Tuning Systems, Perceptions of Dissonance, and Mathematical Models of Consonance

Brandon Cui

April 27, 2026

Word Count: 5476

Introduction

I: Background

Since the beginnings of music theory, both Western and Eastern scholars have sought to create a fundamental framework of notes that all complex music would rest on. Particularly, a *tuning system* is a "set of musical pitches from which subsets (scales) are employed for melody and harmony" (Hearne). These systems define the various pitches and frequencies that can be used by musicians, and thus, what kind of music people can compose and write. Over the centuries, different tuning systems were adapted and refined by the ancient Greeks, the Chinese, and, later on, the Europeans. As the systems evolved, more and more emphasis was placed on one of the most important concepts in modern psychoacoustics: dissonance.

Musical dissonance is defined by Merriam-Webster as "a mingling of sounds that strike the ear harshly" and "a clashing or unresolved musical interval or chord". More specifically, dissonance is the roughness between various frequencies of notes in either a 2-note interval/dyad or a 3-note chord (Terhardt). Every sound is made up of waves of different frequencies that oscillate, such as 440Hz - 440 oscillations per second - and 220Hz - 220 oscillations per second. Combinations of certain frequencies produce a characteristic beating or buzzing sound to human ears, and the clashing is what Western tuning systems have sought to minimize.

The primary tuning system in the last century has been *12-tone equal temperament* (12-TET). In modern music theory, twelve evenly-spaced notes are picked from the infinite spectrum of pitches to form virtually every single song and piece of Western music that has been released within the last hundred years. This dominance in music theory is in part due to the tuning system's effectiveness at minimizing perceptions of C/D. The intervals within 12-TET are

positioned such that there is, in general, a minimal amount of beating. This pleasing absence of dissonance is called *consonance*.

Human perceptions of *consonance and dissonance* (C/D) are not static; in fact, C/D has "... gone through a wide sequence of transformations" over the past two millennia (Tenney). For example, current Western music theory now deems more intervals as consonant than ancient Greek polymaths like Pythagoras and Archytas. Tuning systems are developed, favored, and forgotten as human perception of dissonance within cultures changes. Because of this, in the 21st century, as previous aspects of modern music theory are being stretched out and pushed to their very limits, distinct tuning and temperament systems have been developed with dissonance in mind. One of the popular ways to analyze the effectiveness of the systems is through mathematical models of consonance and dissonance.

Similar to tuning systems, dozens of math models of C/D were developed in the past fifty years. These aim to represent the physical and mental perception of dissonance by assigning quantitative metrics to every combination of pitches within a given range, most commonly in the form of a 2 or 3 dimensional graph of dissonance. Models that examine the dissonance of dyads are most common, though some can be extended to measure the dissonance of triads (3 notes). On paper, these are mathematically sound; however, how well do models predict the perception of dissonance within actual human populations? Specifically, are they effective in characterizing dissonance in humans living in the Southeastern United States?

II: Gap

The majority of prior studies on the effectiveness of the various models of C/D are purely mathematical and analytical, and thus do not test how they line up with real human perception of sound (Terhardt, Guillet, Cook, Erlich, Strohman). Much of the research is also outdated,

especially when considering the ever-evolving definitions of dissonance within cultures. For those papers that did have human experimenters (Plomp-Levelt), only one model was pitted for accuracy against humans; no study has been conducted where two or more dissonance models are compared side-by-side in a trial. Additionally, comparisons of various mathematical models at their effectiveness are also currently limited to dyads, and no research has been conducted on their effectiveness on triads. Finally, no source focuses on the Southeastern US; a large factor in perceptions of dissonance is psychological, with culture having a "dominant role in shaping aesthetic response to music" (McDermott et al.). Consequently, narrowing the range of people and cultures can more clearly reflect patterns in the community.

The study conducted here attempts to bridge the gap between abstract models and concrete human groups by analyzing data collected in perceptions of C/D and measuring how close the physical models are to multiple mathematical models of dissonance. This study aims to provide an understanding of the efficacy of them in both dyads and triads. With this, the study intends to answer the following question: To what extent are mathematical models of sensory and psychoacoustic dissonance effective at predicting the perception of dissonance of dyads and triads within 1200 cents among people in the Southeastern United States?

Literature Review

I: Tuning and Temperament Systems

I.I: Scientific Knowledge

As mentioned before, every single sound humans hear consists of combinations of oscillating sine waves within air. The frequency at which they oscillate per second is measured in Hertz (Hz). For example, when a musician hears "A=440Hz", the air molecules vibrate back and

forth at 440 times per second. However, pure sine waves don't exist in nature; instead, all sounds contain a *fundamental frequency*, and the material the sound is produced in creates additional pitches above the frequency. In instruments, integer multiples of the frequency are also exhibited. As an example, on the violin, playing an A at 440Hz will also produce 880Hz, 1320Hz, 1760Hz, and so on. These integer multiple frequencies of the fundamental (including the fundamental) are called *harmonics*, and the infinite series of numbers from 1, 2, 3, 4, etc., times the fundamental is aptly named *the harmonic series*. This harmonic information forms the basis of *timbre* and is the reason why instruments have unique sounds.

Whole numbers are extremely important in tuning, with one metric of intervals and chords being ratios. For example, an *octave* is a significant interval where two notes have the same note name but different pitches; it occurs when one note is double the frequency of the other, such as 440Hz and 880Hz. As such, it is a 2:1 ratio. Other common intervals include the perfect fifth (3:2), perfect fourth (4:3), and major third (5:4). In fact, this tuning system, where each interval is represented as an exact ratio between two numbers, is so prevalent that it has a name: just intonation (JI). Within just intonation exist categories called *limits*, where a p-*limit* refers to all intervals where the prime factorization does not contain a factor exceeding p. For example, the interval 7:6 would be within 7-limit JI but not 5-limit JI, the category most similar to 12-TET.

Because of these ratios, sound is not perceived linearly. Going up one octave doubles the frequency, but up two octaves quadruples the frequency. In order to combat this, musicians also measure in *cents*, a logarithmic scale where 1200 cents is equal to an octave (12 notes in 12-TET times 100 cents per note = 1200). Now, going up one octave is 1200 cents, while two octaves is 2400 cents.

The following formulas are used to convert from ratios to cents and vice versa:

$$cents(n, d) = 1200 \cdot \log_2(n/d) \qquad ratio = 2^{cents/1200}$$

With the concepts of harmonics, ratios, and cents cemented, it is now critical to understand the history of tuning systems.

I.II: History and Dissonance of Tuning

One of the very first tuning systems was developed by Pythagoras. In Pythagorean tuning, 12 notes are formed by repeatedly "stacking" perfect fifths, as they were considered to be the most consonant interval to the ancient Greeks. This produced a decently consonant sequence of notes that was used extensively to tune string instruments such as the lyre; however, due to the very nature of stacking 3:2 ratios, an extremely dissonant interval of about 23 cents was generated additionally, taking the name of a "wolf interval" due to the roughness resembling a wolf howling (Sethares, *Tuning, Timbre, Spectrum*).

Although the tuning system worked for millennia, by the 1500s, music has grown in complexity, and along with it, the very perceptions of what intervals are dissonant. To the ancient Greeks, while the perfect fifth and perfect fourth were consonant, the major third was dissonant and not included in Pythagorean tuning (Tenney). However, by the Renaissance, Europeans considered the third to be consonant. Thus, a new tuning system was devised: Quarter-comma meantone.

Quarter-comma meantone sacrifices the accuracy of the perfect fifth and trades it in for an exact representation of the major third to address the changing definitions of C/D. The system became widespread throughout medieval Europe during the sixteenth and seventeenth centuries. However, with it came the same problem that Pythagorean tuning faced: The wolf interval was even more pronounced, with a 35 cent deviation that caused notes to become extremely

dissonant and out of tune with each other. Music theorists began seeking a way to "close the octave" and eliminate the dissonant wolf fifth. 12-TET was developed.

12-TET trades exact representations of both the perfect fifth and the major third for the removal of the wolf fifth. In the tuning, twelve notes are now equally spaced apart, where each successive note is 1200/12=100 cents above the previous. With this system, musicians can now freely play in different keys without worrying about whether notes would clash *with each other* at the cost of accuracy for some intervals.

II: Mathematical Models of Dissonance

II.I: Human Tuning Systems in relation to Mathematical Models

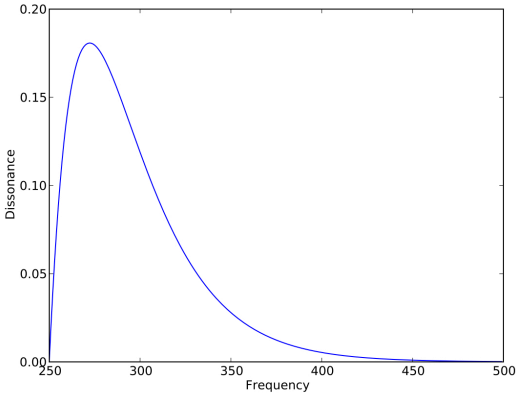
In a way, the tuning systems humans have developed over the course of centuries are a model themselves on perceptions of dissonance; certain tuning systems in time periods favor the consonance of an interval over the dissonance of another, forming a model of what groups of people viewed as dissonant. Therefore, it is important to acknowledge that human tuning systems are closely tied to the mathematical models that will be explained in the following section.

II.II: Scientific Explanation of Models

While analyzing models, it must be known that different models rely on different theories of the factors within human perception of dissonance. Some models, deemed here from now on as "acoustic-based", are based on human experiments and various acoustic metrics like harmonics. Other models are "complexity-based", in which just intonation ratios are input into mathematical functions, outputting a single number representing how "simple" the ratio is. One model is "entropy-based", where different frequencies of notes are inputted and analyzed through information theory to model the uncertainty of the human ear regarding the notes. Finally, this

study includes an unsophisticated model that classifies intervals based on historic perceptions of the interval.

The very first models of dissonance in recent history were *dissonance curves*, developed by Plomp and Levelt in *Tonal Consonance and Critical Bandwidth*. Researchers presented subjects with various intervals along the octave, then asked them to determine whether they were consonant or dissonant. Of note, most participants viewed notes within 200 cents of each other as extremely dissonant. To complement this, one of the most important curves in psychoacoustics was created; the Plomp-Levelt dissonance curve, shown in **Fig. 1**, models the *sensory dissonance* of two simple notes of pure sine waves. In other words, no additional information regarding timbre, sound, or harmonics is considered in the model. Because of the lack of contextual information, the curve is extremely simplistic and does not accurately represent perceptions of dissonance (Sethares, *Tuning, Timbre, Spectrum*). Instead, it is mostly used within other larger models. For example, Plomp and Levelt additionally introduces harmonic information to produce a complex curve modelling more complex sounds in **Fig. 2**.



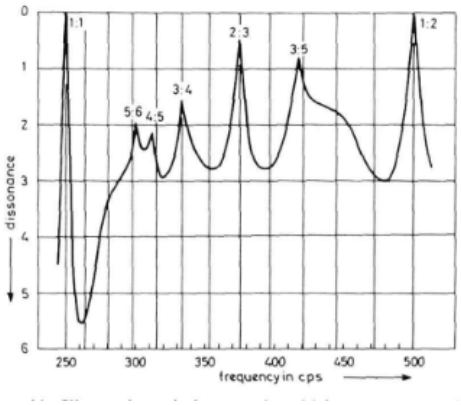


Fig. 1: The original 1965 Plomp-Levelt curve

Fig. 2: Constructed Plomp-Levelt graph

Despite the breakthroughs, there was no mathematical way to model the Plomp-Levelt graph. Thus, in 1993, William Sethares refined the 1965 experiment and "[explicitly parametrized]... the consonant curves" into the difference of two exponential functions, creating the first acoustic-based mathematical model with the form

$$d(x) = e^{-ax} - e^{-bx}.$$

More refinements upon his and Plomp's work were done by other scientists. Notably, Vassilakis's 2001 thesis and Cook's 2002, 2006, and 2009 papers all tweak various elements of Plomp-Levelt's findings. In all of these models, harmonic information is incorporated, allowing for timbre to be considered into the C/D output.

Around the same time, complexity-based models came into play. Among them were Tenney height, Naive norm (also known as Weil norm), and Wilson norm. Their mathematical formulas are as follows:

$$Tenney(x_1, \ldots, x_n) = \log_2(\prod_i x_i)$$

$$Naive(x_1, \ldots, x_n) = \log_2(max(x_1, \ldots, x_n))$$

$$\|H\vec{m}\|_1 = 2|m_1| + 3|m_2| + \ldots + p|m_{\pi(p)}| = \text{sopfr}(2^{|m_1|} \cdot 3^{|m_2|} \cdot \ldots \cdot p^{|m_{\pi(p)}|})$$

Complexity models are limited to just intonation intervals and take in the ratios as arguments. Tenney height computes the logarithm of the product of the intervals; Naive norm computes the logarithm of the largest ratio; Wilson norm computes the sum of all prime factors of all the ratios. As an example, for the 9:8 2-note interval:

$$Tenney(9, 8) = \log_2 9 \cdot 8 = \log_2 72 \approx 6.17$$

$$Naive(9, 8) = \log_2(max(9, 8)) = \log_2 9 \approx 3.17$$

$$Wilson(9,8) = Wilson(3^2 \cdot 2^3) = 3 + 3 + 2 + 2 + 2 = 12$$

So far, only one entropy-based model has been developed: Harmonic entropy, a relatively complex model to "measure the 'uncertainty' of the fit of a harmonic template to a … sound spectrum" (Sethares, "Local Consonance"). Uncertainty, through this, is conveyed through the concept of entropy in information theory, where one "bit" corresponds to a 50% uncertainty in outcome. In theory, a higher harmonic entropy corresponds to a higher uncertainty in the ear to distinguish sounds, which means that the interval is more dissonant (Strohman).

III: Analysis of Benefits and Downsides of the Categories of Dissonance Models

Each of the above mathematical models comes with benefits and detriments. For acoustic-based models, as mentioned, timbre can be easily incorporated to better convey information regarding the sound's fundamental nature to the human ear. Despite this, though, it is unclear whether harmonic information is the primary factor when predicting C/D perceptions.

Complexity-based models rely on the assumption that the human ear views numerically simplistic ratios as more consonant than complex ratios. Several psychoacoustic theories back up this claim, most notably Kameoka and Kuriyagawa's 1969 study. Because of the model relying on ratios, however, only just-intonated intervals can be measured, and harmonic information is not taken into account.

Finally, entropy-based models perform better in terms of modelling the human ear and can accept any real number, unlike complexity-based models. However, like complexity-based models, timbre is not factored into the algorithm, sacrificing potentially vital harmonic information.

Methodology

I: Quantitative Methods

This study focuses on ranking various mathematical models of dissonance in terms of effectiveness through ranked surveys of human test subjects within a sample population. According to the National Institutes of Health, mixed methods "strategically integrate or combine [rigorous quantitative and qualitative research methods] to draw on the strengths of each." The research method for this is a mixed method consisting of both quantitative experimental survey data across a large population and qualitative case studies within a small sample group.

Among the 11 mathematical models chosen are: the acoustic-based models exemplified by Sethares 1993, Vassilakis 2001, Cook 2002, Cook 2006, and Cook 2009; entropy-based models like Harmonic Entropy; complexity-based models like Tenney height, Naive norm, Wilson norm, and a custom-developed Continuous Fraction norm based on "convergent [sequences] of rational approximations" (Guillet); and the binary classification of common Western C/D intervals. These were chosen for the study because of the vastly different ways they quantify dissonance, illuminating particular aspects of perception. Furthermore, the models included are popular within the field of psychoacoustics and are numerical, allowing for comparisons between human data obtained quantitatively.

The process of collecting data points was adapted and modified from Masina, Presti, and Stanzial's research. Firstly, a 2-note interval was played, and participants were asked to rate a series of 26 intervals from a scale of 1 to 5, with 1 being very dissonant and 5 being very consonant. Additionally, participants were played 7 three-note triads and were prompted to also rate the clashing of the notes on the same scale, for a total of 33 intervals played. The timbre of

the complex notes consist of 10 harmonics with weights following *1/n*, e.g., the 3rd harmonic is present at ⅓ the volume, and the 4th harmonic is present at ¼ the volume. This reduces potential biases of sound and provides rich harmonic information for the models (Masina et al.). All intervals start on A=440Hz and fall within one octave (1200 cents), covering both 12-TET and 5-limit just intonation, as well as some additional intervals from other tuning systems like 31-TET, quarter-comma meantone, and 7, 11, 13, and 17-limit JI.

The raw data points for each mathematical model was simulated in a computer program written in Python and MatPlotLib via a Jupyter Notebook environment. For some more complex models, guidance was required to implement; specifically, the code for harmonic entropy frequently referenced the corresponding page on the Xenharmonic Wiki for mathematical information on algorithms, and the author utilized the dissonant package from the Python Package Index (PyPI) to manipulate all acoustic-based models. Scipy and Numpy's interpolation features for n-dimensional arrays were essential in processing this data into graphs and general trends. Afterwards, the just intonation representation of the 31 total just-intonated dyadic and triadic chords were plugged into all models. Note that 12-TET intervals (e.g 100 cents, 200 cents) cannot be inputted into complexity-based models like Tenney, since they cannot be represented exactly in just intonation; as such, complexity-based models are omitted from the data of two 12EDO intervals. After this, though, numerical values of dissonance should be generated for every model and chord.

These values are then used to generate graphs for each model, with the human C/D ratings of each chord plotted on the y-axis and the model dissonance rating of the chord plotted on the x-axis. If the model were to perfectly match the human ratings, then the result should be a linear graph similar to the equation y=x. As such, a linear regression was applied to each graph.

The r^2 , or coefficient of determination, of the regression corresponds to how well the graph fits the line; the closer the $r^2$is to 1, the closer the graph resembles a line, and the more accurate the model is at matching the human ratings. So, by comparing r^2values for each model, it is finally possible to accurately compare the effectiveness of the models at predicting human perceptions of dissonance.

Given properly structured data, this portion of the method is entirely replicable. The Python programs used to simulate the mathematical models are located in **Appendix A** as a sequence of Jupyter Notebook (IPyNB) code cells with relevant comments.

II: Qualitative Methods

Participants in the qualitative case study were also interviewed on their musical backgrounds and their previous experience with dissonance. Perception of dissonance can vary drastically from cultural differences; for example, even the physical altitude in which cultures are located fundamentally changes how they perceive C/D (Prete et al.). As such, it is important to understand the background of individuals within the local Southeastern area as to better verify the validity of the numerical data. The interview questions are also designed to match responses to certain mathematical models; as an example, a participant who responds that they enjoy listening to Western classical music supports the effectiveness of the binary classification model of Western classical C/D intervals. More interview questions are given in **Appendix B** for additional replicability.

Depending on the participant, the length of the interviews varied from 10 minutes to 20 minutes. Three interviews were conducted in English, and one interview was conducted in English and Mandarin Chinese. After verifying the participants' consent to being interviewed, an initial question was provided as a stepping stone. Interviewees were then allowed to elaborate,

with follow-up questions being asked to better guide them in an optimal conversation. Generally, the interview started by discussing the participant's musical tastes, followed by a discussion of the relevance of C/D within said genre, followed by the individual's own perceptions of dissonance as a whole, concluding with potential influences on perceptions of dissonance.

III: Ethical Considerations

Prior to all participants and interviewees inputting data, it was paramount that they gave consent to the usage of their data anonymously for research purposes. For the quantitative survey, individuals were required to fill out an Informed Consent Form before proceeding. Similarly, interviewees in the qualitative interviews gave consent through a Media Publication Consent Form. Both of these can be found in **Appendix C**. Finally, due to the involvement of human subjects, IRB approval was given to conduct this research project.

Data Analysis and Results

In the quantitative data collection portion, 43 respondents were surveyed online via Aidaform. As can be seen in **Fig. 3**, about two-thirds of the population surveyed had experience with music, whether from a music-theory standpoint or a performer standpoint. Therefore, pitch distinguishability within the population surveyed is also relatively high, with 51.16% ranking their pitch discrimination as either above average or excellent, shown in **Fig. 4**.

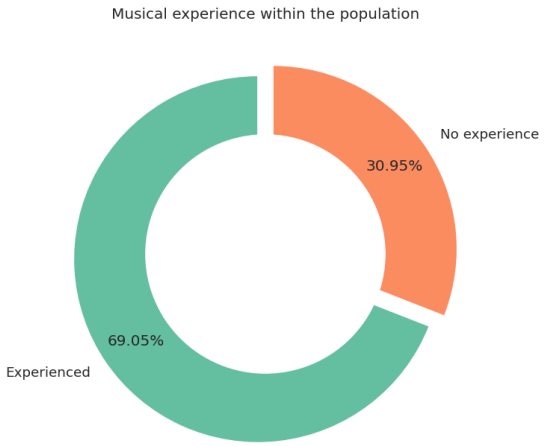


Fig. 3: Percentage of the population based on musical experience

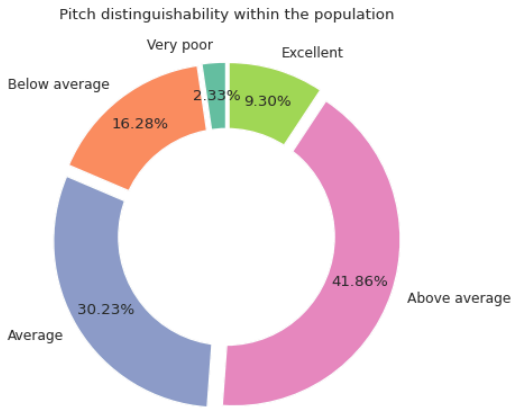


Fig. 4: Percentage of the population based on ability to distinguish between different pitches

I: Analysis of Dyads

Fig. 5 depicts the population's C/D rating for each of the 26 dyadic intervals. From a retrospective standpoint, the data obtained closely matches previously established notions of dissonance; local minima exist in many of the consonant 12-TET intervals like 200c, 300c, 400c. Similarly, local maximas exist in many non-5-limit just intonation intervals, like 347c (11:9), an

11-limit JI interval. Both of these instances support the previous assumptions that the prevalence of 12-TET in music has shaped people into also viewing non-5-limit JI intervals as dissonant (Bailes et al.). Participants ranked the perfect octave (1200c, 2:1) as the least dissonant interval and the smallest interval (32c, 33:32) as the most dissonant interval. Interestingly, participants voted 551c (11:8) as dissonant, confirming centuries-old stigma surrounding the dissonance of the close 12-TET 600c "devil's interval" — the tritone. Likewise, the JI approximation of the 100c semitone matches historical perceived dissonance in 12-TET.

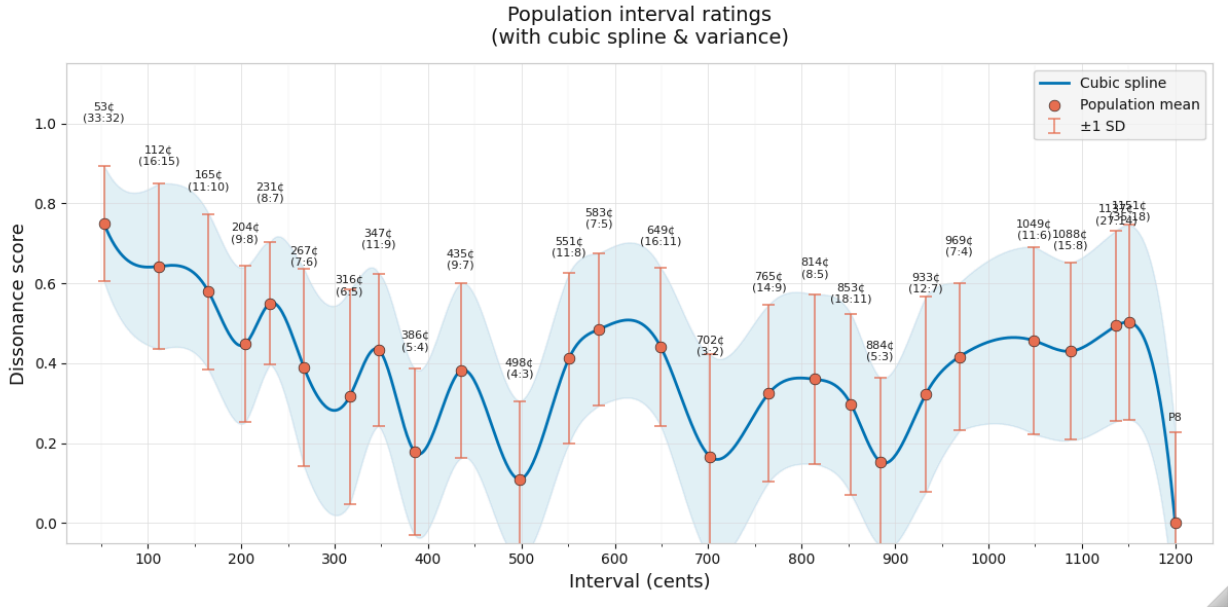
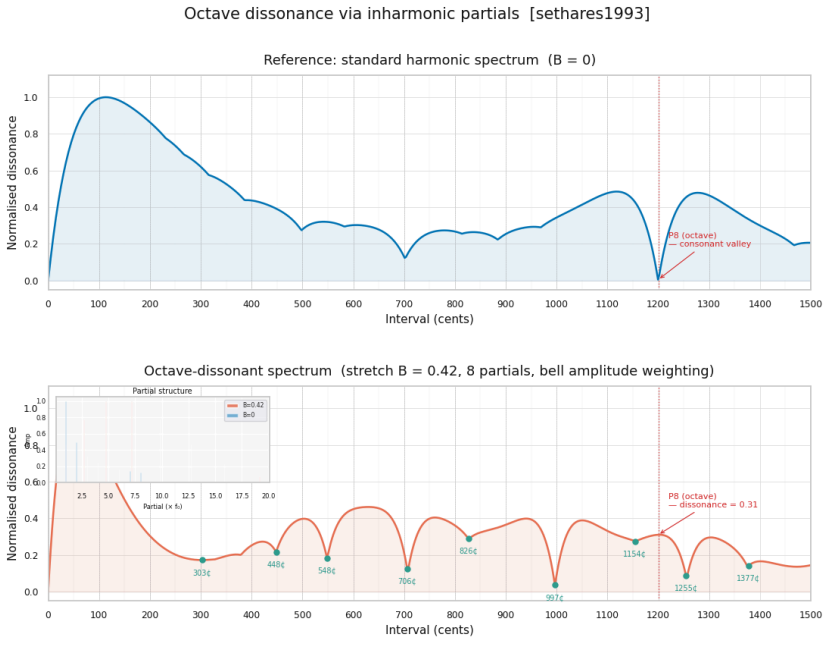


Fig. 5: Population's ratings of each of the 26 dyads. A score of 1 is most dissonant, while a score of 0 is least dissonant.

For the effectiveness of each model at predicting dyads, **Fig. 6** and **Fig. 8** show that the Continuous Fraction norm, a complexity-based model, was most successful in predicting the sample's perception of C/D. Indeed, as is evident in **Fig. 7**, complexity-based models scored the highest, with an r^2 value approximately 2.8% higher than acoustic-based models and 11.6%

higher than entropy-based models. This small discrepancy between complexity-based and acoustic-based models is particularly intriguing; in the landmark book *Tuning, Timbre, Spectrum, Scale*, the author argues that the timbral character of a sound and its harmonic information is the primary factor affecting perceptions of C/D (Sethares 1998). However, it appears that Kameoka and Kuriyagawa's theory of the correlation between simplistic frequency ratios of a dyad and its perceived dissonance outweighs both Sethares's and Plomp-Levelt's theory by a small percentage in this specific case.

It must be acknowledged, though, that harmonics and timbral information still play a fundamental role in shaping perceptions of consonance and dissonance. For example, instruments such as bells and chimes are *inharmonic* and produce a timbre such that even the perfect octave can become perceptually dissonant, as modeled below.



Most likely, the relatively clean and harmonic timbre of the sounds played to the participants diminished the importance of harmonic information, reducing the effectiveness of acoustic-based models (see **Fig. 9** for a graphical depiction) like Cook, Sethares, and Vassilakis.

Model Name	**Type of Model**	**Coefficient of Determination (r^2)**
Continuous Fraction Norm	complexity	0.706
Sethares 1993	acoustic	0.677
Cook 2009	acoustic	0.670
Cook 2006	acoustic	0.666
Tenney Height	complexity	0.666
Cook 2002	acoustic	0.662
Wilson Norm	complexity	0.658
Harmonic Entropy	entropy	0.585
Naive Norm	complexity	0.583
Binary Classification	historic	0.557
Vassilakis 2001	acoustic	0.500

Fig. 6: Coefficient of determination for each mathematical model of dissonance in reference to the population for dyads, sorted highest to lowest

Type of Model	**Number of Models**	**Average r^2**
Complexity	4	0.653
Acoustic	5	0.635
Entropy	1	0.585
Binary Classification	1	0.557
Total	**11**	**0.63**

Fig 7: Average coefficient of determination for each type of mathematical model of C/D in dyads

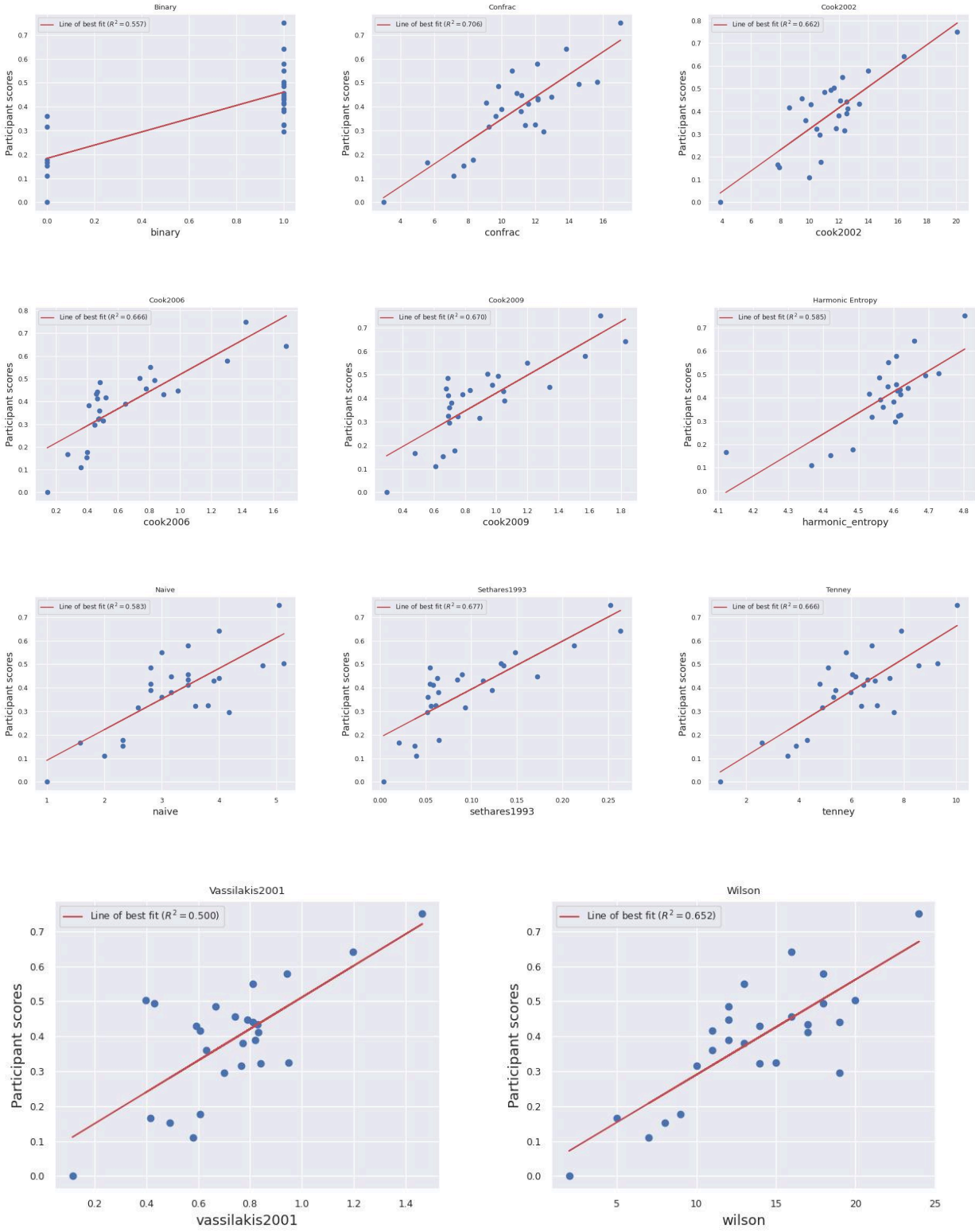


Fig. 8: Scatter plots of participant scores plotted against each dyad model, with the red line representing the line of best fit

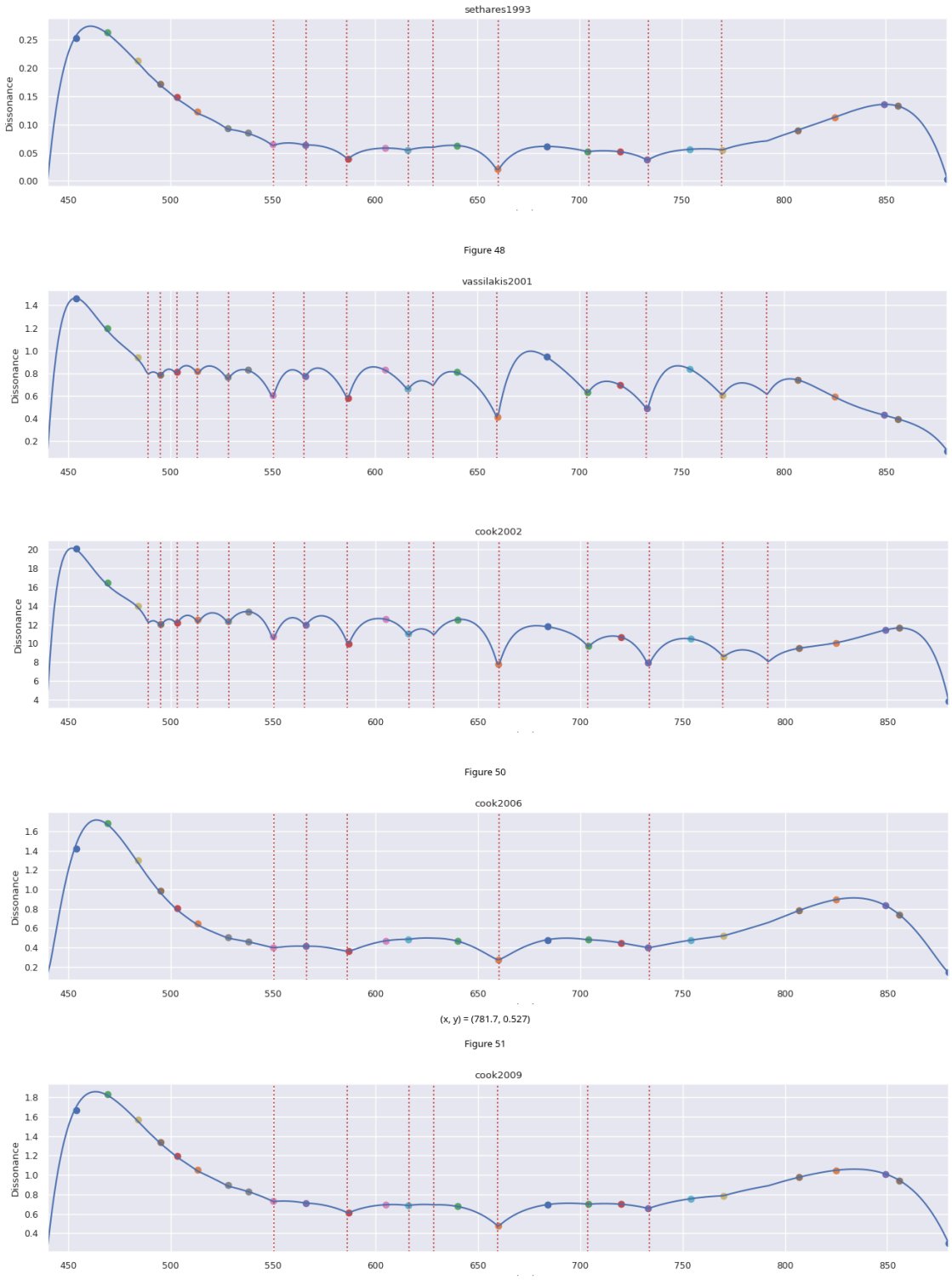


Fig. 9: Graphical representation of all acoustic-based models from 0 to 1200 cents (440-880 Hz)

Overall, from **Fig. 6**, the average of the r^2 values for all models on two-note intervals, 0.63, demonstrates a moderate correlation and decent accuracy of the models when predicting human perceptions of sensory dissonance. While not the highest possible correlation, an r^2 of 0.63 in the social science realm is more than acceptable for models predicting human behavior; lower coefficients of determination are not necessarily bad, and as long as there exist variables that can prove its significance, a 0.63 r^2 is reasonable (Ozili).

II: Analysis of Triads

Quantitative analysis of the 7 triads was performed similarly to the analysis of the 26 dyads. **Fig. 10** demonstrates the population's ratings of each of the 7 triads on a two-dimensional heatmap. Raw tabular data is also located in **Appendix D**. Similar to the dyadic case, the triadic data confirms previous general theories regarding dissonance. The octave-containing chord (2:3:4 JI) was rated the least dissonant triad, while the chord containing the closest intervals (160:165:176 JI) was ranked the most dissonant triad, verifying the dyadic theories presented in the previous section. Surprisingly, participants were able to discern the differences between JI triads and their 12-TET equivalent fairly well, with 12-TET triads being rated, on average, 2.63% more dissonant than their JI equivalent, as depicted in **Fig. 11**. Because just intonation, by definition, contains less beating due to perfect ratios, this shows that within the Southeastern U.S population, people tend to slightly favor just intonated triads over traditional 12-TET, despite being exposed to the tuning system much less than the latter system.

Regarding the effectiveness of the models for triads, **Fig. 12** and **Fig. 14** show that Wilson Norm, also a complexity-based model, scored the highest in r^2, with a value of 0.961.

However, unlike the dyadic case, the difference between the effectiveness of the types of models is much more pronounced; from **Fig. 13**, complexity-based models, on average, possessed r^2 values more than 11.2% higher than acoustic-based models (see **Fig. 15** for a graphical representation) and over 33% higher than the binary classification model. Combined with the previously established preference of just intonation over 12-TET, it is safe to assume that within the population, simple frequency ratios between various notes are valued more than harmonic information like timbre and partials. Note that the harmonic entropy model was not available for triads and was instead replaced with a custom adaptation of Vassilakis's 2001 acoustic model, adjusted from Aatish Bhatia's "Dissonance - A Journey" .

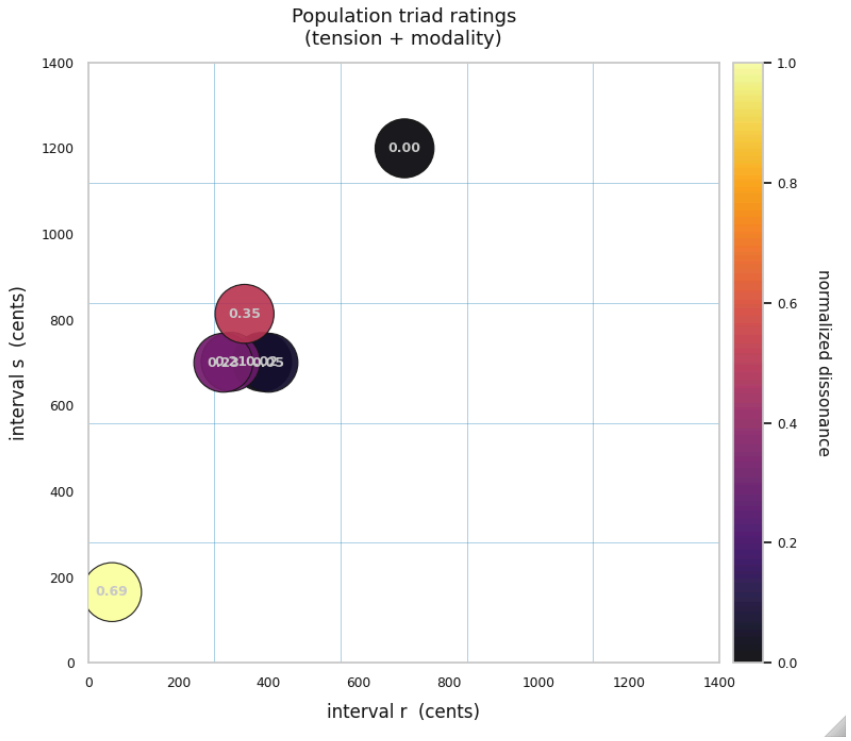


Fig. 10: Population's ratings of the 7 triads on a 2D heatmap. The second and third pitches are located on the x and y axes, and the color corresponds to the perceived dissonance

Chord Quality	**JI Consonance**	**12-TET Consonance**	**Percent Change**
Major (4:5:6, 0-400c-700c)	0.9768	0.9464	**-3.11%**
Minor (10:12:15, 0-300c-700c)	0.7905	0.7734	**-2.16%**

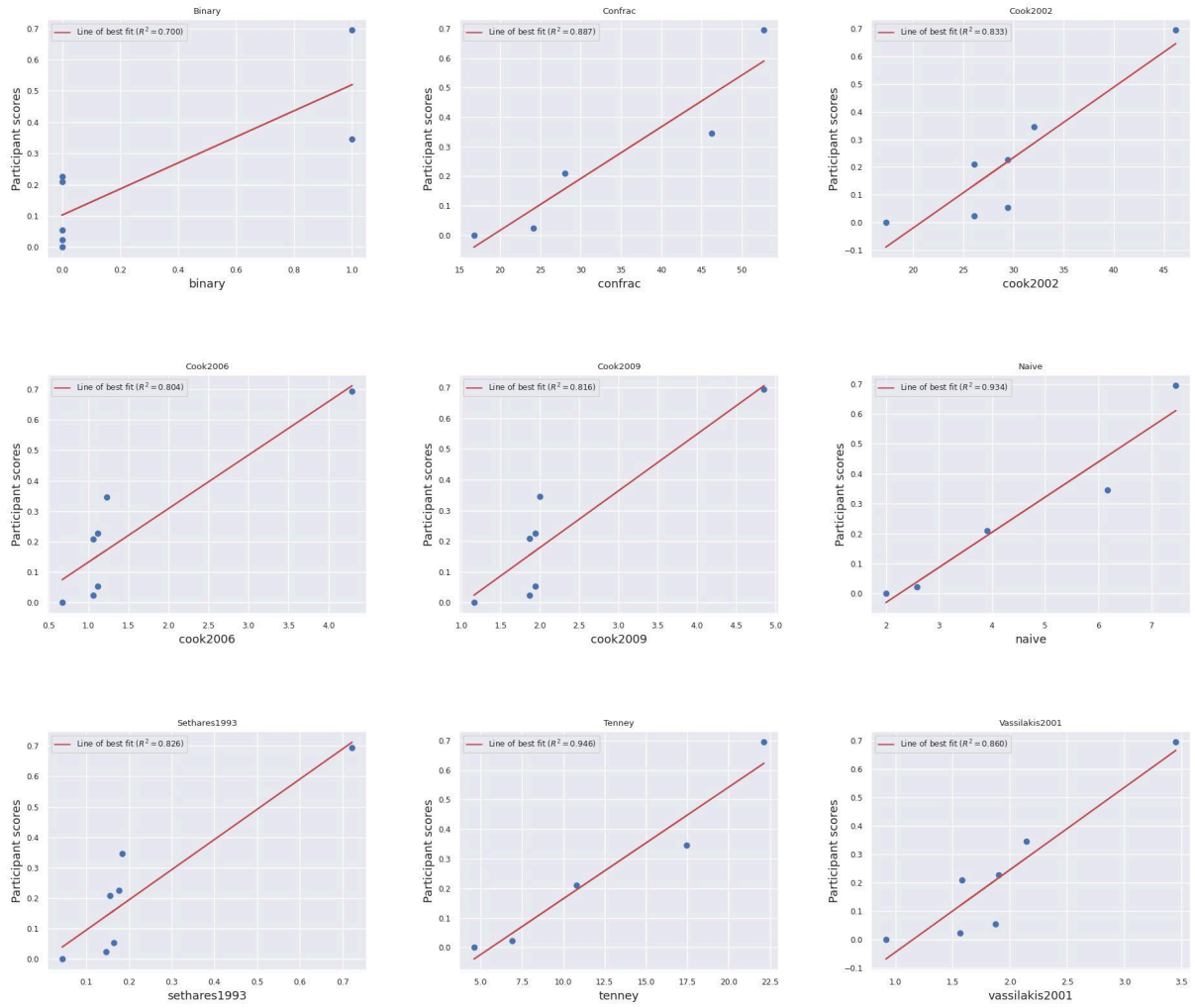
Fig. 11: Consonance ratings of 2 just intonated triads and their 12-TET counterparts

Model Name	**Type of Model**	**Coefficient of Determination (r^2)**
Wilson Norm	complexity	0.961
Tenney Height	complexity	0.946
Naive Norm	complexity	0.934
Vassilakis 2001 (custom)	acoustic	0.888
Continuous Fraction Norm	complexity	0.887
Vassilakis 2001	acoustic	0.860
Cook 2002	acoustic	0.833
Sethares 1993	acoustic	0.826
Cook 2009	acoustic	0.816
Cook 2006	acoustic	0.804
Binary Classification	historical	0.700

Fig 12: Coefficient of determination for each mathematical model of dissonance in reference to the population for triads, sorted highest to lowest

Type of Model	**Number of Models**	**Average r^2**
Complexity	4	0.932
Acoustic	6	0.8378
Binary Classification	1	0.700
Total	**11**	**0.8595**

Fig 13: Average coefficient of determination for each type of mathematical model of C/D in triads



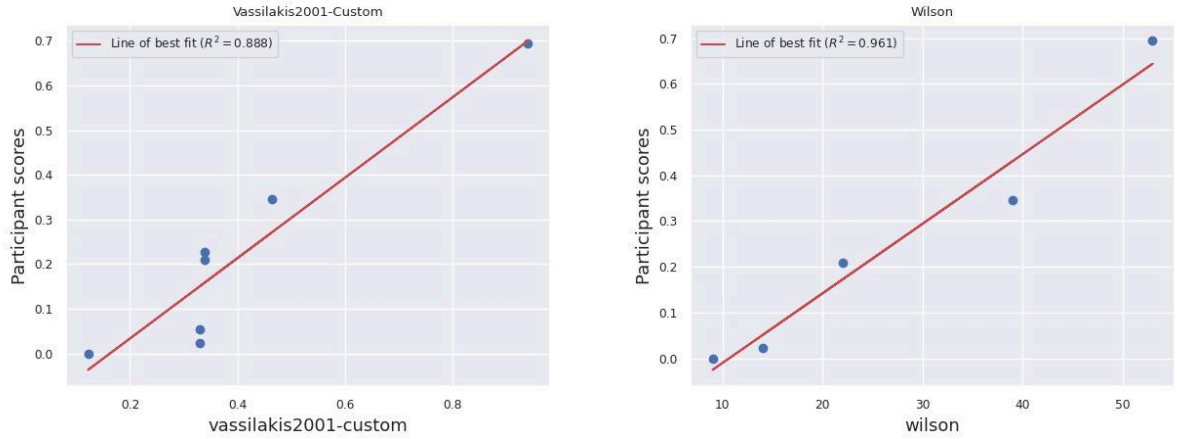


Fig. 14: Scatter plots of participant scores plotted against each triad model, with the red line representing the line of best fit

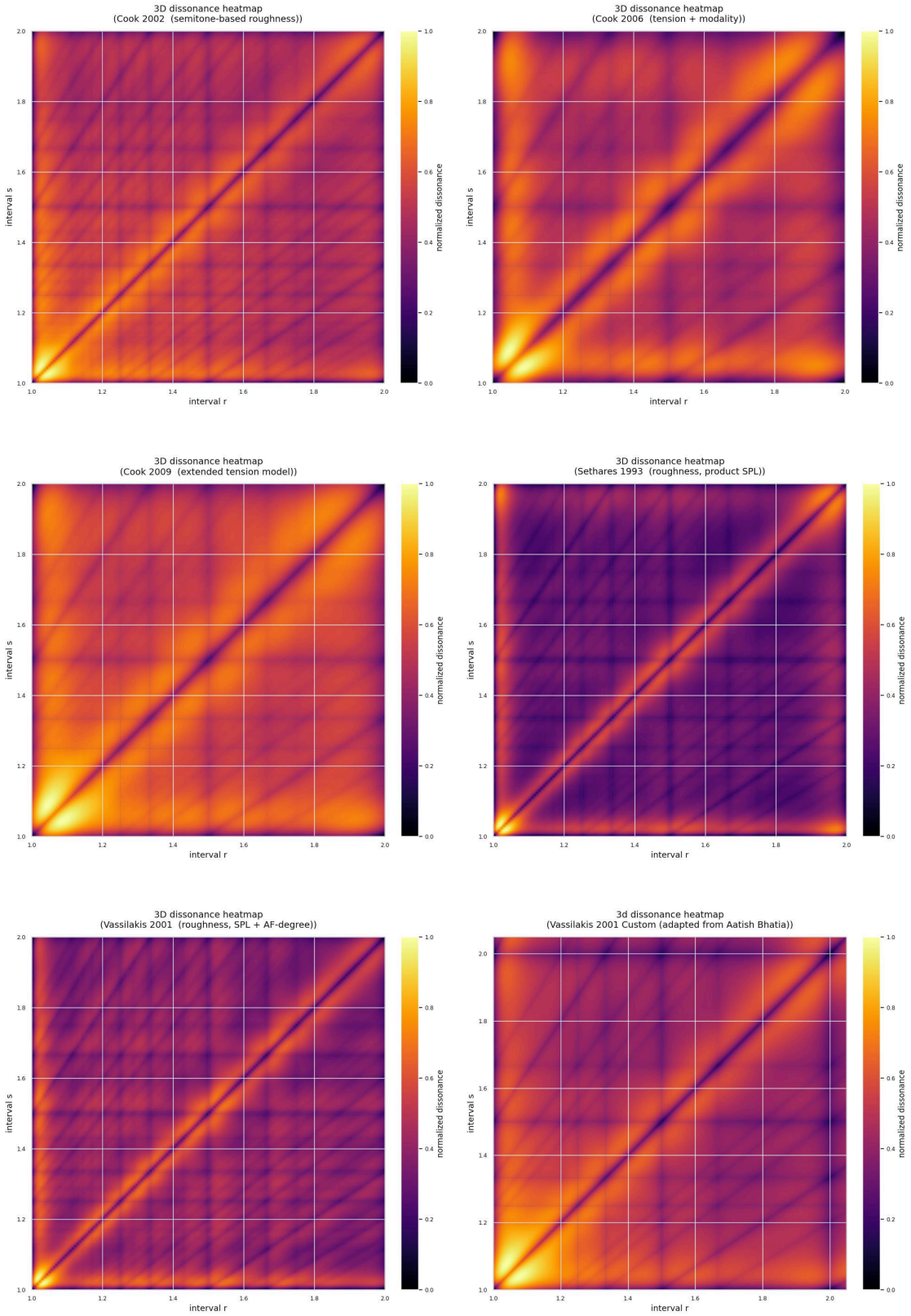


Fig. 15: Graphical representation of all acoustic triadic models from 0 to 1200 cents

Interestingly, the average of the r^2 values for all models on *triads*, 0.86, is much higher than that of dyads, as can be seen from **Fig. 13**. Thus, this demonstrates a strong correlation and excellent accuracy of mathematical models at modelling perceptions of C/D within humans.

III: Thematic Analysis of Interviews

Thematic analysis, defined as a method for "identifying, analysing, and reporting patterns… within data" (Braun and Clarke), was performed to analyze the data obtained through interviews. Transcriptions of the recordings of participants were completed and sorted into sets, which were then assigned themes accordingly. The findings are presented in **Figs. 16 & 17.**

Theme	**Definition**
Classical Music Consonance	Participants stated that classical music, specifically the Baroque, Classical, and Romantic eras, is especially consonant compared to the music of today. Codes include dissatisfaction with current music tastes, consonance of intervals like the major third, and musical tension.
Jazz Relative Dissonance	Participants stated that jazz music, when compared to Western classical music, is intentionally more dissonant, and that the cause is a change in perception of harmony. Codes include music theory, jazz harmony, dialogue, and auditory cues.
Appreciation of Dissonance as a Musical Feature	Individuals noted that dissonance can be a desired feature and mode of music to convey emotions and that perceptions of dissonance can change as exposure to dissonance varies. Codes include auditory environment, musical expressivity, contemporary thought, music theory, and ear training.
Frequency Ratios	Individuals noted that dissonance was particularly noticeable as a subtle wobble in air vibrations, primarily due to the simplicity

	within the ratios of multiple notes. Codes include complexity-based models, sensory dissonance, acoustics, and sound waves
Harmonic Information	Participants stated that harmonic information played a large role in determining whether a sound was dissonant or consonant (e.g tubular bells versus clarinet). Codes include acoustic-based models, timbre, and partials.

Fig 16: Definition of themes

Individual Number	**Interview Summary**	**Themes**
1	Indiv. believes classical music features less dissonance than the modern music they listen to	Classical Music Consonance
	Believes that dissonance can be used sparingly to convey tension	Jazz Relative Dissonance
	Is interested in microtonal music theory and how to utilize different tuning systems to better capture emotions	Appreciation of Dissonance as a Musical Feature
	Having been trained classically, indiv. is taught to tune by way of listening to frequency ratios “wobbling”	Frequency Ratios
2	Listens to classical music regularly (Yundi Li), prefers traditionally more consonant Mozart to more impressionistic Rachmaninoff	Classical Music Consonance
	Agrees that jazz is more dissonant than classical music	Jazz Relative Dissonance
	Upon listening to a microtonal recording, recognizes the irregular beating of various ratios as dissonant	Frequency Ratios
	Ranks some instruments as more dissonant / more consonant than others due to harmonics present within the timbre	Harmonic Information
3	Views that developments in jazz harmony within the last 100 years has contributed to an increase in dissonance compared to classical music	Jazz Relative Dissonance

	Believes that improvisation can play a major factor in shaping perceptions of dissonance due to inherent imperfections while improvising	Appreciation of Dissonance as a Musical Feature
	Tunes mainly through the acoustic properties of the sway of the ratios when tuning with the ear	Frequency Ratios
	Ranks percussive instruments like the tenor drum as more dissonant than wind instruments due to both harmonics and inharmonic partials	Harmonic Information
4	Listens to classical music occasionally, ranks Baroque as most consonant, followed by Classical, Romantic, and Impressionistic	Classical Music Consonance
	Believes jazz theory such as tritone substitution and extended chords give a more dissonant taste	Jazz Relative Dissonance
	Views dissonance as a necessary and enjoyable factor in order to add tension to pieces, which in turn adds resolution	Appreciation of Dissonance as a Musical Feature
	As a guitarist, strings can vibrate at certain speeds and ratios, producing intervals of different C/D values	Frequency Ratios
	With regards to instruments with discernable pitches, does not have a definite ranking of instruments' dissonance for timbre; views them as about equal	Harmonic Information

Fig 17. A summary of the four qualitative interviews

Discussion and Conclusion

I: Findings of Thematic Analysis

I.I: Themes One and Two: Classical Music Consonance / Jazz Relative Dissonance

Almost all interviewees viewed classical music as a consonant genre of music, while all interviews viewed jazz as a more dissonant-heavy genre. For example, individual #4 stated that jazz theory "leans into both musical dissonance and the acoustic parts of it" and that, in their opinion, "dissonance functions as a method to establish a baseline so that resolutions feel more

powerful". As such, understanding how humans perceive dissonance can provide a better understanding of how dissonance can function in the context of music.

Western classical music is usually divided into four historical eras: Baroque, from 1650 to 1750; Classical, circa 1750 to 1820; Romantic, from 1800 to 1910; and Contemporary, from 1910 onward (González-Espinoza et al.). Of the individuals who listen to classical music, most described the Baroque and Classical era as the most consonant, followed by the Romantic and Contemporary. Individual #1 noted that "beginning in the 1800s, composers began to experiment more with dissonance, with the mid-1900s marking a huge shift towards extremely atonal, dissonant, and strange music". From both of these trends, and combined with previously mentioned academic research from Tenney, it is evident that perceptions of dissonance within humans can change drastically, which explains the relatively low success rate of the binary classification model; with r^2 values of 0.557 and 0.700 for dyads and triads, the model performed 11.6% and 18.6% worse than the average, respectively.

I.II: Theme Three: Dissonance as a Feature

Most interviewees agreed that dissonance can play a role in the structure of music; for instance, individual #3 expressed that "dissonance can invoke a sense of 'improvisation' and spontaneousness within music". On the other hand, individual #2 argued that dissonance "is an unwanted effect that shouldn't have a role in music". In other words, the range of opinions on dissonance seems to be wide; no matter how good a mathematical model is, it cannot fully encompass and satisfy all possible human perceptions of C/D.

I.III: Themes Four and Five: Frequency Ratios / Harmonic Information

All interviewees cited frequency ratios as a measure of identifying dissonance within sounds. In particular, individuals with experience playing instruments frequently utilized the

inherent disparities within small ratios to listen to faint "wobbles" in the sound. Individual #1 notes: "The faster the wobble, the bigger the ratio, and the more out-of-tune someone is. The slower the wobble, the smaller the ratio". Even non-experienced participants, such as individual #2, were able to recognize dissonance through this method; when playing a recording of a piece in just intonation, the ubiquity of frequency ratios demonstrates that complexity-based models, by utilizing the exact same JI ratios, can prove to be extremely effective at modeling perceptions of dissonance within the Southeastern US population.

Similarly, most interviewees agreed that harmonic information is a crucial factor in determining dissonance in acoustics. For example, individual #3 noted that certain musical instruments, such as flutes and violins, were naturally more consonant than others, like oboes and bassoons. Since timbre is defined by harmonics, acoustic-based models that take harmonics into consideration are effective in predicting human perceptions of dissonance. However, individual #4 stated that there "was no discernible difference between instruments regarding dissonance", reinforcing the idea that even the concept of dissonance can be subject to interpretation. Because more interviewees identified frequency ratios as a primary factor in determining dissonance than harmonic information, though, complexity-based models appeared to produce more accurate results than acoustic-based models.

II: Effectiveness of Mathematical Models and New Understanding

In general, most mathematical models performed well at predicting human perceptions of C/D in the Southeastern US, with complexity-based models consistently outperforming acoustic-based, entropy-based, and historic models. Models fared exceptionally well for triads, with an average r^2 value 36% higher than that of dyads. This could be due to many factors, such as the nature of the triads chosen. The addition of a note likely lowered potential discrepancies of

C/D between participants due to supplementary harmonic and pitch information present, creating a "general consensus" in what is considered consonant. In contrast, dyads, with only two notes, may potentially offer a wider range of C/D perception within the participants.

As previously mentioned, complexity-based models likely outpredicted acoustic-based models due to the relatively clean harmonic information of the timbre played. The weights of 1/n up to n=10 produced a slightly buzzy yet distinctly sinusoidal sound. As a result, harmonic information was added to a lesser degree, reducing the advantage that acoustic-based models would normally bring.

III: Potential Applications and Implications

In the future, tuning systems will continue to develop; *microtonal* tuning systems — anything not 12-TET — have especially been looked into in the last few years, and music theory revolving around them puts a heavy emphasis on dissonance in terms of defining their usefulness (Bailes et al.). With the findings that complexity-based models exceed acoustic-based models, microtonal music theorists can develop systems based on how prevalent dissonance is to be; JI and JI-based temperament systems will feature less dissonance, while TETs may feature more dissonance. This understanding can then be used to create new music in these tuning systems, expanding the scope of modern music theory.

IV: Limitations and Future Research

Before concluding, it is important to address several limitations of this study regarding both methodology and analysis. Due to time constraints, 43 responses were collected in the quantitative survey. With more numerical data, the accuracy of the mathematical models would increase.

Similarly, only seven triads were played to participants, as opposed to 26 dyads played due to time limitations for each participant. If a wider range of triads were to be included, the mathematical model accuracy for triads may adjust and become more similar to that of dyads.

Finally, triadic harmonic entropy (3-HE) was not included for triads. Information on harmonic entropy is incredibly limited online, with almost all research focusing on dyadic HE and coming from its creator, Paul Erlich. Contact to him was unavailable, and as such, code was unable to be written to simulate three simultaneous notes.

Future studies could overcome these limitations by sampling and evaluating more triads, as well as successfully modeling 3-HE in order to better keep models constant between dyads and triads. Additionally, four-note clusters could also be considered, allowing for the better establishment of a trend between the number of pitches and perceptions of dissonance.

Sources Cited

Bailes, Freya, et al. “How Different Are Our Perceptions of Equal-Tempered and Microtonal Intervals? A Behavioural and EEG Survey.” PLOS ONE, Public Library of Science, 18 Aug. 2015, journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0135082

Bhatia, Aatish. “Dissonance - A Journey Through Musical Possibility Space.” Aatish Bhatia, aatishb.com/dissonance/. Accessed 14 Sept. 2025.

Braun, Virginia, and Victoria Clarke. “Using Thematic Analysis in Psychology.” Qualitative Research in Psychology, vol. 3, no. 2, Jan. 2006, pp. 77–101, doi:10.1191/1478088706qp063oa.

Cook, Norman D. “Calculation of the acoustical properties of triadic harmonies.” The Journal of the Acoustical Society of America, vol. 142, no. 6, 1 Dec. 2017, pp. 3748–3755, https://doi.org/10.1121/1.5018342

Erlich, Paul. “The Forms of Tonality.” Tonalsoft, 2001, tonalsoft.com/enc/e/erlich/erlich-tFoT.pdf

González-Espinoza, Alfredo, et al. “Arrow of Time across Five Centuries of Classical Music.” Physical Review Research, vol. 2, no. 3, 29 July 2020, doi:10.1103/physrevresearch.2.033166.

Guillet, Alexandre. “Elementary Spectrum for the Dissonance Curve: From Biophysics to Number Theory of Musical Harmony.” Accueil - Archive Ouverte HAL, 21 June 2025, hal.science/hal-05124442/

Hearne, Lillian M. “The Cognition of Harmonic Tonality in Microtonal Scales.” Western Sydney University, 2020, researchers-admin.westernsydney.edu.au/ws/portalfiles/portal/94900730/uws_58606.pdf

Kameoka, Akio, and Mamoru Kuriyagawa. “Consonance Theory Part II: Consonance of Complex Tones and Its Calculation Method.” SciHub, 3 Feb. 1969, 2024.sci-hub.se/2124/94ae586f76c1bb302609dda36b5963b1/kameoka1969.pdf

Ozili, Peterson K. “The Acceptable R-Square in Empirical Modelling for Social Science Research by Peterson K Ozili :: SSRN.” Social Science Research Network, 17 June 2022, papers.ssrn.com/sol3/papers.cfm?abstract_id=4128165. Accessed 02 Apr. 2026.

Plomp, R, and J M Levelt. “Tonal Consonance and Critical Bandwidth.” Max Planck Institute for Psycholinguistics, 26 Apr. 1965, www.mpi.nl/world/materials/publications/levelt/Plomp_Levelt_Tonal_1965.pdf

Prete, Giulia, et al. “Universality vs Experience: A Cross-Cultural Pilot Study on the Consonance Effect in Music at Different Altitudes.” PeerJ, vol. 8, 7 July 2020, doi:10.7717/peerj.9344.

Sethares, William A. “Local Consonance and the Relationship between Timbre and Scale.” William A. Sethares (Bill), 25 May 1993, sethares.engr.wisc.edu/papers/consance.html.

Sethares, William A. Tuning, Timbre, Spectrum, Scale. Springer, 2005

Strohman, Gregory J. “Psychoacoustic Entropy Theory And Its Implications For Performance Practice .” TUScholarShare, May 2014, scholarshare.temple.edu/server/api/core/bitstreams/195ff4f6-f409-468e-96a4-f43767cf8b95/content

Terhardt, Ernst. “The concept of musical consonance: A link between music and psychoacoustics.” Music Perception, vol. 1, no. 3, 1984, pp. 276–295, https://doi.org/10.2307/40285261

Appendix A: Full Code

I: Pip installation

```
pip install ipympl
pip install dissonant
pip install scipy
pip install pandas
pip install seaborn
```

II: Initial complexity-based

```
# Tenney, other stuff
from math import log, prod

def tenney(*args):
    return log(prod(args), 2)

# also called naive
def weil(*args):
    return log(max(args), 2)

def cents_to_frequency(base, cents):
    return base * 2 ** (cents / 1200)

def ratio_to_cents(n, d):
    return 1200 * log(n / d, 2)

def sum_all_prime_factors(n):
    total_sum = 0
    i = 2
    temp = n

    while i * i <= temp:
        while temp % i == 0:
            total_sum += i
            temp //= i
        i += 1
```

```
    if temp > 1:
        total_sum += temp

    return total_sum

def wilson(*args):
    return sum(map(sum_all_prime_factors, args))

def rational_to_continued_fraction(n, d):
    terms = []
    while d != 0:
        quotient, remainder = divmod(n, d)
        terms.append(int(quotient))
        n, d = d, remainder
    return terms

def confrac(n, d):
    continued_frac_terms = rational_to_continued_fraction(n, d)
    return log(len(continued_frac_terms) ** 2 *
sum(continued_frac_terms) * prod(continued_frac_terms) *
continued_frac_terms[0] * d, 2)

def confrac_triad(a: int, b: int, c: int) -> float:
    a, b, c = sorted([a, b, c])
    if a <= 0 or b <= 0 or c <= 0:
        raise ValueError("All triad members must be positive
integers.")
    if a == b or b == c:
        raise ValueError("Triad members must be distinct (no
unisons).")

    score_ba = confrac(b, a)   # interval: root → middle note
    score_ca = confrac(c, a)   # interval: root → top note
    score_cb = confrac(c, b)   # interval: middle → top note

    return score_ba + score_ca + score_cb
```

III: Harmonic Entropy

```
# Harmonic Entropy
import numpy as np
from math import *

a = 1.0000000001
SIGMA = (log(0.01 + 1, 2) * 1200)

def calculate_ratios(n: int = 10000):
    # "Tenney height" where num:denom; num*denom < n

    ratios = []
    for rolling_n in range(1, n + 1): # 1, 2, 3, .... 10k
        for factor in range(1, int(rolling_n ** 0.5) + 1): # Max
factor is sqrt before it loops again
            other_factor = rolling_n // factor
            if rolling_n % factor == 0 and gcd(factor, other_factor)
== 1:
                ratios.append((factor, other_factor))
                if factor != other_factor:
                    ratios.append((other_factor, factor))


    return ratios

ratios = calculate_ratios()

def padded_list(length: int):
    l = 2 ** ceil(log(length, 2))
    if l == length:
        return [0.0 for _ in range(l * 2)]
    return [0.0 for _ in range(l)]

def convolve(a, b):
    a_fft = np.fft.fft(a)
    b_fft = np.fft.fft(b)

    a_real = [i.real for i in a_fft]
    a_imag = [i.imag for i in a_fft]
```

```
    # print(a_real, a_imag)

    inorm = 1 / len(a)

    for i in range(len(a)):
        temp = a_real[i]
        a_real[i] = (a_real[i] * b_fft[i].real - a_imag[i] *
b_fft[i].imag) * inorm
        a_imag[i] = (temp * b_fft[i].imag + a_imag[i] *
b_fft[i].real) * inorm

    new = [a_real[i] + (a_imag[i] * 1j) for i in range(len(a))]
    return [i.real * len(a) for i in np.fft.ifft(new)]

def harmonic_entropy():
    padding = round(100 * SIGMA)
    # padding = 0
    min = 0 - padding
    max = 1200 + padding

    k = []
    ak = []
    for i in range(max - min + 1):
        k.append(0.)
        ak.append(0.)

    for ratio in ratios:
        rcent = 1200 * log(ratio[0] / ratio[1], 2)
        rcompl = sqrt(ratio[0] * ratio[1])

        if rcent < min or rcent > max:
            continue

        if rcent == round(rcent):
            index = round(rcent) - min
            k[index] += 1 / rcompl
            ak[index] += 1 / pow(rcompl, a)
        else:
            clow = ceil(rcent) - rcent
```

```
            chigh = rcent - floor(rcent)

            index = floor(rcent) - min
            k[index] += 1 / rcompl * clow
            k[index + 1] += 1 / rcompl * chigh

            ak[index] += 1 / pow(rcompl, a) * clow
            ak[index + 1] += 1 / pow(rcompl, a) * chigh

    oldlen = len(k)
    minlen = 1
    while minlen < oldlen:
        minlen *= 2

    # minlen = len(k)
    for _ in range(minlen - oldlen):
        k.append(0.)
        ak.append(0.)

    g = [0. for _ in range(minlen)]
    ag = [0. for _ in range(minlen)]
    g_sum = 0

    for i in range(minlen):
        c = i
        gval = (1/(SIGMA*2*pi)) * exp(-(pow(c,2)/(2*SIGMA*SIGMA))) +
(1/(SIGMA*2*pi)) * exp(-(pow(c-(minlen),2)/(2*SIGMA*SIGMA)))
        g[i] = gval
        g_sum += gval

    for i in range(minlen):
        g[i] /= g_sum
        ag[i] = pow(g[i], a)

    ent = convolve(ak, ag)
    nrm = convolve(k, g)

    out = [0. for _ in range(1201)]
    for i in range(padding, oldlen - padding):
```

```
            outval = (1 / (1 - a)) * log(ent[i]/pow(nrm[i], a))
            out[i - padding] = outval

        return out
```

IV: Vassilakis 2001 Custom (Adopted from Aatish Bhatia)

```
def amp_to_loudness(amp):
    dB = 20 * math.log10(amp)
    loudness = 2 ** (dB / 10) / 16
    return loudness

def dissonance(f1, f2, l1, l2):
    x = 0.24
    s1 = 0.0207
    s2 = 18.96
    fmin = min(f1, f2)
    fmax = max(f1, f2)
    s = x / (s1 * fmin + s2)
    p = s * (fmax - fmin)
    b1 = 3.51
    b2 = 5.75
    l12 = min(l1, l2)
    return l12 * (math.exp(-b1 * p) - math.exp(-b2 * p))

def get_data_3d(spectrum, ref_freq, max_interval, step_size_3d):
    freq_array = spectrum["freq"]
    amp_array = spectrum["amp"]
    num_partials = len(freq_array)
    loudness_array = [amp_to_loudness(a) for a in amp_array]

    data_array = []
    max_z = 0

    r = 1.0
    while r <= max_interval:
        data_array_slice = []
        s = 1.0
        while s <= max_interval:
```

```
            dissonance_score = 0
            for i in range(num_partials):
                for j in range(num_partials):
                    f1 = ref_freq * freq_array[i]
                    f2 = ref_freq * freq_array[j]
                    l1 = loudness_array[i]
                    l2 = loudness_array[j]
                    d = (
                        dissonance(f1, f2, l1, l2)
                        + dissonance(r * f1, r * f2, l1, l2)
                        + dissonance(f1, r * f2, l1, l2)
                        + dissonance(s * f1, s * f2, l1, l2)
                        + dissonance(f1, s * f2, l1, l2)
                        + dissonance(r * f1, s * f2, l1, l2)
                    )
                    dissonance_score += d
            dissonance_score /= 2
            data_array_slice.append(dissonance_score)
            if dissonance_score > max_z:
                max_z = dissonance_score
            s += step_size_3d
        data_array.append(data_array_slice)
        r += step_size_3d

    x_data = []
    r = 1.0
    while r < max_interval:
        x_data.append(r)
        r += step_size_3d
    y_data = x_data[:]

    z_data = [[val / max_z for val in row] for row in data_array]

    return x_data, y_data, z_data

def lookup_dissonance(r, s, x_data, y_data, z_data):
    # Find nearest grid indices
    i = min(range(len(x_data)), key=lambda idx: abs(x_data[idx] - r))
    j = min(range(len(y_data)), key=lambda idx: abs(y_data[idx] - s))
```

```
    return z_data[i][j]


# 3D Generation
spectrum = {
    "freq": [1, 2, 3, 4, 5, 6],
    "amp": [1, 1/2, 1/3, 1/4, 1/5, 1/6]
}
ref_freq = 261.6256
max_interval = 2.05
step_size = 0.0025

x_data, y_data, z_data = get_data_3d(spectrum, ref_freq,
max_interval, step_size)
def plot_heatmap(r_vals, s_vals, Z, model: str,
                 output_path: str = None, show: bool = False):
    """Render a single heatmap in the reference style."""

    z_min, z_max = Z.min(), Z.max()
    Z_norm = (Z - z_min) / (z_max - z_min) if z_max > z_min else
np.zeros_like(Z)

    bg, fg = "#ffffff", "#111111"

    fig, ax = plt.subplots(figsize=(10, 9), facecolor=bg)
    ax.set_facecolor(bg)

    im = ax.imshow(
        Z_norm.T,
        origin="lower",
        extent=[r_vals[0], r_vals[-1], s_vals[0], s_vals[-1]],
        aspect="equal",
        cmap="inferno",
        interpolation="bilinear",
        vmin=0, vmax=1,
    )

    cbar = fig.colorbar(im, ax=ax, fraction=0.046, pad=0.04)
    cbar.set_label("normalized dissonance", color=fg, fontsize=12)
```

```
    cbar.ax.yaxis.set_tick_params(color=fg)
    plt.setp(cbar.ax.yaxis.get_ticklabels(), color=fg)

    ax.set_xlabel("interval r", color=fg, fontsize=13)
    ax.set_ylabel("interval s", color=fg, fontsize=13)
    ax.set_title(
        f"3D dissonance heatmap\n({MODEL_DESCRIPTIONS.get(model,
model)})",
        color=fg, fontsize=14, pad=14,
    )
    ax.tick_params(colors=fg)
    for spine in ax.spines.values():
        spine.set_edgecolor("#cccccc")

    # JI guide lines
    for label, ratio in JI_GUIDES.items():
        if r_vals[0] <= ratio <= r_vals[-1]:
            ax.axvline(ratio, color="#0077b6", linewidth=0.5,
alpha=0.4, linestyle="--")
            ax.axhline(ratio, color="#0077b6", linewidth=0.5,
alpha=0.4, linestyle="--")

    plt.tight_layout()

    if output_path:
        fig.savefig(output_path, dpi=150, bbox_inches="tight",
                    facecolor=fig.get_facecolor())
        print(f"  Saved → {output_path}")

    if show:
        plt.show()
        plt.savefig(f"triadic-acoustic/{model}.png")
    else:
        plt.close(fig)
# Processing of 3D
print(lookup_dissonance(1.5, 1.5, x_data, y_data, z_data))

z_arr = np.array(z_data)
```

```
fig, ax = plt.subplots(figsize=(10, 9), facecolor='white')
ax.set_facecolor('white')

im = ax.imshow(
    z_arr.T,
    origin='lower',
    extent=[x_data[0], x_data[-1], y_data[0], y_data[-1]],
    aspect='equal',
    cmap='inferno',
    interpolation='bilinear',
    vmin=0, vmax=1
)

cbar = fig.colorbar(im, ax=ax, fraction=0.046, pad=0.04)
cbar.set_label('normalized dissonance', color='#111111', fontsize=12)
cbar.ax.yaxis.set_tick_params(color='#111111')
plt.setp(cbar.ax.yaxis.get_ticklabels(), color='#111111')

ax.set_xlabel('interval r', color='#111111', fontsize=13)
ax.set_ylabel('interval s', color='#111111', fontsize=13)
ax.set_title('3d dissonance heatmap\n(Vassilakis 2001 Custom (adapted
from Aatish Bhatia))', color='#111111', fontsize=14, pad=14)
ax.tick_params(colors='#111111')
for spine in ax.spines.values():
    spine.set_edgecolor('#cccccc')

ratios2 = {
    'p8 (2:1)': (2.0, 2.0),
    'p5 (3:2)': (1.5, 1.5),
    'p4 (4:3)': (4/3, 4/3),
    'm3 (5:4)': (5/4, 5/4),
    'm3 (6:5)': (6/5, 6/5),
}
for label, (rx, ry) in ratios2.items():
    if x_data[0] <= rx <= x_data[-1] and y_data[0] <= ry <=
y_data[-1]:
        ax.axvline(rx, color='#0066b6', linewidth=0.5, alpha=0.4,
linestyle='--')
        ax.axhline(ry, color='#0066b6', linewidth=0.5, alpha=0.4,
```

```
linestyle='--')

plt.tight_layout()
plt.savefig('dissonance_heatmap.png', dpi=150, bbox_inches='tight',
facecolor=fig.get_facecolor())
plt.savefig('triadic-acoustic/vassilakis2001-custom.png')
plt.show()
```

V: Load and process data

```
import json
with open("intervals.json") as f:
    interval_data = json.load(f)
intervals = interval_data['2']

with open("experiment_data.json") as f:
    experiment_data = json.load(f)
experiment_intervals = experiment_data['2']

# HE datapoints
he_datapoints = harmonic_entropy() # index i = i cents

processed_data = {}
historic_consonant_intervals = {(2, 1), (3, 2), (4, 3), (5, 4), (6,
5), (5, 3), (8, 5)}
variance =
[2.267786838,2.560886474,2.076872596,1.954351705,2.091219846,2.349100
344,2.084324238,2.693456243,1.848048266,1.904997217,1.447493729,2.470
871278,2.118055064,1.529698615,2.192557632,1.954635006,2.132904165,1.
90674039,1.952083817,2.451749212,2.216169313,2.217917573,1.980245742,
2.257508697,2.389993907,2.44451156]
for i, (n, d) in enumerate(intervals): # usually, lower means more
consonant and higher maeans more dissonant
    data_dict = {}
    models = {}
    print(f"Processing index {i} with ratio {n}:{d}..")
    cents = round(1200 * log(n / d, 2))

    models["harmonic_entropy"] = he_datapoints[cents]
```

```
    models["tenney"] = tenney(n, d)
    models["naive"] = weil(n, d)
    models["wilson"] = wilson(n, d)
    models["confrac"] = confrac(n, d)
    models["binary"] = 0 if (n, d) in historic_consonant_intervals
else 1
    data_dict["ratio"] = [n, d]
    data_dict["cents"] = cents
    data_dict["score"] = 1 - experiment_intervals[i]
    data_dict["models"] = models
    data_dict["stdev"] = variance[i]

    processed_data[i] = data_dict
```

VI: Extraction of data from "perceptual" models

```
import dissonant
from scipy.signal import argrelmin

def freq_space(base_f=440, octaves=2, steps=440):
    ratios = np.linspace(1, octaves, steps)
    freqs = base_f * ratios
    return freqs

def dissonance_curve(freq_space, n_partials=10,
model='sethares1993'):
    dissonances = []
    for freq in freq_space:
        h_freqs, h_amps = dissonant.harmonic_tone([freq_space[0],
freq], n_partials=n_partials, profile="inverse")
        d = dissonant.dissonance(h_freqs, h_amps, model=model)
        dissonances.append(d)
    dissonances = np.array(dissonances)
    return dissonances

def plot_dissonance_curve(freqs, dissonances, name):
    minima = argrelmin(dissonances)[0]

    plt.figure(figsize=(16, 3))
    plt.xlim(freqs[0], freqs[-1])
```

```
    plt.plot(freqs, dissonances)
    for m in minima:
        plt.axvline(freqs[m], c='r', ls=':')
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Dissonance')
    plt.title(name)


freqs = freq_space()
for model in ("sethares1993", "vassilakis2001", "cook2002",
"cook2006", "cook2009"):
    curve = dissonance_curve(freqs, n_partials=10, model=model)
    plot_dissonance_curve(freqs, curve, model)
    for i, (n, d) in enumerate(intervals):
        freq = round(cents_to_frequency(440, ratio_to_cents(n, d)))
        dis = curve[freq - 440 - 1]
        # print(f"Ratio {n}:{d} has dissonance value of {curve[freq -
440 - 1]}")
        plt.scatter(freq, dis, marker="o")
        processed_data[i]["models"][model] = dis

with open("processed_data.json", "w") as w:
    json.dump(processed_data, w, indent=4)



with open("processed_data.json") as f:
    data = json.load(f)
plt.figure()
for interval in data.values():
    plt.scatter(interval["cents"], interval["score"])
plt.show()


import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from scipy.interpolate import CubicSpline
```

```
centsu = []
scoresu = []
ratiosu = []

for interval in data.values():
    centsu.append(interval["cents"])
    scoresu.append(interval["score"])
    ratiosu.append(f"{interval['ratio'][0]}:{interval['ratio'][1]}")
#
------------------------------------------------------------------------
------

INTERVAL_NAMES = {
    0: "P1", 100: "m2", 200: "M2", 300: "m3", 400: "M3",
    500: "P4", 600: "TT", 700: "P5", 800: "m6",
    900: "M6", 1000: "m7", 1100: "M7", 1200: "P8",
}

cents     = np.array(centsu,   dtype=float)
scores    = np.array(scoresu,  dtype=float)
variances = np.array(variance, dtype=float)
std_devs  = np.array(variances) / 10

order  = np.argsort(cents)
cents, scores, std_devs = cents[order], scores[order],
std_devs[order]

# Cubic spline through the means
spl    = CubicSpline(cents, scores)
x_fine = np.linspace(cents[0], cents[-1], 800)
y_fine = spl(x_fine)

# Interpolate std at fine grid for the band
std_fine = np.interp(x_fine, cents, std_devs)

# ── Plot
──────────────────────────────────────────────────────────────────────
──────────────────

bg, fg = "#ffffff", "#111111"
```

```
fig, ax = plt.subplots(figsize=(12, 6), facecolor=bg)
ax.set_facecolor(bg)

# Spline band
ax.fill_between(x_fine, y_fine - std_fine, y_fine + std_fine,
                color="#0077b6", alpha=0.10, zorder=1)
# Spline line
ax.plot(x_fine, y_fine, color="#0077b6", linewidth=2.0,
        zorder=2, label="Cubic spline")

# Variance bars
ax.errorbar(cents, scores, yerr=std_devs,
            fmt="none", ecolor="#e76f51", elinewidth=1.2,
            capsize=4, capthick=1.2, alpha=0.8,
            zorder=3, label="±1 SD")

# Scatter dots
ax.scatter(cents, scores, color="#e76f51", s=60, zorder=4,
           edgecolors=fg, linewidths=0.4, label="Population mean")

# Interval labels
for c, s, r in zip(centsu, scoresu, ratiosu):
    label = INTERVAL_NAMES.get(int(round(c)), f"{c:.0f}¢\n({r})")
    ax.annotate(label, xy=(c, s), xytext=(0, 70),
                textcoords="offset points",
                ha="center", va="bottom", fontsize=8,
                color=fg, alpha=0.95)

ax.set_xlabel("Interval (cents)", color=fg, fontsize=13)
ax.set_ylabel("Dissonance score", color=fg, fontsize=13)
ax.set_title("Population interval ratings\n(with cubic spline &
variance)",
             color=fg, fontsize=14, pad=14)

ax.set_xlim(cents[0] - 40, cents[-1] + 40)
ax.set_ylim(-0.05, 1.15)
ax.xaxis.set_major_locator(ticker.MultipleLocator(100))
ax.xaxis.set_minor_locator(ticker.MultipleLocator(50))
```

```
ax.yaxis.set_major_locator(ticker.MultipleLocator(0.2))
ax.tick_params(colors=fg, which="both")
for spine in ax.spines.values():
    spine.set_edgecolor("#cccccc")
ax.grid(True, which="major", color="#dddddd", linewidth=0.6,
alpha=0.7)
ax.grid(True, which="minor", color="#eeeeee", linewidth=0.3,
alpha=0.4)

ax.legend(loc="upper right", fontsize=10,
          facecolor="#f5f5f5", edgecolor="#cccccc", labelcolor=fg)

plt.tight_layout()
plt.show()


from scipy.stats import linregress
from scipy.optimize import curve_fit

def linear_model(x, m, c):
    return m * x + c

processed_models = {}
scores = []
for model_name in data["0"]["models"]:
    processed_models[model_name] = []

for interval in data.values():
    for name, value in interval["models"].items():
        processed_models[name].append(value)
    scores.append(interval["score"])

for name, model_data in processed_models.items():
    fig, ax = plt.subplots()
    if name == "harmonic_entropy":
        processed_model_data = model_data[1:]
        processed_scores = scores[1:]
    else:
        processed_model_data = model_data
```

```
        processed_scores = scores

    ax.scatter(processed_model_data, processed_scores)
    slope, intercept, r_value, p_value, std_err =
linregress(processed_model_data, processed_scores)
    ax.plot(processed_model_data, intercept + slope *
np.array(processed_model_data), 'r', label=f"Line of best fit ($R^2 =
{r_value**2:.3f}$)")
    ax.set_title(name.replace("_", " ").title())
    ax.set_xlabel(name, fontsize=13)
    ax.set_ylabel("Participant scores", fontsize=13)
    ax.legend()
    plt.show()
    plt.savefig(f"dyadic-r2/{name}.png")



import json
with open("intervals.json") as f:
    interval_data = json.load(f)
intervals = interval_data['3']

with open("experiment_data.json") as f:
    experiment_data = json.load(f)
experiment_intervals = experiment_data['3']

processed_data = {}
historic_consonant_intervals = {(2, 3, 4), (4, 5, 6), (10, 12, 15)}
edo_indexes = {1, 3}
edo_cents = ["yo", (400, 700), "gurt", (300, 700)]
for i, (f1, f2, f3) in enumerate(intervals): # usually, lower means
more consonant and higher maeans more dissonant
    data_dict = {}
    models = {}
    print(f"Processing index {i} with ratio {f1}:{f2}:{f3}..")
    models["vassilakis2001-custom"] = lookup_dissonance(f2/f1, f3/f1,
x_data, y_data, z_data)

    if i not in edo_indexes:
```

```
        models["tenney"] = tenney(f1, f2, f3)
        models["naive"] = weil(f1, f2, f3)
        models["wilson"] = wilson(f1, f2, f3)
        models["confrac"] = confrac_triad(f1, f2, f3)
        cents1 = round(1200 * log(f2 / f1, 2))
        cents2 = round(1200 * log(f3 / f1, 2))
    else:
        models["tenney"] = -1
        models["naive"] = -1
        models["wilson"] = -1
        models["confrac"] = -1
        cents1, cents2 = edo_cents[i]
    models["binary"] = 0 if (f1, f2, f3) in
historic_consonant_intervals else 1
    data_dict["ratio"] = [f1, f2, f3]
    data_dict["cents"] = [cents1, cents2]
    data_dict["score"] = 1 - experiment_intervals[i]
    data_dict["models"] = models

    processed_data[i] = data_dict


edo_indexes = {1, 3}
edo_values = ["gurt", (0, 4, 7), "yo", (0, 3, 7)]
for model in ("sethares1993", "vassilakis2001", "cook2002",
"cook2006", "cook2009"):
    for i, (f1, f2, f3) in enumerate(intervals):
        if i not in edo_indexes:
            freq1 = cents_to_frequency(440, ratio_to_cents(f2, f1))
            freq2 = cents_to_frequency(440, ratio_to_cents(f3, f1))
            print([0, freq1, freq2])
            freq, amps = dissonant.harmonic_tone([440, freq1, freq2],
n_partials=10, profile="inverse")
        else:
            print(dissonant.pitch_to_freq(edo_values[i]))
            freq, amps =
dissonant.harmonic_tone(dissonant.pitch_to_freq(edo_values[i]),
n_partials=10, profile="inverse")
        dis = dissonant.dissonance(freq, amps, model=model)
```

```
        processed_data[i]["models"][model] = dis

with open("processed_data_triads.json", "w") as w:
    json.dump(processed_data, w, indent=4)



import numpy as np
import matplotlib.pyplot as plt
import argparse
import os
import time

from dissonant import harmonic_tone, dissonance, pitch_to_freq

#
------------------------------------------------------------------------
------
# Parameters
#
------------------------------------------------------------------------
------
MODELS = ["sethares1993", "vassilakis2001", "cook2002", "cook2006",
"cook2009"]

# C4 = MIDI 60, reference frequency ≈ 261.63 Hz
C4_MIDI  = 60
N_PARTIALS = 10          # harmonics per tone
INTERVAL_MIN = 1.0       # unison
INTERVAL_MAX = 2.0       # octave
DEFAULT_RES  = 150       # grid steps per axis

# JI guide lines (interval ratio value, label)
JI_GUIDES = {
    "P8 (2:1)":  2.0,
    "P5 (3:2)":  3 / 2,
    "P4 (4:3)":  4 / 3,
    "M3 (5:4)":  5 / 4,
    "m3 (6:5)":  6 / 5,
```

```
}

#
------------------------------------------------------------------------
------
# Grid computation
#
------------------------------------------------------------------------
------

def compute_heatmap(model: str, res: int, base_midi: int = C4_MIDI,
                    n_partials: int = N_PARTIALS,
                    r_min: float = INTERVAL_MIN, r_max: float =
INTERVAL_MAX):

    base_freq = pitch_to_freq(base_midi)
    r_vals    = np.linspace(r_min, r_max, res)
    s_vals    = np.linspace(r_min, r_max, res)

    Z = np.zeros((res, res))

    # Pre-compute amplitudes for a single harmonic tone (all three
tones
    # share the same spectral profile; only the frequency differs)
    _, amps_template = harmonic_tone([base_freq],
n_partials=n_partials, profile="inverse")
    # amps_template is shape (n_partials,) with values [1, 0.5, 0.33,
...]

    t0 = time.time()
    print(f"  Computing {res}×{res} grid for model '{model}' ...")

    for i, r in enumerate(r_vals):
        for j, s in enumerate(s_vals):
            # Build the combined spectrum of three harmonic tones
            freqs_trio, amps_trio = harmonic_tone(
                [base_freq, base_freq * r, base_freq * s],
                n_partials=n_partials,
            )
```

```
            Z[i, j] = dissonance(freqs_trio, amps_trio, model=model)

        if i % max(1, res // 10) == 0:
            pct = (i + 1) / res * 100
            print(f"    {pct:5.1f}%  ({time.time()-t0:.1f}s)",
end="\r", flush=True)

    print(f"\n  Done in {time.time()-t0:.1f}s   raw range: "
          f"{Z.min():.4f} ... {Z.max():.4f}")
    return r_vals, s_vals, Z


#
----------------------------------------------------------------------
------
# Plotting
#
----------------------------------------------------------------------
------

MODEL_DESCRIPTIONS = {
    "sethares1993":  "Sethares 1993  (roughness, product SPL)",
    "vassilakis2001": "Vassilakis 2001  (roughness, SPL +
AF-degree)",
    "cook2002":      "Cook 2002  (semitone-based roughness)",
    "cook2006":      "Cook 2006  (tension + modality)",
    "cook2009":      "Cook 2009  (extended tension model)",
}


def plot_heatmap(r_vals, s_vals, Z, model: str,
                 output_path: str = None, show: bool = False):
    """Render a single heatmap in the reference style."""

    z_min, z_max = Z.min(), Z.max()
    Z_norm = (Z - z_min) / (z_max - z_min) if z_max > z_min else
np.zeros_like(Z)

    bg, fg = "#ffffff", "#111111"
```

```
    fig, ax = plt.subplots(figsize=(10, 9), facecolor=bg)
    ax.set_facecolor(bg)

    im = ax.imshow(
        Z_norm.T,
        origin="lower",
        extent=[r_vals[0], r_vals[-1], s_vals[0], s_vals[-1]],
        aspect="equal",
        cmap="inferno",
        interpolation="bilinear",
        vmin=0, vmax=1,
    )

    cbar = fig.colorbar(im, ax=ax, fraction=0.046, pad=0.04)
    cbar.set_label("normalized dissonance", color=fg, fontsize=12)
    cbar.ax.yaxis.set_tick_params(color=fg)
    plt.setp(cbar.ax.yaxis.get_ticklabels(), color=fg)

    ax.set_xlabel("interval r", color=fg, fontsize=13)
    ax.set_ylabel("interval s", color=fg, fontsize=13)
    ax.set_title(
        f"3D dissonance heatmap\n({MODEL_DESCRIPTIONS.get(model,
model)})",
        color=fg, fontsize=14, pad=14,
    )
    ax.tick_params(colors=fg)
    for spine in ax.spines.values():
        spine.set_edgecolor("#cccccc")

    # JI guide lines
    for label, ratio in JI_GUIDES.items():
        if r_vals[0] <= ratio <= r_vals[-1]:
            ax.axvline(ratio, color="#0077b6", linewidth=0.5,
alpha=0.4, linestyle="--")
            ax.axhline(ratio, color="#0077b6", linewidth=0.5,
alpha=0.4, linestyle="--")

    plt.tight_layout()
```

```
    if output_path:
        fig.savefig(output_path, dpi=150, bbox_inches="tight",
                    facecolor=fig.get_facecolor())
        print(f"  Saved → {output_path}")

    if show:
        plt.show()
        plt.savefig(f"triadic-acoustic/{model}.png")
    else:
        plt.close(fig)

#
------------------------------------------------------------------------
------
# Entry point
#
------------------------------------------------------------------------
------

def main():
    models_to_run = MODELS


    for model in models_to_run:
        print(f"\n{'='*55}")
        print(f"  Model: {model}")
        print(f"{'='*55}")

        r_vals, s_vals, Z = compute_heatmap(
            model, DEFAULT_RES, n_partials=N_PARTIALS
        )

        plot_heatmap(r_vals, s_vals, Z, model,
                     show=True)

    print("\nAll done.")
```

```
if __name__ == "__main__":
    main()



"""
Dissonance scatter plot from triadic interval data (~7 data points).
"""

import json
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import Normalize

with open("processed_data_triads.json") as f:
    data = json.load(f)
scores = []
intervals = []
for interval in data.values():
    intervals.append(interval["cents"])
    scores.append(interval["score"])


def plot_dissonance_scatter(
    intervals: list,
    scores: list,
    cmap: str = "inferno",
    title: str = "Triadic dissonance",
    subtitle: str = "(human ratings)",
    xlabel: str = "interval r  (cents)",
    ylabel: str = "interval s  (cents)",
):
    data = np.array(intervals, dtype=float)
    vals = np.array(scores,    dtype=float)

    if data.ndim != 2 or data.shape[1] != 2:
        raise ValueError("Each element of `intervals` must be a
2-element list.")
    if len(vals) != len(data):
```

```
        raise ValueError("`intervals` and `scores` must have the same
length.")

    v_min, v_max = vals.min(), vals.max()
    vals_norm = (vals - v_min) / (v_max - v_min) if v_max > v_min
else np.zeros_like(vals)

    r_vals = data[:, 0]
    s_vals = data[:, 1]

    r_span = r_vals.max() - r_vals.min() or 200
    s_span = s_vals.max() - s_vals.min() or 200
    r_pad  = r_span * 0.20
    s_pad  = s_span * 0.20
    xlim   = (r_vals.min() - r_pad, r_vals.max() + r_pad)
    ylim   = (s_vals.min() - s_pad, s_vals.max() + s_pad)
    xlim = (0, 1200 + 200)
    ylim = (0, 1200 + 200)

    fg = "#111111"

    fig, ax = plt.subplots(figsize=(8, 7), facecolor="white")
    ax.set_facecolor("white")
    ax.set_xlim(*xlim)
    ax.set_ylim(*ylim)

    fig.canvas.draw()
    ax_width_pts   = ax.get_window_extent().width
    ax_height_pts  = ax.get_window_extent().height
    pts_per_unit_x = ax_width_pts  / (xlim[1] - xlim[0])
    pts_per_unit_y = ax_height_pts / (ylim[1] - ylim[0])
    radius_units   = min(r_span, s_span) * 0.09
    radius_pts     = min(radius_units * pts_per_unit_x,
                         radius_units * pts_per_unit_y)
    marker_size    = np.pi * radius_pts ** 2

    norm = Normalize(vmin=0, vmax=1)
    sc = ax.scatter(
        r_vals, s_vals,
```

```
        c=vals_norm,
        s=marker_size,
        cmap=cmap,
        norm=norm,
        edgecolors=fg,
        linewidths=0.8,
        zorder=3,
        alpha=0.92,
    )

    for rx, sx, v in zip(r_vals, s_vals, vals):
        ax.text(rx, sx, f"{v:.2f}", ha="center", va="center",
                color="#ccc", fontsize=9, fontweight="bold",
zorder=4)

    cbar = fig.colorbar(sc, ax=ax, fraction=0.046, pad=0.02)
    cbar.set_label("normalized dissonance", color=fg, fontsize=11,
labelpad=12,
                   rotation=270, va="bottom")
    cbar.ax.yaxis.set_tick_params(color=fg)
    plt.setp(cbar.ax.yaxis.get_ticklabels(), color=fg)
    cbar.outline.set_edgecolor("#cccccc")

    n_lines = 4
    for rv in np.linspace(*xlim, n_lines + 2)[1:-1]:
        ax.axvline(rv, color="#0077b6", linewidth=0.6, alpha=0.35)
    for sv in np.linspace(*ylim, n_lines + 2)[1:-1]:
        ax.axhline(sv, color="#0077b6", linewidth=0.6, alpha=0.35)

    ax.set_xlabel(xlabel, color=fg, fontsize=12, labelpad=8)
    ax.set_ylabel(ylabel, color=fg, fontsize=12, labelpad=8)
    ax.tick_params(colors=fg)
    for spine in ax.spines.values():
        spine.set_edgecolor("#cccccc")
    ax.grid(False)

    full_title = f"{title}\n{subtitle}" if subtitle else title
    ax.set_title(full_title, color=fg, fontsize=13, pad=12)
```

```
    plt.tight_layout()
    plt.show()


plot_dissonance_scatter(
    intervals=intervals,
    scores=scores,
    title="Population triad ratings",
    subtitle="(tension + modality)",
    xlabel="interval r  (cents)",
    ylabel="interval s  (cents)",
)



from scipy.stats import linregress
from scipy.optimize import curve_fit

def linear_model(x, m, c):
    return m * x + c

with open("processed_data_triads.json") as f:
    data_triads = json.load(f)

processed_models = {}
scores = []
for model_name in data_triads["0"]["models"]:
    processed_models[model_name] = []

for interval in data_triads.values():
    for name, value in interval["models"].items():
        processed_models[name].append(value)
    scores.append(interval["score"])

for name, model_data in processed_models.items():
    fig, ax = plt.subplots()
    processed_model_data = []
    processed_scores = []
    for i, data in enumerate(model_data):
```

```
        if data != -1:
            processed_scores.append(scores[i])
            processed_model_data.append(data)

    print(name, processed_model_data)
    ax.scatter(processed_model_data, processed_scores)
    slope, intercept, r_value, p_value, std_err =
linregress(processed_model_data, processed_scores)
    ax.plot(processed_model_data, intercept + slope *
np.array(processed_model_data), 'r', label=f"Line of best fit ($R^2 =
{r_value**2:.3f}$)")
    ax.set_title(name.replace("_", " ").title())
    ax.set_xlabel(name, fontsize=13)
    ax.set_ylabel("Participant scores", fontsize=13)
    ax.legend()
    plt.show()
    plt.savefig(f"triadic-r2/{name}.png")



import seaborn as sns

sns.set_theme(font_scale = 1.2)
plt.figure(figsize=(8,8))

plt.pie(
    x=[29, 13], labels=["Experienced", "No experience"],
    colors=sns.color_palette('Set2'),
    startangle=90,
    # show percentages
    autopct='%1.2f%%',
    # move the percentage inside the arcs
    pctdistance=0.80,
    # add space between the arcs
    explode=[0.05, 0.05]
)

### Add a hole in the pie
hole = plt.Circle((0, 0), 0.65, facecolor='white')
```

```
plt.gcf().gca().add_artist(hole)
plt.title(label="Musical experience within the population")

plt.show()

sns.set_theme(font_scale = 0.8)
plt.figure()
plt.pie(
    x=[1, 7, 13, 18, 4], labels=['Very poor', 'Below average',
'Average', 'Above average', 'Excellent'],
    colors=sns.color_palette('Set2'),
    startangle=90,
    # show percentages
    autopct='%1.2f%%',
    # move the percentage inside the arcs
    pctdistance=0.80,
    # add space between the arcs
    explode=[0.05, 0.05, 0.05, 0.05, 0.05]
)

### Add a hole in the pie
hole = plt.Circle((0, 0), 0.65, facecolor='white')
plt.gcf().gca().add_artist(hole)
plt.title(label="Pitch distinguishability within the population")

plt.show()

categories = ['Very poor', 'Below average', 'Average', 'Above
average', 'Excellent']
values = [1, 7, 13, 18, 4]
plt.figure()
plt.bar(categories, values)
plt.title('Basic Bar Chart')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from dissonant import dissonance

#
--------------------------------------------------------------------
------
# Spectrum design
#
--------------------------------------------------------------------
------

def inharmonic_spectrum(f0: float, n_partials: int, stretch: float,
                        amp_fn=None):
    """
    Build a stretched-partial spectrum for a single tone.

    partial_k = f0 * k^(1 + stretch)   for k = 1, 2, ..., n_partials

    Parameters
    ----------
    f0         : fundamental frequency in Hz
    n_partials : number of partials
    stretch    : stretching exponent B  (0 = harmonic, >0 =
stretched)
    amp_fn     : callable(k) → amplitude for partial k; default is
1/k

    Returns
    -------
    freqs : array of partial frequencies
    amps  : array of partial amplitudes (normalised to max=1)
    """
    if amp_fn is None:
        amp_fn = lambda k: 1.0 / k

    ks    = np.arange(1, n_partials + 1, dtype=float)
    freqs = f0 * ks ** (1.0 + stretch)
```

```
    amps  = np.array([amp_fn(k) for k in ks])
    amps  = amps / amps.max()
    return freqs, amps


def two_tone_spectrum(f0: float, interval_ratio: float,
                      n_partials: int, stretch: float, amp_fn=None):
    """
    Combined spectrum of two tones a ratio apart, both using the same
    inharmonic partial structure.
    """
    f1, a1 = inharmonic_spectrum(f0, n_partials, stretch, amp_fn)
    f2, a2 = inharmonic_spectrum(f0 * interval_ratio, n_partials,
stretch, amp_fn)
    freqs  = np.concatenate([f1, f2])
    amps   = np.concatenate([a1, a2])
    return freqs, amps


# Bell-curve peaking around partial 3-4: maximises beating at the
octave
# because it concentrates energy where the stretched 2nd partial of
the
# root lands close to the 1st partial of the octave note.
def bell_amp(k, peak=3.5, width=2.0):
    return np.exp(-((k - peak) ** 2) / (2 * width ** 2))

# Plain 1/k falloff (familiar harmonic weighting)
def harmonic_amp(k):
    return 1.0 / k

# Flat spectrum -- all partials equal amplitude -- maximises total
roughness
def flat_amp(k):
    return 1.0


# Parameters -- edit these to explore
F0         = 261.63          # C4 in Hz
```

```
N_PARTIALS = 8               # partials per tone
STRETCH    = 0.42            # key parameter: B ≈ 0.4-0.6 breaks
octave consonance
AMP_FN     = lambda k: bell_amp(k, peak=3.5, width=2.0)
MODEL      = "sethares1993"  # or vassilakis2001 / cook2002 /

CENTS_MIN  = 0
CENTS_MAX  = 1500            # go past the octave to see the valley
disappear
N_STEPS    = 600


#
----------------------------------------------------------------------
------
# Print the partial structure so you can hear/verify it
#
----------------------------------------------------------------------
------
print(f"Base frequency: {F0:.2f} Hz")
print(f"Stretch exponent B = {STRETCH}")
print(f"\nPartials of the ROOT note ({F0:.1f} Hz):")
freqs_root, amps_root = inharmonic_spectrum(F0, N_PARTIALS, STRETCH,
AMP_FN)
for k, (f, a) in enumerate(zip(freqs_root, amps_root), 1):
    ratio = f / F0
    print(f"  k={k}: {f:8.2f} Hz  (ratio {ratio:.4f})  amp={a:.3f}")

octave_f0 = F0 * 2.0
print(f"\nPartials of the OCTAVE note ({octave_f0:.1f} Hz):")
freqs_oct, amps_oct = inharmonic_spectrum(octave_f0, N_PARTIALS,
STRETCH, AMP_FN)
for k, (f, a) in enumerate(zip(freqs_oct, amps_oct), 1):
    ratio = f / F0
    print(f"  k={k}: {f:8.2f} Hz  (ratio {ratio:.4f})  amp={a:.3f}")

print(f"\nNote: with B=0 (harmonic), root partial k=2 would land at "
      f"{F0*2:.1f} Hz = octave partial k=1 → consonant.")
print(f"With B={STRETCH}, root partial k=2 lands at "
```

```
      f"{freqs_root[1]:.2f} Hz ≠ {octave_f0:.2f} Hz → roughness at
the octave.")


#
------------------------------------------------------------------
------
# Compute dissonance curve
#
------------------------------------------------------------------
------
print(f"\nComputing dissonance curve ({N_STEPS} steps,
model={MODEL})...")
cents_arr = np.linspace(CENTS_MIN, CENTS_MAX, N_STEPS)
dissonance_arr = np.zeros(N_STEPS)

for i, cents in enumerate(cents_arr):
    ratio = 2 ** (cents / 1200.0)
    freqs, amps = two_tone_spectrum(F0, ratio, N_PARTIALS, STRETCH,
AMP_FN)
    dissonance_arr[i] = dissonance(freqs, amps, model=MODEL)

# Normalise to [0, 1]
d_min, d_max = dissonance_arr.min(), dissonance_arr.max()
d_norm = (dissonance_arr - d_min) / (d_max - d_min)

# Reference: same curve with standard harmonic partials (B=0)
print("Computing reference harmonic curve (B=0)...")
d_harmonic = np.zeros(N_STEPS)
for i, cents in enumerate(cents_arr):
    ratio = 2 ** (cents / 1200.0)
    freqs, amps = two_tone_spectrum(F0, ratio, N_PARTIALS,
stretch=0.0,
                                    amp_fn=harmonic_amp)
    d_harmonic[i] = dissonance(freqs, amps, model=MODEL)
d_harm_norm = (d_harmonic - d_harmonic.min()) / (d_harmonic.max() -
d_harmonic.min())
```

```
#
------------------------------------------------------------------------
------
# Plot
#
------------------------------------------------------------------------
------
bg, fg = "#ffffff", "#111111"

fig, axes = plt.subplots(2, 1, figsize=(13, 9), facecolor=bg,
                         gridspec_kw={"hspace": 0.45})

# ── Common interval markers
──────────────────────────────────────────────────────────

JI = {
    "P1\n(0¢)":      0,
    "m2\n(100¢)":   100,
    "M2\n(200¢)":   200,
    "m3\n(300¢)":   300,
    "M3\n(400¢)":   400,
    "P4\n(500¢)":   500,
    "TT\n(600¢)":   600,
    "P5\n(700¢)":   700,
    "m6\n(800¢)":   800,
    "M6\n(900¢)":   900,
    "m7\n(1000¢)": 1000,
    "M7\n(1100¢)": 1100,
    "P8\n(1200¢)": 1200,
    "m9\n(1300¢)": 1300,
}

for ax in axes:
    ax.set_facecolor(bg)
    ax.tick_params(colors=fg)
    for sp in ax.spines.values():
        sp.set_edgecolor("#cccccc")
    ax.grid(True, which="major", color="#e0e0e0", linewidth=0.6)
    ax.grid(True, which="minor", color="#f0f0f0", linewidth=0.3)
    ax.xaxis.set_major_locator(ticker.MultipleLocator(100))
```

```
    ax.xaxis.set_minor_locator(ticker.MultipleLocator(50))
    ax.set_xlim(CENTS_MIN, CENTS_MAX)
    ax.set_ylim(-0.05, 1.12)
    for c in JI.values():
        if CENTS_MIN <= c <= CENTS_MAX:
            ax.axvline(c, color="#aaaaaa", linewidth=0.4,
linestyle="--", alpha=0.6)
    # Highlight the octave
    ax.axvline(1200, color="#d62728", linewidth=1.0, linestyle=":",
alpha=0.8)

# ── Top panel: reference harmonic spectrum
─────────────────────────────────────────────

ax0 = axes[0]
ax0.plot(cents_arr, d_harm_norm, color="#0077b6", linewidth=1.8,
label="Harmonic (B=0)")
ax0.fill_between(cents_arr, d_harm_norm, alpha=0.08, color="#0077b6")
ax0.set_title(f"Reference: standard harmonic spectrum  (B = 0)",
              color=fg, fontsize=13, pad=10)
ax0.set_ylabel("Normalised dissonance", color=fg, fontsize=11)
ax0.set_xlabel("Interval (cents)", color=fg, fontsize=11)

# Label octave on top panel
octave_d = d_harm_norm[np.argmin(np.abs(cents_arr - 1200))]
ax0.annotate("P8 (octave)\n-- consonant valley", xy=(1200, octave_d),
             xytext=(1220, octave_d + 0.18),
             fontsize=8, color="#d62728",
             arrowprops=dict(arrowstyle="->", color="#d62728",
lw=0.8))

# ── Bottom panel: inharmonic (octave-dissonant) spectrum
──────────────────────────

ax1 = axes[1]
ax1.plot(cents_arr, d_norm, color="#e76f51", linewidth=1.8,
         label=f"Inharmonic (B = {STRETCH})")
ax1.fill_between(cents_arr, d_norm, alpha=0.10, color="#e76f51")
ax1.set_title(
    f"Octave-dissonant spectrum  (stretch B = {STRETCH}, "
    f"{N_PARTIALS} partials, bell amplitude weighting)",
```

```
    color=fg, fontsize=13, pad=10)
ax1.set_ylabel("Normalised dissonance", color=fg, fontsize=11)
ax1.set_xlabel("Interval (cents)", color=fg, fontsize=11)

# Label octave on bottom panel
octave_d_inh = d_norm[np.argmin(np.abs(cents_arr - 1200))]
ax1.annotate(f"P8 (octave)\n-- dissonance = {octave_d_inh:.2f}",
xy=(1200, octave_d_inh),
             xytext=(1220, octave_d_inh + 0.15),
             fontsize=8, color="#d62728",
             arrowprops=dict(arrowstyle="->", color="#d62728",
lw=0.8))

# Mark the new consonance valleys (local minima in inharmonic curve)
from scipy.signal import argrelmin
local_min_idx = argrelmin(d_norm, order=8)[0]
for idx in local_min_idx:
    c, d = cents_arr[idx], d_norm[idx]
    if d < 0.35 and CENTS_MIN + 50 < c < CENTS_MAX - 50:
        ax1.plot(c, d, "o", color="#2a9d8f", ms=5, zorder=5)
        ax1.annotate(f"{c:.0f}¢", xy=(c, d), xytext=(0, -14),
                     textcoords="offset points", ha="center",
                     fontsize=7, color="#2a9d8f")

# ── Partial structure inset (bottom panel)
──────────────────────────────────────────────

inset = ax1.inset_axes([0.01, 0.55, 0.28, 0.40])
inset.set_facecolor("#f8f8f8")
inset.set_title("Partial structure", fontsize=7, color=fg, pad=3)
k_arr = np.arange(1, N_PARTIALS + 1)
harm_freqs = F0 * k_arr                                  # harmonic
inh_freqs  = freqs_root                                  # inharmonic
(already computed)
inh_amps   = amps_root
harm_amps  = np.array([harmonic_amp(k) for k in k_arr])
harm_amps /= harm_amps.max()

inset.bar(inh_freqs / F0, inh_amps, width=0.04, color="#e76f51",
          alpha=0.85, label=f"B={STRETCH}")
```

```
inset.bar(k_arr.astype(float), harm_amps, width=0.04,
color="#0077b6",
          alpha=0.5, label="B=0")
inset.set_xlabel("Partial (× f₀)", fontsize=6, color=fg)
inset.set_ylabel("Amp", fontsize=6, color=fg)
inset.tick_params(labelsize=6, colors=fg)
for sp in inset.spines.values():
    sp.set_edgecolor("#cccccc")
inset.legend(fontsize=5.5, loc="upper right", framealpha=0.7)

fig.suptitle(
    f"Octave dissonance via inharmonic partials  [{MODEL}]",
    color=fg, fontsize=15, y=0.98, fontweight="500"
)

plt.show()
```

Appendix B: Sample Guiding Interview Questions

1. What is your favorite musical genre that you frequently listen to, and how do you think dissonance plays a role in that genre?
2. Of the standard musical instruments, which instrument sounds the most dissonant, and which sounds the most consonant?
3. [If the interviewee is a musician] How do you tune your instrument relative to a base pitch?
4. Have you heard of microtonality, and if so, what is your consensus on it?
5. Compared to classical music, do you think jazz is more dissonant or consonant?
6. Is dissonance pleasant or unpleasant to you?

Appendix C: Informed Consent Form and Media Release Form

INFORMED CONSENT FORM: This study is anonymous and entirely voluntary, and I will not be collecting any information about your identity. Findings from the study will not include any information that would make it possible to identify you, and data will be kept entirely confidential to the best of my ability. All records will only be used for educational purposes. Your individual answers to this survey are accessible only to me.

Please note that the survey is being conducted with the help of Aidaform, a company not affiliated with me and with its own privacy and security policies that you can find at its website.

Participation in this survey presents no greater risk than everyday use of the Internet.

All parts mentioned in this survey are voluntary, and participants may refuse to participate before the study begins, discontinue at any time, or skip any unrequired questions that may make them feel uncomfortable with no penalty. This survey, again, should take less than 5 minutes to complete top-to-bottom.

QUESTIONS AND CONCERNS: You have the right to ask questions about this study and to have those questions answered before or after the research. If you have any further questions, please feel free to contact me via email at **[REDACTED]**

Checking the box below indicates that you agree to participate in this study and that you understand the information provided above.

[X] I acknowledge the Informed Consent Form outlined above and wish for my response to be collected anonymously.

MEDIA RELEASE FORM: I, _____, do hereby grant **[REDACTED]**, in behalf of **[REDACTED]**, my permission to record my voice and interview for use in the anonymous research study. I acknowledge that the audio recording and transcription that follows will not be shared to other parties and will be kept as private as possible.

Furthermore, I understand that after the study, any audio recordings will be disposed of properly to maintain maximal anonymity.

Signature: _____ Date: _____

Appendix D: Raw Data of Population Triad C/D Perception Ratings

Triad	**Population Triad Ratings**
4:5:6	0.02315
4:5:6 12-TET Equivalent (0-400c-700c)	0.05359
10:12:15	0.20950
10:12:15 12-TET Equivalent (0-300c-700c)	0.22655
160:165:176	0.69428
2:3:4	0.00000
45:55:72	0.34592