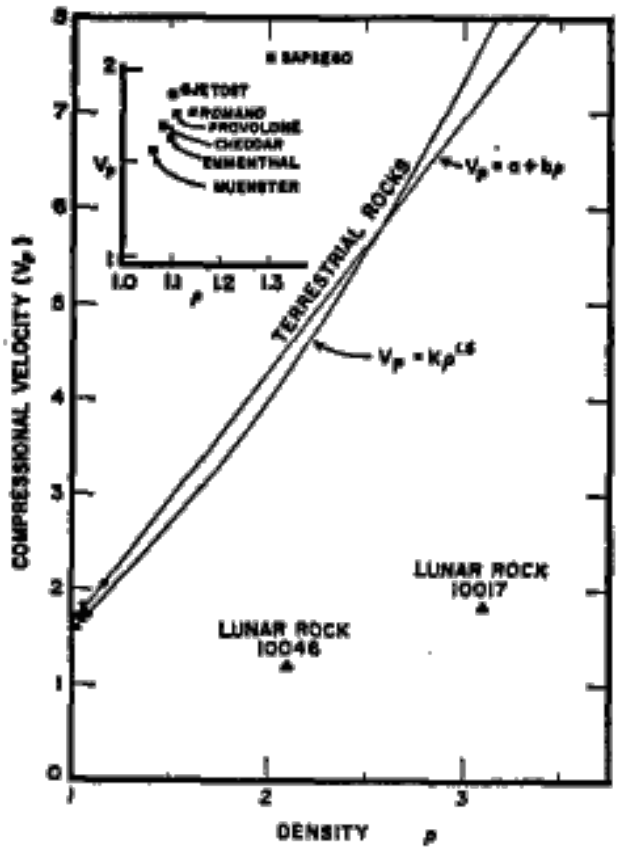

Fig. 1. Comparison of the velocity of sound for rocks with that of earth materials.

Figure 2.7: Scaling of shear wave velocity vs. density, for terrestrial rocks, for moon rocks and for cheeses. Cheeses hold; moon rocks do not, demonstrating that the moon is not made of cheese, but opening the question as to why.

2.5.1 Seismic waves in the Earth

- Niigata earthquake (go to Keynote presentation).
- Shaking in security cameras in Kobe `https://www.youtube.com/watch?v=LkZLjH2yUo4`
- STANDING WAVES: Taiwan earthquake
- GLOBAL: Tohoku, body and surface wave (go to Keynote presentation)
- SURFACE WAVES... dispersion (Niigata earthquakes)
- Free Oscillations of the Earth:
  `https://vimeo.com/245658903`

2.5.2 How does the wave speed depend on temperature in solids?

The effect of temperature on the wave speeds in solids is the opposite of that in gasses, and has a very different physical origin. The higher the temperature the lower the velocity. This is because the elastic modulus decreases with increasing temperature !

2.5.3 How do we measure the wave speed in a material ?

There are many many ways. However, the simplest is to know precisely when you trigger a source, and measure how long it takes to reach an instrument (measuring the wave form at that spot), called the delay time δt. Or you can measure the δt between two instruments. In these two cases, the distance between the source and station is known, dx. So the equation is simply, $V = dx/dt$. How do you expand this equation to a form that lets you determine V from measurements ?

Chapter 3

Sonification

In the previous chapter we scratched the surface of vast questions on the human perception of sound and light. Here, we begin to generate sounds, keeping in mind the idea of tuning to our perception, but starting with the basics, and moving to the ridiculously complex. Regarding representation of data, there is a simple spectrum to lay out, shown in Fig. 3.1. On the left, we have "direct sonification" that some call "audification" [1]. Direct sonification is the place to begin when the data is oscillatory (around 0, more or less, or shiftable by a "DC offset"), like seismic waves or ocean tides or planetary orbits or human heartbeats, etc. However, most data will not be that and more aesthetic choices need to be made– more freedom and more responsibility! So on the other end of the spectrum, we define "tonal representation", in which pure tones can be generated whose properties are mapped to some aspect or aspects of the data itself. In between is a spectrum of directness of those mappings. This one-dimensional spectrum may not hold up for long! We will start with direct sonification (and even below that), and then go to the far end, and then the middle.

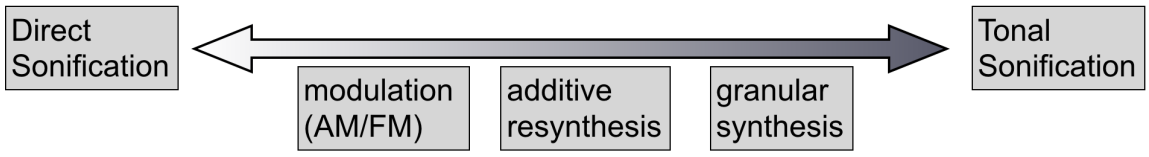

Figure 3.1: Spectrum of sonification approaches from direct to tonal, with the respective sections in the text.

[1] "The Sonification Handbook", Thomas [ ] editor... Chapter X.

Table 3.1: List of codes for sonification. Each one is distributed as both a notebook (.ipynb) and a script (.py)

Name	Description	Section ref.
s00_SimplestSounds		
s01_DirectSon_ex1		
s02_TonePrimer		
s03_TonalRep		
s04_GranSynth1		

3.1 Making simple sounds (s00_SimplestSounds)

Before we start sonifying real data, we are going to first generate sound in the very simplest way, and before we do that, we are going to generate very simple data !
FIND BETTER WAY TO INCORPORATE CODE! **NB0_make_simple_sounds**

3.1.1 Geometry without time

The sine function is a common way to generate an oscillatory signal that can be turned into a sound. At its simplest, it is purely a geometric relationship between limbs of a triangle, described by an angle, in degrees or radians. The meaning is easiest to visualize as a triangle embedded in a circle. sine, cosine, etc... The sine wave can be generated by considering the number of times you go around the circle, `n_cycles`, with no notion of how fast the motion is.

[add image of the circle with sine and cosine ratios]

3.1.2 Frequency

Frequency, f, is then how many cycles a point makes per second. See the code for explanations.
$p = 1/f$ in units of seconds: `npts_cycle` is the number of points per cycle.
`dt` is the time between each data point: `dt = p/npts_cycle`
`fs` is the sampling frequency which is not the frequency and must be higher than the frequency you want to hear ! (explain Nyquist frequency).

3.1.3 Adding oscillators

3.1.4 Basic Sound Properties

pitch, volume, attack, timbre...

3.2 Amplitude and Envelopes

amplitude is "volume" in the sound perception sense (refer back to section on perceived loudness and loudness curves). Decibel is a measure of relative volume. Sounds are normalized. Bitdepth.

3.2.1 Envelopes: impose time variation of the amplitude.

Time variation of the amplitude !

The envelope alone can be the data– this idea will come back later in additive synthesis (related to phase vocoder and maybe auto-tune).

3.3 Micro-primer on Sound Analysis (signal processing)

3.3.1 Fourier transform and the Spectrogram

The (Joseph) Fourier transform (FT) has to be among the most useful pieces of math ever invented. It converts a signal x from the time domain $x(t)$ to the frequency domain $X(f)$, also called the spectrum.

$$X(f) = \int_{-\inf}^{+\inf} x(t) e^{-i2\pi ft} dt \tag{3.1}$$

In an algorithmic sense, you calculate as follows:

1. for one value of f_j, perform the integral over t, where j is the index of the frequency vector.
2. assign that scalar value to $X(j)$
3. repeat for the next j

It is really conceived and designed for periodic oscillatory signals (frequency content not changing in time) but it is used all the time for signals that change a great deal in time.

Phase information related to i tells you how the oscillator for a given frequency is shifted in time. Discrete FT (DFT) is how it is calculated for digital signals... the cosine transform gets the real part and the sine transform gets the imaginary part (phase).... Fast FT (FFT)... etc..

3.3.2 The inverse FT

And then a signal can be reconstructed using the inverse Fourier transform.

3.3.3 The Spectrogram, short time FT

Many many methods for sound analysis are based on the spectrogram, and we will use it often...

[illustrate how to calculate the minimum time window for the STFT from the frequency content of interest.

3.3.4 Other means of extracting frequency information

Wavelets !

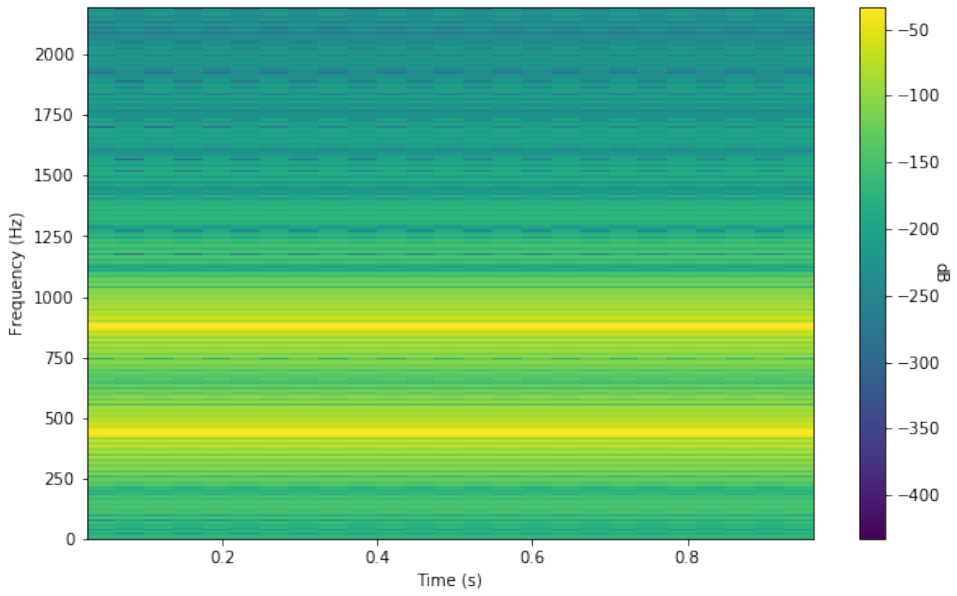

Figure 3.2: The spectrogram representation of the signal in Fig. **??**. (Replace this with a more interesting one or two, for comparison purposes.)

3.4 Direct Sonification or "audification" (NB03_DirectSonification)

When the data is a signal that oscillates around zero, it is called "stationary" in that the mean does not change with time. Elastic waves, like a sound wave or seismic wave, always restore to a reference position because in an elastic material, the force required to restore the material to its original position increases with distance from that position! Particle motion in ocean waves are also data because gravitational forces and mass balance cooperate: the displaced mass increases in one spot and decreases in another, creating gradients in "gravitational potential" that drive them back. Waves gradually dissipate this energy.

Stationary data can be simply converted to a sound and saved as a sound file, which we refer to here as "Direct Sonification". The duration can be sped up or slowed down to bring it into the audible range, if it is not already there. In this section, we describe some methods for doing this.

Before sonifying any data, of course you have to get the data into python. In the Appendix, Section **??** is a primer on getting data into python, and an accompanying notebook full of demonstrations.

NBA01_GetDataIn

We will use three examples: (1) a normal earthquake, (2) the free oscillations of the Earth, and (3) the first LIGO recording of gravitational waves.

3.4.1 Time shifting

NB03_DirectSonification

The basic approach is easy when you are not changing the number of samples in a signal, n_{samp}. The duration of a signal t_{signal} is equal to the number of samples times the time difference between

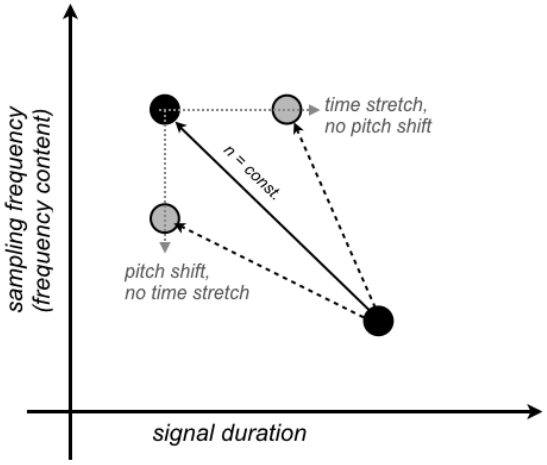

Figure 3.3: To shorten a signal, its sampling frequency must increase. If the number of samples is constant, duration and sampling frequency are dependent; if n can be varied, these can be independent, but the range is limited.

each signal, dt_{signal}, which is the inverse of the sampling frequency, $dt_{signal} = 1/f_s^{signal}$, i.e.:

$$t_{signal} = n_{samples} dt_{signal} = \frac{n_{samples}}{f_s^{signal}} \tag{3.2}$$

If you know the duration of the sound that you want to make (i.e. you know the length of the movie that you want to synchronize with the sound), t_{sound}, then re-arrange that equation to the form $n_{samp} = t_{signal} f_{signal}$ and set it equal to the same equation for the sound (i.e. n isn't changing), and re-arrange to solve for the new sampling frequency for the time shift:

$$f_s^{sound} = f_s^{signal} \frac{t_{signal}}{t_{sound}}. \tag{3.3}$$

Alternatively, if you can choose a reference frequency in the original signal and a target frequency in the audible range to which you want to shift the refreference frequency. This ratio has to be true:

$$\frac{f_{ref}}{f_s^{signal}} = \frac{f_{target}}{f_s^{sound}}. \tag{3.4}$$

Rearrange this equation such that $f_s^{sound} = f_{target} f_s^{signal} / f_{ref}$. For example, if you want to focus on a band centered on $f_{ref} = 0.001$ Hz in the seismic signal, and shift it up to the low end of the hearing range, say $f_{target} = 300$ Hz, Then you put this sampling rate into the wave file writing command in the code. EZ-peazy.

The scaling can also be calculated by choosing a new duration. If the original duration is dur_0, and the new desired duration is dur_{new} then the "speed factor" $s = \frac{dur_{new}}{dur_0}$, and the new sampling frequency $f_s^{sound} = s.f_s^{signal}$.

This path is demonstrated in the first notebook on direct sonification:

NB3_DirectSonification

Another example, listening to the Earth's free oscillations (bell modes) is demonstrated in

NB5_DS_FreeOscillations

3.4.2 Limited time stretching

Clearly, above, the frequency shift and the resulting sound duration are not independent. If you need to alter that relationship there are a few ways to decouple them. You can change the number of samples, by decimating or interpolating to a lower number of samples. . Decimating involves making a new signal comprised of every Nth data point; if $N = 2$, the signal will be half as long. To get the lower slope (a) ...

However, to take the other slope, you need to add information, which is much more complicated and usually does not work very well because our ears can pick it up fairly quickly. For example, add cycles of each frequency. Some methods involve reconstruction the signal based on the Fourier transform or the spectrogram. (RELATE THIS TO THE FIGURE !) There time stretching without pitch-shifting algorithms, or equivalently pitch-shifting without time-stretching, as shown in Fig. 3.3. However, these algorithms are quite limited in their range, only giving a factor of 1.5 or so in either direction before they start to sound less natural, because of introduction of artificial frequencies (digital artifacts) from the algorithm.

You can try in librosa (or Audacity) : to time stretch without changing the pitch: `https://librosa.github.io/librosa/generated/librosa.effects.time_stretch.html`
to pitch-shift without changing the duration: `https://librosa.github.io/librosa/generated/librosa.effects.pitch_shift.html`
How do these work? What is their practical limit?

3.4.3 Application to LIGO-detected gravitational waves

Here we try these on the LIGO data ! Why on the LIGO data? because the signal is so short in terms of the number of cycles that we can barely perceive the interesting parts. Here's a nice explanation: `https://www.youtube.com/watch?v=B4XzLDM3Py8`
But what is the wavelength ?
Recall the relation between wavespeed, wavelength and frequency $c = \lambda f$. The speed of light is $c = 299792458$ [m/s] or $\approx 3e8$ [m/s] . The frequency of the chirp increases up to about 10 Hz in the real data (almost in the human range!), so the wavelength is $\lambda = c/f = 3e7$ [m] or $3e4$ [km]. The diameter of the Earth is $12,742$ [km] or $1.3e4$ km, about half the wavelength of the gravitational wave.

NB3_DS_LIGO

3.5 Filtering

(NB01_FilteringIntro)

Filter theory is vast. Here we keep it practical and point you to resources for deeper study (Musimathics book 2 by Gareth Loy, MIT Press, for a good mathematical introduction; or 100 other books).

Convolution of two signals in time domain is equivalent to multiplication of the spectra of two signals in the frequency domain.

3.5.1 A simple example:

We will vary the "order", the steepness of the sides of the filter, and the "quality factor" Q, essentially the width around the center frequency (determines the cutoff frequencies in this case). `https://en.wikipedia.org/wiki/Q_factor` :

> "Higher Q indicates a lower rate of energy loss relative to the stored energy of the resonator; the oscillations die out more slowly. A pendulum suspended from a high-quality bearing, oscillating in air, has a high Q, while a pendulum immersed in oil has a low one. Resonators with high quality factors have low damping, so that they ring or vibrate longer."

There is some tradeoff between these two factors. **[calculate a damped signal !**
ARTHUR's code !

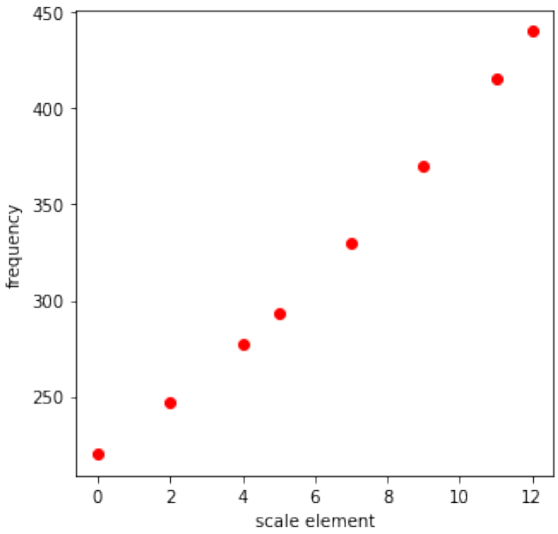

Figure 3.4: Frequency as a function of element of the scale (starting at a root note of A3 = 220 Hz, in this case).

3.6 A bit of music theory

Relation between notes and frequency:

So called "equal tempered" intervals are constructed by (as nicely explained in *Musimathics*[2]):

$$f(k) = f_0 \cdot 2^{k/12} \tag{3.5}$$

In "classical western music" for lack of a better term (as far as I know), these are whole numbers, integers: **np.arange(12)**, for example.

Or, a more useful form, with v as the number of the octaves shifting from the reference:

$$f(k, v) = f_0 \cdot 2^{v+k/12} \tag{3.6}$$

So, if we take the reference octave to contain A440 (A4, where the 4 is the octave on the 88-key piano keyboard), C4 is

$$C4 = 440.0 \cdot 2^{-1+3/12} = \frac{440 \cdot 2^{3/12}}{2} = 261.626\ Hz. \tag{3.7}$$

There are many other ways of constructing intervals, in western and other musical traditions and evolutions, that you can easily produce to make listening experiments for yourself (for more, Dave Sulzer's class, "Sound: Physics and Perception" or "Music, Math and Mind"). My favorite reference for this stuff is *Musimathics*[3]. "Just intonation" is an older tuning, constructed from ratios of small integers, as explained in the notebook

NB4_IntroTonalRep_RTcmix

Our first RTcmix score is to listen to these.

[2]Gareth Loy, Musimathics: the mathematical foundations of music, vol. I. MIT Press, 2006, Chapter 3
[3]Gareth Loy "Musimathics, vol. 1" 2006, ch. 3

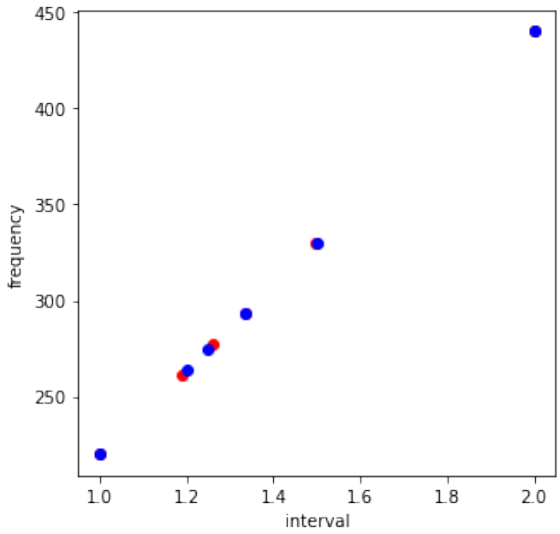

Figure 3.5: A comparison of major 3rds, 4ths, 5ths in just intonation (blue) and equal temperament (red). Here, in the middle of the piano keyboard, they are quite similar, and I certainly cannot tell the difference, but it is possible that the differences become more perceptible at either end of the scale. Test this!

1. Using the 12-dot clock face to visualize scale structure
2. Modes ! and Dictionaries !
3. Visualizing modes on the clock face !
4. Reduce the intervals down to the limit of our perception !

3.6.1 Visualizing scales and tone relationship on a wheel

Right now, this is just a nice pretty thing, but these might become useful in visualization of musical patterns later on.

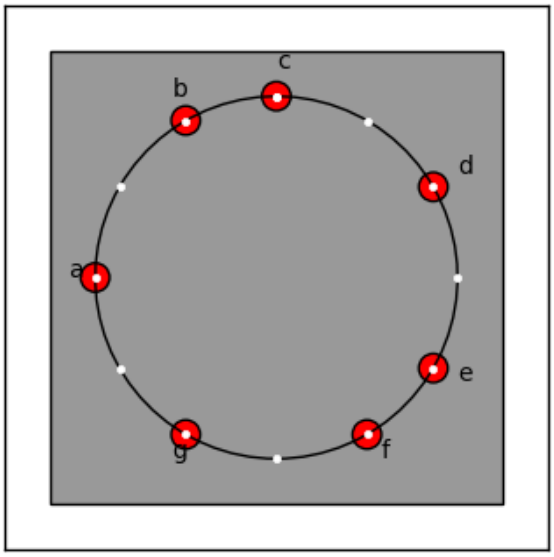

Figure 3.6: The most standard reference-y scale in western music, represented on a 12-tone wheel.

3.7 Intro to RTcmix

The roots of RTcmix and unit generator languages:
`https://en.wikipedia.org/wiki/MUSIC-N`
(Brad's history flowchart of sound synthesis languages!).
Book on RTcmix from [ ]:
`http://sites.music.columbia.edu/cmc/courses/g6611/spring2017/week2/2015-0626_RTcmixbookrough1.pdf`

3.7.0.1 The basics of unit generator

Recall the NB1: the sine wave oscillator we built in python is essentially what is meant by "unit". Instead of building long waveforms by starting over each time (as we did), the idea is that you only store a very few short waveforms in the computer memory, and do quick operations (scaling, shaping, etc) to them many times to generate a new synthetic waveform that does not have to be stored in memory. They built a world on the idea of the unit generator !

3.7.1 Making computer music 0.01: constructing scales in RTcmix

Run the notebook **NB4_IntroTonalRep_RTcmix**
First make western scales... Now we can play with microtonal music ! How small of a pitch increment can we hear? Make an array of k with smaller and smaller increments with **np.linspace(0,2,0.1)**, for example (and you can get rid of the 12!).

Try making sequences of notes constructed from stringing different modes or root notes or whatever together... Dictionary of the "modes", that can be used to construct different scales in different keys.

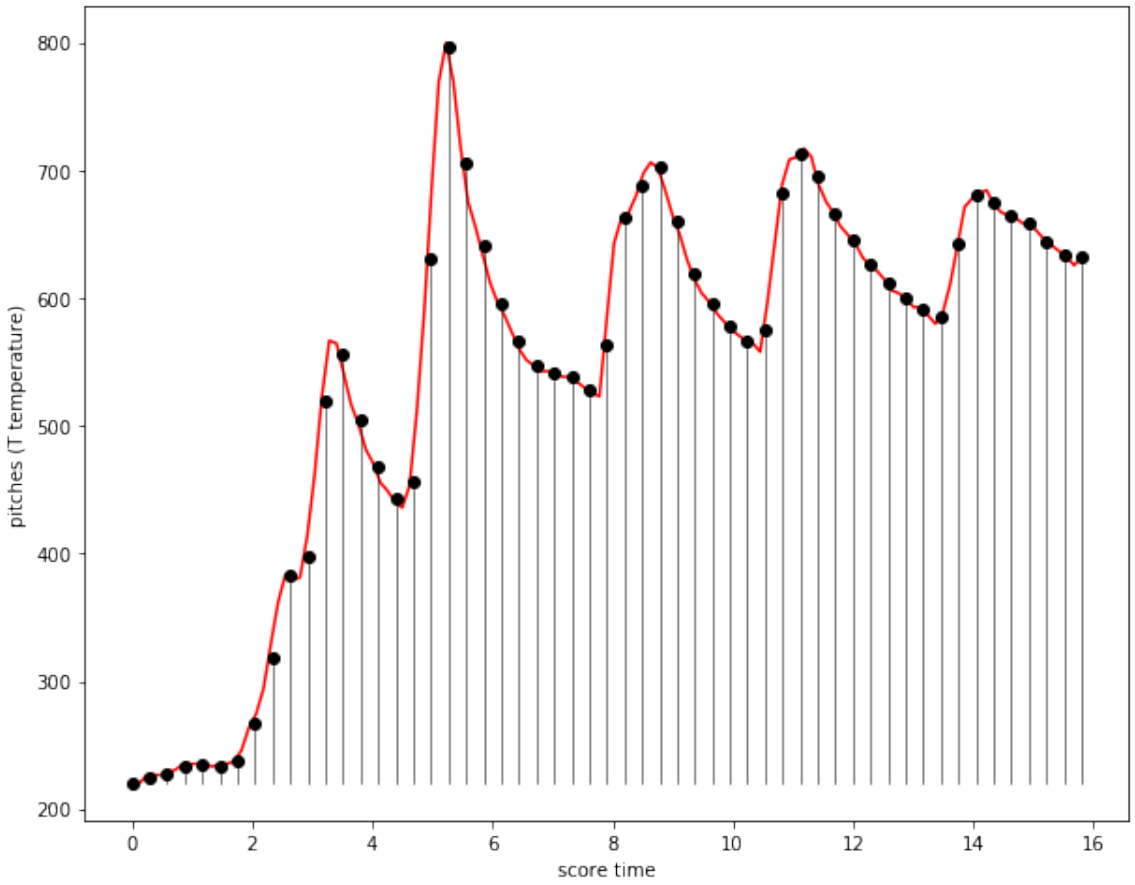

Figure 3.7: Select the times at which you want notes to start, and then find the values (mapped to frequencies) that occurs at that time.

3.8 Mapping data to tonal sound

Here, we start at the other end of Fig. 3.1.

3.8.1 Non-stationary, non-oscillatory data

As discussed earlier, most data is not oscillatory or stationary, so we cannot apply direct sonification methods. However, a whole interesting array of tonal representations and other synthesis methods are available to us. We will use RTcmix for most of these, because it is a relatively low-level musical synthesis language allowing us to generate original sounds. We will talk about many of sound properties that we will map from and modulate with the raw data.

3.8.2 Data-driven Composition !

3.8.2.1 No Rhythm ! time determines pitches

No rhythm here means that every note is the same distance apart. No note has any greater temporal importance, or rhythmic value; there is no "1", etc. This might be useful in some situations, and we can generate a very smooth tonal sliding with the data, kind of like FM synthesis. [**Demonstrate this in the notebook**].

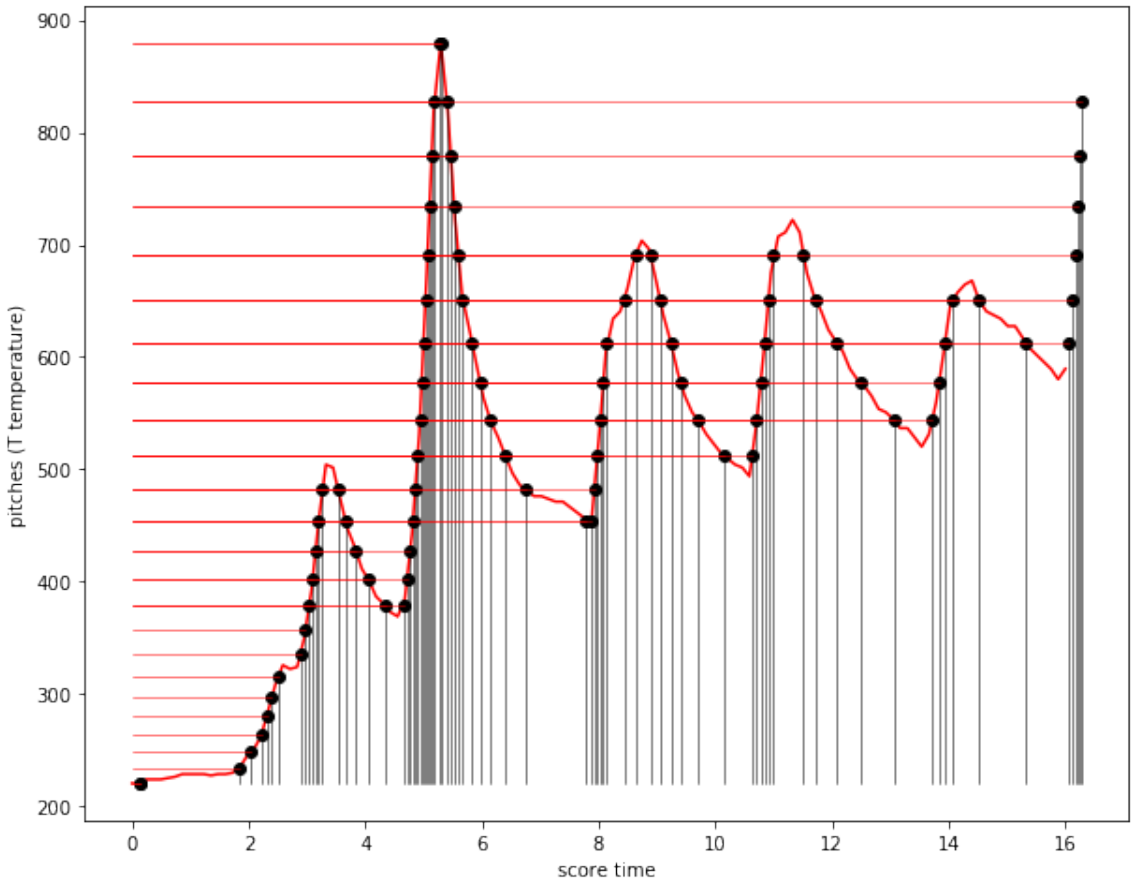

Figure 3.8: The frequencies or note sequences are first determined (on y-axis) and then determine when the notes occur by where the occur in the data.

Of course, you could add rhythm to this, by picking non-regular time intervals, or generating multiple tracks with different sonic aspects that give the notes different weight in time. Many possibilities for composing this way !

3.8.2.2 Got Rhythm! pitches determine time

Another way to generate tone sequences was developed by Owen Evans and Cian Wilson (and me), and is hence called the "Owecianizer". The idea is that you choose the note sequences, tonal structures, whatnot, to represent your data, and then find where in time, those notes occur in the data. This way, the rate of change (the slope) of the data curves determine the rhythm, the timing of the occurrence of notes. The great benefit of this method is that it gives a very physical sensation of how quickly the values are changing, which is very important when trying to convey temporal patterns of a physical process.

The algorithm is very elegant. I will explain it here in pseudo-code, but for now, try to read it in `notepicker.py` in `./1_codes/modules/`.

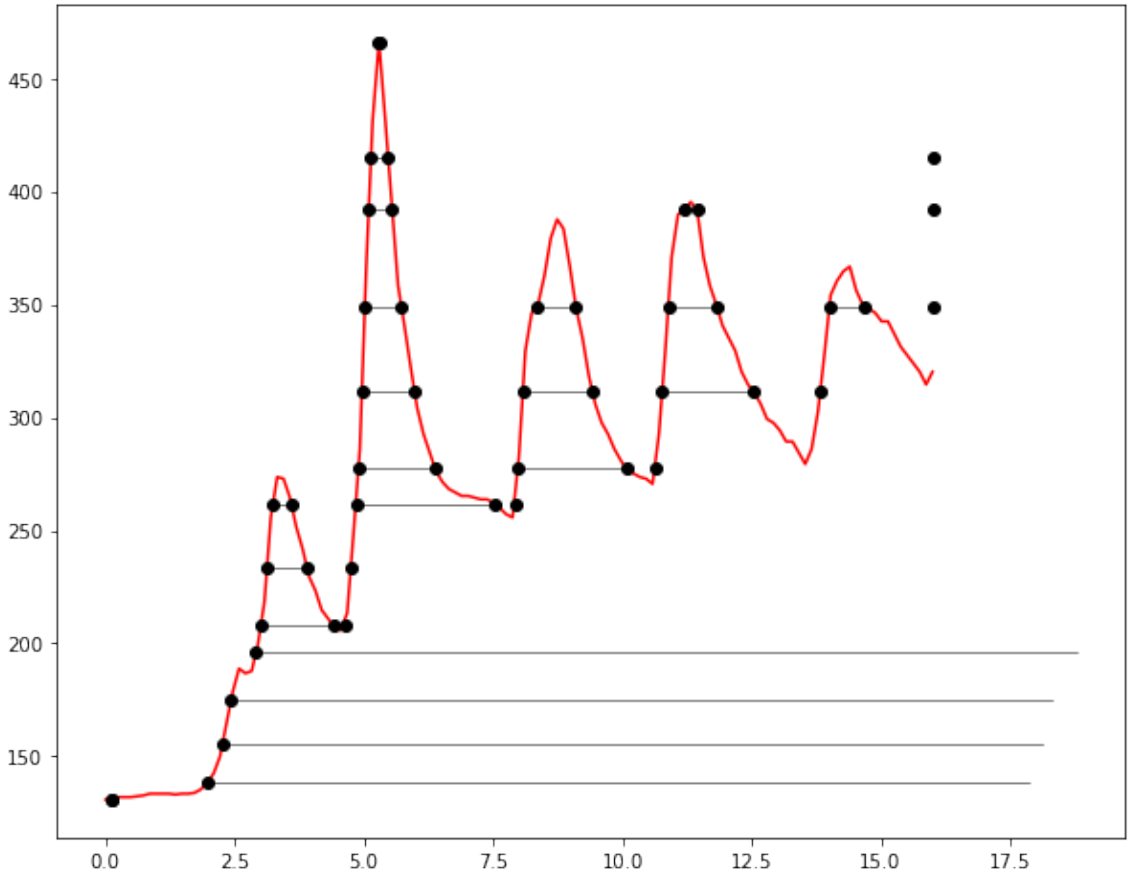

Figure 3.9: Construct chords from the shape of the data.

3.8.2.3 Got Harmony! Data-driven chord construction

For now, one more way to construct a synthetic sound from the data builds on the Owecianizer. Instead of one note at a time, each pitch is played until it recurs in the data, so the chords are built up and swell the higher the data goes. They could also be built downwards to enhance the sense of expansion of the chords.

3.8.3 Tonal choices

How do we choose? These are artistic/emotional/aesthetic decisions ! Big open door ! But (in this class) you have to explain your decisions !

There was a discovery of a geometrical relationship between keys that allowed them (tone centers) to be plotted and mapped in a very elegant way: Tymoczko "leading voices". This representation lends itself well to data-driven composition because it allows a smooth mapping from several parameters to drive migration through key changes. (I hope we have time to explore this further this semester, after Spring Break).

3.9 The middle ground: In between tonal representation and direct sonification

Much more advanced RTcmix !

The middle ground in Figure 3.1.

Often, the time shifting is not flexible enough– time stretching doesn't get you there.. can't get the sound into a good pitch range and still convey enough information to give it much meaning. There are a few approaches that can extract information from the waveform and use it to generate a new sound that is a close representation of the data, but not as direct as audification.

What kinds of data? Stationary and oscillatory but few cycles: synthesis, modulation, additive resynthesis (Use the LIGO data as an example). Non-oscillatory... (breathalyzer).

3.9.0.1 Mapping data to various parameters...

3.9.1 Data-driven Signal Modulation

3.9.1.1 Frequency Modulation

YES add this !!

3.9.1.2 Amplitude Modulation

how does this work for radio transmission and synthesis?

3.9.2 Granular Synthesis

[LOTS TO DO HERE !!! STUDY THE RTCMIX SECTION !

3.9.3 Additive Re-synthesis (ARS)

[We will cover this later in the semester... as needs emerge]

Extract the envelop with the Hilbert transform: $(abs(hilbert(y)))$.

do the above many many times:

1. filter bank..
2. extract envelope for each
3. oscillator bank with envelope multiplied
4. sum them all up...

3.9.4 Chord Stacking

(HAWAII GPS)

3.9.5 Chord Sweeping

Lone star tilt

3.10 Summary of sonification

Return to the Spectrum idea...

Chapter 4

A micro-primer on the physics and perception of light

4.1 Overview

4.2 Physics of Light

the frequency band of light...

4.3 Perception of Light and Motion

The human vision system similarly refers to ...

temporal perception of motion, "framerate" – 30 fps for cinema... if you shake your head you can see frames.. a sort of aliasing.

4.3.1 Strengths and limitations of visual perception

4.3.2 How do they talk to each other, neurologically speaking?

Endless everyday examples. How do we design and generate representations of data from processes and dynamics in any domain that are not inherently visual or sonic, using both senses to perceive patterns in that data?

Chapter 5

Animation

5.1 Strengths and Limitations of visual perception

- Frequencies of the visual spectrum, spatial gradients in color !
- Examples: An animal in a tree, faces, landscapes, textures in materials, sharpness of objects ?
- Perception of time through vision: frames per second, function of distance...

faces (in time)
`http://journal.frontiersin.org/article/10.3389/fpsyg.2015.01248/full`
hand waving, near, far, seconds ticking... metronome.

5.1.0.1 How do we see motion ?

What is our time resolution ? How does distance from the object affect time resolution ?

5.2 Simple animations

The basic idea:

1. Set the Stage (all that is fixed and unchanging)
2. Build the moving/changing elements
3. Know how to remove them !
4. The loop– either drawing out frames or using matplotbib.animate

5.2.1 Example 1: Throbbing dot (changing size and color)

5.2.2 Example 2: Slider (motion)

5.2.3 Example 3: Visualization of data: Breathalyzer

Work through animation parts of Breathalyzer

5.3 Thoughts on Color!

5.3.1 How do we design the colormaps?

https://cre8math.com/2015/09/05/josef-albers-and-interaction-of-color/

Kristen Thyng, analysis of colormaps in scientific visualization !
http://kristenthyng.com/media.html

5.3.2 Do: Albers programs Interactions of Color animated!

5.4 Visualization of music and tonal structures

Paul Klee... Methods for animation (without sound), run time and frame writing. Music as data; algorithmic music. Animate circle of 12 tone scale, circles of fifths, fourths, etc, and key changes, etc. Representation of time ? Visualizing western musical structure. What aspects of data would trigger aspects of music ?

Moving through a musical structure IS moving through a process !
Tymozcko geometry of music structure.
Circles of fifths, etc. with rhythm, but no sound YET !

5.5 Introduction to Object Oriented Programming

Multiple objects.. Intro to OOP ? 2D: Dots changing, Dots moving.. Lines, colors.. basemaps ?
Animation packages that we could use:
http://zulko.github.io/moviepy/

http://zulko.github.io/blog/2014/11/29/data-animations-with-python-and-moviepy/
https://bitbucket.org/pyglet/pyglet/wiki/Home
http://stackoverflow.com/questions/169810/2d-animation-in-python
Good tutorial on object oriented programming:
https://jeffknupp.com/blog/2014/06/18/improve-your-python-python-classes-and-object-oriented-
Vispy interactive scientific visualization:
http://vispy.org/index.html

5.5.1 Dots movie

5.5.2 Sliders movie

5.6 Synchronization of sound and animation

http://superuser.com/questions/508331/ffmpeg-add-two-audio-streams-to-video

https://trac.ffmpeg.org/wiki/AudioChannelManipulation

5.6.1 Synchronization of sound and image

What questions emerge as we try to bring the images and the sounds together ?

Chapter 6

Back to the data: what is nature of the process?

Now that we have a sense of the tools at hand, this chapter brings us back to the data. What do we want to ask of it ? What do we want to demonstrate and convey to ourselves and to the audience? (Who is the audience? Does that matter?) What is the nature of the process ? Across what length and time scale is it discrete? Across what length and time scale is it smooth?

6.1 Discrete events: Beeps and bloops

Discrete events: sound "grains" (coarse granular synthesis)

6.2 Smooth changes: sliding tones and motion

Chapter 7

Machine Learning and Pattern Revelation

How to test for the presence or absence of some pattern that you think might be in the data ? How to explore your data with no preconceived notions of what patterns may be lurking in there? What do we mean by patterns ? Correlations ? Relationships? Causes and effects?

The starling flight patterns ? What are the rules ? Simple rules with many agents produce complex patterns... What are those rules?

Machine learning is becoming pervasive in our daily lives and will soon be everywhere, already entering our subconscious as expectations in the behavior of objects, cars, appliances and web applications. It is often the machinery underlying processes that are often mis-labeled "artificial intelligence". Here, we will use it in its capacity for seeking patterns, for helping us pick out patterns in complex multi-dimensional data, and then use the output to modify our movies, to illuminate those patterns.

7.1 Clustering

"Unsupervised" - no telling the algorithm what to look for.

7.2 Neural networks

Chapter 8

Spatialization of sound and graphics

8.1 Spatialization of Sound

Multi-channel sound .. what can we perceive (Hartmann? physics today article... ? Why do it ?

Mixing and Motion: (VBAP, Ambisonics, etc).

8.1.1 Do: simple VBAP

DO: a really simple example of VBAP in python...

8.2 3D graphics, AR/VR

8.2.1 discussion subsection

simple ones in python, other environments– Processing, OpenGL, Unity...

Chapter 9

APPENDIX: The Tools!

9.1 Installing python

[>> is the prompt in the terminal– don't type it.]

1. First decide how you want to do the installation. You probably have python already installed (just type >> **python** and it will tell you what version you have. We will be using python 3, which is at 3.5.2 or 3.6.0 at this point. If you want to keep your 2.7 (the last maintained version of the python 2 lineage), you can create different python environments, described here: `http://conda.pydata.org/docs/py2or3.html`.

2. If you haven't used python at all, the very simplest thing to do is install the latest version of python 3 (3.5.2 I think) using Anaconda: `https://www.continuum.io/downloads`. There are a few ways for macs. Once it is installed, you can type >> **conda info –envs** to see what versions you have installed. When you do the installation, allow it to write the path, to make this the default version of python. Type >> **jupyter notebook** and a browser window will open up. I'll show you the rest.

3. If you have a version of python that you want to keep using, but do not have jupyter, here's what you do: `http://ipython.readthedocs.io/en/stable/install/kernel_install.html`. You can decide what version of python you want the notebook to run.

4. We will install other python packages later, using **conda**, the installer that comes with the Anaconda distribution.

9.2 Do: Introduction to Python

There are many many great python tutorials and resources and references. Here is one of my favorites: `https://learnpythonthehardway.org/book/` The main thing is learning how to ask questions to google, which will then find the discussion on stackexchange. This is essential to learning to program, especially in python, because there is such a lively, helpful and non-arrogant community of developers and users.

1. NB1: How to use notebooks
2. NB2: Lists
3. NB3: very small intro to numpy and matplot lib

9.2.1 Do: Get data in to python!

1. Getting data in to python with various methods, depending on the nature of the data.
2. Looking at your data with static plots. Plotting trends, statistics, correlations. What is the meaning in the data?
3. I dunno right now.

9.2.2 Librosa

installation...

9.3 Installing RTcmix

Install RTcmix: `http://rtcmix.org/standalone/` and follow the instructions– probably easiest to use the first method if you do not use **git**. Put it here: **/usr/local/src/RTcmix** and learn with these: **/usr/local/src/RTcmix/docs/sample_scores**

9.3.1 The basic idea and ways to do it

9.3.1.1 Standalone RTcmix

9.3.1.2 python package

9.3.2 Troubleshooting

9.3.2.1 problems in Mac

9.3.2.2 problems in Linux

9.3.2.3 the OpenFrameworks standalone for Windows...

9.4 FFMPEG

Install ffmpeg: On a mac, what worked for me was
`https://github.com/fluent-ffmpeg/node-fluent-ffmpeg/wiki/Installing-ffmpeg-on-Mac-OS-X`
.
Type >> **ffmpeg** to see if it worked.

9.5 A note on code sharing

I hereby request that we keep these codes within the class, until the end. Then, when we see what we have created, we can decide what to do with it. I believe in the open source and free software tenets (`https://en.wikipedia.org/wiki/Free_software_movement`). However, we can certainly develop our codes in this microcosm before sharing them with the outside world. So, inside the class, I'll encourage sharing pieces of code as much as possible, so we can see and understand how that works, and then we'll discuss what to do with our codes at the end.