The Entropy of Musical Classification

A Thesis
Presented to
The Division of Mathematics and Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Regina E. Collecchia

May 2009

Approved for the Division
(Mathematics)

Joe Roberts

Acknowledgements

Without the brilliant, almost computer-like minds of Noah Pepper, Dr. Richard Crandall, and Devin Chalmers, this thesis would absolutely not have been taken to the levels I ambitiously intended it to go. Noah encouraged me to write about the unity of digitization and music in a way that made me completely enamored with my thesis throughout the entire length of the academic year. He introduced me to Dr. Crandall, whose many achievements made me see the working world in a more vivid and compelling light. Then, Devin, a researcher for Richard, guided me through much of the programming in Mathematica and C that was critical to a happy ending.

Most of all, however, I am indebted to Reed College for four years of wonder, and I am proud to say that I am very satisfied with myself for finding the clarity of mind that the no-bullshit approach to academia helped me find, and allowed me to expose a topic by which I am truly enthralled. I rarely find music boring, even when I despise it, for my curiosity is always peaked by the emotions and concrete thoughts it galvanizes and communicates. Yet, I doubt that I could have really believed that I had the power to approach popular music in a sophisticated manner without the excitement of my fellow Reedies and their nearly ubiquitous enthusiasm for music. This year more than any other has proven to me that there is no other place quite like Reed, for me at least.

I would also like to thank my "wallmate" in Old Dorm Block for playing the most ridiculous pop music, not just once, but the same song 40 times consecutively, and thus weaseling me out of bed so many mornings; my parents, for ridiculous financial and increasing emotional support; Fourier and Euler, for some really elegant mathematics; the World Wide Web (as we used to call it in the dinosaur age known as the nineties), for obvious reasons; and last but not least, Portland beer.

Table of Contents

Abstract

Music has a vocabulary, just in the way English and machine code do. Using chords as our "words" and chord progressions as our "sentences," there are some different ways to think about "grammar" in music by dividing music by some classification, whether it be artist, album, year, region, or style. How strict are these classifications, though? We have some intuition here, that blues will be more strict a classification than jazz, say, and songs of Madonna will be more similar to each other than that of Beethoven. To solve this problem computationally, we digitally sample songs in order to filter out chords, and then build a Markov chain based on the order of the chords, which documents the probability of transitioning between all chords. Then, after restricting the set of songs by some (conventional) classification, we can use the measure of entropy from information theory to describe how "chaotic" the nature of the progressions are, where we will test if the chain with the highest amount of entropy is considered the least predicable classification, i.e., most like the rolling of a fair die, and if the lowest amount corresponds to the most strict classification, i.e., one in which we can recognize that classification upon hearing the song. In essence, I am trying to see if there exist functional chords (i.e., a chord i that almost always progresses next to the chord j) in music not governed by the traditional rules from harmonic music theory, such as rock. In addition, from my data, a songwriter could compose a song in the style of Beethoven's Piano Sonata No. 8, Op. 13 ("Pathetique"), or Wire's *Pink Flag*, or more broadly, The Beatles. Appendices include some basic music theory if the reader wishes to learn about chord construction, and a discussion of hidden Markov models commonly used in speech recognition that I was unable to implement in this thesis.

Introduction

Motivations

Why chord progressions?

What gives a song or tune an identity? Many might say lyrics or melody, which seems to consider the literal and emotional messages music conveys as the most distinguishable components. However, many times, two songs will have lyrics that dictate the same message, or two melodies that evoke the same emotion. Instead, what if we consider the temporal aspect of music to be at the forefront of its character—the way it brings us through its parts from start to middle to end? In many popular songs, the chorus will return to a section we have heard before, but frequently, the melody and lyrics will be altered. Why do we recognize it as familiar if the literary and melodic material is actually new?

To go about solving the difficult problem of defining musical memory, I choose to analyze harmony, which takes the form of chord progressions, as an indicator of musical identity. Not only are chords harder to pick out than melody and lyrics, but they construct the backbone of progress in (most) music, more than other aspects such as rhythm or timbre. Many times we will hear two songs with the same progression of chords, and recognize such, but what about the times when just one chord is off? A difference of one chord between two songs seems like a much bigger difference than a difference in melodies by one note, but the entropy of these two songs is very near identical. The inexperienced ear cannot detect these differences, when if a song only differs from another song by one chord, they are likely classified by the same style. Styles seem to possess tendencies between one chord and another. I believe that all of this is evidence of the strength of our musical memory.

In support of the existence of these "tendencies," most classical music before the movement of modernism (1890-1950) was quite strictly governed by rules that dictated the order of chords [29]. A viio chord must be followed by a chord containing the tonic (the note after which the key is named) in Baroque counterpoint, for example, and this chord is usually I or i. Secondary dominant chords, and other chords not within the set of basic chords of the given key, must progress to a chord from the key in which they are the dominant, i.e., V/iii must progress to iii, or another chord from the third scale degree, such as ii/iii, in music before modernism. These chords are said to be "functional," but we can still label a chord as a secondary dominant even if it does not resolve as specified, when it is simply called "non-functional". Therefore, the marriage of probability theory and harmonic (chord) progression is hardly a distant relative of music theory.

Remarkably, the length of one chord seems to take up a single unit in our memory. The chord played after it seems to depend only on the one before it, and in this way, a sequence of chords could be said to possess the *Markov property.*

Memory in music is often referenced by musicologists and music enthusiasts alike, just like memory in language and psychology. Think of those songs that, upon hearing for the first time, you have the ability to anticipate exactly their structure. To recognize components of a song, whether having heard it before or not, is something that everyone can do, either from the instrumentation, or the era (evidenced by quality of recording, at the very least), or the pitches being played. Clearly, there are a lot of patterns involved in the structure of music, and if we can quantify any of these patterns, we should.

Why digital filtering?

Everything in this thesis is tied to information theory, and one of the largest problems approached by the field is that of the noisy channel. "Noise," however, can be thought of in two ways: (1) background interference, like a "noisy" subway station, or a poor recording where the medium used to acquire sound contains a lot of noise, like a revolver or even a coaxial cable; and then, (2) undesired frequencies, like a C, C$\sharp$, and F all at once, when we just want to know if a C is being played. Both of these problems can be addressed by filtering in digital signal processing, and in a quick, faster-than-real-time fashion, can give us the results digitally that our ear can verify in analog.

Why entropy?

"Complex" and "difficult" are adjectives that many musicians and musicologists are hesitant to use [21], even though it seems readily applicable to many works in both music and the visual arts. I was provoked by this hesitance, as well as very interested in some way of quantifying the "difference" between two songs. Realizing that pattern matching in music was too chaotic a task to accomplish in a year (I would have to hold all sorts of aspects of music constant—pitch/tuning, rhythm, lyrics, instrumentation—and then, what if the song is in $\frac{3}{4}$ time?), I turned to the concept of entropy within information and coding theory for some way of analyzing the way a sequence of events behaves.

If you have heard of entropy, you probably came across it in some exposition on chaos theory, or perhaps even thermal physics. Qualitatively, it is the "tendency towards disorder" of a system. It is at the core of the second law of thermodynamics, which states that, in an isolated system, entropy will increase as long as the system is not in equilibrium, and at equilibrium, it approaches a maximum value. When using many programs or applications at once on your laptop, you have probably noticed that your computer gets hotter than usual. This is solely due to your machine working harder to ensure that an error does not occur, which has a higher chance of happening when there is more that can go wrong.

The entropy of a Markov chain boils down a song (or whatever sequence of data is being used) to the average certainty of each chord. If we didn't want the average certainty, we would simply use its probability mass function. But, in trying to compare songs to one another, I wanted to handle only individual numbers (gathered from the function of entropy) versus individual functions for each set or classification. As you will see, entropy is a fine measure not only of complexity, but of origination in music. It can tell us if a song borrows from a certain style or range of styles (since there are so many) by comparing their harmonic vocabularies of chords, and telling us just how an artist uses them. This is better than simply matching patterns, even of chords, because two songs can have similar pitches, but be classified completely differently from one another. Hence, entropy can prove a very interesting and explanatory measure within music.

Automatic Chord Recognition

Current methods for Automatic Chord Recognition have only reached about 75% efficiency [25], and I cannot say I am expecting to do any better. However, I do know how to part-write music harmonically, so I will have a control which I know to be correct to the best of my abilities.

Using several (12 times the number of octaves over which we wish to sample, plus one for the fundamental frequency) bandpass filters in parallel, we receive many passbands from a given sampling instant, all corresponding to a level of power [measured in watts (W)]. We sample the entire song, left to right, and document the relative powers of our 12 passbands at each instant (any fraction a second seems like more than enough to me), and choose chords based on the triple mentioned above: key, tonality, and root. The root is always present in a chord, and since we are normalizing pitch to its fundamental (i.e., octaves do not matter), inverted chords will be detected just as easily as their non-inverted counterparts. Thus, we match our 1-4 most powerful pitches to a previously defined chord pattern and label the chord with the appropriate Roman numeral, based on the key of the song or tune.

Smoothing

Too bad it is not that easy. The 1-4 most powerful pitches in the digital form rarely comprise a chord, so we have to smooth the functions of each pitch over time in order to see which ones are actually meaningful. We do this by averaging the power of a frequency over some range of samples, optimally equal to the minimum duration of any chord in the progression.

Western Convention in Music

How was harmony discovered? Arguably, that is like asking, “How was gravity discovered?” It was more *realized* than discovered. The story is that, one day in fifth century B.C.E., Pythagoras was struck by the sounds coming from blacksmiths’ hammers hitting anvils [22]. He wondered why occasionally two hammers would combine to form a melodious interval, and other times, would strike a discord. Upon investigation, he found that the hammers were in almost exact integer proportion of their relative weights: euphonious sounds came from hammers where one was twice as heavy as the other, or where one hammer was 1.5 times as heavy as the other. In fact, they were creating the intervals of an octave (12 half steps) and a perfect fifth (7 half steps).

His curiosity peaked, Pythagoras sat down at an instrument called the monochord and played perfect fifths continuously until he reached an octave of the first frequency he played. For example, if he began at A (27.5 Hz), he went up to E, then B, F$\sharp$, C$\sharp$, G$\sharp$, D$\sharp$, A$\sharp$, E$\sharp$ (F), B$\sharp$ (C), F$\sharp$ $\sharp$ (G), C$\sharp$ $\sharp$ (D), and finally G$\sharp$ $\sharp$, the “enharmonic equivalent” of A[1]. However, you may notice that $(1.5)^{12} = 129.746338 \neq 128 = 2^7$, because 12 perfect fifths in succession spans 7 octaves. Pythagoras’ estimation of 1.5 thus had a 1.36% error from the true value of the difference in frequency between a note and the note 7 half steps above it, which is $2^{(7/12)}$.

Sound is not the only wave motion to which harmony applies: the light from a prism also has proportional spacings of color. Sir Isaac Newton even devised a scheme coordinating audible frequencies with the visible spectrum in something he called a spectrum-scale, depicted below, matching colors to musical tones.

The piano was designed in 1700 based on acoustical techniques applied to the harpsichord, which was most likely invented in the late Middle Ages (16th century). The lowest A is 27.5000 Hz, and since we double the initial frequency of a note to find one octave higher, each A above it is an integer. The piano produces sound by striking a key and thereby dropping a hammer onto a wire (which behaves like a string), which causes it to vibrate. The speed v of propagation of a wave in a string is proportional to the square root of the tension of the string T and inversely proportional to the square root of the linear mass m of the string:

$$v = \sqrt{\frac{T}{m}}$$

The frequency f of the sound produced by this wave only requires knowledge of the length L of the string and the speed of propagation, and is

$$f = \frac{v}{2L}$$

[1]In music theory, the difference between these two notes is larger than you might expect

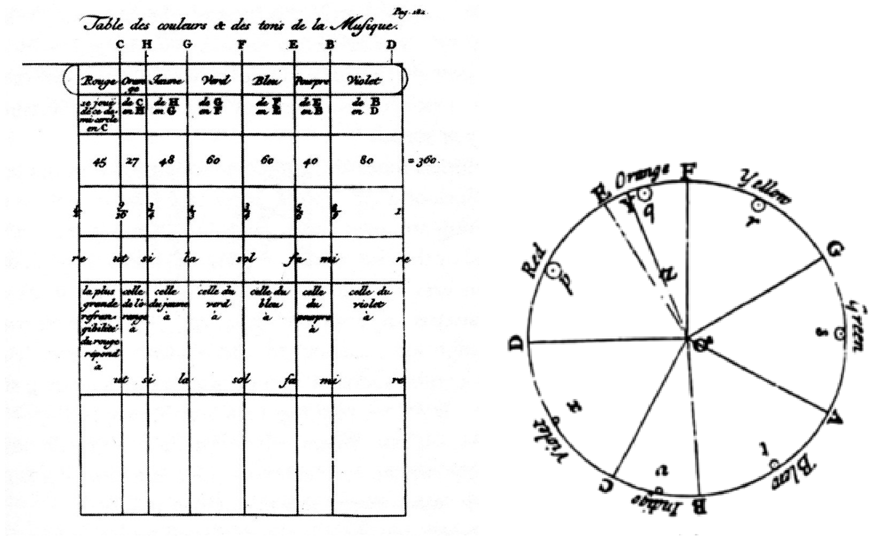


Figure 1: Two representations of Newton's "spectrum-scale," matching pitches to colors. The circular one is highly reminiscent of depictions of the circle of fifths. Note that the primary colors (red, yellow, blue) are the first, third, and fifth colors of the seven on the color wheel (RoYgBiv), just as a chord is constructed of a first, third, and fifth of seven scale degrees in a key. Both images reproduced from Voltaire's *Elémens de la philosophie de Neuton* (1738) in [22].

The wavelength λ of the fundamental harmonic is simply

$$\lambda = 2L.$$

Therefore,

$$v = \lambda f.$$

Hence, shortening the string, increasing the tension of the string, or decreasing the mass of the string all make for a higher fundamental frequency.

In Western music, the piano is archetypical. It is a linear system where notes to the left have lower frequencies (and longer wires) and to the right, higher. The common chromatic notation of music where we have a treble and bass clef with middle-C written the same way, at the same height, is natural to the keyboard and its linear nature, because the lowest note is indeed the leftmost note on the keyboard and the highest note is the rightmost note on the keyboard. It is also nice because the bass clef usually designates those notes played by the left hand, and the treble clef those played by the right. The notation adds another dimension, that of time, to make the system a sort of plane with lines of melody ("voices") according to pitch.

Now, a half-step (A to A♯, for example) multiplies the initial frequency (the lower one) by $2^{1/12}$. It is clear that an octave, which is equivalent to 12 half-steps, multiplies the initial frequency by 2. This distribution of frequencies on the piano is known as "equal temperament," which has a ring of political incorrectness, since many Eastern cultures use quarter-tones (simply half of a half-step). But we attach a positive meaning to "harmony," and indeed the frequencies that occur from a note's harmonic overtone series are considered "pleasant," and shape much of Western convention in songwriting.

For more about the notation techniques and harmonic part-writing used in this thesis, please see Appendix A.

Procedure

This project requires evidence from a vast range of mathematics, acoustics, and music theory, so I try to develop each relevant discipline as narrowly as possible.

1. We import songs into a series of bandpass filters in parallel, and pick out the most intense frequencies by analyzing their power.

2. We count up the number of times we transition between chord c_i and chord c_j for all chords c_i and c_j s.t. $c_i \neq c_j$ (or simply, $i \neq j$) in a progression X with state space C.

3. We divide these counts by the total number of times we transition from the initial chord c_i and obtain a probability distribution. We insert this as a row into a Markovian transition matrix representing a Markov chain, where the chain is a chord progression, in which each row of the matrix sums to 1 and the diagonal entries are 0, since we are not paying mind to rhythm and therefore cannot account for the duration of states.

4. We take the entropy of the Markov chain for each (set of) progression(s) using the measure $\sum_i p_i \sum_j p_{j|i} \log_N(1/p_{j|i})$, where p_i is the probability of hearing chord c_i, N is the number of distinct states in C, and $p_{j|i}$ is the probability of transitioning to chord c_j from chord c_i.

5. We compare the levels of entropy against each other and see what the measure means in terms of the strictness of the musical classification of the set of chords.

The first of these procedures, i.e. filtering and digital signal processing, is developed in the first chapter. The second and third are discussed in Chapter 2 on Markovian probability. The final two are found in the third chapter on entropy and information theory, where the conclusions and some code can be found.[2]

[2]Because so many of aspects of this thesis come from very distinct fields of mathematics, physics, and engineering, I would not be surprised to hear that I go into far too much detail trying to describe the elementary nature of each of the fields. However, I wanted to make sure that this document was "compact," and contained all definitions that one might need to reference. For those that I didn't state immediately after their introduction, they can (hopefully) be found in the glossary.

Chapter 1

Automatic Recognition of Digital Audio Signals

1.1 Digital Audio Signal Processing

Digital signal processing (DSP) originated from Jean Baptiste Joseph, Baron de Fourier in his 1822 study of thermal physics, *Théorie analytique de la chaleur.* There, he "evolved" [10] the Fourier series, applying to periodic signals, and the Fourier transform, applying to aperiodic, or non-repetitive, signals. The discrete Fourier transform became popular in the 1940s and 1950s, but was difficult to use without the assistance of a computer because of the huge amount of computations involved. James Cooley and John Tukey published the article "An algorithm for the machine computation of complex Fourier series" in 1965, and thereby invented the fast Fourier Transform, which reduced the number of computations in the discrete version from $O(n^2)$ to $O(n \log n)$ by a recursive algorithm, since roots of unity in the Fourier transform behave recursively. Oppenheimer and Schafer's *Digital Signal Processing* and Rabiner and Gold's *Theory and Application of Digital Signal Processing* remain the authoritative texts on digital signal processing since their publication in 1975, though the highly technical style keeps them fairly inaccessible to the non-electrical engineer. Also, the Hungarian John von Neumann's architecture of computers from 1946 was the standard for more than 40 years because of two main premises: (1) there does not exist an "intrinsic difference" between instructions and data [10], and (2) instructions can be partitioned into two major fields containing an operation and data upon which to operate, creating a single memory space for both instructions and data.

Essentially, DSP's goal is to maximize the signal-to-noise ratio, and it does so with **filters** like the discrete Fourier transform, bandpass filters, and many others. Digital media is always discrete, unlike its analog predecessor, but the digital form has its advantages. For one, a CD or vinyl will deteriorate over time, or become soiled and scratched, and this does not happen to digital files. Second, it is easy and fast to replicate a digital version and store it in many places, again increasing the likelihood of maintaining its original form. Finally, with analog's "infinite sampling rate" [10] comes infinite variability, and this also contributes to digital media's robustness over analog.

To sample a signal, we take a discrete impulse function, element-wise multiply it (i.e., take the dyadic product of it) with an analog signal, and retrieve a sampling of that signal with which we can do many things. The construction of a filter essentially lies in the choice of coefficients; we will go through a proof of our choice in coefficients to show that the filtering works.

1.1.1 Sampling

There are three different ways of sampling an analog signal: *ideal*, *instantaneous*, and *natural* [14]. The simplest of these and the one which we will use is the ideal sampling method, which consists of a sequence of uniformly spaced impulses over time with amplitude equivalent to the sampled signal at a given time. An **impulse** is a vertical segment with zero width and infinite amplitude, extending from $y = 0$ to $y = \infty$. It is dotted with the analog signal to heed a **discrete-time signal** just described.

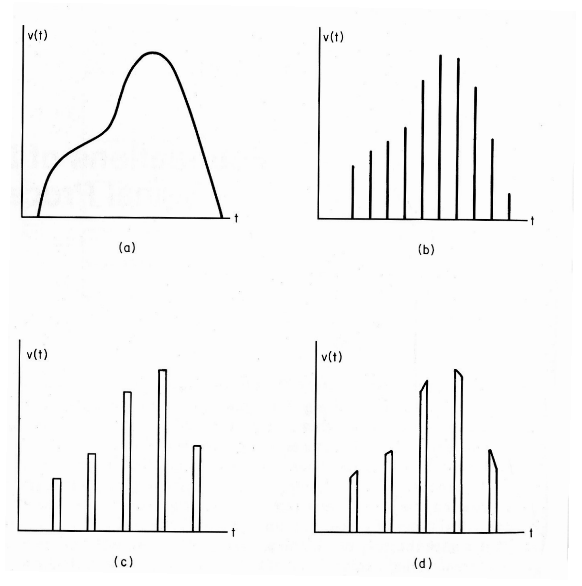


Figure 1.1: Three different types of digital sampling for an analog signal (a): (b) ideal, (c) instantaneous, and (d) natural [14].

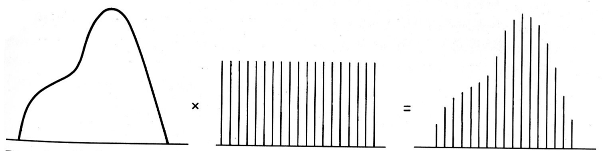


Figure 1.2: The dyadic product of an impulse function with a signal in ideal sampling [14].

Formally, this is given by

$$\begin{aligned} x_s(t) &= x(t) \cdot (\delta(t-\infty) + \ldots + \delta(t-T) + \delta(t) + \delta(t+T) + \ldots + \delta(t+\infty)) \\ &= x(t) \sum_{n=-\infty}^{\infty} \delta(t-nT) \end{aligned}$$

where $x_s(t)$ is the sampled (discrete-time) signal, δ is the impulse function, and T is the period of $x(t)$, i.e., the spacing between impulses. We compute the **power** p at a point in time t

$$p(t) = |x(t)|^2$$

and the **energy** of a system, or entire signal

$$E = \sum_{t=-\infty}^{\infty} |x(t)|^2.$$

Energy is measured in joules, where 1 joule is 1 watt $\times$ 1 second. The average power P is then

$$P = \lim_{T \to \infty} \frac{1}{2T} \sum_{t=-T}^{T} |x(t)|^2.$$

A signal has a minimum frequency f_L (it is fine to assume that this is 0 Hz, in most applications) and a maximum frequency f_U. If a signal is undersampled, i.e., the sampling frequency is $f_s < 2 \cdot f_U$, the impulses will not be spaced far enough apart, and the resulting spectrum will have overlaps. This is known as **aliasing**, and when two spectra overlap, they are said to **alias** with each other. We have a theorem that forebodes the problems that occur from undersampling.

The Nyquist-Shannon Sampling Theorem. If a signal $x(t)$ contains no frequencies greater than f_U cycles per second (Hz), then it is completely determined by a series of points spaced no more than $\frac{1}{2f_U}$ seconds apart, i.e., the sampling frequency $f_s \geq 2f_U$. We reconstruct $x(t)$ with the function

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT) \frac{\sin[2f_s(t-nT)]}{2f_s(t-nT)}.$$

The minimum sampling frequency is often referred to as the **Nyquist frequency** or **Nyquist limit**, and this rate of $\frac{1}{2f_U}$ is referred to as the **Nyquist rate**. The **bandwidth** of a signal is its highest frequency component, f_U.

Aliasing is depicted in the following sequence of images, all taken from [7]. The construction of the impulse function is crucial to avoiding aliasing.

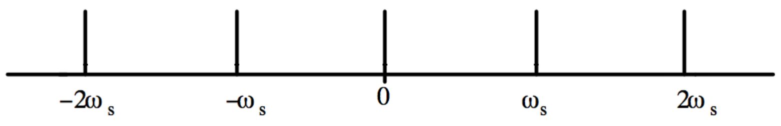


Figure 1.3: The impulse function δ for sampling a signal at ω_s.

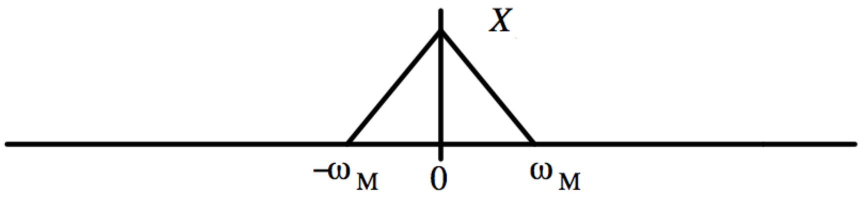


Figure 1.4: The spectrum $X(\omega_M)$ of the signal $x(t)$, where ω_M is the maximum frequency of $x(t)$.

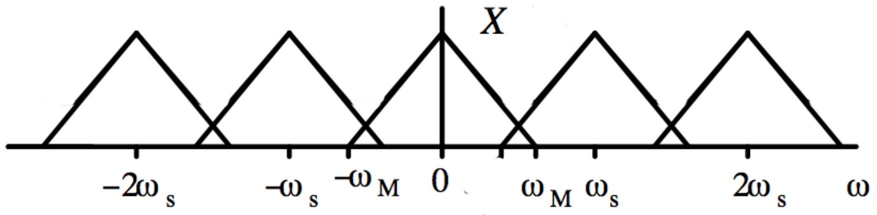


Figure 1.5: The result of choosing a sampling rate $\omega_s < 2\omega_M$.

We are already witnessing some characteristics of the important operation of **convolution**, denoted by $*$: a spectrum convolved with an impulse function is actually the (discrete) Fourier transform ($\mathcal{F}$) of the discrete-time signal, i.e.,

$$\mathcal{F}\left[x(t)\sum_{n=-\infty}^{\infty}\delta(t-nT)\right] = \mathcal{F}(x(t)) * \left[f_s\sum_{m=-\infty}^{\infty}\delta(t-mf_s)\right],$$

where $\mathcal{F}$ denotes the Fourier transform [14]. Note also that a spectrum of a signal is given by its Fourier transform, $\mathcal{F}(x(t))$.

The operation of convolution is commutative, additive, and distributive, but does not have a multiplicative inverse. However, in the delta distribution, which is such that

$$\begin{aligned}\int_{-\infty}^{\infty}\delta(t-t_0)f(t)dt &= f(t_0),\\ \delta(t-m)*\delta(t-n) &= \delta(t-(m+n)),\end{aligned}$$

the convolution of a function with δ returns the function, i.e.,

$$f * \delta = f.$$

1.1.2 The Discrete Fourier Transform

The Discrete Fourier Transform (DFT) and its inverse (the IDFT) are used on aperiodic signals to establish which peak frequencies are periodic (i.e., are overtones[1]), and those that are aperiodic. The DFT represents the spectrum of a signal, and the IDFT reconstructs the signal (with a phase shift) and retrieves only those frequencies that are fundamental. It is a heavy but simple algorithm with many variables, so it is best to approach it slowly to truly understand its mechanisms[2]. It was born from the Fourier series in Fourier analysis, and it attempts to approximate the abstruse waves of a spectrogram by simpler trigonometric piecewise functions, for the behavior of a frequency can be modeled by sinusoidal functions. This helps clarify what is noise in a signal and what is information by reducing a signal to a sufficiently large, finite number of its fundamental frequencies in a given finite segment of the signal.

Definition: Fourier transform. The **Fourier transform** is an invertible linear transformation

$$\mathcal{F} : \mathbb{C}^N \to \mathbb{C}^N$$

where $\mathbb{C}$ denotes the complex numbers. Hence, it is complete. The Fourier transform of a **continuous-time signal** $x(t)$ is represented by the integral [34]

$$\begin{aligned} X(\omega) &= \int_{-\infty}^{\infty} x(t)e^{-i\omega t}dt, \qquad \text{and inversely,} \\ x(t) &= \int_{-\infty}^{\infty} X(\omega)e^{i\omega t}d\omega, \quad \omega \in \mathbb{R}, \end{aligned}$$

where $X(\omega)$ is the **spectrum** of x at frequency ω.

It is not difficult to see why the discrete-time signal case is then represented by

$$X(\omega_k) = \sum_{n=0}^{N-1} x(t_n)e^{-i\omega_k t_n}, \qquad k = 0, 1, 2, \ldots, N-1,$$

which, because $\omega_k = 2\pi k/(NT)$ and $t_n = nT$, can also be written

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-i2\pi kn/N}, \qquad k = 0, 1, 2, \ldots, N-1,$$

and the inverse DFT, or IDFT, is

$$x(t_n) = \frac{1}{N}\sum_{k=0}^{N-1} X(\omega_k)e^{i\omega_k t_n}, \quad n = 0, 1, 2, \ldots, N-1,$$

[1]The overtone series of a fundamental frequency is the sequence of frequencies resulting from multiplying the fundamental frequency by each of the natural numbers (i.e., $\{1 \cdot f_{fund}, 2 \cdot f_{fund}, \ldots\}$). The overtone series of A=55 Hz is {55 Hz = A, 110 Hz = A, 165 Hz = E, 220 Hz=A, 275 Hz=C $\sharp$, 330 Hz = E, 385 Hz=G, 440 Hz=A, 495 Hz=B $\flat$, ...}.

[2]The book [34] is a very good resource for those new to the Fourier transform.

which similarly can be rewritten

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{i2\pi kn/N}, \qquad n = 0, 1, 2, \ldots, N-1.$$

List of symbols in the DFT.

$:=$	$:=$	"defined as"
$x(t)$	$:=$	the amplitude (real or complex) of the input signal at time t (seconds)
T	$:=$	the sampling interval, or period, of $x(t)$
t	$:=$	$t \cdot T =$ sampling instant, $t \in \mathbb{N}$
$X(\omega_k)$	$:=$	spectrum of x at frequency ω_k
ω_k	$:=$	$k\Omega = k^{th}$frequency sample
Ω	$:=$	$\dfrac{2\pi}{NT} =$ radian-frequency sampling interval (radians/sec)
ω	$:=$	$2\pi f_s$
f_s	$:=$	$1/T = \dfrac{\omega}{2\pi} =$ the sampling rate, in hertz (Hz)
N	$:=$	the number of time samples = the number of frequency samples $\in \mathbb{Z}^+$

The signal $x(t)$ is called a **time domain** function and the corresponding spectrum $X(k)$ is called the signal's **frequency domain** representation. The **amplitude** A of the DFT X is given by

$$A(k) = |X(k)| = \sqrt{\Re(X(k))^2 + \Im(X(k))^2},$$

and its **phase** ϕ is

$$\phi(k) = 2\arctan\frac{\Im(X(k))}{|X(k)| + \Re(X(k))}$$

This trigonometric function finds the angle in radians between the positive x-axis and the coordinate $(\Im(X(k)), \Re(X(k)))$. It is positive and increasing from $(0, \pi)$ (counter-clockwise), and negative and decreasing from (0,-π) (clockwise). It equals π at π.

You may notice that the exponentials in both the DFT and its inverse resemble **roots of unity**, of which Euler's identity will be helpful in explaining.

Euler's identity. For any real number x,

$$\begin{aligned} e^{ix} &= \cos x + i\sin x, \qquad \text{and} \\ e^{-ix} &= \cos x - i\sin x. \end{aligned}$$

Definition: Roots of unity. We call the set $W = \{W_N^0, W_N^1, \ldots, W_N^{N-1}\}$ the N^{th} roots of unity corresponding to points on the unit circle in the complex plane, where

$$\begin{aligned} W_N &= e^{\frac{2\pi i}{N}}, && \text{the primitive } N^{th} \text{ root of unity;} \\ W_N^k &= e^{\frac{2\pi ik}{N}} = (W_N)^k, && \text{the } k^{th}\ N^{th} \text{ root of unity;} \\ W_N^N &= W_N^0. \end{aligned}$$

By Euler's identity,

$$W_N^{kn} = e^{\frac{2\pi ikn}{N}} = \cos(2\pi kn/N) + i\sin(2\pi kn/N).$$

This is called the "kth sinusoidal function of f" [34].

By Euler's identity, $e^{-i2\pi kt/N} = \cos(2\pi kt/N) - i\sin(2\pi kt/N)$ and $e^{i2\pi kt/N} = \cos(2\pi kt/N) + i\sin(2\pi kt/N)$. Therefore, we can also write the DFT and its inverse as follows:

$$\begin{aligned} X(k) &= \sum_{t=0}^{N-1} x(t)e^{-i2\pi kt/N}, \qquad k = 0, 1, \ldots, N-1 \\ &= \sum_{t=0}^{N-1} x(t)\cos(2\pi kt/N) - i\sum_{t=0}^{N-1} x(t)\sin(2\pi kt/N) \\ x(t) &= \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{i2\pi kt/N}, \qquad t = 0, 1, \ldots, N-1 \\ &= \frac{1}{N}\sum_{k=0}^{N-1} X(k)\cos(2\pi kt/N) + \frac{i}{N}\sum_{k=0}^{N-1} X(k)\sin(2\pi kt/N). \end{aligned}$$

Then,

$$e^{-i2\pi kt/N} = \cos(2\pi kt/N) - i\sin(2\pi kt/N)$$

is the *kernel* of the discrete Fourier transform.

For the sake of clarity, I will show that the IDFT is indeed the inverse of the DFT. To do so, all we need to note is the boundedness of k in either of the sums:

$$\begin{aligned} X(k) &= \sum_{t=0}^{N-1} x(t)e^{\frac{-2\pi itk}{N}} \\ &= \sum_{t=0}^{N-1}\left(\frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{\frac{2\pi itk}{N}}\right)e^{\frac{-2\pi itk}{N}} \end{aligned}$$

Since the k in $e^{\frac{-2\pi itk}{N}}$ is not bounded by the k in the inner sum, we change the inside

k to l:

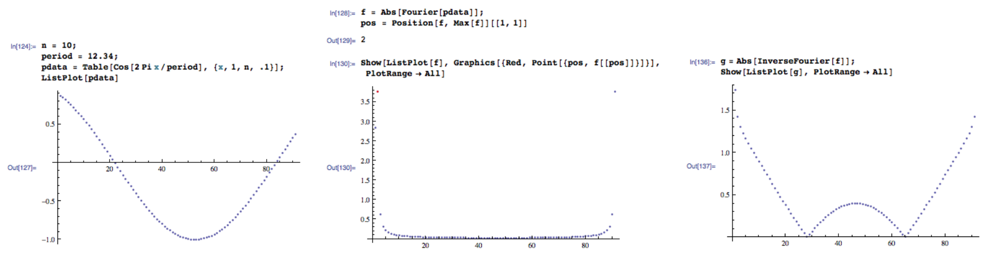
$$
\begin{aligned}
&= \sum_{t=0}^{N-1}\left(\frac{1}{N}\sum_{l=0}^{N-1} X(l)e^{\frac{2\pi itl}{N}} e^{\frac{-2\pi itk}{N}}\right)\\
&= \sum_{t=0}^{N-1}\frac{1}{N}\sum_{l=0}^{N-1} X(l)e^{\frac{2\pi it(l-k)}{N}}\\
&= \frac{1}{N}\sum_{l=0}^{N-1} X(l)\sum_{t=0}^{N-1} e^{\frac{2\pi it(l-k)}{N}}\\
&= N \quad \text{when } l=k,\ 0 \text{ otherwise}
\end{aligned}
$$

Thus, our double sum is

$$\frac{1}{N}\sum_{l=0}^{N-1} X(l)N\delta_{lk} = X(k),$$

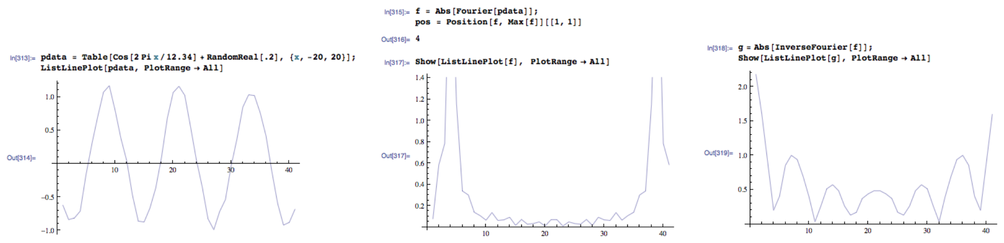
where $\delta_{lk} = 1$ for $l = k$ and 0 otherwise.

To help you visualize how the DFT and IDFT manipulate a signal, four examples of signals are shown in Figures 1.6-1.9: two are periodic, two aperiodic, two noiseless, and two containing noise. All transforms are absolute values; note the phase shift in the IDFT reconstruction when the signal contains negative values.



```
In[124]:= n = 10;
          period = 12.34;
          pdata = Table[Cos[2 Pi x / period], {x, 1, n, .1}];
          ListPlot[pdata]
Out[127]=

In[128]:= f = Abs[Fourier[pdata]];
          pos = Position[f, Max[f]][[1, 1]]
Out[129]= 2
In[130]:= Show[ListPlot[f], Graphics[{Red, Point[{pos, f[[pos]]}]}],
           PlotRange → All]
Out[130]=

In[136]:= g = Abs[InverseFourier[f]];
          Show[ListPlot[g], PlotRange → All]
Out[137]=
```

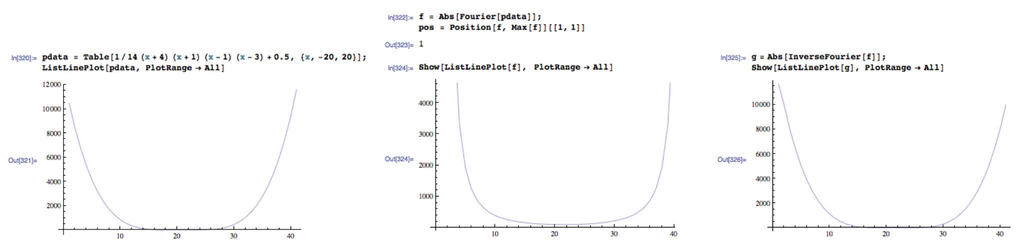
Figure 1.6: Noiseless periodic signal, its DFT, and its IDFT.



```
In[313]:= pdata = Table[Cos[2 Pi x / 12.34] + RandomReal[.2], {x, -20, 20}];
          ListLinePlot[pdata, PlotRange → All]
Out[314]=

In[315]:= f = Abs[Fourier[pdata]];
          pos = Position[f, Max[f]][[1, 1]]
Out[316]= 4
In[317]:= Show[ListLinePlot[f], PlotRange → All]
Out[317]=

In[318]:= g = Abs[InverseFourier[f]];
          Show[ListLinePlot[g], PlotRange → All]
Out[319]=
```

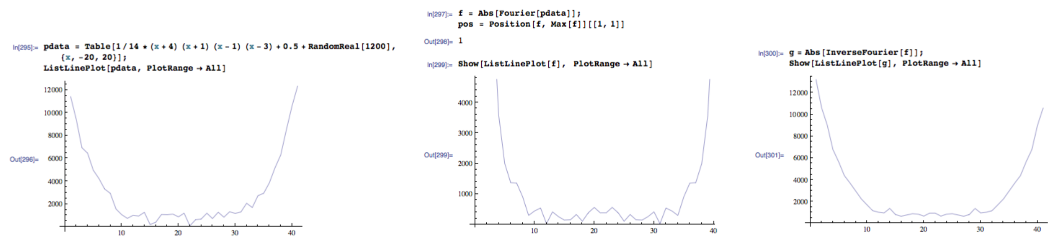
Figure 1.7: Noisy periodic signal, its DFT, and its IDFT.

Note that the transforms are symmetric about the middle term, $k = t = N/2$, the Nyquist frequency. This happens because the signals are real-valued. For complex-valued signals, the Nyquist frequency is always zero.



```
In[320]:= pdata = Table[1 / 14 (x + 4) (x + 1) (x - 1) (x - 3) + 0.5, {x, -20, 20}];
          ListLinePlot[pdata, PlotRange → All]
Out[321]=

In[322]:= f = Abs[Fourier[pdata]];
          pos = Position[f, Max[f]][[1, 1]]
Out[323]= 1
In[324]:= Show[ListLinePlot[f], PlotRange → All]
Out[324]=

In[325]:= g = Abs[InverseFourier[f]];
          Show[ListLinePlot[g], PlotRange → All]
Out[326]=
```

Figure 1.8: Noiseless aperiodic signal, its DFT, and its IDFT.



```
In[295]:= pdata = Table[1 / 14 * (x + 4) (x + 1) (x - 1) (x - 3) + 0.5 + RandomReal[1200],
             {x, -20, 20}];
          ListLinePlot[pdata, PlotRange → All]
Out[296]=

In[297]:= f = Abs[Fourier[pdata]];
          pos = Position[f, Max[f]][[1, 1]]
Out[298]= 1
In[299]:= Show[ListLinePlot[f], PlotRange → All]
Out[299]=

In[300]:= g = Abs[InverseFourier[f]];
          Show[ListLinePlot[g], PlotRange → All]
Out[301]=
```

Figure 1.9: Noisy aperiodic signal, its DFT, and its IDFT.

1.1.3 Properties of the Transform

The nature of complex numbers provides the Fourier transform (both the discrete and continuous versions) with many nice properties. We will only have time to discuss those relevant to its function as a filter, and those include **convolution** and **linearity**.

Cyclic Convolution. The **cyclic convolution** of two signals each of length N is equal to the inverse discrete Fourier transform of the dyadic product of the discrete Fourier transform of each signal. The operation of convolution is notated by the symbol $*$.

Proof. Let $\mathcal{F}$ denote the discrete Fourier transform operator. For two signals $x(t_1)$ and $y(t_2)$,

$$\begin{aligned}
\mathcal{F}(x(t)) &= X(k) = \sum_{t=0}^{N-1} x(t_1) e^{-2\pi i k t_1/N}, \\
\mathcal{F}(y(t')) &= Y(k) = \sum_{t'=0}^{N-1} y(t_2) e^{-2\pi i k t_2/N}.
\end{aligned}$$

The inverse discrete Fourier transform of their dyadic (element-wise) multiplication is

$$\begin{aligned}
\mathcal{F}^{-1}(\mathcal{F}(x(t_1)) \cdot \mathcal{F}(y(t_2))) &= \frac{1}{N}\sum_{k=0}^{N-1} X(k)Y(k)e^{2\pi itk/N} \\
&= \frac{1}{N}\sum_{k=0}^{N-1}\left(\sum_{t_1=0}^{N-1} x(t_1)e^{-2\pi ikt_1/N}\right)\left(\sum_{t_2=0}^{N-1} y(t_2)e^{-2\pi ikt_2/N}\right)e^{i2\pi kt/N} \\
&= \frac{1}{N}\sum_{t_1=0}^{N-1}\sum_{t_2=0}^{N-1} x(t_1)y(t_2)\sum_{k=0}^{N-1} e^{2\pi ik(t-(t_1+t_2))/N}.
\end{aligned}$$

Summing over k, we get

$$\sum_{k=0}^{N-1} e^{2\pi ik(t-(t_1+t_2))/N} = N\delta,$$

where δ is 1 when $n \equiv (t_1 + t_2) \bmod N$ and 0 otherwise. Therefore, the above triple sum becomes

$$\begin{aligned}
\sum_{t_1+t_2 \equiv t \bmod N} x(t_1)y(t_2) &= \sum_{t_1=0}^{N-1} x(t_1)y(t-t_1) \\
&= \sum_{t_2=0}^{N-1} x(t-t_2)y(t_2) \\
&= x * y(t),
\end{aligned}$$

which proves the theorem. □

Corollary.

$$\mathcal{F}(x(t) \cdot y(t)) = \mathcal{F}(x(t)) * \mathcal{F}(y(t)).$$

This is how we obtain the aforementioned result

$$\mathcal{F}\left[x(t)\sum_{n=-\infty}^{\infty}\delta(t-nT)\right] = \mathcal{F}(x(t)) * \left[f_s\sum_{m=-\infty}^{\infty}\delta(t-mf_s)\right].$$

Another important property of the Fourier transform is its **linearity**.

Linearity of the DFT. A signal x can be scaled by a constant a such that $\mathcal{F}(a \cdot x) = a \cdot X$.

Proof.

$$\begin{aligned}
ax(t) &= \frac{a}{N}\sum_{t=0}^{N-1} X(f)e^{i2\pi ft/N} \\
&= aX(f).
\end{aligned}$$

□

Also, the sum of two signals equals the sum of their transforms, i.e., $a \cdot x + b \cdot y = a \cdot X + b \cdot Y$, a and b constants.

Proof.

$$\begin{aligned} ax(t) + by(t) &= \frac{a}{N}\sum_{t=0}^{N-1} X(f)e^{i2\pi ft/N} + \frac{b}{N}\sum_{t=0}^{N-1} Y(f)e^{i2\pi ft/N} \\ &= aX(f) + bY(f). \end{aligned}$$

□

The DFT is called a **linear filter** because of these properties, and it is time-invariant or time-homogeneous, meaning that over time, it does not change.

One final interesting property of the DFT that is likely relevant to filtering, though I have not come across a literal relevance, is in **Parseval's theorem**, which states that

$$\sum_{t=0}^{N-1} |x(t)|^2 = \frac{1}{N}\sum_{k=0}^{N-1} |X(k)|^2,$$

or, the DFT is unitary. A more general form of Parseval's theorem is Plancherel's theorem [33].

The Fourier transform can be used to detect instrumentation, because each instrument (including each person's voice) has a unique **timbre**. Timbre means "tone color," and is simply the word we use to distinguish a note played by a violin from the same note played by a piano. Timbre affects the amplitudes of a frequency's **overtone series**, which is the sequence of frequencies found by multiplying the fundamental frequency by each of the natural numbers (i.e., $\{1 \cdot f_{fund}, 2 \cdot f_{fund}, \ldots\}$). So, in this way, the human ear is foremost a Fourier device: it can distinguish between instruments, just as it can other humans' voices. However, computers give us the details about what it is that makes the quality of voices differentiable. Although this is fascinating, we will not be paying mind to timbre in this project, which I am sure will prove problematic in trying to separate the (background) chord progression from the (foreground) melody.

1.1.4 Filtering

We use a filter when we want to retrieve a single frequency or set of frequencies from a signal, and decimate the remainder. Decimation via digital filtering is called **attenuation**. One way to think of a digital filter is as a Kronecker delta function. It is a spectrum with amplitude 1 at the frequencies it is designed to maintain in a signal, and amplitude 0 at the frequencies it is designed to kill. But because it is a continuous function, it contains amplitudes in between 0 and 1, and where the amplitude is 0.5 is called a **cutoff frequency**. There can be only zero, one, or two cutoff frequencies in the filters described below.

In working with digital signals, we use Finite Impulse Response (FIR) techniques [14]. An important concept in digital filtering is the **impulse response** $h(t)$, which is a measure of how the outputted signal responds to a unit impulse from the impulse function δ. A **unit impulse** is given by $\delta(t)$, the continuous case of familiar Kronecker delta function. Both a filter and a signal have an impulse function. We can integrate or sum over the impulse response to obtain the step response $a(t)$, and use the **Laplace transform**, a definition of which can be found in the glossary but is beyond the scope of this thesis, to obtain the **transfer function** $H(s)$, where $s = \frac{1-z}{1+z}$, z from the z-**transform**, also beyond the scope of this thesis but defined in the glossary.

The general form for a linear time-invariant FIR system's output y at time t is given by

$$\begin{aligned} y(t) &= \sum_{\tau=0}^{N-1} h(\tau)x(t-\tau) \\ &= \sum_{T=0}^{N-1} h(t-T)x(T) \end{aligned}$$

by a change of variables, $T = t - \tau$.

Filters are specified by their **frequency response** $H(f)$, which is the discrete Fourier transform of the impulse response $h(t)$.

$$H(f) = \sum_{t=0}^{N-1} h(t)e^{-i2\pi tf/N}, \qquad f = 0, 1, \ldots, N-1.$$

A filter modifies the frequencies within the spectrum (DFT) of a signal. When we want to rid a signal of frequencies below some cutoff (or critical) frequency f_c, such that we are left with only frequencies above f_c, we design a **highpass filter**. Its impulse response is defined for a user-defined value N equal to the number of **taps**, or the number of frequencies, at which the response is to be evaluated, and is given by

$$h(t) = h(N-1-t) = -\frac{\sin(m\lambda_c)}{m\pi}$$

where $m = t - \frac{N-1}{2}$ and $\lambda_c = \omega_c T = 2\pi f_c T$. If N is odd, then we compute

$$h\left(\frac{N-1}{2}\right) = 1 - \frac{\lambda_c}{\pi}.$$

Like the discrete Fourier transform and its inverse, the impulse response is symmetric about $t = \frac{N}{2}$ for N even or about $t = \frac{N-1}{2}$ for N odd, and therefore half-redundant. But since audio signals are real-valued, the Nyquist frequency is not necessarily 0.

Now, for a **lowpass filter** in which only low frequencies, or those below some f_c, may "pass through," our impulse response function is

$$h(t) = h(N-1-t) = \frac{\sin(m\lambda_c)}{m\pi},$$

where again $m = t - \frac{N-1}{2}$ and $\lambda_c = 2\pi f_c T$. If N is odd, then

$$h\left(\frac{N-1}{2}\right) = \frac{\lambda_c}{\pi}.$$

Putting the two together is one way of designing a **bandpass filter**, whose **passband** (the resulting bandwidth of any kind of "pass" filter) is restricted by a lower critical frequency, f_l, and an upper critical frequency, f_u. If $f_l = f_L$, the minimum frequency in the signal, and $f_u = f_U$, the maximum frequency in the signal, then the filter is an **all pass filter**. Its impulse response function is

$$h(t) = h(N-1-t) = \frac{1}{m\pi}[\sin(m\lambda_u) - \sin(m\lambda_l)]$$

and once again $m = t - \frac{N-1}{2}$ and $\lambda_u = 2\pi f_u T, \lambda_l = 2\pi f_l T$. If N is odd, then we have

$$h\left(\frac{N-1}{2}\right) = \frac{\lambda_u - \lambda_l}{\pi}.$$

The opposite of a bandpass filter is a **bandstop filter** or **notch filter**. Its impulse response is given by

$$h(t) = h(N-1-t) = \frac{1}{t\pi}[\sin(m\lambda_l) - \sin(m\lambda_u)].$$

For odd N,

$$h\left(\frac{N-1}{2}\right) = 1 + \frac{\lambda_l - \lambda_u}{\pi}.$$

Again, m, λ_l, and λ_u are all defined as above.

All of these impulse response functions output the coefficients for an N-tap filter of their type, and the IDFT gives the corresponding discrete-time impulse response [14].

1.2 A Tunable Bandpass Filter in Mathematica

1.2.1 Design

A bandpass filter takes an input of a frequency f, quality Q, and signal x, and outputs a signal with bandwidth centered at f. It does not matter whether f exists in the spectrum, but if it does, the bandpass filter gets excited, and shows that the frequency contributes to the amplitude (power) of the signal. The value Q is positive and real, and a high Q (¿20) means a sharp focus around this "resonant" frequency f by defining the sinusoid to have a dramatic and steep peak centered at f, where a low Q means a gentler-sloped sinusoid is chosen for the filter. In essence, Q determines how quickly the impulse response of a filter goes from 0 to 1, and/or 1 to 0.

When dealing with a digital signal, we must **tune** our desired frequency to take the sampling frequency into account. We do this by scaling f by $\frac{2\pi}{f_s}$, where f_s is the sampling frequency. We name the tuned frequency $\theta_f = \frac{2\pi}{f_s} \cdot f$.

To show that the coefficients

$$\begin{aligned}
\beta &= \frac{1}{2}\left(\frac{1-\tan\left(\frac{\theta_f}{2Q}\right)}{1+\tan\left(\frac{\theta_f}{2Q}\right)}\right), \\
\alpha &= \frac{1}{2}\left(\frac{1}{2}-\beta\right), \\
\gamma &= \left(\frac{1}{2}+\beta\right)\cos(\theta_f)
\end{aligned}$$

in the recursion

$$y(t) = 2\left(\alpha\left(x(t)-x(t-2)\right)+\gamma y(t-1)-\beta y(t-2)\right)$$

will define such a filter, let us prove them using the *Ansatz* ("onset") solution method, where we will let $A(\theta_f)$ designate our educated guess.

Proof. Since we know that $x(t)$ and likewise its filtered signal $y(t)$ are sinusoidal functions, let

$$\begin{aligned}
x(t) &= e^{i\omega t}, \\
y(t) &= A(\omega)x(t).
\end{aligned}$$

Then we can solve for A by

$$\begin{aligned}
A(\omega)e^{i\omega t} &= 2\left(\alpha\left(e^{i\omega t}-e^{i\omega(t-2)}\right)+\gamma\left(A(\omega)e^{i\omega(t-1)}\right)-\beta\left(A(\omega)e^{i\omega(t-2)}\right)\right) \\
\frac{A(\omega)}{2} &= \alpha(1-e^{-2i\omega})+\gamma A(\omega)e^{-i\omega}-\beta A(\omega)e^{-2i\omega} \\
&= \alpha-\alpha e^{-2i\omega}+\gamma A(\omega)e^{-i\omega}-\beta A(\omega)e^{-2i\omega}.
\end{aligned}$$

Therefore,

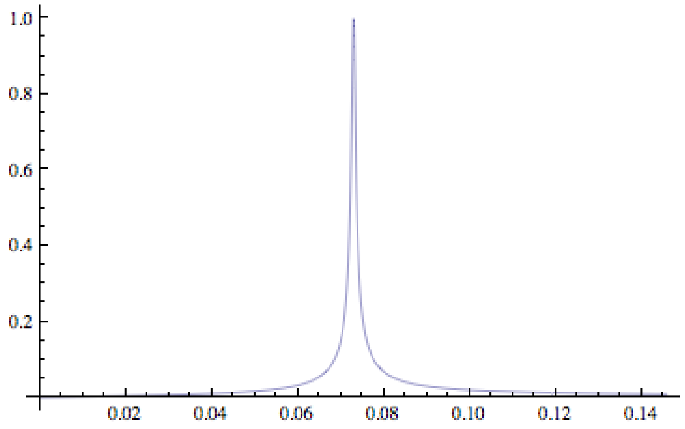
$$A(\omega)\cdot\left(\frac{1}{2}-\gamma e^{-i\omega}+\beta e^{-2i\omega}\right) = \alpha-\alpha e^{-2i\omega}$$

making A the ratio

$$A(\omega) = \frac{\alpha-\alpha e^{-2i\omega}}{\frac{1}{2}-\gamma e^{-i\omega}+\beta e^{-2i\omega}}.$$

Here is a plot of A for the frequency $f_{center} = 256$, sampling frequency $f_s = 22050$, and quality $Q = 80$.

In[101]:=
```
Plot[Abs[A], {w, 0, Thetaf*2}, PlotRange → All]
```
Out[101]=



We will show that A is indeed our ideal curve for bandpass filtering.

We can define Q precisely by the ratio [13]

$$Q = \frac{f_{center}}{f_u - f_l},$$

where f_u is the high cutoff frequency and f_l is the low cutoff frequency of the filter. The frequencies f_l and f_u are cutoff frequencies because they determine the domain for which the amplitude of the filter $|A| \geq 0.5$, meaning frequencies outside this domain are attenuated, or decimated, and those inside the domain appear in the passband. The difference $f_u - f_l$ is the bandwidth of the passband.

Since θ_f is simply the frequency f scaled, Q is also the ratio

$$Q = \frac{\theta_{f_{center}}}{\theta_{f_u} - \theta_{f_l}}.$$

So, for our parameters for Q and f_{center}, it should be the case in A that $f_u - f_l = f_{center}/Q = 256/80 = 3.2$ Hz, the bandwidth of the filter, and judging by the apparent symmetry of A, the cutoff frequencies should be close to $f_u = 256 + 1.6 = 257.6$ Hz and $f_l = 256 - 1.6 = 254.4$ Hz. When we tune the filter, the width of A at half-power ($A = 0.5 \pm 0.5i$, or $|A| = 0.5$) is the ratio

$$\frac{\theta_{256Hz}}{Q} = \frac{2 \cdot \pi \cdot 256}{22050 \cdot 80} = 0.0009118455.$$

Now, $|A(\theta_f)| = 0.5$ at $\theta_{f_l} = 0.0724931385$ and $\theta_{f_u} = 0.073404984$, meaning that the untuned cutoff frequencies are $f_l = 254.404991$ Hz and $f_u = 257.604991$ Hz. Their difference is $\theta_{f_u} - \theta_{f_l} = 0.0009118455$, exactly the expected width of A for the quality

$Q = 80$, and the ratios

$$\begin{aligned}
\frac{\max_{\theta_f}(A(\theta_f))}{\theta_{f_u} - \theta_{f_l}} &= \frac{\theta_{256} \text{ Hz}}{\theta_{257.604991} \text{ Hz} - \theta_{254.404991} \text{ Hz}} \\
&= \frac{0.07294763904 \text{ Hz}}{0.0009118455 \text{ Hz}} \\
&= 80 \\
&= Q
\end{aligned}$$

and

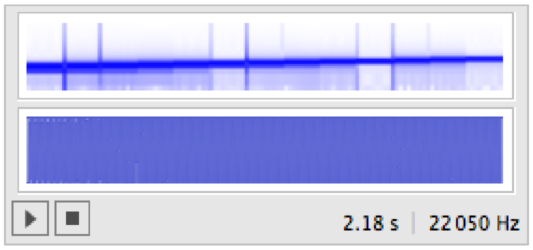
$$\begin{aligned}
\frac{f_{center}}{f_u - f_l} &= \frac{256 \text{ Hz}}{257.604991 \text{ Hz} - 254.404991 \text{ Hz}} \\
&= \frac{256 \text{ Hz}}{3.2 \text{ Hz}} \\
&= 80 \\
&= Q
\end{aligned}$$

are as expected.

1.2.2 Implementation

To demonstrate how our bandpass filter picks out a given frequency, we will design a 13 note scale, C to C, in Mathematica. Since we do not change Q for different frequencies, this filter is called a **constant-Q bandpass filter**.

```
len = 48 000;
F[t_] := 256.0 * 2^(1 / 12 Floor[13 t / (len + 1)]);
x = Table[16 000 Sin[2 Pi F[t] t / 22 050], {t, 1, len}];
ListPlay[x, SampleRate → 22 050, SampleDepth → 16, PlayRange → {-2^15, 2^15 - 1}]
```



The "SampleRate" set to 22050 indicates that the **frequency resolution** of our filter is 22050 Hz, or 22050 samples per second. CD-quality sound has a frequency resolution of 44100 Hz, so when it sampled at only 22050 Hz, the sound has twice the duration. The "SampleDepth" indicates the **amplitude resolution** of our filter. At 16, each sample therefore may have any of $2^{16} = 65536$ amplitudes [13]. Now, the filter itself contains an initial frequency f; a "tuning" θ_f (designated below by $T0$) equal to $2\pi f$ divided by the frequency resolution; and the quality Q, hard-wired at 80. The 3 coefficients, α, β, and γ, are constructed from θ_f and Q (as shown in the proof by *Ansatz* above) to change our signal x into a filtered signal y:

```
Bandpass[x_, f_, Q_] := Module[{alpha, beta, gamma, T0, y = x},
   T0 = 0.000279545 * f; (*Tuning for freq.f.*)
   beta = 0.5 * (1 - Tan[T0 / (2 Q)]) / (1 + Tan[T0 / (2 Q)]);
alpha = (1 / 2 - beta) / 2;
gamma = (1 / 2 + beta) Cos[T0];
y = x;
   y[[1]] = y[[2]] = 0;
len = Length[x];
Do[y[[n]] = 2 * (alpha * (x[[n]] - x[[n - 2]]) + gamma * y[[n - 1]] - beta * y[[n - 2]]),
    {n, 3, len}];
   Return[y]
  ];

BandPower[x_, f_, Q_] := Plus @@ Bandpass[x, f, Q]^2

freq[step_] := 440 2^(step/12) // N

freq[-9]

261.626
```

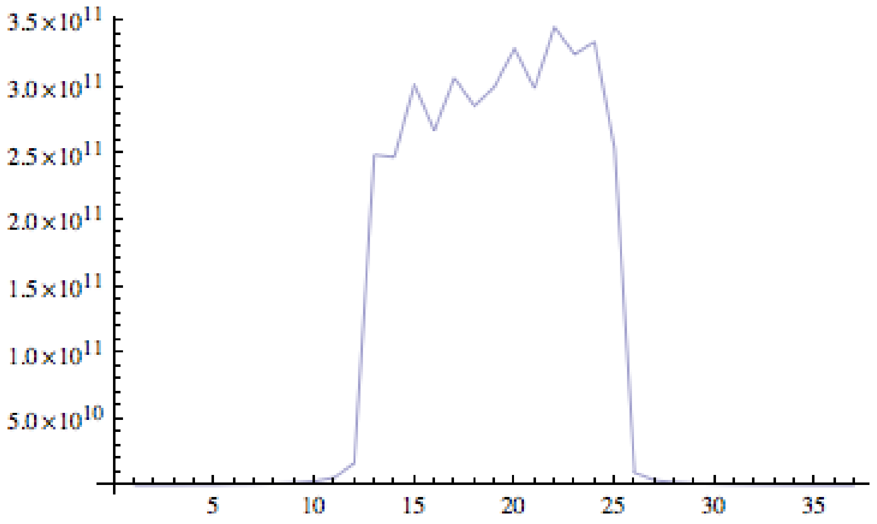
We calculate the total power, or energy, of the sound through the bandpass filter in the function $BandPower(x, f, Q)$ by, as defined above, squaring the terms and then summing them. Since we chose 440 Hz (A) to be our initial frequency (it is the only piano frequency that is an integer, not to mention the most common reference point amongst musicians since it lies within our vocal range and is close in proximity to the center of the keyboard, middle-C), 9 steps down will give us middle-C, at 261.626 Hz.

To create a **filter bank**, implementing many filters at once to obtain power data about the signal at each frequency and at each point in time, we define a table $ThreeOctavePower(x)$ in Mathematica as a storage location.

```
ThreeOctavePower[x_] := Table[BandPower[x, freq[k], 80], {k, -21, 15}] // Normalize

FourOctavePower[x_] := Table[BandPower[x, freq[k], 80], {k, -33, 15}] // Normalize

FiveOctavePower[x_] := Table[BandPower[x, freq[k], 80], {k, -57, -9}] // Normalize

Timing[power = ThreeOctavePower[x]]

{13.8084, {3.55591×10^8, 4.27738×10^8, 4.31383×10^8, 6.10964×10^8, 6.31153×10^8,
  9.47895×10^8, 1.11677×10^9, 1.4332×10^9, 1.96414×10^9, 3.06585×10^9, 5.8046×10^9,
  1.67382×10^10, 2.48831×10^11, 2.4699×10^11, 3.01692×10^11, 2.67095×10^11,
  3.06727×10^11, 2.85567×10^11, 2.99852×10^11, 3.28897×10^11, 2.98936×10^11,
  3.45032×10^11, 3.24466×10^11, 3.33795×10^11, 2.54115×10^11, 9.59815×10^9,
  3.86602×10^9, 2.29647×10^9, 1.58693×10^9, 1.18503×10^9, 9.24573×10^8, 7.44567×10^8,
  6.13709×10^8, 5.15115×10^8, 4.38038×10^8, 3.77052×10^8, 3.27589×10^8}}
```

So, a 2.1-second sample in Mathematica 7 takes more than 6 times its duration when filtered over three octaves. $ThreeOctavePower(x)$ stores the individual powers that each pitch generates in a range of three octaves. Its range here is from the C that is 21 half steps down from A440, up to the C that is 15 half steps up from A440, so it is actually of 37 frequencies, not 36. The first element after the timing is the power at low C; it took additional time to list out the powers..

```
ListPlot[power, Joined → True]
```



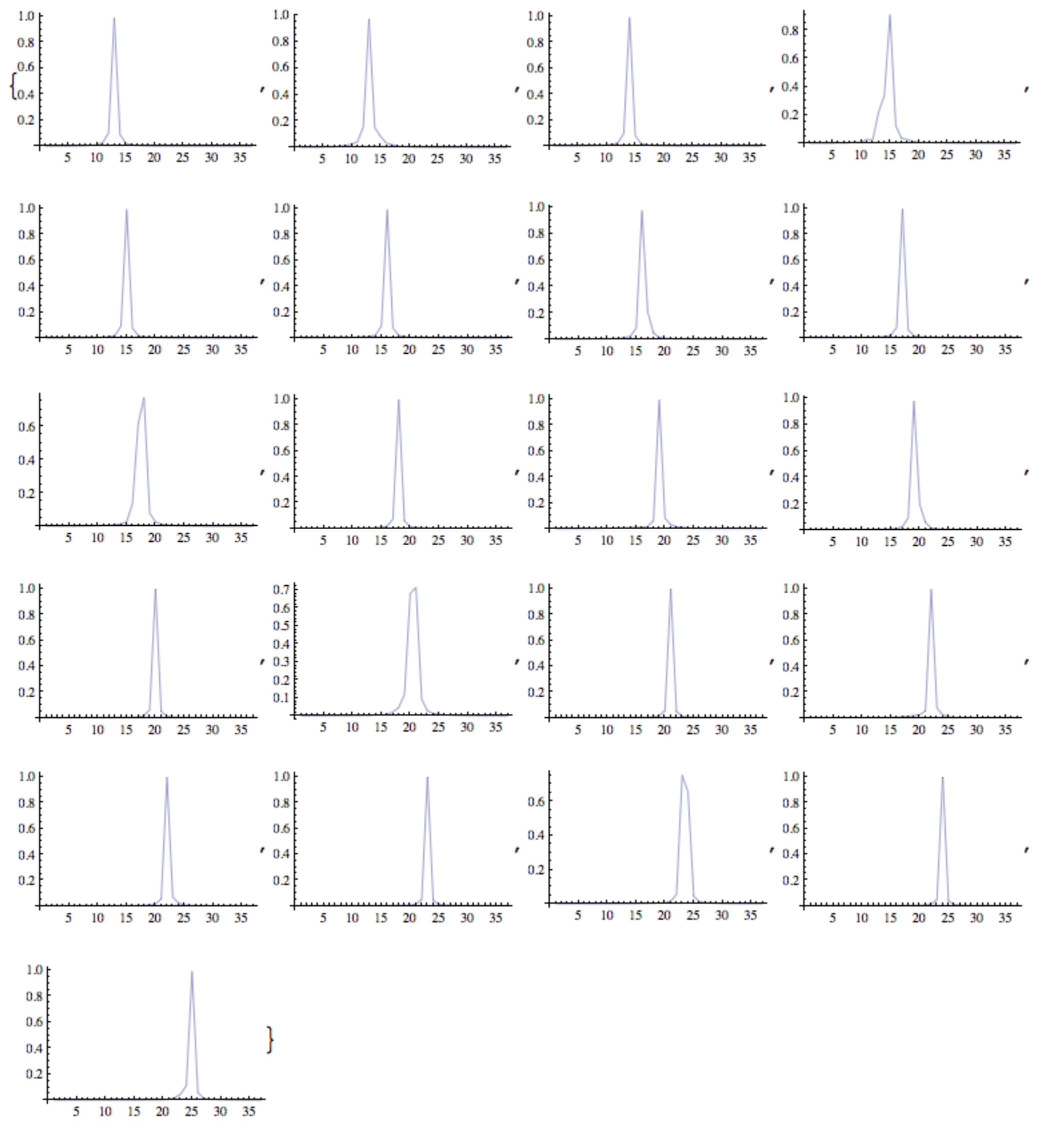
So, the frequencies 13 (261.626 Hz) through 25 (523.25 Hz), 13 notes in total, produce the most power, because they were the only ones present in our signal.

Now, we want to partition the sample (song) such that we can retrieve the frequencies from time 0 to time 0.1, 0.1 to 0.2, etc. until we have realized the entire signal. To do this, we partition x and multiply the segment of time (0.1 seconds) by the sampling rate, 22050 Hz, since it directly corresponds to the duration of the sample (a song sampled at 44100 Hz will be real-time, and twice as fast as one at 22050 Hz):

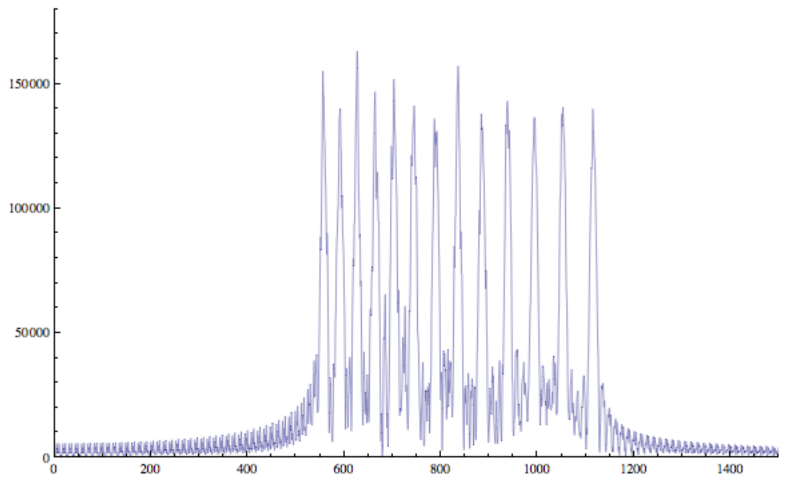
```
SoundPartition[x_, t_, sampleRate_] := Partition[x, Round[sampleRate * t]]

xs = SoundPartition[x, 0.1, 22050];
Table[ListPlay[xs[[k]], SampleRate → 22050, SampleDepth → 16,
  PlayRange → {-2^15, 2^15 - 1}], {k, 1, Length[xs]}]
```

Then, we can plot the power of each frequency at each interval of time. Note that those partitions in which two notes occur have a wider peak in their power graph. The IDFT of $x(t)$, also plotted below, reproduces the absolute value of the original signal from its spectrum, with a phase shift $\phi(k) = 2\arctan\frac{\Im(X(k))}{|X(k)|+\Re(X(k))}$, as given above. The frequencies present in $x(t)$ are shifted on the x-axis and appear to be about twice that of their actual value.

```
Table[ListPlot[powers[[k]], Joined → True, PlotRange → All], {k, 1, Length[powers]}]
```

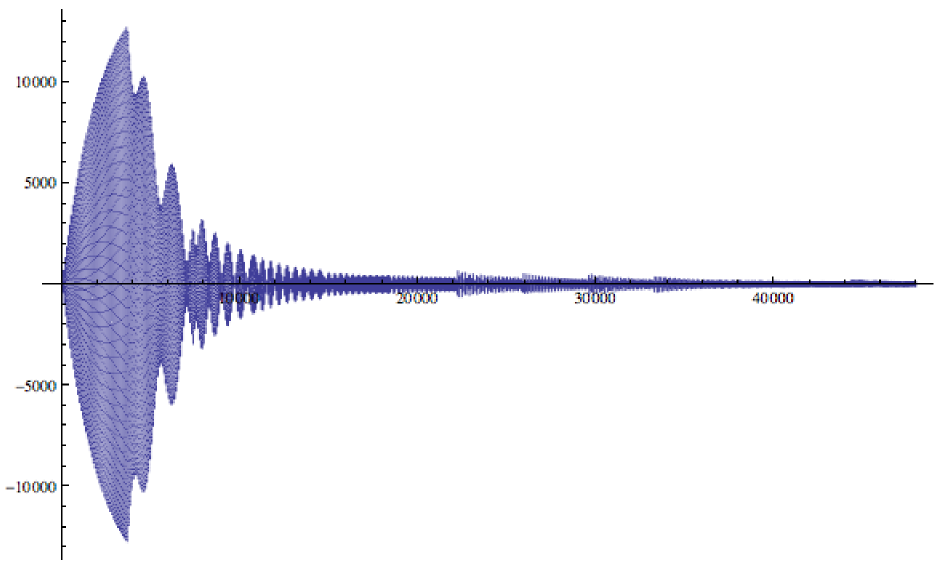


```
ListPlot[Abs[Take[InverseFourier[x], Length[y] / 2]], Joined → True, PlotRange → {{0, 1500}, {0, 180000}}]
```



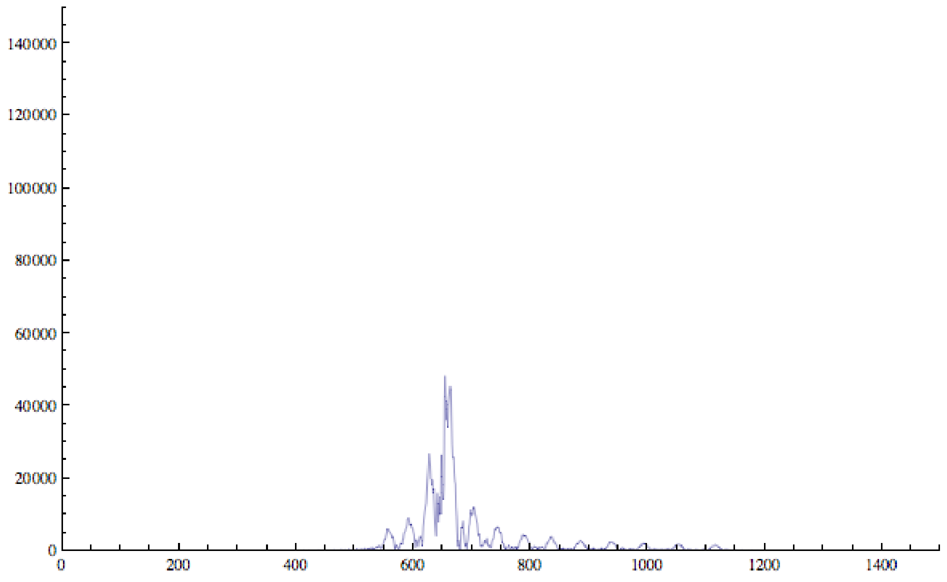
Now, let us filter out just one frequency in our sample. When we set our frequency f to C = 261.62 Hz, all the other frequencies are muted while the low note, C, passes through, because of the way y is designed. Thus, $y = BandPower(x, 261.62, 80)$.

```
ListPlot[y, Joined → True, PlotRange → All]
```



The x-axis shows the sample number, and there are a total of 22050Hz·2.18s = 48069 of them. The y-axis indicates volume, or power. The IDFT of this passband is shown in the following graph.

```
ListPlot[Abs[Take[InverseFourier[y], Length[y] / 2]], Joined → True, PlotRange → {{0, 1500}, {0, 150000}}]
```

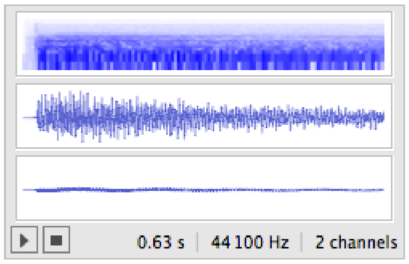


Again, the x-axis indicates twice the frequency of the actual signal. We can see that our filter does a fairly accurate job of only allowing 1 given note to pass through.

If we increased Q, we might get even more precise results, but $Q = 80$ seems to be sufficient for this project.

Now, let's test it on a chord. I chose the recognizable augmented chord at the beginning of The Beatles' "Oh! Darling". We can use just the left channel of the signal in our analysis to reduce the number of operations required.

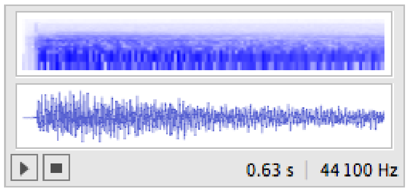
```
ohd = Import["oh!darlingfirstchord.wav"]
```



```
0.63 s | 44 100 Hz | 2 channels
```

```
samples = ohd[[1, 1]];
lch = samples[[1]];
```

```
Sound[SampledSoundList[lch, 44 100]]
```

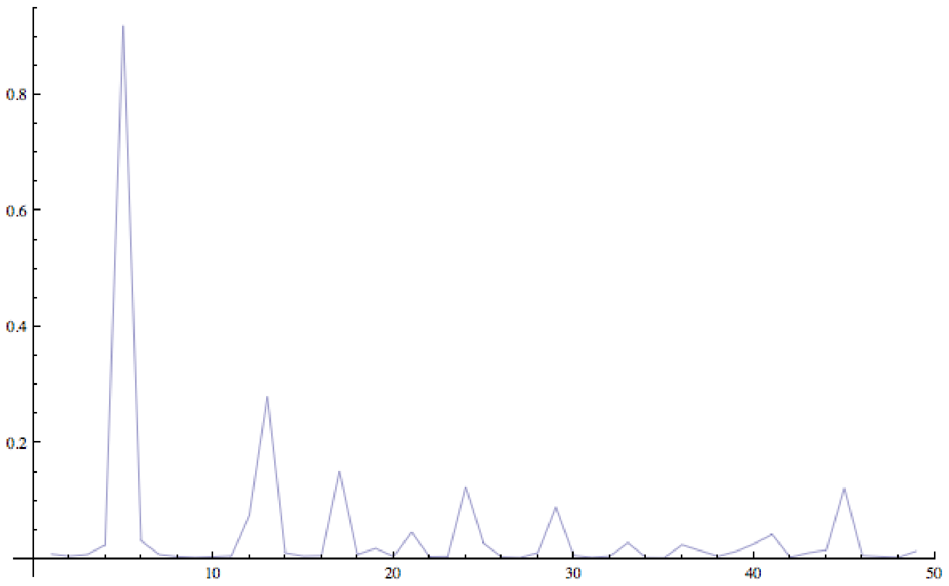


```
0.63 s | 44 100 Hz
```

```
chord = FourOctavePower[lch]; // Timing
{11.1814, Null}
```

```
chord = ThreeOctavePower[lch]; // Timing
{8.44973, Null}
```

```
ListPlot[chord, Joined → True, PlotRange → All]
```



The 0.63 second sample through 49 bandpass filters for four octaves (plus high C) takes 1.323 times the time the three octave (37 filters) module takes, which is approximately the ratio $49/37 = 1.324$. Thus, we can expect this program to run in approximately 13.412 times the length of the sample.

```
Select[Table[If[chord[[k]] > 0.05, k, 0], {k, 1, Length[chord]}], #1 > 0 &]
{5, 12, 13, 17, 24, 29, 45}

% + 47
{52, 59, 60, 64, 71, 76, 92}

chord
{0.726702, 0.0268361, 0.0127536, 0.0145642, 0.393619, 0.0194947, 0.0481771,
 0.0106354, 0.121394, 0.00959363, 0.0108587, 0.32254, 0.0704695, 0.00804902, 0.00531678,
 0.0276743, 0.231979, 0.0149965, 0.00671237, 0.0123157, 0.0736147, 0.00551785,
 0.00435736, 0.0644375, 0.0376369, 0.0116074, 0.0343393, 0.069216, 0.111089, 0.00734521,
 0.0269894, 0.0408107, 0.316938, 0.0147859, 0.0104242, 0.00686019, 0.0355728}
```

Finally, we retrieve the normalized powers of each note in the three octave spectrum, and see peaks at 5, 13, 17, 24, 29, and 45. We add 47 to each of these to get the MIDI keyboard equivalent, playable through GarageBand and similar platforms, and got E=52, B=59, C=60, E=64, B=71, E=76, and G♯=92, which actually spell out an augmented C-seventh chord, but the peak at E makes it clear that it is the root and that the B is incorrect. Indeed, it is an overtone of E, and the actual chord played is an E+ (augmented) chord.

1.3 A Tunable Bandpass Filter in C

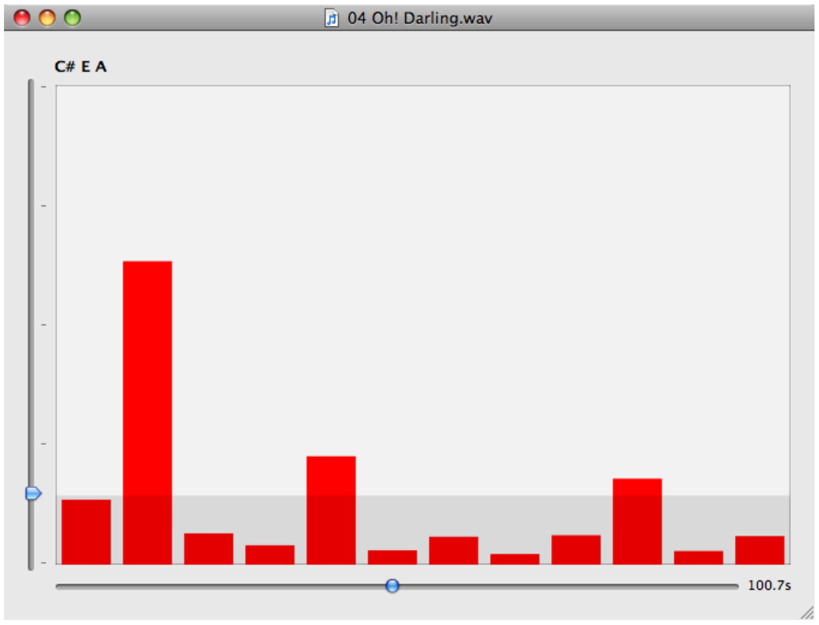


Figure 1.10: A screenshot of the application **BandPower** correctly identifying the chord A Major in the Beatles song "Oh! Darling" from *Abbey Road*.

To run a 3-minute song in Mathematica would take some 40 minutes, a gross amount of memory, and all of my very current MacBook Pro's 2.4 GHz dual processor power, so naturally, the main thing to improve upon is speed. In the language of C, Devin Chalmers built for me the incredibly fast application, **BandPower.app**, shown above. The application does exactly what the Mathematica program does (except it adds the powers at every octave together, so that we get only one power for each pitch class), but Mathematica can be very slow when dealing with a large quantity of numbers, which is obviously the case when we multiply our sampling frequency of 44100 by the duration (in seconds) of a sound file!

Importing and analyzing a WAVE-format song (60 seconds translate to about 10.1 MB) over five octaves takes about 4 seconds, including the 34.7 MB, 3:26-long Beatles' song, "Oh! Darling." Upon the selection of a file, a new window opens in **BandPower**. Once the song is loaded, we see a histogram with two sliding levers. On the x-axis are buckets for 12 pitches, ordered C through B, and on the y-axis, the power of each of these pitches is shown. We set the desired sampling instant by moving the lever along the x-axis, and the desired threshold power value by moving the lever along the y-axis. Those pitches whose powers at the designated sampling instant are above this power level are displayed at the top of the window, and the sampling instant is displayed at the bottom right corner.

According to **BandPower.app**, the chord progression of the first 33 seconds of "Oh! Darling" is

Chord	Time (s)
E	0.9
A	2.3
E	7.5
F♯ (unknown quality)[7]	10.8
b	14.9
E^7	21.2
b	23.0
A	27.8
D^7	29.6
A^7	31.2
E^7	33.4

The true chord progression of this section of "Oh! Darling" is

Chord	Time (s)
E+	0.2
A	2.3
E	6.8
f♯^{7}	10.6
D	14.9
b^7	19.0
E^7	21.0
b^7	23.0
E^7	25.0
A	27.0
D^7	29.0
A	31.0
E^7	33.0

Much can be said to excuse these errors, but since **BandPower** only includes 4 power plots per second, the times occasionally did not match up, so a complete understanding of its effectiveness (and error) cannot be attained. Here, we were able to name 6 of the 13 chords exactly correctly, and seldom did we mislabel the root, save when we failed to detect two chords at 14.9 and 25.0 seconds in. But note that D (D, F♯, A) contains all of the pitches (except B) that are in b^7 (B, D, F♯, A), and E^7 (E, G♯, B, D) contains two of the pitches in b^7 (B, D, F♯, A).

It is perfectly possible that the extra sevenths we noted with **BandPower** were present in the vocal melody or harmony, since sevenths make for a more "dramatic" sound, or simply the overtones of the thirds in the triads (the dominant is quite loud in the overtone series). The chord that was the most off base was the first, which should have been labeled "E+," but, although our ears are bad detectors of loudness (even a very soft noise, coming from silence, will seem loud to our ears, because it startles us), the first chord is played on a piano with no accompaniment, unlike the rest of the song.

1.3.1 Smoothing

Since it is so hard to find an actual triad that makes it to the three most powerful frequencies, I decided to smooth the data and see if meaningful chords would evolve then. The entirety of "Oh! Darling," unsmoothed and then smoothed in the program **R**, is depicted in Figures 1.11 and 1.12. These colorful graphs are known as **Pitch Class Profiles**, in which the y-axis shows "pitch class" (one of C through B), the x-axis is time, and the entries are the individual powers. Those with little to no power are colored dark blue, and those with some to a lot of power are tinted somewhere between green and orange. This graphical system, developed by Fujishima in 1999, is a powerful way of visualizing the harmony of a song because it is three-dimensional. It is clearly much easier to understand and conjecture about the chord progression from the smoothed contour map versus the unsmoothed one.

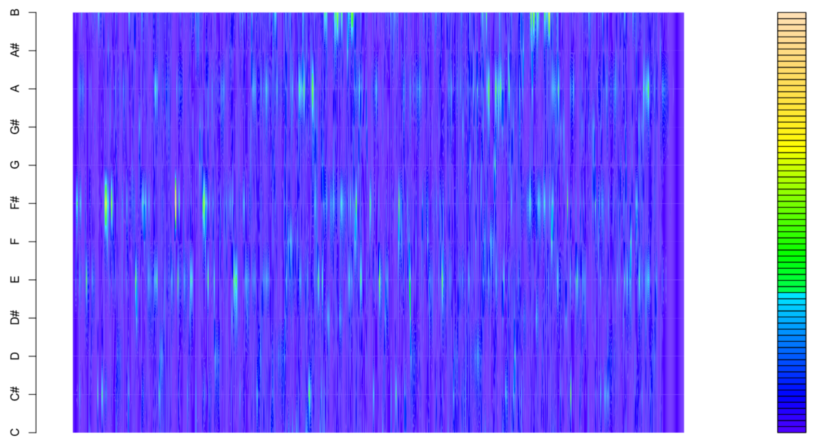


Figure 1.11: Unsmoothed contour plot of "Oh! Darling".

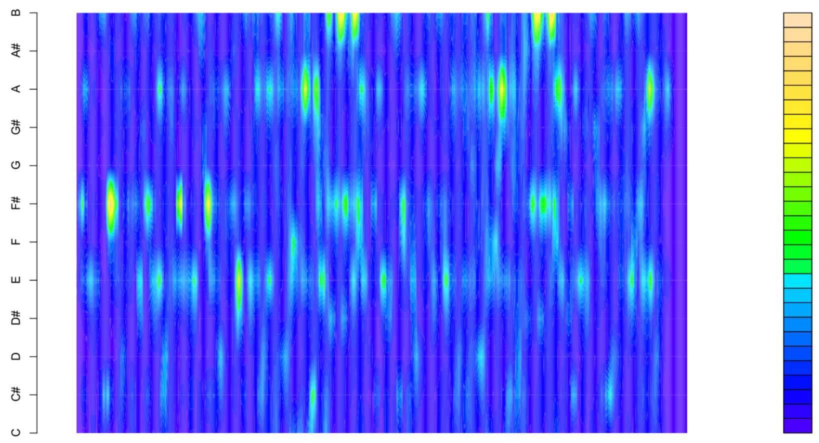


Figure 1.12: Smoothed contour plot of "Oh! Darling".

After exporting the data into a Comma Separated Values (.csv) file from **Band-Power**, smoothing of it was achieved by averaging over a user-defined[3] range of partitions, each $\frac{10000}{44100}$ seconds in length, with the function "windowsmooth" that takes the range of partitions and data as its input:

```
> windowsmooth <- function (ww=5, dd) {
+   # padd vector at two ends
+   pp <- dd
+   for (i in 1:ww) {pp <- c(dd[1], pp)}
        # padd start ww times
+   for (i in 1:ww) {pp <- c(pp, dd[length(dd)])}
        # padd end ww times
+
+   # create smoothed vector
+   ss <- dd
+   for (i in 1:length(dd)) {ss[i] <- mean(pp[i:(i+ww+ww)])}
+   ss
+ }
>
> # read the data
> ohd = read.table('Desktop/ohd.csv')
> # convert to a matrix
> ohdm=as.matrix(ohd)
>
> # next two lines plot a contour and note labels
> # variables: 32 (number of frames to smooth), nlevels is the number of contours
> filled.contour(windowsmooth(4,ohdm), color.palette=topo.colors, asp=0.7,
axes=FALSE, frame.plot=FALSE, nlevels=50)
> axis(2, at=(0:11)/11, labels=notes, line=-2)
```

From these smoothed values (and its corresponding pitch class profile representation), we can see that some pitches are struck far more than others. Since only 7 comprise a scale, we will see if the 7 most played pitches build the (or any, for that matter) scale.

[3]If the minimum duration of any chord in a progression is known, the user should input this duration into the smoothing function to achieve more robust plots.

1.3.2 Key Detection

Because the majority of humans do not have perfect pitch, intervals of two notes sound the same regardless of their location on the keyboard. For example, the interval of 5 half steps known as a perfect fourth (P4) sounds the same whether between an A and D, or a G and C. To account for this aural normalization, we write chords as Roman numerals relative to their key, where "I" and "i" both occur on the first **scale degree**[4] (the **tonic**), "ii" and "iio" on the second degree of the scale (the **supertonic**), and so on. The notation "$\flat$ II" means that the chord is rooted at a half step above the first scale degree and a half step below the second scale degree. The seven "ordinary" triads within a major key are labeled I, ii, iii, IV, V, vi, and viio, where a capitalized Roman numeral indicates a major **quality**, i.e., the second note in the chord is 4 half steps above the root and the third note in the chord is 7 half steps above the root. A lowercase Roman numeral indicates a minor quality, where the second note is 3 half steps above the root and the third is also 7 half steps above the root. Finally, a lowercase Roman numeral followed by a "o" indicates a diminished chord quality, where the second note is 3 half steps above the root and the third is 6 half steps away from the root.

The natural triads in a minor key are i, iio, III, iv, v, IV, and VII. If you take a moment to look at the pattern of qualities of the chords in the minor key lineup against that of the major key, you'll see that the minor key begins on the sixth scale degree of a major key. This major key is called its **relative** major key because the notes are the same.

When we are in a major key but encounter a chord like v, we say it is **borrowing** from the **parallel** minor key, for the root is the same but the scale degrees are in different places (i.e., G Major and g minor are in a parallel-key relationship). It is very common for both major and minor keys to borrow chords from their relative counterparts, in the Beatles' and Beethoven's music alike. In fact, I wish I had accounted for this common event in my analysis and naming with Roman numerals, by saying that every song is in a major key, since the chords can simply be written vi, viio, I, etc. instead of i, iio, III. That way, the behavior of major and minor songs could be analyzed together, and the labeling would be normalized.

When we encounter a chord that is neither in the major or parallel minor scale, we consider two things before labeling it with a Roman numeral. First, we analyze its quality. If it is a major chord, we check to see if its root is at $\flat\hat{2}$, in which case it is the Neapolitan chord. Otherwise, we label it with a "V/" and see what note is 7 half steps below its root, and put the appropriate Roman numeral from our key (or its parallel key) underneath this slash. Most of these chords, called secondary dominants, transition next to the chord underneath the V. This is called **resolution**. For instance, V/iii (consisting of the scale degrees $\hat{7}$, $\sharp\hat{2}$, and $\sharp\hat{4}$) "resolves" to the chord *iii* (consisting of $\hat{3}$, $\hat{5}$, and $\hat{7}$). But in modern music, this is not as strict of a rule as it was in classical compositions, so we will label a chord constructed with $\hat{7}$, $\sharp\hat{2}$, and $\sharp\hat{4}$ as a V/iii even if it does not resolve to iii.

Now, summing the powers of each pitch in the song "Oh! Darling," we calculate

[4]See glossary.

the energy of each note. We will see if the top seven pitches translate to its true key of A Major.

Pitch	Energy
C	1080.135
C♯	1463.439
D	1147.141
D♯	1388.449
E	2540.088
F	1523.343
F♯	2492.183
G	1100.494
G♯	1503.934
A	2419.257
A♯	1270.820
B	2322.360

So, we have that E is our most encountered pitch, then F♯, A, B, F, G♯, and finally C♯. This is not quite a scale, but since F, C, and G are all sharp (and A is not), we can assume that we are in a scale containing sharps and not flats, meaning that at least F is sharp. So, let's throw away the F natural (unsharpened), so that we have C♯ - E - F♯ - G♯ - A - B. We have narrowed our options down, then, to the keys of c♯ minor, E Major, f♯ minor, and A Major. This could very well be E Major or c♯ minor (they contain the same frequencies in their scales), both of which contain a D♯. But D♯ is the eleventh most common pitch, so we try our other option, and see that D is the tenth most common pitch. Therefore, we choose the scale A - B - C♯ - D - E - F♯ - G♯.

Now, should we pick the major or minor version? Looking again at our order of pitches, we see that F♯ is the third most heard frequency. This would lead us to incorrectly label the key as f♯ minor when the true key is A Major, but our attempts came close. Another way of prioritizing the pitches might be searching for dominant relationships. For example, we see that E and A are the first and third most played pitches, and F♯ and C♯ just the second and seventh. This would lead us to a correct naming of the key.

1.3.3 Summary

In sum, the bandpass filter within the application **BandPower** does an impressive job of identifying key and chords. I did not do any fiddling with Q, hard-wired in **BandPower** at 80, and could see how modifying it for each frequency could improve accuracy, because the difference between 25 and 26 Hz is much greater sonically than 2025 and 2026 Hz. All of the observations of its capabilities were made without computer assistance, so the development of an algorithm to retrieve triads from the smoothed data is the next step. I did not get around to implementing some Python code that I wrote to label chords with their appropriate Roman numeral, so it is located in Appendix A.

Chapter 2

Markovian Probability Theory

This temporal world is filled with events that depend on the past. Nearly every present manifestation of human behavior is based on what one has learned from mistakes and successes from the past, appearing to approach some optimal limit after enough time has passed. **Markovian processes** characterize events that happen in sequence, and models their local behavior with a matrix.

The hidden Markov model (HMM) is used when we want to predict the behavior of a data set, and we are not sure of how to exactly characterize its prior behavior. Thus, we build a Markov chain based on previous observations of the action or set of actions we wish to predict, whether the stock market, or the weather, or one's health—or, here, the typical chord progression of a style, or region, or artist, or album, and then we can produce guesses for future observations, given that the model is not completely random. Will Landecker studied HMMs and attempted to compose a new, Bach-like melody using data from one of Bach's works, Minuet in G. My project aims not to compose music from my data, but to study the difference in the models constructed from various pools of songs, so it just involves counting up the number of times we transition from chord to chord, and dividing by the total sum of chords to find a probability, instead of the complex algorithms involved with hidden Markov models. For instance, I would like to know the difference between Irving Berlin's and Cole Porter's songs, so I create two chains from their bodies of work, thereby forming two (presumably distinct) transition matrices, and subtract the two matrices. But when I want to look at more than two models, another measure of comparison must be made. Hence, I look at the entropy of the system, since it measures uncertainty. We will look at the meaning derived from the subtraction versus that derived from the measure of entropy, and decide if entropy is indeed a good indicator of style, relative to other styles, in music.

For more on hidden Markov models and their application to automatic speech (and chord) recognition, see Appendix B.

2.1 Probability Theory

The idea of Markovian processes is easily relatable and accessible in the real world of choices and decision-making, like much of the notions enveloped by probability theory: a Markov chain describes the probability of some state s_j occurring next given that we are "in" state s_i with a matrix that contains this probability in the (i, j)th entry. This can be thought of as the probability that it will rain tomorrow (state s_j), or any other designated period of time, given that it is raining today (states $s_i = s_j$), or the probability that it will not rain tomorrow (state s_j) given that it is not raining today (states $s_i \neq s_j$), and so on, to describe 4 total transition probabilities. Say the probability of it raining tomorrow, given rain today, is 0.7, and the probability that it will rain tomorrow given that it is not raining today is 0.2. Then we can infer that the probability of it not raining tomorrow given that it is raining today is $1 - 0.7 = 0.3$, and the probability of no rain tomorrow given no rain today is $1 - 0.2 = 0.8$. This is visualizable with a transition matrix:

$$A = \begin{bmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{bmatrix}$$

The entry $a_{ij}, 1 \leq i, j \leq 2$ is the probability of going from state s_i to state s_j, so here, state s_1 is the case in which it is not raining, and state s_2 is the case in which it is raining. Note that the rows of A must sum to 1 if the state appears in the Markov chain at any point before time $t = T$. Otherwise, the rows sum to 0.

Now, if this data did not exist somewhere, we would have to count it up ourselves from some subset of observations of the weather, and build these probabilities ourselves. This is the naïve version of a Hidden Markov Model, and we leave the algorithmic one to be explained in Appendix B. The naïve version is more manageable when we do not need to compare our observations with our expectations. Because we only have a little intuition on the abstract nature of quantifying musical harmony, we can describe our expectations qualitatively, and test our results by experimentation.

Before we get too far into Markov chains, however, let us review some essential parts of probability theory [5] that will help our intuitions on the subject.

Definition: Discrete-Time Random Variable. A real-valued variable X is a **discrete-time random variable** if it can only take on at most a countable number of possible values s_i for all arguments t of X, $0 \leq t \leq T$, $t, T \in \mathbb{N}$. We call the set of these possible values that X can take on the **state space** S of X, $|S| = N \leq T$. We can formally define the function X by

$$X : \mathbb{N} \to S.$$

For example, say that our state space consisted of 24 major and minor chords, and we ordered them so that $s_1 =$ C Major and $s_N = s_{24} =$ b minor. Then, the random variable X would map each point in time to one of these chords, with replacement (i.e., it could be the case that $X(t) = X(t') = s_1 =$ C Major, for $t \neq t'$). This makes it clear that it is not necessarily true that $X(1) = s_1$; $X(1)$ could be any value of S.

Definition: Discrete-Time Random Process. A **discrete-time random process**

$$\{X(0), X(1), \ldots, X(T)\}$$

is an ordered sequence of real, positive integer values corresponding to the value of the random variable X at time t. We say that $X(t)$ is in state s_i at time t iff $X(t) = s_i$, $1 \leq i \leq N$, $i, t, T, N \in \mathbb{N}$.

Definition: Event. We define an **event** E to be any set of outcomes in S, such as the event that the sum of the two dice is 7, in which case $E = \{(1,6), (2,5), (3,4), \ldots\} \subseteq S$, and $(1,6)$ is an **outcome** $e_i \in S$. The union of all the events in S equals S. When we define events such that they are disjoint, the sum of the probabilities of all the events is 1.

Definition: Independence. Two events or outcomes in S are **independent** if the probability of one of them occurring does not influence the probability of the other occurring.

For example, when we toss a die, whether it is weighted or unweighted ("fair"), its outcome is not influenced by previous outcomes. There are few real world examples in which the outcomes are independent of each other.

Definition: Conditional Probability. The **conditional probability** of event F occurring given that E occurred is given by the following formula:

$$P(F|E) = \frac{P(EF)}{P(E)}$$

for $P(E) > 0$. We consider the concept of conditional probability to be the opposite of independence, since when E and F are independent, $P(F|E) = P(F)$.

Keeping our example of the sum of two dice, let E be the event (outcome) in which the first die equals 2. What, then, is the probability of the sum of that die's outcome plus that of another die yet to be rolled equals 8? It is simply $P(\text{second die is } 6) = \frac{1}{6}$, which is not equal to $P(\text{sum of two dice is } 8) = \frac{|\{(2,6),(3,5),(4,4),(5,3),(6,2)\}|}{|S|} = \frac{5}{36}$.

Definition: Bayes' Formula. Suppose that $F_1, F_2, \ldots, F_n$ are mutually exclusive (disjoint) events such that

$$\bigcup_{i=1}^{n} F_i = S,$$

i.e., exactly one of the events F_i must occur. Define E by

$$E = \bigcup_{i=1}^{n} EF_i,$$

and note that the events EF_i themselves are mutually exclusive. Then we obtain

$$\begin{aligned} P(E) &= \sum_{i=1}^{n} P(EF_i) \\ &= \sum_{i=1}^{n} P(E|F_i)P(F_i). \end{aligned}$$

In other words, we can compute $P(E)$ by first conditioning on the outcome F_i. Suppose now that E has occurred and we are interested in determining the probability that F_j also occurred. This is the premise of **Bayes' formula**, which is as follows:

$$\begin{aligned} P(F_j|E) &= \frac{P(EF_j)}{P(E)} \\ &= \frac{P(E|F_j)P(F_j)}{\sum_{i=1}^{n} P(E|F_i)P(F_i)}. \end{aligned}$$

This formula may be interpreted as evidencing how the "hypotheses" $P(F_j)$ should be modified given the conclusions of our "experiment" E.

Definition: Probability Mass Function. For a discrete-time random variable X, we define the **probability mass function** $p(x)$

$$\text{for } x \in \mathbb{R}, \quad p(x) = P\{t \in N : X(t) = x\}.$$

The probability mass function **(pmf)** is positive for at most a countable number of values of x, since X is discrete. Since X must take on one of the values x, we have

$$\sum_{x \in X} p(x) = 1.$$

The **distribution function F(x)** of X is similar in that it finds the probability that the random variable is less than or equal to x, and is defined by

$$F(x) = P\{x \in R : X \leq x\} = \sum_{y=-\infty}^{x} p(y),$$

and is also referred to as the **cumulative distribution function** or **cdf** of X. It is nondecreasing, such that $P\{a \leq X \leq b\} = F(b) - F(a)$ is nonnegative.

Definition: Expectation. If X is a discrete random variable having a probability mass function $p(x)$, the **expectation** or the **expected value** of X, denoted by $E[X]$, is defined by

$$E[X] = \sum_{x:p(x)>0} x\, p(x),$$

where the x's are the values that X takes on. This can be thought of as a weighted average of the state space S.

For example, flipping a fair coin heeds the probability mass function $p(\text{heads}) = \frac{1}{2} = p(\text{tails})$, so we could write

$$p(0) = \frac{1}{2} = p(1)$$

$$E[X] = 0\left(\frac{1}{2}\right) + 1\left(\frac{1}{2}\right) = \frac{1}{2}.$$

Definition: Finite Probability Scheme. A **finite probability scheme** E of the sample space S is a partition of S into disjoint events E_i with probabilities that sum to 1, represented by the array

$$S = \begin{bmatrix} E_1 & E_2 & \dots & E_n \\ p_1 & p_2 & \dots & p_n \end{bmatrix}$$

where n is the number of parts in the partition of S, and p_i is the probability of event E_i, $\sum_{i=1}^{n} p_i = 1$.

Each event E_i has its own set of individual outcomes, which can be thought of as elementary events (events with cardinality 1), written $e_1, \dots, e_m$ with probabilities $\sum_{j=1}^{m} p(e_j) = 1$ as well.

2.2 Markov Chains

Now that we have laid the framework for the terminology used in defining our hidden Markov models, we can continue to a formal definition of a Markov chain, and to the properties they exhibit. Here, we will use the notation $(X_0, \dots, X_T)$ in place of $\{X(0), \dots, X(T)\}$, only because it is widespread.

Definition: Markov Property. If the conditional distribution of the variable X_{t+1}, given the process $(X_0, X_1, \dots, X_t)$, where X_i is some state such as a rain on day i, depends only on the previous variable, X_t, i.e.,

$$p(X_{t+1}|X_0, \dots, X_t) = p(X_{t+1}|X_t),$$

the process is said to satisfy the **memoryless property**, or **Markov property**.

The word "depends" here is somewhat misleading. Independent outcomes can be modeled by a Markov chain: then, $p(X_{t+1}|X_t) = p(X_{t+1})$. Therefore, two consecutive states do not necessarily have to be related to write them as a conditional probability.

Instinctively, Markov chains possess the Markov property. They can be "homogeneous," "irreducible," "aperiodic," and "stationary," all of which are defined below. Hidden Markov models are "time-homogeneous," which means that the probability of transitioning between two given states does not change over time. Hence, even when we observe a sequence of four heads from a fair coin, the probability of flipping a head next is still 0.5.[1]

Although music is much like speech in its grammar-like structure, two musicians actually differ less than two speakers in their spectrograms. Therefore, there is no need to waste time constructing a whole hidden Markov model for our data since the $F\sharp$ from a piano does not differ drastically from the F played from a guitar. For more on hidden Markov models, see Appendix B.

Now we have arrived at our definition of a Markov chain. We call our set of chords the "state space" S[2], and our chord progression an ordered sequence X of chords in S from time $t = 0$ to $t = T$. We record the transition probabilities in an $|S| \times |S|$ matrix A.

Definition: Markov Chains. Let A be a $n \times n$ matrix with real-valued elements $\{a_{ij} : i, j = 1, \ldots, n\}$, all of which are nonnegative and sum to 1. A random process $(X_0, \ldots, X_T)$ with values from the finite state space $S = \{s_1, \ldots, s_n\}$ is said to be a Markov chain with transition matrix A, if, for all t, i, and k such that $0 \leq t \leq T$, $1 \leq \{i, j, i_0, i_1, \ldots, i_k\} \leq n$

$$\begin{aligned} p(X_{t+1} = s_j | X_0 = s_{i_0}, \ldots, X_{t-1} = s_{i_k}, X_t = s_i) &= p(X_{t+1} = s_j | X_t = s_i) \\ &= a_{ij}, \end{aligned}$$

Thus, a_{ij} is the transition probability of moving from state s_i to state s_j.

More specifically, this is a **time-homogeneous** Markov chain, since, from one time to another, the transition probabilities do not change.

In other words, a Markov chain is a random process usually governed by something that cannot be concretely understood, like the weather governs rain or harmony governs music. Harmony in music can come from unexpected aspects, like new chords that have rarely been played before or thunderstorms in Oregon, and that is why we consider this and other Markov chains random processes. Note that just because it is random does not mean there do not exist patterns that we can identify after the fact—in other words, it can still contain conditional probabilities, but the weather tomorrow does not necessarily abide by those conditional probabilities. Rather, the character of the newly observed state of the weather is added to the Markov chain, and the transition probabilities are then updated.

Property: Probability mass function. X has its probability mass function

[1]Since we do classify music by the era from which it comes, our hidden Markov model is not time-homogeneous, and is instead "state-homogeneous," since there are only a finite number of possible states.

[2]In section 2.2.1, Song as a Probability Scheme, we call the state space C.

based on the real values of elements of its transition matrix A [3], where

$$p(a_{ij}) = P\{A = a_{ij}\} = p(X_{t+1} = s_j | X_t = s_i).$$

For each i,

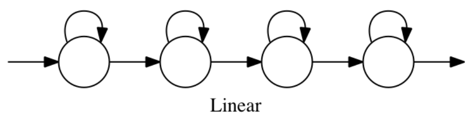
$$p(X_{t+1} = s_1 | X_t = s_i) + p(X_{t+1} = s_2 | X_t = s_i) + \ldots + p(X_{t+1} = s_j | X_t = s_i) = 1.$$

Property: Initial distribution. The initial distribution is the distribution of the Markov chain X at time 0 and is represented by a row vector π_0 [3] given by

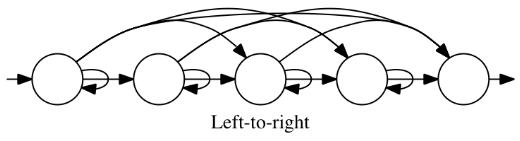
$$\begin{aligned} \pi_0 &= (p(s_1), p(s_2), \ldots, p(s_n)) \\ &= (p(X_0 = s_1), p(X_0 = s_2), \ldots, p(X_0 = s_n)) \end{aligned}$$

where s_i is the ith state. The probability $p(X_0 = s_i)$ is simply the number of times s_i occurs over the total number of states that occurred, i.e., the number of C Major chords we observe in a sequence of N chords. Thus, all of the initial probabilities sum to 1, just as do all of the transition probabilities in each row of A.

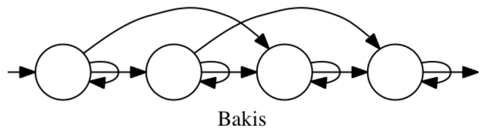
A powerful way of visually conveying a Markov process is with a "transition graph," which designates states as nodes and transitions as arrows with the possible transitions of a state s_i written underneath it. Four common ones are depicted as follows, all images from [45]:



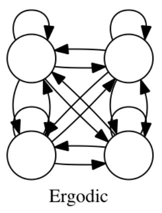
$s_i \to \{s_i, s_{i+1}\}$



$s_i \to \{s_i, s_{i+1}, \ldots, s_k\}$



$s_i \to \{s_i, s_{i+2}\}$



$s_i \to \{s_1, \ldots, s_k\}$

Since I am assuming here (perhaps naïvely) that any style of music could contain any chord, and that chord could transition to any other chord, we expect our Markov chain to be ergodic, i.e., any state can transition to any other state.

Definition: Irreducibility. A Markov chain is said to be irreducible if it does not decimate in time, i.e., if for all states s_i, s_j in $S = \{s_1, ..., s_n\}$, we have $s_i \leftrightarrow s_j$

(any state may transition to any state) as depicted in the "ergodic" transition graph above. Otherwise, the chain is said to be reducible.

We check for irreducibility in a chain by multiplying its transition matrix by itself n times, for some n, and seeing if the diagonal entries of A are greater than 0. In other words, there exists an n such that the (i,j)th term of the matrix $A^m_{ij} > 0$ for some $i,j \in \{1,\ldots,k\}$, where $A^m_{ij} = p(X_{l+m} = s_j | X_l = s_i)$.

We multiply a transition matrix by itself when we are trying to find the probability of a sequence after some quantity of "steps," i.e., the probability of transitioning to G Major after hearing C Major and F Major in the key of C, or in by their equivalent Roman numerals, of transitioning to V after I and IV. Thus, A^2_{00} is the probability of returning to state s_0 given that s_0 occurred 2 states back, or two steps ago, no matter what happened in between. This sort of analysis of Markov chains is certainly relevant to the goal of the complexity of a chord progression, but will not be quantified here. However, the computation is simple, and it can be shown that many (usually simple) progressions do approach a distinct limit for each j for large m in A^m_{ij}.

Definition: Aperiodicity. The period $d(s_i)$ of a state s_i is given by

$$d(s_i) = gcd\{m \geq 1 : [A^m]_{i,i} > 0\}.$$

If $d(s_i) = 1$, i.e., if there does not exist a common factor between the diagonal entries of $A^1, A^2, \ldots$, we say that the state s_i is **aperiodic**. A Markov chain is said to be aperiodic if all of its states are aperiodic. Otherwise, it is said to be periodic.

Example. A state in a Markov chain is periodic if the probability of returning to it is 0 except at regular intervals, and at these regular intervals, the probabilities have some greatest common denominator (note that these will be less than 1, and in some cases, 1 is actually reasonable). An example of a periodic state, say s_0, is given by the transition matrix

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1-p & 0 & p \\ 0 & 1 & 0 \end{bmatrix},$$

$$\begin{aligned} A^1_{00} &= 0 \\ A^2_{00} &= 1-p \\ A^3_{00} &= 0 \\ A^4_{00} &= 1-p \\ A^5_{00} &= 0 \\ A^6_{00} &= 1-p \\ &\vdots \end{aligned}$$

so for $m = 2k, k \in \mathbf{Z}^+$, $A^n_{00} = 1-p > 0$, and for $m = 2k+1$, $A^n_{00} = 0$, making the $gcd\{m \geq 1 : A^m_{00}\} = 1-p$. Likewise, A_{11} and A_{22} are periodic, for in fact, $A = A^{2j+1}$, and $A^{2j} = A^{2k}$ for all positive integers j, k.

A finite, irreducible, aperiodic Markov chain is said to be **ergodic**. Ergodic Markov chains satisfy $A_{ij}^m = \sum_{k=0}^n A_{ik}^r A_{kj}^{m-r}$ for all $0 < r < m$, a set of equations known as the **Chapman-Kolmogorov equations** [5].

Proof.

$$\begin{aligned} A_{ij}^m &= p(X_m = s_j | X_0 = s_i) \\ &= \sum_k p(X_m = s_j, X_r = s_k | X_0 = s_i) \\ &= \sum_k p(X_m = s_j | X_r = s_k, X_0 = s_i) p(X_r = s_k | X_0 = s_i) \\ &= \sum_k A_{kj}^{m-r} A_{ik}^r, \end{aligned}$$

where the third step is by the Markov property. $\square$

Now, since $A_{ij}^m > 0$ for all $i, j = 0, 1, \ldots, n$, A_{ij}^m converges as $m \to \infty$ to a value π_j that depends only on j (A First Course in Probability, p. 469). We call this π_j the **limiting probability** of the state s_j. Since by the Chapman-Kolmogorov equations we have

$$A_{ij}^{m+1} = \sum_{k=0}^n A_{ik}^m A_{kj},$$

it follows that, as m approaches infinity,

$$\pi_j = \sum_{k=0}^n \pi_k A_{kj}.$$

Furthermore, since $\sum_{j=0}^n A_{ij}^m = 1$,

$$\sum_{j=0}^n \pi_j = 1.$$

Thus, the $\pi_j, 0 \leq j \leq n$, are the distinct, nonnegative solutions to the equations $\pi_j = \sum_k \pi_k A_{kj}$ and $\sum_j \pi_j = 1$. We say that for an ergodic Markov chain with transition matrix A, as m goes to infinity in the limit

$$\lim_{m \to \infty} A_{ij}^m,$$

the Markov chain is approaching **equilibrium**.

2.2.1 Song as a Probability Scheme

We use the above definition of a probability scheme to describe a Markov chain X of chords as follows:

$$\begin{aligned}
X &:= (X_0, X_1, \ldots, X_T) = \text{a probability scheme, where events are the chord} \\
&\ \text{progressions of individual songs, classified in any (user-defined) way(s),} \\
&\ \text{including but not limited to conventional style, artist, region, era,} \\
&\ \text{and/or album;} \\
C &:= \text{the state space of } X; \\
c_i &:= (\mathit{key}, \mathit{tonality}, \mathit{root}) = \text{a triple corresponding to the } i\text{th chord in } C; \\
a_{c_i c_j} &:= a_{ij} = \text{the transition probability of transitioning from chord } c_i \text{ to } c_j; \\
p(c_i) &:= p_i = \text{the relative frequency of } c_i \in X; \\
E[X] &:= \text{the expected chord progression of classification } X; \\
H(X) &:= \text{the average entropy per chord in } X; \\
N &:= \text{the order of, or number of distinct chords in, } C
\end{aligned}$$

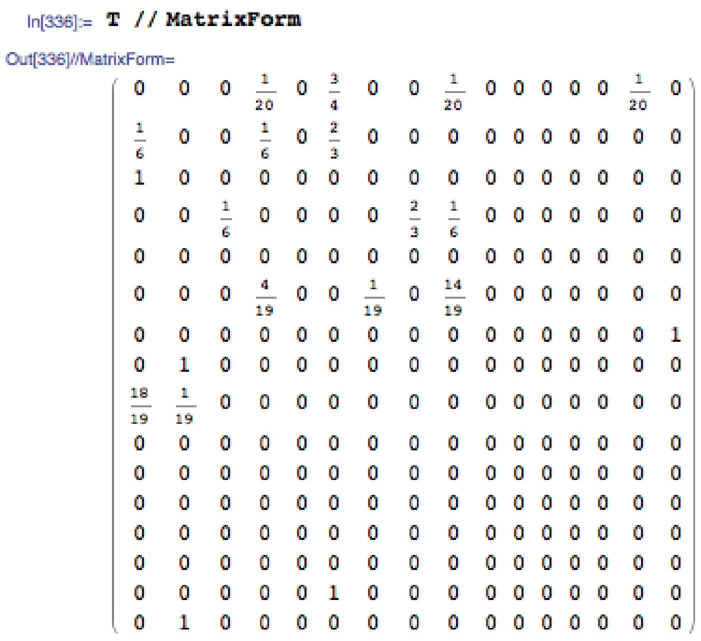
where $1 \leq i \leq N$; $\mathit{key}, \mathit{root} \in \{0, 1, \ldots, 11\}$; and $\mathit{tonality} \in \{\text{major, major}^7\text{, minor, minor}^7\text{, diminished, diminished}^7\text{, augmented, augmented}^7\}$. Key, root, and tonality defined this way comprise everything that one needs to know to build every kind of chord, including those with major or minor sevenths, but excluding (the somewhat rare) chords with ninths, elevenths, and so on. I exclude these because the intervals ninth and above are excluded from classical chord labeling in which I am trained, not that identifying ninths is hard to do, but because frequently, a ninth's function is that of an accidental.

An event in the probability scheme X is a song X_k with state space $C_k \subseteq C$, but here we will refer to songs by their title, because there are (unfortunately) so few of them.

2.2.2 The Difference between Two Markov Chains

Before we assume that entropy, discussed in the next chapter, is the best way to evaluate the difference in certainty between Markov chains, let us actually take the difference, i.e., subtraction, of two transition matrices that we will later analyze entropically.

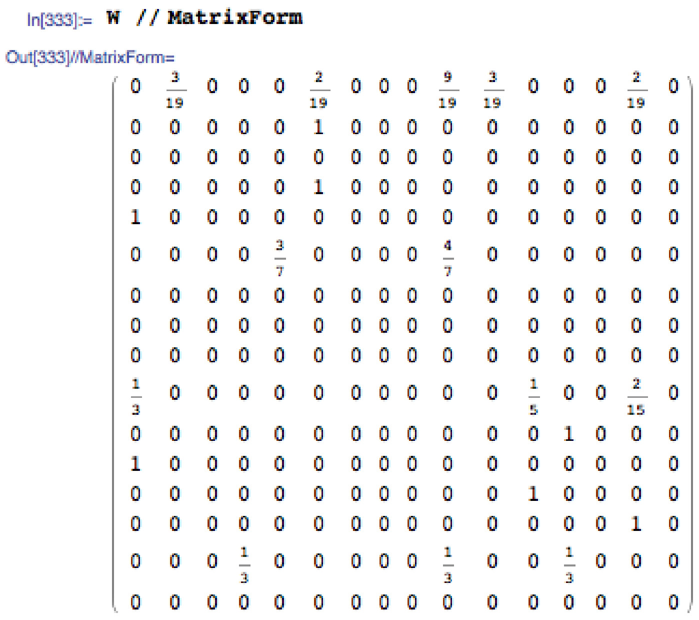
Consider the transition matrices for the chord progressions of the songs "Tell Me Why" from the Beatles' *A Hard Day's Night* and "When I'm 64" off of their *Sgt. Pepper's Lonely Hearts Club Band.*



```
In[336]:= T // MatrixForm
Out[336]//MatrixForm=
```

$$\begin{pmatrix}
0 & 0 & 0 & \frac{1}{20} & 0 & \frac{3}{4} & 0 & 0 & \frac{1}{20} & 0 & 0 & 0 & 0 & 0 & \frac{1}{20} & 0 \\
\frac{1}{6} & 0 & 0 & \frac{1}{6} & 0 & \frac{2}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{6} & 0 & 0 & 0 & 0 & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{4}{19} & 0 & 0 & \frac{1}{19} & 0 & \frac{14}{19} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{18}{19} & \frac{1}{19} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Figure 2.1: “Tell Me Why” transition matrix.

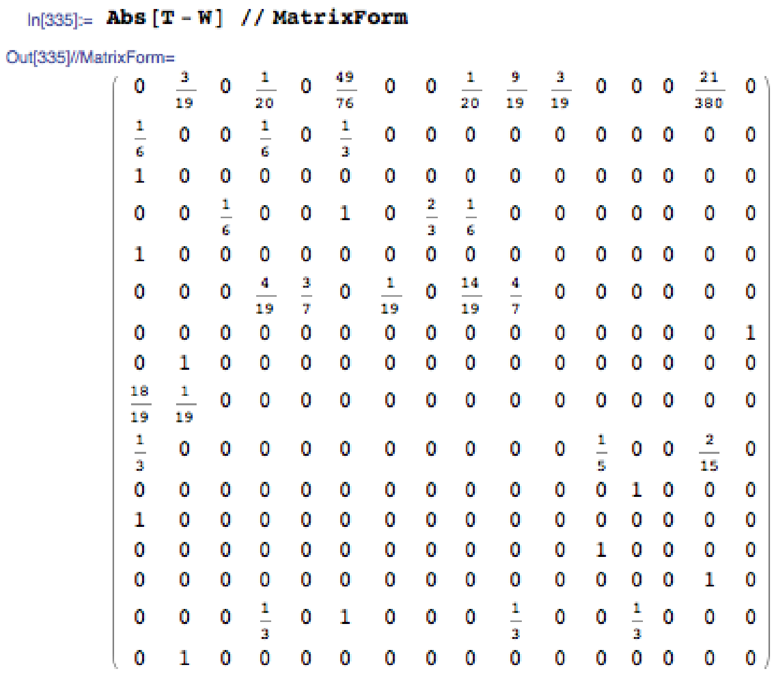


```
In[333]:= W // MatrixForm
Out[333]//MatrixForm=
```

$$\begin{pmatrix}
0 & \frac{3}{19} & 0 & 0 & 0 & \frac{2}{19} & 0 & 0 & 0 & \frac{9}{19} & \frac{3}{19} & 0 & 0 & 0 & \frac{2}{19} & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{3}{7} & 0 & 0 & 0 & 0 & \frac{4}{7} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{5} & 0 & 0 & \frac{2}{15} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Figure 2.2: “When I’m 64” transition matrix.

Because the state space of “When I’m 64” contains states that “Tell Me Why” does not, and vice versa, both have rows of all zeros so that their matrices are the same sizes and may be subtracted from one another. We can accept this because transitions with 0 probability do not contribute or take away from their entropy rate, since $H(0) := 0$. Their absolute difference is



```
In[335]:= Abs[T - W] // MatrixForm
Out[335]//MatrixForm=
```

$$\begin{pmatrix}
0 & \frac{3}{19} & 0 & \frac{1}{20} & 0 & \frac{49}{76} & 0 & 0 & \frac{1}{20} & \frac{9}{19} & \frac{3}{19} & 0 & 0 & 0 & \frac{21}{380} & 0 \\
\frac{1}{6} & 0 & 0 & \frac{1}{6} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{6} & 0 & 0 & 1 & 0 & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{4}{19} & \frac{3}{7} & 0 & \frac{1}{19} & 0 & \frac{14}{19} & \frac{4}{7} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{18}{19} & \frac{1}{19} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{5} & 0 & 0 & \frac{2}{15} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & \frac{1}{3} & 0 & 1 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Figure 2.3: The absolute difference between the two transition matrices.

Clearly, there is not much meaning to be derived from this matrix, besides that the two songs have wildly different transitions! However, we will see that their entropy rates (defined in the next chapter) are very close: "Tell Me Why" has an entropy rate of 0.2438 10-ary digits (or simply, "symbols") per time interval, and "When I'm 64" has an entropy rate of 0.2822 11-ary digits per time interval.

Therefore, merely subtracting two transition matrices does not indicate similarity in their levels of certainty.

In summary, this chapter exists to further the reader's understanding of the bridge connecting chord progressions and Markov chains. The definitions of properties surrounding them are of minimal importance: the concept of ergodicity (the freedom each state has to be able to move to any other state) is likely the only one you should worry about taking away from them. Other than that, a Markov chain is simply an ordered sequence of states, so their movement or behavior is characterized by conditional probabilities, but since a Markov process is "memoryless" (i.e., it has the Markov property), these conditional probabilities only take one state (the present state) as "given".

Chapter 3

Information, Entropy, and Chords

3.1 What is information?

The study of communication requires certain statistical and probabilistic parameters that the theory of information provides. Many consider the mathematical definition of information unintuitive. Indeed, information and uncertainty have a directly proportional relationship: the more that is left up to chance in a message, the more information a message must contain in order to be transmitted unambiguously. However, they are in fact opposites, because the more uncertain a message, the less information it contains. For compression purposes as well as efficient coding methods, knowing how much information is contained in a given message is highly relevant and vital to developing a well-functioning model.

Unary is the simplest language for humans and computers alike to "speak". It is also the least efficient. If we communicated in unary, where our language was only "0" (permitting a space bar), every distinct word in our language would be translated to a distinct length of 0's. Therefore, binary is the second simplest, but with it we can actually communicate in half the amount of total symbols. We use binary digits (and the base-2 logarithm) exclusively when we talk about information theory.

Consider a string of independent binary digits. We should treat it as a queue of symbols where each has one of two actions, one with probability p, and the other with probability $1 - p$. There are two possibilities, 0 and 1, for each symbol in the string, and 2^N possible strings, where N is the length of the string. Since a symbol, if not 0, is 1, each symbol's identity can be realized with only one (binary) question: "Is the digit a 0?" Therefore, there are at most $\log_2(2^N) = N$ yes/no questions required to uniquely identify each string.

Claude Shannon discusses the encoding of the English alphabet in his transcendental work, *The Mathematical Theory of Communication* (1948). Counts are made from some given set of text, say a dictionary, to see the relative frequencies of the letters 'e' and 'a' versus 'z' or 'x'. Noting that 'e' is one of the more common symbols, we would want to represent it with a shorter string of code than one we might choose for 'z' or 'x', the least occurring symbols, so that a message could be transmitted as quickly as possible.

Coding theory (a subfield of information theory) aids the construction of binary code to the extent that a string of letters translated into binary can be read unambiguously. Shannon worked with Robert Fano to develop Shannon-Fano coding, which ensures that the code assigned to a character in the alphabet does not exceed its optimal binary code length by more than 1 [6]. David Huffman was a student of Fano at MIT in 1951, when he outdid his professor in the creation of a maximally efficient binary code with his own algorithm, named Huffman coding. Examine the following Huffman code for some letters in English:
Then the string "promise me this" is

Character	Relative Frequency	Huffman Code
space	4	111
a	4	010
e	4	000
f	3	1101
h	2	1010
i	2	1000
m	2	0111
n	2	0010
s	2	1011
t	2	0110
l	1	11001
o	1	00110
p	1	10011
r	1	11000
u	1	00111
x	1	10010

Figure 3.1: Huffman binary code, based on the relative frequencies of 16 characters from the English alphabet.

10011110000011001111000101100011101110001110110101010001011

and is unambiguous, so it cannot be read any other way. Let us walk through the string to show this. The first symbol is either "100", "1001", or "10011" since strings can only be between 3 and 5 symbols in length. From our given set of 16 symbols, only one of them begins with "100", and that is "p" with 10011. So we advance 5 symbols to the next string, any of "110", "1100", or "11000", and see that "r" is the only symbol that fits one of those possibilities. Note that we do not pick "f" even though it begins with "110" because it is four bits and the last one is "1". We advance another 5 symbols to "001", "0011", or "00110", and unambiguously decide it is "o". And so on.

It is easiest to start at the beginning of the string, but it can also be unambiguously determined at any other point based on the construction of the binary coding scheme. Consider starting at the middle with "111" (the last 3 bits of "0111" corresponding to "m") and deciding that is "space". The next is either "000", "0001", or "00011", and we decide it is "e" (mapped to "000"). Then the previous string is any of "110", "1110", or "01110", and we realize that this does not correspond to any of the binary codes assigned to our given vocabulary, so our initial starting point was not at the beginning of a symbol's code.

Huffman developed his encoding methods such that any set of symbols (not just 16, although the fact that 16 is a power of 2 does contribute to the maximal efficiency of the given encoding scheme) could be assigned unambiguous bit representations, enabling shorter signals that do not utilize all 27 characters of the English alphabet to be transmitted even faster—in fact, as briefly as binary will allow. In this way, entropy measures the compressibility of a set of symbols, when we know the probability of correctly transmitting them.

In this thesis, the letters of our alphabet will be pitches, such that the cardinality of our alphabet is 12. Therefore, words are chords, sentences are musical phrases, and paragraphs are an entire song. We could extend the idea of supersets in musical classification even further, chapters to albums, books to discographies, but there it gets a little fuzzy[1].

Information theory and logarithms have a deeply seeded connection, namely within the function of entropy. Entropy is the uncertainty, or information, contained in a message. It is a measure of the likeliness that a sent message will not convey the intended meaning it was given, and it is equal to the number of bits[2] per symbol of improbability that the message contains. It follows that a scenario in which all of the events are independent and equally likely contains the most entropy of any scenario, for there are the most possible messages: if you wanted to transmit a "6" from tossing a fair die, there would only be a 1/6 probability of that happening.

To further color this important characteristic, consider a probability scheme A with outcomes A_1 and A_2, say heads and tails, $p(A_1) = p(A_2) = \frac{1}{2}$. Also, say that we are transmitting these H's and T's at 1000 symbols per second[3]. If we want to transmit the sequence "HTHH", in a model where H and T are transmitted with the same probability and they are the only symbols we can transmit, the sequence "HTHH" is just as likely to be transmitted as "THTT" or "THHT" or any sequence of 4 symbols. Thus, we expect only half of the symbols to be transmitted correctly, left entirely up to chance (entropy), so we say that our source is actually transmitting 0 bits of information (i.e., the certainty that the transmitted symbol was the one the source intended to transmit). This is the same as calculating $-\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = -\left(\frac{1}{2}\cdot -1 + \frac{1}{2}\cdot -1\right) = 1$ bit/symbol, and multiplied by our transmission rate of 1000 symbols/second is 1000 bits/second. Our given message and the resulting message after going through this probabilistic encoding process work against each other, so

[1]We could also call chords phonemes or morphemes, as in linguistics, because they contribute to the meaning/emotion of a song, as well as are designed keeping the restrictions of the apparatus (instrument) in mind.

[2]John Tukey's shortening of "binary digit" [6]

[3]What would the transmission rate of music be? If you find yourself asking this, observe that a "rate" implies that is only a scaling factor, and therefore the entropy rate is all that is important here.

we subtract this rate from our transmission rate to determine how much information is actually being processed, which is $1000 - 1000 = 0$ bits/symbol as we expected [6].

Now, for a probability scheme B, also with 2 outcomes B_1 and B_2 such that $p(B_1) = 0.99$ and $p(B_2) = 0.01$, we would expect our system to transmit information at a high rate, and have low entropy. We might guess that the transmission of information would have a rate of 990 bits/second, but this does not take into account the independence of the symbols and replacement. We find that the entropy is equal to $-\sum_{i=1}^{2} p(B_i)\log_2 p(B_i) \;= -\left(0.99\log_2 0.99 + 0.01\log_2 0.01\right) = 0.081$ bits/symbol, multiplied by our sampling rate is 81 bits per second, and see that the system is actually transmitting at $1000 - 81 = 919$ bits/second of information.

This is the independent case, at least. It is rare that two events in this world, however, are unrelated to each other. Therefore, it is even more rare for the tth symbol in a sequence to be independent of what came before (the first through $(t-1)$st symbols, or perhaps, as in Markovian processes, just the $(t-1)$st symbol), and through this we can transcribe any sort of progression for the purpose of effective communication, in some harmonic and witty arrangement. The words of a single statement can be jumbled up to the point of nonsense, but together, their meaning is likely still translatable. But, like intelligent and correct speech, the genius of music lies in perfectly aligned rhythm between the melodic line and lyric, the synthesis of several instruments or occasionally the choice to go solo, and, the most easily analyzable characteristic, the clever patterning of harmony to structure the aforementioned.

So, by nature, it is more interesting to look at cases involving conditional or joint probabilities. When we want some way of probabilistically describing our Markov chain, information theory chimes in with many measures and applications to coding. Entropy is one of these measures. It is interesting to compare the entropies of different systems with one another, and see how distinct in "propensity for chaos" two systems are. It is, in actuality, rather useless to look at the translation of this measure to its binary code-length, but the binary form does correspond to code containing however many symbols required to represent the number of chords in the vocabulary, say d, so the measure we will find is easily scaled by $\frac{\log(2)}{\log(d)}$. Therefore, it does not really matter if we look at the binary interpretation or the d-ary one, for some positive integer d. For this reason, all logarithms used in this thesis, unless otherwise noted, are base-2 and should be considered the "binary representation".

3.2 The Properties of Entropy

First, we will remind the reader of the definition of a probability scheme, as stated in section 2.2.

Definition: Finite probability scheme. Given a finite sample space S, we define an **event** $A_i \subseteq A \subseteq S$ to be any set of m **outcomes** $a_1, a_2, \ldots, a_m \in A_i$ with respective probabilities $p(a_j) = q_j$, $\sum_j q_j = 1$. The probability of an event A_i is given by $p(A_i) = p_i$, $\sum_{i=1}^{n} p_i = 1$, where n is the number of events in A. The subsets A_i are disjoint $\forall i$, and partition all of S [4]. We call the matrix

$$A = \begin{bmatrix} A_1 & A_2 & \dots & A_n \\ p_1 & p_2 & \dots & p_n \end{bmatrix}$$

the **finite probability scheme** of S.[4]

As alluded to earlier, the function of entropy gives the average number of (binary) bits per symbol needed to correctly transmit a string of these symbols, where each has a given probability of being correctly transmitted. The following lengthy proof evolves the notion of entropy to the function $H(X) = -\sum_{x \in X} p(x) \log p(x)$.

Theorem. Let $H(p_1, p_2, ..., p_n)$ be a function defined for any integer n and for all values $p_1, p_2, \dots, p_n$, which are the probabilities of the subsets $A_1, A_2, \dots, A_n$ in a finite probability scheme A such that $p_k \geq 0$ and $\sum_{k=1}^{n} p_k = 1$. If, for any n, this function is continuous $\forall p_i$, and if

1. H is maximized when $p_k = \frac{1}{n}$ $\forall k$ (the characteristic we just showed);

2. For the product scheme AB, $H(AB) = H(A) + H(B|A)$; and

3. $H(p_1, p_2, \dots, p_n, 0) = H(p_1, p_2, \dots, p_n)$, i.e., adding an impossible event to a scheme does not change H;

then

$$H(A) = H(p_1, p_2, \dots, p_n) = -\lambda \sum_{k=1}^{n} p_k \log p_k,$$

where λ is a positive constant.

Proof. Let $H\left(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}\right) = \phi(n)$. We will show that $\phi(n) = \lambda \log(n)$, where $\lambda > 0$.

Since H is maximized when $p_k = \frac{1}{n}$ $\forall k$ by the first property, we have

$$\begin{aligned} \phi(n) &= H\left(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}\right) \\ &= H\left(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}, 0\right) \\ &\leq H\left(\frac{1}{n+1}, \frac{1}{n+1}, \dots, \frac{1}{n+1}, \frac{1}{n+1}\right) \\ &= \phi(n+1), \end{aligned}$$

[4]Note that it is possible for $A_i = A$, in which case the finite probability scheme features a_1 through a_m in its first row and q_1 through q_m in its second.

i.e., ϕ is a non-decreasing function of n. Now, let m, r be positive integers. Let $S_1, S_2, \ldots, S_m$ be mutually exclusive schemes all containing r equally likely events, i.e., for all k,

$$\begin{aligned} H(S_k) &= H\left(\frac{1}{r}, \frac{1}{r}, \ldots, \frac{1}{r}\right) \\ &= \phi(r). \end{aligned}$$

Then, since all the schemes are independent by their disjointness,

$$\begin{aligned} H(S_1 S_2) &= H(S_1) + H(S_2) \\ &= \phi(r) + \phi(r) \\ &= 2 \cdot \phi(r) \\ &\vdots \\ H(S_1 \cdot S_2 \cdot \ldots \cdot S_m) &= H(S_1) + H(S_2) + \ldots + H(S_m) \\ &= m \cdot \phi(r). \end{aligned}$$

But $S_1 \cdot S_2 \cdot \ldots \cdot S_m$ (the "product scheme") contains r^m equally likely events, so $H(S) = \phi(r^m)$. Hence,

$$\phi(r^m) = m \cdot \phi(r).$$

Now, let s and n be arbitrary numbers such that

$$r^m \leq s^n \leq r^{m+1}.$$

Then

$$\begin{aligned} m \log r \leq & \quad n \log s & < (m+1) \log r \\ \frac{m}{n} \leq & \quad \tfrac{\log s}{\log r} & < \frac{m+1}{n}. \end{aligned}$$

Since ϕ is nondecreasing,

$$\phi(r^m) \leq \phi(s^n) \leq \phi(r^{m+1})$$

which, because $\phi(r^m) = m \cdot \phi(r)$, is equivalent to

$$\begin{aligned} m \cdot \phi(r) \leq & \quad n \cdot \phi(s) & \leq (m+1) \cdot \phi(r) \\ \frac{m}{n} \leq & \quad \tfrac{\phi(s)}{\phi(r)} & \leq \frac{(m+1)}{n}. \end{aligned}$$

Then, since $\frac{m+1}{n} - \frac{m}{n} = \frac{1}{n}$,

$$\left| \frac{\phi(s)}{\phi(r)} - \frac{\log s}{\log r} \right| \leq \frac{1}{n}.$$

Since n is arbitrarily large,

$$\begin{aligned} \frac{\phi(s)}{\log s} &= \frac{\phi(r)}{\log r} \\ \phi(r) &= \frac{\phi(s)}{\log s} \log r \\ &= \lambda \log r \end{aligned}$$

where λ is a constant. Then

$$\phi(n) = \lambda \log n$$

by the arbitrariness of r and s. Since $\log n \geq 0$, and $\phi(n) \geq 0$, it is the case that $\lambda \geq 0$, and so we have proved our assertion. □
In fact, λ is simply the scalar $\frac{\log 2}{\log x}$ for a language with x symbols, because the log function above is base-2.

To prove the general case, let A and B be two dependent schemes such that A consists of n events with probabilities $p_1, p_2, ..., p_n$ where

$$p_k = \frac{g_k}{g}$$

(so, here it is not necessary that $p_k = \frac{1}{n} \forall k$), and

$$\sum_{k=1}^{n} g_k = g,$$

and B consists of g events, which are divided into n subsets each containing g_i events, $1 \leq i \leq n$. Then, for the event $A_k \subseteq A$, we have g_k events in the kth group of B all with a $\frac{1}{g_k}$ probability, and all other events in the 1st, 2nd, ... , (k-1)st, (k+1)st, ... , nth subsets of B have probability 0. Then

$$\begin{aligned} H_k(B) &= H\left(\frac{1}{g_k}, \frac{1}{g_k}, ..., \frac{1}{g_k}\right) \\ &= \phi(g_k) \\ &= \lambda \log g_k. \end{aligned}$$

Noting that the sum of all the p_k is 1, this heeds the conditional entropy

$$\begin{aligned} H(B|A) &= \sum_{k=1}^{n} \frac{g_k}{g} H_k(B) \\ &= \sum_{k=1}^{n} \frac{g_k}{g} \lambda \log g_k \\ &= \lambda \sum_{k=1}^{n} \frac{g_k}{g} \log g\ p_k \\ &= \lambda \sum_{k=1}^{n} p_k (\log g + \log p_k) \\ &= \lambda \log g + \lambda \sum_{k=1}^{n} p_k \log p_k. \end{aligned}$$

Now, consider the product scheme $A \cdot B = AB (= A \cap B)$. The total number of possible events in AB is g, which is equal to the number of events in B, and since each event is equally likely to occur,

$$\frac{p_k}{g_k} = \frac{1}{g}.$$

Therefore,

$$H(AB) = \phi(g) = \lambda \log g.$$

Now, by the second property, and from above,

$$\begin{aligned} H(AB) &= H(A) + H(B|A) \\ &= H(A) + \lambda \sum_{k=1}^{n} p_k \log p_k + \lambda \log g. \end{aligned}$$

Then we can subtract $\lambda \log g$ from both sides to get

$$0 = H(A) + \lambda \sum_{k=1}^{n} p_k \log p_k.$$

Thus,

$$\begin{aligned} H(A) &= H(p_1, \ldots, p_n) \\ &= -\lambda \sum_{k=1}^{n} p_k \log p_k, \end{aligned}$$

and we have arrived at the function of entropy. Since H is continuous by assumption, this holds for any value of p_i. This completes the proof. □

To recapitulate, we proved the functional form of entropy first for the maximal case, when all the $p_i = \frac{1}{n}$. We did this mostly by the nondecreasing nature of ϕ. Then, to prove the general case, we used the second property of H and the conditional entropy (defined in the next section) $H(B|A)$ to make the function $H(A) = -\sum_{k=1}^{n} p_k \log p_k$ to emerge ($\lambda = 1$ in the binary case).

Now let us backtrack and define $H(B|A)$ and $H(AB)$ more precisely.

3.3 Different Types of Entropy

Events where not all outcomes are independent of one another, as in a Markov chain, have different entropy than the independent case. All of these terms are defined for events, but do extend to probability schemes.

3.3.1 Conditional Entropy

For simplicity of notation, we will write $P(AB)$ to denote $P(A \cap B)$ and $P(A \cdot B)$. Recall that the conditional probability $P(A|B)$, where the event A may or may not be independent of event B, is

$$P(B|A) = \frac{P(AB)}{P(A)},$$

with the common notation of $A \cap B = AB$ demonstrated. When A and B are independent, i.e. disjoint, i.e. unrelated, $P(B|A) = P(B)$.

Definition: Conditional Entropy. The **conditional entropy** of c_j given c_i, where c_i and c_j are distinct chords, is

$$H(c_j|c_i) = -p(c_j)p(c_j|c_i) \log p(c_j|c_i),$$

where $p(c_i) = \sum_{k=0}^{n} p(c_k)p(c_i|c_k)$, $p(c_j|c_i)$ is the probability of transitioning from c_i to c_j, and $p(c_j)$ is the probability of observing c_i.

We will see in the definition of joint entropy that this quantity is also

$$H(c_j|c_i) = -p(c_i, c_j) \log p(c_j|c_i),$$

i.e., $p(c_i, c_j) = p(c_j)p(c_j|c_i)$.

Now, the conditional entropy $H(D|c_i)$, where C and D are events and $|C| = m$, $|D| = n$, is conceptually the amount of entropy in D given the probability of one chord c_i occurring.

$$H(D|c_i) = -\sum_{j=1}^{n} p(d_j|c_i) \log_2 p(d_j|c_i)$$

and $H(D|C)$ where the entirety of A is given is

$$\begin{aligned} H(D|C) &= -\sum_{i=1}^{m} p(c_i)H(D|C = c_i) \\ &= -\sum_{i=1}^{m}\sum_{j=1}^{n} p(c_i)p(d_j|c_i)\log_2 p(d_j|c_i). \end{aligned}$$

As in probability theory, C and D are mutually exclusive if and only if $P(D|C) = P(D)$, so likewise,

$$(C \cup D) = \emptyset \Longleftrightarrow H(D|C) = H(D).$$

It should follow intuitively that the conditional entropy when the events are dependent is always less than the conditional entropy of two independent events, since knowing any information about C should only aid the correct transmission of D when the outcomes of C affect the outcomes of D.

Proposition.

$$H(D|C) \leq H(D),$$

with equality only when C and D are independent.

Proof. Consider the function $f(x) = x \log x$. It is convex for $x > 0$. For some $\lambda_i \geq 0, \sum \lambda_i = 1$,

$$\sum_{i=1}^{n} \lambda_i f(x_i) \geq f\left(\sum_{i=1}^{n} \lambda_i x_i\right).$$

This is Jensen's inequality. But since entropy is negative, the inequality flips so that

$$-\sum_{i=1}^{n} \lambda_i f(x_i) \leq -f\left(\sum_{i=1}^{n} \lambda_i x_i\right).$$

Letting $\lambda_i = p(c_i)$, $|C| = m$, $|D| = n$, and $x_i = p_{c_i}(d_j)$,

$$\begin{aligned}
H(D|C) &= -\sum_{i=1}^{m} p(c_i) \sum_{j=1}^{n} p_{c_i}(d_j) \log p(d_j|c_i) \\
&= -\sum_{j=1}^{n} \sum_{i=1}^{m} p(c_i) f(p(d_j|c_i)) \\
&\leq -\sum_{j=1}^{n} f\left(\sum_{i=1}^{m} p(c_i) p(d_j|c_i)\right) \\
&= -\sum_{j=1}^{n} f(p(d_j)) \\
&= -\sum_{j=1}^{n} p(d_j) \log p(d_j) \\
&= H(D).
\end{aligned}$$

This proves our assertion. □

Entropy is an *a priori* measure to guide efficient coding because we do not measure the actual entropy of a potential code, in practice; we measure only those that came before it. It follows that the expected value of the entropy of the event C, $E[H(C)]$, is just $H(C)$. Similarly, the conditional expectation $E[H(D|C)] = H(D|C)$.

3.3.2 Joint Entropy

The joint entropy of two events C and D is also the entropy of their intersection, CD. We consider pairs of outcomes (c_i, d_j) with probabilities $p(c_i, d_j) = p(c_i)p(d_j|c_i)$ from the definition of conditional probability given above, and they form the function of joint entropy as follows:

$$H(C, D) = H(CD) = H(C) + H(D|C).$$

Proof. Since $p(c_i, d_j) = p(c_i)p(d_j|c_i)$ $\forall i, j$,

$$\begin{aligned}
H(C, D) &= -\sum_{i} \sum_{j} p(c_i, d_j) \log p(c_i, d_j) \\
&= -\sum_{i} \sum_{j} p(c_i) p(d_j|c_i) [\log p(c_i) + \log p(d_j|c_i)] \\
&= -\sum_{i} p(c_i) \log p(c_i) - \sum_{i} p(c_i) \sum_{j} p(d_j|c_i) \log p(d_j|c_i) \\
&= H(C) + H(D|C).
\end{aligned}$$

□

Before we get to the entropy of Markov chains, note that, for the outcomes c, d, and e,

$$\begin{aligned} p(c,d,e) &= p(c,d)p(d,e|c) = p(c,d)\frac{p(d)p(e|d)}{p(d)} = p(c,d)p(e|d) \\ &= p(c)p(d|c)p(e|d). \end{aligned}$$

3.4 The Entropy of Markov chains

The probability of a given chord progression X is simply [4]

$$P(X) \;=\; p_1^{p_1N} p_2^{p_2N} \cdots p_n^{p_nN}$$

where N is the length of the progression and p_iN is the number of occurrences of the ith chord in the sequence X. We measure the **entropy rate of a Markov chain** X by quantifying the entropy per chord $X_t = c_i$ in X

$$\begin{aligned} H(X) &= \sum_{X_t \in X} p(X_t)H(X_t) = -\sum_{i=0}^{m}\sum_{j=0}^{m} p(c_i)p(c_j|c_i)\log p(c_j|c_i) \\ &= -\sum_{i=0}^{m} p(c_i) \sum_{j=0}^{m} p(c_j|c_i)\log p(c_j|c_i), \end{aligned}$$

c_i, $c_j \in C$, the state space of X, whose cardinality $|C| = m$. This is the entropy per chord[5] because it multiplies each of the inner sums by the probability of observing the initial chord, and each of those probabilities ($p(c_i) = p_i$) is found by dividing the number of its occurrences (p_iN) by the total number of chords observed, N. Hence, the entropy of an entire Markov chain is just NH.

3.4.1 How to interpret this measure

Since the entropy of an entire Markov chain is NH, where N is the length of the sequence and H is the entropy of each chord, systems with more observations (larger N) will tend to have more entropy than just as chaotic systems with a smaller N. Therefore, in characterizing a system by its entropy, it is clearer to use simply H to describe it.

The entropy of a Markov chain has the same form as the entropy of two conditional events, because by the Markov property, we only consider two states in the calculation of transition probabilities. We already knew that a Markov chain was simply a sequence of conditional probabilities, so it should be unsurprising that its entropy is modeled after this conditional character.

[5] In fact, our entropy rate is also the "entropy per time interval." Since there is no rhythm associated with our Markov chain, these time intervals are likely not uniform (though it is certainly possible for a song to uniformly change over time).

It seems possible that two systems, one with a dictionary (chord vocabulary) many times larger than the other, could have the same amount of entropy per chord, or for the larger to be more certain (lower entropy) than the smaller. This happens when the system with a larger vocabulary contains transitions with a higher amount of certainty than the smaller one's higher transitions, and/or the larger contains transitions that have more uncertainty than the smaller one's lower transitions. This isn't strictly the case, but is shown by the two systems with transition matrices S and L $[p(s_i) = \frac{1}{4} \;\; \forall i, p(l_i) = \frac{1}{5} \;\; \forall i]$

$$S = \begin{bmatrix} 0 & .5 & .25 & .25 \\ .75 & 0 & .25 & 0 \\ .25 & .25 & 0 & .5 \\ .5 & .5 & 0 & 0 \end{bmatrix}$$
$$L = \begin{bmatrix} 0 & .5 & .25 & 0 & .25 \\ 0 & .25 & 0 & 0 & .75 \\ .25 & .25 & 0 & .5 & 0 \\ .5 & .5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

The smaller system (with state space C) has entropy 1.20282 bits/chord, and the larger (with state space D) has entropy 0.96226 bits/chord.[6] However, the two systems have the exact same amount of total entropy (4.81128 bits) because the fifth chord in D has no uncertainty associated with it, and thus contributes no entropy to the system. Therefore, when we find the total entropy of S,

$$|C| \cdot H(S) = 4 \cdot \text{-} \left(\frac{1}{4} H(row_1) + \frac{1}{4} H(row_2) + \frac{1}{4} H(row_3) + \frac{1}{4} H(row_4) \right),$$

and of L,

$$|D| \cdot H(L) = 5 \cdot \text{-} \left(\frac{1}{5} H(row_1) + \frac{1}{5} H(row_2) + \frac{1}{5} H(row_3) + \frac{1}{5} H(row_4) + \frac{1}{5} H(row_5) \right),$$

the two quantities are clearly going to be equal ($H(row_5) = 0$) when we increase N by 1, increase the denominator of the uniformly distributed probabilities by 1, and keep the sum of the entropies of each row the same. Note that in both cases, the probability distribution is uniform, i.e., $p(x) = 1/n$ for all x, where n is the number of distinct states.

We will see in the Beatles diverse music several different styles, and thereby measure the strictness of these classifications.

[6]Note that these entropy rates are actually higher than 1, so they are greater than the maximum entropy rate: the entropy rate of a completely random (binary) system. To normalize this, we should scale the binary entropy rate by $\frac{\log(2)}{\log(4)}$ for the entropy rate of S, making it 0.60141 "nits" per chord, and by $\frac{\log(2)}{\log(5)}$ for L, so, 0.41442 nits per chord.

3.5 Expectations

Studying the entropy rate of a sequence of symbols is one way of determining their unique source. For example, entropy has been applied to the four gospels of the Bible to establish whether or not they were indeed written by four different authors—and it was found that they were not![7] Music and art seem to borrow from previous material more shamelessly than literature, and artists can have a repertoire containing a vast range of styles and instrumentation, whereas authors are usually confined to one language, and cannot adapt their style so easily as a band of four or more musicians can in parallel. Even when the members of a band do not collectively write their songs, each of their individual styles and preferences in instrumentation do appear in the frequency spectrum. Therefore, I suspect that the application of entropy to music will not specify artist, but rather, origin of the style.

As one listens to more and more music under distinct classifications, one starts to learn its language This is true especially of music that gains popularity, primarily because it is easy to discover. "What makes a hit?" is the big question amongst songwriters—either because they want the satisfaction that goes with creating something that has appeal and makes money, or because they want to discern what it is about certain songs that appeals to them, and what turns them off. With a probabilistic model like a Markov chain, our quantified findings cannot be completely off base with describing the tendencies of a set of music.

This is only because we name chords relative to their key, I would like to point out. From the light spectrum, say that we treated "colors" like the scales of the key, [R, O, Y, G, B, I, V] = [1, 2, 3, 4, 5, 6, 7]. Do humans perceive the combination of orange ($\hat{2}$) and violet ($\hat{7}$) the same way they do red ($\hat{1}$) and indigo ($\hat{6}$)? Not in the way that humans cannot distinguish the root of sonic intervals, though interestingly, those of us that do have perfect pitch usually perceive distinct pitches as a unique color in the spectrum!

But, maybe we should challenge the notion that **perfect pitch** (the ability to detect pitch, not limited to those on the keyboard) is a gift received at birth. Allegedly, the percent of humans with perfect pitch is something like 0.001%, and I've met just one in my lifetime (who did in fact have the color-pitch synesthesia). After this thesis, I can tell that *I* am much better at estimating pitch than before, though because I sing, I know almost the exact range that my voice can handle. The acquisition of **relative pitch** (the ability to detect melodic intervals) from years of music theory, plus knowing one's vocal range, gives me all the tools I need to identify a note with a smaller amount of error.

Now, I expect popular music to have a noticeably lower entropy rate than any other style, and popular musicians alike, just because popular songs are simple and predictable on average. The Beatles were certainly interesting, but what gave them that extra spark was their fun rhythms and lyrics, not necessarily complex chord progressions. However, I would say that few other musicians used the same sets of chords and transitions between chords that the Beatles did in their songs. Since I

[7] R. Crandall, private communication (2009).

have played Beethoven and music from other classical musicians (though, I seem to have a propensity for the Romantics), as well as some knowledge of music theory in jazz, I know that many accidentals are used in the two styles. Therefore, I expect a higher amount of entropy in these systems as well as a greater state space of chords. I do not expect the data from our bandpass filter to be noise-free, because I could not familiarize myself enough with the discrete Fourier transform to the point that I could trust the frequencies it picks out to be fundamental frequencies, and not a harmonic thereof.

The Beatles released 11 studio albums (EP's) to the U.K. in the following order[8]:

1963 :	*Please Please Me*
1963 :	*With the Beatles*
1964 :	*A Hard Day's Night*
1964 :	*Beatles for Sale*
1965 :	*Help!*
1965 :	*Rubber Soul*
1966 :	*Revolver*
1967 :	*Sgt. Pepper's Lonely Hearts Club Band*
1967 :	*The Magical Mystery Tour*
1968 :	*The Beatles (White Album)*
1969 :	*Yellow Submarine*
1969 :	*Abbey Road*
1969 :	*Let It Be*

All of the albums feature fairly "unexpected" chords and chord progressions, but the first few (before *Rubber Soul*, or arguably pre-*Help!*) contain many songs that sound remarkably alike, some of which are (highly predictable) standards not written by The Beatles. Then, in the albums *Rubber Soul* through *Abbey Road*, The Beatles expanded their musical vocabulary, or at least differentiated between songs on the same album more. But their final album, *Let It Be*, they seemed to return to their old sound. Because of this, comparing the affectation of a few of these albums should be a good test of our hypothesis that music under a loose classification, like "jazz," has a more uniform Markov chain (or, it will be more like throwing a die), and music under a strict classification has a greater certainty about it. Below are a few songs with their chord names and, beneath those, the Roman numerical equivalent, to show you just how their harmonic vocabulary grew—and with *Let It Be*, down-sized. I predict

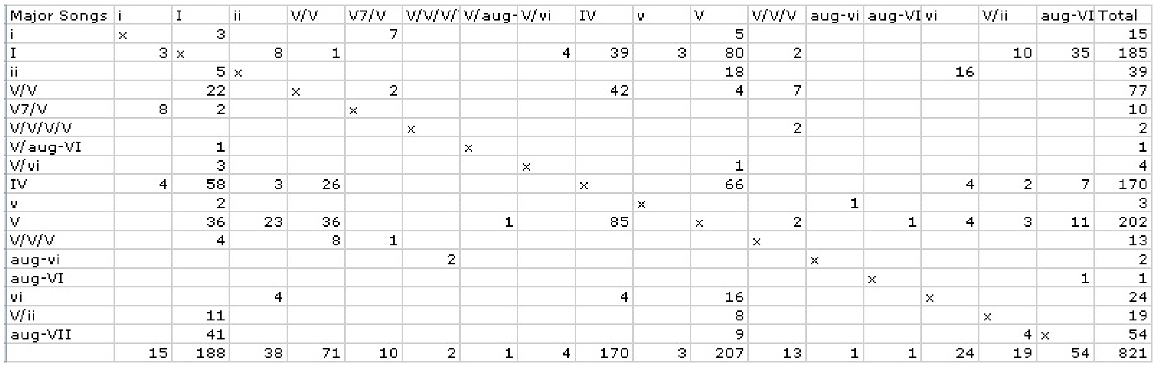
[8]I have ordered this list as best I can, for the following anachronisms and peculiarities are true: (1) The Beatles released *The Magical Mystery Tour* to the U.K. in 1976 but to the U.S. upon its completion in 1967; (2) *Yellow Submarine* features many orchestral songs from the film of which it is a soundtrack, but "Only a Northern Song," "It's All too Much," "Hey Bulldog," and "All Together Now" do not appear on any other album; and finally, (3) *Abbey Road* was actually the last album The Beatles recorded but was released before *Let It Be* because of issues with Phil Spector's direction.

that the album *Sgt. Pepper's* will have less songs with similar progressions, indicating more uncertainty and a more vague notion of what the album's sound is, and *Please Please Me* will have the more certainty about it, and a small vocabulary of chords.

3.6 Results

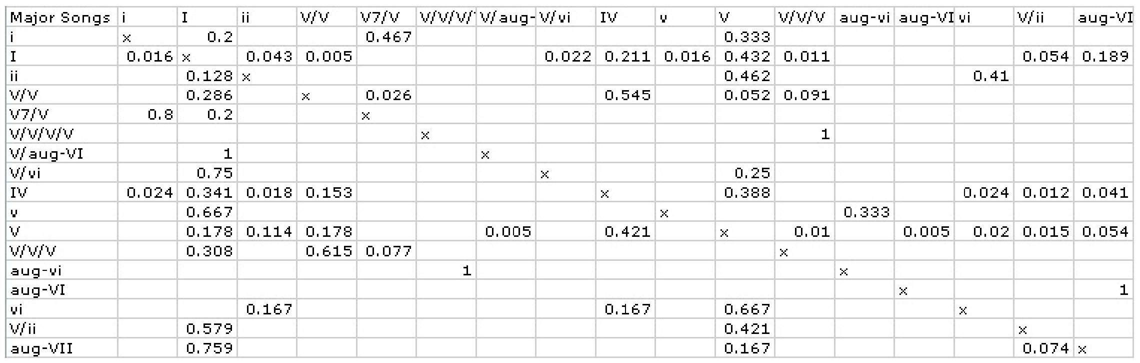
3.6.1 Manual

The easiest to calculate (by hand) was Wire's *Pink Flag* (1977) because of its small state space in every song, and in its entirety. This was the first example I did, and I divided the album into a set for those songs in a major key, and the remainder into a set for those in a minor key. I noted 16 songs in a major key in *Pink Flag*, and 5 songs in a minor key. I accidentally skipped analysis of the last song on the album because it was not imported onto my computer. I deciphered all of the chords from ear on guitar, because every (admittedly free) source of chord transcription I looked at for the album contained a multitude of errors. The entire album is only 35:19 long, less the last song, meaning the average song length is around 100 seconds, and that each of them consist of only a few chords, thus a minimal amount of transitions to count up. *Pink Flag* has been my favorite album for going on a year, so the project was nothing short of exciting. Still, it took at least 20 hours (including mistakes) to acquire the following transition matrices and corresponding entropy rates. Blank entries are zeros.



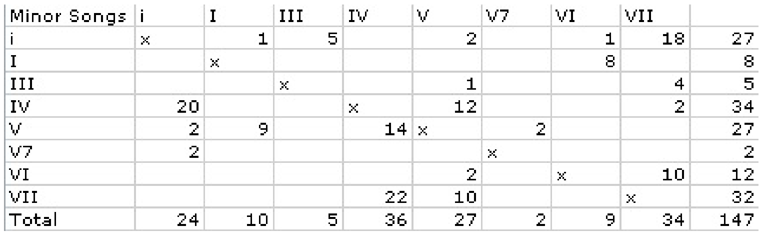
Major Songs	i	I	ii	V/V	V7/V	V/V/V/	V/aug-	V/vi	IV	v	V	V/V/V	aug-vi	aug-VI	vi	V/ii	aug-VI	Total
i	x	3			7						5							15
I	3	x	8	1				4	39	3	80	2				10	35	185
ii		5	x								18				16			39
V/V		22		x	2				42		4	7						77
V7/V	8	2			x													10
V/V/V/V						x						2						2
V/aug-VI		1					x											1
V/vi		3						x			1							4
IV	4	58	3	26					x		66				4	2	7	170
v		2								x			1					3
V		36	23	36			1		85		x	2		1	4	3	11	202
V/V/V		4		8	1							x						13
aug-vi						2							x					2
aug-VI														x			1	1
vi			4						4		16				x			24
V/ii		11									8					x		19
aug-VII		41									9					4	x	54
	15	188	38	71	10	2	1	4	170	3	207	13	1	1	24	19	54	821

Figure 3.2: The counts of observations of each chord transition within major songs from Wire's *Pink Flag*.



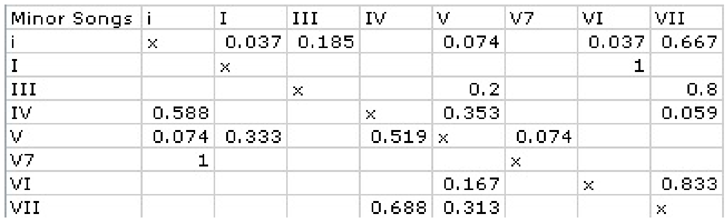
Major Songs	i	I	ii	V/V	V7/V	V/V/V/	V/aug-	V/vi	IV	v	V	V/V/V	aug-vi	aug-VI	vi	V/ii	aug-VI
i	x	0.2			0.467						0.333						
I	0.016	x	0.043	0.005				0.022	0.211	0.016	0.432	0.011				0.054	0.189
ii		0.128	x								0.462				0.41		
V/V		0.286		x	0.026				0.545		0.052	0.091					
V7/V	0.8	0.2			x												
V/V/V/V						x						1					
V/aug-VI		1					x										
V/vi		0.75						x			0.25						
IV	0.024	0.341	0.018	0.153					x		0.388				0.024	0.012	0.041
v		0.667								x			0.333				
V		0.178	0.114	0.178			0.005		0.421		x	0.01		0.005	0.02	0.015	0.054
V/V/V		0.308		0.615	0.077							x					
aug-vi						1							x				
aug-VI														x			1
vi			0.167						0.167		0.667				x		
V/ii		0.579									0.421					x	
aug-VII		0.759									0.167					0.074	x

Figure 3.3: The corresponding transition matrix of the major songs from Wire's *Pink Flag*.



Minor Songs	i	I	III	IV	V	V7	VI	VII	
i	x	1	5		2		1	18	27
I		x					8		8
III			x		1			4	5
IV	20			x	12			2	34
V	2	9		14	x	2			27
V7	2					x			2
VI					2		x	10	12
VII				22	10			x	32
Total	24	10	5	36	27	2	9	34	147

Figure 3.4: The counts of observations of each chord transition within minor songs from Wire's *Pink Flag*.



Minor Songs	i	I	III	IV	V	V7	VI	VII
i	x	0.037	0.185		0.074		0.037	0.667
I		x					1	
III			x		0.2			0.8
IV	0.588			x	0.353			0.059
V	0.074	0.333		0.519	x	0.074		
V7	1					x		
VI					0.167		x	0.833
VII				0.688	0.313			x

Figure 3.5: The transition matrix of the 5 minor songs from Wire's *Pink Flag*.

Next, I attempted to analyze every song the Beatles ever recorded, and find their (as well as their albums') entropy rate. I got through five albums, *A Hard Day's Night*, *Help!*, *Sgt. Pepper's Lonely Hearts Club Band*, *Abbey Road*, and *Let It Be* before I got to a point of disorganization at which I was convinced I was double-checking my work for the fourth time. Fortunately, the true entropy rates revealed results according to my intuitions: their music from before 1965 and the age of psychedelics is almost 20% less entropic than the music they produced between 1965 and 1969, until returning to a more rock 'n' roll, true-to-their-beginnings sound with *Let It Be*.

Then, I found the transition matrices of each song on the five aforementioned albums, and calculated their entropy rates. Additionally, I picked one song from each album and measured its entropy rate, and chose the song with the closest T to the ratio $\frac{T_{album}}{|Songs|_{album}}$, i.e., the total number of states that a given album transitions to over the number of songs in that album. Interestingly, this ratio was just about 70 in all cases—not to mention that T for each of the albums released between 1965 and 1969 are all within a range of 4!

Finally, I analyzed a sonata for piano by Beethoven with which I was very familiar: the decadent Piano Sonata No. 8, Op. 13 ("Pathetique"). I analyzed all three movements from the score by hand in under three hours, and the number of distinct states meant listing out the transition probabilities was a nightmare. However, this project took less time overall than did Wire, perhaps only because I had done it before, but also because I did not have to backtrack and triple-check the chords by ear since I had the score right in front of me.

I only wish to reproduce here the first of the three movements of "Pathetique," based on the amount of space the following (single) transition matrix consumes (Figures 3.6 and 3.7 at the end of this section. The binary entropy rate of each chord appears in the rightmost column, AU, and at the bottom we have the entropy rate of the entire movement. At the very bottom-right corner of the matrix (cell AU45), I found the binary entropy of the system by multiplying through by the total number of states T in the progression. Adjusted for the size of its dictionary $N = 43$, the true entropy rate of the first movement of Beethoven is 0.3933457 43-ary digits per chord.

The results are given in the following tables. The first shows the binary entropy rate, bits per symbol, of the given classification, as well as the size of sequence T and the size of vocabulary N; the second shows the entropy rate of the given classification adjusted for the size of the classification's chord vocabulary, N; and the rest break down each of the five albums by song, and show their true entropy rates only. Remember that the base-N entropy rates are simply the binary entropy rate scaled by $\lambda = \frac{\log(2)}{\log(N)}$. Only then is one able to compare their entropy rates with one another.

Classification	**Binary Entropy Rate**	T	N
"Tell Me Why" from *A Hard Day's Night*	0.8097371	78	10
"You're Going Lose That Girl" from *Help!*	0.6606718	69	12
"When I'm 64" from *Sgt. Pepper's*	0.9763367	69	11
"Oh! Darling" from *Abbey Road*	1.0961483	67	11
"Two of Us" from *Let It Be*	0.7129891	76	9
A Hard Day's Night	2.0693784	993	38
Help!	2.3278379	962	28
Sgt. Pepper's	2.3199219	960	30
Abbey Road	2.4892385	958	39
Let It Be	1.5592536	855	24
First mvmt. of Beethoven	2.1343981	481	43
Second mvmt. of Beethoven	1.1371771	141	23
Third mvmt. of Beethoven	2.0581977	321	41
All of Beethoven	2.6348037	943	54
Major *Pink Flag* songs	1.9627405	821	17
Minor *Pink Flag* songs	1.4553366	142	8

Classification	**True Entropy Rate**
"Tell Me Why" from *A Hard Day's Night*	0.2437552
"You're Going Lose That Girl" from *Help!*	0.1828974
"When I'm 64" from *Sgt. Pepper's*	0.2822246
"Oh! Darling" from *Abbey Road*	0.3168579
"Two of Us" from *Let It Be*	0.2249230
A Hard Day's Night	0.3943230
Help!	0.4842243
Sgt. Pepper's	0.4727886
Abbey Road	0.4709648
Let It Be	0.3357670
First mvmt. of Beethoven	0.3933457
Second mvmt. of Beethoven	0.2513897
Third mvmt. of Beethoven	0.3841676
All of Beethoven	0.4578376
Major *Pink Flag* songs	0.4801855
Minor *Pink Flag* songs	0.4851122

Observe that *Help!* is the album with the highest (true) entropy rate of the 5 albums, yet it contains the song ("You're Going to Lose That Girl") with the lowest entropy rate of the songs analyzed in the previous sample. Intrigued by this anomaly, I found the (true) entropy rate of each song from *Help!*, in addition to the other albums.

Song from *Help!*	**Entropy Rate**	T	N	T/N
"Help!"	0.2152401	48	9	5.33
"The Night Before"	0.2086572	101	12	8.42
"You've Got to Hide Your Love Away"	0.3887134	103	6	17.17
"I Need You"	0.2571845	60	10	6
"Another Girl"	0.3201159	79	8	9.875
"You're Going to Lose That Girl"	0.1828974	69	12	5.75
"Ticket to Ride"	0.3713393	49	6	8.17
"Act Naturally"	0.3685859	40	4	10
"It's Only Love"	0.2893418	53	7	7.57
"You Like Me Too Much"	0.2863428	76	10	7.6
"Tell Me What You See"	0.2325980	91	6	15.17
"I've Just Seen a Face"	0.4595894	60	6	10
"Yesterday"	0.1616613	88	13	6.77
"Dizzy Miss Lizzy"	0.4251817	56	3	18.67

Funny that "Yesterday," the most covered song of all time[9], had one of the lowest entropy rate of any classification analyzed. Looking at its progression, it is extremely

[9] "Yesterday" has the Guinness World Record for the most recorded cover versions (or, renditions), with over 3,000 documented.

periodic, repeating itself four times with only a few chords sticking out. Perhaps its low entropy has some correlation to musicians' desire to produce a version of it themselves.

Noting that some of the songs on *Help!* with a relatively high ratio $T/N \geq 9$ also had the higher entropy rates, and the songs with relatively low $T/N < 9$ have the lower entropy rates, I regressed the two linearly to see if there was some relationship. A linear regression of the entropy rate against T/N shows that only 32.33% of the data can be explained in a linear relationship, i.e., $R^2 = 0.3233$. Removing the outlier "Ticket to Ride" improves R^2 to 0.3786—not very much. The remainder of the albums also show that the relationship is not statistically significant.

Song from *A Hard Day's Night*	**Entropy Rate**	T	N	T/N
"A Hard Day's Night"	0.3693949	97	10	9.7
"I Should Have Known Better"	0.2472442	127	6	21.17
"If I Fell"	0.2393726	65	10	6.5
"I'm Happy Just to Dance with You"	0.3958344	109	9	12.11
"And I Love Her"	0.2493109	58	14	4.14
"Tell Me Why"	0.2437552	78	10	7.8
"Can't Buy Me Love"	0.3201860	54	6	9
"Any Time at All"	0.4268255	63	7	9
"Things We Said Today"	0.1092970	84	10	8.4
"When I Get Home"	0.2711269	51	8	6.38
"You Can't Do That"	0.2244769	54	10	5.4
"I'll Be Back"	0.1881344	58	11	5.27

These "poppy" songs were meant to put the Beatles on the map (i.e., the charts), and they did just that. "Can't Buy Me Love" and the title track were the biggest hits of the two, and their entropy rates are very close to one another. The dramatic "I'm Happy Just to Dance with You" and "Any Time at All" sound quite similar, as well as "You Can't Do That," but the third does not really compare with the high entropy rates of the first two. Additionally, "Things We Said Today" and "If I Fell" are both slow, sadder songs, but their entropy rates are also not very similar. This indicates that the classification of "dramatic" or "sad" could be a loose one.

Song from *Sgt. Pepper's Lonely Hearts Club Band*	**Entropy Rate**	T	N	T/N
"Sgt. Pepper's Lonely Hearts Club Band"	0.4314825	51	7	7.28
"With a Little Help from My Friends"	0.2695691	85	9	9.44
"Lucy in the Sky with Diamonds"	0.2344558	103	10	10.3
"Getting Better"	0.1764410	62	7	8.86
"Fixing a Hole"	0.3147904	101	8	12.63
"She's Leaving Home"	0.1885085	66	8	8.25
"Being for the Benefit of Mr. Kite"	0.2084119	105	13	8.08
"Within You Without You"	0	20	2	10
"When I'm Sixty-Four"	0.3037015	69	11	6.27
"Lovely Rita"	0.4010419	80	11	7.27
"Good Morning, Good Morning"	0.2307419	87	5	17.4
"Sgt. Pepper's Lonely Hearts Club Band (Reprise)"	0.4132076	33	7	4.71
"A Day in the Life"	0.2617964	97	11	8.82

The "psychedelic" style of rock music appears most on *Sgt. Pepper's*, especially within "Lucy in the Sky with Diamonds," "Fixing a Hole," "Being for the Benefit of Mr. Kite," and "A Day in the Life." All of these have entropy rates ranging from 0.2084119 and 0.3147904, expressing the wide scope of the classification, and thus a less defined rate of entropy. The title track and its reprise have very close rates, unsurprisingly.

Song from *Abbey Road*	**Entropy Rate**	T	N	T/N
"Come Together"	0.1673155	26	6	4.33
"Something"	0.1637179	67	14	4.79
"Maxwell's Silver Hammer"	0.3043236	105	11	9.55
"Oh! Darling"	0.2658129	67	11	6.09
"Octopus' Garden"	0.2658129	75	9	8.33
"I Want You (She's So Heavy)"	0.1278756	83	12	6.92
"Here Comes the Sun"	0.3368234	113	8	14.13
"You Never Give Me Your Money"	0.2606540	95	20	4.75
"Sun King"	0.1526444	47	10	4.7
"Mean Mr. Mustard"	0.3628259	25	5	5
"Polythene Pam"	0.1980518	72	7	10.29
"She Came in through the Bathroom Window"	0.2109171	44	8	5.5
"Golden Slumbers"	0.1194754	29	10	2.9
"Carry That Weight"	0.2089544	36	11	3.27
"The End"	0.1450917	50	12	4.17
"Her Majesty"	0.1975524	25	10	2.5

Surprisingly, the B-side of *Abbey Road*[10] ("Here Comes the Sun" through "The End", so, excluding "Her Majesty" which is something of an afterthought, featuring Paul solo on acoustic guitar) is not very similar in entropy rates, ranging from very low (0.1194754 nits/second for the brief "Golden Slumbers") to somewhere in the middle

[10]For some reason, "Because" was not included in the reference [27], where I found the chords for the rest of the songs, so it is left out here.

of our findings (0.3628259 nits/second for the even briefer “Mean Mr. Mustard”). However, “Carry That Weight” and “You Never Give Me Your Money” are fairly close in entropy rate, and the former recapitulates the latter in chord progression. Also, “Maxwell’s Silver Hammer” and “Octopus’ Garden” are very close in entropy rate (0.3043236 and 0.2658129), and it is said that Ringo Starr’s lack of songwriting skills led “Octopus’ Garden” to sound very close to “Maxwell’s.”

Song from ***Let It Be***	**Entropy Rate**	T	N	T/N
“Two of Us”	0.2249230	76	9	8.44
“I Dig a Pony”	0.1964138	98	7	14
“Across the Universe”	0.2702220	54	8	6.75
“ I Me Mine”	0.1732533	66	11	6
“Dig It”	0.3154649	36	3	12
“Let It Be”	0.3054991	127	6	21.17
“Maggie Mae”	0.3647870	12	4	3
“I’ve Got a Feeling”	0.1565072	123	8	15.38
“One After 909”	0.4193111	40	4	10
“The Long and Winding Road”	0.2858642	83	8	10.38
“For You Blue”	0.3738869	57	4	14.25
“Get Back”	0.3372992	82	5	16.4

Now, “Act Naturally,” “Dizzy Miss Lizzy,” “One After 909,” “For You Blue,” and “Get Back” are very similar and reminiscent of the 12-bar blues chord progression. The 12-bar blues is I-I-I-I-IV-IV-I-I-V-IV-I-I, and it repeats. Note that all of these songs have similar entropy rates to each other. The chord progressions in these songs, and the entropy rate of the 12-bar blues, is

Song	**Chord Progression**	**Entropy Rate**
12-bar blues	I-I-I-I-IV-IV-I-I-V-IV-I-I	0.4206198
“Act Naturally”	I-I-IV-IV-I-I-V-V-I-I-IV-IV-V-V-I-I	0.3685859
“Dizzy Miss Lizzy”	I-I-I-I-IV-IV-I-I-V^7-IV-I-V^7	0.4251817
“One After 909”	I^7-I^7-I^7-I^7-I^7-I^7-IV^7-IV^7-I^7-V^7-I^7-I^7	0.4193111
“For You Blue”	I^7-IV^7-I^7-I^7-IV^7-IV^7-I^7-I^7-V^7-IV^7-I^7-V^7	0.3738869
“Get Back”	I-I-I-I-IV-IV-I-I-I-I-I-I-IV-IV-I-I	0.3372992

In order to analyze the 12-bar blues by the methods applied to the others, I condensed it only to the transitions, making it simply I-IV-I-V-IV-I, so, 6 chords. The entire chord progressions of none of the songs were reproduced above, especially in the case of “Get Back,” and, excluding “Get Back” and “Dizzy Miss Lizzy,” they all contain the secondary dominant chord V/V or V^7/V. Since I treated V^7 and V as if they were unrelated, we can probably identify some of the error resulting from that here, since “Act Naturally” does not have a single seventh chord.

George Harrison wrote “I Need You,” “Within You Without You,” “Something,” “Here Comes the Sun,” and “For You Blue” in the songs studied. These entropy rates range from 0 to 0.3738869, and excluding “Within You Without You” and the standard blues song “For You Blue,” 0.1637179 to 0.3368234, the rates of his songs from *Abbey Road.* Paul McCartney's songs (of which there are too many to list) have a larger range of entropy rates [0.1194754 ("Golden Slumbers") to 0.4595894 ("I've Just Seen a Face")] than John Lennon's [0.1278756 for "I Want You (She's So Heavy)" to 0.3887134 for "You've Got to Hide Your Love Away"], perhaps indicative of his greater knowledge of music than any of the Bealtes, and John's tendency to be more bluesy and almost American in sound. It would be interesting to see how each of their solo albums compare with these ranges.

Song from ***Pink Flag***	**Entropy Rate**	T	N	T/N
"Reuters"	0.2246579	31	3	10.33
"Field Day for the Sundays"	0.0863660	18	5	3.6
"Three Girl Rhumba"	0.1224055	70	6	11.67
"Ex-Lion Tamer"	0.2901141	93	5	18.6
"Lowdown"	0	16	4	4
"Start to Move"	0.0517911	39	5	7.8
"Brazil"	0.1861653	33	5	6.6
"It's So Obvious"	0.2349974	41	3	13.67
"Surgeon's Girl	0.2483058	8	3	2.67
"Pink Flag"	0	5	2	2.5
"The Commercial"	0	16	4	4
"Straight Line"	0.5190109	52	4	13
"106 Beats That"	0.1759036	22	8	2.75
"Mr. Suit"	0.3097678	114	3	38
"Strange"	0.1186875	72	4	18
"Fragile"	0.3144970	57	5	11.4
"Mannequin"	0.2455547	83	5	16.6
"Different to Me"	0.3868528	16	6	2.67
"Champs"	0.0582476	59	6	9.83
"Feeling Called Love"	0.3114073	60	3	20
"12XU"	0.2766039	66	6	11

By my rules, a binary system of chords cannot have any entropy, since I call the probability of transitioning from a chord to itself 0, making the probability of transitioning to the other chord 1. However, "Within You Without You" is a sitar song all in C, simply moving from C to C^7, and "Pink Flag" (the song) is only 5 chords in duration, and completely predictable in my opinion (and certainly periodic). "The Commercial" is also periodic, giving it zero entropy.

Wire's music is called "post-punk," a movement that began in the 1970s and is still thriving, especially in Portland. The only one of these that I can connect in feel to the Beatles' music is the love song "Feeling Called Love," for its vocabulary is strictly I, IV, and V. It is not quite the 12-bar blues, nor even a blues song arguably, but the small vocabulary and "typical pop song" quality gives it an entropy rate similar to the blues songs analyzed above.

3.6.2 Automatic

And then, there is **BandPower.app**. Its speediness deserves more praise than its effectiveness, but we have shown that even without smoothing its output, it can accurately derive the frequencies C through B present in a spectrum (Figure 3.6). After smoothing (Figure 3.8), we can see that its effectiveness is more readily apparent.

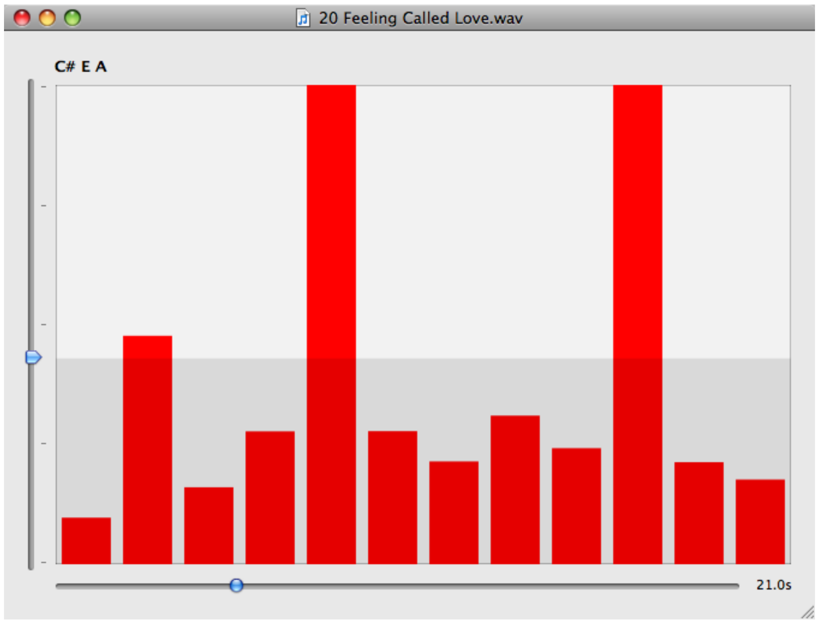


Figure 3.6: The application **BandPower** displays a bar graph of powers for each pitch, pitches on the x-axis, power in watts on the y. It has two sliding levers, one for each axis, where the x-axis lever controls time, and the other sets a threshold power value at which to print out the pitches that meet that value, displayed above. Here, it is correctly identifying an A Major chord at time 21.0s in the Wire song "Feeling Called Love" from *Pink Flag*.

Although the chords are easily visualizable in this plot, the data they model is not as easy to retrieve, and I ran out of time before I was able to examine the effectiveness of the smoothed bandpass powers.

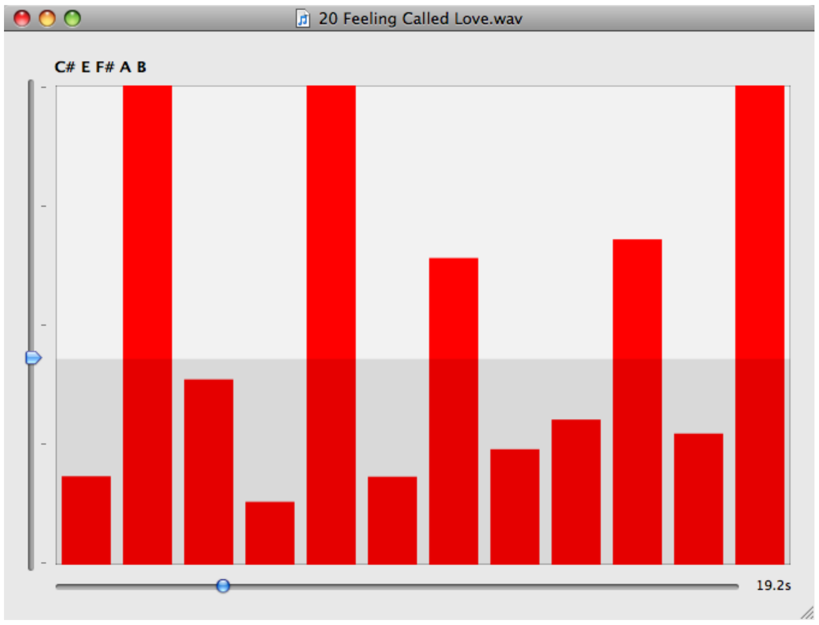


Figure 3.7: Sometimes, the three most powerful pitches at a given time do not create a triad, nor resemble the actual chord being played (here, the actual chord is A Major, but the three most powerful pitches are C♯, E, and B, which we would probably name C♯ minor[7]).

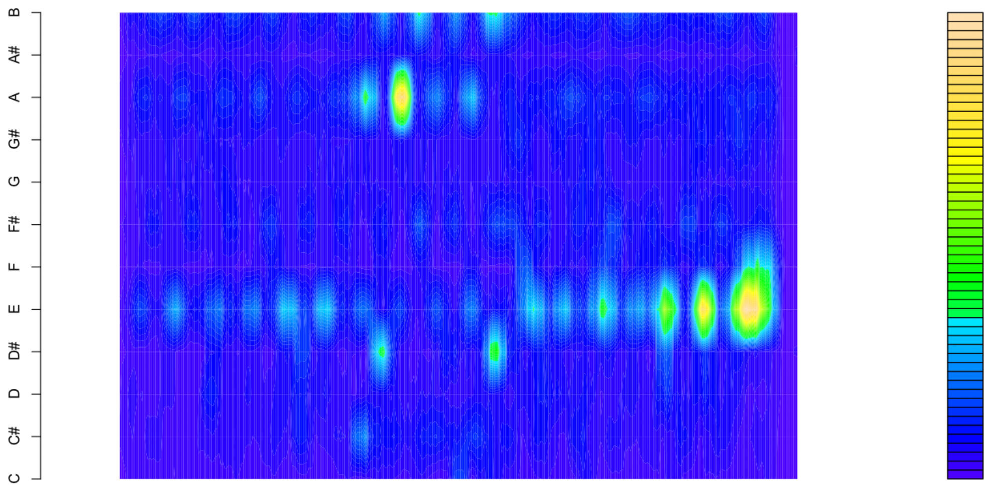
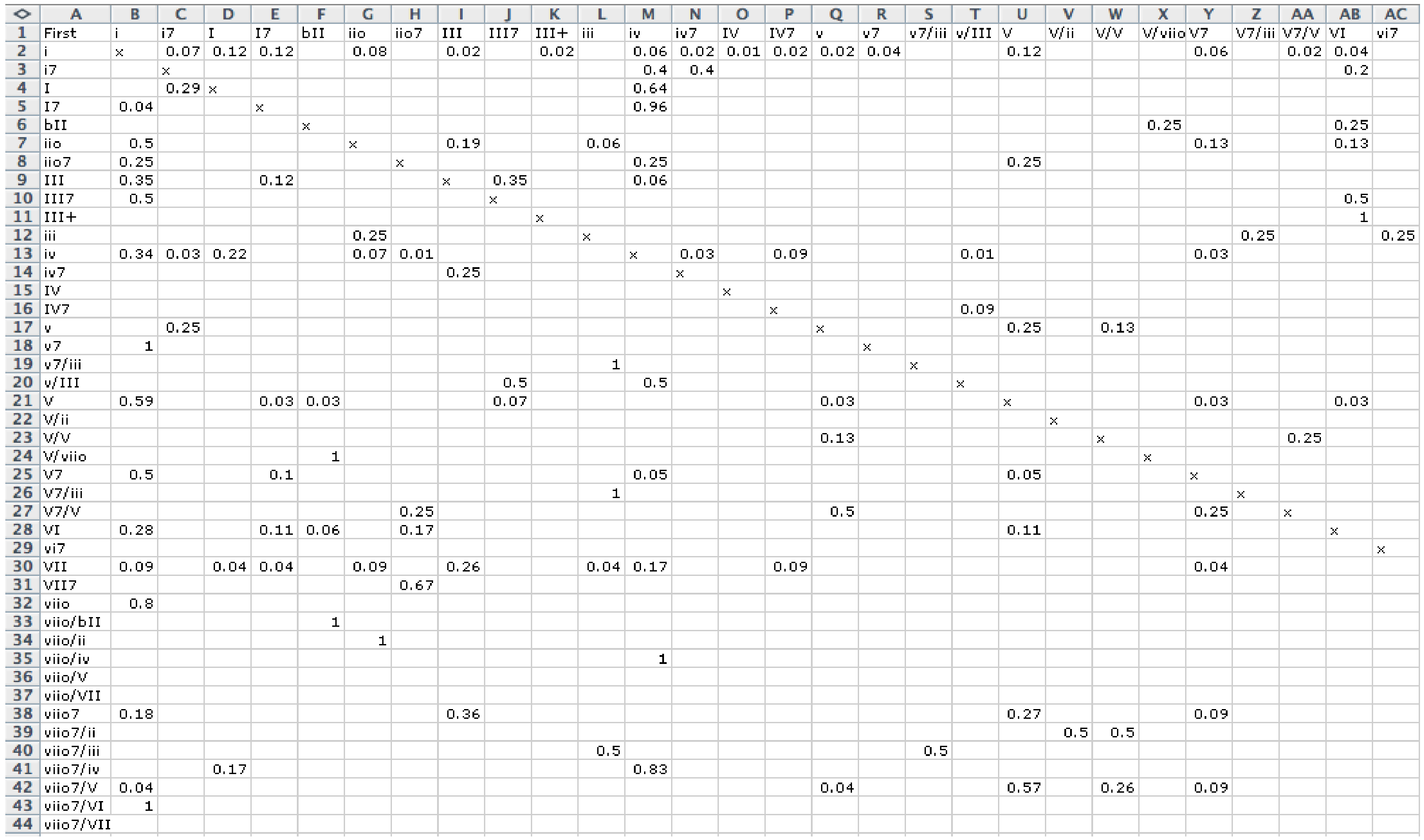
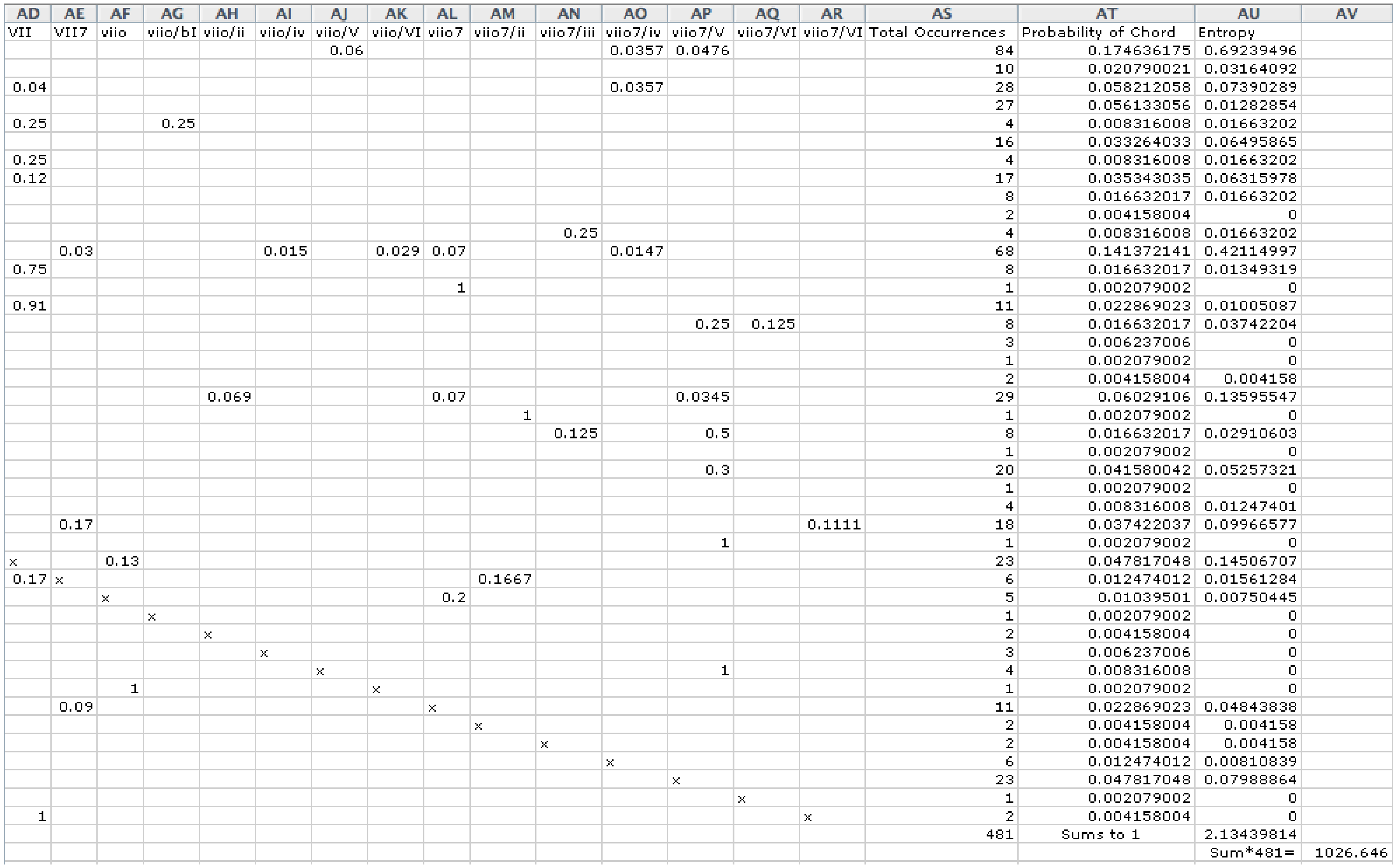


Figure 3.8: The contour plot of the Wire song "Feeling Called Love" from *Pink Flag*.



◇	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	First	i	i7	I	I7	bII	iio	iio7	III	III7	III+	iii	iv	iv7	IV	IV7	v	v7	v7/iii	v/III	V	V/ii	V/V	V/viio	V7	V7/iii	V7/V	VI	vi7
2	i	x	0.07	0.12	0.12		0.08		0.02		0.02		0.06	0.02	0.01	0.02	0.02	0.04			0.12				0.06		0.02	0.04	
3	i7		x										0.4	0.4														0.2	
4	I		0.29	x									0.64																
5	I7	0.04			x								0.96																
6	bII					x																		0.25				0.25	
7	iio	0.5					x		0.19			0.06													0.13			0.13	
8	iio7	0.25						x					0.25								0.25								
9	III	0.35			0.12				x	0.35			0.06																
10	III7	0.5								x																		0.5	
11	III+										x																	1	
12	iii						0.25					x														0.25			0.25
13	iv	0.34	0.03	0.22			0.07	0.01					x	0.03		0.09				0.01					0.03				
14	iv7								0.25					x															
15	IV														x														
16	IV7															x				0.09									
17	v		0.25														x				0.25		0.13						
18	v7	1																x											
19	v7/iii											1							x										
20	v/III									0.5			0.5							x									
21	V	0.59			0.03	0.03				0.07							0.03				x				0.03			0.03	
22	V/ii																					x							
23	V/V																0.13						x				0.25		
24	V/viio					1																		x					
25	V7	0.5			0.1								0.05								0.05				x				
26	V7/iii											1														x			
27	V7/V							0.25									0.5								0.25		x		
28	VI	0.28			0.11	0.06		0.17													0.11							x	
29	vi7																												x
30	VII	0.09		0.04	0.04		0.09		0.26			0.04	0.17			0.09									0.04				
31	VII7							0.67																					
32	viio	0.8																											
33	viio/bII					1																							
34	viio/ii						1																						
35	viio/iv												1																
36	viio/V																												
37	viio/VII																												
38	viio7	0.18							0.36												0.27				0.09				
39	viio7/ii																					0.5	0.5						
40	viio7/iii											0.5							0.5										
41	viio7/iv			0.17									0.83																
42	viio7/V	0.04															0.04				0.57		0.26		0.09				
43	viio7/VI	1																											
44	viio7/VII																												

Figure 3.9: A portion of the large transition matrix of Beethoven's Piano Sonata No. 8, Op. 13 ("Pathetique"), First Movement.



AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV
VII	VII7	viio	viio/bI	viio/ii	viio/iv	viio/V	viio/VI	viio7	viio7/ii	viio7/iii	viio7/iv	viio7/V	viio7/VI	viio7/VI	Total Occurrences	Probability of Chord	Entropy	
						0.06					0.0357	0.0476			84	0.174636175	0.69239496	
															10	0.020790021	0.03164092	
0.04											0.0357				28	0.058212058	0.07390289	
															27	0.056133056	0.01282854	
0.25			0.25												4	0.008316008	0.01663202	
															16	0.033264033	0.06495865	
0.25															4	0.008316008	0.01663202	
0.12															17	0.035343035	0.06315978	
															8	0.016632017	0.01663202	
															2	0.004158004	0	
										0.25					4	0.008316008	0.01663202	
	0.03				0.015		0.029	0.07			0.0147				68	0.141372141	0.42114997	
0.75															8	0.016632017	0.01349319	
								1							1	0.002079002	0	
0.91															11	0.022869023	0.01005087	
												0.25	0.125		8	0.016632017	0.03742204	
															3	0.006237006	0	
															1	0.002079002	0	
															2	0.004158004	0.004158	
				0.069				0.07				0.0345			29	0.06029106	0.13595547	
									1						1	0.002079002	0	
										0.125		0.5			8	0.016632017	0.02910603	
															1	0.002079002	0	
												0.3			20	0.041580042	0.05257321	
															1	0.002079002	0	
															4	0.008316008	0.01247401	
	0.17													0.1111	18	0.037422037	0.09966577	
												1			1	0.002079002	0	
x		0.13													23	0.047817048	0.14506707	
0.17	x								0.1667						6	0.012474012	0.01561284	
		x						0.2							5	0.01039501	0.00750445	
			x												1	0.002079002	0	
				x											2	0.004158004	0	
					x										3	0.006237006	0	
						x						1			4	0.008316008	0	
		1					x								1	0.002079002	0	
	0.09							x							11	0.022869023	0.04843838	
									x						2	0.004158004	0.004158	
										x					2	0.004158004	0.004158	
											x				6	0.012474012	0.00810839	
												x			23	0.047817048	0.07988864	
													x		1	0.002079002	0	
1														x	2	0.004158004	0	
															481	Sums to 1	2.13439814	
																	Sum*481=	1026.646

Figure 3.10: The rest of the transition matrix of Beethoven's Piano Sonata No. 8, Op. 13 ("Pathetique"), First Movement.

Conclusion

The application **BandPower.app** successfully picks out frequencies from the Western, 12-pitch scale, and allows one to view the frequencies excited at any time in the scale, and set some threshold value for their amplitude in order to retrieve the frequencies with the most power. By smoothing these results, we can throw out the unexplained (and not meaningful) pitches that often but irregularly appear, and keep the ones that locally and consistently appear. The fact that **BandPower.app** runs in less than 10 seconds (for any WAVE-format file converted within iTunes) is a real achievement, and I anticipate even smarter versions of it to come.

The values of the calculated entropy rates from musical examples did not match my intuitions, in general—but they did show me that it isn't the value, it is the range of values. Indeed, the blues songs studied had very similar entropy rates, and the psychedelic and emotional ones had a wider range of them. It is hard to say that this is the case for the classification in general, of course, since our sample size was so small, but is intriguing nonetheless.

If we did consider rhythm in our Markov chain, the entropy rates would most certainly turn out considerably different, and identify the true chaos of a classification much more accurately. Consider, for example, the song "Tomorrow Never Knows" by the Beatles, featuring just one chord: C Major. This song cannot be a Markov chain by my rules, because a row that sums to 0 in the transition matrix implies that the given state does not exist in C, the state space of the song. And, as mentioned before, songs with only 2 chords also have 0 entropy, since by my rules, the probability of transitioning to the other chord is automatically 1.

Another method which I did not implement that would improve the ranges of entropy rates toward my association of it with "strictness of musical classification" is the assignment of relative weights to those states that are closely related. For instance, the chords V, v, and V^7 function harmonically more closely than do ii, VI, and viio, in general. Songs containing chords such that some of them have the same root should therefore be adjusted to have lower entropy than other songs with the same amount of states but less of these "neighborhoods" of chords.

It is also possible that, generally speaking, music is not an ergodic, and thus not an aperiodic, Markov chain. Support for this claim comes from repetition of the same progressions for each verse, chorus, and bridge of virtually all popular songs—i.e., there exist patterns governed by rhythm (and hence periodic) within chord progressions. It makes sense that, because we ignored rhythm, the T above is too small in all cases, since it documents only times when we move from the present chord. However, if it were indeed not ergodic, this fact would not affect our calculated entropy rates.

In summary, musicians should be less hesitant to apply mathematics to songwriting and the art of music, just as we have to language with linguistics. Those whose devotion to music extends to seeking out the best training for their personal style, practicing for hours on end to perfect the fluidity of movement on their instrument(s), must be well acquainted with the probabilistic nature of their sound, whether they treat it mathematically, or with a qualitative, psychological sense of likelihood. I understand that boiling down music to probability theory is a little dry (and also insist that I have hardly "boiled things down" here, as I analyzed *only* chord progressions in music, something liberated by expression), but what if it is the solution to the great enigma that is our musical memory? It is no wonder that I applied many of the same techniques used in speech recognition to music, for music is a language, and those who love to speak it have one of the richest and most accessible ways of escaping from reality.

Appendix A

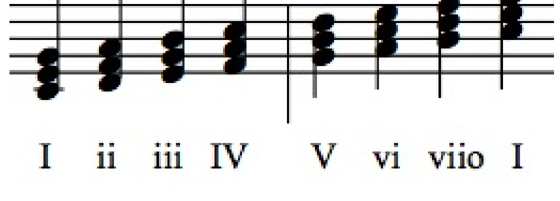
Music Theory

A.1 Key and Chord Labeling

A key is usually restricted to two classifications: **major** and **minor**. We call each note of the 7 in a key a **scale degree**, and mark them $\hat{1}$ through $\hat{7}$. The first scale degree ($\hat{1}$) of C Major, for instance, is C. The second is D, and it is 2 half steps (2 keys on a piano, one black, one white) above C. C is the only major key with no sharps or flats, so $\hat{1}$ through $\hat{7}$ is simply C-D-E-F-G-A-B. The difference between E and F and B and C is 1 half step, and the rest of the notes have an **interval** of two half steps (one **whole step**) between them. We can also write this using interval notation, where **m2**=half step, **M2**=whole step, so a major key is M2, M2, m2, M2, M2, M2, m2.

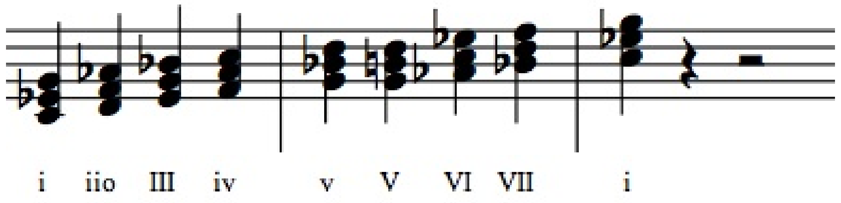
Minor keys are written with lowercase letters, and a minor is the same as C Major in pitches, but not the same in scale degrees. The key of a minor begins on C Major's sixth scale degree. For a minor, $\hat{1}$ through $\hat{7}$ is A-B-C-D-E-F-G, or in intervals, M2, m2, M2, M2, m2, M2, M2.

Now, chords are **triads** with a **root**, **third**, and **fifth** with scale degrees $\hat{n}$, $\widehat{(n+2)}$, and $\widehat{(n+4)}$[1], though they are not required to have the third or fifth to be called a chord. We call a single note a chord when the third and/or the fifth occurred just before, and the root is kept, or the third and/or the fifth will occur next in time. The "basic" chords of the C Major scale, meaning those not borrowed from any other key, are

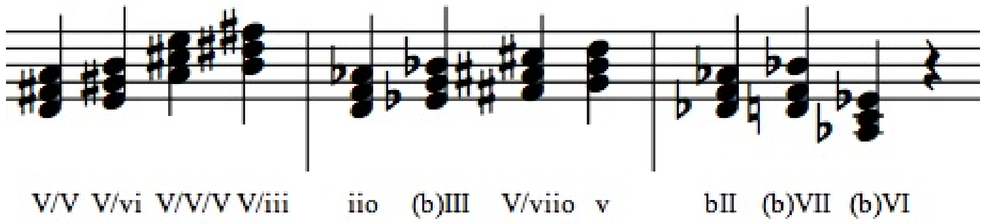


and c minor consists of the chords

[1]When $n + i > 7$, we mod it by seven and add 1, because there is no scale degree "$\hat{0}$".



We label them by scale degree, but also make them upper- or lowercase depending on their **tonality**. Tonality is limited to 4 classifications: major, minor, **diminished**, and **augmented**. The “viio” and “iio” above are diminished, and there are no augmented chords in either key. I, IV, and V are the only major chords within the major scale, and III, VI, and VII the only ones within the minor scale. However, it is common for both to **borrow** chords from each other, especially the dominant (V) from the major to the minor. The most commonly (and pretty much only) borrowed chords in C Major are

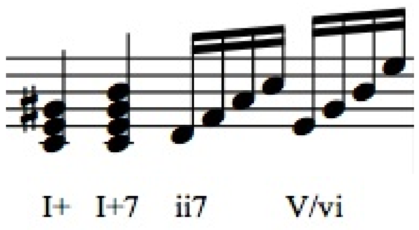


where iio, III, v, VI, and VII come from the **parallel minor**, c minor. Actually, all of the chords from c minor can occur within C Major, and vice versa. There is a flat symbol next to III, VI, and VII because they are rooted at scale degrees $\flat\hat{3}$, $\flat\hat{6}$, and $\flat\hat{7}$, but the flat is not necessary to write because in the parallel minor, that is the correct scale degree.

However, ♭II is the **Neapolitan** chord. It is the equivalent of V/V/viio, but V/viio rarely occurs and is considered slightly bad notation.

The notation “X/Y” means the Xth scale degree of key Y. Here, key Y is G because G is the fifth scale degree (the **dominant**) of C, so V/V is a D major chord, and we make the F sharp. Borrowed chords of this type almost always **resolve** (when the song returns to the original key) to the chord Y, but this is not a rule in modern pop music like it is in (some) classical music.

An augmented chord is also rare, but they do appear in a few Beatles songs and even classical compositions. Also, chords do not have to be played simultaneously, i.e., all three notes at once. They can be **arpeggiated**, which is depicted alongside an augmented chord:



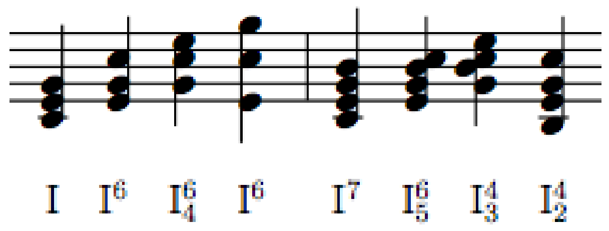
A.2 Ornaments

Not all pitches in a composition contribute to its chord structure/progression. These are called **ornaments**, and include **passing tones**, **neighboring tones**, **suspensions**, **anticipations**, and **escape tones**. Unfortunately, I did not have time (and apparently neither did many others in their constructions of automatic chord recognizers) to specify such ornaments and their misplacement in a chord progression. More unfortunately, this leads to some very bad specification of chord labels, but it is likely only a matter of deleting those that contain non-chord tones (ornaments) in most cases, since the chord should occur without non-chord tones within proximity—otherwise we are mislabeling it in the first place, and the non-chord tones are chord tones! However, with multiple voices and timbres as in most popular music, it is hard to say this is generally the case.

Neighbor tones (NT) are within a half or whole step of any chord tone, and return to the original tone. Passing tones (PT) move stepwise upward or downward from a chord tone, and keep within the key (no **accidentals**). Anticipations (A) (resolving upward) and suspensions (S) (resolving downward) initialize a chord with a non-chord tone, and resolve to the true chord. An escape tone (ET) is a combination of an anticipation and a suspension, straddling the chord tone it is leaving.

A.3 Inversion

This thesis does not take the inversion of chords into account, but it may be interesting to point out why inversion is not important to the realization of a chord. When using a pitch class profile system, the lowest C is notated the same as the highest C, and every C in between. This is because of the noise that occurs in digital audio processing and the sometimes strange voicings in popular music. **Inversion** simply means that the bass note is different from the root—i.e., that the chord is not spelled root-third-fifth(-seventh). When the bass note is the third and there is no seventh, we write " ^{6} " to the upper right of the Roman numeral, e.g., I^6. When the bass note is the fifth and there is no seventh, we write "$^6_4$" to the right of the Roman numeral, e.g., V^6_4. When a seventh is present, a chord inverted with its third in the bass is written with "$^6_5$" to the right; with its fifth in the bass, "$^4_3$" appears to the right; and with its seventh as the lowest pitch, we write "$^4_2$" instead of the uninverted "^{7}" notation. Here are two chords, C Major (I) and C Major^7 (I^7), with their inversions labelled:



The use of a pitch class profile system automatically inverts a chord such that its bass note (not to be confused with its root) is the first appearing in the sequence

{C, C♯, ..., B}. This, in addition to the fact that most listeners cannot distinguish between two types of inversions, renders analysis of inversion obsolete.

A.4 Python Code for Roman Numeral Naming

The following code in the language Python is incomplete, and has thus never been implemented. However, its implementation would clarify the functions of chords within a key, as well as differentiate the functions of chords in a minor key from those in a major key [recall that if a chord has a function, its row vector π is 1 at some i and 0 elsewhere (but the opposite is not necessarily true); otherwise, it is considered functionless].

```
#The Pitch Class Profile system assigns the integers 0 through 11 to the
#12 pitches in 1 octave:

#C=0
#C#=1
#D=2
#D#=3
#E=4
#F=5
#F#=6
#G=7
#G#=8
#A=9
#A#=10
#B=11

#major keys=0 thru 11, minor keys=12 thru 23
for j in range(0,24):
        key.append(j)

#query "What is the key in pitch class value, where 0 through 11 denote the
#major keys C through B major, and 12 through 23 denote the minor keys C
#through B minor?"; j

#The root of a chord is its Roman numeral plus the PCP of the key j,
#mod 12; the third is either 3 or 4 PCP's away from the root; the
#fifth is either 6, 7, or 8; and the seventh is either 9, 10, or 11.
#When there is not a seventh specified, the fourth and fifth entry of
#the distinct row vector defining the Roman numeral name are -1, i.e.,
#not a PCP. When a seventh is specified, it can either be a minor or
#major third away from the fifth (3 or 4 PCP's away), except in the case
#of an augmented seventh chord, which can only have a seventh 3 PCP's
#from its fifth. For augmented seventh chords, the fifth entry of their
#Roman numeral's row vector is -1, and for all other seventh chords,
#all entries are nonnegative.
```

```
I=[(0+key[j])%12, (4+key[j])%12, (7+key[j])%12, -1, -1]

Isev=[(0+key[j])%12, (4+key[j])%12, (7+key[j])%12, (10+key[j])%12,
(11+key[j])%12]

Iaug=[(0+key[j])%12, (4+key[j])%12, (8+key[j])%12, -1, -1]

Iaugsev=[(0+key[j])%12, (4+key[j])%12, (8+key[j])%12, (11+key[j])%12, -1]

littlei=[(0+key[j])%12, (3+key[j])%12, (7+key[j])%12, -1, -1]

littleisev=[(0+key[j])%12, (3+key[j])%12, (7+key[j])%12, (10+key[j])%12,
(11+key[j])%12]

bii=[(1+key[j])%12, (5+key[j])%12, (8+key[j])%12, -1, -1]

biisev=[(1+key[j])%12, (5+key[j])%12, (8+key[j])%12, (11+key[j])%12,
(0+key[j])%12]

littleii=[(2+key[j])%12, (5+key[j])%12, (9+key[j])%12, -1, -1]

littleiisev=[(2+key[j])%12, (5+key[j])%12, (9+key[j])%12, (0+key[j])%12,
(1+key[j])%12]

iidim=[(2+key[j])%12, (5+key[j])%12, (8+key[j])%12, -1,-1]

iidimsev=[(2+key[j])%12, (5+key[j])%12, (8+key[j])%12, (11+key[j])%12,
(0+key[j])%12]

III=[(3+key[j])%12, (7+key[j])%12, (10+key[j])%12, -1, -1]

IIIsev=[(3+key[j])%12, (7+key[j])%12, (10+key[j])%12, (1+key[j])%12,
(2+key[j])%12]

IIIaug=[(3+key[j])%12, (7+key[j])%12, (11+key[j])%12, -1, -1]

IIIaugsev=[(3+key[j])%12, (7+key[j])%12, (11+key[j])%12, (2+key[j])%12, -1]

littleiii=[(4+key[j])%12, (7+key[j])%12, (11+key[j])%12, -1, -1]

littleiiisev=[(4+key[j])%12, (7+key[j])%12, (11+key[j])%12, (2+key[j])%12,
(3+key[j])%12]

IV=[(5+key[j])%12, (9+key[j])%12, (0+key[j])%12, -1, -1]

IVsev=[[(5+key[j])%12, (9+key[j])%12, (0+key[j])%12, (3+key[j])%12,
(4+key[j])%12]

IVaug=[(5+key[j])%12, (9+key[j])%12, (1+key[j])%12, -1, -1]

IVaugsev=[(5+key[j])%12, (9+key[j])%12, (1+key[j])%12, (4+key[j])%12, -1]

littleiv=[(5+key[j])%12, (8+key[j])%12, (0+key[j])%12, -1, -1]

littleivsev=IVsev=[[(5+key[j])%12, (8+key[j])%12, (0+key[j])%12, (3+key[j])
%12, (4+key[j])%12]

V=[(7+key[j])%12, (11+key[j])%12, (2+key[j])%12, -1, -1]
Vsev=[(7+key[j])%12, (11+key[j])%12, (2+key[j])%12, (5+key[j])%12,
(6+key[j])%12)]

Vaug=[(7+key[j])%12, (11+key[j])%12, (3+key[j])%12, -1, -1]

Vaugsev=[(7+key[j])%12, (11+key[j])%12, (3+key[j])%12, (6+key[j])%12, -1]

littlev=[(7+key[j])%12, (10+key[j])%12, (2+key[j])%12, -1, -1]

littlevsev=[(7+key[j])%12, (10+key[j])%12, (2+key[j])%12, (5+key[j])%12,
(6+key[j])%12)]

VI=[(8+key[j])%12, (0+key[j])%12, (3+key[j])%12, -1, -1]

VIsev=[(8+key[j])%12, (0+key[j])%12, (3+key[j])%12, (6+key[j])%12,
(7+key[j])%12]

VIaug=[(8+key[j])%12, (0+key[j])%12, (4+key[j])%12, -1, -1]

VIaugsev=[(8+key[j])%12, (0+key[j])%12, (4+key[j])%12, (7+key[j])%12, -1]

littlevi=[(9+key[j])%12, (0+key[j])%12, (4+key[j])%12, -1, -1]

littlevisev=[(9+key[j])%12, (0+key[j])%12, (4+key[j])%12, (7+key[j])%12,
(8+key[j])%12]

VII=[(10+key[j])%12, (2+key[j])%12, (5+key[j])%12, -1, -1]

VIIsev=[(10+key[j])%12, (2+key[j])%12, (5+key[j])%12, (8+key[j])%12,
(9+key[j])%12]

VIIaug=[(10+key[j])%12, (2+key[j])%12, (6+key[j])%12, -1, -1]

VIIaugsev=[(10+key[j])%12, (2+key[j])%12, (6+key[j])%12, (9+key[j])%12, -1]

viidim=[(11+key[j])%12, (2+key[j])%12, (5+key[j])%12, -1,-1]

viidimsev=[(11+key[j])%12, (2+key[j])%12, (5+key[j])%12, (8+key[j])%12,
(9+key[j])%12]

VofV=[(2+key[j])%12, (6+key[j])%12, (9+key[j])%12, -1,-1]

VofVsev=[(2+key[j])%12, (6+key[j])%12, (9+key[j])%12, (0+key[j])%12,
(1+key[j])%12]

littlevofV=[(2+key[j])%12, (5+key[j])%12, (9+key[j])%12, -1,-1]

littlevofVsev=[(2+key[j])%12, (5+key[j])%12, (9+key[j])%12, (0+key[j])%12,
(1+key[j])%12]

Vofvi=[(4+key[j])%12, (8+key[j])%12, (11+key[j])%12, -1,-1]

Vofvisev=[(4+key[j])%12, (8+key[j])%12, (11+key[j])%12, (2+key[j])%12,
(3+key[j])%12]

Vofiii=[(11+key[j])%12, (3+key[j])%12, (6+key[j])%12, -1, -1]

Vofiiisev=[(11+key[j])%12, (3+key[j])%12, (6+key[j])%12, (9+key[j])%12,
(10+key[j])%12]
```

```
Vofii=[(9+key[j])%12, (1+key[j])%12, (4+key[j])%12, -1, -1]

Vofiisev=[(9+key[j])%12, (1+key[j])%12, (4+key[j])%12, (7+key[j])%12,
(8+key[j])%12]

Vofviidim=[(6+key[j])%12, (10+key[j])%12, (1+key[j])%12, -1, -1]

Vofviidimsev=[(6+key[j])%12, (10+key[j])%12, (1+key[j])%12, (4+key[j])%12,
(5+key[j])%12]

viidimofV=[(6+key[j])%12, (9+key[j])%12, (0+key[j])%12, -1, -1]

viidimofVsev=[(6+key[j])%12, (9+key[j])%12, (0+key[j])%12, (3+key[j])%12,
(4+key[j])%12]

viidimofii=[(1+key[j])%12, (4+key[j])%12, (7+key[j])%12, -1, -1]

viidimofiisev=[(1+key[j])%12, (4+key[j])%12, (7+key[j])%12, (10+key[j])%12,
(11+key[j])%12]

viidimofIV=[(5+key[j])%12, (8+key[j])%12, (11+key[j])%12, -1, -1]

viidimofIVsev=[(5+key[j])%12, (8+key[j])%12, (11+key[j])%12, (2+key[j])%12,
(3+key[j])%12]

viidimofbII=[(0+key[j])%12, (3+key[j])%12, (6+key[j])%12, -1, -1]

viidimofbIIsev=[(0+key[j])%12, (3+key[j])%12, (6+key[j])%12, (9+key[j])%12,
(10+key[j])%12]

viidimofiii=[(3+key[j])%12, (6+key[j])%12, (9+key[j])%12, -1, -1]

viidimofiiisev=[(3+key[j])%12, (6+key[j])%12, (9+key[j])%12, (0+key[j])%12,
(1+key[j])%12]

littlevofiii=[(11+key[j])%12, (2+key[j])%12, (6+key[j])%12, -1, -1]

littlevofiiisev=[(11+key[j])%12, (2+key[j])%12, (6+key[j])%12, (9+key[j])
%12, (10+key[j])%12]

ivofiv=[(10+key[j])%12, (1+key[j])%12, (5+key[j])%12, -1, -1]

ivofivsev=[(10+key[j])%12, (1+key[j])%12, (5+key[j])%12, (8+key[j])%12,
(9+key[j])%12]

viidimofVI=[(7+key[j])%12, (10+key[j])%12, (1+key[j])%12, -1, -1]

viidimofVIsev=[(7+key[j])%12, (10+key[j])%12, (1+key[j])%12, (4+key[j])%12,
(5+key[j])%12]

viidimofVII=[(9+key[j])%12, (0+key[j])%12, (3+key[j])%12, -1, -1]

viidimofVIIsev=[(9+key[j])%12, (0+key[j])%12, (3+key[j])%12, (6+key[j])%12,
(7+key[j])%12]
```

Appendix B

Hidden Markov Models

B.1 Application

We see Hidden Markov Models (HMMs) most often in research on speech recognition, but because music is inherently like language in that every artist is unique, it has useful application to the recognition and algorithmic composition of music. However, I did not quite figure out the reasonable place for HMMs in chord recognition when one does not wish to compose a song from the data, so I relocated this section to an appendix to avoid confusion.

B.2 Important Definitions and Algorithms

Definition: Hidden Markov Chain. A *hidden Markov chain* consists of two sequences of states, assigned to integer values: a predicted Markov chain $Q = (q_1, \ldots, q_T)$ to be modified, and a sequence of observations $O = (o_1, \ldots, o_T)$, where o_t is an observation about state q_t (Landecker). The Markov property holds so that

$$P(q_t = x_t | q_1 = x_1, q_2 = x_2, \ldots, q_{t-1} = x_{t-1}) = P(q_t = x_t | q_{t-1} = x_{t-1}).$$

So in our case with chord progressions deducing something about musical style, we think of q_i as a chord within a given style at time i, and o_i as the actual (observed) chord played at time i. Since we want to predict the behavior of the chords within a style, we construct a model that will tell us how closely O adheres to Q. This model, a hidden Markov model, is defined by the following.

Definition: Hidden Markov Model. A *hidden Markov model (HMM)* $\lambda = (A, B, \pi)$ is a triple containing three different sets of probabilities: (1) the state transition probabilities a_{ij}, (2) the probabilities of observing each o_t given q_t (defining the observation matrix B), and (3) the probabilities of beginning the chain in each of the states q_i (the initial distributions π_i).

We can think of λ as a set of parameters to be modified again and again until $P(O|\lambda)$ is maximized.

There are three main obstacles one must overcome to build a satisfactory HMM:

1. Given O and λ, calculate $P(O|\lambda)$.
2. Given O and λ, how can we "refresh" Q so that $P(Q|O,\lambda)$ is maximized?
3. Given O, alter λ to maximize $P(O|\lambda)$.

We surmount the first of these problems with the *forward algorithm*, the second with the *Viterbi algorithm* or *backward algorithm*, and the third can be solved with any of the *backward-forward algorithm* (sometimes called the *forward-backward algorithm*), the *Baum-Welch algorithm*, or the *Expectation Maximization (EM) algorithm*. The most common and best for speech recognition is the Baum-Welch, so I believe since music is a language, what with all of its "speakers" communicating distinctly (thereby making the possibilities for the set S infinite, but the established vocabulary Q finite), it will also be best suited for music. These algorithms will be explicated once we lay out more formal definitions for the many variables designated above.

Now, let there be N distinct possible states, i.e., chords, and define the **state space** as $S = \{s_i\}, 1 \le i \le N$. Let there be M distinct possible observations, and define the **observation space** as $V = \{v_i\}, 1 \le i \le M$. Then, each state q_i comes from S, and each observation o_i comes from V.

We represent the random variable of the **observation** at time t by O_t, and likewise the random variable of the **state** at time t by Q_t, such that there are T-many random variables for each of O and Q. By the Markov property, the probability of being in a state depends only on the previous state, and the current observation depends only upon the current state, so therefore,

$$P(Q_t = q_t|Q_1 = q_1, \dots, Q_T = q_T, O_1 = o_1, \dots, O_T = o_T, \lambda) = P(Q_t = q_t|Q_{t-1} = q_{t-1}, \lambda)$$

and

$$P(O_t = o_t|Q_1 = q_1, \dots, Q_T = q_T, O_1 = o_1, \dots, O_T = o_T, \lambda) = P(O_t = o_t|Q_t = q_t, \lambda)$$

Without loss of generality, the above (beastly) equalities are notationally equivalent (in this thesis, unless otherwise noted) to writing

$$P(q_t|q_1, \dots, q_T, o_1, \dots, o_T, \lambda) = P(q_t|q_{t-1}, \lambda),$$

and

$$P(q_t|q_1, \dots, q_T, o_1, \dots, o_T, \lambda) = P(o_t|q_t, \lambda).$$

We construct our **state transition matrix** A, relating to Q's behavior over time, with the probabilities

$$A = \{a_{ij} : a_{ij} = P(q_t = s_j|q_{t-1} = s_i, \lambda), 1 < t \le T\},$$

$$\sum_{j=1}^{N} a_{ij} = 1.$$

At initialization of our hidden Markov chain Q, or when $t = 1$, we define the **initial distribution** π by the row matrix

$$\pi = \{\pi_i : \pi_i = P(q_1 = s_i|\lambda)\},$$

$$\sum_{i=1}^{N} \pi_i = 1.$$

Then, our **observation matrix** B is given by

$$B = \{b_{ij} : b_{ij} = P(o_t = v_j|q_t = s_i, \lambda)\},$$

$$\sum_{j=1}^{M} b_{ij} = 1,$$

and contains the probabilities

$$\begin{aligned} b_i(j) &= b_{ij} = P(o_t = v_j|q_t = s_i, \lambda), \\ b_{q_t}(o_t) &= b_{q_t o_t} = P(O_t = o_t|Q_t = q_t, \lambda). \end{aligned}$$

We assume that A and B are **time-homogenous**, that is, the probability of transitioning between any two states is the same at all times. This is also a byproduct of the Markov property.

Now we can attempt to tackle the aforementioned three main problems that, once solved, will only better our HMM. First, we want to know how well our current model λ predicts O, i.e., $P(O|\lambda)$. Naively, this is

$$P(O|\lambda) = \sum_{i=1}^{T} P(O|q_i, \lambda)P(q_i|\lambda)$$

by Bayes' formula. Now,

$$\begin{aligned} P(O|Q, \lambda) &= \prod_{i=1}^{T} P(o_i|q_i, \lambda) \\ &= b_{q_1}(o_1) \cdot b_{q_2}(o_2) \cdots b_{q_T}(o_T), \end{aligned}$$

and

$$P(Q|\lambda) = \pi_{q_1} \cdot a_{q_1 q_2} \cdot a_{q_2 q_3} \cdots a_{q_{(T-1)} q_T},$$

so the probability $P(O|\lambda)$ becomes

$$\sum_{i=1}^{T} P(O|q_i, \lambda)P(q_i|\lambda) = \sum_{i=1}^{T} \pi_{q_i} b_{q_1}(o_1) a_{q_1 q_2} \cdots b_{q_T}(o_T) a_{q_{(T-1)} q_T}.$$

However, we can also discover this probability recursively using the **forward algorithm**.

Definition: Forward Algorithm. The **forward variable** $\alpha_j(t)$ assists computation of the probability $P(O|\lambda)$ with the function

$$\alpha_j(t) = P(o_1, o_2, \ldots, o_t, q_t = s_j|\lambda),$$

the probability of observing O up to time t and being in state s_j at that time. We calculate $\alpha_j(t)$ by multiplying $P(O_t = o_t|q_t = s_j)$ by each probability of transitioning to state s_j from the instant before, time $t-1$. This heeds the recursion

$$\alpha_j(t) = \sum_{i=1}^{N} \alpha_i(t-1)a_{ij}b_j(o_t)$$

initialized by $\alpha_j(1) = \pi_j b_j(o_1)$. Hence,

$$P(O|\lambda) = \sum_{j=1}^{N} \alpha_j(T).$$

We call this recursion the **forward algorithm**.

This is found in TN^2 operations versus the $2TN^T$ operations made in the naïve calculation of $P(O|\lambda)$ above. It is called the "forward" algorithm because we first compute $\alpha_j(2)$, then $\alpha_j(3)$, up to $\alpha_j(t)$, whereas the "backward" algorithm we are about to discuss is first computed at time $T-1$, down to time $t+1$.

Second, we want to calculate the probability of receiving our remaining observations, $o_{t+1}, \ldots, o_T$, given the current state q_t. This is the same as finding $P(o_{t+1}, \ldots, o_T|q_t = s_i, \lambda)$ by which we define our "backward variable".

Definition: Backward Algorithm. The **backward variable** $\beta_i(t)$ is given be

$$\beta_i(t) = P(o_{t+1}, \ldots, o_T|q_t = s_i, \lambda),$$

and is the probability of a "finishing observation sequence" starting at time t, given that we are currently in the state $s_i = q_t$. It analyzes the set of observations mutually exclusive from those analyzed with the forward variable, and thus is calculated by multiplying the probability of the finishing observation sequence with the possible transition probabilities between times t and $t+1$. This is the recursion

$$\beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1)a_{ij}b_j(o_{t+1})$$

initialized by $\beta_i(T) = 1$. Hence, another way of getting $P(O|\lambda)$ is found:

$$P(O|\lambda) = \sum_{i=1}^{N} \beta_i(1)\pi_i b_i(o_1).$$

Now that we have our forward and backward algorithm, we want to redefine λ so that $P(O|\lambda)$ is maximized. This will solve our third problem.

First, note that

$$\alpha_i(t)\beta_i(t) = P(o_1, \ldots, o_t, q_t = s_i|\lambda)P(o_{t+1}, \ldots, o_T, q_t = s_i|\lambda)$$

which by the definition of conditional probability is

$$= \left(\frac{P(o_1, \ldots, o_t, q_t = s_i, \lambda)}{P(\lambda)}\right)\left(\frac{P(o_{t+1}, \ldots, o_T, q_t = s_i, \lambda)}{P(q_t = s_i, \lambda)}\right).$$

Since the Markov property states that the probability of the t-th observation o_t depends only on the t-th state q_t and λ, i.e., $q_t, o_{t+1}, \ldots, o_T$ are independent of one another,

$$\begin{aligned}\alpha_i(t)\beta_i(t) &= \frac{P(o_1, \ldots o_t, q_t = s_i, \lambda)P(o_{t+1}, \ldots, o_T)P(q_t = s_i, \lambda)}{P(\lambda)P(q_t = s_i, \lambda)} \\ &= \frac{P(o_1, \ldots o_t, q_t = s_i, \lambda)P(o_{t+1}, \ldots, o_T)}{P(\lambda)}.\end{aligned}$$

Again calling upon the Markov property, we see that

$$P(o_1, \ldots, o_t, q_t = s_i, \lambda)P(o_{t+1}, \ldots, o_T) = P(o_1, \ldots, o_T, q_t = s_i, \lambda),$$

so the product of the forward and backward variables is then

$$\begin{aligned}\alpha_i(t)\beta_i(t) &= \frac{P(o_1, \ldots, o_T, q_t = s_i, \lambda)}{P(\lambda)} \\ &= P(O, q_t = s_i|\lambda).\end{aligned}$$

Thus, we can calculate the probability we are trying to maximize a third way, by

$$\begin{aligned}P(O|\lambda) &= \sum_{i=1}^{N} P(O, q_t = s_i|\lambda) \\ &= \sum_{i=1}^{N} \alpha_i(t)\beta_i(t)\end{aligned}$$

for any t.

Now we want to define a variable that will give us the expected number of times of entering state s_i given O and λ. This is found with the conditional probability

$$\begin{aligned}P(q_t = s_i|O, \lambda) &= \frac{P(O, q_t = s_i|\lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)} \\ &= \gamma_i(t).\end{aligned}$$

We call this conditional probability $\gamma_i(t)$ and see that the value $\sum_{t=1}^{T} \gamma_i(t)$ gives us the expected occurrence of state s_i, as desired above. This will help refine all of the variables in λ.

We are almost finished describing the initial structure of an HMM. Next, we need to calculate the probability of transitioning from state s_i to state s_j at any time t. We call this, keeping in line with Landecker's notation, $\zeta_{ij}(t)$, and it is defined by

$$\begin{aligned}\zeta_{ij}(t) &= P(q_t = s_i, q_{t+1} = s_j | O, \lambda)\\ &= \frac{P(q_t = s_i, q_{t+1} = s_j, O | \lambda)}{P(O|\lambda)}\end{aligned}$$

which by Bayes' formula is

$$= \frac{(\alpha_i(t) a_{ij})(\beta_j(t+1) b_j(o_{t+1}))}{\sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_k(t) a_{kl} \beta_l(t+1) b_l(o_{t+1})}.$$

Thus, the value $\sum_{t=1}^{T-1} \zeta_{ij}(t)$ is the expected number of transitions from state s_i to state $s_j \in O$.

At last we can refine our model's parameters, π, A, and B as follows:

$$\begin{aligned}\hat{\pi}_i &= \gamma_i(1),\\ \hat{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \zeta_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)},\\ \hat{b}_{ij} &= \frac{\sum_{t=1}^{T} \gamma_i(t) \delta_{o_t, v_j}}{\sum_{t=1}^{T} \gamma_i(t)},\end{aligned}$$

where δ_{o_t, v_j} is equal to 1 when $o_t = v_j$ and is 0 otherwise.

B.3 Computational Example of an HMM

Let's say that we want to model (what we believe to be) a blues song, and that there are only 3 possible states, I, IV, and V^7, to which it can transition. We are pretty sure that blues songs typically begin with either I or V^7, and it is more likely to hear I initially. Then, any of the three states can transition to any of the other three states. We deduce from the twelve-bar blues progression, $Q = \{I, I, I, I, IV, IV, I, I, V7, IV, I, I\}$ that our initial transition matrix is

$$A = \begin{bmatrix} 5/7 & 1/7 & 1/7 \\ 2/3 & 1/3 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

where $s_1 = I, s_2 = IV$, and $s_3 = V^7$.

A only depends on Q, and Q comes from some pre-existing notion of a typical blues progression. We cannot build B just yet since it depends on O which we have not yet observed, but we can define our initial probabilities π_i as follows:

$$\pi_i(1) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

Now, we observe a repeating sequence of 8 chords in a song we think should be classified as "blues". We observe $O = (I, I,\ I, I,\ V^7, IV,\ I, I,\ I, IV,\ V^7, I)$, while our "typical" blues progression Q is defined as $Q = (I, I,\ I, I,\ IV, IV,\ I, I,\ V^7, IV,\ I, I)$. Thus, Q does not equal O at times 5, 9, 10, or 11, and $N = 3, T = 12$. So, in Mathematica,

```
(* State observation sequence, state sequence, set of observation states,
and set of possible states *)
OO = {1, 1, 1, 1, 3, 2, 1, 1, 1, 2, 3, 1};
QQ = {1, 1, 1, 1, 2, 2, 1, 1, 3, 2, 1, 1};
VV = {1, 2, 3};
SS = {1, 2, 3};
(* Initialize initial probability distribution p(v_i) *)
p = {1, 0, 0};
(* Initialize transition matrix A *)
A = {{5 / 7, 1 / 7, 1 / 7},
   {2 / 3, 1 / 3, 0},
   {0, 1, 0}};
(* Initialize observation matrix B *)
B = {{5 / 7, 1 / 7, 1 / 7},
   {.5, 0, .5},
   {.5, .5, 0}};
```

and hence we can define $\alpha_j(t)$ as a matrix, $[\alpha_{j,t}]$, by

$$\begin{aligned} \alpha_{j,1} &= \pi_j B_{j,1}, && 1 \le j \le 3; \\ \alpha_{j,t} &= \sum_{k=1}^{3} \alpha_{k,t-1} A_{k,j} B_{j,o_t}, && 1 \le j \le 3,\ 2 \le t \le 12. \end{aligned}$$

Then, our backward variable $\beta_i(t)$ is

$$\begin{aligned} \beta_{i,12} &= 1, && 1 \le i \le 3; \\ \beta_{i,t} &= \sum_{k=1}^{3} \beta_{k,t+1} A_{i,k} B_{k,o_{t+1}}, && 1 \le i \le 3,\ 1 \le j \le 11. \end{aligned}$$

Doing these two algorithms by hand required 10 pages (and included many a computational error), so I defined the following loops in Mathematica:

```
(* Initialize forward variable alpha_j[1]=alpha[[j,1]] *)
alpha = {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};
Do[
  alpha[[j, 1]] = p[[j]] * B[[j, OO[[1]]]], {j, 1, 3}];
(* Compute forward algorithm alpha_j[t]=alpha[[j,t]] *)
Do[
  alpha[[j, t]] = Sum[alpha[[i, t - 1]] * A[[i, j]] * B[[j, OO[[t]]]], {i, 1, 3}],
  {j, 1, 3}, {t, 2, 12}];
```

The matrix form of α is

$$\alpha_{N\times T} = \begin{bmatrix} .7143 & .3644 & .1859 & .09484 & .009677 & .0009874 & .0005034 & .000257 & .0001311 & 1.338E\text{-}5 & 1.365E\text{-}6 & 6.964E\text{-}7 \\ 0 & .0510 & .03453 & .01903 & .009946 & 0 & 7.053E\text{-}5 & 4.774E\text{-}5 & 2.631E\text{-}5 & 0 & 9.556E\text{-}7 & 2.568E\text{-}7 \\ 0 & .05102 & .02603 & .01328 & 0 & .0006912 & 7.053E\text{-}5 & 3.598E\text{-}5 & 1.836E\text{-}5 & 9.365E\text{-}6 & 0 & 9.750E\text{-}8 \end{bmatrix},$$

and we find β by the recursion

```
(* Initialize backward variable beta_j(T)=beta[[j,T]] *)
beta = {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};
Do[
  beta[[i, 12]] = 1, {i, 1, 3}];

(* Define the backward algorithm beta_i[t]=beta[[i,t]],
counting backwards from t=T-1 to t=1 *)
Do[
  beta[[i, 12 - t]] = Sum[beta[[j, 12 - t + 1]] * A[[i, j]] * B[[j, OO[[12 - t + 1]]]],
    {j, 1, 3}], {i, 1, 3}, {t, 1, 11}];
```

The matrix form of β is

$$\beta_{N\times T} = \begin{bmatrix} 1.248E\text{-}6 & 2.446E\text{-}6 & 4.8E\text{-}6 & 9.399E\text{-}6 & 9.211E\text{-}5 & .0009028 & .00177 & .003469 & .0068 & .06663 & .653 & 1 \\ 1.777E\text{-}6 & 3.671E\text{-}6 & 8.326E\text{-}6 & .0000231 & 8.598E\text{-}5 & .001237 & .002367 & .004295 & .006346 & .1693 & .6428 & 1 \\ 1.835E\text{-}6 & 4.163E\text{-}6 & .00001155 & 4.299E\text{-}5 & 0 & .001184 & .002148 & .003173 & 0 & .3214 & .5 & 1 \end{bmatrix}.$$

Then I defined $\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{k=1}^{t}\alpha_k(t)\beta_k(t)}$ in Mathematica:

```
(* Define forward-backward variable,
gamma_i[t]=gamma[[i,t]] from alpha and beta *)
gamma = {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};
Do[
 If[
  Sum[alpha[[k, t]] * beta[[k, t]], {k, 1, 3}] > 0,
  gamma[[i, t]] = (alpha[[i, t]] * beta[[i, t]]) /
    Sum[alpha[[k, t]] * beta[[k, t]], {k, 1, 3}],
  gamma[[i, t]] = 0], {i, 1, 3}, {t, 1, 12}]
```

Then, the matrix form of γ is

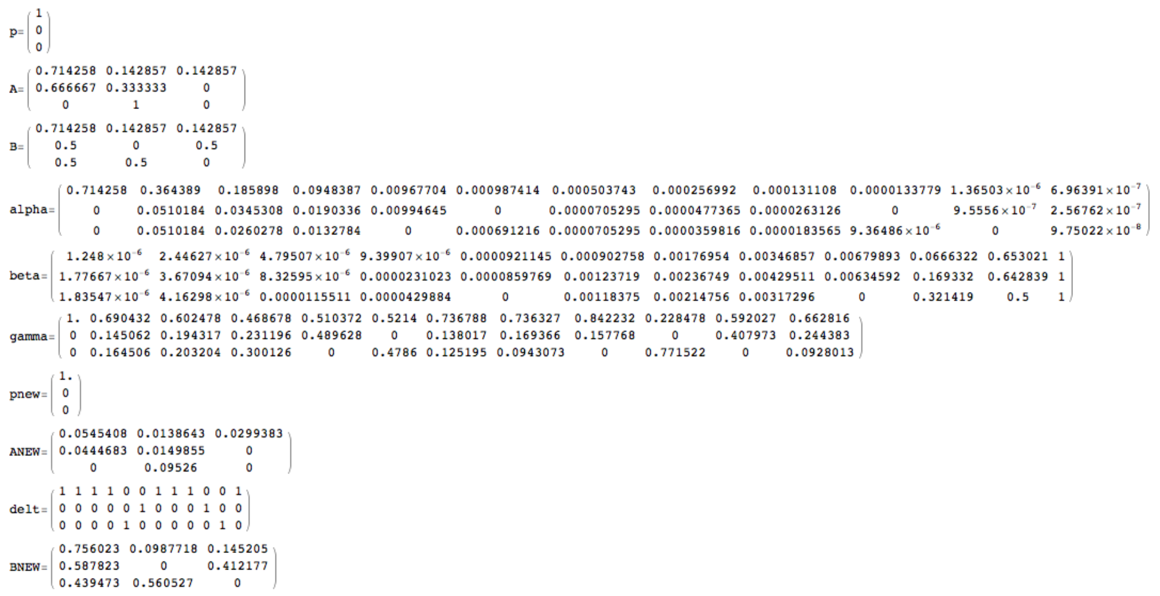
$$\gamma_{N\times T} = \begin{bmatrix} 1 & .6904 & .6025 & .4687 & .5104 & .5214 & .7368 & .7363 & .8422 & .2285 & .5920 & .6628 \\ 0 & .1451 & .1943 & .2312 & .4896 & 0 & .1380 & .1694 & .1578 & 0 & .408 & .2444 \\ 0 & .1645 & .2032 & .3001 & 0 & .4786 & .1252 & .0943 & 0 & .7715 & 0 & .0928 \end{bmatrix}$$

We can now re-estimate our π, α, and β. I called the re-estimates $\hat{\pi}$ "**PNEW**", $\hat{A}$ "**ANEW**", and $\hat{B}$ "**BNEW**" in Mathematica, as well as "**delt**" (since "delta" is reserved in Mathematica) for the Kronecker delta, δ_{o_t,v_k}. Since $\hat{A}$ is quite a lengthy summand, I broke up the numerator and denominator and called them "**numA**" and "**denumA**". The recursion is as follows:

```
(* Reestimate initial probability distribution p_i *)
pnew = {gamma[[1, 1]], gamma[[2, 1]], gamma[[3, 1]]};
```

```
(* Reestimate transition matrix A *)
numA = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};
denumA = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};
ANEW = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};
Do[
 Do[
  Do[
   numA[[i, j]] = alpha[[i, t]] * A[[i, j]] * beta[[j, t + 1]] * B[[j, OO[[t + 1]]]];
   denumA[[i, j]] =
    Sum[Sum[alpha[[k, t]] * A[[k, l]] * beta[[l, t + 1]] * B[[l, OO[[t + 1]]]], {l, 1, 3}],
     {k, 1, 3}],
   {j, 1, 3}],
  {i, 1, 3}],
 {t, 1, 11}]

Do[
 Do[
  If[Sum[gamma[[i, t]], {t, 1, 11}] > 0 && denumA[[i, j]] > 0,
   ANEW[[i, j]] =
    Sum[
      (alpha[[i, t]] * A[[i, j]] * beta[[j, t + 1]] * B[[j, OO[[t + 1]]]]) /
       (Sum[Sum[alpha[[k, t]] * A[[k, l]] * beta[[l, t + 1]] * B[[l, OO[[t + 1]]]],
          {l, 1, 3}], {k, 1, 3}]),
      {t, 1, 11}]
     / (Sum[gamma[[i, t]], {t, 1, 11}]), ANEW[[i, j]] = 0],
  {i, 1, 3}],
 {j, 1, 3}]

(* Reestimate observation matrix B *)
BNEW = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}};
delt  = {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};

(* delt is the Kronecker delta matrix where delt[v_j,t]=1 if v_j=o_t,
and 0 otherwise *)
Do[
  Do[
   If[OO[[t]] == VV[[j]], delt[[VV[[j]], t]] = 1, delt[[VV[[j]], t]] = 0],
   {t, 1, 12}],
  {j, 1, 3}];
Do[
  If[Sum[gamma[[i, t]], {t, 1, 12}] > 0,
   BNEW[[i, j]] = Sum[gamma[[i, t]] * delt[[VV[[j]], t]], {t, 1, 12}] /
     Sum[gamma[[i, t]], {t, 1, 12}], BNEW[[i, j]] = 0],
  {i, 1, 3}, {j, 1, 3}];
```

In summary,



```
p=    1
      0
      0

A=    0.714258 0.142857 0.142857
      0.666667 0.333333    0
         0        1        0

B=    0.714258 0.142857 0.142857
        0.5       0       0.5
        0.5      0.5       0

alpha=  0.714258  0.364389  0.185898  0.0948387 0.00967704 0.000987414 0.000503743  0.000256992  0.000131108  0.0000133779 1.36503×10^-6 6.96391×10^-7
           0      0.0510184 0.0345308 0.0190336 0.00994645      0      0.0000705295 0.0000477365 0.0000263126       0        9.5556×10^-7  2.56762×10^-7
           0      0.0510184 0.0260278 0.0132784     0      0.000691216 0.0000705295 0.0000359816 0.0000183565 9.36486×10^-6       0        9.75022×10^-8

beta=   1.248×10^-6   2.44627×10^-6 4.79507×10^-6 9.39907×10^-6 0.0000921145 0.000902758 0.00176954 0.00346857 0.00679893 0.0666322 0.653021 1
       1.77667×10^-6  3.67094×10^-6 8.32595×10^-6 0.0000231023 0.0000859769  0.00123719 0.00236749 0.00429511 0.00634592 0.169332  0.642839 1
       1.83547×10^-6  4.16298×10^-6 0.0000115511  0.0000429884       0        0.00118375 0.00214756 0.00317296     0      0.321419    0.5    1

gamma=  1. 0.690432 0.602478 0.468678 0.510372 0.5214 0.736788 0.736327  0.842232 0.228478 0.592027 0.662816
        0  0.145062 0.194317 0.231196 0.489628   0    0.138017 0.169366  0.157768    0     0.407973 0.244383
        0  0.164506 0.203204 0.300126    0     0.4786 0.125195 0.0943073    0     0.771522    0     0.0928013

pnew=   1.
        0
        0

ANEW=   0.0545408 0.0138643 0.0299383
        0.0444683 0.0149855     0
            0      0.09526      0

delt=   1 1 1 1 0 0 1 1 1 0 0 1
        0 0 0 0 0 1 0 0 0 1 0 0
        0 0 0 0 1 0 0 0 0 0 1 0

BNEW=   0.756023 0.0987718 0.145205
        0.587823     0     0.412177
        0.439473 0.560527     0
```

So $\hat{\pi}, \hat{A}, \hat{B}$, and δ_{o_t,v_k} are, in matrix form,

$$
\begin{aligned}
\hat{\pi}_{N\times 1} &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \\
\hat{A}_{N\times N} &= \begin{bmatrix} 0.701542 & 0.159441 & 0.266733 \\ 0.553007 & 0.198254 & 0 \\ 0 & 0.810903 & 0 \end{bmatrix} \\
\delta_{N\times T} &= \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \\
\hat{B}_{N\times N} &= \begin{bmatrix} 0.75602 & 0.0987759 & 0.145204 \\ 0.587813 & 0 & 0.412187 \\ 0.439457 & 0.560543 & 0 \end{bmatrix}.
\end{aligned}
$$

The rows of A do not add up to 1, but the rows of B and the columns of γ, π, and δ do. This is because

$$
\begin{aligned}
P(O|\lambda) &= \sum_{j=1}^{N} \alpha_j(T) = \alpha_1(12) + \alpha_2(12) + \alpha_3(12) \\
&= 0.0000010514 = \sum_{i=1}^{N} \alpha_i(12)\beta_i(12),
\end{aligned}
$$

but

$$\begin{aligned} P(O|\lambda) &= \sum_{i=1}^{N} \beta_i(1)\pi_i b_i(o_1) = \beta_1(1)\cdot 1 \cdot 5/7 + 0 + 0 \\ &= 0.0000008929042 = \sum_{i=1}^{N} \alpha_i(1)\beta_i(1). \end{aligned}$$

The failure of this very simple example perfectly encapsulates the difficulties associated with HMMs, namely with keeping all of the variables straight. A good resource for those unfamiliar with HMMs is [45].

Bibliography

Entropy, Information Theory, and Probability Theory

[1] Blachman, Nelson M. *Noise and Its Effect on Communication*. New York: McGraw-Hill, 1966.

[2] Brattain-Morrin, Eric. *Entropy, Computation, and Demons*. Portland, OR: 2008.

[3] Khinchin, A. I. "On the Fundamental Theorems of Information Theory ". *Mathematical Foundations of Information Theory*. New York: Dover Publications, Inc., 1957, pp. 30-120.

[4] Khinchin, A. I. "The Entropy Concept in Probability Theory. "*Mathematical Foundations of Information Theory*. New York: Dover Publications, Inc., 1957, pp. 2-29.

[5] Ross, Sheldon. *A First Course in Probability, Seventh Edition*. Upper Saddle River, NJ: Pearson Education, Inc., 2006, pp. 24-26, 66-87, 132, 138-141, 205-206, 209, 214, 466-483.

[6] Shannon, Claude E. and Warren Weaver. *The Mathematical Theory of Communication*. Urbana, IL: University of Illinois Press, 1998.

Digital Signal Processing

[7] Fleet, D.J. and A.D. Jepson. "Linear Filters, Sampling, and Fourier Analysis." http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/, last modified 22 September 2005.

[8] Johnson, Johnny R. *Introduction to Digital Signal Processing*. New Delhi, India: Prentice-Hall of India, 1998.

[9] Lane, John and Garth Hillman. *Motorola Digital Signal Processing: Implementing IIR/FIR Filters with Motorola's DSP56000/DSP56001*. Motorola Inc., 1993.

[10] Marven, Craig and Gillian Ewers. *A Simple Approach to Digital Signal Processing*. New York: John Wiley and Sons, Inc., 1996, pp. 31-189.

[11] Ed. Rabiner, Lawrence R. and Charles M. Rader. *Digital Signal Processing*. New York: The Institute of Electrical and Electronics Engineers, Inc., 1972.

[12] Rabiner, Lawrence and Bing-Hwang Juang. *Fundamentals of Speech Recognition.* Englewood Cliffs, NJ: PTR Prentice-Hall, Inc.

[13] Roads, Curtis. *The Computer Music Tutorial.* Cambridge, MA: The MIT Press, 1996, pp. 189, 193-196, 506-520, 934-935..

[14] Rorabaugh, C. Britton. *DSP Primer.* New York: McGraw-Hill, 1999.

Music

[15] Ames, C. and M. Domino. "Cybernetic Composer: An Overview". Balaban, Mira, Kemal Ebcioglu, and Otto Laske Ed.'s. *Understanding Music with AI: Perspectives on Music Cognition.* Cambridge, MA: The AAAI Press/The MIT Press, 1992, pp. 186-205.

[16] "Beatles Unknown 'A Hard Day's Night' Chord Mystery Solved Using Fourier Transform." Updated October 2008. Accessed April 2009. `http://www.scientificblogging.com/news_releases/beatles_unknown_hard_days_night_chord_mystery_solved_using_fourier_transform?`

[17] Cope, D. "On the Algorithmic Representation of Musical Style." Balaban, Mira, Kemal Ebcioglu, and Otto Laske Ed.'s. *Understanding Music with AI: Perspectives on Music Cognition.* Cambridge, MA: The AAAI Press/The MIT Press, 1992, pp. 354-363.

[18] Courtot, F. "Logical Representation and Induction for Computer Assisted Composition." Ed. Balaban, Mira, Kemal Ebcioglu, and Otto Laske. *Understanding Music with AI: Perspectives on Music Cognition.* Cambridge, MA: The AAAI Press/The MIT Press, 1992, pp. 156-181.

[19] Duckworth, William. *20/20: 20 New Sounds of the 20th Century.* New York: Schirmer Books, 1999.

[20] Ebcioglu, K. "An Expert System for Harmonizing Chorales in the Style of J.S. Bach." Balaban, Mira, Kemal Ebcioglu, and Otto Laske Ed.'s. *Understanding Music with AI: Perspectives on Music Cognition.* Cambridge, MA: The AAAI Press/The MIT Press, 1992, pp. 294-334.

[21] Gelbart, Matthew. *The Invention of "Folk Music" and "Art Music": Emerging Categories from Ossian to Wagner.* Cambridge, UK: The Cambridge University Press, 2007, pp. 1-6.

[22] Isacoff, Stuart. *Temperament.* New York: Alfred A. Knopf, 2001.

[23] Kugel, P. "Beyond Computational Musicology." Balaban, Mira, Kemal Ebcioglu, and Otto Laske Ed.'s. *Understanding Music with AI: Perspectives on Music Cognition.* Cambridge, MA: The AAAI Press/The MIT Press, 1992, pp. 30-48.

[24] Laske, O. "Artificial Intelligence and Music: A Cornerstone of Cognitive Musicology." Balaban, Mira, Kemal Ebcioglu, and Otto Laske Ed.'s. *Understanding Music with AI: Perspectives on Music Cognition.* Cambridge, MA: The AAAI Press/The MIT Press, 1992, pp. 3-29.

[25] Lee, Kyogu. "Automatic Chord Recognition from Audio Using Enhanced Pitch Class Profile." Stanford, CA: Center for Computer Research in Music and Acoustics, 2006, 8 pages.

[26] Maxwell, H.J. "An Expert System for Harmonic Analysis of Tonal Music." Balaban, Mira, Kemal Ebcioglu, and Otto Laske Ed.'s. *Understanding Music with AI: Perspectives on Music Cognition.* Cambridge, MA: The AAAI Press/The MIT Press, 1992, pp. 335-353.

[27] Rooksby, Rikky. *The Beatles Complete Chord Songbook.* Milwaukee, WI: H. Leonard Corp., 1999.

[28] Rowe, Robert. *Interactive Music Systems: Machine Listening and Composing.* Cambridge, MA: The MIT Press, 1993.

[29] Schoenberg, Arnold. *Structural Functions of Harmony.* London: Williams and Norgate Limited, 1954.

[30] Stephenson, Ken. *What to Listen for in Rock: A Stylistic Analysis.* New Haven, CT: Yale University Press, 2002.

[31] Sullivan, Anna T. *The Seventh Dragon: The Riddle of Equal Temperament.* Lake Oswego, OR: Metamorphous Press, 1985.

Fourier Acoustics

[32] Cormen, Thomas, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition.* Cambridge, MA: The MIT Press, 2002.

[33] Smith, Julius O. *Mathematics of the Discrete Fourier Transform (DFT): With Music and Audio Applications.* Stanford, CA: Center for Computer Research in Music and Acoustics, 2003.

[34] Zonst, Anders E. *Understanding the FFT: A Tutorial on the Algorithm and Software for Laymen, Students, Technicians and Working Engineers.* Titusville, FL: Citrus Press, 1995.

Hidden Markov Models

[35] Blunsom, Phil. "Hidden Markov Models." Updated August 2004. Accessed April 2009. `www.cs.mu.oz.au/460/2004/materials/hmm-tutorial.pdf`

[36] Chai, Wei and Barry Vercoe. "Folk Music Classification Using Hidden Markov Models." *Proceedings of International Conference on Artificial Intelligence, 2001.*

[37] Elliot, Robert J., Lakhdar Aggoun, and John B. Moore. *Hidden Markov Models: Estimation and Control*. New York: Springer-Verlag, 1995.

[38] Fraser, Andrew M. *Hidden Markov Models and Dynamical Systems*. Philadelphia: Society for Industrial and Applied Mathematics, 2008.

[39] Haggstrom, Olle. *Finite Markov Chains and Algorithmic Applications*. Cambridge, MA: Cambridge University Press, 2002.

[40] Landecker, Will. *Convergence of the Baum Welch Algorithm and Applications to Music*. Portland, OR: 2007.

[41] Lee, Kyogu and Malcolm Slaney. "A Unified System for Chord Transcription and Key Extraction Using Hidden Markov Models," in *Proceedings of International Conference on Music Information Retrieval*, 2007.

[42] Raphael, Christopher and Josh Stoddard. "Harmonic Analysis with Probabilistic Graphical Models". *International Symposium on Information Retrieval, 2003*. Ed. Holger Hoos and David Bainbridge. Baltimore: 2003, pp. 177-181.

[43] Sheh, Alexander and Daniel P.W. Ellis. "Chord Segmentation and Recognition Using EM-Trained Hidden Markov Models" in *Proceedings of the First ACM Workshop on Audio and Music Computing Multimedia, 2006*, pp. 11-20.

[44] Takeda, Haruto, Naoki Saito, Tomoshi Otsuki, Mitsuru Nakai, Hiroshi Shimodaira, Shigeki Sagayama. "Hidden Markov Model for Automatic Transcription of MIDI Signals" in *IEEE Workshop on Multimedia Signal Processing*, Dec. 9-11, 2002, pp. 428-431.

[45] "What is a Hidden Markov Model?" Updated May 2008. Accessed April 2009. `http://intoverflow.wordpress.com/2008/05/27/what-is-a-hidden-markov-model/`

Glossary

accelerated intelligence Raymond Kurzweil states that "accelerated intelligence" refers to "the quickening pace of our knowledge and intelligence will ultimately alter the nature of what it means to be human."

aliasing when an analog signal is undersampled to a digital form, i.e., when the sampling frequency $f_s < 2f_U$, where f_U is the highest frequency (bandwidth) of the signal, the digital form undergoes aliasing, emitting false amplitudes for the signal at frequencies $n \cdot f_s, n \in \mathbb{Z}$

bandwidth the maximum frequency of a signal, denoted f_U or W (Shannon)

conditional probability the probability of something happening given information about previous happenings

cyclic convolution a binary operation with no multiplicative inverse denoted by $*$; the cyclic convolution of $\{a_0, \ldots, a_{N-1}\}$ and $\{b_0, \ldots, b_{N-1}\}$ is the sequence $\{c_0, \ldots, c_{N-1}\}$ where

$$\sum_{j=0}^{N-1} c_0 x^j \equiv \left(\sum_{j=0}^{N-1} a_j x^j\right)\left(\sum_{j=0}^{N-1} b_j x^j\right)\left(\mathrm{mod} x^N - 1\right).$$

dominant the frequency 7 half steps above the key to which it has a dominant relationship

entropy a measure of the propensity of a signal or system of signals to be incorrectly transmitted as intended

Euler's identity $e^i\pi = -1$, where e is Euler's number, the base of the natural logarithm; i is the imaginary unit equal to $\sqrt{-1}$; and π is the ratio of the circumference of a circle to its diameter. Also, $e^{ix} = \cos(x) + i\sin(x)$ and $e^{-ix} = \cos(x) - i\sin(x)$ for any value (real or imaginary) x

event in probability theory, an event is a set of outcomes

expectation in probability theory, the expectation, or expected value, of a random variable is the predicted value that, on average, the variable will take on

fifth the frequency either 6, 7, or 8 half steps above the fundamental frequency of a chord. This can also refer to an interval separating two notes by 6, 7, or 8 half steps, depending if the quality of the interval is (respectively) diminished, perfect, or augmented

filter in digital and analog signal processing, filters are used to reduce a signal to some desired set of frequencies

impulse in physics, impulse is the integral of force with respect to time, equal to the change in momentum over time

impulse function $\delta(t)$

impulse response a function $h(t)$ mapping the power of a signal over time

independence in probability theory, two schemes or events or outcomes are independent if the occurrence of one does not effect the behavior of the other

Markov chain a Markov chain, or Markov process, is a stochastic process that possesses the Markov property, and its behavior is modeled by a transition graph and transition matrix

Markov property in probability theory, a random variable is said to possess the Markov property, or memoryless property, if every state $X_t = s_j$ only depends on the previous state $X_t - 1 = s_i$

outcome in probability theory, an outcome is a possible value of a random process, so, one state in its state space

parallel in music theory, the parallel minor key of a major key is the minor key with the same root as the major key but different scale degrees to reflect the minor quality; the parallel major key of a minor key is likewise the major key with the same root as the minor key but with scale degrees (staff) to represent the major quality

probability mass function (pmf) the function describing the individual probabilities of every outcome in a system

quality in music theory, the nature of a chord, limited to major, minor, diminished and augmented. Chords with major quality have their third 4 half steps above the root and fifth 7 half steps above the root; those with minor quality have their third 3 half steps above the root and fifth 7 half steps above the root; diminished chords have a third 3 half steps above the root and fifth 6 half steps above the root; and augmented chords are characterized by a third 4 half steps above the root and a fifth 8 half steps above the root. "Quality" can also apply to intervals; for instance, a major second is 2 half steps above the root while a minor second is only 1 half step above the root

root the fundamental frequency of a chord, after which it is labeled with a Roman numeral

root of unity the nth roots of unity are defined as the set of all complex numbers that are 1 when raised to the power n. These are also called de Moivre numbers

scale degree an integer denoted $\hat{n}$ is called the nth scale degree of some scale, where $\hat{n}$ is the nth note in the scale

spectrum the positive real function of a variable f denoting frequency representing the Fourier transform of a signal over time

state space the possible values a random process X can take on, also thought of as its range

tap a frequency in the spectrum $X(f)$ of a signal $x(t)$ at which the impulse response is to be evaluated

third the frequency either 3 or 4 half steps above the fundamental frequency of a chord. This can also refer to an interval separating two notes by 3 or 4 half steps, depending if the quality of the interval is (respectively) minor or major