

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Systematic Assessment of Musical Audio Rhythmic Compatibility

Cláudio Fischer Lemos

DISSERTAÇÃO

U. PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Gillberto Bernardes de Almeida

July 14, 2021

A Systematic Assessment of Musical Audio Rhythmic Compatibility

Cláudio Fischer Lemos

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. Daniel Mendes

External Examiner: Prof. Maria Navarro Cáceres

Supervisor: Prof. Gilberto Bernardes de Almeida

July 14, 2021

Abstract

Throughout the years, technological developments have been hand in hand with innovations in the music field, helping artists tap into their creativity with new and modern tools. Today, a single computer can be used as the only instrument in a professional musician's tool rack.

In the early 2000s, Web 2.0 was an essential factor for the widespread of content online, which quickly helped the emergence of public databases, where music producers and artists can search for and download audio samples. These audio samples, commonly known as loops, are short segments of audio, around 15 to 30 seconds, and are repeated to create a rhythmic sequence. Artists are no longer tied to the tedious process of recording live instruments, and can now browse through these databases to find the perfect loop. Exploring and listening through these extensive musical archives can be a very time-consuming task, especially if the user wishes to search for a specific audio sample that is similar or will complement and be rhythmically compatible with a pre-selected one.

Developments in the field of Music Information Retrieval (MIR) have helped tackle this challenge, with the assist of computational methods that can decode an audio signal into meaningful audio descriptors that are essential for the indexation and navigation of these large databases, as they provide contextual information about the signal through different levels of abstraction.

By looking at musical structures as mathematical objects, this research project aims to retrieve metrics for the similarity of musical rhythms from musical audio signals, advancing a novel prototype for rhythmic compatibility that will help analyse and quantify the retrieved information and suggest a way to navigate and create expressive musical results.

The main goal is to create a visualization tool for rhythmic compatibility from musical audio in a 2-dimensional topology space, which in the future, will ultimately lead to the development of a music performance application for the recombination and retrieval of compatible musical rhythms from audio samples.

In this dissertation, we analyse a multi-track database composed and performed by professional musicians and artists. The analysis is computed using methods for the rhythmic analysis of audio content, Rhythmic Patterns (RP), Rhythmic Histograms (RH) and Beat Spectrum (BS). RH is better at discriminating rhythmic patterns and for that reason, it was the method used in the prototype. The developed prototype uses Uniform Manifold Approximation and Projection (UMAP) to reduce the dimensions of the data generated in order to visualize audio loop databases.

Keywords: Musical audio signal, Rhythmic compatibility, Sound computing, Music Information Retrieval, Music Analysis

Resumo

Ao longo dos anos, a tecnologia tem vindo a evoluir a par e passo com as inovações no ramo da música, o que ajuda os artistas a explorar a sua criatividade com ferramentas novas e modernas. Atualmente, um computador pode ser o único instrumento na bagagem de um músico profissional.

No início dos anos 2000, a Web 2.0 ajudou à disseminação de conteúdo online, o que ajudou ao aparecimento de bases de dados públicas, onde produtores musicais e artistas podem encontrar e descarregar *loops* de áudio. Estes *loops* de áudio são curtos segmentos de áudio entre 15 a 30 segundos, e são repetidas de maneira a formar uma sequência rítmica. Os artistas deixam assim de estar presos à necessidade de ter que gravar os instrumentos ao vivo, podendo explorar estas bases de dados para encontrar o *loop* perfeito. Explorar e ouvir estes arquivos musicais pode-se tornar numa tarefa bastante demorada, especialmente se o utilizador quiser encontrar um *loop* específico que seja semelhante ou vá complementar e ser ritmicamente compatível com um pré-selecionado.

Os desenvolvimentos ocorridos na área de Recuperação de Informação Musical ajudam a resolver este desafio, com a ajuda de métodos computacionais que permitem descodificar um sinal de áudio em descritores relevantes, os quais são essenciais para a indexação e navegação destas grandes bases de dados, pois fornecem informações sobre o sinal que permitem interpretá-lo sob diferentes níveis de abstração.

Ao interpretar estruturas musicais como objetos matemáticos, esta dissertação procura recuperar métricas para a similaridade de ritmos a partir de um sinal de áudio musical, e desenvolver um protótipo de compatibilidade rítmica que ajude na análise e quantificação da informação recuperada e apresente uma maneira de navegar e criar resultados musicalmente expressivos.

O objetivo principal é a criação de um ferramenta de visualização para compatibilidade rítmica de áudio musical num espaço topológico bidimensional, a partir do qual no futuro, será desenvolvido uma aplicação para performance musical que irá recombina e recuperar ritmos musicais compatíveis a partir de *loops* de áudio.

Nesta dissertação, é analisada uma base de dados multi-pista, escrita e tocada por músicos e artistas profissionais. A análise é calculada usando métodos para a análise rítmica do sinal de áudio, Rhythmic Patterns (RP), Rhythmic Histograms (RH) e Beat Spectrum (BS). O RH é melhor a discriminar padrões rítmicos e por essa razão, foi o método usado no protótipo. O protótipo desenvolvido usa Uniform Manifold Approximation and Projection (UMAP) para reduzir as dimensões dos dados gerados para a criação da visualização da base de dados de *loops* de áudio.

Keywords: Sinal de áudio musical, Compatibilidade rítmica, Computação sonora, Recuperação de Informação Musical, Análise musical

Acknowledgements

To my family, for their continuous support throughout these five years at FEUP, which felt like five seconds.

To my Father.

To my best friends, Mariana and João Bernardo, for pushing me through every step of the way.

To my dissertation supervisor, Gilberto Bernardes, for guiding me through this process and helping me at every moment since I started writing this document.

A special thanks to Diogo Cocharro for assisting in writing the article that was fundamental for the development of the prototype presented here.

Cláudio Lemos

*“Computers aren’t just a lifeless, cold object,
but rather a musical instrument that you can familiarize with .”*

A. G. Cook

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation	1
1.3	Objectives	2
1.4	Structure	2
1.5	Publication	3
2	Rhythm Representation and Metrics for the Similarity in Musical Audio: A State-Of-The-Art Review	5
2.1	Rhythm	5
2.1.1	Beat	6
2.1.2	Tempo	6
2.1.3	Meter	7
2.1.4	Timing	7
2.1.5	Grouping	8
2.2	Rhythmic Representations	8
2.2.1	Rhythmic Representation from Symbolic Manifestations	8
2.2.2	Rhythmic Representation from Audio Manifestations	11
2.3	Similarity	15
2.3.1	Feature-based Distance Metrics	16
2.3.2	Transformation-based Distance Metrics	17
2.4	Spatial Representation and Visualisation of Rhythmic Information	18
2.4.1	Conceptual Spaces	18
2.4.2	Navigation and Visualisation in Rhythmic Spaces	19
2.4.3	Rhythm Spaces	22
2.4.4	Visualizing multidimensional musical data	23
3	Understanding Cross-Genre Rhythmic Audio Compatibility	25
3.1	Assessing rhythmic compatibility	27
3.2	Dataset	27
3.3	Rhythmic Similarity Metrics	28
3.4	Computational Methods	30
3.5	Data and results	31
4	Mapping rhythmic information in a 2D topology space	37
4.1	Prototype for rhythmic compatibility retrieval	37
4.2	Topological visualisation	39

5	Conclusions	43
5.1	Summary	43
5.2	Contributions	43
5.3	Future Work	44
A	Code	47
B	Rhythmic Patterns	73
B.1	Full window stem analysis	73
B.1.1	Classical	73
B.1.2	Electronic/Fusion	74
B.1.3	Jazz	74
B.1.4	Musical Theatre	75
B.1.5	Pop	75
B.1.6	Rock	76
B.1.7	Singer/Songwriter	76
B.1.8	World/Folk	77
B.2	Windowed stem analysis	77
B.2.1	Classical	77
B.2.2	Electronic/Fusion	78
B.2.3	Jazz	78
B.2.4	Musical Theatre	79
B.2.5	Pop	79
B.2.6	Rock	80
B.2.7	Singer/Songwriter	80
B.2.8	World/Folk	81
C	Rhythmic Histograms	83
C.1	Full window stem analysis	83
C.1.1	Classical	83
C.1.2	Electronic/Fusion	84
C.1.3	Jazz	84
C.1.4	Musical Theatre	85
C.1.5	Pop	85
C.1.6	Rock	86
C.1.7	Singer/Songwriter	86
C.1.8	World/Folk	87
C.2	Windowed stem analysis	87
C.2.1	Classical	87
C.2.2	Electronic/Fusion	88
C.2.3	Jazz	88
C.2.4	Musical Theatre	89
C.2.5	Pop	89
C.2.6	Rock	90
C.2.7	Singer/Songwriter	90
C.2.8	World/Folk	91

D Beat Spectrum	93
D.1 Full window stem analysis	93
D.1.1 Classical	93
D.1.2 Electronic/Fusion	94
D.1.3 Jazz	94
D.1.4 Musical Theatre	95
D.1.5 Pop	95
D.1.6 Rock	96
D.1.7 Singer/Songwriter	96
D.1.8 World/Folk	97
D.2 Windowed stem analysis	97
D.2.1 Classical	97
D.2.2 Electronic/Fusion	98
D.2.3 Jazz	98
D.2.4 Musical Theatre	99
D.2.5 Pop	99
D.2.6 Rock	100
D.2.7 Singer/Songwriter	100
D.2.8 World/Folk	101
References	103

List of Figures

2.1	Beat perception	7
2.2	Formal Strings Representations	9
2.3	Geometric Representations	10
2.4	Onset detection	12
2.5	Rhythmic Patterns and Rhythmic Histogram	14
2.6	Global tempo estimation	15
2.7	Chronotonic Distance	18
2.8	Islands of Music	20
2.9	EarGram	21
2.10	MixMash	21
2.11	Rhythm Spaces	22
2.12	Iberian Folk UMAP representation	24
3.1	Overview of the system	26
3.2	Rhythmic Patterns, Rhythmic Histogram and Beat Spectrum representations on two different stems	29
3.3	RH results for Pop genre	31
3.4	Statistical significance (full stem analysis)	34
3.5	Statistical significance (windowed stem analysis)	35
3.6	Statistical significance (full stem vs windowed analysis)	36
4.1	Prototype overview	37
4.2	Prototype UI	38
4.3	UMAP results: all databases	40
4.4	UMAP results: drums and harmonic	40
4.5	UMAP results: harmonic	41
4.6	UMAP results: drums	41
5.1	Max for Live device	45

List of Tables

3.1	Rhythmic compatibility between genres (full stem analysis - median)	32
3.2	Rhythmic compatibility between genres (windowed stem analysis - median) . . .	32
3.3	Rhythmic compatibility between genres (full stem analysis - IQR)	32
3.4	Rhythmic compatibility between genres (windowed stem analysis - IQR)	33

Abbreviations

AIS	Adjacent Interval Spectrum
BPM	Beats Per Minute
BS	Beat Spectrum
CSI	Creativity Support Index
CSS	Concatenative Sound Synthesis
DAW	Digital Audio Workspace
Hz	Hertz
ILR	Interval-Length Representation
IOI	Inter-Onset Interval
MIR	Music Information Retrieval
M4L	Max for Live
PD	Pure Data
RH	Rhythmic Histograms
RP	Rhythmic Patterns
SMC	Sound and Music Computing
TEDAS	Temporal Elements Displayed As Squares
t-SNE	t-distributed Stochastic Neighbor Embedding
TUBS	Time Unit Box System
UMAP	Uniform Manifold Approximation and Projection
VST	Virtual Studio Technology

Chapter 1

Introduction

1.1 Context

Ever since the boom of Web 2.0 in the early 2000s, the amount of user-created content made available on the internet has increased immensely [65, 55]. The propagation of content online has led to the creation and imminent emergence of collections of musical content, which have proven to be popular within professional studio environments and personal music libraries. These audio collections can become rather large, which can make the process of exploring these databases a very time-consuming task.

Most music repositories today, restrict the search of their content through query-based retrieval of the database, through textual search, which finds items based on their metadata information, or current and more advanced systems which support search through audio queries, like the example of Shazam [61].

While professional music production has gradually been shifting from the live recording of musical instruments in a studio environment to the use of musical audio samples from these libraries [15], the lack of efficient retrieval methods so artists and producers can easily explore them is noticeable.

1.2 Motivation

Current methods used in the organisation of extensive musical collections are mostly textual. Metadata delivers information of music files which includes song title, artist name, release date, and musical genre [43, 42]. However, even this metadata information can be wrong or not readily available, which in turn, will lead to unreliable query results or in the latter situation, the inability to browse the database.

The ability to search and organise musical archives by similarity or even compatibility is something regarding the research area of Music Information Retrieval (MIR), which has made immense progress in recent years by finding solutions that help solve this challenge. Approaches by MIR include content-based analyses that help in the retrieval of audio descriptors. These descriptors can capture significant audio features such as tempo, beat, pitch, and timbre. These features are essential for the tasks we are trying to solve, such as the organisation of musical content by similarity and the grouping into categories.

1.3 Objectives

The aim of this project is the development of a visualization tool for the representation of rhythmic compatibility, which allows the user to navigate intuitively and visually through large musical databases. Upon this work and the developed prototype, further work will include the development an application that lets users create new musical content from their personal sample libraries.

With that in mind, we are able to remark some objectives for this dissertation as the following list:

- Development of a visualization tool for rhythmic compatibility retrieval from musical audio in a topology space, departing from rhythmic similarity metrics and techniques. The basis of the topology is a 2-dimensional space based on the information driven from the application of the common periodicity representing musical rhythm.
- Understand the best representation for musical audio rhythmic activations as a time-series from multiple audio descriptions (e.g., novelty function, spectral flux from magnitude, phase and complex domain, high-frequency content, etc.)
- Quantify emergent behaviours with well-established statistical mechanics techniques to understand the intrinsic attributes of the space in existing multi-track musical datasets from symbolic and audio manifestations.
- Develop novel compositional design frameworks central to networks of interconnected rhythmic representations that will be the basis to support future music performance applications for the recombination of musical rhythms from audio.

1.4 Structure

This dissertation is structured into five chapters. In Chapter 1, a brief introduction will be given about the proposal for this dissertation, contextualising the reader and explaining the motivation for this project and its objectives. Chapter 2 is where we present the current state-of-the-art, doing a literature review on the concept of rhythm and presenting the current techniques that can represent it from symbolic and audio manifestations. We elaborate on how similarity can be measured

through feature-based and transformation-based distance metrics and finally present some applications for the navigation and visualisation of rhythmic spaces. Chapter 3 will explain how rhythmic compatibility can be systematically assessed and reviewed, while analysing the behaviour of common metrics for rhythmic similarity when applied at scale in a database. Chapter 4 will showcase how multi-dimensional similarity metrics data at scale can be mapped into a two-dimensional topology space, and reviewed based on the analysis developed in Chapter 3. This chapter will also go over the developed prototype used for mapping this data. Finally, in Chapter 5 we will give some concluding remarks based on what was presented in the previous chapters and mention future work that could originate from the developed prototype.

1.5 Publication

The initial research for this dissertation and resulting state-of-the-art analysis led to the submission and pre-publication of a collaborative research paper [17] titled:

- Diogo Cocharro, Gilberto Bernardes, Gonçalo Bernardo, and Cláudio Lemos. A review of musical rhythm representation and (dis)similarity in symbolic and audio domains. 2021.

Throughout the development of this dissertation, a second collaborative research paper [40] was developed titled:

- Cláudio Lemos, Diogo Cocharro and Gilberto Bernardes. A review of musical rhythm representation and (dis)similarity in symbolic and audio domains. 2021.

The data generated throughout the dissertation was key to write the second research paper, which will be further explored in Chapter 3.

Chapter 2

Rhythm Representation and Metrics for the Similarity in Musical Audio: A State-Of-The-Art Review

2.1 Rhythm

Rhythm is a concept people tend to associate with music. Gustafson [31] states the importance of rhythm not only to music but for other fields and areas, giving the example of how rhythm can be observed through the description of the movement of a painter or the undulating hills on the horizon.

Over the years, many authors have tried to give their own proposal for a definition behind this concept that most people are familiar with. While Cooper and Meyer [18] give a broader definition of rhythm, “*To study rhythm is to study all of music. Rhythm both organises, and is itself organised by, all the elements which create and shape musical processes*”. Lowe [46] puts it as simple as “*The term Rhythm is constantly erroneously applied. It has only one true meaning in music—the number of bars in a phrase.*”. Even earlier definitions can be found, going back to Ancient Greek where Plato [59] implied that “*Rhythm is ordered movement*”.

As noted by Eschman in [22], “*The literature on Rhythm is voluminous*” and there is not a real consensus on the actual definition of rhythm, leading to a diversity of takes on the term from different authors, causing confusion on what its actual meaning.

In his work, London [45] remarks the importance of rhythm to the musical domain, supported by Meyer’s claim denoting rhythm as being one of two¹ primary parameters of the musical struc-

¹Meyer [53] specifies pitch as the second essential musical structure

ture. London finds that by specifying the rhythmic organisation of a musical piece, we can capture its essential structure. While alterations in instrumentation, orchestration or dynamics can be presented as a different arrangement of the same musical work, changes in rhythm result in a new piece.

Musical rhythm is often described as the temporal aspects of music [77], assuming different representations in a score, measured from a live performance or just existing through the perception of the person that is listening.

The disposition of musical notes in a musical sheet can be analysed alongside their duration and rhythmic patterns, which is what rhythm is concerned with [45]. When notes are more or less regular, it can lead to the listener having a perception of beat or tempo, a phenomenon that happens in a frequency range below the human hearing [16].

When listening to a musical piece, the listener can tap their feet along with it. In musical genres such as pop and rock, there are constant and strong regularities, commonly referred to in literature [67] as pulses². Going beyond popular music, we can find intricate musical pieces where these regularities can suddenly change. In the first movement of Beethoven's *Pathétique sonata*, the slow opening reappears later in the piece, just after a fast main section. While this process is natural to us, almost an intuitive response while listening to music, transferring the process of automatic rhythm estimation to the machine can be quite challenging.

Zapata [77] divides musical rhythm into five components: beat, tempo, meter, timing, and grouping, which we detail next.

2.1.1 Beat

Beat can be a collection of events and accents in music characterised by the perception that listeners get while tapping their feet. Temperley [68] lists six cues agreed upon researchers for beat finding:


1. for beats to coincide with note onsets,
2. for beats to coincide with longer notes,
3. for regularity of beats,
4. for beats to align with the beginning of musical phrases,
5. for beats to align with points of harmonic change,
6. for beats to align with the onsets of repeating melodic patterns.

2.1.2 Tempo

Tempo relates to beat as being the frequency rate at which pulses occur, and it is measured by beats per minute (BPM) [29], a term commonly used by musicians when referring to the speed of

²In this thesis, the pulse felt by the listeners will be referred as beat

a song. As mentioned above, the perception that listeners experience from the beat, can translate into the rate at which they would tap along with the music, but as noted by Lapidaki [39] this perception can vary from listener to listener. These variations can occur due to the listener's age, musical training, musical preferences, and the context in which the music is being listened to. However, these differences in perception are not just random variations, but somewhat different ways the listener is focusing on the beat and can be quantifiable as ratios [11]. Ultimately, the tempo of a musical piece depends on the listener's perception of what the beat is. While listener 1 might count the tempo as 'one and two and three and four and' (Figure 2.1.a), listener 2 can count it as 'one two three four one two three four' (Figure 2.1.b).

1. 

a) 1 and 2 and 3 and 4 and 1 and 2 and 3 and 4 and

b) 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4

Figure 2.1: Two different ways the same beat can be perceived by different listeners.

2.1.3 Meter

In their work *The Generative Theory of Tonal Music* [41], Lerdahl and Jackendoff define meter as the metrical structure of a musical piece based on the coexistence of a collection of regularities that can span from shorter to longer time divisions. These regularities alternate between stronger and weaker pulses in the music, making it easier for them to be differentiated from each other. In turn, it is easier to find the strongest meter, one that is more accentuated, making the process of meter perception as simple as finding and filtering through musical accents in order to retrieve rhythmical periodicities.

2.1.4 Timing

Timing can tell us when events occur, by giving us a picture of the temporal representation. Temporal deviations can lead to expressive timing, that can manifest in a musical performance with tempo changes, event shifts or swing factors [77]. These factors provide musical genres with a sense of syncopation, swing, expressive performance and groove [19].

In [38], Kendall and Carterette were able to demonstrate successfully how listeners would pick up these expressive timing deviations from performers, on a variety of musical instruments. The most interesting take from this research was that there was no difference in timing perception results from musicians and non-musicians participants.

These concepts will be important later in this thesis, when we discuss how can two instruments in the same musical piece have different rhythmic patterns, each giving a different groove feeling to the listener and still be rhythmically compatible with each other.

2.1.5 Grouping

In [41], Lerdahl and Jackendoff clarify the elements that compose the rhythmic structure of music and make a clear distinction between the concepts of meter and grouping. While, as noted above in Section 2.1.3, meter is the alternation between stronger and weaker pulses, grouping is concerned as to how music is organised and segmented at different scales and (phrase structure) [16, 77] and over specified durations, making it a hierarchical property of musical structures. Furthermore, although the two concepts of meter and grouping are theoretically independent, Lerdahl and Jackendoff find in their work that there is a relationship between the two, manifested in musical arrangements by the alignment of the stronger pulses with grouping boundaries.

In [69], Todd compares meter and grouping to the time- and frequency-domains, making the analogy of meter corresponding to the frequency and grouping to wavelength.

2.2 Rhythmic Representations

When encoding rhythm, we have to take in consideration that the way we choose to represent music is going to influence its composition, observation, as well as the way it can be understood or analysed [35]. In [33], Hewlett and Selfridge-Field remark the importance of musical information into constructing a model for the measurement of rhythmic similarity.

Manifestations of music can either occur in the symbolic or sub-symbolic domains. In this section, we are going to analyse and present two different forms of retrieving rhythmic information from both symbolic and audio-form musical manifestations.

2.2.1 Rhythmic Representation from Symbolic Manifestations

In the symbolic domain, rhythmic information is represented explicitly, which in turn, makes the process of extracting this information much more straightforward. From symbolic manifestations, rhythm can be encoded into two different methods: formal strings and geometric representations.

In Sections 2.2.1.1 and 2.2.1.2, we will review the computational methods that best help represent symbolic information. Figures 2.2 and 2.3 help demonstrate the two types of symbolic representations. Formal strings adopt binary symbols for rhythm representation and geometric representations use text symbols and graphics to translate rhythm.

2.2.1.1 Formal Strings

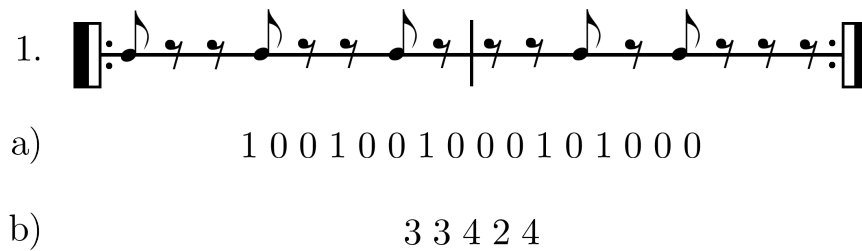


Figure 2.2: Formal Strings representations of the Clave Son rhythm, a typical rhythmic pattern used in Cuban music, represented here in 1 in its atomic beat divisions. a) Binary notation. b) Interval-Length Representation.

Using binary symbols, we can translate note onsets into 1 and silent intervals into 0, resulting in a binary string of ones and zeros [44]. In [66], Sethares claims the binary notation to be one of the simplest ways to represent rhythm. The resulting string will be composed of a sequence of values indicating the pulse's activity or silence, where the length of this sequence represents the duration of the rhythm being represented.

The Interval-Length Representation (ILR) is a numerical notation which aims at representing the duration of intervals between the onset times of consecutive notes, commonly known as inter-onset intervals (IOI) [75]. In [71], while Toussaint indicates the compactness and ease-of-use of ILR as an advantage, he also mentions how the temporal dimension of the interval's relative durations can get lost by not being easily observed. ILR is further expanded into two visual displays of rhythm based on the histogram representation that Gustafson proposes in [31], adjacent-interval-spectrum (AIS) and temporal elements displayed as squares (TEDAS). Gustafson points out that the two parameters being used in this 2-dimensional representations should be time-based.

2.2.1.2 Geometric Representations

Visual iterations of the binary notation have since appeared, including the box notation and the Time Unit Box System (TUBS). The box notation uses text symbols to represent rhythm. Similar to the ones and zeros from binary notation, this notation uses an X to represent an active pulse and a hyphen or dot to represent silence. TUBS enhances both notations by translating a rhythmic sequence into sequences of square boxes. While a black box represents a note onset, an empty box represents silence [44, 75].

Departing from the ILR representation mentioned in Section 2.2.1.1, AIS is a spectrum-based representation of IOIs resulting into a 2-dimensional plot, where the sequences of intervals are mapped into the Y-axis, while the temporal information of rhythmic patterns gets lost in the X-axis. TEDAS tries to solve this problem by combining both ILR and AIS and expanding the active pulses information into the X-axis. In TEDAS, rhythmic information from IOIs is represented as

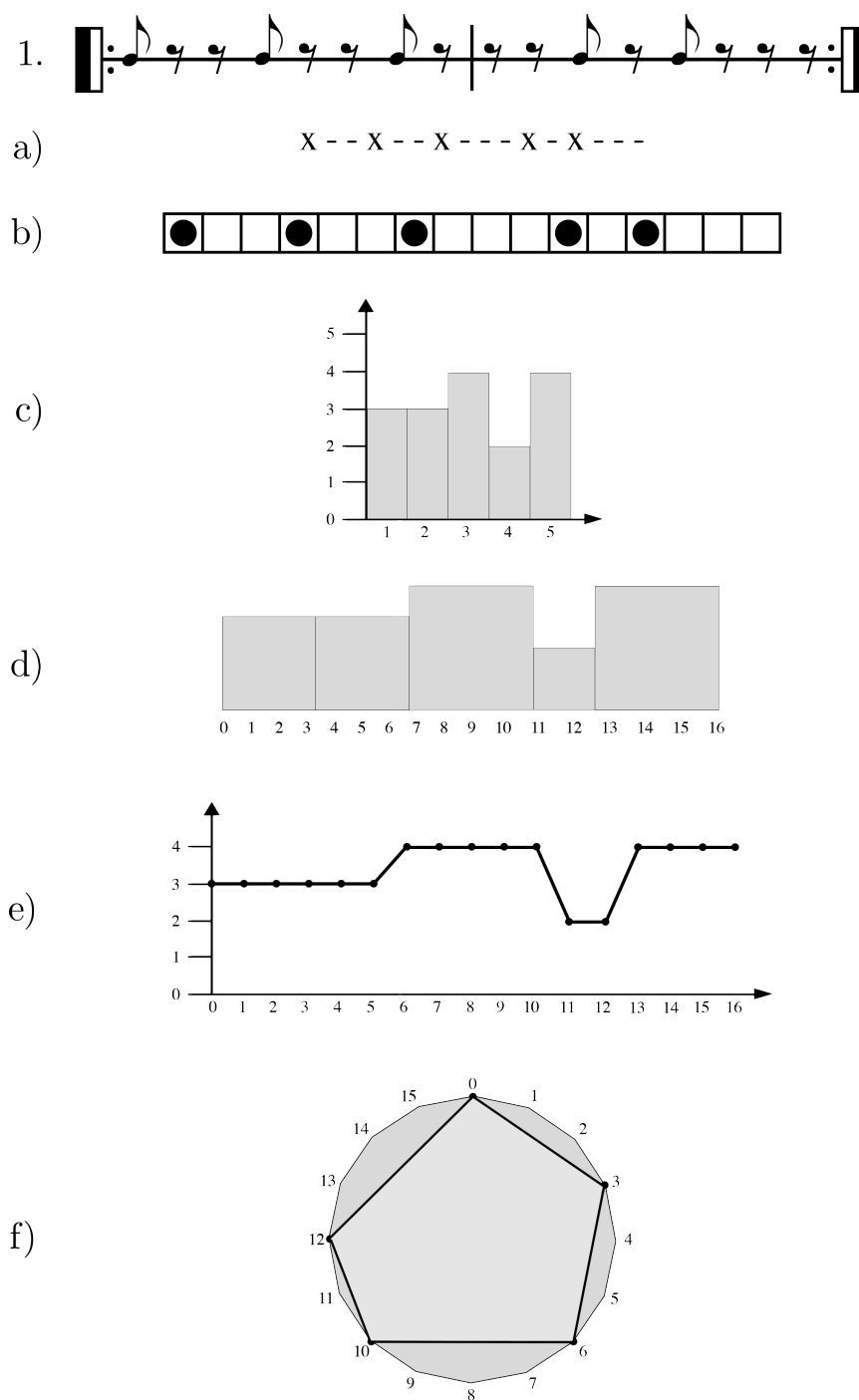


Figure 2.3: Geometric representations of the Clave Son rhythm (1). a) Box notation. b) Time Unit Box System. c) Adjacent-Interval-Spectrum. d) Temporal Elements Displayed As Squares. e) Chronotonic Chain. f) Convex Polygon.

square-shaped boxes. The expanded information retrieved from the X-axis can further indicate the temporal index of the pulse and the relative durations between onset events.

Hoffman-Engl presents in [35] another geometric representation that exposes the same rhythmic information as TEDAS. The Chronotonic Chain results from connected points representing atomic beats. A rhythm gets broken down into its smallest unit, a 16th note, which Hoffman-Engl refers to as atomic beats. Each atomic beat will get assigned a y-coordinate based on its IOI. Finally, all the points are connected into polygonal curves that form the chronotonic chains.

The Convex Polygon was presented by Toussaint in [70], where he imagines a clock divided into 16 time units instead of the typical 12, where each vertex represents a pulse of a cyclic rhythmic pattern. The Convex Polygon introduces a more realistic cyclic representation since the first vertex coincides with the last vertex, something that others representations such as the Chronotonic Chain lack since these two beats are distant from each other.

2.2.2 Rhythmic Representation from Audio Manifestations

Audio feature extraction has been one of the most important research areas inside the field of Music Information Retrieval, aiming at the analysis of audio signals in order to extract meaningful information which then gets translated into descriptors that can be understood by the computer [54]. Unlike symbolic manifestations where information is readily available, in the audio realm, this rhythmic information is implicitly encoded so that new challenges arise to access the same information. A simple example demonstrating these challenges is the detection and transcription of polyphonic sounds, where multiple sounds are being played simultaneously.

Roads [64] splits rhythmic description into three levels of abstraction: low-level, mid-level and high-level. In Section 2.2.2.1, we are going to review techniques of feature extractions based on physical characteristics of the audio signal. Section 2.2.2.2 introduces the perceptual impact of the audio manifestations, where extraction techniques try to describe the properties of the audio signal based on human perception [2]. We will not be reviewing high-level rhythmic description techniques since these abstract in a cognitive way, extracting information such as musical style and genre. For that reason, in this dissertation, we will be focusing on low- and mid-level representations of rhythmic description, as these techniques analyse the signal formally and rhythmically.

2.2.2.1 Low-level Representation of Rhythmic Description

At a low level, the audio descriptor analysis can capture rhythmic descriptors and categorise them through energy and spectral changes. The main goal at this level is to locate these sudden changes in the audio signal characterised by the beginning of transient regions, which Bello [3] describes as a sudden burst of energy or a change in the short-time spectrum of the signal. For the most part, an increase in the signal's amplitude envelope will usually indicate the occurrence of an onset. In some situations, multiple note onsets can occur without a change of the amplitude envelope. For example, when a violinist plays slurred notes in a single down bow, it results in a pitch change, while the amplitude stays at the same level.

An onset is composed of three different regions: the attack onset, the actual onset and its transient. The attack will occur when there is a sudden increase in energy, which corresponds to the build-up of the amplitude envelope. A transient occurs right at the beginning of a musical tone and is characterised by a short and high amplitude noise-like sound component typically followed by a dampening of the sound and slow decay of energy. An onset refers to the single instant at which the transient starts.

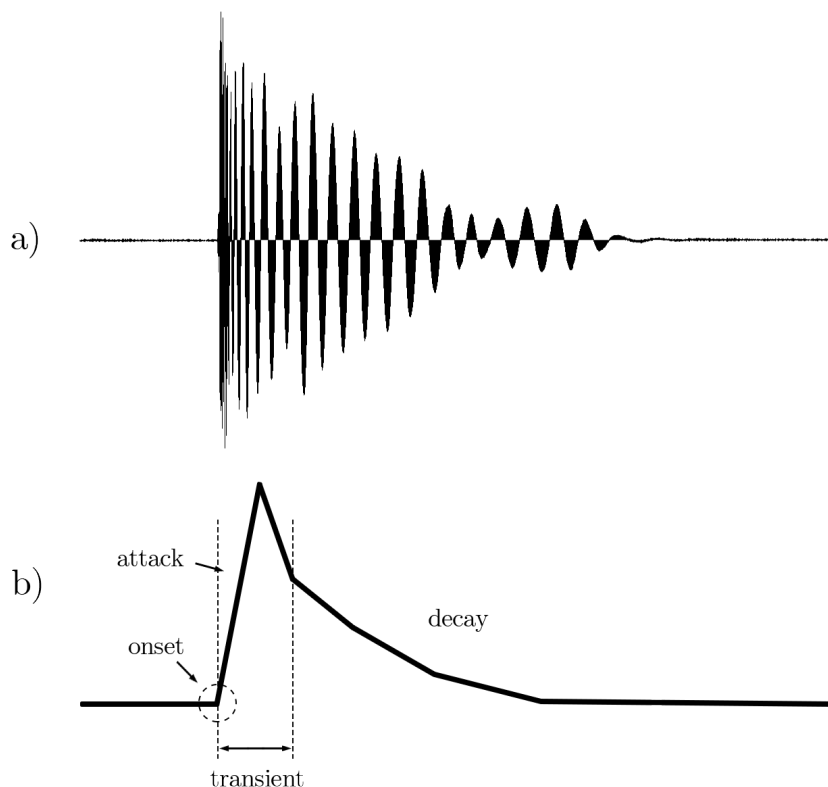


Figure 2.4: a) Waveform of a single note being played. b) Resulting amplitude envelope corresponding of an onset,.

Novelty functions can extract low-level information from an audio signal by detecting changes in properties from said signal such as energy and spectral content. In this section, we will take a look at the two different types of novelty functions, energy-based and spectral-based. Even though both types perform at a low-level of abstraction, each one will give better and more accurate results in different situations.

When playing a single note on a synthesiser, the performer creates a sudden increase in energy in the audio signal. Transforming the signal into an energy function can give information about local energy levels for each time instance. Subsequently, by getting the difference of two consecutive energy values from the first function, we will have the derivative of the local energy function, from which only the positive values are kept since we are interested in sudden energy increases.

For the mentioned example above of the violin, we need to use a spectral-based novelty function. Since there is no sudden burst of energy, but rather a change in the pitch of the notes being played, this may be captured across the frequency domain. Looking at the signal's short-time spectrum can not only help overcome this challenge but even more complicated ones such as polyphonic note onset events. Spectral-based novelty functions aim at converting the audio signal into a time-frequency representation followed by the analysis of the frequency domain content to detect changes. This method works in the same way as energy-based novelty functions, but instead of looking at the energy values, the difference between consecutive spectral vectors is computed, resulting in the spectral flux [3].

2.2.2.2 Mid-level Representation of Rhythmic Description

The low-level audio features analysed in Section 2.2.2.1, reveal the positions of note onsets occurrences through local maxima from the novelty functions and are the stepping-stone for the computation of periodicity functions³, crucial for the subsequent detection of mid-level features such as tempo and beat [28].

Mid-level audio features try to approximate to perceptual properties known by human listeners [78] (e.g. pitch, loudness, rhythm, and harmonicity) while revealing the same level of information as symbolic representations. Richards [63] states that some strategies exist that can be combined to integrate human perception in the processing of the audio signal, through the use of perceptually relevant features which help characterise different aspects of the audio content.

Multiple periodicity functions have been proposed by different authors, such as the auto-correlation function [20] and beat spectrum [23]. Each function differs on the attributes it focus on and the way the information is presented. In this section, we will review the Rhythmic Patterns function, also known as Fluctuation Patterns, from which the metrics for rhythmic compatibility proposed in Chapter 3 will evolve from.

Rhythmic Patterns [58, 57, 62, 42] is a matrix representation of fluctuations in different frequencies on critical bands to the human's listening range, that describes rhythm as amplitude modulations. The two-stage extraction process of the Rhythmic Patterns starts by grouping the frequency bands by loudness sensation, which is computed using a Short-time Fourier Transform. The resulting spectrum is then transformed into a time-invariant 24 critical Bark bands representation based on the modulation frequency, achieved by applying a second Fourier transform. The Rhythmic Patterns function is then able to capture recurring rhythmic patterns in the individual critical bands. If, for example, there is an occurrence of a high amplitude at the modulation frequency of 2 Hz in the resulting matrix, that indicates the presence of a rhythm with 120 BPM [42]. To the human listener, the notion of rhythm ends at about 15 Hz⁴. For that reason, in the Rhythmic Patterns, only information up to 10 Hz is considered [42].

³A sequence of note onset times detected by low-level novelty functions is computed into IOIs, revealing the rhythmic structure's periodicity.

⁴Equivalent to 900 BPM.

The Rhythmic Histogram feature [43] is used to represent the general rhythmic information of an audio track throughout all bands. The information is not stored per critical band but rather the sum of the magnitudes from each modulation frequency bin, which form the final histogram of rhythmic energy per modulation frequency. The histogram is composed of 60 bins that reflect the modulation frequencies between the same range of 0 and 10 Hz.

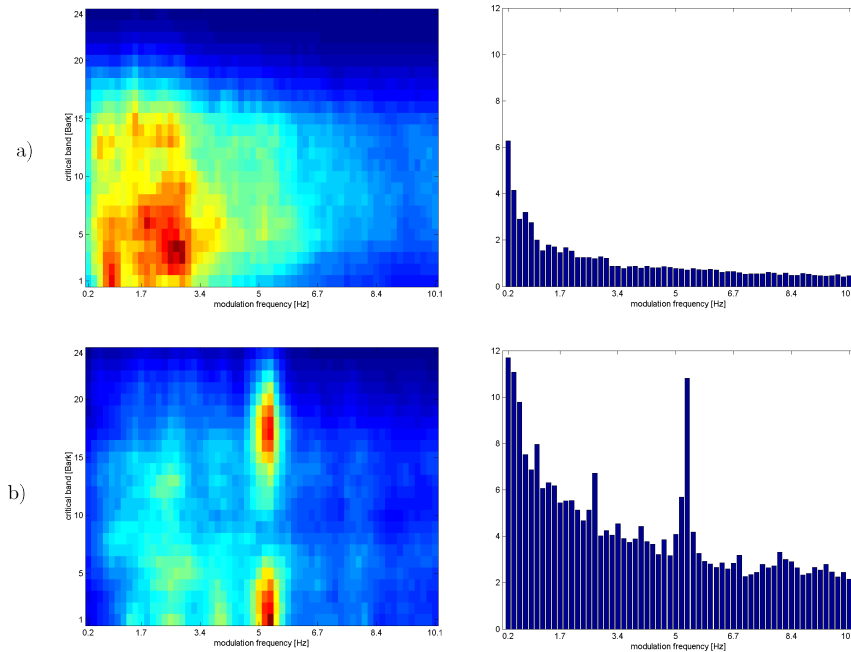


Figure 2.5: Corresponding Rhythmic Patterns and Rhythmic Histogram functions for a) classical music ("Blue Danube Waltz" by Johann Strauß) and b) rock music ("Go With The Flow" by The Queens Of The Stone Age) [42].

In Figure 2.5, both the Rhythmic Patterns matrix and the Rhythmic Histogram are presented for two different tracks. The first is a classical piece titled "Blue Danube Waltz" by Johann Strauß, while the second track is a song by the rock band Queens of the Stone Age titled "Go With The Flow". The analysis of the Rhythmic Patterns matrices show a prominent rhythm in (b) at the modulation frequency of 5.34 Hz, most likely indicating the bass guitar's presence, in (a), we cannot detect a clean and distinctive band presence, but rather a blob in the lower region of the modulation frequencies, which is a typical indication of classical pieces [42]. Analysing the Rhythmic Histograms, the histogram corresponding to the Queens of The Stone Age track clearly shows a peak at 5.34 Hz, while the classical piece stays predominantly at the lower modulation frequencies, indicating less energy typical of classical music.

Tempo estimation and beat tracking describe mid-level rhythmic representation which capture the predominant local pulse as described in Section 2.1. Both are measured as a single value, typically expressed as beats per minute. In classical music, standard tempo markings range from *grave* at around 30 BPM to *prestissimo* measuring at 200 BPM. Estimating the global tempo of a

musical track can be a challenge when there are fluctuations throughout the song. Current state-of-the-art methods perform well in commercial genres with strong and accentuated regular beats, such as pop and rock music. In Figure 2.6, a tempo estimation technique is applied to a drum loop, resulting in evenly spaced out red lines that align with the onsets of the audio signal. Tempo estimation and beat tracking can get more challenging in classical music which is characterised by the alternation between music markings and is typically a non-percussive musical genre [51]. Recent researchers have found progress on the use of deep learning algorithms for tempo detection and beat tracking, showing excellent results for music with no drums presence [27].

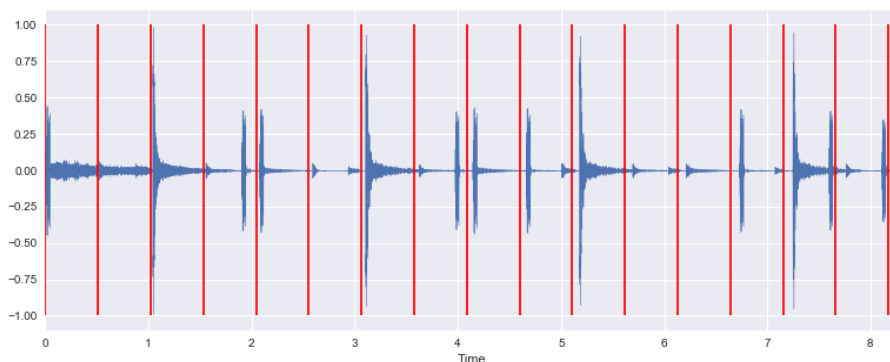


Figure 2.6: Global tempo estimation of a drum loop.

2.3 Similarity

In [35], Hofmann suggests that while at first the interest in melodic and rhythmic similarity may seem unmotivated and unprovoked, this issue starts gaining importance when we look at the innumerable amount of contexts in the musical field where the metrics for similarity may be applied. Hofmann offers some examples such as: a) a composer trying to produce a variation to a musical piece, b) an ethnomusicologist trying to classify the melody of a song, c) a music teacher trying to evaluate how close to the original song a student’s performance is, e) a court judge trying to make a final decision on a copyright infringement legal case, and finally, relevant to our dissertation, d) a user trying to navigate a musical database while querying for a specific melody.

The process of measuring the similarity of two different rhythms is a fundamental problem in computational music theory [70]. When picking a similarity metric for comparing a pair of rhythms, we should take into consideration how the rhythm is represented. These metrics are guided by what should be measured in the rhythm and how it should be measured. Toussaint presents in [70] two different approaches to rhythm similarity; 1) Feature-based Distance Metrics compare the number of common attributes between two rhythms, while 2) Transformation-based Distance Metrics compute the necessary effort to transform one rhythm into another. Both metrics will be reviewed in Sections 2.3.1 and 2.3.2.

2.3.1 Feature-based Distance Metrics

Feature-based Distance Metrics are computational methods that compare the number of rhythmic aspects that two different rhythms have in common. In this section, we will review two methods: the Hamming Distance and the Euclidean Interval.

2.3.1.1 Hamming Distance

The Hamming distance, d_H , was proposed by Richard Hamming in [32] and is a natural measure of dissimilarity between two binary sequences. It analyses the number of indexes in the two strings where the values do not match. Some researchers have preferred to compute the number of matching values when the interest is to calculate similarity [52].

Each rhythm is represented by a vector $X = (x_1, x_2, \dots, x_n)$ where x_n represents the pulse in a similar way to the binary notation presented in Section 2.2.1.1. When a note is played at index i , $x_n = 1$, when there is silence $x_n = 0$.

The Hamming Distance between two rhythms $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$ is given by the following equation:

$$d_H(X, Y) = \sum_{i=1}^n |x_i - y_i| \quad (2.1)$$

Toussaint [70] does not recommend the use of this metric for the problem of rhythmic similarity, since it only measures the occurrence of a mismatch and not the displacement of this dissimilarity. Bookstein *et al.* [10] propose their approach to this metric called the Fuzzy Hamming Distance. This new metric tries to overcome the limitations present in the original proposal by Richard Hamming, by accounting not only the shift of onsets but also insertions and deletions [72].

2.3.1.2 Euclidean Interval Vector Distance

The Euclidean Interval Vector Distance, d_E , is a better approach than the Hamming Distance for measuring rhythmic similarity since it takes advantage of the inter-onset intervals representation. Rhythms are represented by a vector of numbers characterising the IOIs, in $X = (x_1, x_2, \dots, x_n)$ where $x_n = i$ is the number of vertices skipped by the i^{th} convex polygon edge, counting from vertex 0. [70].

With this metric, the dissimilarity between two different rhythms $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$ can be calculated with:

$$d_E(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2.2)$$

The Euclidean distance can be further applied to the similarity and categorisation of rhythms, which is helpful when organising rhythms in classes through the use of clustering algorithms [21].

2.3.2 Transformation-based Distance Metrics

The Feature-based metrics presented in the previous section, do not consider displacements in the metrical or temporal structure of the rhythm. Transformation-based metrics solve this issue by computing the distance that it takes to alter one rhythm into another. This section will present the Swap Distance and the Chronotonic Distance.

2.3.2.1 Swap Distance

The swap distance, d_{swap} , computes the minimum swaps necessary to convert one rhythm into another, taking a binary rhythm representation as input. In [75], Toussaint considers this metric as being equivalent to the minimum value of the sum of distances⁵ travelled by all the onsets during the transformation. For an effective transformation, the swap distance requires the two rhythms to be composed of the same amount of pulses, and its computation is only possible by interchanging adjacent elements.

For example, consider rhythms $X = [x - x - x x - x - x - x]$ and $Y = [x - x x - x x - x - x -]$ with the same length. The i -th onset of X must travel a certain distance to reach the i -th onset of Y . When $i = 1$, that distance is 0, since the first onsets of both X and Y have an index of 0. For $i = 3$, that distance is 1, because while the third onset of X has an index of 4, the third onset of Y has an index of 3. We can now compute vectors U and V that store the indexes at which the i -th onset of X and Y occur. The result is $U = (1, 3, 5, 6, 8, 10, 12)$ and $V = (1, 3, 4, 6, 7, 9, 11)$. The difference between u_i and v_i is the necessary distance to bring those two onsets into alignment.

After computing vectors U and V , we can calculate the swap distance of rhythms X and Y using the following formula:

$$d_{swap}(U, V) = \sum_{i=1}^k |u_i - v_i| \quad (2.3)$$

2.3.2.2 Chronotonic Distance

In [35], Hofmann-Engl considers the chronotonic chains representation, previously presented in Section 2.2.1.2, as being a vector of atomic units [70]. A weighted Euclidean distance is computed on the corresponding chronotonic vectors of two different rhythms to calculate their similarity.

Considering the chronotonic chains representation as a function, opens the possibility for the use of distance functions, which have been previously used for pattern recognition [73]. When given two probability density functions $f_1(x)$ and $f_2(x)$, we can measure the distance between the two. One technique is the discrimination information, d_{KL} , also known as Kullback-Liebler divergence, and is computed by the following function:

$$d_{KL} = \int f_1(x) \log \frac{f_1(x)}{f_2(x)} dx \quad (2.4)$$

⁵These distances are measured as the number of pulses.

Toussaint [74] proposes the Kolmogorov variational distance, d_K , closely related to the Kullback-Liebler divergence:

$$d_K = \int |f_1(x) - f_2(x)| dx \quad (2.5)$$

Figure 2.7 shows an original rhythm and a variant rhythm are presented in the Chronotonic Chains representation. The grey area represents what is measured by the Kolmogorov variational distance, or area-difference between the two Chronotonic Chains.

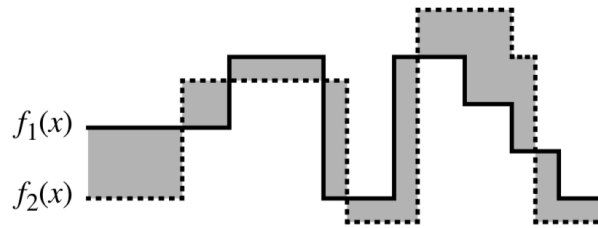


Figure 2.7: Chronotonic Distance [70].

This approach includes a mechanism presented in [35] that transforms one chronotonic chain into another while adopting this strategy to help measure the similarity between the two rhythms corresponding to each chain. Similarly with the Swap Distance, the Chronotonic Distance expects the two rhythms to be of the same length. However, as shown in [35], if the two rhythms have different lengths, the chronotonic chain corresponding to the shorter rhythm can be extended until it matches the longer chain.

2.4 Spatial Representation and Visualisation of Rhythmic Information

Once similarity is computed within a musical database, it is in our best interest to organise this information in groups so that we can explore it intuitively. This section will analyse the work of Gärdenfors [24], that explores the idea of a conceptual space for rhythmic information. We will also present some applications for navigation and visualisation of this information, Islands of Music, EarGram and MixMash. The three applications are similar to what we intend to develop at the end of this dissertation, and we will explain how these two differentiate from our proposal.

2.4.1 Conceptual Spaces

In [24], Gärdenfors finds that the symbolic representation as shown in Section 2.2.1 approach may not be the best suit for the computation of rhythmic similarity, claiming that it models and

explains problems at different levels of abstraction. He then resumes proposing an intermediate representation that is geometrically based but is framed as a conceptual space.

The proposal in [24] consists of a framework for conceptual spaces that aims at solving the problems not tackled by the symbolic approach, being complementary to this representation, even though in the author's opinion, symbolic representation lessens the value of rhythmic similarity. Similarity is only represented by rules or axioms and an explicit representation of symbols, which produce similarity values [24, 36].

In his work, Gärdenfors gives the following definition of concepts: "*a set of regions in a number of domains together with an assignment of salience weights to the domains and information about how the regions in different domains are correlated.*". He proceeds to make the distinction between concepts and properties. While the first is modelled by several domains, the latter is only modelled by one. Concepts are linked together with the idea of rhythmic similarity, as they group objects with similar traits in clusters, acting as a rhythmic similarity classifier.

2.4.2 Navigation and Visualisation in Rhythmic Spaces

In this section, we will analyse different proposals that help visualise and navigate rhythmic information in intuitive ways. The applications we are going to present are Islands of Music, EarGram and MixMash.

2.4.2.1 Islands of Music

Pampalk proposes in his thesis a new way to visualise and navigate through musical collections [57] called Islands of Music.

In this graphical interface, geographical maps are used as a metaphor where musical genres get represented as islands and elevations in the land such as mountains and hills represent sub-genres. Similar genres are placed close together and may even be connected by a land passage if the similarity value is high. Genres that are perceptually different are separated by sea.

A musical database is placed on the map according to genre or musical and rhythmic attributes. Mountains and hills are labelled with these attributes, as seen in Figure 2.8, which helps the user to navigate the database better.

Pampalk initially developed Islands of Music for the exploration of unknown musical collections, but its purpose can extend to other use cases. For example, the interface can help users discover new music, organise their personal music collection or simply be used as an interface for a digital music library.

2.4.2.2 EarGram

EarGram is an open-source application developed by Bernardes [5, 6] in Pure Data (PD) for the interactive exploration of musical databases and creative sonic creation with real-time concatenative sound synthesis (CSS) techniques. The system collects information from the audio signal through data mining capabilities that reveal musical patterns and temporal organizations [5, 6]. It

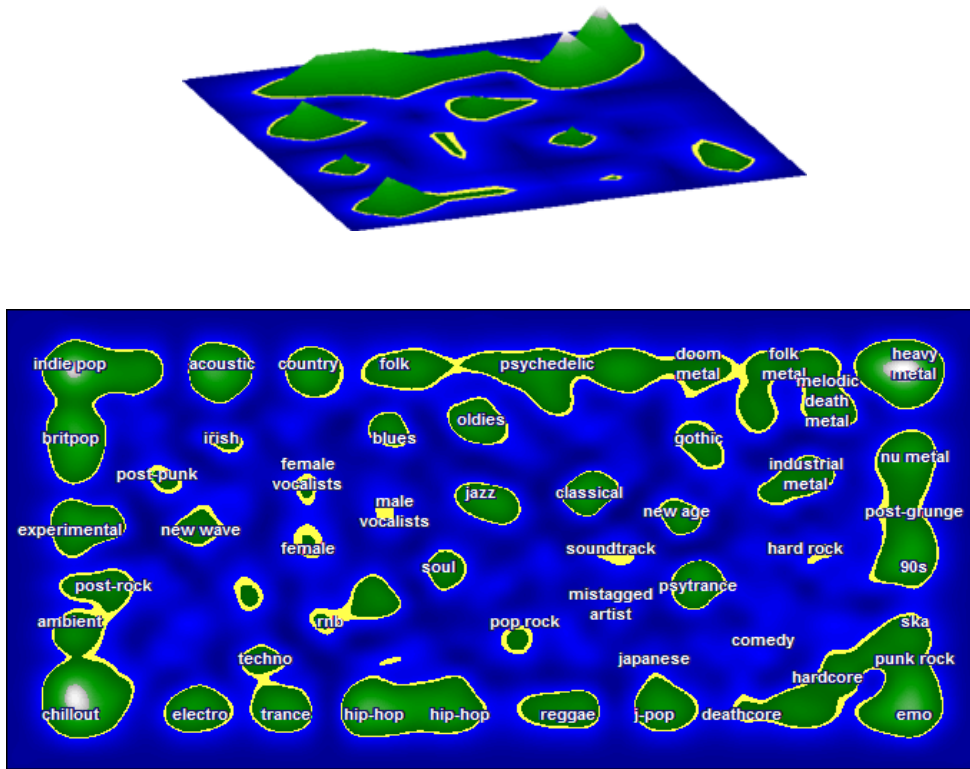


Figure 2.8: Islands of Music interface [57].

presents the collected information in a user-friendly interface that helps the use of this application for its creative intention.

Bernardes presents four different generative strategies that automatically rearrange and explore the database of audio snippets into musically and sonically coherent creative output: *infiniteMode*, *shuffleMeter*, *soundscapeMap* and *spaceMap*. The four modes help the user explore a wide range of musical applications, such as the automatic generation of soundscapes, remix of an original song and mashups [7].

EarGram displays the retrieved information in a two-dimensional space, providing an intuitive way for the user to understand intrinsic perceptual qualities from a collection of sounds, which are used to organise the database into clusters.

2.4.2.3 MixMash

Departing from Bernardes *et al.* [4] proposal of the harmonic mixing techniques for musical mashups, *MixMash* [47, 48] is an interactive application that guides the user into the creation of musical mashups based on cross-modal associations between the analysis of musical content analysis and further visualisation in a two-dimensional space.

MixMash tries to overcome a few design limitations identified in [4] by proposing a new method for interactive visualisation of audio attributes such as timbral similarity, harmonic compatibility, the density of note onsets and spectral region. Each track is represented as a node in

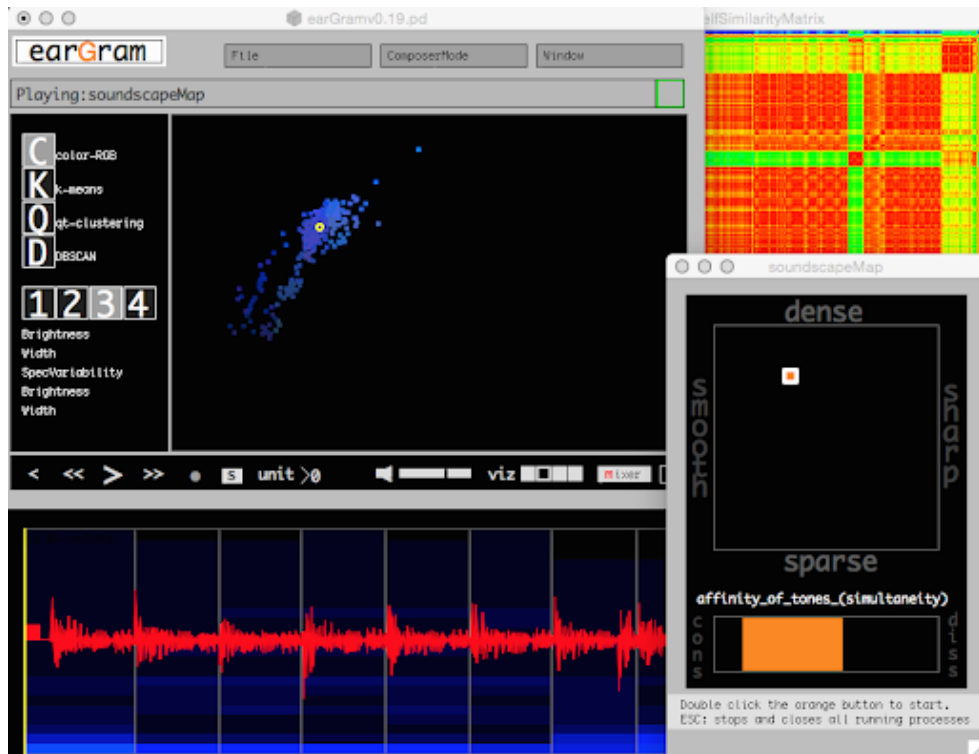


Figure 2.9: EarGram interface [5, 6].

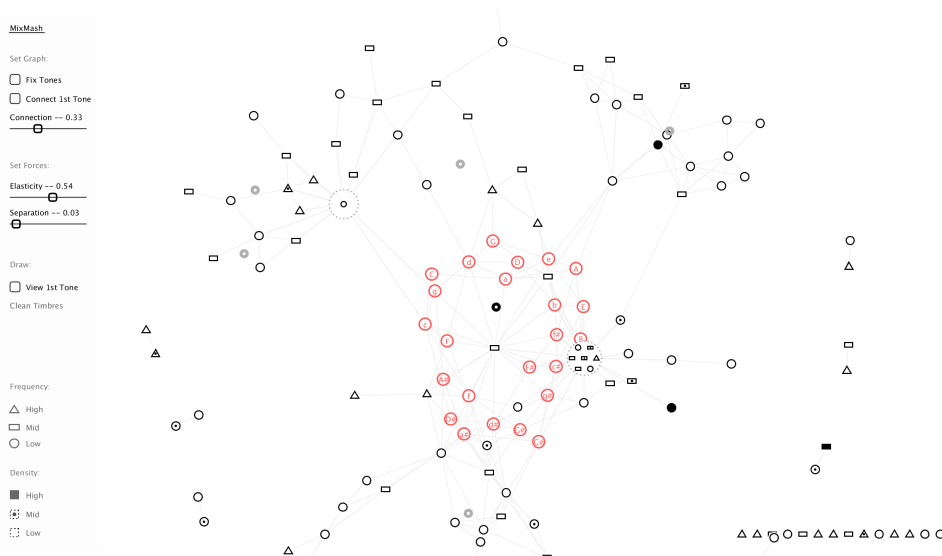


Figure 2.10: MixMash interface [47, 48].

a force-directed graph, in which the edge connections represent the harmonic compatibility. The intuitive visual language, as presented in Figure 2.10, is used to promote the user's creativity when creating musical mashups.

2.4.3 Rhythm Spaces

In [25], Daniel Gómez-Marín *et al.* present rhythm spaces as metaphors to visualize and interact with rhythms by taking advantage of previous research made on alternatives low-dimensional music spaces, music cognition, and also music interaction made possible with intelligent musical agents.

In the developed paper, the team focuses on symbolic sequences and how this rhythmic information can be organized into low-dimensional spaces, in order for this data to be grasped by the reader, while also acting as a way to organize it into clusters and generate music. These tools can be the backbone for the next generation of music creation applications, where currently creativity is limited to unidimensional orderings such as drop-down menus in drum-loop creation tools (e.g., Logic Pro’s Drum Machine Designer or Ableton’s Drum Rack).

Through the analysis of state-of-the-art systems based on the transformation of symbolic representations and audio loops [37, 56] and computational approaches based in neural networks and genetic algorithms [13], Daniel Gómez-Marín *et al.*’s goal is to create a system that can become an essential tool used for music production, where producers are able to browse and get a sense of rhythmic patterns occurring in their own collection of audio loops.

Daniel Gómez-Marín *et al.* demark that a tool for this context should be able to maintain its low dimension, so it is intuitive for the user to browse through the musical data, and keep closeness when similar rhythmic patterns are occurring, while separate opposite patterns. In Figure 2.11, the team maps out a collection of different rhythmic patterns and musical styles (e.g., soul, hip-hop, garage, house) .

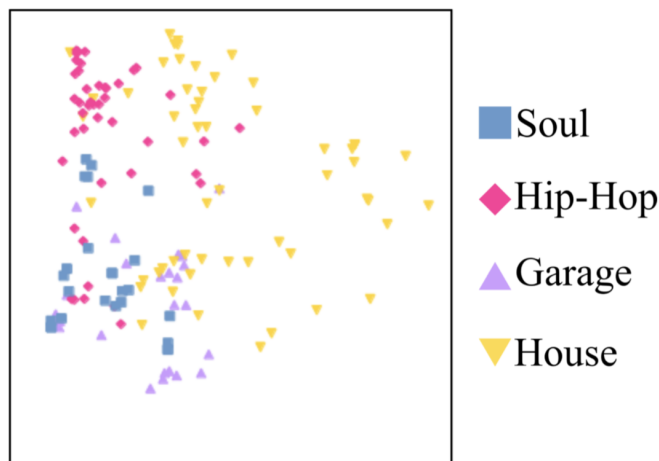


Figure 2.11: A rhythm space obtained from a similarity metric maps out a collection of different patterns and styles [25]

Finally, the authors agree on the fact that automatically representing drum spaces is possible, from a mathematical perspective, in two different ways:

- In the best case scenario, where the developed metrics can achieve a good prediction at polyphonic similarity, the low dimensional rhythm space will directly inherit the essential relations between audio loops that were calculated by the distances.
- In the case where metrics are not fully precise, users could still take advantage of the resulting space as a way of exploring the collection, where only some regions would reveal themselves useful for the analysis at hand.

2.4.4 Visualizing multidimensional musical data

In [14], Nadia Carvalho *et al.* propose and detail a new way to encode, analyse and model Iberian Folk music. The research not only brings awareness to a genre that can get forgotten in the digital age, but creates a tool for the navigation and retrieval of musical content from the database generated.

The research adopts two algorithms for the visualization step:

1. t-distributed Stochastic Neighbor Embedding (t-SNE) is a statistical method proposed by Laurens van der Maaten [76] based on Sam Roweis and Geoffrey Hinton previous work in [34]. This new method reduces high-dimensional data into a two- or three-dimensional map, applying a location in the map to each data point. By using a dimension reduction technique, it is able to compress and model the high-dimensional data where similar data is represented by points close in distance.
2. Uniform Manifold Approximation and Projection (UMAP) [50] uses a very similar process to t-SNE, while assuming that the data received as input is distributed in an uniform way.

In Figure 2.12, Nadia Carvalho *et al.* present the results given by the two methods (t-SNE and UMAP) when applied on the database created for the research. The UMAP provides a more precise way of organizing the clusters when compared to the representation given out by t-SNE. The database also has annotations for the country of origin (Portugal or Spain). UMAP clearly puts each country's songs into precise clusters, while t-SNE presents some fuzzy results, where songs that should be present in the left cluster, get moved to the right cluster.

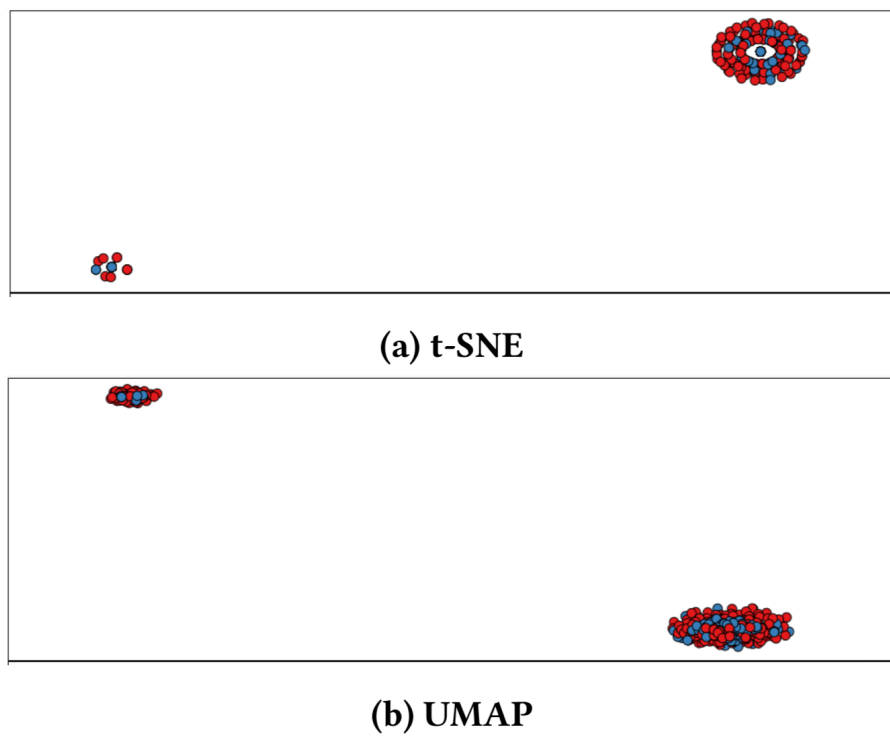


Figure 2.12: t-SNE (a) and UMAP (b) 2D visualizations of the database. Points with closer spatial locations denote songs that share the same country. The points' colors map the respective song according to their Genre (red: Corro, blue: Cuna). [14]

Chapter 3

Understanding Cross-Genre Rhythmic Audio Compatibility

The current chapter is based on a paper [40] which the author co-authored. The contents used on this chapter are solely from the author.

In [12], Nardelli compares the abstraction of musical structures as mathematical objects as being one of the great accomplishments in contemporary music theory and computer music fields. These musical structures range from just simple notes, to complex melodies, chords, harmonic and rhythmic progressions. From this starting point, this project will aim at unveiling metrics for the similarity and compatibility of musical rhythms from musical audio manifestations.

The process for the development of our application will start by understanding rhythmic patterns from previously existing music, through the analysis of multi-track datasets. These datasets by being multi-track have the advantage of including the individual tracks for each instrument separate from each other, which will help us compute similarity distances between each and every track. This calculations will help us develop a prototype for the analysis of rhythmic compatibility to be applied at databases at scale.

In this chapter, we will go through a proposal on understanding rhythmic compatibility when systematically assessing the behaviour of common rhythmic similarity metrics previously explained in Section 2.2.2.2. The metrics are applied to musical examples at scale, which have been composed and performed by real-life musicians and artists.

We also try to unveil levels of rhythmic similarity across multiple instrument families and musical genres, and discuss the conceptual difference between similarity and compatibility, one of the main challenges in repurposing musical audio [26].

Finally, we aim to apply the collected results in supporting creative music composition applications that rely on the analysis of audio content.

In Figure 3.1, we provide an overview of the full system and how we retrieve the audio analysis data from the database. We compute three prominent audio rhythmic similarity metrics - rhythmic patterns (RP) [58, 57, 62, 42, 30], rhythmic histograms (RH) [43, 30] and beat spectrum (BS) [23, 60] - on a multi-track audio dataset of multiple musical genres. The dataset selected for this research is MedleyDB [9], which is a dataset of annotated and royalty-free multi-track recordings curated specifically for the support of research regarding melody and rhythmic extraction. This database includes 122 songs from 8 generic genres ¹

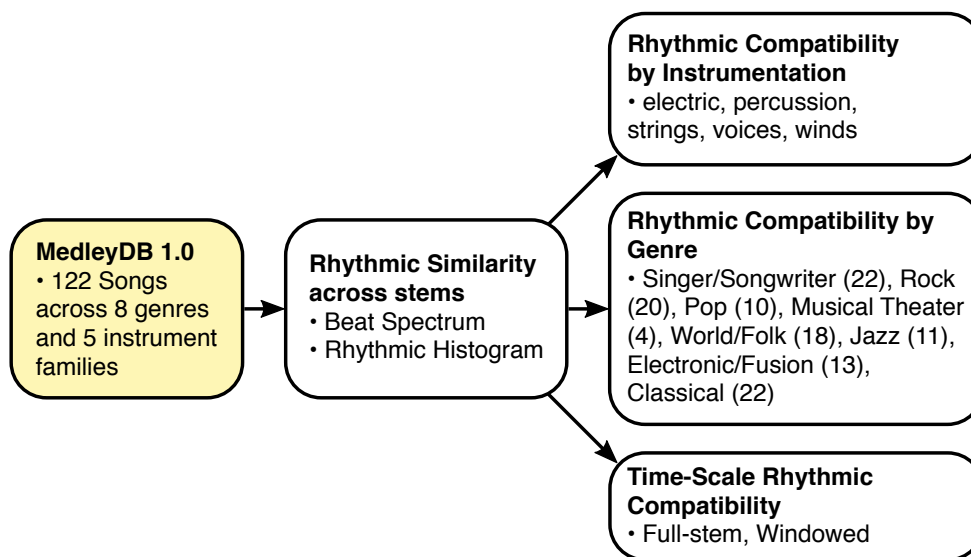


Figure 3.1: Overview of the computation methods [40]

First, we calculate RP, RH, and BS metrics across all stems ² from all songs, and the distances between these metrics for stems per song. Then, we compute global statistic, median and inter-quartile range (IQR), per each distance metric. Moreover, the above similarity metrics are applied throughout the database at two different time scales to inspect the potential impact of the similarity metrics between full songs and short-time windowed analysis. Results ought to unveil the typical values for rhythmic compatibility in professional musical production scenarios.

This chapter is structure as follows. In Section 3.1 we will give a brief explanation of how this study was proceeded. Section 3.2 will present the MedleyDB database in more detail. Finally, Sections 3.3 and 3.4 will detail the rhythmic similarity metrics and describe the computational methods adopted for the statistical analysis.

¹The dataset includes 9 genres in total, but due to the Rap genre having an unbalanced number of 2 examples when compared to other genres (4 to 22 songs), we decided to exclude this genre.

²a subset of one or more raw tracks created during the recording of one instrument

3.1 Assessing rhythmic compatibility

To empirically assess the rhythmic compatibility of the dataset, three different levels of analysis were set:

1. pair-wise instrument family categories
2. musical genre
3. time scale

We then inspect multiple rhythmic similarity metrics across different instrument families and genres. This study unfolds in three main tasks:

1. the computation of rhythmic representations (RP, RH, BS) for each stem
2. the annotation of each dataset stem by an instrument family (electric, percussion, strings, voices, winds)
3. the computation of statistics on rhythmic similarity per instrument family categories and genres at two different time scales (full songs and short windows analysis)

3.2 Dataset

We analyzed 122 songs from the MedleyDB 1.0 database [9] which span across 8 musical genres: classical (22), electronic/fusion (13), jazz (13), musical theater (4), pop (10), rock (20), singer/songwriter (22), world/folk (18).

The dataset includes audio files, metadata and annotations for each song. The audio content includes the raw unprocessed multi-track stems for each individual instruments and a final mix of the song, which is equivalent to its original release version, when all instruments are mixed down into the final track. All audio files are recorded at a 44.1 kHz sampling rate, 16-bit depth in stereo, except for the raw tracks, which were only made available in mono. Metadata files include information per each song such as artist, composer, genre, instrumentation, included files, and if there's any audio bleed or spill occurring throughout the stems. Annotations provide information per track and stem regarding its instrument and pitch content, by including the fundamental frequency (f_0) of the melody. Since this study is analysing rhythmic information, these files were not used.

This study adopts the MedleyDB stems, which the dataset considers as a subset of one or more raw tracks created during recordings for one instrument, e.g., drum set or a multi-mic piano processed into a stereo track after applying effects and panning. In a multi-instrument recording, the occurrence of bleed or spill from other instruments might happen. Stems in the database

which include the recordings of an entire ensemble, labeled as Main System, were disregarded from analysis.

Before the analysis, every stem is mixed to mono and annotated according to its instrumental family, adopting the taxonomy proposed by the MedleyDB team in [8]. The following set of instrumental families were adopted in our analysis: electric (e), percussion (p), strings (s), voices (v), and winds (w). For example, a piano would be categorized under the strings label, while a drum set would be labeled as percussion. The imposed reduction from instrument labels to instrument family aims to reduce the number of conditions under study while loosely assume that instruments that are categorized in the same instrumental family, behave the same and perform the same rhythmic function within a musical structure.

3.3 Rhythmic Similarity Metrics

We adopt three prominent descriptors to represent the rhythmic content of each stem track: RP [57], RH [43] and BS [23] from which similarity is computed. Next, we detail each of these representations, their interpretation, computation, and the distance metric adopt to calculate their similarity.

Rhythmic Histograms [43] represents the rhythmic information of musical audio as amplitude modulations. It derives from Rhythmic Patterns [58, 57, 62, 42], a matrix representation of fluctuations in different frequencies on critical bands of the human’s listening range. Their fundamental difference is that Rhythmic Histograms accumulates all frequency bands onto a single bin, resulting in a vector of 60 frequency modulation bins in the [0, 10] Hz range.³ The main difference between Rhythmic Histograms instead of the most common Rhythmic Patterns is to minimization of timbral or spectral content information in further similarity computation. RH adopts a two-stage extraction process. First, it groups the frequency bands by loudness sensation, using a short-time Fourier transform. The resulting spectral representation is then transformed into a time-invariant 24 critical Bark bands modulation frequency spectrum by applying a second Fourier transform. High amplitudes values in the RH denote a recurrent periodicity in the musical audio. For example, a modulation frequency of 2 Hz indicates the presence of periodicity with 120 BPM [42].

Beat Spectrum is an audio representation for acoustic self-similarity presented in [23], computed from pairwise spectral similarity distances to characterize rhythmic and temporal structure over time. Musical audio with repetitive rhythmic and highly structured content will perform stronger within this analysis, revealing peaks at the time occurrence of repetitions. Peaks in the BS will be aligned with major rhythmic components occurring in the audio signal, inferring that they will correspond to rhythmic periodicity when these peaks are periodic. The BS makes a clear distinction between tempo and rhythm since different rhythm styles occurring at the same

³The human cognition of rhythm ends exists below 15 Hz, roughly equivalent to 900 BPM. To this end, Rhythmic Histograms only considers frequencies up to 10 Hz [42].

tempo will perform differently. The BS is computed from the audio signal in three steps. First, we parametrize the audio content by a short-window spectral analysis (e.g., using mel-frequency cepstral coefficients). Second, we compute a (squared) distance matrix of all pairwise distances across the musical structure spectral representations. Third, we find the BS periodicities in the resulting matrix by diagonal sums or auto-correlation.

We adopt cosine distance to compute rhythmic similarity of RH and BS representations, thus disregarding their weight or magnitude and rather capturing the peak alignments in both functions, while RP uses the euclidean distance.

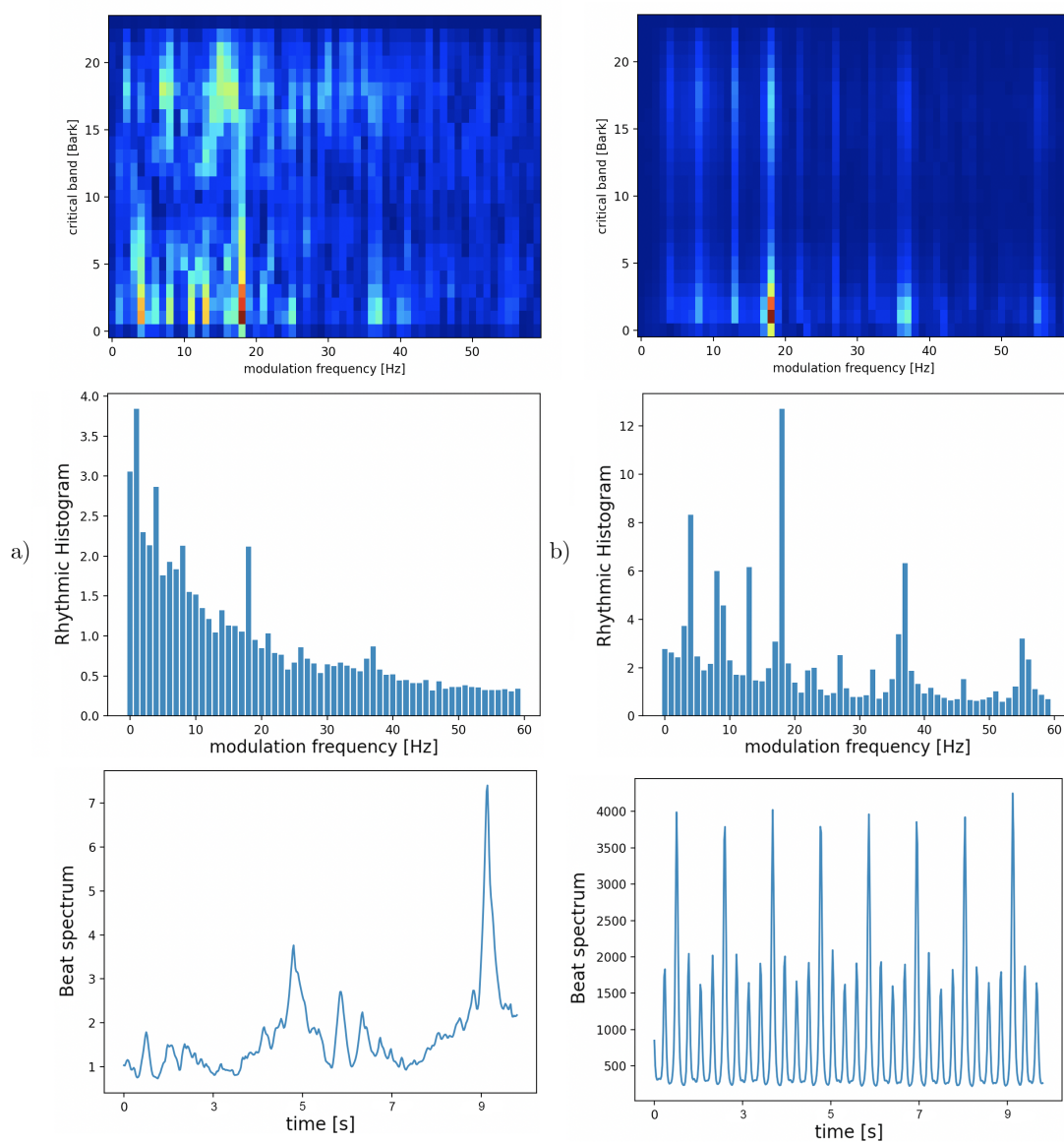


Figure 3.2: Respective Rhythmic Patterns, Rhythmic Histogram and Beat Spectrum representations for the first ten seconds of a) a voice stem and b) a percussion stem from the pop song 'Bounty' by Steven Clark

In figure 3.2, we show the RP, RH and BS for the first ten seconds of two different stems from Steven Clark’s ‘Bounty.’ The stem a) is vocal, and stem b) is percussive. The RP and RH representations for the percussive stem indicate clear peaks at frequencies of 18 and 36 Hz, whereas the vocal stem denotes less rhythmic periodicity content, concentrating most of its energy at lower modulation frequencies. In the BS representation, we note the equal-spacing peaks in the percussive stem plot at both multiples of the tempo. In the vocal BS plot, we cannot infer any structural repetitions across time, as no equal-spacing peaks exist.

3.4 Computational Methods

We computed the RP, RH and BS for all analysed stems of the Medley DB 1.0 database using the Rhythm Pattern Audio Feature Extractor [30] and REPET [60] libraries, respectively. Two time-scale analysis were applied to compute the RP, RH and BS representations per each stem: full stem and windowed stem. Full stem comprises one unique vector describing the rhythmic content across the duration of the entire stem. Windowed stem comprises of an array of different analysis vectors per stem, resulting from consecutive sequences of $2^{18} = 262144$ samples⁴, roughly about 6 seconds long.

Similarity values across each stem per song are then computed using cosine (RH, BS) and euclidean (RP) distances at the two time-scales under analysis. Full stem similarity results in a single similarity value per pairwise stem comparison, and windowed stem similarity result is a list of similarity values per pairwise stem comparison.

To assess the rhythmic similarity across instrument families per genre, we compute typical values from the resulting rhythmic similarity and adopt global statistic indicators. Median and inter-quartile range (IQR)⁵ are adopted for full stem statistics. It results in distances matrices for each RP, RH and BS across different instrument families per genre. A total of 8 genres * 2 similarity metrics * 2 statistical indicators = 48 matrices are computed. Figure 3.3 is an example matrix computed for the Pop genre using the Rhythmic Histograms metric. The top row represent the inter-quartile range distances while the bottom row represents the median distances. In the left column, the full stem analysis are displayed while the right column show the windowed stem analysis. Higher median values denote a smaller degree of stem similarity across or within a certain instrumental family. It should be noted that more than one stem per instrument family can exist in one song, therefore distances between similar families can result in values above zero. Higher IQR values denote a greater degree of similarity dispersion, roughly meaning a greater degree of rhythmic interaction modalities.

By inspecting the degree of rhythmic similarity per genre, irrespective of each instrumental family, we are able to aggregate all similarity per full stem genre and apply the non-paired and non-parametric Mann-Whitney U test. The result p values inform the statistical significance of

⁴a sample rate of 44.1 kHz is adopted throughout this whole study

⁵the difference between the 25th and the 75th percentiles of the sequence of values

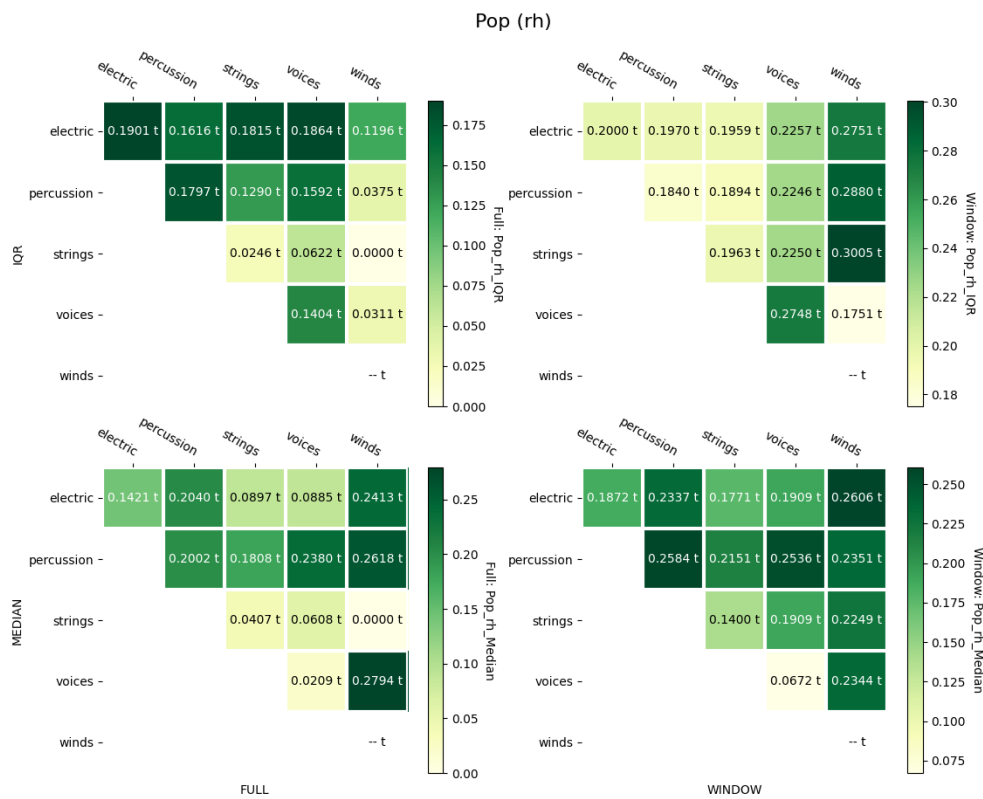


Figure 3.3: Matrix with the Rhythmic Histograms results in the Pop genre

the central tendency difference between the two sets of similarity values. The same statistical test is adopted to understand the impact of time scale in the resulting rhythmic similarity metrics across the full stem and windowed stem. Ultimately, the results of the latter ought to inform us of the importance and diversity across multiple hierarchies. To be able to reject the null hypothesis, the value of $p < 0.05$ and $p < 0.01$ are adopted to denote statistically significance and highly significant significance, respectively.

3.5 Data and results

The statistical analysis across instrumental family per genre produces a large collection of data, and for that reason we attach this complementary material in appendices B, C, and D.

Across all genres, we first identified typical range values of pairwise similarities, which ought to characterize the minimum and maximum values of rhythmic compatibility for the RP, RH and BS similarity. To avoid outliers, minimum and maximum correspond to the 25th and 75th percentiles, respectively. RP values range from 0.002 (classical genre, between electric and winds families) to 3.862 (musical theater, percussion and voices), RH values range from 0.002 (classical, between electric and winds) to 0.354 (musical-theater, percussion and voices) and BS values

range from 0.1 (jazz, electric and electric) to 0.881 (classical, percussion and voices).

Genre	Beat Spectrum				Rhythmic Histogram				Rhythmic Patterns			
	Min.	Instr.	Max.	Instr.	Min.	Instr.	Max.	Instr.	Min.	Instr.	Max.	Instr.
Classical	0,172	w-w	0,881	p-v	0,002	e-w	0,277	e-p	0,002	e-w	0,538	e-s
Electronic/Fusion	0,287	p-p	0,404	e-e	0,038	p-p	0,345	s-s	0,501	p-p	1,495	e-e
Jazz	0,100	e-e	0,542	p-w	0,023	w-w	0,278	v-w	0,769	e-s	1,268	e-w
Musical	0,172	p-p	0,572	p-v	0,060	s-v	0,354	p-v	1,003	s-v	3,862	p-v
Pop	0,191	s-s	0,541	p-p	0,021	v-v	0,279	v-w	0,381	e-s	0,976	p-p
Rock	0,220	e-s	0,492	s-v	0,098	e-e	0,173	p-v	0,539	e-s	1,522	p-p
Singer/Songwriter	0,222	v-w	0,462	p-v	0,056	v-v	0,254	v-w	0,380	e-w	0,860	p-p
World/Folk	0,128	s-s	0,638	p-p	0,005	v-v	0,195	e-e	0,402	e-e	1,424	p-w

Table 3.1: Scope of rhythmic compatibility between genres, expressed by the minimum and maximum median ranges and respective pairwise instrument family for full windowed analysis.

Genre	Beat Spectrum				Rhythmic Histogram				Rhythmic Patterns			
	Min.	Instr.	Max.	Instr.	Min.	Instr.	Max.	Instr.	Min.	Instr.	Max.	Instr.
Classical	0.178	p-p	0.448	e-e	0.032	e-e	0.260	e-p	0.011	e-e	0.721	e-s
Electronic/Fusion	0.310	p-p	0.403	p-s	0.153	p-p	0.353	s-s	1.006	s-s	2.164	p-s
Jazz	0.186	s-s	0.536	p-w	0.066	s-s	0.231	p-w	1.074	s-v	1.697	e-p
Musical	0.121	v-v	0.501	p-v	0.152	p-p	0.464	p-v	1.719	s-v	4.441	p-v
Pop	0.125	s-s	0.476	p-p	0.067	v-v	0.261	e-w	0.552	s-w	1.100	v-w
Rock	0.082	e-s	0.395	p-p	0.055	v-v	0.250	p-v	0.799	e-s	1.746	p-s
Singer/Songwriter	0.117	e-e	0.430	p-p	0.121	v-v	0.242	p-v	0.612	e-e	1.166	w-w
World/Folk	0.074	v-v	0.592	p-w	0.032	v-v	0.253	p-w	0.456	e-w	2.033	p-w

Table 3.2: Scope of rhythmic compatibility between genres, expressed by the minimum and maximum median ranges and respective pairwise instrument family for windowed stems analysis

Genre	Beat Spectrum				Rhythmic Histogram				Rhythmic Patterns			
	Min.	Instr.	Max.	Instr.	Min.	Instr.	Max.	Instr.	Min.	Instr.	Max.	Instr.
Classical	0.020	p-v	0.482	p-p	0.045	e-e	0.403	e-p	0.015	e-p	0.765	p-w
Electronic/Fusion	0.034	s-w	0.307	e-e	0.056	p-s	0.248	e-e	0.089	p-w	1.177	e-w
Jazz	0.062	e-e	0.284	w-w	0.018	w-w	0.266	s-s	0.187	w-w	0.779	s-s
Musical	0.087	s-v	0.188	e-p	0.087	e-e	0.229	e-p	0.172	p-v	0.963	e-v
Pop	0.035	v-w	0.350	e-e	0.025	s-s	0.190	e-e	0.245	s-s	0.897	e-p
Rock	0.160	v-v	0.333	e-p	0.093	p-s	0.227	s-v	0.202	e-s	1.495	e-p
Singer/Songwriter	0.002	w-w	0.536	p-w	0.028	w-w	0.316	v-w	0.197	w-w	0.832	v-v
World/Folk	0.031	v-v	0.600	e-p	0.002	v-v	0.224	e-e	0.133	v-v	1.509	s-w

Table 3.3: Scope of rhythmic compatibility between genres, expressed by the minimum and maximum inter-quartile ranges and respective pairwise instrument family for full windowed analysis

In the upper median range, the percussive instruments family takes the center stage appearing in most of the pairwise comparisons between the remaining instrumental families. On the contrary, in the lower median range, we predominantly find electric, strings, and voice families. The IQR results from the BS analysis from the BS analysis reveal more dispersion across all genres by the percussion and electric families, whilst voices, winds, and strings are predominantly towards the lower range. The RP and RH results show greater variance in IQR than BS.

Genre	Beat Spectrum				Rhythmic Histogram				Rhythmic Patterns			
	Min.	Instr.	Max.	Instr.	Min.	Instr.	Max.	Instr.	Min.	Instr.	Max.	Instr.
Classical	0.250	s-v	0.415	e-w	0.180	p-v	0.478	e-p	0.462	p-v	1.290	e-s
Electronic/Fusion	0.174	s-s	0.307	p-p	0.128	p-p	0.244	s-s	0.239	s-s	1.549	e-e
Jazz	0.153	p-v	0.310	p-s	0.059	e-s	0.254	v-w	0.502	e-e	1.260	p-v
Musical	0.095	p-p	0.269	s-v	0.079	v-v	0.278	e-p	0.261	p-p	1.692	v-v
Pop	0.220	s-s	0.493	s-w	0.175	v-w	0.300	s-w	0.291	s-s	1.956	p-w
Rock	0.196	e-s	0.418	s-v	0.130	p-s	0.247	v-v	0.431	e-s	1.640	v-v
Singer/Songwriter	0.218	e-e	0.480	p-w	0.096	s-s	0.312	v-w	0.542	e-e	1.505	v-w
World/Folk	0.120	s-s	0.524	e-e	0.021	v-v	0.345	e-w	0.265	v-v	1.967	p-p

Table 3.4: Scope of rhythmic compatibility between genres, expressed by the minimum and maximum inter-quartile ranges and respective pairwise instrument family for windowed stem analysis

Tables 3.1, 3.2, 3.3 and 3.4 summarize the minimum and maximum median and inter-quartile range values, respectively, per full or windowed analysis, and also per genre.

Figures 3.4, 3.5 and 3.6 show the statistical significance (p values from the Mann-Whitney U test) for genre and time scale comparisons. A single (*) and double (**) asterisk symbol denote statistical significance and highly statistical significance. From the resulting analysis, we verify that RH can better discriminate genres using rhythmic patterns from pairwise stem differences, as it presents highly statistical significant results for most paired genres. Overall, the time scale analysis between full and windowed stems result in statistical significant across all pairwise comparisons. Therefore, the time scale in rhythmic similarity implies important differences in the analysis.

It's important to demark the fact that for RH and BS, the cosine distance is adopted, and no magnitude differences are considered. Two noticeable exceptions occur for the Musical and Rock genres in the RH and BS, respectively.

The first steps taken during this data analysis are important towards the understanding of rhythmic compatibility from the empirical data generated. Typical rhythmic compatibility values lie in the [0.002, 0.354] range for Rhythmic Histograms, [0.002, 3.862] for Rhythmic Patterns and [0.1, 0.881] for Beat Spectrum.

We are able to verify that RH outperforms both RP and BS in discriminating genres and that rhythmic compatibility using RP, RH and BS distances is statistically significant for different time scale analysis (full song and short windows).

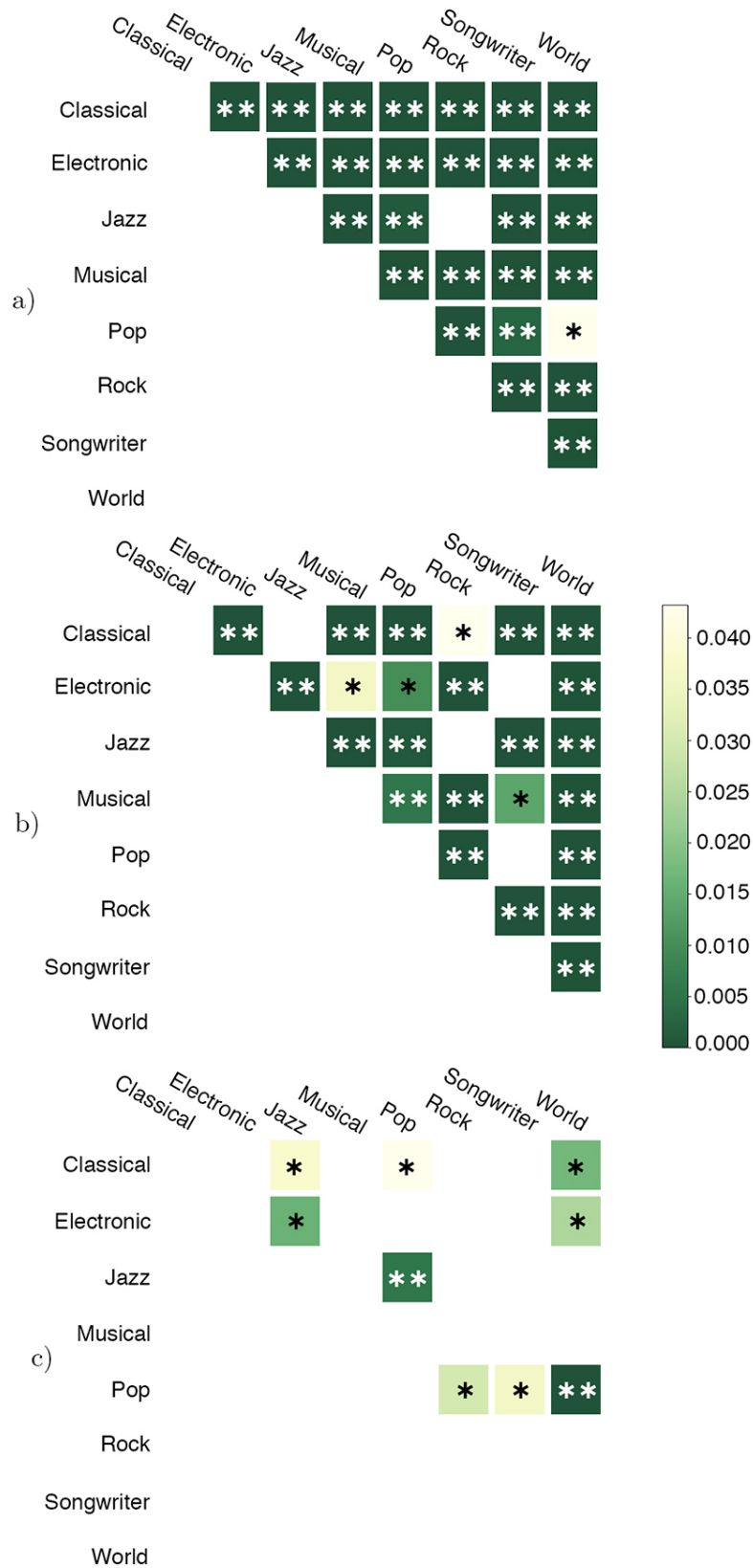


Figure 3.4: Statistical significance between a) RP, b) RH and c) BS full-stem analysis for pairwise genre rhythmic similarity. Color indicates the p value and a single (*) and double (**) asterisk symbols denote statistical significance and highly statistical significance.

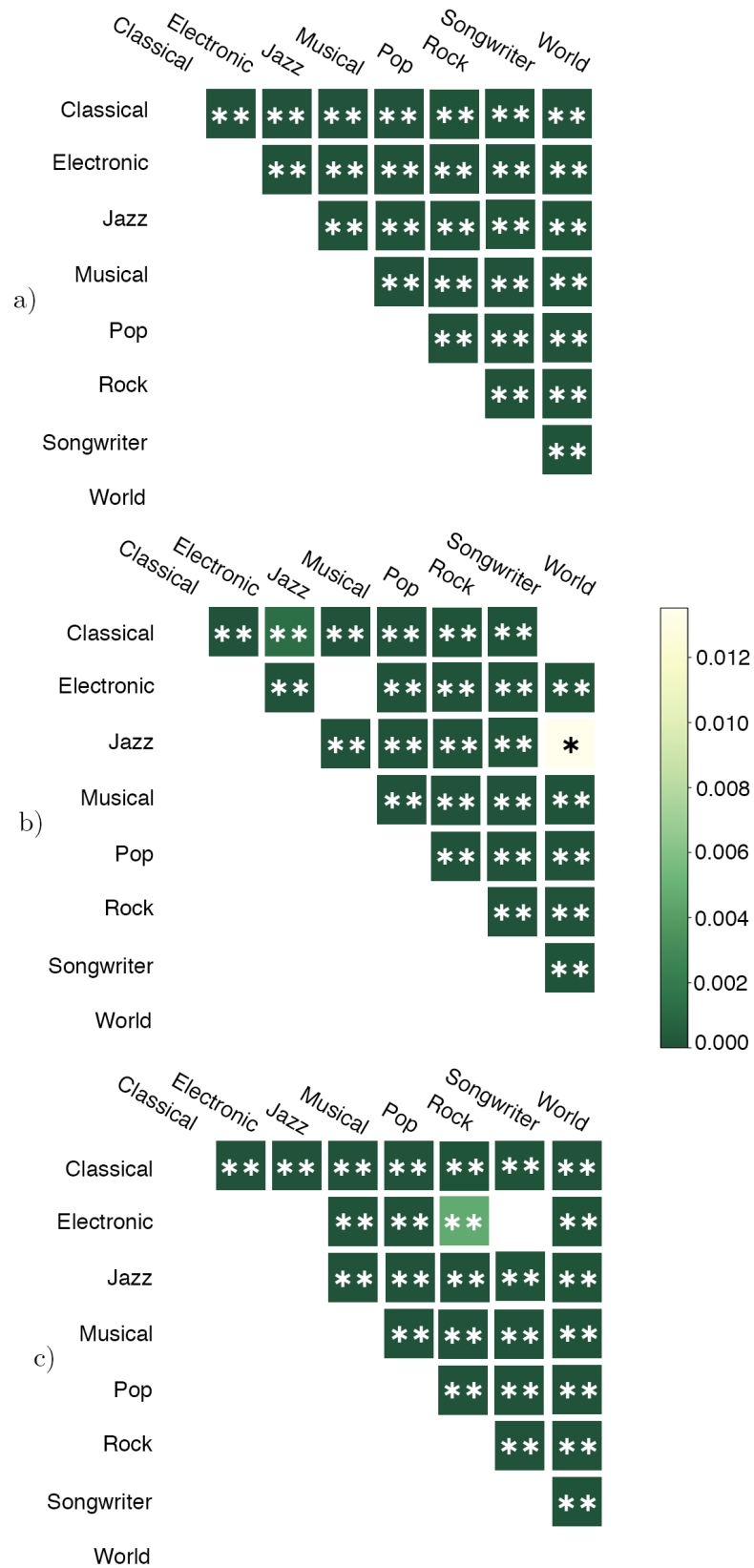


Figure 3.5: Statistical significance between a) RP, b) RH and c) BS windowed stem analysis for pairwise genre rhythmic similarity. Color indicates the p value and a single (*) and double (**) asterisk symbols denote statistic significance and highly statistical significance.

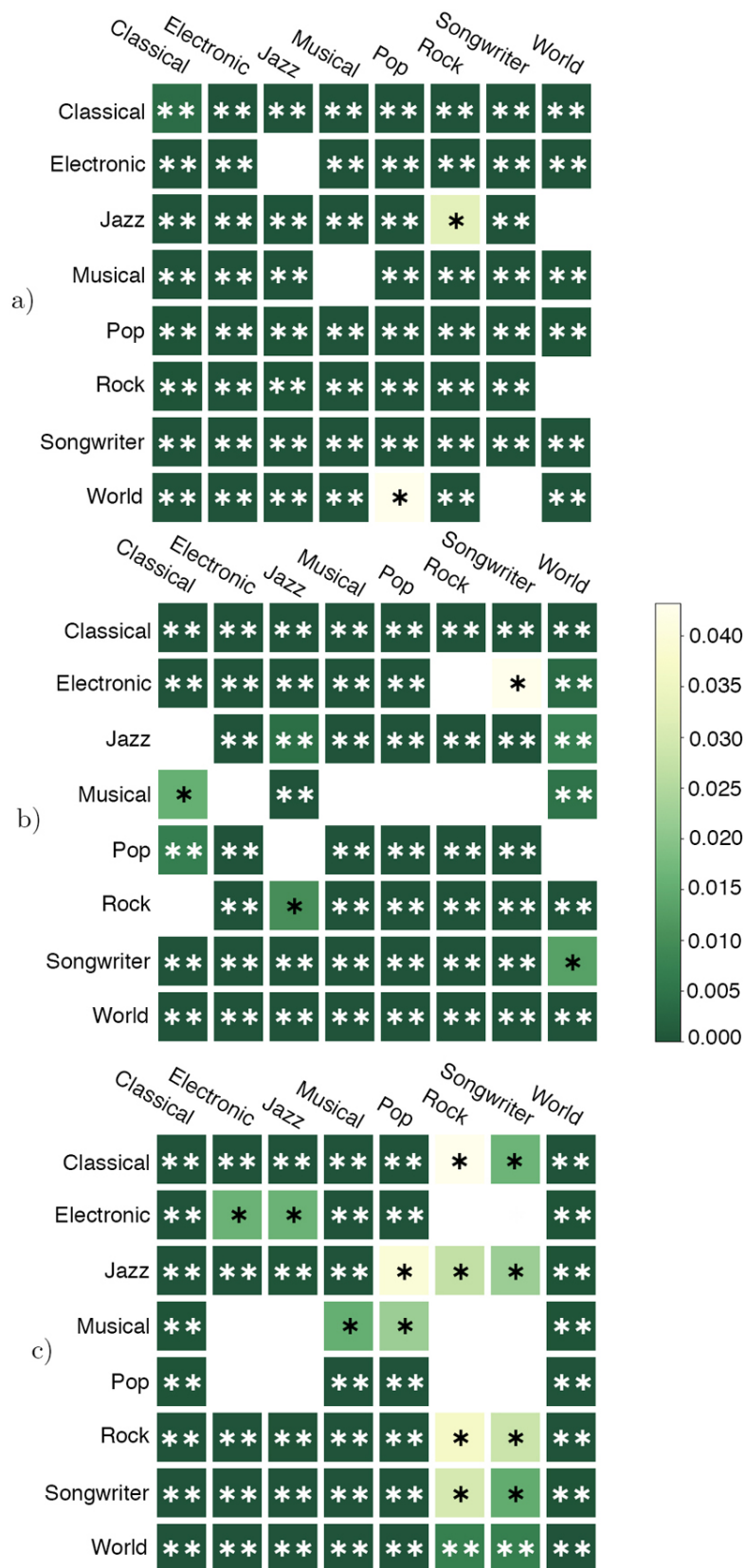


Figure 3.6: Statistical significance between a) RP, b) RH and c) BS time scale analysis (full versus windowed) and genre. Full-stem are in the horizontal axis and windowed in the vertical axis. Color indicates the p value and a single (*) and double (**) asterisk symbols denote statistic significance and highly statistical significance.

Chapter 4

Mapping rhythmic information in a 2D topology space

In this section, we are going to explore the prototype developed for mapping rhythmic information in a two-dimensional space. The main overview of the prototype's workflow is as described in Figure 4.1. We start by feeding the application audio loop databases, which then get the RH computed for all the audio files included in it. Finally, the RH data is visualized in a two-dimensional plot provided by the UMAP dimension reduction algorithm.

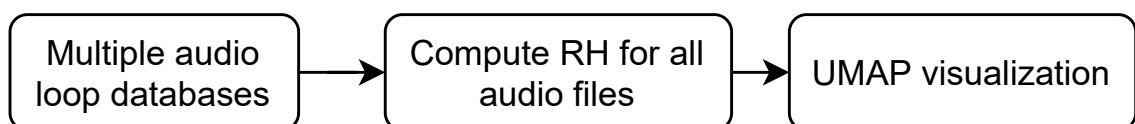


Figure 4.1: Overview of the developed prototype.

4.1 Prototype for rhythmic compatibility retrieval

Our prototype was developed in a Python environment to take use of plotting and visualization libraries such as Matplotlib and UMAP, respectively.

The prototype initiates by opening a window where the user can drag-and-drop first-level directories and audio files, as seen in Figure 4.2. When prompted, the prototype will go through all the directories and files added to the list, to try and find all the WAV audio files available. Only WAV files are selected since it is the only audio formatted accepted by the REPET library [60].

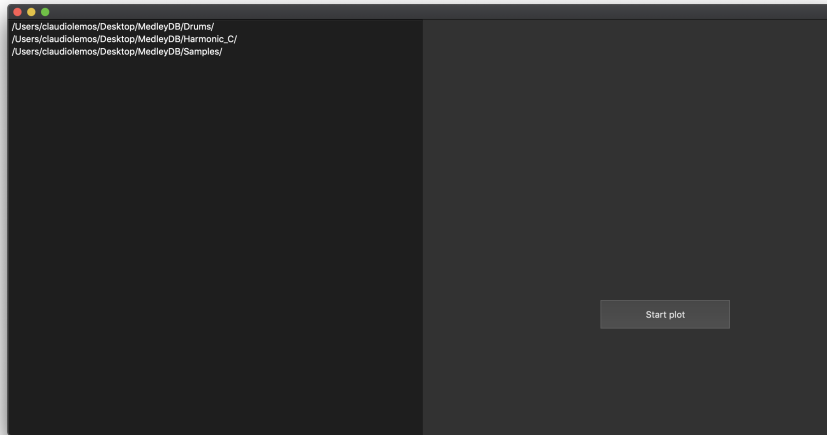


Figure 4.2: Prototype UI where the user can drag and drop folders and files containing audio loops.

Based on the conclusions reached in Chapter 3 and its 24 dimensions ¹, we decided to only use the RH metric for the two-dimensional visual representation of databases.

For our prototypes tests, we used the following audio loop databases:

1. Harmonic C is a database composed of various instrumental loops in the key of C. The loops fall under different BPM and can run from orchestral synth stabs to a retro synth arpeggio.
2. Drums is composed of simple and complex rhythmic drum loops playing at 120BPM ²
3. Samples is an assorted library composed by instrumental loops in different keys and BPM

In all the databases mentioned, the loops are very short, where the longest loop can time at around 15 seconds. The databases audio loop sizes are 202, 170 and 33 loops respectively.

For each loop, the Rhythmic Histograms are calculated and stored in an array that is given as input data. Besides the data, UMAP also receives four parameters [49]:

1. *n_neighbors*, controls the way the algorithm balances local data structure versus global data structure, by constraining the size of the neighborhood UMAP will look at when detecting data patterns.
2. *min_dist*, will effect how tight can nearby points be packed together, and the minimum threshold distance set for that effect. Lower values on this parameter will lead to the creation of more clusters.
3. *n_components* allows to user to select the dimension space of the reduction algorithm.

¹The 24 dimensions of the RH are already reduced from the RP's 24x60 matrix

²corresponding to a frequency of 2Hz

4. *metric*, the last parameter is to set the distance metric will be using when analysing the provided data. UMAP allows a variety of distance metrics such as euclidean, manhattan, cosine, correlation, and hamming.

For our data, we did some experiences and found the following parameter values to give the best results:

$$n_neighbors = 10, min_dist = 0.01, n_components = 2, metric = cosine$$

Figures 4.3, 4.4, 4.5, and 4.6 present results given by the UMAP algorithm when given different combinations of audio loop databases. In Figure 4.3, we added a dotted circle to represent the typical rhythmic distances as discovered in Chapter 3. Dots representing individual audio loops inside a circle would be perceived as rhythmic compatible.

4.2 Topological visualisation

After the prototype was ready, we were able to start testing using the three databases mentioned in Section 4.1. We decided to use four different combinations of input data to help differentiate and understand the results given by UMAP.

The first combination was using all the three databases (harmonic, drums, and samples) and the results can be seen in Figure 4.3. By analysing the image, we cannot detect immediate clusters appearing, besides the fuzzy cluster showing up at the bottom of the plot.

In Figure 4.4, we remove the Samples database, which could have been adding noise to the other two databases, for the fact that it is composed of miscellaneous audio loops not in any particular key or BPM. The results given by UMAP start showing in fact more detailed clusters, but still unorganized and fuzzy.

It is only when we isolate the databases, harmonic and drums, in Figures 4.5 and 4.6 respectively, when we can detect in fact more clearer and defined clusters.

Figure 4.5 shows the results for the Harmonic database only, and as we can confirm by looking at the plot, isolating the audio samples to a specific source does in fact give better results. The clusters for the harmonic loops are much more detailed than the previous two figures (4.3 and 4.4) and even though the clusters may appear a bit fuzzy, we can start retrieving rhythmic patterns just by looking at the plot.

Finally, Figure 4.6, displaying the results for the drums database, gives out the best results. We can detect fully clear and well defined clusters throughout the whole plot, without any noticeable presence of noise.

The results for the Samples database were not presented here due to the small size of it. By analysing the figures, we can reach the conclusion that UMAP visualization results are clearer when the databases given as input are isolated.

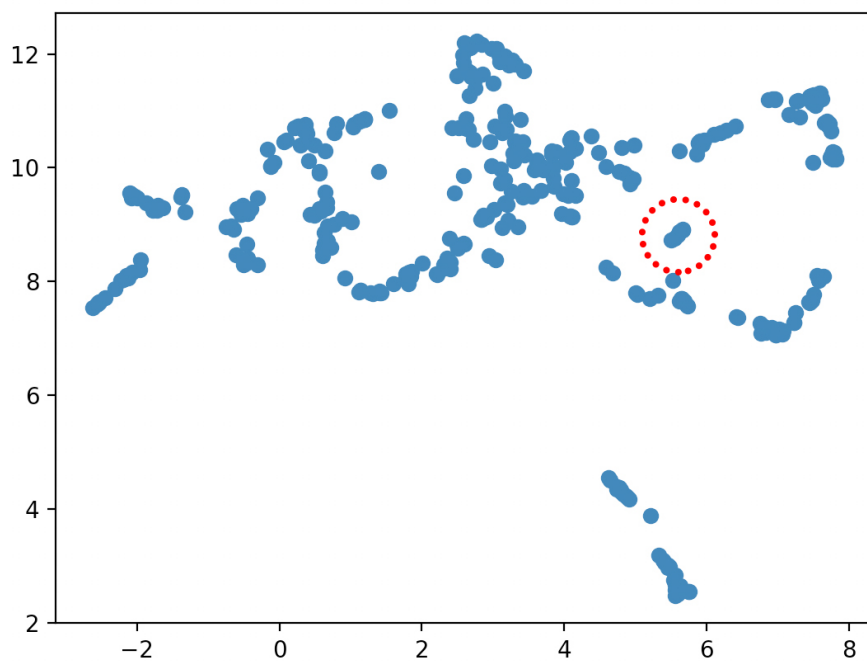


Figure 4.3: UMAP results for all databases (harmonic, drums and samples)

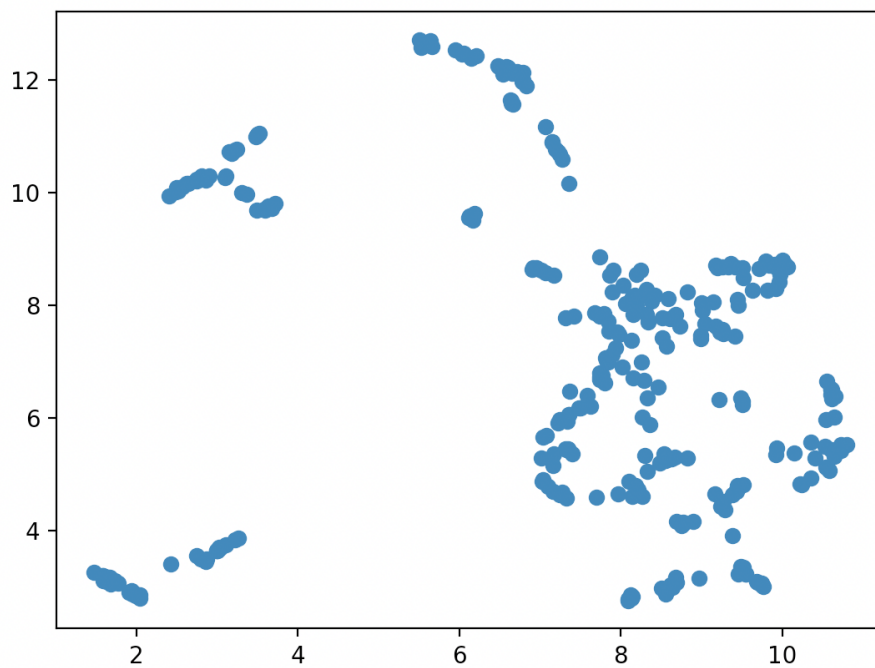


Figure 4.4: UMAP results for harmonic and drums databases

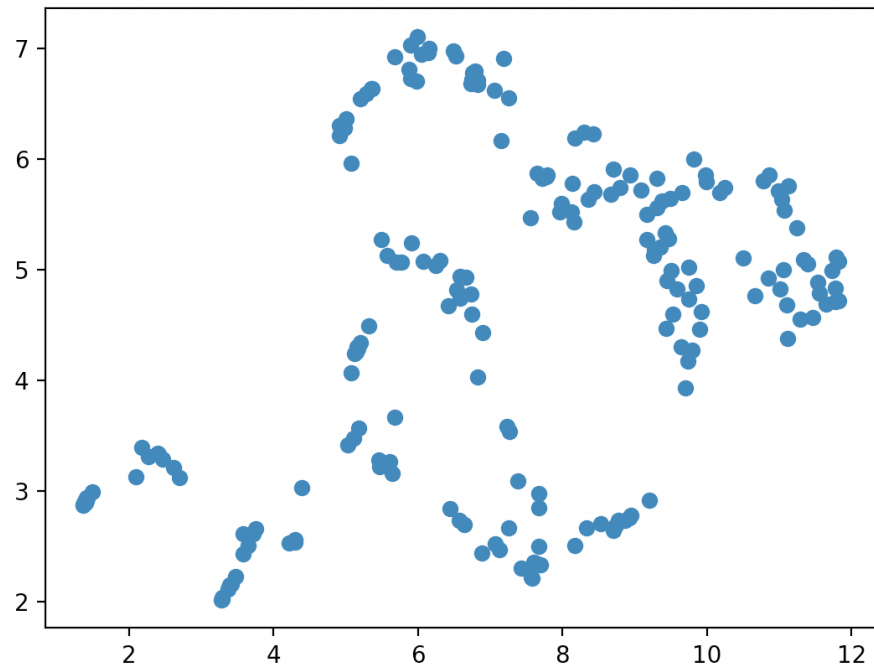


Figure 4.5: UMAP results for harmonic database

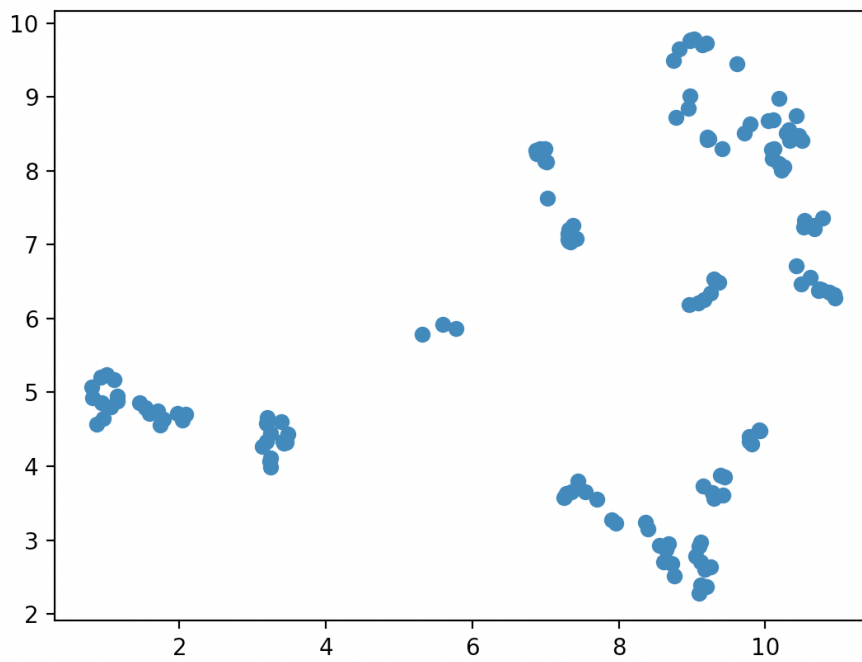


Figure 4.6: UMAP results for drums database

Chapter 5

Conclusions

5.1 Summary

In this dissertation, we present a prototype for the visualization and navigation of multidimensional rhythmic data in a two-dimensional topology space.

This application was developed using the conclusions reached in Chapter 3 by analysing the MedleyDB database and understanding common rhythmic distances in mainstream genres.

This prototype will serve as the base work for a more complex plugin which we will describe in Section 5.3.

5.2 Contributions

To be able to achieve the work developed throughout this dissertation, the main contributions are as follows:

- *MedleyDB* is a multi-track dataset of songs performed by professional musicians. It was a fundamental resource for the progress of this dissertation, as its annotated database helped understand normal rhythmic distances in mainstream genres.
- *REpeating Pattern Extraction Technique (REPET)*, a simple and fast Python library for extracting the repeating background from the non-repeating foreground in an audio mix. While this is the main purpose of this library, we only used the function that computes the Beat Spectrum (BS) on an audio file.
- *Rhythm Pattern* music feature extractor Python library to extract semantic features from audio files. We used the Rhythmic Patterns (RP) and Rhythmic Histograms (RH) extractor functions, and the plot functions for both RP and RH.

- *Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP-Learn)* is a Python library that takes advantage of the UMAP algorithm. We use UMAP-Learn on the development phase of our prototype to cluster and visualize the RH data from the user's audio loop databases.

5.3 Future Work

The work developed throughout this dissertation is intended to be ground zero for a variety of applications intended to be used for creative output. The main idea would be the creation of a Max for Live device ¹(Figure 5.1) for direct interaction with the Ableton Live DAW.

Using this plugin, artists and producers could interact directly with their personal audio loop databases without leaving the production environment they are already familiarized with.

With the Max for Live device ready, this could be further exported into other DAWs such as Logic Pro and ProTools, by turning the plugin into a VST, in order to be compatible with every DAW available.

¹A Max 4 Live device is a plugin developed in the Cycling Max environment and is exclusively compatible with Ableton Live

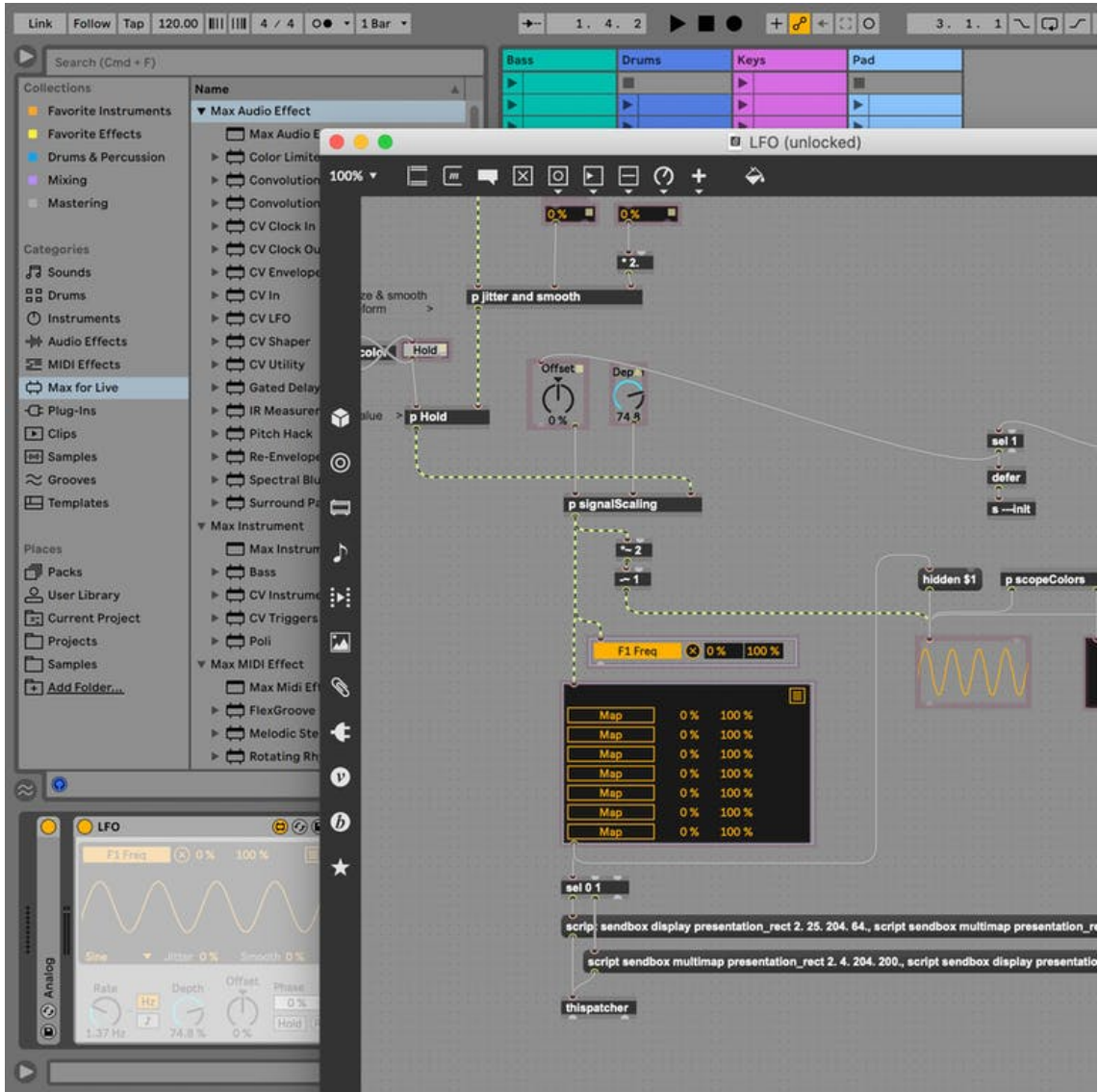


Figure 5.1: Example Max for Live device interacting with Ableton Live. [1]

Appendix A

Code

```
1 import numpy as np
2 import scipy
3 import repet
4 import rp
5 import rp_plot
6 import os
7 import yaml
8 import matplotlib.pyplot as plt
9 import matplotlib.ticker as tckr
10 from matplotlib.ticker import StrMethodFormatter
11 import csv
12 import glob
13 import plotly.graph_objects as go
14 import plotly.offline as pyo
15 import umap.umap_ as umap
16 import umap as umap_plt
17 import sys, os
18 from PyQt5.QtWidgets import QApplication, QMainWindow, QListWidget, QListWidgetItem
19                                     , QPushButton
20
21 from PyQt5.QtCore import Qt, QUrl
22
23 MEDLEY_DB = '~/Desktop/MedleyDB/'
24 AUDIO = 'Audio/'
25 METADATA = 'Metadata/'
26 SAMPLES = 'Samples/'
27 DATA = 'Data/'
28 FEATURES = 'Features/'
29 DISTANCES = 'Distances/'
30 HEATMAPS = 'Heat Maps/'
31 CSVs = 'CSVs/'
32 MATRICES = 'Matrices/'
33 SORTED = 'Sorted/'
34 NPZ = '.npz'
```

```

33 PNG = '.png'
34 CSV = '.csv'
35 TXT = '.txt'
36 TAXONOMY = 'taxonomy.yaml'
37 SAMPLING_WINDOW = pow(2,18)
38 FAMILIES = ['electric', 'percussion', 'strings', 'voices', 'winds']
39 GENRES = ['Classical', 'Electronic/Fusion', 'Jazz', 'Musical Theatre', 'Pop', 'Rap'
           , 'Rock', 'Singer/Songwriter', 'World/
           Folk']

40 RP = 'rp'
41 RH = 'rh'
42 BS = 'bs'
43 FULL = 'full'
44 WINDOW = 'window'
45 IQR = 'iqr'
46 MEDIAN = 'median'
47
48 def get_songs(genre):
49     '''
50         Gets list of songs in MedleyDB database
51         Returns:
52             list (arr): List of MedleyDB's songs
53     '''
54
55     list = os.listdir(os.path.expanduser(f'{MEDLEY_DB}{AUDIO}'))
56     list = [song for song in list if os.path.isdir(os.path.expanduser(f'{MEDLEY_DB}
57                                     {AUDIO}{song}'))]
58     list = [song for song in list if get_song_genre(song) == genre]
59     list = sorted(list)
60
61     return list
62
63 def get_metadata_location(song):
64     '''
65         Gets the absolute path of a song's metadata file
66         Parameters:
67             song (str): Song title
68         Returns:
69             path (str): Absolute path of the song's metadata file
70     '''
71
72     path = os.path.expanduser(f'{MEDLEY_DB}{METADATA}{song}_METADATA.yaml')
73
74     return path
75
76 def get_stem_location(song, stem):
77     '''
78         Gets the absolute path of a stem's wav file from a specific song
79         Parameters:

```

```
79         song (str): Song title
80         stem (int): Stem number
81     Returns:
82         path (str): Absolute path of the stem from a song
83     '''
84
85     path = os.path.expanduser(f'{MEDLEY_DB}{AUDIO}{song}/{song}_STEMS/{song}_STEM_{
86                                     stem:02d}.wav')
87
88     return path
89
90 def get_song_metadata(song):
91     '''
92     Gets the song's complete metadata
93     Parameters:
94         song (str): Song title
95     Returns:
96         data (arr): Complete metadata
97     '''
98     with open(get_metadata_location(song), 'r') as stream:
99         try:
100             data = yaml.safe_load(stream)
101         except yaml.YAMLError as exc:
102             print(exc)
103
104     return data
105
106 def get_song_genre(song):
107     '''
108     Gets the song's genre
109     Parameters:
110         song (str): Song title
111     Returns:
112         genre (str): Song genre
113     '''
114
115     metadata = get_song_metadata(song)
116     genre = metadata['genre']
117
118     return genre
119
120 def get_stem_instrument(metadata, stem):
121     '''
122     Gets the instrument of a stem
123     Parameters:
124         metadata (array): Song metadata
125         stem (int): Stem number
126     Returns:
```

```

127         instrument (int): Stem's instrument
128     '''
129
130     if not isinstance(metadata['stems'][f'S{stem:02d}']['instrument'], list):
131         instrument = metadata['stems'][f'S{stem:02d}']['instrument']
132     else:
133         instrument = metadata['stems'][f'S{stem:02d}']['instrument'][0]
134
135     return instrument
136
137 def get_stems_array(metadata):
138     '''
139         Gets a boolean array that excludes stems that should not be analysed (Main
140                                     System and Unlabeled)
141
142         Parameters:
143             metadata (array): Song metadata
144         Returns:
145             stems (array): Boolean stem array
146     '''
147
148     stems = []
149
150     for i in range(len(metadata['stems'])):
151         instrument = get_stem_instrument(metadata, i+1)
152
153         if instrument != 'Main System' and instrument != 'Unlabeled':
154             stems.append(False)
155         else:
156             stems.append(True)
157
158     return stems
159
160 def get_stem_number(stems_array, stem_index):
161     '''
162         Corrects the index of a stem by skipping over the Main System and Unlabeled
163                                     stems
164
165         Parameters:
166             stems_array (array): Boolean stems array
167             stem_index (int): Current stem index
168         Returns:
169             stem_number (int): New stem number
170     '''
171
172     stem_number = [i for i, n in enumerate(stems_array) if n == False][stem_index]
173                     + 1
174
175     return stem_number
176
177 def load_stems(song, metadata):

```



```

173     '''
174     Loads the stem files of a song
175     Parameters:
176         song (str): Song title
177         metadata (array): Song metadata
178     Returns:
179         audio_signals (arr): Array with the audio signals of the song's
180                               stems
181         sampling_frequencies (arr): Array with the sampling frequencies of
182                               the song's stems
183     '''
184
185     audio_signals, sampling_frequencies = [], []
186
187     for i in range(len(metadata['stems'])):
188         instrument = get_stem_instrument(metadata, i+1)
189
190         if instrument != 'Main System' and instrument != 'Unlabeled':
191             audio_signal, sampling_frequency = repet.wavread(get_stem_location(song
192                                                         , i+1))
193
194             audio_signals.append(audio_signal)
195             sampling_frequencies.append(sampling_frequency)
196
197     return audio_signals, sampling_frequencies
198
199 def get_instruments_family():
200     '''
201     Creates a dictionary that returns an instruments family when called (e.g.
202         taxonomy['kick drum'] = '
203         percussion')
204
205     Returns:
206         taxonomy (dict): MedleyDB taxonomy dictionary
207     '''
208
209     with open(os.path.expanduser(f'{MEDLEY_DB}{TAXONOMY}'), 'r') as stream:
210         try:
211             data = yaml.safe_load(stream)
212         except yaml.YAMLError as exc:
213             print(exc)
214
215     taxonomy = {}
216
217     for i in data:
218         for j in data[i]:
219             taxonomy[j] = i
220
221     return taxonomy
222
223 def calculate_rp_rh(audio_signals, sampling_frequencies):

```

```

217     '''
218     Calculates Rhythmic Patterns and Rhythmic Histogram for a song
219     Parameters:
220         audio_signals (arr): Array with the audio signals of the song's
221                             stems
222         sampling_frequencies (arr): Array with the sampling frequencies of
223                                     the song's stems
224     Returns:
225         rhythmic_patterns_histogram (arr): Array with the calculated
226                                             Rhythmic Patterns and
227                                             Rhythmic Histogram
228     '''
229
230     rhythmic_patterns_histogram = []
231
232     for audio_signal, sampling_frequency in zip(audio_signals, sampling_frequencies
233                                                 ):
234         rhythmic_patterns_histogram.append(rp rp_extract(audio_signal,
235                                                         sampling_frequency, extract_rp =
236                                                         True, extract_rh = True))
237
238     return rhythmic_patterns_histogram
239
240 def calculate_bs(audio_signals, sampling_frequencies):
241     '''
242     Calculates Beat spectrum for a song
243     Parameters:
244         audio_signals (arr): Array with the audio signals of the song's
245                             stems
246         sampling_frequencies (arr): Array with the sampling frequencies of
247                                     the song's stems
248     Returns:
249         beat_spectrum (arr): Array with the calculated Beat Spectrum
250     '''
251
252     beat_spectrum = []
253
254     for audio_signal, sampling_frequency in zip(audio_signals, sampling_frequencies
255                                                 ):
256         beat_spectrum.append(beat_spectrum_fx(audio_signal, sampling_frequency))
257
258     return beat_spectrum
259
260 def plot_bs(beat_spectrum):
261     '''
262     Plots the beat spectrum
263     Parameters:
264         beat_spectrum (arr): Beat spectrum array
265     '''

```

```
256
257     xrange = range(0, beat_spectrum.shape[0])
258     plt.plot(xrange, beat_spectrum)
259     # plot_index = range(0, int(beat_spectrum.shape[0]/44100), 60)
260     # plot_base = np.array(plot_index)
261     # bpm = np.around(plot_base * 60, 0).astype(int) # integer for legend
262     # plt.xticks(plot_index, bpm)
263     plt.ylabel('Beat spectrum')
264     plt.xlabel('time [s]')
265     plt.show()
266     return
267
268 def plot_rp(rp):
269     '''
270         Plots the rhythmic patterns
271         Parameters:
272             rp (arr): Rhythmic Patterns array
273     '''
274
275     rp_plot.plotrp(rp)
276     return
277
278 def plot_rh(rh):
279     '''
280         Plots the rhythmic histogram
281         Parameters:
282             rh (arr): Rhythmic Histogram array
283     '''
284
285     rp_plot.plotrh(rh, False)
286     return
287
288 def beat_spectrum_fx(audio_signal, sampling_frequency):
289     '''
290         Calculates the beat spectrum of an audio file
291         Parameters:
292             audio_signal (int): Audio signal of the wav file
293             sampling_frequency (int): Sampling frequency of the wav file
294         Returns:
295             beat_spectrum (arr): Beat spectrum of the audio file
296     '''
297
298     # get the number of samples and channels in the audio signal
299     number_samples, number_channels = np.shape(audio_signal)
300
301     # set STFT parameters
302     stft_ms = 0.04
303     stft_pow = 2
```

```

304 window_length = pow(stft_pow, int(np.ceil(np.log2(stft_ms * sampling_frequency
305                                     )))
306 window_function = scipy.signal.hamming(window_length, sym=False)
307 step_length = int(window_length / 2)
308
309 # derive the number of time frames
310 number_times = (int(np.ceil(((number_samples + 2 * int(np.floor(window_length /
311                                     2))) - window_length) / step_length)
312                                     ) + 1)
313
314 # initialize the STFT
315 audio_stft = np.zeros((window_length, number_times, number_channels), dtype=
316                       complex)
317
318 # loop over the channels from the signal
319 for i in range(number_channels):
320     # compute the STFT of the current channel
321     audio_stft[:, :, i] = repet._stft(audio_signal[:, i], window_function,
322                                     step_length)
323
324 # derive the magnitude spectrogram with the DC component and without the
325     mirrored frequencies
326 audio_spectrogram = abs(audio_stft[0 : int(window_length / 2) + 1, :, :])
327
328 # compute the beat spectrum of the spectrograms averaged over the channels
329 beat_spectrum = repet._beatspectrum(np.power(np.mean(audio_spectrogram, axis=2)
330                                     , 2))
331
332 return beat_spectrum
333
334 def save_features(song, rhythmic_patterns_histogram, beat_spectrum, window_number =
335                 None):
336     '''
337     Saves features data (rp, rh, bs)
338     Parameters:
339         song (str): Song title
340         rhythmic_patterns_histogram (array): RP and RH array
341         beat_spectrum (array): BS array
342         window_number (int): Window slice index
343     '''
344     if not os.path.exists(os.path.expanduser(f' {MEDLEY_DB} {DATA}')):
345         os.makedirs(os.path.expanduser(f' {MEDLEY_DB} {DATA}'))
346
347     if not os.path.exists(os.path.expanduser(f' {MEDLEY_DB} {DATA} {FEATURES}')):
348         os.makedirs(os.path.expanduser(f' {MEDLEY_DB} {DATA} {FEATURES}'))

```

```

344     if not os.path.exists(os.path.expanduser(f' {MEDLEY_DB}{DATA}{FEATURES}{song}'))
345         :
346         os.makedirs(os.path.expanduser(f' {MEDLEY_DB}{DATA}{FEATURES}{song}'))
347     np.savez_compressed(os.path.expanduser(f' {MEDLEY_DB}{DATA}{FEATURES}{song}/{
348         song}{" " if window_number is None
349         else "_" + str(window_number+1)}'),
350         rp_rh_array = np.array(rhythmic_patterns_histogram, dtype="object"),
351         bs_array = np.array(beat_spectrum, dtype="object")
352     )
353     return
354 def save_distances(genre, rp_full_distances, rp_window_distances, rh_full_distances
355     , rh_window_distances, bs_full_distances,
356     bs_window_distances):
357     '''
358     Saves genres distance data (rp_full, rh_window, rh_full, rh_window, bs_full
359     , bs_window)
360     Parameters:
361     genre (str): Genre
362     rp_full_distances (array): Calculated RP full distances
363     rp_window_distances (array): Calculated RP window distances
364     rh_full_distances (array): Calculated RH full distances
365     rh_window_distances (array): Calculated RH window distances
366     bs_full_distances (array): Calculated BS full distances
367     bs_window_distances (array): Calculated BS window distances
368     '''
369     if not os.path.exists(os.path.expanduser(f' {MEDLEY_DB}{DATA}')):
370         os.makedirs(os.path.expanduser(f' {MEDLEY_DB}{DATA}'))
371     if not os.path.exists(os.path.expanduser(f' {MEDLEY_DB}{DATA}{DISTANCES}')):
372         os.makedirs(os.path.expanduser(f' {MEDLEY_DB}{DATA}{DISTANCES}'))
373     np.savez_compressed(os.path.expanduser(f' {MEDLEY_DB}{DATA}{DISTANCES}{genre.
374         replace("/", "_')}'),
375         rp_full_array = np.array(rp_full_distances, dtype="object"),
376         rp_window_array = np.array(rp_window_distances, dtype="object"),
377         rh_full_array = np.array(rh_full_distances, dtype="object"),
378         rh_window_array = np.array(rh_window_distances, dtype="object"),
379         bs_full_array = np.array(bs_full_distances, dtype="object"),
380         bs_window_array = np.array(bs_window_distances, dtype="object"),
381     )
382     return
383 def analyse_genre(genre):

```

```

386     '''
387     Analyses features for songs of a genre
388     Parameters:
389         genre (str): Genre name
390     '''
391
392     songs = get_songs(genre)
393
394     print(f'Analysing {genre} songs')
395
396     for song in songs:
397         analyse_song(song)
398
399     return
400
401 def analyse_song(song):
402     '''
403     Analyses features for one song (full and window)
404     Parameters:
405         song (str): Song title
406     '''
407
408     metadata = get_song_metadata(song)
409     audio_signals, sampling_frequencies = load_stems(song, metadata)
410
411     print(f'Analysing {song}')
412
413     # Analyse full song
414
415     print(f'Analysing full song')
416     analyse(song, audio_signals, sampling_frequencies)
417
418     # Analyse windows
419
420     parts = int(np.shape(audio_signals)[1]/SAMPLING_WINDOW)
421     audio_signals = np.array(audio_signals)
422
423     for window_number in range(parts):
424         print(f'Analysing window {window_number+1}/{parts}')
425         analyse(song, audio_signals, sampling_frequencies, window_number)
426
427     return
428
429 def analyse(song, audio_signals, sampling_frequencies, window_number = None):
430     '''
431     Analyses features (rp, rh, bs) of audio signals
432     Parameters:
433         song (str): Song title
434         audio_signals (array): Audio signals of a song

```



```

506         rp_rh_array'[j]['rp'
            distance.cosine(
                loaded['rp_rh_array']
                [i]['rh'], loaded['
                rp_rh_array'][j]['rh'
            ]))
507         bs_window_distances[index1][index2].append(scipy.spatial.
            distance.cosine(
                loaded['bs_array'][i]
                , loaded['bs_array'][
                j]))
508
509     save_distances(genre, rp_full_distances, rp_window_distances, rh_full_distances
                    , rh_window_distances,
                    bs_full_distances,
                    bs_window_distances)
510
511     return
512
513 def generate_family_distances_heatmap(genre, feature, show, saveImage, saveCSV):
514     '''
515         Creates distance heatmaps by instrumentation
516         Parameters:
517             genre (str): Genre name
518             feature (str): Feature (RP, RH, BS) name
519             show (boolean): Show plot
520             saveImage (boolean): Save plot as an image
521             saveCSV (boolean): Save matrix as a CSV
522     '''
523
524     loaded = np.load(os.path.expanduser(f'{MEDLEY_DB}{DATA}{DISTANCES}{genre.
                    replace("/", "_")}{NPZ}'),
                    allow_pickle=True)
525
526     full_iqr, full_median = calculate_stat([[k for k in j if k == k] for j in i]
                    for i in loaded[f'{feature}
                    _full_array'])
527     window_iqr, window_median = calculate_stat([[k for k in j if k == k] for j in i]
                    for i in loaded[f'{feature}
                    _window_array'])
528
529     results = [[full_iqr, full_median], [window_iqr, window_median]]
530
531     if show or saveImage:
532         plt.rc('figure', figsize=(13, 10))
533
534         fig, axs = plt.subplots(2, 2)

```

```

535     fig.suptitle(f'{genre} ({feature})', fontsize=16)
536
537     for window_index, window_size in enumerate(['Full', 'Window']):
538         for stat_index, stat in enumerate(['IQR', 'Median']):
539
540             im, cbar = create_heatmap(np.array(results[window_index][stat_index
541                                     ]), FAMILIES, FAMILIES,
542                                     axs[stat_index,
543                                     window_index], cmap="YlGn
544                                     ", cbarlabel=f' {
545                                     window_size}: {genre}_{
546                                     feature}_{stat}')
547
548             texts = annotate_heatmap(im, valfmt="{x:.4f} t")
549
550             axs[0,0].set(ylabel='IQR')
551             axs[1,0].set(xlabel='FULL', ylabel='MEDIAN')
552             axs[1,1].set(xlabel='WINDOW')
553
554     if show:
555         plt.show()
556
557     if saveImage:
558         plt.savefig(os.path.expanduser(f' {MEDLEY_DB}{DATA}{HEATMAPS}{genre.replace
559                                     ("/", "_")}_{feature}{PNG}'))
560
561     if saveCSV:
562         for window_index, window_size in enumerate(['full', 'window']):
563             for stat_index, stat in enumerate(['iqr', 'median']):
564                 with open(os.path.expanduser(f' {MEDLEY_DB}{DATA}{HEATMAPS}{genre.
565                                     replace("/", "_")}_{
566                                     feature}_{window_size}_{
567                                     stat}{CSV}'), 'w',
568                                     newline='') as file:
569
570                     current = results[window_index][stat_index]
571                     writer = csv.writer(file)
572                     writer.writerow(['x', 'electric', 'percussion', 'strings', '
573                                     voices', 'winds'])
574                     writer.writerow(['electric', current[0][0], current[0][1],
575                                     current[0][2],
576                                     current[0][3],
577                                     current[0][4]])
578                     writer.writerow(['percussion', None, current[1][1], current[1][
579                                     2], current[1][3],
580                                     current[1][4]])
581                     writer.writerow(['strings', None, None, current[2][2], current[
582                                     2][3], current[2][4]]
583                                     )

```

```
564         writer.writerow(['voices', None, None, None, current[3][3],
565                           current[3][4]])
566         writer.writerow(['winds', None, None, None, None, current[4][4]
567                           ])
568
569     return
570
571 def calculate_stat(matrix):
572     '''
573     Calculates iqr and median for a matrix
574     Parameters:
575         matrix (arr): Distance matrix
576     Returns:
577         iqr (arr): IQR matrix
578         iqr (arr): Median matrix
579     '''
580     iqr = [[0 for _ in range(5)] for i in range(5)]
581     median = [[0 for _ in range(5)] for i in range(5)]
582
583     for i in range(5):
584
585         for j in range(i, 5):
586             iqr[i][j] = scipy.stats.iqr(matrix[i][j-i])
587             median[i][j] = np.median(matrix[i][j-i])
588
589     iqr = mirror_matrix(iqr)
590     median = mirror_matrix(median)
591
592     return iqr, median
593
594 def mirror_matrix(matrix):
595     '''
596     Mirrors upper triangle to the lower triangle of the matrix
597     Parameters:
598         matrix (arr): Distance matrix
599     Returns:
600         iqr (arr): Mirrored matrix
601     '''
602     matrix_size = len(matrix)
603     mirror = matrix
604
605     for i in range(matrix_size):
606         if i + 1 == matrix_size:
607             break
608
609         for j in range(i+1, matrix_size):
610             mirror[j][i] = mirror[i][j]
```

```

611
612     return mirror
613
614 def get_matrix_indexes(metadata, stems_array, taxonomy, i, j):
615     """
616         Gets the first and second index for the distances by instrumentation matrix
617                                     of two different stems
618
619         Parameters:
620             metadata (arr): Song metadata
621             stems_array (arr): Stems boolean array
622             taxonomy (dict): MedleyDB taxonomy dictionary
623             i (int): stem 1 index
624             j (int): stem 2 index
625
626         Returns:
627             index1 (int): stem 1 matrix index
628             index2 (int): stem 2 matrix index
629     """
630
631     stem1 = get_stem_number(stems_array, i)
632     stem2 = get_stem_number(stems_array, j)
633
634     family1 = taxonomy[get_stem_instrument(metadata, stem1)]
635     family2 = taxonomy[get_stem_instrument(metadata, stem2)]
636
637     index1 = FAMILIES.index(min(family1, family2))
638     index2 = FAMILIES.index(max(family1, family2)) - index1
639
640     return index1, index2
641
642 def create_empty_nested_list(size):
643     """
644         Creates a empty nested list of size size x size
645         Parameters:
646             size (int): List size
647         Returns:
648             list (arr): Created nested list
649     """
650
651     list = [[] for _ in range(size)]
652
653     for i in range(size):
654         list[i] = [[] for _ in range(size-i)]
655
656     return list
657
658 def create_heatmap(data, row_labels, col_labels, ax=None,
659                   cbar_kw={}, cbarlabel="", **kwargs):
660     """
661         Create a heatmap from a numpy array and two lists of labels.

```

```
659     Parameters
660     -----
661     data
662         A 2D numpy array of shape (N, M).
663     row_labels
664         A list or array of length N with the labels for the rows.
665     col_labels
666         A list or array of length M with the labels for the columns.
667     ax
668         A 'matplotlib.axes.Axes' instance to which the heatmap is plotted. If
669         not provided, use current axes or create a new one. Optional.
670     cbar_kw
671         A dictionary with arguments to 'matplotlib.figure.colorbar'. Optional.
672     cbarlabel
673         The label for the colorbar. Optional.
674     **kwargs
675         All other arguments are forwarded to 'imshow'.
676     """
677
678     if not ax:
679         ax = plt.gca()
680
681     # windows
682     mask = np.tri(data.shape[0], k=-1)
683
684     # genres
685     # mask = np.zeros((data.shape[0], data.shape[1]))
686
687     # Additional mask for Mann-Whitney U rank
688     for i in range(data.shape[0]):
689         for j in range(data.shape[1]):
690             if(data[i][j] >= 0.05):
691                 mask[i][j] = 1
692
693     data = np.ma.array(data, mask=mask)
694
695     # Plot the heatmap
696     im = ax.imshow(data, **kwargs)
697
698     # Create colorbar
699     cbar = ax.figure.colorbar(im, ax=ax, **cbar_kw)
700     cbar.ax.set_ylabel(cbarlabel, rotation=-90, va="bottom")
701
702     # We want to show all ticks...
703     ax.set_xticks(np.arange(data.shape[1]))
704     ax.set_yticks(np.arange(data.shape[0]))
705     # ... and label them with the respective list entries.
706     ax.set_xticklabels(col_labels)
707     ax.set_yticklabels(row_labels)
```

```

708
709 # Let the horizontal axes labeling appear on top.
710 ax.tick_params(top=True, bottom=False,
711               labeltop=True, labelbottom=False)
712
713 # Rotate the tick labels and set their alignment.
714 plt.setp(ax.get_xticklabels(), rotation=-30, ha="right",
715         rotation_mode="anchor")
716
717 # Turn spines off and create white grid.
718 for edge, spine in ax.spines.items():
719     spine.set_visible(False)
720
721 ax.set_xticks(np.arange(data.shape[1]+1)-.5, minor=True)
722 ax.set_yticks(np.arange(data.shape[0]+1)-.5, minor=True)
723 ax.grid(which="minor", color="w", linestyle='-', linewidth=3)
724 ax.tick_params(which="minor", bottom=False, left=False)
725
726 return im, cbar
727
728
729 def annotate_heatmap(im, data=None, valfmt="{x:.2f}",
730                   textcolors=["black", "white"],
731                   threshold=None, **textkw):
732     """
733     A function to annotate a heatmap.
734     Parameters
735     -----
736     im
737         The AxesImage to be labeled.
738     data
739         Data used to annotate. If None, the image's data is used. Optional.
740     valfmt
741         The format of the annotations inside the heatmap. This should either
742         use the string format method, e.g. or be a
743         `matplotlib.ticker.Formatter`. Optional.
744     textcolors
745         A list or array of two color specifications. The first is used for
746         values below a threshold, the second for those above. Optional.
747     threshold
748         Value in data units according to which the colors from textcolors are
749         applied. If None (the default) uses the middle of the colormap as
750         separation. Optional.
751     **kwargs
752         All other arguments are forwarded to each call to `text` used to create
753         the text labels.
754     """
755
756     if not isinstance(data, (list, np.ndarray)):

```

```

757         data = im.get_array()
758
759         # Normalize the threshold to the images color range.
760         if threshold is not None:
761             threshold = im.norm(threshold)
762         else:
763             threshold = im.norm(data.max())/2.
764
765         # Set default alignment to center, but allow it to be
766         # overwritten by textkw.
767         kw = dict(horizontalalignment="center",
768                 verticalalignment="center")
769         kw.update(textkw)
770
771         # Get the formatter in case a string is supplied
772         if isinstance(valfmt, str):
773             valfmt = tckr.StrMethodFormatter(valfmt)
774
775         # Loop over the data and create a 'Text' for each "pixel".
776         # Change the text's color depending on the data.
777         texts = []
778         for i in range(data.shape[0]):
779             for j in range(data.shape[1]):
780                 kw.update(color=textcolors[int(im.norm(data[i, j]) < threshold)])
781                 text = im.axes.text(j, i, valfmt(data[i, j], None) if j >= i else '',
782                                     **kw)
783
784                 # text = im.axes.text(j, i, valfmt(data[i, j], None), **kw)
785                 # if not j >= i:
786                 #     text = im.axes.text(j, i, '', **kw)
787                 # if 0 <= data[i, j] < 0.01:
788                 #     text = im.axes.text(j, i, '***', **kw)
789                 # elif 0.01 <= data[i, j] < 0.05:
790                 #     text = im.axes.text(j, i, '*', **kw)
791                 # else:
792                 #     text = im.axes.text(j, i, '', **kw)
793                 texts.append(text)
794
795         return texts
796
797 def generate_genres_p_value_heatmap(feature, window, show = False):
798     '''
799     Creates p values heatmaps by genres for one window size
800     Parameters:
801         feature (str): Feature (RP, RH, BS) name
802         window (str): Window size (full, widnow)
803         show (boolean): Show plot
804     '''
805     distances = [[] for _ in range(len(GENRES))]

```

```

805     results = [[0 for _ in range(len(GENRES))] for i in range(len(GENRES))]
806
807     for genre in range(len(GENRES)):
808         loaded = np.load(os.path.expanduser(f' {MEDLEY_DB}{DATA}{DISTANCES}{GENRES[
809             genre].replace("/", "_")}{NPZ}'),
810             allow_pickle=True)
811
812         loaded = loaded[f' {feature}_{window}_array']
813
814         for i in range(len(loaded)):
815             for j in range(len(loaded[i])):
816                 distances[genre].extend(loaded[i][j])
817
818         for i in range(len(results)):
819             for j in range(len(results[i])-i):
820                 statistic, pvalue = scipy.stats.mannwhitneyu(distances[i], distances[j+
821                     i])
822                 results[i][j+i] = pvalue
823
824     plt.rc('figure', figsize=(13, 10))
825
826     fig, axs = plt.subplots(1)
827     fig.suptitle(f'Mann-Whitney rank ({feature}_{window})', fontsize=16)
828
829     im, cbar = create_heatmap(np.array(results), GENRES, GENRES, axs, cmap="YlGn_r"
830                             )
831
832     texts = annotate_heatmap(im, valfmt="{x:.4f} t")
833
834     if show:
835         plt.show()
836
837     return
838
839 def generate_genres_windows_p_value_heatmap(feature, show = False):
840     '''
841     Creates p values heatmaps by genres for both window sizes
842     Parameters:
843         feature (str): Feature (RP, RH, BS) name
844         show (boolean): Show plot
845     '''
846
847     distances_full = [[] for _ in range(len(GENRES))]
848     distances_window = [[] for _ in range(len(GENRES))]
849     results = [[0 for _ in range(len(GENRES))] for i in range(len(GENRES))]
850
851     for genre in range(len(GENRES)):

```



```

848     loaded = np.load(os.path.expanduser(f' {MEDLEY_DB} {DATA} {DISTANCES} {GENRES [
                                           genre].replace("/", "_")} {NPZ}'),
                       allow_pickle=True)
849
850     loaded_full = loaded[f' {feature}_full_array']
851
852     for i in range(len(loaded_full)):
853         for j in range(len(loaded_full[i])):
854             distances_full[genre].extend(loaded_full[i][j])
855
856
857     loaded_window = loaded[f' {feature}_window_array']
858
859     for i in range(len(loaded_window)):
860         for j in range(len(loaded_window[i])):
861             distances_window[genre].extend(loaded_window[i][j])
862
863     for i in range(len(results)):
864         for j in range(len(results[i])):
865             statistic, pvalue = scipy.stats.mannwhitneyu(distances_full[i],
                                                         distances_window[j])
866
867             results[i][j] = pvalue
868
869     plt.rc('figure', figsize=(13, 10))
870
871     fig, axs = plt.subplots(1)
872     fig.suptitle(f'Mann-Whitney rank ({feature} - full x window)', fontsize=16)
873
874     im, cbar = create_heatmap(np.array(results), GENRES, GENRES, axs, cmap="YlGn_r"
                               )
875
876     texts = annotate_heatmap(im, valfmt="{x:.4f} t")
877
878     if show:
879         plt.show()
880
881     return
882
883 def plot_stem(song, stem, feature):
884     '''
885     Plots the feature of a song stem
886     Parameters:
887         song (str): Song title
888         stem (int): Stem number
889         feature (str): Feature (RP, RH, BS) name
890     '''
891     audio_signal, sampling_frequency = repet.wavread(get_stem_location(song, stem))
892
893     if feature == RP:

```

```

893     rhythmic_patterns = rp.rp_extract(audio_signal, sampling_frequency,
                                       extract_rp = True)
894     plot_rp(rhythmic_patterns['rp'])
895     elif feature == RH:
896         rhythmic_histogram = rp.rp_extract(audio_signal, sampling_frequency,
                                             extract_rh = True)
897         plot_rh(rhythmic_histogram['rh'])
898     elif feature == BS:
899         beat_spectrum = beat_spectrum_fx(audio_signal, sampling_frequency)
900         plot_bs(beat_spectrum)
901
902 def sort_heatmaps(genre, feature, window_size, stat):
903     '''
904         Sorts calculated heatmaps by instrumentation into a new CSV with the
905         following structure [stat_1_value
906         , instrument_1, instrument_1,
907         stat_2_value], where stat_1_value
908         is the value being sorted in
909         ascending order
910
911         Parameters:
912             genre (str): Genre name
913             feature (str): Feature (RP, RH, BS) name
914             window_size (str): Window size (full, window)
915             stat (str): Stat 1 name (iqr, median)
916     '''
917     iqr_array, median_array, data1, data2, results = [], [], [], [], []
918
919     with open(os.path.expanduser(f' {MEDLEY_DB} {DATA} {HEATMAPS} {CSVs} {MATRICES} {
920         genre.replace("/", "_")}_{feature}_{
921         window_size}_{IQR} {CSV}')) as csvfile
922         :
923
924         reader = csv.reader(csvfile)
925         for row in reader:
926             iqr_array.append(row)
927
928     with open(os.path.expanduser(f' {MEDLEY_DB} {DATA} {HEATMAPS} {CSVs} {MATRICES} {
929         genre.replace("/", "_")}_{feature}_{
930         window_size}_{MEDIAN} {CSV}')) as
931         csvfile:
932
933         reader = csv.reader(csvfile)
934         for row in reader:
935             median_array.append(row)
936
937     iqr_array = np.array(iqr_array)[1:,1:]
938     median_array = np.array(median_array)[1:,1:]
939     data1 = iqr_array if stat == IQR else median_array
940     data2 = median_array if stat == IQR else iqr_array
941

```

```

929     for i in range(len(data1)):
930         for j in range(len(data1[i])-i):
931             results.append((data1[i][j+i], FAMILIES[i], FAMILIES[i+j], data2[i][j+i
932                                     ]))
933
934 results = sorted(results, key = lambda x: x[0])
935
936 with open(os.path.expanduser(f' {MEDLEY_DB} {DATA} {HEATMAPS} {CSVs} {SORTED} {genre.
937                                     replace("/", "_")}_{feature}_{
938                                     window_size}_{stat}{CSV}'), 'w',
939                                     newline='') as file:
940
941     writer = csv.writer(file)
942     writer.writerow((stat, 'instrument 1', 'instrument 2', IQR if stat ==
943                                     MEDIAN else MEDIAN))
944
945     for i in results:
946         writer.writerow(i)
947
948     return
949
950 def generate_main_csv():
951     '''
952     Creates a CSV with all genres, features, windows and stats
953     '''
954
955     data = []
956
957     for genre in GENRES:
958         for feature in [RP, RH, BS]:
959             for window in [FULL, WINDOW]:
960                 with open(os.path.expanduser(f' {MEDLEY_DB} {DATA} {HEATMAPS} {CSVs} {
961                                     SORTED} {genre.replace
962                                     ("/", "_")}_{feature}_{
963                                     window}_{IQR}{CSV}')) as
964                                     csvfile:
965
966                     reader = csv.reader(csvfile)
967                     for row in reader:
968                         if row[0] != 'iqr':
969                             row.append(genre)
970                             row.append(feature)
971                             row.append(window)
972                             data.append(row)
973
974     with open(os.path.expanduser(f' {MEDLEY_DB} {DATA} {HEATMAPS} {CSVs} main{CSV}'), 'w
975                                     ', newline='') as file:
976
977         writer = csv.writer(file)
978         writer.writerow(['iqr', 'instrument1', 'instrument2', 'median', 'genre', '
979                                     feature', 'window'])
980
981     for row in data:
982         writer.writerow(row)

```

```
967
968     return
969
970 class ListBoxWidget (QListWidget):
971     def __init__(self, parent=None):
972         super().__init__(parent)
973         self.setAcceptDrops(True)
974         self.resize(600, 600)
975
976     def dragEnterEvent(self, event):
977         if event.mimeData().hasUrls():
978             event.accept()
979         else:
980             event.ignore()
981
982     def dragMoveEvent(self, event):
983         if event.mimeData().hasUrls():
984             event.setDropAction(Qt.CopyAction)
985             event.accept()
986         else:
987             event.ignore()
988
989     def dropEvent(self, event):
990         if event.mimeData().hasUrls():
991             event.setDropAction(Qt.CopyAction)
992             event.accept()
993
994             links = []
995             for url in event.mimeData().urls():
996                 # https://doc.qt.io/qt-5/qurl.html
997                 if url.isLocalFile():
998                     links.append(str(url.toLocalFile()))
999                 else:
1000                     links.append(str(url.toString()))
1001             self.addItem(links)
1002         else:
1003             event.ignore()
1004
1005 class AppDemo (QMainWindow):
1006     def __init__(self):
1007         super().__init__()
1008         self.resize(1200, 600)
1009
1010         self.listbox_view = ListBoxWidget(self)
1011
1012         self.btn = QPushButton('Start plot', self)
1013         self.btn.setGeometry(850, 400, 200, 50)
1014         self.btn.clicked.connect(lambda: self.umap_scatter())
1015
```

```
1016     def getItems(self):
1017         items = []
1018         for x in range(self.listbox_view.count()):
1019             items.append(self.listbox_view.item(x).text())
1020         return items
1021
1022     def umap_scatter(self):
1023
1024         samples = []
1025
1026         for path in self.getItems():
1027             if os.path.isdir(os.path.expanduser(path)):
1028                 samples_list = os.listdir(os.path.expanduser(path))
1029                 samples_list = [path + song for song in samples_list if song.
                                endswith('.wav')]
1030
1031                 samples_list = sorted(samples_list)
1032                 samples.extend(samples_list)
1033             elif os.path.isfile(os.path.expanduser(path)) and path.endswith('.wav')
1034                 :
1035                 samples.append(path)
1036
1037         audio_signals, sampling_frequencies = [], []
1038
1039         for i in range(len(samples)):
1040             audio_signal, sampling_frequency = repet.wavread(os.path.expanduser(
1041                 samples[i]))
1042
1043             if np.shape(audio_signal)[0] >= 262144:
1044                 audio_signals.append(audio_signal)
1045                 sampling_frequencies.append(sampling_frequency)
1046
1047         data = calculate_rp_rh(audio_signals, sampling_frequencies)
1048         rh_data = []
1049
1050         for i in range(len(audio_signals)):
1051             rh_data.append(data[i]['rh'])
1052
1053         fig = plt.figure()
1054         fit = umap.UMAP(
1055             n_neighbors = 15,
1056             min_dist = 0.01,
1057             n_components = 2,
1058             metric='cosine'
1059         )
1060         umap_res = fit.fit_transform(rh_data)
1061
1062         ax = fig.add_subplot(111)
1063         ax.scatter(umap_res[:, 0], umap_res[:, 1])
```

```
1062     plt.show()
1063
1064 if __name__ == '__main__':
1065
1066     # methods used to generate data
1067
1068     # analyse_genre('Pop')
1069
1070     # calculate_genre_distances('Pop')
1071
1072     # for genre in GENRES:
1073     #     for feature in [RP, RH, BS]:
1074     #         generate_family_distances_heatmap(genre, feature, False, False, True)
1075
1076     # generate_genres_p_value_heatmap(RH, WINDOW, True)
1077
1078     # generate_genres_windows_p_value_heatmap(RH, True)
1079
1080     # plot_stem('StevenClark_Bounty', 5, BS)
1081
1082     # for genre in GENRES:
1083     #     for feature in [RP, RH, BS]:
1084     #         for window in [FULL, WINDOW]:
1085     #             for stat in [IQR, MEDIAN]:
1086     #                 sort_heatmaps(genre, feature, window, stat)
1087
1088     # application
1089
1090     app = QApplication(sys.argv)
1091
1092     demo = AppDemo()
1093     demo.show()
1094
1095     sys.exit(app.exec_())
```

Appendix B

Rhythmic Patterns

B.1 Full window stem analysis

B.1.1 Classical

Instrument 1	Instrument 2	Median	IQR
percussion	voices	0.0	0.0
voices	winds	0.0	0.0
electric	percussion	0.0155610285436777	0.0150630379475371
electric	electric	0.0	0.2263363681988202
strings	strings	0.1974384812474462	0.2314022421161702
electric	winds	0.0024769704079286	0.3515754724330122
electric	strings	0.538348312948561	0.4762954875232341
percussion	strings	0.2669434707097612	0.4817964053393617
winds	winds	0.4956891865656954	0.506157397985748
percussion	percussion	0.0163015231515015	0.5643591046638512
strings	voices	0.2076545497681873	0.6246696723750136
strings	winds	0.2785566450514535	0.7278110742046313
percussion	winds	0.3651049827614006	0.7649724895502195
electric	voices		
voices	voices		

B.1.2 Electronic/Fusion

Instrument 1	Instrument 2	Median	IQR
strings	strings	0.8471075273314546	0.0
percussion	winds	0.789415039722784	0.0891447946819183
percussion	percussion	0.501039098665326	0.4318623787758511
percussion	strings	1.3785989529807037	0.5266963280779182
strings	winds	0.9898763278252134	0.7108451320806798
electric	percussion	1.331972458225962	0.7890785321114975
electric	strings	1.2282805357517677	0.9753091548772216
electric	electric	1.4954272274488258	1.120675270905218
electric	winds	1.3152597316807086	1.176694248822456
electric	voices		
percussion	voices		
strings	voices		
voices	voices		
voices	winds		
winds	winds		

B.1.3 Jazz

Instrument 1	Instrument 2	Median	IQR
percussion	voices	0.9361574863896373	0.0
voices	winds	0.9361574863896373	0.0
winds	winds	0.9074343151481276	0.1873094656750688
electric	percussion	1.2322283608690476	0.2742880870607331
electric	electric	1.150882623662822	0.3018284598453549
strings	voices	0.806358876347041	0.3243048896823629
percussion	winds	1.257395907919284	0.3475993957814955
electric	strings	0.7686026788065365	0.4895551612235381
electric	winds	1.2684923782682134	0.5002234116482129
strings	winds	1.1051805806000328	0.5524115012674198
percussion	strings	0.7814668257127946	0.6464519909774816
strings	strings	0.8403803909305847	0.7790747073707055
electric	voices		
percussion	percussion		
voices	voices		

B.1.4 Musical Theatre

Instrument 1	Instrument 2	Median	IQR
percussion	percussion	2.3503347747723	0.0
voices	voices	2.574817360336036	0.0
percussion	voices	3.861786008123509	0.1723750684736691
electric	electric	2.1786447773741795	0.4986738767463294
strings	voices	1.0028518542078566	0.550714269973613
electric	percussion	2.5796471365257374	0.5946466076089134
electric	voices	3.362637219555637	0.9631873863955668
electric	strings		
electric	winds		
percussion	strings		
percussion	winds		
strings	strings		
strings	winds		
voices	winds		
winds	winds		

B.1.5 Pop

Instrument 1	Instrument 2	Median	IQR
strings	winds	0.0	0.0
strings	strings	0.4781561454666431	0.2454208645692842
strings	voices	0.4822523721509801	0.2943576124838427
electric	strings	0.3812809358637726	0.3163235175242551
voices	winds	0.4184537064496189	0.3186235841916887
electric	winds	0.5411397269023275	0.3548839969105775
voices	voices	0.4393655512934354	0.3693975303239916
electric	voices	0.5097281343555853	0.5201875781270704
electric	electric	0.5053304897647672	0.5638823507400303
percussion	voices	0.8628306530410201	0.5998970521078836
percussion	strings	0.5801563791719229	0.6652855772131667
percussion	winds	0.7705449884643261	0.7274752828236535
percussion	percussion	0.9761284873678896	0.8011754086144981
electric	percussion	0.6472624548935012	0.8973066444042781
winds	winds		

B.1.6 Rock

Instrument 1	Instrument 2	Median	IQR
electric	strings	0.5394036831057101	0.2017558619466202
electric	electric	0.6611348379136464	0.4861616030125387
electric	voices	0.7449646411344303	0.6442444052740494
strings	voices	0.7624901553843382	0.993254547681894
percussion	voices	1.1272986317454916	1.0443210634759832
percussion	strings	0.9657211206473468	1.2580855781708666
voices	voices	0.0	1.2827619467727454
percussion	percussion	1.5223111665428657	1.3050213081811175
electric	percussion	1.2231505418477742	1.4946159015972484
electric	winds		
percussion	winds		
strings	strings		
strings	winds		
voices	winds		
winds	winds		

B.1.7 Singer/Songwriter

Instrument 1	Instrument 2	Median	IQR
winds	winds	0.5767761536652342	0.1966136948912991
strings	strings	0.6343087849960829	0.302876582890455
strings	winds	0.4033360779883623	0.3358300796688155
electric	strings	0.4738132243623206	0.4054625781566633
electric	electric	0.4279633336430531	0.451527109512205
percussion	strings	0.6205828128491804	0.4763246159467673
percussion	winds	0.7394193283670673	0.5070694645860598
electric	percussion	0.4947725396080875	0.5432003337055119
electric	voices	0.3990955660323217	0.5453046157995567
electric	winds	0.3801624587739349	0.556855768871089
strings	voices	0.4742727385230508	0.5572265164714858
percussion	percussion	0.8595370789661951	0.6484162454633589
percussion	voices	0.5428990759501952	0.7707884951316821
voices	winds	0.3963621857221962	0.7961339123934488
voices	voices	0.4749117161893334	0.8322322347654185

B.1.8 World/Folk

Instrument 1	Instrument 2	Median	IQR
winds	winds	0.0	0.0
voices	voices	0.4579732438154719	0.1328869335222671
percussion	winds	1.4241848502569052	0.1491024832854024
electric	electric	0.4022114156645625	0.4189745801741759
electric	winds	0.0	0.4189745801741759
electric	percussion	1.213381184039927	0.5129404912671964
electric	strings	0.4189745801741759	0.8449452709252021
strings	strings	1.036673495581185	0.9616026258058687
percussion	strings	1.3115269191491603	1.0115634237085964
percussion	percussion	1.0977597321454675	1.1988689880414158
strings	winds	1.005968170048499	1.509286206686167
electric	voices		
percussion	voices		
strings	voices		
voices	winds		

B.2 Windowed stem analysis

B.2.1 Classical

Instrument 1	Instrument 2	Median	IQR
percussion	voices	0.0	0.4621131933779906
voices	winds	0.0	0.5496506477216082
strings	voices	0.2189797335687294	0.6532622177358958
percussion	percussion	0.1785741753651623	0.6978515911088199
strings	strings	0.3345700778088076	0.7837029301351033
percussion	strings	0.307307649739341	0.7909789934591447
strings	winds	0.3641332600940313	0.8379134663560867
percussion	winds	0.2676491657744274	0.841826593767996
winds	winds	0.4270104125950045	0.9103693741325956
electric	electric	0.0107526392542068	0.9596989885926286
electric	percussion	0.4425315770261221	0.9689316366069816
electric	winds	0.1874054358861331	1.20783634403079
electric	strings	0.7208174661075885	1.2896105261227346
electric	voices		
voices	voices		

B.2.2 Electronic/Fusion

Instrument 1	Instrument 2	Median	IQR
strings	strings	1.0060833240247136	0.2393210464064626
percussion	winds	1.6083904742485338	0.783564101292213
percussion	strings	2.16441999695111	0.9144146032759988
percussion	percussion	1.3754458211975966	1.152608832919353
strings	winds	2.0122187210814784	1.165977123187293
electric	percussion	1.992418588469603	1.217747011332834
electric	winds	1.921688999500156	1.387462840890641
electric	strings	2.0001719966931786	1.4509767741136392
electric	electric	1.9790511759482223	1.548850097743618
electric	voices		
percussion	voices		
strings	voices		
voices	voices		
voices	winds		
winds	winds		

B.2.3 Jazz

Instrument 1	Instrument 2	Median	IQR
electric	electric	1.3541924297403287	0.5015730419543274
winds	winds	1.428408506836818	0.513177610918834
electric	strings	1.5491382019293836	0.5882123660307017
percussion	strings	1.10566283661232	0.5944693020898413
strings	strings	1.3168417865837805	0.7003823818129649
electric	percussion	1.6972442903602911	0.7230462994989955
percussion	winds	1.6348653204712191	0.7316690731743565
electric	winds	1.6442914918157254	0.783278131105517
strings	winds	1.6128504303580835	0.7841666176701827
strings	voices	1.073608978800202	1.0597295122857844
voices	winds	1.3075701992567856	1.0651089447535518
percussion	voices	1.08427642083017	1.2603227970905055
electric	voices		
percussion	percussion		
voices	voices		

B.2.4 Musical Theatre

Instrument 1	Instrument 2	Median	IQR
percussion	percussion	2.753421720111425	0.2609135499968911
percussion	voices	4.44091108228621	0.4800457170029846
electric	electric	2.216911737975396	0.4875120001249895
electric	voices	4.108641883461537	1.1376257454794456
strings	voices	1.718827120847119	1.140122579492828
electric	percussion	2.625945102915365	1.284098941464888
voices	voices	3.26554343308437	1.692026590239195
electric	strings		
electric	winds		
percussion	strings		
percussion	winds		
strings	strings		
strings	winds		
voices	winds		
winds	winds		

B.2.5 Pop

Instrument 1	Instrument 2	Median	IQR
strings	strings	0.6492341212351977	0.2906497321148148
electric	strings	0.6686165198695325	0.385976905332771
strings	voices	0.817737252787195	0.5260997446461824
percussion	strings	0.7822681923453696	0.5658051839806558
voices	voices	0.7080503691139989	0.7366374271242659
electric	electric	0.722079066605307	0.7561889517844431
electric	voices	0.7884339390316408	0.7712449880221897
electric	winds	0.7868335511260214	0.9796618957503412
voices	winds	1.1002161568298772	0.9861028681149324
electric	percussion	0.769991234816764	1.0329953170071495
percussion	voices	1.0367689188096123	1.0685683379393842
strings	winds	0.5517257872650613	1.0882726141932753
percussion	percussion	0.95388517191628	1.1533969445903236
percussion	winds	1.07445574451538	1.9563319569586457
winds	winds		

B.2.6 Rock

Instrument 1	Instrument 2	Median	IQR
electric	strings	0.7994165712937558	0.4313992012348306
electric	electric	0.9439670120935558	0.6124737032045477
electric	voices	1.0671262058222777	0.9245690189064896
electric	percussion	1.3165826564495076	1.1025528286379704
percussion	voices	1.578732521866831	1.1477942751149162
percussion	percussion	1.6465727647659143	1.1810975052646806
percussion	strings	1.7464814922491487	1.3201345861410938
strings	voices	0.8203932515974548	1.473264915177413
voices	voices	0.8125566687921656	1.6398023498567054
electric	winds		
percussion	winds		
strings	strings		
strings	winds		
voices	winds		
winds	winds		

B.2.7 Singer/Songwriter

Instrument 1	Instrument 2	Median	IQR
electric	electric	0.6119762943248241	0.5420690386054134
electric	strings	0.6509021749121181	0.5993808966538321
strings	strings	0.8140481484760924	0.6222200573660539
percussion	strings	0.7370128846140138	0.6803310238370585
electric	percussion	0.6392742399832936	0.7381954199720907
electric	voices	0.6789772006655321	0.7677294667987095
strings	voices	0.7682887770045914	0.7731624692657747
percussion	winds	1.1237114928395298	0.900446942729761
percussion	percussion	0.8136266159864853	0.9313008653628512
percussion	voices	0.7720254002253356	0.9694175698713088
voices	voices	0.6623159434483056	0.987094421772598
electric	winds	0.6983962536001862	1.0684761064376245
strings	winds	0.8330973323466416	1.2331721689337645
winds	winds	1.16578100200309	1.4439896446203933
voices	winds	1.05315907573605	1.5047277266265224

B.2.8 World/Folk

Instrument 1	Instrument 2	Median	IQR
voices	voices	1.1210641656381988	0.2652805493832839
winds	winds	0.0	0.4616469994132335
percussion	winds	2.032652910107922	0.6638220134122492
electric	winds	0.4560598484382023	0.6639554207599625
electric	electric	0.5091291468658559	0.7763300358035011
strings	strings	1.153112490201111	0.9646442263263376
electric	percussion	1.739143360434568	1.0196324963398933
electric	strings	0.8402289231747981	1.1960454602068478
strings	winds	0.898310933556285	1.2210987847548291
percussion	strings	1.3993454365951523	1.331950319361594
percussion	percussion	1.1625244576009528	1.9666658230462648
electric	voices		
percussion	voices		
strings	voices		
voices	winds		

Appendix C

Rhythmic Histograms

C.1 Full window stem analysis

C.1.1 Classical

Instrument 1	Instrument 2	Median	IQR
percussion	voices	0.0	0.0
voices	winds	0.0	0.0
electric	electric	0.0	0.0449984043359558
winds	winds	0.0401851272205177	0.1579474002198712
electric	winds	0.0022247840634176	0.1862234787380952
electric	strings	0.2116737946894145	0.1917067455062225
percussion	percussion	0.0380156132391277	0.2011670589816445
percussion	winds	0.09685503567469	0.2437403599627653
strings	winds	0.1484243744302222	0.2877868759820329
percussion	strings	0.1783085476142377	0.2915709463582242
strings	voices	0.0112297878366678	0.2951291527756006
strings	strings	0.0561805246691273	0.3132010390911582
electric	percussion	0.2770428683399515	0.4031340302189433
electric	voices		
voices	voices		

C.1.2 Electronic/Fusion

Instrument 1	Instrument 2	Median	IQR
strings	strings	0.3448663459419216	0.0
percussion	strings	0.1408991844311652	0.0564657734502656
percussion	percussion	0.037646275751244	0.0660055861089574
percussion	winds	0.1509777237236846	0.0672334494506026
strings	winds	0.1225578787420649	0.0832656981890669
electric	strings	0.1456775855408131	0.1208493441427755
electric	winds	0.144443931497845	0.1284704404215746
electric	percussion	0.1542727567815493	0.172007184888018
electric	electric	0.2056226483604087	0.2483086514987641
electric	voices		
percussion	voices		
strings	voices		
voices	voices		
voices	winds		
winds	winds		

C.1.3 Jazz

Instrument 1	Instrument 2	Median	IQR
percussion	voices	0.2779029352697539	0.0
voices	winds	0.2779029352697539	0.0
winds	winds	0.0227402273535169	0.0182961613358726
electric	strings	0.0314920274363003	0.0201228958927013
percussion	winds	0.1716424291368614	0.0457514043460923
electric	electric	0.087497019841242	0.056555013123609
electric	percussion	0.0897606525311717	0.0819196413571327
strings	winds	0.101828914156402	0.1152642487793129
electric	winds	0.1388503317602502	0.1370952930319692
percussion	strings	0.0670487249922782	0.1820519716722791
strings	voices	0.1850174160029902	0.2020656307903222
strings	strings	0.0983800936224432	0.2658893495943176
electric	voices		
percussion	percussion		
voices	voices		

C.1.4 Musical Theatre

Instrument 1	Instrument 2	Median	IQR
percussion	percussion	0.0633810661291762	0.0
voices	voices	0.1692859484636668	0.0
electric	electric	0.2468790628509208	0.0870985882272483
percussion	voices	0.3543448588506659	0.0976663143921891
strings	voices	0.0598519939202125	0.1140760741116917
electric	voices	0.1135501632793665	0.1246854211837031
electric	percussion	0.2509342020496166	0.2285236164583637
electric	strings		
electric	winds		
percussion	strings		
percussion	winds		
strings	strings		
strings	winds		
voices	winds		
winds	winds		

C.1.5 Pop

Instrument 1	Instrument 2	Median	IQR
strings	winds	0.0	0.0
strings	strings	0.0406580744064454	0.0246066987411866
voices	winds	0.2793517632240263	0.0311459017637678
percussion	winds	0.2618013235968497	0.0374514640881761
strings	voices	0.0607587119141813	0.0621925538699075
electric	winds	0.2413121948232184	0.1196042026643656
percussion	strings	0.1808107556210725	0.1289847811686102
voices	voices	0.0208852684517739	0.1404011576517361
percussion	voices	0.2379748142935399	0.1591967787235212
electric	percussion	0.2040172770785004	0.1615906496665868
percussion	percussion	0.2002401147356791	0.1796529560634363
electric	strings	0.0897085082362449	0.1815360287102735
electric	voices	0.0885065870805887	0.1864111468496726
electric	electric	0.1421214109221821	0.1901204946250084
winds	winds		

C.1.6 Rock

Instrument 1	Instrument 2	Median	IQR
percussion	strings	0.1454799344520303	0.0930597002044714
electric	voices	0.1174550976680469	0.1224822346072658
electric	percussion	0.1515840013628205	0.1224910115878794
electric	electric	0.0976866342498445	0.133447056517287
percussion	percussion	0.17175046383888	0.1518249184206807
percussion	voices	0.1731139616172717	0.1541085726414011
voices	voices	0.0	0.1808445647916876
electric	strings	0.1195893868173058	0.1816088082387426
strings	voices	0.1157546403062891	0.2270064266148925
electric	winds		
percussion	winds		
strings	strings		
strings	winds		
voices	winds		
winds	winds		

C.1.7 Singer/Songwriter

Instrument 1	Instrument 2	Median	IQR
winds	winds	0.2422978975094911	0.0283473630934736
strings	strings	0.1768478483759346	0.0841555888751253
percussion	winds	0.1683510065735599	0.1183566474635944
electric	strings	0.1415341651743371	0.1475028378519283
percussion	percussion	0.2110738020362873	0.1479967160156772
strings	voices	0.1250509248431908	0.1534739642890812
percussion	strings	0.1946331551665611	0.1540095732101506
strings	winds	0.1171558823721972	0.1728795465377078
electric	electric	0.1153637056167362	0.1777314020123365
electric	voices	0.1136087470776989	0.1970378466690267
electric	winds	0.1376189731829781	0.2078613010469566
electric	percussion	0.1983574907478098	0.2171645152733909
percussion	voices	0.2315340955948685	0.2573935572304056
voices	voices	0.0563564031904201	0.3075267084997894
voices	winds	0.2540570024954454	0.3159795307683642

C.1.8 World/Folk

Instrument 1	Instrument 2	Median	IQR
winds	winds	0.0	0.0
voices	voices	0.0045405226461243	0.0016384621192513
electric	percussion	0.1611682610320174	0.026708452415958
percussion	winds	0.1611682610320174	0.0310379133137465
strings	strings	0.0358508295915953	0.0556787304830995
strings	winds	0.0427986285936122	0.061372780952824
percussion	strings	0.1714758901523908	0.0673911032299946
percussion	percussion	0.1230709512582338	0.1230709512582338
electric	strings	0.0241813272554028	0.1779298971173445
electric	winds	0.0	0.2167266936369561
electric	electric	0.1949695004690947	0.2243281499735601
electric	voices		
percussion	voices		
strings	voices		
voices	winds		

C.2 Windowed stem analysis**C.2.1 Classical**

Instrument 1	Instrument 2	Median	IQR
percussion	voices	0.0	0.1801697799037577
electric	strings	0.2588821975991879	0.2814008610644439
electric	electric	0.0320813659054321	0.2843762810092747
electric	winds	0.1025325611001406	0.3148724257956614
winds	winds	0.0521081167617382	0.3234016489630652
voices	winds	0.0	0.3525593354047696
strings	strings	0.1188944132133014	0.3614334270000284
strings	voices	0.0610390853771858	0.3756749016543803
percussion	winds	0.092063727565788	0.3795688011858951
percussion	percussion	0.0634338623079095	0.3846295888540689
percussion	strings	0.1672948118403657	0.3866992232402907
strings	winds	0.1543494601790265	0.3874271746834639
electric	percussion	0.2599625629632499	0.4778490288024554
electric	voices		
voices	voices		

C.2.2 Electronic/Fusion

Instrument 1	Instrument 2	Median	IQR
percussion	percussion	0.1530578937655629	0.1279349088449112
percussion	strings	0.2232005877275373	0.1412672044697417
strings	winds	0.2048259413458819	0.1456097022598051
electric	winds	0.2291175766639013	0.1591861149890516
percussion	winds	0.2288397729380372	0.1603127065353083
electric	percussion	0.2161948066700195	0.1706943026673129
electric	strings	0.232044568611065	0.183373184056681
electric	electric	0.2659260529604445	0.2239484384386552
strings	strings	0.3528743020980116	0.2444535077671301
electric	voices		
percussion	voices		
strings	voices		
voices	voices		
voices	winds		
winds	winds		

C.2.3 Jazz

Instrument 1	Instrument 2	Median	IQR
electric	strings	0.0979717779278862	0.0586489017129578
electric	percussion	0.1277288030711531	0.0865168332960955
electric	electric	0.1686524348451619	0.1162057433468289
strings	winds	0.190897613833444	0.1300980972792084
percussion	strings	0.0657918924137915	0.1350914814455349
winds	winds	0.1048378907913946	0.1456306381094271
percussion	winds	0.2310328408000824	0.1477773051558177
electric	winds	0.1996505365279083	0.1511025346548208
strings	strings	0.0655163737930055	0.1855133356980604
strings	voices	0.1806032087757292	0.2214581545683016
percussion	voices	0.1474610398575644	0.2357266193646768
voices	winds	0.2115024423592474	0.2535486234344573
electric	voices		
percussion	percussion		
voices	voices		

C.2.4 Musical Theatre

Instrument 1	Instrument 2	Median	IQR
voices	voices	0.1704047712117571	0.0793779128293276
electric	voices	0.2004221049619012	0.1223758425046044
strings	voices	0.2173428505044549	0.1486668614768024
electric	electric	0.2476132837486623	0.1558109136251142
percussion	percussion	0.152235430991572	0.1896850570133629
percussion	voices	0.4636067927031348	0.2538162661135202
electric	percussion	0.317390959673905	0.2778954831965613
electric	strings		
electric	winds		
percussion	strings		
percussion	winds		
strings	strings		
strings	winds		
voices	winds		
winds	winds		

C.2.5 Pop

Instrument 1	Instrument 2	Median	IQR
voices	winds	0.2344104842748223	0.1751366446031688
percussion	percussion	0.2584154563887018	0.1840234484493645
percussion	strings	0.2151253682585329	0.1894306519060237
electric	strings	0.1771198347107504	0.195873166349147
strings	strings	0.1400047855771958	0.1963214233792641
electric	percussion	0.2336680087539078	0.1969594658270302
electric	electric	0.187153421674845	0.2000263882682882
percussion	voices	0.2536163024959101	0.2246426749947677
strings	voices	0.1908537364823523	0.2250331350071232
electric	voices	0.1909052326147143	0.2257045460521739
voices	voices	0.0672135432966916	0.2747570255178551
electric	winds	0.2606361167916394	0.2751435113632862
percussion	winds	0.2351264150419437	0.2880414156300919
strings	winds	0.2248659977057909	0.3004696087157057
winds	winds		

C.2.6 Rock

Instrument 1	Instrument 2	Median	IQR
percussion	strings	0.1847950038906671	0.1302958645251121
percussion	voices	0.2500907570596318	0.1338799333062
electric	percussion	0.1981227945563357	0.1546826815970115
percussion	percussion	0.2304508006293319	0.1570877779369165
electric	strings	0.1349463768157545	0.1605895972351058
electric	electric	0.1480400519105976	0.161889488011706
electric	voices	0.1766136137056219	0.1850623161351685
strings	voices	0.1593861579492115	0.1984232959084139
voices	voices	0.0549392227192976	0.2469784853564129
electric	winds		
percussion	winds		
strings	strings		
strings	winds		
voices	winds		
winds	winds		

C.2.7 Singer/Songwriter

Instrument 1	Instrument 2	Median	IQR
strings	strings	0.2107585251737765	0.0956867508285814
percussion	strings	0.2037832023249303	0.178020901251208
electric	strings	0.1799011881826798	0.1796601483740364
strings	voices	0.206764705753967	0.1884910552899602
percussion	winds	0.2059980319963995	0.1963726201966615
electric	percussion	0.2176929263825182	0.2082088230289477
electric	electric	0.1788349677305958	0.2358298360031591
percussion	percussion	0.2273239242671507	0.2398672845069022
strings	winds	0.1768531363647272	0.246145772374728
electric	voices	0.2048778830609736	0.2530782173846944
percussion	voices	0.2420806034764483	0.2539914328940703
electric	winds	0.1668948155324139	0.2601843881443998
winds	winds	0.2162200080507323	0.2778170844016964
voices	voices	0.1209381937058586	0.3089315014682178
voices	winds	0.158320351877923	0.3121507653698753

C.2.8 World/Folk

Instrument 1	Instrument 2	Median	IQR
voices	voices	0.0317580826497802	0.0210061236902936
winds	winds	0.0	0.065669047210759
strings	winds	0.0626850118398215	0.1230215041827156
strings	strings	0.1086502357087317	0.1240875206383562
electric	percussion	0.2403991077737278	0.1360099032598557
percussion	strings	0.2204093930211723	0.1641529392428666
percussion	winds	0.2530267303859891	0.1849144980592858
electric	strings	0.1570955528676618	0.2131147895977286
percussion	percussion	0.1627372095229429	0.251847735105665
electric	electric	0.1685643585552104	0.2784921400032396
electric	winds	0.1645778496376954	0.3451874594757529
electric	voices		
percussion	voices		
strings	voices		
voices	winds		

Appendix D

Beat Spectrum

D.1 Full window stem analysis

D.1.1 Classical

Instrument 1	Instrument 2	Median	IQR
percussion	voices	0.8814334527328331	0.0199589918083579
voices	winds	0.6275391722653556	0.0463238302118829
strings	voices	0.6034116369267515	0.1639938235982005
electric	percussion	0.4796700361125901	0.1765482384615473
winds	winds	0.1724344243475404	0.1907176858216969
strings	winds	0.29723249364805	0.2241483880937388
percussion	winds	0.3197444189472944	0.2634884947488032
electric	electric	0.5904163206027633	0.2885076817095022
strings	strings	0.3363130822293538	0.3080017570994642
percussion	strings	0.3987331520747512	0.3127557660631858
electric	strings	0.4989497490149192	0.3251810910456595
electric	winds	0.4471340027980235	0.3632090909206009
percussion	percussion	0.3531628590018463	0.4816217867570512
electric	voices		
voices	voices		

D.1.2 Electronic/Fusion

Instrument 1	Instrument 2	Median	IQR
strings	strings	0.3666669319089931	0.0
strings	winds	0.3087457821994755	0.0341889676172316
percussion	winds	0.3427781102181041	0.1101403445945287
percussion	strings	0.3080600482967804	0.1840167209793737
electric	winds	0.325452862733506	0.2102095365995812
electric	strings	0.3499306706657292	0.2584864932833084
percussion	percussion	0.2867733280894862	0.2620760341098798
electric	percussion	0.3065487876869499	0.2869875445037635
electric	electric	0.4037840631473234	0.3071472734056404
electric	voices		
percussion	voices		
strings	voices		
voices	voices		
voices	winds		
winds	winds		

D.1.3 Jazz

Instrument 1	Instrument 2	Median	IQR
percussion	voices	0.2427038562416703	0.0
voices	winds	0.2874314513543708	0.0
electric	electric	0.0995650910470605	0.0618815752621635
strings	voices	0.3246046861437598	0.1025257114076577
strings	winds	0.2217245904584863	0.127808434528005
electric	percussion	0.2949563287724219	0.156191392334986
electric	winds	0.3945460864191295	0.1858028660035532
strings	strings	0.1300268590708198	0.2067036526553294
percussion	winds	0.5419334699003676	0.2123090965297342
percussion	strings	0.3894356605142833	0.2382844099108306
electric	strings	0.2926824003546584	0.2626684477207761
winds	winds	0.1269484862664848	0.2835550741868541
electric	voices		
percussion	percussion		
voices	voices		

D.1.4 Musical Theatre

Instrument 1	Instrument 2	Median	IQR
percussion	percussion	0.1718240842652977	0.0
voices	voices	0.1810184798102884	0.0
strings	voices	0.3991127889310372	0.0865244165273967
electric	electric	0.2016076139531598	0.1202813277820299
electric	voices	0.3497110243890743	0.1390029139989222
percussion	voices	0.5716424569067353	0.1788036601971447
electric	percussion	0.4842112593047114	0.1884709269248227
electric	strings		
electric	winds		
percussion	strings		
percussion	winds		
strings	strings		
strings	winds		
voices	winds		
winds	winds		

D.1.5 Pop

Instrument 1	Instrument 2	Median	IQR
strings	winds	0.330256632903477	0.0
voices	winds	0.2898890918353003	0.0350839399475848
strings	strings	0.1905089933785082	0.073584285890041
percussion	percussion	0.5407942582096636	0.0863544020979336
voices	voices	0.2398752699676553	0.1569191825944265
percussion	voices	0.4887279394176481	0.1853934052228906
strings	voices	0.254632829010997	0.1930612736179412
electric	winds	0.4089255310670547	0.2154106192753121
percussion	winds	0.3892184225477559	0.2220912214303206
electric	voices	0.3286697499840544	0.2304885825045379
electric	strings	0.2998579887997557	0.303754585665408
electric	percussion	0.4151851965072956	0.3039483633513359
percussion	strings	0.4398437401364913	0.323365203615734
electric	electric	0.2857545612840517	0.3501392217238072
winds	winds		

D.1.6 Rock

Instrument 1	Instrument 2	Median	IQR
voices	voices	0.2387586234241281	0.1603176196193056
strings	voices	0.4916206269772892	0.2046845605098747
electric	strings	0.2199754423489212	0.2083519027147758
percussion	percussion	0.3238111659694204	0.2408828932422316
percussion	voices	0.4458698593032	0.2459357711534263
electric	voices	0.3624724423440551	0.2592246497362245
percussion	strings	0.3873809415857812	0.2745181147153108
electric	electric	0.2397991152248316	0.3066459611386474
electric	percussion	0.2822608133013385	0.3328947961858882
electric	winds		
percussion	winds		
strings	strings		
strings	winds		
voices	winds		
winds	winds		

D.1.7 Singer/Songwriter

Instrument 1	Instrument 2	Median	IQR
winds	winds	0.4121948974342662	0.0018007220168291
strings	strings	0.2868365092197837	0.0849845923193608
voices	winds	0.2218225098880601	0.0905852279463187
electric	winds	0.2689185502963097	0.1789977595672307
electric	voices	0.3297537833024849	0.1839600868437469
voices	voices	0.2760793722462693	0.2106823296789537
percussion	voices	0.461695616393485	0.2351359182257038
strings	voices	0.3443558062409076	0.2568870117388347
electric	electric	0.2318464841711795	0.2989009177557874
strings	winds	0.2773543950525299	0.3017880193165022
electric	percussion	0.3700602491435258	0.315230276552212
electric	strings	0.2705597805760777	0.3450455031315722
percussion	strings	0.3633413840689413	0.3475811214090823
percussion	percussion	0.4194535570095235	0.4248563579052434
percussion	winds	0.2402945561189582	0.5358241697116739

D.1.8 World/Folk

Instrument 1	Instrument 2	Median	IQR
voices	voices	0.1437221114976652	0.031493210163555
winds	winds	0.1553225517534661	0.0609848107402151
electric	winds	0.4090957336556335	0.0949559412831774
strings	strings	0.1279609953301874	0.1203027341846362
percussion	winds	0.4556431312339109	0.121100395627855
percussion	percussion	0.6380300633317362	0.1463988913492344
electric	strings	0.3278687390811312	0.2267910779988417
strings	winds	0.203479005357631	0.249602996349566
electric	electric	0.4826479853197017	0.4431591858866065
percussion	strings	0.4934708112087574	0.4845311243216633
electric	percussion	0.2659476556159903	0.5998248693885387
electric	voices		
percussion	voices		
strings	voices		
voices	winds		

D.2 Windowed stem analysis

D.2.1 Classical

Instrument 1	Instrument 2	Median	IQR
strings	voices	0.247238472755282	0.2499845207184224
strings	strings	0.2009598788017023	0.2779627334540293
percussion	voices	0.2449720817406388	0.2954581086334786
percussion	strings	0.3079084495226455	0.2984994447034951
voices	winds	0.1795771489420632	0.3005670935644928
strings	winds	0.2816001952935888	0.3065085072768764
electric	percussion	0.3854275566944323	0.3186463481246954
percussion	winds	0.3481090173079078	0.3190657465885481
percussion	percussion	0.1776973174606138	0.3230806699193782
winds	winds	0.1791715916254323	0.3260817761031257
electric	electric	0.4482873658598877	0.3427078842091012
electric	strings	0.2925649132540056	0.3700070927031043
electric	winds	0.2609153247650393	0.4154245854389389
electric	voices		
voices	voices		

D.2.2 Electronic/Fusion

Instrument 1	Instrument 2	Median	IQR
strings	strings	0.323681093450467	0.1741815339830399
percussion	winds	0.390125075924774	0.208723484089314
electric	percussion	0.3720508763518159	0.2590107312137978
electric	winds	0.3556742799646182	0.2611194961090236
percussion	strings	0.4027071931217371	0.2662410214307139
strings	winds	0.378739010566409	0.2716919157501273
electric	electric	0.3826802405521433	0.2854899767737374
electric	strings	0.3919075090693074	0.2999537708768326
percussion	percussion	0.3102351541835237	0.3067218617056552
electric	voices		
percussion	voices		
strings	voices		
voices	voices		
voices	winds		
winds	winds		

D.2.3 Jazz

Instrument 1	Instrument 2	Median	IQR
percussion	voices	0.413825966923565	0.1532048152376904
strings	strings	0.1864254472230065	0.187252095574129
strings	voices	0.365039719024043	0.1884800113109381
electric	strings	0.2502721060247933	0.1925289345132552
strings	winds	0.2917015493361939	0.2046529820698475
percussion	winds	0.5361895843462385	0.2047098149756541
electric	electric	0.2364639127605172	0.2121533421386086
voices	winds	0.2928070091976067	0.2201776642454075
electric	percussion	0.3647681315862419	0.2247888204952193
electric	winds	0.4010857227718424	0.2625092077709372
winds	winds	0.2272980026016466	0.2721189743017561
percussion	strings	0.3988760237776072	0.3097699573597975
electric	voices		
percussion	percussion		
voices	voices		

D.2.4 Musical Theatre

Instrument 1	Instrument 2	Median	IQR
percussion	percussion	0.2376379855074618	0.0948092966457941
voices	voices	0.1212886692696906	0.1099109722249814
percussion	voices	0.5011610598391594	0.11186043466435
electric	voices	0.2893369045768056	0.1227152693374869
electric	electric	0.2329383313824334	0.1509258450236287
electric	percussion	0.4654650628448886	0.2271338257242168
strings	voices	0.4336703231724159	0.2688957494359713
electric	strings		
electric	winds		
percussion	strings		
percussion	winds		
strings	strings		
strings	winds		
voices	winds		
winds	winds		

D.2.5 Pop

Instrument 1	Instrument 2	Median	IQR
strings	strings	0.1246143804698286	0.2202896745563018
electric	strings	0.1620800184366855	0.2494249575545062
strings	voices	0.2320257981966419	0.2641705569070928
electric	electric	0.1493318802247706	0.2674274269672554
electric	winds	0.1389441259868215	0.2852204043997119
voices	winds	0.2882393248021209	0.3068272666290691
electric	voices	0.2473875303622678	0.3095918254737542
percussion	voices	0.3994366261979108	0.316812480469208
voices	voices	0.1546122760507499	0.3283054801563754
percussion	percussion	0.4761179472014795	0.3428102071438335
percussion	strings	0.3199102672021232	0.3842936611178106
electric	percussion	0.3495479509309117	0.3943089575235469
percussion	winds	0.3074261265726062	0.4546743607456586
strings	winds	0.2205745521942765	0.4925215051019872
winds	winds		

D.2.6 Rock

Instrument 1	Instrument 2	Median	IQR
electric	strings	0.0820370951973075	0.1956590651405659
electric	electric	0.1242125927984618	0.222189568974638
percussion	percussion	0.3948788601124923	0.2612768339318448
percussion	voices	0.3937479282752408	0.2844682075880272
percussion	strings	0.3164721657105997	0.2931714562885995
electric	percussion	0.3249455998837695	0.2993526654406703
electric	voices	0.2721919406012131	0.3597615575738049
voices	voices	0.1442906837366585	0.4155675860906906
strings	voices	0.3915552937644917	0.4179327530441099
electric	winds		
percussion	winds		
strings	strings		
strings	winds		
voices	winds		
winds	winds		

D.2.7 Singer/Songwriter

Instrument 1	Instrument 2	Median	IQR
electric	electric	0.117487850977241	0.2184669663692517
strings	strings	0.2113827854527659	0.2212305211101528
electric	winds	0.1567546300720601	0.2249116092351396
electric	strings	0.1374138141043554	0.2271658608004003
strings	winds	0.1807459376091351	0.2311696597946221
winds	winds	0.1687982944925509	0.2810401641465938
strings	voices	0.2766663708015502	0.3146574414491131
voices	winds	0.2745127356818532	0.3402246799674938
electric	voices	0.2401753328610791	0.3524317431880498
percussion	strings	0.3309935303259546	0.3604549085113415
voices	voices	0.1941727364321372	0.3777505730367052
percussion	voices	0.4207574189567894	0.3984565074135773
percussion	percussion	0.4303678551070395	0.4006577610046351
electric	percussion	0.3407429343989802	0.4273658387011625
percussion	winds	0.3368147348469695	0.479769300021619

D.2.8 World/Folk

Instrument 1	Instrument 2	Median	IQR
voices	voices	0.0735375666689234	0.1095478435546599
winds	winds	0.1867699778235564	0.1494353861667034
strings	strings	0.2302595957618168	0.1956237863674957
percussion	winds	0.592125600304306	0.1969461679650997
strings	winds	0.2274365560373265	0.2175197817020806
electric	winds	0.252677820199104	0.247742949368767
percussion	strings	0.3723844871302938	0.2651117363987192
electric	strings	0.2455727249618196	0.2759400604158251
percussion	percussion	0.4514860799851928	0.324155295882484
electric	percussion	0.4052802211523432	0.370863512971767
electric	electric	0.2727733664817235	0.5241205713728313
electric	voices		
percussion	voices		
strings	voices		
voices	winds		

References

- [1] Cycling '74. Max for live. <https://www.ableton.com/en/live/max-for-live/>, 2011.
- [2] Francesc Alías, Joan Claudi Socoró, and Xavier Sevillano. A review of physical and perceptual feature extraction techniques for speech, music and environmental sounds. *Applied Sciences*, 6(5), 2016.
- [3] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, 2005.
- [4] Gilberto Bernardes, Matthew Davies, and Carlos Guedes. *A Hierarchical Harmonic Mixing Method*, pages 151–170. 11 2018.
- [5] Gilberto Bernardes, C. Guedes, and B. Pennycook. Eargram : an application for interactive exploration of large databases of audio snippets for creative purposes. 2012.
- [6] Gilberto Bernardes, Carlos Guedes, and Bruce Pennycook. Eargram: An application for interactive exploration of concatenative sound synthesis in pure data. In *From Sounds to Music and Emotions - 9th International Symposium, CMMR 2012, Revised Selected Papers*, pages 110–129, 2013.
- [7] Peter Beyls, Gilberto Bernardes, and Marcelo Caetano. eargram actors: An interactive audiovisual system based on social behavior. *Journal of Science and Technology of the Arts*, 7:43, 11 2015.
- [8] Rachel Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Bello. Medleydb instrument taxonomy. <https://github.com/marl/medleydb/blob/master/medleydb/resources/taxonomy.yaml>, 2014.
- [9] Rachel Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. MedleyDB Audio: A Dataset of Multitrack Audio for Music Research, October 2014.
- [10] Abraham Bookstein, Shmuel Tomi Klein, and Timo Raita. Fuzzy hamming distance: A new dissimilarity measure (extended abstract). In Amihood Amir, editor, *Combinatorial Pattern Matching*, pages 86–97, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
- [11] Roberto Bresin, M. Karjalainen, Aki Kanerva, T. Mäki-Patola, A. Huovilainen, Sergi Jordà, Martin Kaltenbrunner, G. Geiger, R. Bencina, Amalia Götzen, Pietro Polotti, and D. Rocchesso. *Sound to Sense, Sense to Sound. A State of the Art in Sound and Music Computing*. Logos, 01 2008.

- [12] Marco Buongiorno Nardelli. Topology of networks in generalized musical spaces. *Leonardo Music Journal*, 30:1–9, 02 2020.
- [13] A. Burton. A hybrid neuro-genetic pattern evolution system applied to musical composition. 1998.
- [14] Nadia Carvalho, Sara Gonzalez-Gutierrez, Javier Merchan Sanchez-Jara, Gilberto Bernardes, and Maria Navarro-Caceres. Encoding, analysing and modeling i-folk: A new database of iberian folk music. In *DLfM2021 Virtual Conference, GA, USA*. Digital Libraries for Musicology, 2021.
- [15] Matteo Casu, Marinos Koutsomichalis, and Andrea Valle. Imaginary soundscapes: The soda project. In *Proceedings of the 9th Audio Mostly: A Conference on Interaction With Sound, AM '14*, New York, NY, USA, 2014. Association for Computing Machinery.
- [16] Eric F Clarke. Rhythm and timing in music. In Diana Deutsch, editor, *The psychology of Music*, pages 473–500. Academic Press, San Diego, US, 2nd edition, 1999.
- [17] Diogo Cocharro, Gilberto Bernardes, Gonçalo Bernardo, and Cláudio Lemos. A review of musical rhythm representation and (dis)similarity in symbolic and audio domains. 01 2021.
- [18] Grosvenor Cooper and Leonard B. Meyer. *The rhythmic structure of music*. University of Chicago P., 1963.
- [19] Matthew EP Davies, Guy Madison, Pedro Silva, and Fabien Gouyon. The Effect OF Microtiming Deviations on the Perception of Groove in Short Rhythms. *Music Perception*, 30(5):497–510, 2013.
- [20] S. Dixon, E. Pampalk, and G. Widmer. Classification of dance music by periodicity patterns. In *Proc. of the ISMIR*, 2003.
- [21] Simon Dixon, Elias Pampalk, and Gerhard Widmer. Evaluating rhythmic descriptors for musical genre classification. In *AES: 25th Int. Conf. Metadata for audio*. Audio Engineering Society, 2004.
- [22] Karl Henry Eschman. *Changing Forms in Modern Music*. Boston, Mass: E. C. Schirmer music company, 1945.
- [23] J. Foote and Shingo Uchihashi. The beat spectrum: a new approach to rhythm analysis. *IEEE Int. Conf. on Multimedia and Expo (ICME)*, pages 881–884, 2001.
- [24] P. Gärdenfors. *Conceptual Spaces: The Geometry of Thought*. A Bradford book. MIT Press, 2004.
- [25] Daniel Gómez Marín, Sergi Jordà Puig, Herrera Boyer, et al. Rhythm spaces. In *Pasquier P, Bown O, Eigenfeldt A. Proceedings of the 4th International Workshop on Musical Metacreation (MUME 2016). Seventh International Conference on Computational Creativity, ICC3 2016; 2016 Jun 27; Paris, France.[place unknown]: International Workshop on Musical Metacreation; 2016.[5 p.]*. International Workshop on Musical Metacreation, 2016.
- [26] M. Goto. Grand challenges in music information research. In *Multimodal Music Processing*, 2012.

- [27] Masataka Goto. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30, 09 2002.
- [28] Fabien Gouyon. *A Computational Approach to Rhythm Description — Audio Features for the Computation of Rhythm Periodicity Functions and their use in Tempo Induction and Music Content Processing*. PhD thesis, University Pompeu Fabra, Barcelona, Spain, November 2005.
- [29] Fabien Gouyon, Anssi Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14:1832 – 1844, 10 2006.
- [30] TU Wien MIR Group. Rhythm pattern audio feature extractor for music similarity, classification and recommendation. https://github.com/tuwien-musicir/rp_extract, 2015.
- [31] Kjell Gustafson. The Graphical Representation of Rhythm. (*PROPH*) *Progress Reports from Oxford Phonetics*, 3:6–26, 1988.
- [32] Richard W. Hamming. *Coding and Information Theory (2nd Ed.)*. Prentice-Hall, Inc., USA, 1986.
- [33] W.B. Hewlett, E. Selfridge-Field, and Center for Computer Assisted Research in the Humanities. *Melodic Similarity: Concepts, Procedures, and Applications*. Computing in musicology. MIT Press, 1998.
- [34] Geoffrey Hinton and Sam Roweis. Stochastic neighbor embedding. *Advances in neural information processing systems*, 15:833–840, 2003.
- [35] Ludger Hofmann-engl. Rhythmic similarity: A theoretical and empirical approach. In *Proc. of the 7th Int. Conf. on Music Perception and Cognition*, 2002.
- [36] Simon Johannng. *Towards a representation of musical rhythm*. PhD thesis, 08 2011.
- [37] Maximos A. Kaliakatsos-Papakostas, Andreas Floros, Nikolaos Kanellopoulos, and Michael N. Vrahatis. Genetic evolution of l and fl-systems for the production of rhythmic sequences. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '12*, page 461–468, New York, NY, USA, 2012. Association for Computing Machinery.
- [38] Roger A. Kendall and Edward C. Carterette. The Communication of Musical Expression. *Music Perception*, 8(2):129–163, 12 1990.
- [39] E. Lapidaki. *Consistency of Tempo Judgments as a Measure of Time Experience in Music Listening*. Northwestern University, 1996.
- [40] Cláudio Lemos, Diogo Cocharro, and Gilberto Bernardes. Understanding cross-genre rhythmic audio compatibility: A computational approach. *AudioMostly*, 09 2021.
- [41] Fred Lerdahl and Ray Jackendoff. *A generative theory of tonal music*. The MIT Press, 1983.
- [42] Thomas Lidy. *Evaluation of New Audio Features and Their Utilization in Novel Music Retrieval Applications*. PhD thesis, 01 2006.

- [43] Thomas Lidy and Andreas Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. pages 34–41, 01 2005.
- [44] Yang Liu and Godfried T Toussaint. Mathematical Notation, Representation, and Visualization of Musical Rhythm: A Comparative Perspective. *Int. Journal of Machine Learning and Computing*, 2(3):261–265, 2012.
- [45] Justin London. Rhythm. *The new Grove dictionary of music and musicians*, 21:277–309, 2001.
- [46] G. Egerton Lowe. *What is Rhythm?* Musical Times, 1942.
- [47] Catarina Maçãs, Ana Rodrigues, Gilberto Bernardes, and Penousal Machado. Mix mash: A visualisation system for musical mashup creation. In *22nd International Conference Information Visualisation, IV 2018, Fisciano, Italy, July 10-13, 2018*, page 471–477, 2018.
- [48] Catarina Maçãs, Ana Rodrigues, Gilberto Bernardes, and Penousal Machado. Mix mash: An assistive tool for music mashup creation from large music collections. *IJACDT*, 8(2):20–40, 2019.
- [49] Leland McInnes. Umap-learn documentation. <https://umap-learn.readthedocs.io/en/latest/parameters.html>, 2018.
- [50] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, September 2018.
- [51] Martin F McKinney, Dirk Moelants, Matthew EP Davies, and Anssi Klapuri. Evaluation of audio beat tracking and music tempo extraction algorithms. *Journal of New Music Research*, 36(1):1–16, 2007.
- [52] Benoit Meudic. Musical similarity in polyphonic context: a model outside time. In *In Proceedings of the XIV Colloquium on Musical Informatics*, 05 2003.
- [53] Leonard B. Meyer. *Explaining Music: Essays and Explorations*. University of California Press, 1973.
- [54] Dalibor Mitrovic, Matthias Zeppelzauer, and Christian Breiteneder. Features for content-based audio retrieval. *Advances in Computers*, 78:71–150, 01 2010.
- [55] Organisation (OECD). *Participative Web and User-Created Content: Web 2.0, Wikis and Social Networking*. 09 2007.
- [56] Francois Pachet. Rhythms as emerging structures. 01 2000.
- [57] E. Pampalk. Islands of Music: Analysis, Organization, and Visualization of Music Archives. Master’s thesis, Vienna University of Technology, December 2001.
- [58] Elias Pampalk, Andreas Rauber, and Dieter Merkl. Content-based organization and visualization of music archives. pages 570–579, 01 2002.
- [59] Plato. *Plato: Laws*. Cambridge Texts in the History of Political Thought. Cambridge University Press, 2016.

- [60] Zafar Rafii. Repeating pattern extraction technique. <https://github.com/zafarrafii/REPET-Python>, 2020.
- [61] Andreas Rauber and Markus Frühwirth. Automatically analyzing and organizing music archives. In Panos Constantopoulos and Ingeborg T. Sølvsberg, editors, *Research and Advanced Technology for Digital Libraries*, pages 402–414, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
- [62] Andreas Rauber, Elias Pampalk, and Dieter Merkl. The som-enhanced jukebox: Organization and visualization of music collections based on perceptual models. *Journal of New Music Research*, 32:193–210, 06 2003.
- [63] G. Richard, S. Sundaram, and S. Narayanan. An overview on perceptually motivated audio indexing and classification. *Proceedings of the IEEE*, 101(9):1939–1954, 2013.
- [64] Curtis Roads. *The Computer Music Tutorial*. MIT Press, Cambridge, MA, USA, 1996.
- [65] Adam P. Segal. Dissemination of Digitized Music on the Internet: A Challenge to the Copyright Act. *Santa Clara High Tech. Law Journal*, 12(4), 1996.
- [66] William A Sethares. *Rhythm and Transforms*. Springer-Verlag London, London, 1st edition, 2007.
- [67] Bob Snyder. *Music and Memory: An Introduction*. MIT Press, Massachusetts, USA, 2001.
- [68] David Temperley and Christopher Bartlette. Parallelism as a factor in metrical analysis. *Music Perception: An Interdisciplinary Journal*, 20(2):117–149, 2002.
- [69] N.P.M. Todd. Metre, grouping and the uncertainty principle. *Proceedings of the Third International Conference of Music Perception and Cognition*, pages 395–396, 01 1994.
- [70] Godfried Toussaint. A comparison of rhythmic similarity measures. 01 2004.
- [71] Godfried Toussaint. A Comparison of Rhythmic Dissimilarity Measures. *Forma*, 21:129–149, 2006.
- [72] Godfried Toussaint. Computational geometric aspects of rhythm, melody, and voice-leading. *Computational Geometry*, 43:2–22, 01 2010.
- [73] Godfried T Toussaint. On the divergence between two distributions and the probability of misclassification of several decision rules. In *Proceedings of the Second International Joint Conference on Pattern Recognition*, pages 27–35, 1974.
- [74] Godfried T. Toussaint. Sharper lower bounds for discrimination information in terms of variation. *IEEE Transactions on Information Theory*, 21(1):99–100, January 1975. Copyright: Copyright 2015 Elsevier B.V., All rights reserved.
- [75] G.T. Toussaint. *The Geometry of Musical Rhythm: What Makes a "Good" Rhythm Good?, Second Edition*. CRC Press, 2019.
- [76] Laurens van der Maaten and Geoffrey Hinton. Visualizing non-metric similarities in multiple maps. *Machine Learning*, 87(1):33–55, December 2011.
- [77] C. Weihs, D. Jannach, I. Vatolkin, and G. Rudolph. *Music Data Analysis: Foundations and Applications*. Chapman & Hall/CRC Computer Science & Data Analysis. CRC Press, 2016.

- [78] Tong Zhang and C.-C. Jay Kuo. *Content-Based Audio Classification and Retrieval for Audiovisual Data Parsing*. 01 2001.