

Similarity of structures in popular music

Benoît Corsini

Mathematics & Computer Science Department, Eindhoven University of Technology,
Eindhoven, the Netherlands

ARTICLE HISTORY

Compiled February 20, 2024

ABSTRACT

The study of the similarity matrix of a song has been a particularly efficient technique to characterize song structures. This method transforms a song into a matrix representing the proximity between its different sections and is usually used to automatically detect structural properties such as its verse, its chorus, its tempo, etc. In this paper, these matrix representations are used not to study the inherent structure of a song, but to compare them with each other. This allows to create a metric on songs related to their pattern matrices, on which statistical tools can be applied. This metric is used to create groups of songs with similar structures and leads to interesting observations on patterns commonly used by certain artists, for certain years, and in certain genres. Moreover, this approach also unveils structures used across different features, such as songs from different decades and genres. Finally, this metric on songs is evaluated on classification tasks and shows that its interest lies in its ability to highlight specific behaviours rather than general trends.

KEYWORDS

similarity matrix, pattern structure

Contents

1	Introduction	2
2	The pattern similarity matrix	6
3	Comparing songs	8
4	Results	13
5	Evaluation	21
6	Discussion	26
A	Dataset properties	30
B	Special cases of the bar distance	31
C	Extra figures	31

1. Introduction

The study of the structure of music is an enduring question that has been addressed in many ways in *Music Information Retrieval* (MIR) research. A common approach consists of grouping songs according to some dimension, such as genre and year, before analysing repetitions and common patterns in their composition [Simms, 1996]. This technique usually requires an in-depth analysis and annotation of the songs using music theory; examples of this technique can be found in [de Clercq, 2012, de Clercq, 2017], with the analysis of the structure of rock songs, in [Epstein, 1986], where the structure of a specific song is being decomposed, or in [Osborn, 2013], where a general structure is highlighted and corresponding songs are analysed and given as examples. An alternative approach is using mathematical models to analyse song structures [Harkleroad, 2006]. The appeal of mathematical modelling is the potential of highlighting hidden properties that human-based approaches cannot find [Cope, 2009].

When analysing song structures, a common task is the identification of *sections* within a song. These sections can appear at different levels: local short motifs, verse and chorus, or movements. To solve this task, an early method consists of separating songs into distinct sections by identifying *boundary points* [Ulrich et al., 2014]. Since then, new methods have used unsupervised learning applied on sections at various levels of the songs [Buisson et al., 2022] or combined the identification of sections with other characterisation tools, such as pre-existing classifications or pre-implemented auto-tagging models, to extract features of the songs [Salamon et al., 2021, Wang et al., 2021]. A recent line of work [Wang et al., 2022a, Wang et al., 2022b] has also been focused on describing the sections of a song by using a time-function which, for every timestamp of the song, provides one of seven possible categories (intro, verse, chorus, bridge, outro, instrumental, and silence). We refer to [Nieto et al., 2020] for a survey on section analysis within songs. Finally, it is worth mentioning the recent development of a structure extracting method applied to symbolic music scores [Jiang et al., 2022], where the authors use neural networks to create multi-level groups (such as singleton, pairs, and quadruplets) of bars and help annotate music scores.

One common approach to analyse music structure which shows promising results is to use 2-dimensional similarity matrices to represent songs [Foote, 1999]. This method is based on transforming a song into a matrix, whose entries correspond to the similarities between the different parts of the song. To compute these matrices, the authors usually use audio files and transform them into sequences of frequencies; they then compare the frequencies between the different frames to compute the similarity matrix. The most common task performed using these matrices is the decomposition of songs into sections [Bartsch and Wakefield, 2001, Bartsch and Wakefield, 2005, Foote, 1999, Foote, 2000, Paulus and Klapuri, 2006, Paulus and Klapuri, 2009]; however, they can also be used to extract information from the song [Foote and Uchihashi, 2001], such as the tempo, or to distinguish the different signals in the song [Rafi and Pardo, 2011, Rafi and Pardo, 2012], such as background and foreground voices. Similarity matrices have also recently been used to quickly identify different versions of the same song [Silva et al., 2018]. It is worth noting that, although [Jiang et al., 2022] studies the structure of songs using symbolic music scores, and similarity matrices are standard in the analysis of song structure, the combination of these two ideas has not been done so far.

Since the method described in this article defines and studies a similarity metric on songs, it is worthwhile to compare it with other song clustering techniques.

While there exists a rich literature on defining a measure of similarity for songs, most of them are based either on a harmonic analysis [Eerola et al., 2001], on a motive analysis [Cambouropoulos and Widmer, 2000], or a combination of the two [Pienimäki and Lemström, 2004]. These methods can then be further combined with efficient melodic analysis tools, but none of them base their similarity distances, and hence their resulting clusters, on overall song structures.

The study presented in this paper is based on the use of similarity matrices with two novel approaches. First, songs are analyzed not using a music file, but with a direct digital notation score of the music, using partitions of the different instruments. This allows a direct comparison of patterns in the song and could open the door to more complex analysis, for example by combining our method with other tools analysing similarities between chords. Second, the resulting similarity matrices are not used to analyse the structure of a specific song but to make a comparison between each other, calculating the distance on songs based on their patterns. This allows the characterization of groups of songs with similar structures and highlights common patterns which may arise by artist, years, or genres.

In order to conduct a statistical analysis of patterns of songs based on their digital notation score, a large dataset of songs in that format was required. For this reason, this article and the code provided hereby also contain a new dataset of 4166 popular songs in a digital notation score format (GuitarPro format [Guitar Pro,]), along with a set of 6 features for each song, further described below. This dataset and some further information are available on *GitHub* [Corsini,].

2-dimensional representation of songs

Consider a song whose structure falls into the typical verse-chorus-verse-chorus-solo-chorus progression. If comparing the different sections of the song, one could see that the verses and chorus repeat themselves. Moreover, if possible to quantify the difference between the sections, one could see that the solo is likely to be *more different* than the verses and the chorus. Representing these observations into a 2-dimensional matrix whose entries correspond to the general similarity between sections, something similar to Figure 1 would be obtained. Now, this representation is not strongly dependent on the song in terms of notes, chords, or tempo, but rather on its general structure, making it a useful tool to compare songs based on their pattern structure.

The previous analysis and example use a rather simple approach to divide a song into large sub-sections: verse, chorus, solo, etc. By doing a finer level of analysis, one could hope to highlight these macroscopic behaviours, as well as finer sub-patterns. Overall, the level at which the analysis is done can greatly influence the appearance of the results. In this study, since the representation is obtained from standard musical notation scores, a common subdivision is the bar length. The types of results and figures obtained in this study are summarized in Figure 2, which transforms the song BEAT IT by MICHAEL JACKSON into a 2-dimensional matrix where each entry describes the similarity between the bars of the song. Observe that the diagonal (top-left to bottom-right) is white since it corresponds to the similarity between the same two bars.

An important property for representing songs into matrices is that it is possible to transform abstract objects (songs) into mathematical objects (matrices), for which standard analysis techniques can be applied. This study uses this observation to define a distance D on songs based on their pattern structure. This distance is meant to

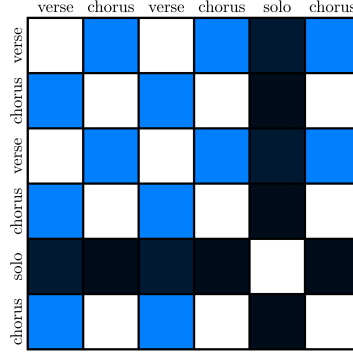


Figure 1. A possible depiction of a song with the structure verse-chorus-verse-chorus-solo-chorus. White means that the two sections are the same, dark means that they are very different, and a scale of blue is used to interpolate in-between. In most songs, the solo is very different from the rest of the songs, whereas the difference between the verse and the chorus is not as significant. This representation only depends on the structure of the song and not on its notes, tempo, or genre.

express the similarity of patterns between different songs. Using D , it becomes possible to identify groups of songs with similar structures.

Having defined D , this study will next extrapolate the statistical properties of song patterns. In particular, the following questions will be addressed.

- Given a song feature, such as year, genre or artist, do there exist specific feature values with a common pattern structure? Given a feature, what are the feature values with the least variability in pattern structures?
- Given a group of songs with similar patterns, does there exist a corresponding feature value? Does this group correspond to a specific category of songs?

Finally, this study concludes with an analysis of the ability of D to classify songs based on their features. In particular, we will see that the results obtained in the previous section seem to correspond to specific (yet interesting) cases of song similarities, but that the metric D does not encapsulate further properties of the general feature space of songs and thus provides poor results on classification tasks. This then confirms the goal of this tool and the method developed here: it highlights outliers and extreme behaviours while being, on average, rather uncorrelated with the features of the songs.

Methodology

This study aims to better understand the pattern structure of *popular* music, i.e. songs that at any point reached the *Billboard Hot 100 chart* [Billboard,]. To compute the pattern matrices, all songs were used in the form of *Guitar Pro* files, which contain a wide range of information on the song, such as key signature, chords, tempo, standard notation scores and tablatures, etc; we provide more details on the transformation of standard music notations into similarity matrices in Section 2 and we refer to the official Guitar Pro website [Guitar Pro,] as well as the Python library used in this work [Abakumov,] for more information on the characteristic of the format of files used here. On top of the Guitar Pro file, each song is defined by 6 features:

- (1) **artist**: the artist of the song.
- (2) **title**: the title of the song.

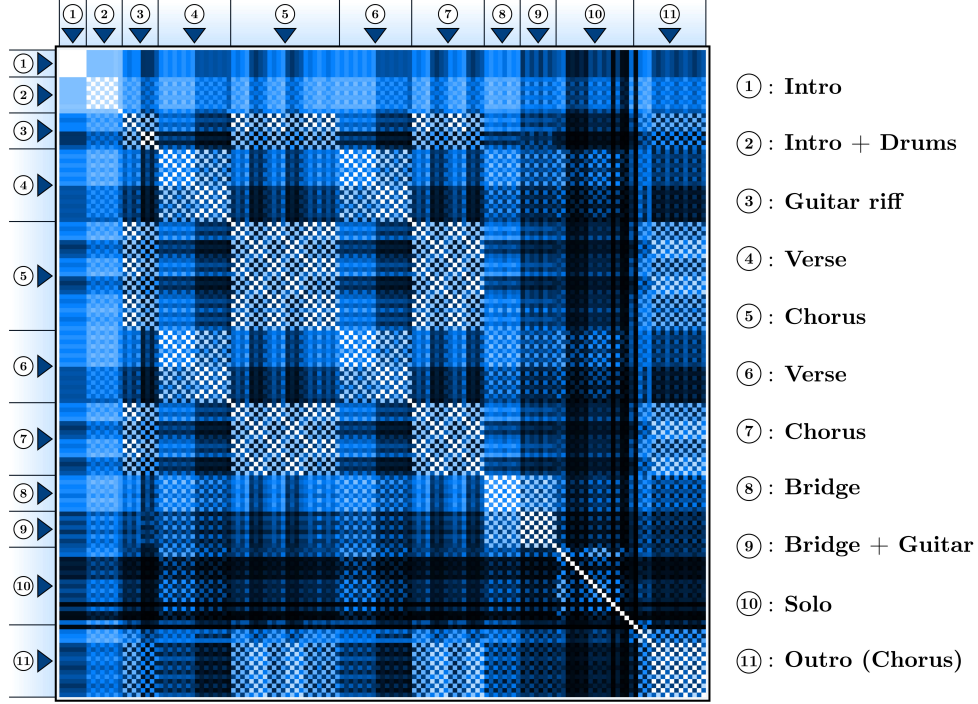


Figure 2. The representation of the song MICHAEL JACKSON - BEAT IT and its structure decomposition. The song can be read in two directions: from left to right or from top to bottom, each square representing the similarity between two bars (see Section 2 for the precise definition of the matrix and Section 3 for its re-normalization). Lighter squares correspond to smaller entries in the pattern matrix and darker squares correspond to larger entries, making the different sections of the song visible from the variation of shades.

- (3) **year**: the first year the song appeared on the chart.
- (4) **decade**: the decade corresponding to the feature **year**.
- (5) **genre**: the genre(s) of the song.
- (6) **type**: types of genre, obtained from the feature **genre**.

The reason for considering the year and decade of first appearance on the chart rather than the actual release date is to focus on the evolution of patterns through popular taste, rather than when the songs were actually written and produced. The last feature, **type**, corresponds to general genres like ‘rock’, ‘pop’, or ‘hip hop’, as well as genre adjectives, such as ‘experimental’, ‘classical’, or ‘psychedelic’. The full list can be found in [Corsi,] in the file `dataset/types.json`, while the 10 types corresponding to the most genres can be found in Table A3; this table can give an idea of what the values of the feature **genre** and **type** look like.

Dataset

For this article, a new dataset of 4166 songs in their symbolic music scores format along with the 6 previously described features was created. The dataset was collected using the *Billboard Hot weekly charts* dataset [Miller,], which includes the first four features. A program [Corsi,] went through the songs of this dataset and collected the available Guitar Pro files online (exploring [Ultimate Guitar,]). The **genre** feature was computed by searching for the song properties on *Wikipedia* [Wikipedia,] and the **type** feature was generated using the **genre** feature. For more details on how the

feature values of **type** were generated, or about this dataset in general, see Appendix A or [Corsi,].

Throughout this process, a lot of automation was used and this dataset may contain noise from any of the following steps:

- Errors in the original Billboard Hot dataset.
- Wrong song downloaded.
- Bad Guitarpro file: fewer instruments, only partial song coverage, not the original version, and mistakes in the writing process.
- Mismatched genres.

For each such step, human-based controls were put in place to reduce noise. Among these steps, the one which was the most difficult to control was the third step, related to the direct quality of the files. However, this type of noise should only concern a small portion of songs and will not strongly influence the analysis and the results.

2. The pattern similarity matrix

Let \mathcal{S} be a song divided into its ordered sequence of b bars: $\mathcal{S} = (\mathcal{B}_k)_{1 \leq k \leq b}$ ¹. The *pattern similarity matrix* P (or simply *pattern matrix*) is obtained by defining a distance d_{bar} on bars, used as the entries of P :

$$P_{k,\ell} = d_{\text{bar}}(\mathcal{B}_k, \mathcal{B}_\ell).$$

If the song is composed of different instruments, the pattern matrix will be computed by summing the contributions of the different instruments:

$$P_{k,\ell} = \sum_{i \in \text{Ins}(\mathcal{S})} d_{\text{bar}}(\mathcal{B}_k^{(i)}, \mathcal{B}_\ell^{(i)}),$$

where $\text{Ins}(\mathcal{S})$ is the set of instruments of \mathcal{S} and $\mathcal{B}_k^{(i)}$ is the k -th bar of instrument i .

Distance of similarity on bars

In order to define d_{bar} , bars need to be transformed. The choice of transformation is to consider them as vectors of size s , where s is the number of time subdivisions in the bar. In other words, a bar \mathcal{B} is defined as a sequence $\mathcal{B} = (\mathcal{N}_t)_{1 \leq t \leq s}$ where each element \mathcal{N}_t corresponds to the note whose onset is at time t , and s corresponds to the product of the smallest note length (from the whole song) and the ratio of the bar time signature. An example of such representation can be found in Figure 3, using the main riff of SEVEN NATION ARMY by WHITE STRIPES.

Let $\mathcal{B} = (\mathcal{N}_t)_{1 \leq t \leq s}$ and $\tilde{\mathcal{B}} = (\tilde{\mathcal{N}}_t)_{1 \leq t \leq s}$ be two bars. A definition for d_{bar} is to count the number of differences between \mathcal{B} and $\tilde{\mathcal{B}}$:

$$d_{\text{bar}}(\mathcal{B}, \tilde{\mathcal{B}}) = \sum_{1 \leq t \leq s} \delta(\mathcal{N}_t, \tilde{\mathcal{N}}_t),$$

¹ It is worth noting that the bar decomposition is not unique; for example, one bar of 4/4 can also be seen as two bars of 2/4. This could be a problem if the file is poorly written and/or if the songs are composed of complex meters, but is mostly irrelevant in the case of popular music, usually encoded by a standard 4/4 time signature.

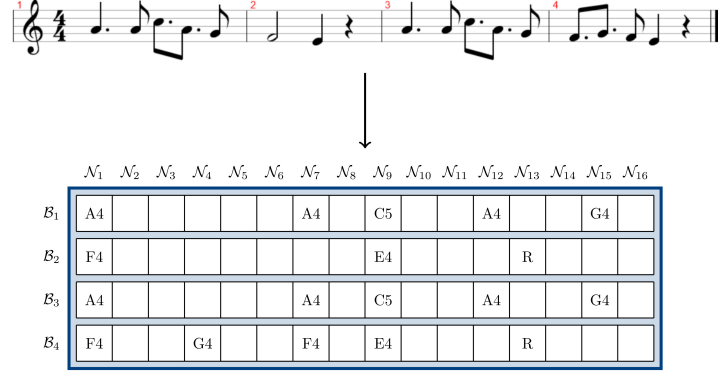


Figure 3. The representation of the four bars ($\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$) into four vectors of size $s = 16$. This choice for s follows from the smallest subdivision being a sixteenth note (due to the dotted eighth note) and the bars having a time signature of 4/4, meaning that there is at most $16 \times 4/4 = 16$ notes in each bar. For each time $t \in \{1, \dots, 16\}$, the entry t of the vector corresponds to the onset of the note being played at time t , and is empty otherwise. A special letter, R, is used for a rest.

where

$$\delta(\mathcal{N}, \tilde{\mathcal{N}}) = \begin{cases} 0 & \text{if } \mathcal{N} \text{ and } \tilde{\mathcal{N}} \text{ are the same} \\ 1 & \text{if exactly one of } \mathcal{N} \text{ and } \tilde{\mathcal{N}} \text{ is empty} \\ 2 & \text{otherwise.} \end{cases}$$

The reason for this definition of d_{bar} and δ is that, each time the onset of a note is being played in one of the two bars, it adds one to their distance, except if the same note is being played at the same time in the other bar. From this definition, $d_{\text{bar}}(\mathcal{B}, \tilde{\mathcal{B}}) \in \{0, 1, \dots, 2s\}$, which implies that the range of d_{bar} is related to the number of notes per bar.

In the definition of d_{bar} , it is assumed that the two bars have the same size s . This does not necessarily follow from the previous definition of s , but can be obtained by normalizing the two bars using the smallest note subdivision and the longest time signature ratio. With this definition, if the two bars have different time signatures, then the matrix representation of the bar with a smaller time signature ratio has empty entries past the last time of the bar. This makes every note played at the end of the other bar count as one in the distance d_{bar} . We provide an example of the computation of d_{bar} for two bars with different time signatures in Figure 4.



Figure 4. An example of two bars with different time signatures. If computing d_{bar} for this pair of bars, the result would be 1 since the first four notes are the same and the last note of the second bar corresponds to a time which does not exist in the first bar. This choice comes from the fact that longer bars are usually used to extend a previous pattern (and similarly, shorter bars are usually used to reduce a previous pattern). This choice, however, does not have a significant effect on the results since most of the songs considered here have a classical 4/4 time signatures.

To further illustrate how d_{bar} is computed, we provide in Appendix B a few special cases: when the two bars have different time signatures, how rests are dealt with, and

how chords are compared to each other. It is worth noting that the definition of d_{bar} does not depend on s , provided s is large enough to allow \mathcal{B} and $\tilde{\mathcal{B}}$ to be properly defined (the onset of each note in \mathcal{B} and $\tilde{\mathcal{B}}$ corresponds to an exact index between 1 and s). Using this remark, when computing the pattern matrix, the same s will actually be used for all bars of the song.

With d_{bar} being defined for any pair of bars, the pattern matrix P of any song can now be computed. As an example, the pattern matrix of the main riff of SEVEN NATION ARMY, whose vector representation was given in Figure 3, can be found in Figure 5. For the rest of the paper, this exact definition of d_{bar} is used to compute the pattern matrix of a song. It is worth mentioning however that the images found in Section 4 as well as that of Figure 2 are all re-normalized using ν (defined in Section 3) for clarity.





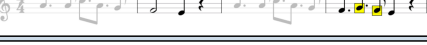

pair of bars	distance	vector representation of the two bars										depiction of the distance in the original music sheet									
(1, 2)	8	A4				A4	C5		A4		G4										
		F4					E4			R											
(1, 3)	0	A4				A4	C5		A4		G4										
		A4				A4	C5		A4		G4										
(1, 4)	10	A4				A4	C5		A4		G4										
		F4		G4		F4	E4			R											
(2, 3)	8	F4					E4			R											
		A4				A4	C5		A4		G4										
(2, 4)	2	F4					E4			R											
		F4		G4		F4	E4			R											
(3, 4)	10	A4				A4	C5		A4		G4										
		F4		G4		F4	E4			R											

Figure 5. A representation of the entries of the pattern matrix of the main riff of SEVEN NATION ARMY by WHITE STRIPES. For each row, the first column gives the indices of the two bars being compared, the second column gives the distance between these two bars, the third column gives the two vector representations of the bars obtained in Figure 3, and the last column gives the original music sheet used to compute the pattern matrix. The notes highlighted in yellow in the third and fourth columns correspond to the differences between the two bars and each yellow square counts as one in the distance of the two bars. From this figure, one can

see that the corresponding pattern matrix is given by
$$\begin{pmatrix} 0 & 8 & 0 & 10 \\ 8 & 0 & 8 & 2 \\ 0 & 8 & 0 & 10 \\ 10 & 2 & 10 & 0 \end{pmatrix}.$$

3. Comparing songs

With the pattern matrix of a song defined in Section 2, a distance on pattern matrices can now be computed in order to obtain D , the distance on song structures. This section focuses on the definition of D .

Directly comparing pattern matrices presents a few problems. First of all, they do not necessarily all have the same shape, as P has size $b \times b$, where b is the number of bars of the song. Second, recall that the entries in these matrices are related to the number of notes being played, implying that they can greatly vary between songs. Finally, this link between entries of the pattern matrix and the number of notes played can also create high variability of entries inside a single song, for example, if a sub-section has a much higher density of notes (such as a solo, see Figure 1).

For the rest of this section, the goal is to define the function

$$\nu : P \in \mathbb{N}^{b \times b} \mapsto \nu(P) \in [0, 1]^{s_0 \times s_0}$$

used to normalize pattern matrices so that they can be easily compared with each other. Here, s_0 is a constant over all songs and $\nu(P)$ is a normalized pattern matrix and should reflect the pattern structure of P while avoiding the previously raised issues. With ν being defined, the distance over songs D is set as

$$D(\mathcal{S}_1, \mathcal{S}_2) = \|\nu(P_1) - \nu(P_2)\|_p, \quad (1)$$

where P_1 and P_2 are the pattern matrices of \mathcal{S}_1 and \mathcal{S}_2 . Here, $\|\cdot\|_p$ is defined as the average L^p norm; that is, for all A of size $s_0 \times s_0$, we have $\|A\|_p = \left(\frac{1}{s_0^2} \sum_{1 \leq k, \ell \leq s_0} |A_{k, \ell}|^p\right)^{1/p}$. The value of p is implied in the definition of D and set to $p = 2$ for the rest of this study.

It is worth noting that D is actually not a *proper* metric on the set of songs, but rather a *pseudometric*: it satisfies all the axioms of a metric except that we might have $D(\mathcal{S}_1, \mathcal{S}_2) = 0$ for two distinct songs $\mathcal{S}_1 \neq \mathcal{S}_2$. This property comes from the fact that the pattern matrices of two distinct songs could be the same and that ν is likely not injective (consider the case $s_0 = 2$ for example). The norm $\|\cdot\|_p$ on $[0, 1]^{s_0 \times s_0}$ however corresponds to a proper metric space.

Pattern matrix normalization

Let P be a pattern matrix of size $b \times b$ whose entries take values in \mathbb{N} . In order to visualize the impact of the different modifications made on P by the normalizing function ν , matrices are depicted as in Figure 2: a white square corresponds to a 0 in the matrix, whereas a black square corresponds to the maximal possible entry, and a scale of blue is used to interpolate values in-between.

As explained earlier, the entries of P are related to the number of notes being played, which means that these entries can greatly vary between songs. In order to avoid scaling problems when comparing two pattern matrices, their entries have to be taking values in $[0, 1]$. The importance of this constraint is represented in Figure 6, where the two songs represented have similar structures but different ranges of entries in their pattern matrices.

The problem of large entries in pattern matrices does not simply affect comparing two different songs, but can also prevent the matrix of a given song from clearly highlighting its structure. For example, a song with a section denser than the other ones will have very high entries in specific rows and columns of its pattern matrix, making the rest of the entries small in comparison; and this could hide possible sub-patterns of the song. To avoid this issue, the variation of the entries in P is normalized by using the corresponding percentile index rather than the exact value².

Write $\vec{P} = (P_{k_i, \ell_i})_{1 \leq i \leq b^2}$ for the sequence of ordered entries of P ; in other words, $(k_i, \ell_i) \neq (k_j, \ell_j)$ for $i \neq j$, and $P_{k_i, \ell_i} \leq P_{k_{i+1}, \ell_{i+1}}$. Let $q_\alpha = P_{k_{\lfloor \alpha b^2 \rfloor}, \ell_{\lfloor \alpha b^2 \rfloor}}$ be the α quantile of \vec{P} and $p_j = q_{j/100}$ be the j -th percentile of \vec{P} . The percentile normalization

² The reason for choosing the percentile index instead of a more common activation function, such as *softmax* or *softmin*, comes from the desire to completely remove the bias created by large entries in the matrix, instead of just reducing it.

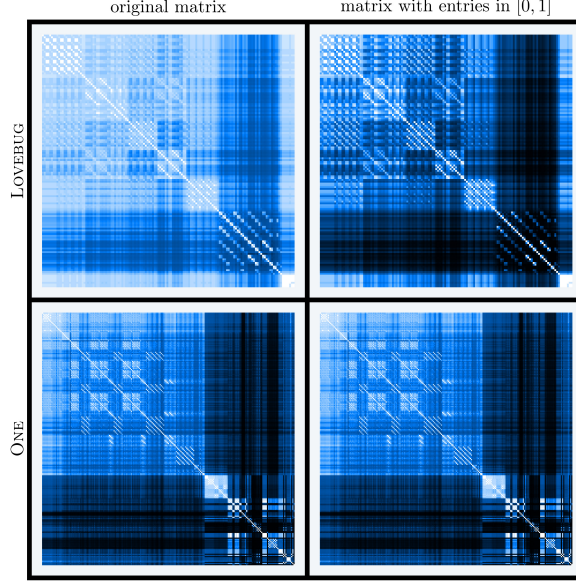


Figure 6. Comparison of the songs JONAS BROTHERS - LOVEBUG and METALLICA - ONE before and after normalizing their entries between 0 and 1. Each column represents the two pattern matrices on the same scale of blue; the left column corresponds to the original pattern matrices, whereas the right column corresponds to the pattern matrices whose entries have been scaled between 0 and 1. As can be observed from the right column, these two songs have a similar structure. However, if only comparing the values of the two pattern matrices (left column), the entries of the matrix of the METALLICA song are much larger than the entries of the matrix of the JONAS BROTHERS song.

function ν_p is defined as

$$\nu_p : P \mapsto \left(\max\{j : p_j \leq P_{k,\ell}\} \right)_{1 \leq k, \ell \leq b},$$

and replaces the entries of P by their corresponding percentile index. An example of the importance of applying ν_p can be found in Figure 7.

After applying ν_p to a pattern matrix P , the maximal entry is always 100, corresponding to the maximal entries of P . However, the minimal entry might not be 0, for example if there are many 0 in P . In order to avoid this issue, a second normalizing function ν_0 is applied, forcing the entries of the matrix to take values in $[0, 1]$, with 0 and 1 included. More precisely ν_0 is defined as

$$\nu_0 : \tilde{P} \mapsto \frac{1}{100 - \min\{\tilde{P}\}} (\tilde{P} - \min\{\tilde{P}\}),$$

where $\{\tilde{P}\} = \{\tilde{P}_{k,\ell}, 1 \leq k, \ell \leq b\}$. Here, \tilde{P} is used instead of P to highlight the fact that ν_0 is supposed to be applied to the image of P through ν_p and not directly to the pattern matrix P . The effect of ν_0 on \tilde{P} is depicted in Figure 8. The changes operated by ν_0 are not meant to have a strong impact on the whole process, but give a desired property for the resulting matrix: bars which are the same correspond to entries with value 0, and bars with most differences correspond to entries with value 1.

When applying ν_p and then ν_0 to a pattern matrix P , it is transformed into a matrix of size $b \times b$ whose entries take values in $[0, 1]$. Moreover, the resulting matrix clearly represents the structure of the original song, meaning that it can be used to

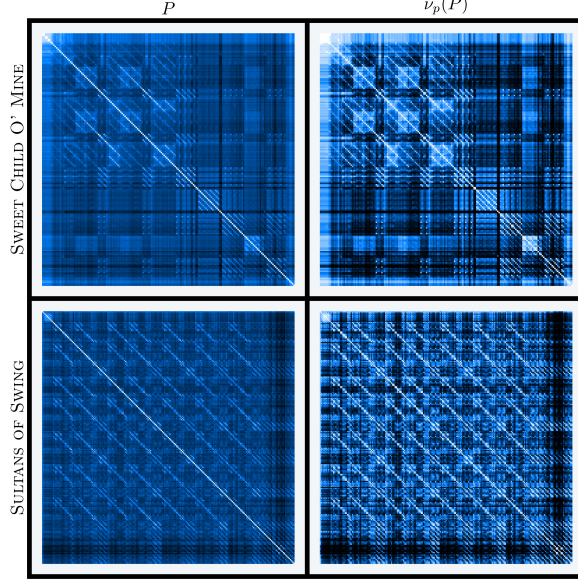


Figure 7. Comparison of the songs GUNS N' ROSES - SWEET CHILD O' MINE and DIRE STRAITS - SULTANS OF SWING with and without applying ν_p to their pattern matrices. The left column corresponds to the original pattern matrix and the right column corresponds to its image through ν_p . As can be observed from the left column, a few bars with high distance to the rest of the song tend to make the overall picture uniformly blue, making both of these pattern matrices look similar. However, after applying ν_p , different patterns appear more clearly, showing the diversity in structure of these two songs. This figure also represents the importance of ν_p when representing patterns of songs, since it highlights their patterns.

compare the structure of different songs. Before being completely able to compare songs according to their pattern matrices, it remains to address the variety in sizes b of the matrices. Since b corresponds to the number of bars of the song, it is most likely different from one song to the other.

Resizing matrices

Consider two normalized pattern matrices \tilde{P} and \tilde{Q} of respective sizes $b \times b$ and $d \times d$, obtained as the image of two pattern matrices through $\nu_0 \cdot \nu_p$. A straightforward way to compare \tilde{P} and \tilde{Q} in spite of their respective sizes is to extend them by repeating each of their entry multiple times. More precisely, \tilde{P} and \tilde{Q} are transformed using σ , defined as follows

$$\sigma : (\tilde{P}, \tilde{Q}) \mapsto \left((\tilde{P}_{\lceil k/d \rceil, \lceil \ell/d \rceil})_{1 \leq k, \ell \leq bd}, (\tilde{Q}_{\lceil k/b \rceil, \lceil \ell/b \rceil})_{1 \leq k, \ell \leq bd} \right).$$

The image of (\tilde{P}, \tilde{Q}) through σ is a pair of matrices, whose sizes are both $bd \times bd$, and whose entries are related to the entries of \tilde{P} and \tilde{Q} repeated multiple times: d^2 times for the entries of \tilde{P} and b^2 for the entries of \tilde{Q} . Using σ , let the true distance d_{true} of \tilde{P} and \tilde{Q} be defined by

$$d_{\text{true}}(\tilde{P}, \tilde{Q}) = \|\tilde{P}^\sigma - \tilde{Q}^\sigma\|_p,$$

where $(\tilde{P}^\sigma, \tilde{Q}^\sigma) = \sigma(\tilde{P}, \tilde{Q})$. Note that this distance also corresponds to a graphon-like approach [Lovász and Szegedy, 2006]. Indeed, if \tilde{P} and \tilde{Q} are transformed into

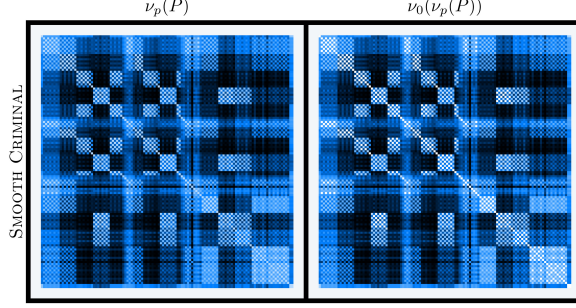


Figure 8. Comparison of $\nu_p(P)$ and $\nu_0(\nu_p(P))$ for the pattern matrix P of the song MICHAEL JACKSON - SMOOTH CRIMINAL. The main difference between these two pictures is that the lightest colour on the left image is not exactly white, meaning that the smallest entries of the matrix are not exactly 0. Even though ν_0 does not greatly modify the matrix, the patterns appear slightly clearer on the right.

functions $f^{\tilde{P}}, f^{\tilde{Q}} : [0, 1]^2 \mapsto [0, 1]$ such that $f^{\tilde{P}}(x, y) = \tilde{P}_{\lceil xb \rceil, \lceil yb \rceil}$ and $f^{\tilde{Q}}(x, y) = \tilde{Q}_{\lceil xd \rceil, \lceil yd \rceil}$, then d_{true} is equal to their integral L_p distance

$$d_{\text{true}}(\tilde{P}, \tilde{Q}) = \left(\iint_{(x,y) \in [0,1]^2} |f^{\tilde{P}}(x, y) - f^{\tilde{Q}}(x, y)|^p dx dy \right)^{\frac{1}{p}}. \quad (2)$$

This approach reinforces the idea that this should be the true distance between pattern matrices.

The function d_{true} , although perfectly adapted to compute the distance between normalized pattern matrices, is not computationally feasible since it requires to use matrices of size $bd \times bd$; even if replacing bd by the lowest common multiple of b and d , the computation time is unrealistic (see the first column of Table 1). To avoid this computational problem, an approximation of d_{true} is used.

Fix an integer $s_0 \geq 1$ and let ν_{s_0} be defined as follows

$$\nu_{s_0} : \tilde{P} \mapsto (\tilde{P}_{\lceil bk/s_0 \rceil, \lceil b\ell/s_0 \rceil})_{1 \leq k, \ell \leq s_0}.$$

This function, similar to σ , creates a matrix of size $s_0 \times s_0$ whose entries are related to the entries of \tilde{P} . However, since s_0 might not be a multiple of b , the number of times each entry of \tilde{P} is repeated in $\nu_{s_0}(\tilde{P})$ is not necessarily constant, creating a small error in the representation of \tilde{P} . Define now d_{s_0} by

$$d_{s_0}(\tilde{P}, \tilde{Q}) = \|\nu_{s_0}(\tilde{P}) - \nu_{s_0}(\tilde{Q})\|_p.$$

Since $\nu_{s_0}(\tilde{P})$ slightly modifies the role of the entries of the matrix \tilde{P} , the distance d_{s_0} is not necessarily going to be equal to d_{true} . However, for any pair of matrices \tilde{P} and \tilde{Q} , the distance $d_{s_0}(\tilde{P}, \tilde{Q})$ converges to $d_{\text{true}}(\tilde{P}, \tilde{Q})$ as s_0 increases to infinity; if considering the integral definition of d_{true} as in (2), d_{s_0} actually corresponds to the Riemann approximation of d_{s_0} using squares of size $\frac{1}{s_0} \times \frac{1}{s_0}$.

Computing d_{s_0} requires using matrices of size $s_0 \times s_0$. This means that the smaller s_0 is, the faster the computation will be. Hence, it is interesting to choose a value for s_0 that balances between the computational time of d_{s_0} and the accuracy compared to the true distance d_{true} . Table 1 gives an approximation of the computational time and the accuracy for different values of s_0 . From this table, any s_0 larger than 100 is an

appropriate choice since the error rate is less than 1%. For the rest of this study, s_0 is set to be 500, corresponding to an error rate as small as computation time reasonably allows.

distance	d_{true}	d_{100}	d_{200}	d_{500}	d_{1000}	d_{2000}
error	0%	0.99%	0.50%	0.20%	0.10%	0.05%
time (ms)	6318	0.89	2.43	13.5	43.0	164

Table 1. Comparison of d_{true} with d_{s_0} for $s_0 \in \{100, 200, 500, 1000, 2000\}$. For a given distance d_{s_0} , its error corresponds to the average error rate between $d_{\text{true}}(\bar{P}, \bar{Q})$ and $d_{s_0}(\bar{P}, \bar{Q})$ (where the error rate between a and b is defined as $|a - b|/|a|$). For each distance d , its time corresponds to the average time to compute $d(\bar{P}, \bar{Q})$. All these results were approximated by randomly choosing 400 pairs of songs from the dataset and averaging their error rate and time to compute.

To conclude this section, ν is set to be the combination of the different normalization functions:

$$\nu : P \mapsto \nu_{500}(\nu_0(\nu_p(P))).$$

In other words, the image of a pattern matrix P through ν corresponds to transforming P using its percentiles (see Figure 7), then forcing its values to span 0 and 1 (see Figure 8), and finally resizing it to 500×500 . Using the image of ν as a normalized pattern matrix, the pattern structure of songs can be compared by defining the similarity distance on songs D as in (1).

4. Results

With the pattern matrix being defined in Section 2 and the distance D set in Section 3, songs can now be compared according to their pattern structures. In this section, the values given by D are combined with the features of the songs (defined in Section 1) to answer the previously asked questions:

- Given specific feature values, what can we say about the pattern structures?
- Given similar pattern structures, do they have corresponding feature values?

To answer these questions, two scores are defined on groups of songs: the *pattern variability* score (\mathcal{V}^P score) and the *feature variability* score (\mathcal{V}^F score).

Let $\mathcal{G} = \{\mathcal{S}_1, \dots, \mathcal{S}_k\}$ be a group of songs. The \mathcal{V}^P score of \mathcal{G} is defined as the average distance between any pair of songs of \mathcal{G} :

$$\mathcal{V}^P = \mathbb{E}_{\mathcal{S} \neq \tilde{\mathcal{S}} \in \mathcal{G}} [D(\mathcal{S}, \tilde{\mathcal{S}})] = \frac{1}{k(k-1)} \sum_{1 \leq i, j \leq k} D(\mathcal{S}_i, \mathcal{S}_j).$$

This score should be interpreted as the amount of variability between the patterns of the songs of this group: groups with a small \mathcal{V}^P score correspond to a set of songs whose patterns are similar³.

Given this same group of songs $\mathcal{G} = \{\mathcal{S}_1, \dots, \mathcal{S}_k\}$, let f_1, \dots, f_k be their feature values, corresponding to the feature F taken from the possible choices of features:

³ An easy way to better understand the \mathcal{V}^P score is by comparing the results of Figure 10 with that of Figure C1; indeed, the lowest scores in Figure C1 all correspond to covers of the same song and are thus lower than the lowest scores in Figure 10.

artist, title, year, decade, genre, or type. The $\mathcal{V}^{\mathcal{F}}$ score of \mathcal{G} (with regards to \mathbf{F}) is defined in one of the following two ways. If f_1, \dots, f_k are real numbers (meaning that $\mathbf{F} \in \{\text{year, decade}\}$), then the $\mathcal{V}^{\mathcal{F}}$ score is the standard deviation of the values f_1, \dots, f_k :

$$\mathcal{V}^{\mathcal{F}} = \sqrt{\frac{1}{k} \sum_{1 \leq i \leq k} (f_i - \mu)^2},$$

where $\mu = \frac{1}{k} \sum_{1 \leq i \leq k} f_i$. Otherwise, if f_1, \dots, f_k are not real numbers, then the $\mathcal{V}^{\mathcal{F}}$ score is defined as the average of the counting process of f_1, \dots, f_k . More precisely, let $(c_1, \tilde{f}_1), \dots, (c_\ell, \tilde{f}_\ell)$ be such that $c_j = \#\{i : f_i = \tilde{f}_j\}$, $c_1 \geq \dots \geq c_\ell$ and $\{f_1, \dots, f_k\} = \{\tilde{f}_1 \neq \dots \neq \tilde{f}_\ell\}$. Then, the $\mathcal{V}^{\mathcal{F}}$ score is defined as

$$\mathcal{V}^{\mathcal{F}} = \frac{1}{c_1 + \dots + c_\ell} \sum_{1 \leq j \leq \ell} j c_j.$$

This choice of score can be seen as transforming f_1, \dots, f_n into a set of probabilities $\{p_1, \dots, p_\ell\}$, where ℓ is the number of different feature values in $\{f_1, \dots, f_n\}$ and p_j is the probability of having the j -th most common feature value when sampling a random f_i . With this approach, $\mathcal{V}^{\mathcal{F}}$ then corresponds to the average popularity of the chosen index. For example, imagine having a group of 10 songs, and consider the three following cases along with their $\mathcal{V}^{\mathcal{F}}$ scores.

- All songs have the same feature value; in that case $\ell = 1$ and $c_1 = 10$, leading to $\mathcal{V}^{\mathcal{F}} = 1$.
- The songs take two possible feature values, separated into two groups of equal sizes; in that case $\ell = 2$ and $(c_1, c_2) = (5, 5)$, leading to $\mathcal{V}^{\mathcal{F}} = 1.5$.
- All songs have different feature values; in that case $\ell = 10$ and $c_i = 1$ for all $i \in \{1, \dots, 10\}$, leading to $\mathcal{V}^{\mathcal{F}} = 5.5$.

In both cases whether f_1, \dots, f_k are real numbers or not, the $\mathcal{V}^{\mathcal{F}}$ score should be interpreted as the amount of variability between the feature values of the songs of this group: groups with a small $\mathcal{V}^{\mathcal{F}}$ score correspond to a set of songs with a lot of similar features.

With the $\mathcal{V}^{\mathcal{P}}$ and the $\mathcal{V}^{\mathcal{F}}$ scores defined, the rest of this section is organized as follows. Section 4.1 uses a *feature-based approach* and studies the $\mathcal{V}^{\mathcal{P}}$ score when grouping songs according to their feature values. For each feature, this allows a ranking of the feature values according to their variability of patterns. Sections 4.2 and 4.3 create groups of songs according to the distance D and study the $\mathcal{V}^{\mathcal{F}}$ score of these groups. Section 4.2 uses a *cluster-based approach* and partitions songs into clusters, whereas Section 4.3 uses a *neighbour-based approach* and focuses on each song's nearest neighbours. These two approaches give similar results and are both efficient in finding underlying properties of features not observable with the approach of Section 4.1. Finally, some extra figures completing those presented in the following sections can be found in Appendix C

4.1. Pattern variability on features (feature-based approach)

Consider the different features of a song: **artist**, **title**, **year**, **decade**, **genre** and **type**. Each of these features has a set of possible values it can take; for example, all

songs of the dataset in this study appeared in the chart for the first time between 1958 and 2019. Now, for each feature value, all corresponding songs can be grouped together and their $\mathcal{V}^{\mathcal{P}}$ score can be computed. For example, the first 20 lowest $\mathcal{V}^{\mathcal{P}}$ scores for the feature `year` are represented in Figure 9.

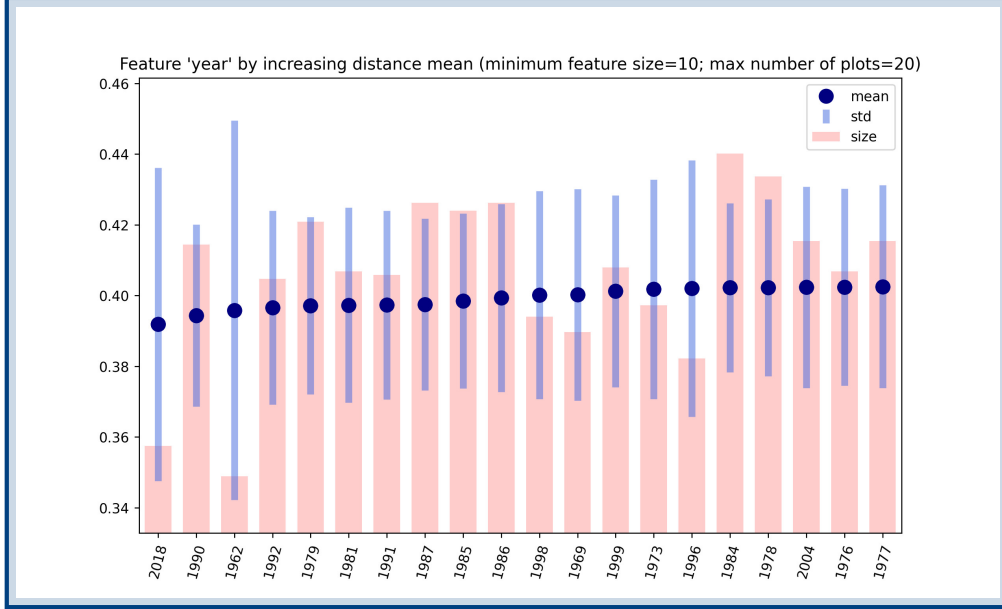


Figure 9. The year 1958 to 2019 ordered according to their $\mathcal{V}^{\mathcal{P}}$ score (only the first 20 lowest ones are depicted here). The blue dots correspond to the $\mathcal{V}^{\mathcal{P}}$ scores, which are obtained by computing the average distance between songs from a given year. The blue bars show the standard deviation around this average distance. The bars in red represent the number of songs who appeared for the first time the corresponding year. No year has a notably lower score than the next one, but this computation gives a ranking on years according to their variability in pattern structure.

The $\mathcal{V}^{\mathcal{P}}$ score of each feature value is a clear and easy-to-interpret metric. However, this score is difficult to combine with the actual pattern matrices, since it reflects a general trend rather than the existence of a unique pattern structure. Indeed, when the number of songs for each feature value is large, then the number of pattern structures tends to be large as well. Figure 10 illustrates this principle by representing the artists with at least 2, 5 and 10 songs and with minimal $\mathcal{V}^{\mathcal{P}}$ score. From this figure, one can see that, as the number of songs considered increases, the number of pattern structures also increases.

A possible explanation for the limited interpretation of $\mathcal{V}^{\mathcal{P}}$, is that it only focuses on the average distance between a given group of songs \mathcal{G} . As explained, this means that, as the size of the group increases, the average distance will converge to a standardized distance. Even without a group of large size, issues of interpretation can appear. For example, it is possible for \mathcal{G} to be composed of songs having one of two distinct pattern structures; in this case, even though the pattern variability of \mathcal{G} would be limited, since patterns of songs in \mathcal{G} could be explained by only two types of pictures, the $\mathcal{V}^{\mathcal{P}}$ score of \mathcal{G} would be high, because the average distance would be related to the distance between the two distinct patterns. In order to avoid such issues, Sections 4.2 and 4.3 focus on studying the variability in features from groups of songs with similar patterns, rather than considering all songs with the same feature value. These two sections give a better understanding of the structural properties of songs and their relation to features.

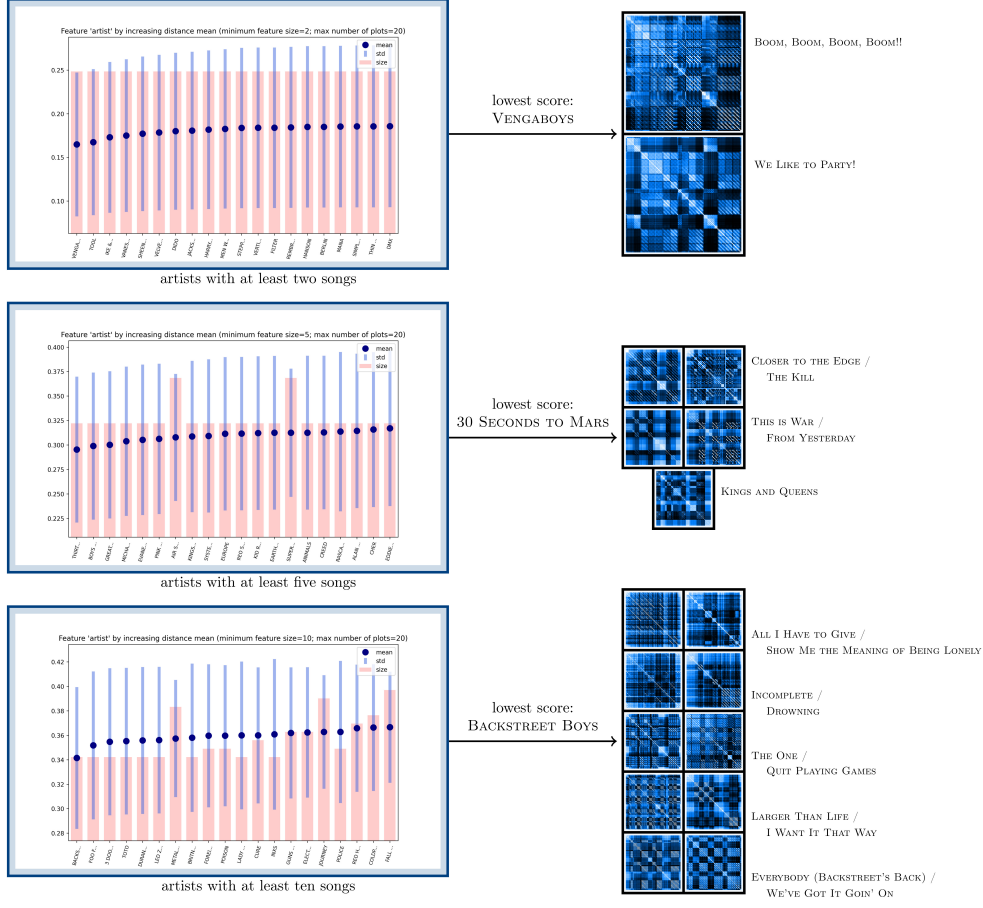


Figure 10. The artists with lowest \mathcal{V}^P score according to different threshold on their minimal number of songs. On the left, three figures correspond to the first 20 scores for the feature **artist**, where the minimal number of songs is set to 2, 5 and 10. On the right, all the normalized pattern matrices of each artist with lowest score. As the number of songs for each feature value increases, so does the variety of patterns, making it more difficult to explain the low score with the pattern matrices.

The \mathcal{V}^P score is limited in its interpretation and might miss interesting underlying properties of the relation between features and patterns. However, this score is a natural approach to studying pattern variability in songs and it gives a metric allowing the direct ranking of feature values. It might require some modification and improvement (see Section 6 for further ideas), but could be used as an interesting metric to study the evolution of variability through years, genres, or artists (see Appendix C for the complete set of figures regarding the \mathcal{V}^P score).

4.2. Feature variability on clusters (cluster-based approach)

After considering each feature independently and giving a score for each feature value according to pattern variability, this section applies first the distance on song D to create clusters. Once these clusters are identified, their \mathcal{V}^F score is computed, highlighting clusters with repeating feature values.

Since clusters can greatly vary according to the method used, three common clustering algorithms were implemented:

- Spectral Clustering [Ng et al., 2001];
- K-Medians [Park and Jun, 2009]; and
- Agglomerative Clustering [Ward Jr, 1963].

The choice of algorithms was motivated by two constraints. First, these algorithms can be directly applied to a pre-computed distance matrix obtained from D , and do not need to access the embedding space of the songs; this first reason explains why K-Medians was chosen over the more classical K-Means. Second, the number of output clusters is determined by a number given as input; this second reason explains why the classical DBSCAN algorithm is not used here. With these two constraints, Spectral Clustering and K-Medians are standard algorithms. The reason for using Agglomerative Clustering is that it fits the songs into a tree-like structure, which is a good way to interpret song patterns (see Figure 11). For this last reason, Agglomerative Clustering gives the best results and all further analysis in this section was obtained using this algorithm (see Appendix C for some results on Spectral Clustering and K-Medians).

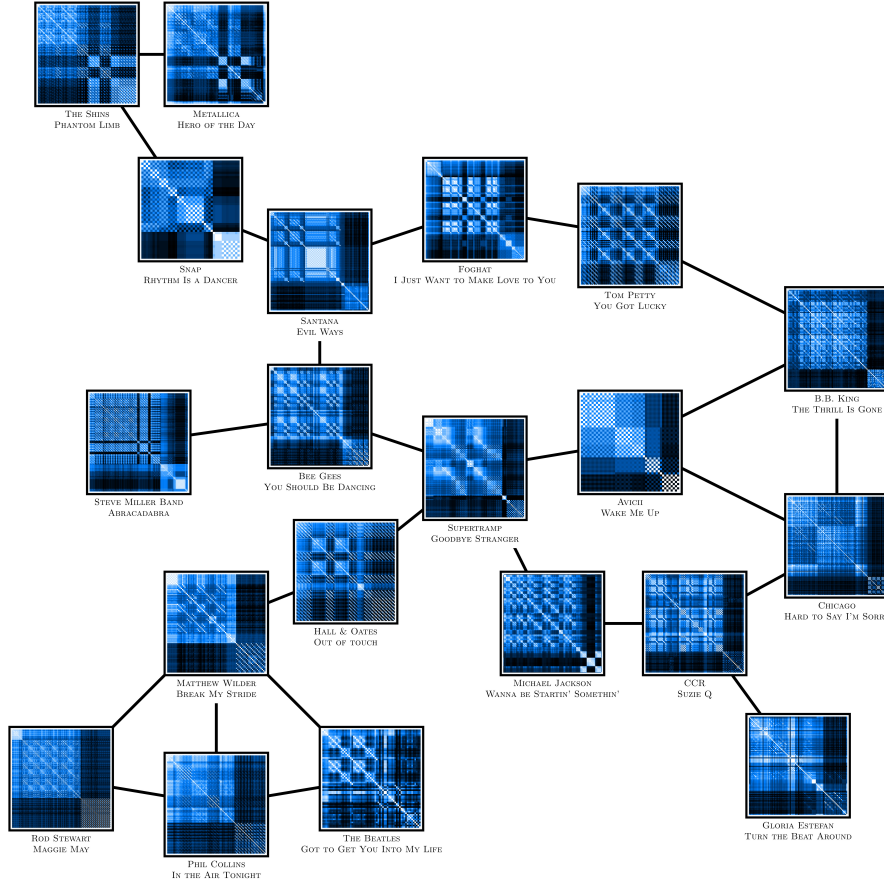


Figure 11. A representation of the tree-like structure of song patterns. This graph was obtained by connecting songs with their nearest neighbours. From this figure, one can observe that all of the above songs fall into the category of songs with a *high outro*, represented by darker bands at the bottom and at the right of the figures. From this general pattern structure category, sub-types of patterns can be observed and the connections between these patterns create a tree-like structure.

Since the dataset contains a large number of songs (4166 in total, see Appendix A), and the size of the clusters should be limited for interpretative reasons, a recursive

approach was used. The actual clustering algorithm is described in Algorithm 1 and is based on applying one of Agglomerative Clustering, Spectral Clustering, or K-Medians recursively, to reduce the clusters to a desirable size. The results that follow are all obtained by using Agglomerative Clustering (with $n_c = 2$ output clusters), no limit on the maximal number of iterations ($m_i = |\mathcal{G}|$), and clusters of size at most 15 ($m_s = 15$).

Algorithm 1: Recursive clustering algorithm

input : A group of songs \mathcal{G} and a metric D
output : A cluster partition $\mathcal{C} = \{C_1, \dots, C_k\}$ of \mathcal{G}
params:

- \mathcal{A}_C , a clustering algorithm, with a given number n_c of output clusters
- m_s , the maximal size of a cluster
- m_i , the maximal number of iterations

begin
 $\mathcal{C} \leftarrow \{\mathcal{G}\}$
for $i \leftarrow 1$ **to** m_i **do**
 $\mathcal{C}_{\text{aux}} \leftarrow \emptyset$
 for $C \in \mathcal{C}$ **do**
 if $|C| > m_s$ **then**
 $\mathcal{C}_{\text{aux}} \leftarrow \mathcal{C}_{\text{aux}} \cup \mathcal{A}_C(C, D)$
 /* $\mathcal{A}_C(C, D)$ partitions C in n_c clusters using D */
 else
 $\mathcal{C}_{\text{aux}} \leftarrow \mathcal{C}_{\text{aux}} \cup \{C\}$
 end
 end
 $\mathcal{C} \leftarrow \mathcal{C}_{\text{aux}}$
end
return \mathcal{C}
end

Once Algorithm 1 is applied to the set of songs, a similar approach to Section 4.1 is used and clusters are ranked according to their $\mathcal{V}^{\mathcal{F}}$ score. A typical outcome of this algorithm can be found in Figure 12, where clusters are ordered according to their $\mathcal{V}^{\mathcal{F}}$ score with regards to the feature **year**. Once this figure is obtained, it needs to be combined with an analysis of the clusters; the cluster with the lowest score in Figure 12 is represented in Figure 13.

Our interest in using clusters instead of the feature-based approach of Section 4.1 can be understood by considering the results of Figure 12. Indeed, the feature-based approach considers every feature value independently of each other and then ranks them according to their $\mathcal{V}^{\mathcal{P}}$ score. However, in the case of the feature **year**, for example, feature values can be compared with each other and a notion of proximity can be defined. Using this property, the cluster represented in Figure 13 can be found, where the pattern clearly corresponds to songs around the year 2010. This could not have been unveiled by the feature-based approach in Section 4.1 since it does not compare songs that have different but similar feature values.

The reason for using the cluster-based approach over the feature-based approach is not limited to the notion of proximity on the feature **year**. As explained in Section 4.1, a possible limitation of the $\mathcal{V}^{\mathcal{P}}$ score is that it is not able to detect groups of songs that would be generated from a couple of distinct pattern structures. Clustering methods,

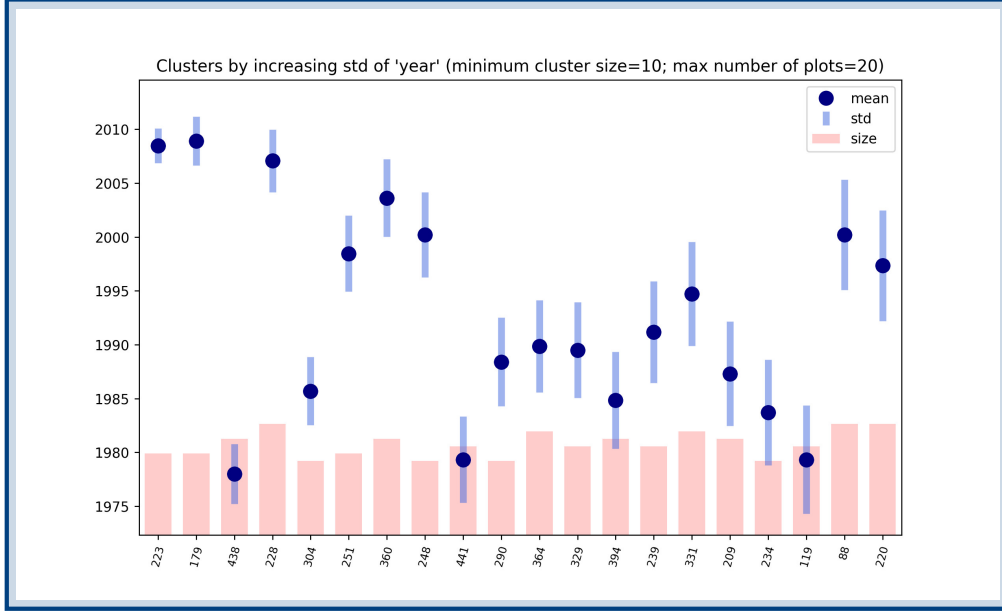


Figure 12. Clusters ordered according to their $\mathcal{V}^{\mathcal{F}}$ score with regards to the feature **year** (only depicting the 20 lowest scores). The blue dots represent the average year of the cluster, the blue bars represent the standard deviation around this year and the red bars represent the sizes of the clusters. This figure needs to be combined with a representation of the clusters in order to appreciate its results. The cluster with the lowest score is represented in Figure 13.

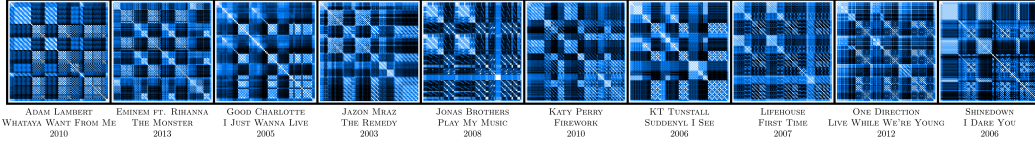


Figure 13. A representation of the cluster with lowest $\mathcal{V}^{\mathcal{F}}$ score as found in Figure 12. As one can see, all songs show a similar pattern and have been released around the same years. Since this cluster was obtained by grouping songs according to their pattern similarities, it also means that this specific pattern is characteristic of the end of the 2000's.

however, are able to separate songs according to pattern similarity and then identify groups with repeated feature values. An example of the power of clusters in identifying sub-groups of patterns inside a given feature value is represented in Figure 14, where the artist LINKIN PARK shows two commonly used structures. The reason this artist did not appear clearly in the results of Figure 10 is due to the two limitations of the $\mathcal{V}^{\mathcal{P}}$ score explained in Section 4.1: this artist has a lot of songs and a combination of different but repeated pattern structures.

Using the $\mathcal{V}^{\mathcal{F}}$ score on clusters shows more promising results than using the $\mathcal{V}^{\mathcal{P}}$ score on songs grouped by their feature values, since it is able to highlight finer properties of song patterns. One reason for this improvement is that the cluster-based approach is able to ignore noisy or out of distribution songs and only focuses on groups with similar structures. This leads to a better ability to identify specific pattern structures related to years (Figure 13) or artists (Figure 14). A few figures showing what happens when considering the features **genre** and **type** can be found in Appendix C.

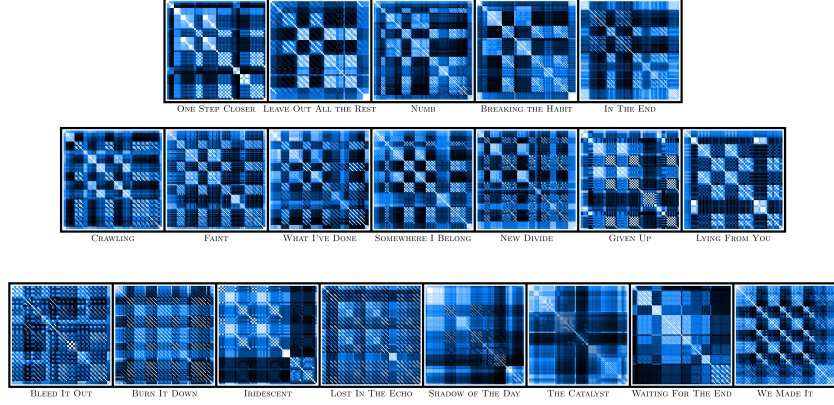


Figure 14. A representation of all the pattern matrices of the artist LINKIN PARK. The top two groups each correspond to a specific type of pattern and the bottom group contains the rest of the songs. This grouping shows that the artist LINKIN PARK commonly uses one of the two top structures. This property is highlighted by clusters but was missed when considering the \mathcal{V}^P score on the artist as shown in Figure 10. Two reasons can explain why this artist does not have a low \mathcal{V}^P score: the large number of songs it has in the dataset, and the fact that they mostly correspond to one of two distinct pattern structures.

4.3. Feature variability on neighbourhoods (neighbour-based approach)

The cluster-based approach used in Section 4.2 is useful since it creates a partition of the songs into groups. This approach also implies that groups with low \mathcal{V}^F scores can be interpreted as patterns characteristic of the given feature value (as explained in Figure 13). However, another possible approach to grouping songs is to consider the set of nearest neighbours for each song.

For each song in the dataset, using the distance D it is possible to identify its neighbourhood, hence creating groups of songs with a centre. With this approach, some songs might appear in more neighbourhoods than others. This remark implies that neighbourhoods cannot be interpreted as characteristic groups, as was the case with clusters; however, this technique allows the definition of a centre song, opening more possibilities with the analysis of features.

Consider computing the 20 nearest neighbours of all the songs. For each of these neighbourhoods, the corresponding \mathcal{V}^F score can be computed as in Section 4.2. Furthermore, since neighbourhoods have a centre song, it is now possible to compare the feature of the centre with its neighbours. By applying this idea to the neighbourhoods ordered according to their \mathcal{V}^F score with regards to the feature **year**, Figure 15 is obtained and compares the neighbours and the centre song years of first appearance in the chart.

An interesting analysis of the feature **year**, made possible by neighbourhoods, can be done by classifying song patterns as *early*, *late* or *on-time* (see Figure 15). By considering neighbourhoods with the lowest \mathcal{V}^F with regards to the feature **year**, meaning neighbourhoods with the lowest year standard deviation, and comparing the average year of the neighbourhood to the year of the centre song, three behaviours can be identified. Songs can either have pattern structures similar to songs appearing later in the chart, meaning that this pattern was ahead of its time. Similarly, songs can have pattern structures that were commonly used in earlier years, meaning that they could have been inspired by previously released songs. Finally, songs can have pattern structures similar to other songs of the same year. In this last case, it is possible that the cluster-based approach of Section 4.2 would have already identified

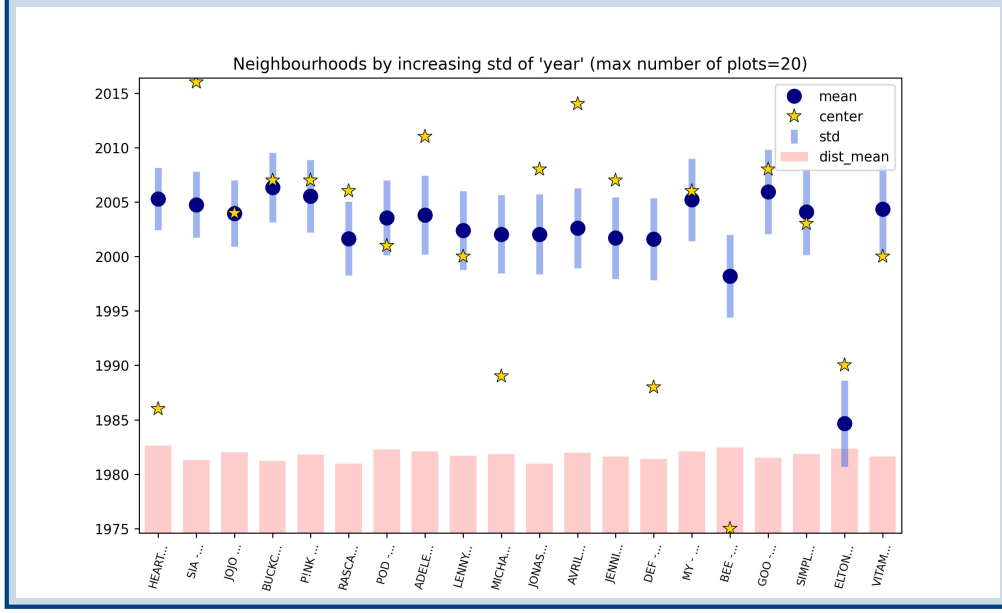


Figure 15. Neighbourhoods ordered according to their $\mathcal{V}^{\mathcal{F}}$ score with regards to the feature **year**. The blue dots, blue bars, and red bars respectively represent the average year, the deviation around this average year, and the average distance between songs of the neighbourhoods. The yellow star represents the year of the centre song. This approach allows the identification of songs with early use of patterns (such as the first song, HEART - IF LOOKS COULD KILL), late use of patterns (such as the second song, SIA - THE GREATEST), or on-time use of patterns (such as the third song, JOJO - LEAVE (GET OUT)). A few neighbours of HEART - IF LOOKS COULD KILL are represented in Figure 16.

this relation between pattern and years, whereas the first two cases would tend to be hidden when studying the $\mathcal{V}^{\mathcal{F}}$ score of the clusters. Two interesting examples of a song with early and late pattern structures can be found in Figures 16 and 17, where both the features **year** and **genre** have interesting behaviours. Indeed, in the case of Figure 16, the pattern is later borrowed by the neighbours of the HEART song and can be interpreted as a revival of the original song style, whereas in the case of Figure 17, the strong difference in both years and genre shows the originality of the structure of the THE CHAINSMOKERS song.

Overall, the cluster-based approach of Section 4.2 and the neighbour-based approach of this section give similar and complementary results. While the cluster-based approach highlights characteristic patterns corresponding to feature values, the neighbourhood approach can be used to find outliers in pattern structures by showing songs whose features are unrelated to those of their neighbours. These two techniques can also be combined to help understand the general structure of the song patterns. Indeed, by comparing clusters and neighbourhoods, one can observe a possible tree-like structure for song patterns, as highlighted in Figure 11. This might also explain why the Agglomerative Clustering algorithm shows better results than the two other clustering techniques considered.

5. Evaluation

In order to complete this study, we evaluate the ability of the metric D to classify songs. More precisely, we implement various classical regression and classification algorithms

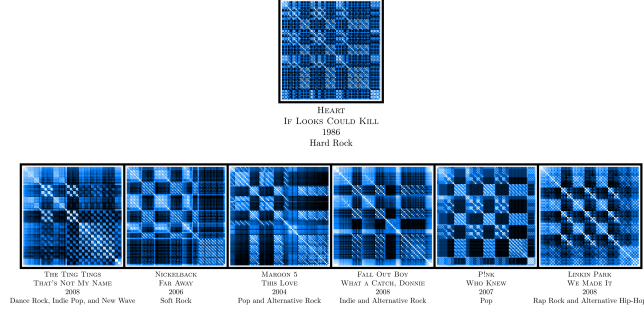


Figure 16. The representation of the song HEART - IF LOOKS COULD KILL, as well as a few chosen neighbours. As it appears from this figure, the pattern shown by HEART - IF LOOKS COULD KILL is borrowed by songs released years later. Moreover, it is interesting to notice that the songs represented below all fall under some genre related to rock and could thus be interpreted as a revival of the original hard rock structure found in the HEART song.

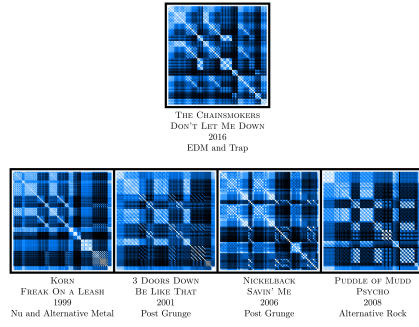


Figure 17. The representation of the song THE CHAINSMOKERS - DON'T LET ME DOWN, as well as a few chosen neighbours. As it appears from this figure, the pattern shown by THE CHAINSMOKERS - DON'T LET ME DOWN is taken from an early period. Moreover, when comparing the genre of these different songs, it also appears as THE CHAINSMOKERS - DON'T LET ME DOWN does not fit the genres of its neighbours. This is an interesting example of song whose patterns is borrowed from another period of time and style of music.

to test whether D can be used to predict the features **year**, **decade**, **artist**, **genre**, and **type**.

The problem we are trying to solve can be formally stated as follows. First of all, we have a set of songs \mathcal{G} which we divide into two sets: $\mathcal{G}_{\text{train}}$ for the train set and $\mathcal{G}_{\text{test}}$ for the test set. In our case, we randomly split \mathcal{G} so that $|\mathcal{G}_{\text{train}}| \simeq 4|\mathcal{G}_{\text{test}}|$, meaning that the train set is 80% of the dataset and the test set corresponds to the remaining 20%. Second, we have a distance D defined on \mathcal{G} and, for each song $S \in \mathcal{G}$, we have a feature f_S (we later explain how the different formats of features are handled). Our goal is to define a classification function $\varphi(\cdot \mid \mathcal{G}_{\text{train}}, D)$ which is constructed using $(\mathcal{G}_{\text{train}}, D)$ and so that

$$\varphi(\mathcal{S} \mid \mathcal{G}_{\text{train}}, D) \simeq f_S.$$

The meaning of the \simeq will be later clarified, but should be understood as φ correctly classifies \mathcal{S} as having the feature f_S . We now describe the different algorithms considered to define φ .

All of the regression and classification algorithms considered here come from the scikit-learn [Pedregosa et al., 2011] library in Python. In particular, we use the following algorithms.

- **LinearRegression** (LR), from the `linear_model` module, computing a simple linear regression;
- **KNeighborsRegressor** (KNR) and **KNeighborsClassifier** (KNC), from the `neighbors` module, computing a nearest neighbour regression and classification;
- **SVR** (SVR) and **SVC** (SVC), from the `svm` module, computing a support vector regression and classification;
- **GaussianNB** (GNB) and **BernoulliNB** (BNB), from the `naive_bayes` module, computing two types of naive Bayes' estimations; and
- **DecisionTreeRegressor** (DTR) and **DecisionTreeClassifier** (DTC), from the `tree` module, computing a decision tree regression and classification.

We further test two versions of the nearest neighbour regression and classification, using either 10 neighbours (10NR and 10NC) or 20 neighbours (20NR and 20NC). The other parameters of the models were unchanged from their defaults features.

Note that only four of the aforementioned algorithms allow the input to be a pre-computed distance: SVR, SVC, KNR, and KNC. For the other five, we decided to emulate a vector space by splitting the train set $\mathcal{G}_{\text{train}}$ using a *basis* of songs. Let $\mathcal{G}_{\text{basis}} \subseteq \mathcal{G}_{\text{train}}$ be a random subset of the training set. Then, we embed the remaining set of songs $\mathcal{G} \setminus \mathcal{G}_{\text{basis}}$ along with the distance D as a vector space of dimension $|\mathcal{G}_{\text{basis}}|$ by saying that the coordinates of $\mathcal{S} \in \mathcal{G} \setminus \mathcal{G}_{\text{basis}}$ are given by $(D(\mathcal{S}, \mathcal{S}_b))_{\mathcal{S}_b \in \mathcal{G}_{\text{basis}}}$. This thus allows us to apply the remaining algorithms, LR, GNB, BNB, DTR, and DTC, to this vector space. For the rest of this study, we set $\mathcal{G}_{\text{basis}}$ so that it corresponds to approximately 5% of the songs of the train set, chosen at random ⁴.

Since we want to test different formats for the features, numerical for **year** and **decade**, categorical for **artist**, and multi-categorical for **genre** and **type**, we need to set up different evaluation methods. In the case of numerical values, that is when $f_{\mathcal{S}}$ and φ both output numbers, we implement the two metrics

$$\text{AVG} = \frac{1}{|\mathcal{G}_{\text{test}}|} \sum_{\mathcal{S} \in \mathcal{G}_{\text{test}}} \left| f_{\mathcal{S}} - \varphi(\mathcal{S} \mid \mathcal{G}_{\text{train}}, d_{\mathcal{S}}) \right|,$$

for the average error in classification, and

$$\text{SUCC} = \frac{1}{|\mathcal{G}_{\text{test}}|} \sum_{\mathcal{S} \in \mathcal{G}_{\text{test}}} \delta_0 \left(f_{\mathcal{S}} - \varphi(\mathcal{S} \mid \mathcal{G}_{\text{train}}, d_{\mathcal{S}}) \right),$$

where $\delta_0(x) = 1$ if and only if $x = 0$, for the number of successes. It is worth noting that a good classification algorithm should output a low AVG and a high SUCC (in percentage). Running the previous algorithms on our dataset leads to Table 2.

As we can see in Table 2, the results show that D is not a good metric for characterizing the year or decade of a song. Indeed, on average it misclassifies the song by at least 10 years and does not classify songs substantially better than a simple random classification. We now provide the results for categorical feature values, that is for the feature **artist**. In this case, regression algorithms cannot be applied and the AVG score defined before is not properly defined anymore. However, the SUCC score still makes sense in this context and we thus provide the results of the classification algorithm on the feature **artist** in Table 3.

⁴ Some extra experiments were implemented using different percentages of the train set for the basis set but mostly provided similar results and we thus decided not to include them for clarity.

	year		decade	
	AVG	SUCC	AVG	SUCC
LR [†]	12.04	1.92%	12.24	22.18%
10NR* [†]	12.81	1.80%	13.03	19.54%
10NC*	17.09	2.76%	14.46	24.94%
20NR* [†]	12.72	1.32%	12.92	19.90%
20NC*	14.72	3.36%	13.84	25.54%
SVR* [†]	15.59	1.92%	15.82	18.35%
SVC*	19.36	2.76%	19.47	12.71%
GNB	18.19	2.28%	17.94	18.11%
BNB	19.30	3.00%	17.10	20.50%
DTR [†]	17.27	2.40%	16.47	20.62%
DTC	17.51	2.64%	16.75	21.46%

Table 2. The classification results for the two numerical features, **year** and **decade**. The algorithms marked with * directly use D as metric while the other ones use the basis set $\mathcal{G}_{\text{basis}}$ to transform the dataset into a vector space. The algorithms marked with [†] are regression algorithms, while the other ones are classification algorithms. As one would expect, the results between the features **year** and **decade** are strongly correlated.

	artist
10NC*	0.24%
20NC*	0.12%
SVC*	0.48%
GNB	0.84%
BNB	1.08%
DTC	0.00%

Table 3. The classification results for the categorical feature **artist** with respect to the SUCC score. The algorithms marked with * directly use D as metric while the other ones use the basis set $\mathcal{G}_{\text{basis}}$ to transform the dataset into a vector space. The results are extremely low, further highlighting some of the observations from Section 4.1 that D fails at characterizing large groups of songs and better highlights singular behaviours.

As we expected given the observations from Section 4.1, and in particular the observations made in Figure 10, the metric D does not properly encapsulate the properties of the artists to characterize them. Furthermore, in the case of the **artist** features, since it is very diverse (1431 different feature values out of the 4166 entries of the dataset), one cannot expect the training set to cover enough information for the algorithm to be able to properly classify songs of the test set. For this feature, we are actually in the case of the *curse of dimensionality* and would likely need to greatly increase the size of the dataset while keeping the same number of artists to have a chance to solve this problem. We now conclude this section with the evaluation strategy for multi-categorical feature values.

Recall that, in the case of the features **genre** and **type**, \mathbf{f}_S and φ are both lists of categorical values. To represent this, we identify \mathbf{f}_S and the output of φ with a distribution on \mathcal{F} , where \mathcal{F} is the set of feature values (for example ‘*alternative rock*’, ‘*brit pop*’, and ‘*gangsta rap*’ for **genre** and ‘*rock*’, ‘*alternative*’, and ‘*art*’ for **type**). More precisely, $\mathbf{f}_S = (\mathbf{f}_S[i])_{i \in \mathcal{F}} \in [0, 1]^{\mathcal{F}}$ is defined by

$$\mathbf{f}_S[i] = \begin{cases} \frac{1}{K} & \text{if } i \text{ is one of the } K \text{ feature values of } \mathcal{S} \\ 0 & \text{otherwise.} \end{cases}$$

In particular, this implies that

$$\sum_{i \in \mathcal{F}} \mathbf{f}_S[i] = 1.$$

A similar definition is used for $\varphi(\mathcal{S} \mid \mathcal{G}_{\text{train}}, D) = (\varphi(\mathcal{S})[i])_{i \in \mathcal{F}}$. In other words, we see $f_{\mathcal{S}}$ and $\varphi(\mathcal{S} \mid \mathcal{G}_{\text{train}}, D)$ as measures of probability on \mathcal{F} corresponding to the odds of \mathcal{S} to take a specific feature value. We now consider the following evaluation metrics for multi-categorical features. Write δ_+ for the function such that $\delta_+(x) = 1$ if and only if $x > 0$.

- The correctness, corresponding to the fact that any one of the guessed feature in φ is correct:

$$\text{CORR} = \frac{1}{|\mathcal{G}_{\text{test}}|} \sum_{\mathcal{S} \in \mathcal{G}_{\text{test}}} \delta_+ \left(\sum_{i \in \mathcal{F}} f_{\mathcal{S}}[i] \cdot \varphi(\mathcal{S})[i] \right);$$

- The precision, corresponding to the ratio of correctly guessed features of \mathcal{S} given φ :

$$\text{PREC} = \frac{1}{|\mathcal{G}_{\text{test}}|} \sum_{\mathcal{S} \in \mathcal{G}_{\text{test}}} \sum_{i \in \mathcal{F}} f_{\mathcal{S}}[i] \cdot \delta_+(\varphi(\mathcal{S})[i]);$$

- The accuracy, corresponding to the probability of correctly guessing a feature of \mathcal{S} using φ as a probability distribution:

$$\text{ACC} = \frac{1}{|\mathcal{G}_{\text{test}}|} \sum_{\mathcal{S} \in \mathcal{G}_{\text{test}}} \sum_{i \in \mathcal{F}} \delta_+(f_{\mathcal{S}}[i]) \cdot \varphi(\mathcal{S})[i]; \text{ and}$$

- The exact matching, corresponding to the number of features exactly evaluated by φ :

$$\text{MATCH} = \frac{1}{|\mathcal{G}_{\text{test}}|} \sum_{\mathcal{S} \in \mathcal{G}_{\text{test}}} \sum_{i \in \mathcal{F}} \delta_0(f_{\mathcal{S}}[i] - \varphi(\mathcal{S})[i]).$$

To understand better these metrics, let us consider a couple of specific cases.

- Imagine that φ always guesses that each song has all possible feature values: $\varphi(\mathcal{S})[i] = 1/|\mathcal{F}|$. In that case, both correctness and precision would be 100%, but the accuracy would be rather low (related to the average number of features per song) and the exact matching likely 0% (except if there exist songs with all features).
- Imagine that φ always guesses a single and correct feature: $\varphi(\mathcal{S})[i] = 1$ for some i such that $f_{\mathcal{S}}[i] > 0$. In that case, the correctness and accuracy would be 100%, the precision would be the average of the inverse number of features of a song (for example, if all songs have exactly 2 feature values, then it would equal 50%) and the exact matching would equal the percentage of songs having a single feature.

With that in mind, a high CORR or PREC score should be evaluated with the corresponding ACC and MATCH scores: if the latter ones are low this only means that the classification function gives a lot of features for each song. The evaluation of d_s with regards to these different metrics on **genre** and **type** can be found in Table 4.

Observing the results from Tables 2, 3, and 4, it is clear that D does not provide good results in terms of classification. Indeed, in all the considered cases, the results do

	genre				types			
	CORR	PREC	ACC	MATCH	CORR	PREC	ACC	MATCH
10NC*	0.12%	0.12%	0.12%	0.60%	48.32%	22.88%	45.20%	6.95%
20NC*	0.00%	0.00%	0.00%	0.48%	54.80%	25.42%	51.50%	7.79%
SVC*	6.35%	4.10%	2.14%	0.36%	51.44%	24.22%	43.74%	6.12%
GNB	69.42%	57.37%	2.26%	0.00%	87.29%	70.39%	5.92%	0.00%
BNB	0.00%	0.00%	0.00%	0.48%	58.87%	26.77%	58.87%	9.35%
DTC	15.83%	9.47%	6.40%	0.60%	54.08%	28.49%	25.04%	1.08%

Table 4. The classification results for the two multi-categorical features, **genre** and **type**. The algorithms marked with * directly use D as metric while the other ones use a basis set to transform the dataset into a vector space. One can observe that the results between **genre** and **type** are not as correlated as that of **year** and **decade** in Table 2; this is not so surprising since the relation between **genre** and **type** is not as straightforward as that of **year** and **decade**. Once again, these results are not notably better than random guesses and the high CORR and PREC scores of GNB likely mean that this algorithms guesses a lot of features, as explained before.

not substantially improve a simple random guess. However, this should not fully come as a surprise: D was designed to extract anomalous behaviours from a song dataset and did not intend to provide global properties.

The low classification scores of D can actually be related to the results from Section 4.1. Indeed, we observed there that D did not provide clear rankings of the features and explained that, when considering many songs, the pattern matrices would cover a wide range of behaviours and would thus cancel the effect of D . Thus, in order for D to show its true potential, one need to more precisely analyse relations between smaller groups; we were able to do so in Section 4.2 and 4.3, where we found interesting and surprising relations between specific features and patterns. It is then not surprising that these specific and outlying behaviours do not show in the results of Tables 2, 3, and 4.

6. Discussion

Embedding songs into 2-dimensional similarity matrices is a powerful tool for modelling their underlying structure. Moreover, this embedding can be applied to compare songs with each other according to their pattern structures. This approach allows the definition of precise metrics on groups of songs which helps to unveil underlying commonalities amongst artists, years, or genres. By defining the similarity matrices and the distance on these matrices, one can observe interesting repetitions and commonly used structures over a large set of songs.

Outcome of present study

A common approach to group songs is according to some known information, such as by artist, or year, before comparing their pattern structures. This leads to the definition of the $\mathcal{V}^{\mathcal{F}}$ score and the study of Section 4.1. Although theoretically interesting, this method shows limitations due to the inevitable variety of structures which appear when comparing multiple songs (see the analysis made in Figure 10). To minimize such problems, two possible improvements on the $\mathcal{V}^{\mathcal{F}}$ score could be implemented.

The first improvement would be to normalize all groups to the same size. This could be done either by changing the dataset directly or by computing differently the $\mathcal{V}^{\mathcal{F}}$ score: for example, instead of summing the average distance over all songs, just consider the N closest songs, with N being fixed over all groups. If N is well-chosen,

this could balance the bias towards small groups, observed in Figure 10, by making large groups more likely to have a small $\mathcal{V}^{\mathcal{P}}$ score. However, this would tend to ignore sub-groups of songs, making this analysis less objective.

The second improvement would be to combine these groups with a clustering algorithm in order to reduce the noise created by outliers. In that sense, this method would be more similar to the cluster-based approach of Section 4.2, but where all the clusters must have the same feature values, and would thus likely only provide small improvements.

The limitation of the feature-based approach, that is sorting features according to their $\mathcal{V}^{\mathcal{F}}$ score, pushed for further studies on the relation between D and the song features. This led to the cluster-based approach of Section 4.2 and the neighbour-based approach of Section 4.3. By first using D to create relevant groups of songs and then computing their $\mathcal{V}^{\mathcal{F}}$ score, it is possible to highlight inherent properties that were not observable when simply grouping songs according to their feature values (see Figures 13 and 14). Both of these two grouping methods showed similar results as they were able to find groups of songs with repeating feature values that were previously missed. These two methods also complete each other. Since the cluster-based approach creates a partition of the songs, each group can be defined as the representation of a specific pattern structure. Hence, by finding groups with similar feature values, these groups can be interpreted as a typical structure used for this feature value and not elsewhere. Conversely, the neighbour-based approach tends to put some songs into more neighbourhoods than others and cannot be interpreted as typical for the structures. However, it allows the comparison of a song with other similar ones and is able to identify songs with unexpected patterns (see Figures 16 and 17).

Possible follow-up analysis

While the novelty of this study was mainly based on comparing similarity matrices of songs, the full use of the information available in standard symbolic music notation could open the door to further development in music analysis. First, new types of similarity matrices could be defined. For example, if instead of comparing the notes played between bars, the representation was using the position of the notes in the current chord, then the similarity matrices could contain some finer information on the evolution of the song. In this case, a simple pattern being repeated on different scales could be discovered, and this could lead to a better definition of pattern matrices. Second, the full use of standard music notation could also be useful when comparing different songs. Either by directly comparing what is being played or by comparing chord progressions, a new distance on songs D could be defined, not directly related to the pattern structure but rather to the composition structure. Finally, by only considering a subset of the instruments of the song, this method could be used to characterize songs according to specific metrics (for example the drum patterns, or the rhythmic patterns). Using this approach could also lead to representing songs with multiple similarity matrices, according to the different instruments, and then using more complex statistical tools to study their properties.

This study opens the door to further development in music structure analysis, especially in the field of similarity matrix representation. With our newly created dataset, made of standard music notations of songs, new types of similarity matrices can be defined, for example by borrowing methods from music theory. The novel approach of this study, based on comparing similarity matrices rather than analyzing them indi-

vidually, also creates new tools for the analysis of song structure. It is now possible to classify a large number of songs by similarity of patterns and to highlight repeated structures used by artists, years, or genres.

Finally, the grouping methods applied above were able to extract certain groups of songs with interesting behaviour. The outcome of this project could thus be combined with a finer analysis of the properties of some of these songs. For example, one could study in more details the full discography of the artist LINKIN PARK, as it appears to often use similar structures (see Figure 14), or compare other properties of the songs from Figure 16 and 17, such as their chord progression or the structure of the different instruments.

Acknowledgements

This project all started from a discussion with J. Barrett on basic drum beats, and his input made me question the structure of popular songs, which lead to this study. I also wish to thank Dr. Addario-Berry for his support on this project, as well as his two colleagues, Dr. Ouzounian and Dr. Hasegawa, for their input in creating the background of this study. I am also very grateful to N. Kauf, T. Voirand, and T. Stokes for proofreading this work and for their advice. Finally, I would like to thank two anonymous referees whose inputs completed the study presented here and made the paper as it is now.

This project has received funding from the Institut des Sciences Mathématiques (ISM) and the European Union’s Horizon 2020 Research and Innovation Programme under the Marie Skłodowska-Curie Grant Agreement No. 101034253.

References

- [Abakumov,] Abakumov, S. PyGuitarPro. <https://github.com/Perlence/PyGuitarPro>.
- [Bartsch and Wakefield, 2001] Bartsch, M. A. and Wakefield, G. H. (2001). To catch a chorus: Using chroma-based representations for audio thumbnailing. In *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No. 01TH8575)*, pages 15–18. IEEE.
- [Bartsch and Wakefield, 2005] Bartsch, M. A. and Wakefield, G. H. (2005). Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*, 7(1):96–104.
- [Billboard,] Billboard. Billboard Hot 100. <https://www.billboard.com/charts/hot-100>.
- [Buisson et al., 2022] Buisson, M., Mcfee, B., Essid, S., and Crayencour, H. C. (2022). Learning multi-level representations for hierarchical music structure analysis. In *International Society for Music Information Retrieval (ISMIR)*.
- [Cambouropoulos and Widmer, 2000] Cambouropoulos, E. and Widmer, G. (2000). Automated motivic analysis via melodic clustering. *Journal of New Music Research*, 29(4):303–317.
- [Cope, 2009] Cope, D. (2009). *Hidden structure: music analysis using computers*, volume 23. AR Editions, Inc.
- [Corsini,] Corsini, B. Music Patterns. <https://github.com/BenoitCorsini/music-patterns>.

- [de Clercq, 2012] de Clercq, T. (2012). *Sections and successions in successful songs: a prototype approach to form in rock music*. University of Rochester.
- [de Clercq, 2017] de Clercq, T. (2017). Interactions between harmony and form in a corpus of rock music. *Journal of Music Theory*, 61(2):143–170.
- [Eerola et al., 2001] Eerola, T., Järvinen, T., Louhivuori, J., and Toiviainen, P. (2001). Statistical features and perceived similarity of folk melodies. *Music Perception*, 18(3):275–296.
- [Epstein, 1986] Epstein, P. (1986). Pattern structure and process in Steve Reich’s “Piano Phase”. *The Musical Quarterly*, 72(4):494–502.
- [Foote, 1999] Foote, J. (1999). Visualizing music and audio using self-similarity. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 77–80.
- [Foote, 2000] Foote, J. (2000). Automatic audio segmentation using a measure of audio novelty. In *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No. 00TH8532)*, volume 1, pages 452–455. IEEE.
- [Foote and Uchihashi, 2001] Foote, J. and Uchihashi, S. (2001). The beat spectrum: A new approach to rhythm analysis. In *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.*, pages 224–224. IEEE Computer Society.
- [Guitar Pro,] Guitar Pro. Software. <https://www.guitar-pro.com/>.
- [Harkleroad, 2006] Harkleroad, L. (2006). *The math behind the music*. Cambridge University Press.
- [Jiang et al., 2022] Jiang, J., Chin, D., Zhang, Y., and Xia, G. (2022). Learning hierarchical metrical structure beyond measures. In *Ismir 2022 Hybrid Conference*.
- [Lovász and Szegedy, 2006] Lovász, L. and Szegedy, B. (2006). Limits of dense graph sequences. *Journal of Combinatorial Theory, Series B*, 96(6):933–957.
- [Miller,] Miller, S. Billboard Hot weekly charts. <https://data.world/kcmillersean/billboard-hot-100-1958-2017>.
- [Ng et al., 2001] Ng, A., Jordan, M., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14:849–856.
- [Nieto et al., 2020] Nieto, O., Mysore, G. J., Wang, C.-i., Smith, J. B., Schlüter, J., Grill, T., and McFee, B. (2020). Audio-based music structure analysis: Current trends, open challenges, and applications. *Transactions of the International Society for Music Information Retrieval*, 3(1).
- [Osborn, 2013] Osborn, B. (2013). Subverting the verse—chorus paradigm: Terminally climactic forms in recent rock music. *Music Theory Spectrum*, 35(1):23–47.
- [Park and Jun, 2009] Park, H.-S. and Jun, C.-H. (2009). A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341.
- [Paulus and Klapuri, 2006] Paulus, J. and Klapuri, A. (2006). Music structure analysis by finding repeated parts. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 59–68.
- [Paulus and Klapuri, 2009] Paulus, J. and Klapuri, A. (2009). Music structure analysis using a probabilistic fitness measure and a greedy search algorithm. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(6):1159–1170.
- [Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

- [Pienimäki and Lemström, 2004] Pienimäki, A. and Lemström, K. (2004). Clustering symbolic music using paradigmatic and surface level analyses. In *Proc. 5th International Conference on Music Information Retrieval*.
- [Rafii and Pardo, 2011] Rafii, Z. and Pardo, B. (2011). A simple music/voice separation method based on the extraction of the repeating musical structure. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 221–224. IEEE.
- [Rafii and Pardo, 2012] Rafii, Z. and Pardo, B. (2012). Music/voice separation using the similarity matrix. In *ISMIR*, pages 583–588.
- [Salamon et al., 2021] Salamon, J., Nieto, O., and Bryan, N. J. (2021). Deep embeddings and section fusion improve music segmentation. In *ISMIR*, pages 594–601.
- [Silva et al., 2018] Silva, D. F., Yeh, C.-C. M., Zhu, Y., Batista, G. E., and Keogh, E. (2018). Fast similarity matrix profile for music analysis and exploration. *IEEE Transactions on Multimedia*, 21(1):29–38.
- [Simms, 1996] Simms, B. R. (1996). *Music of the twentieth century: Style and structure*. Cengage Learning.
- [Ullrich et al., 2014] Ullrich, K., Schlüter, J., and Grill, T. (2014). Boundary detection in music structure analysis using convolutional neural networks. In *ISMIR*, pages 417–422.
- [Ultimate Guitar,] Ultimate Guitar. <https://www.ultimate-guitar.com/>.
- [Wang et al., 2022a] Wang, J.-C., Hung, Y.-N., and Smith, J. B. (2022a). To catch a chorus, verse, intro, or anything else: Analyzing a song with structural functions. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 416–420. IEEE.
- [Wang et al., 2022b] Wang, J.-C., Smith, J. B. L., and Hung, Y.-N. (2022b). MuSFA: Improving music structural function analysis with partially labeled data. In *Late-breaking and demo session, ISMIR*, Bengaluru, India.
- [Wang et al., 2021] Wang, J.-C., Smith, J. B. L., Lu, W.-T., and Song, X. (2021). Supervised metric learning for music structure features. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 730–737, Online.
- [Ward Jr, 1963] Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.
- [Wikipedia,] Wikipedia. https://en.wikipedia.org/wiki/Main_Page.

Appendix A. Dataset properties

In this appendix, we provide some extra information and properties regarding the dataset created for this work. We remind the reader that the code used for this study as well as more details on the dataset can be found in the dedicated Github repository [Corsi,]. The most general properties of the dataset are summarised in Table A1 and some further information on the features can be found in Table A2. Below, we explain the process of creating the feature type from genre.

#Songs	#Artists	#Features
4166	1431	6

Table A1. A summary of the statistics of the dataset used for this study.

Feature	artist	year	genre	type
Number of values	1431	62 (1958 - 2019)	473	120

Table A2. A summary of the statistics of the features.

Types and genres

To compute the list of feature values considered in **type**, we used the following strategy. First, each genre was split into its sequence of words: for example ‘*rock*’ would remain as ‘*rock*’, but ‘*pop rock*’ would lead to both ‘*pop*’ and ‘*rock*’. Then, all the output words from all genres were combined in a list. Finally, each word of that list contained in at least two different genres was added to the **type** feature values.

It is interesting to note that very few words appearing in the genres only belong to a single genre. For example, words such as ‘*avant*’, ‘*italo*’, or even ‘*swamp*’ all appear in two genres (respectively ‘*avant funk*’ and ‘*avant pop*’, ‘*italo dance*’ and ‘*italo disco*’, and ‘*swamp pop*’ and ‘*swamp rock*’), and thus belong to the list of types. However, some words such as ‘*sunshine*’ (from ‘*sunshine pop*’), ‘*viking*’ (from ‘*viking metal*’), or ‘*happy*’ (from ‘*happy hardcore*’) only belong to a single genre and thus do not belong to the list of types. Eventually, once this process of choosing words appearing in two different genres was done, some type names were combined, for clarity. For example ‘*electronic*’ was removed to only keep ‘*electro*’, ‘*doo*’ and ‘*wop*’ were combined as ‘*doo wop*’, ‘*prog*’ and ‘*progressive*’ were combined under the type ‘*progressive*’, and ‘*tex*’ was re-labelled ‘*texan*’. The 10 types with the most corresponding genres can be found in Table A3 while the full list of correspondence can be found in [Corsini, dataset/types.json].

Appendix B. Special cases of the bar distance

In this appendix, we provide a few special cases encountered when computing d_{bar} , the distance between two bars, introduced in Section 2. The three cases shown here are distance between chords (Figure B1), distance with the presence of a rest (Figure B2), and distance regarding onsets (Figure B3).

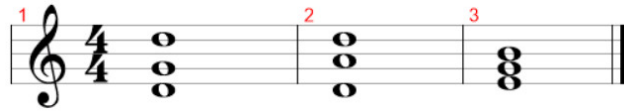


Figure B1. An example of three bars each playing a single chord. While the first two bars seem *more similar* and the two chords played only differ by one note, the distance d_{bar} between each pair of distinct bars will always be 2 here, since the algorithm simply consider the fact that the notes played are the same or not. A more subtle metric d_{bar} could be defined but would require to define a more complex distance between chords.

Appendix C. Extra figures

In this appendix, we provide some extra figures to complete the study presented in Section 4. We start by providing the first $\mathcal{V}^{\mathcal{P}}$ scores of the different features: **title** in Figure C1, **decade** in Figure C2, **genre** in Figure C3, and **type** in Figure C4. We recall that the first $\mathcal{V}^{\mathcal{P}}$ scores for **year** can be found in Figure 9 while those of **artist**

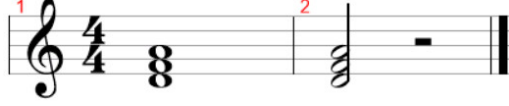


Figure B2. An example of two bars playing the same chord, but with the second one shortening the duration the chord is being played. The distance d_{bar} between these two bars is 1 since the onset of the two chords and the chords are the same, but the rest in the second bar creates a difference between them.



Figure B3. An example of two bars with the same chord played all along but with different onsets. The distance d_{bar} between these two bars is 3 since, even though the same note would be heard all along, the first bar has a unique onset and the second one has four onsets, on the tempo. Thus, the first onset is common to the two bars and does not increase d_{bar} , but the following three are only found in the second bar and each of them increments the distance by 1.

are showcased in Figure 10 with different limits on the minimal number of songs per artist.

After showing the $\mathcal{V}^{\mathcal{P}}$ score for all features, we now show the $\mathcal{V}^{\mathcal{F}}$ scores of our clustering methods (apart from the feature `title`, for which we do not have enough repetitions to have a meaningful interpretation of the clusters with a low $\mathcal{V}^{\mathcal{F}}$ score). While the first $\mathcal{V}^{\mathcal{F}}$ scores of the feature `year` can be found earlier, in Figure 13, we provide here the first $\mathcal{V}^{\mathcal{F}}$ scores of `artist` (in Figure C5), `genre` (in Figure C6), and `type` (in Figure C7). We further show the songs of cluster 471 of our algorithm, which appears as one of the lowest scores in all previous figures.

We conclude this appendix with a study on the use of the different clustering algorithms: Agglomerative Clustering, Spectral Clustering, and K-Medians. To do so, we ran two experiments:

- (1) we apply each of the clustering algorithms once and output 20 clusters.
- (2) we apply each clustering algorithm recursively, as described in Algorithm 1, using the aforementioned parameters ($n_c = 2$, $m_i = |\mathcal{G}|$, and $m_s = 15$).

The general statistics of the clusters in the first experiment are shown in Figure C9. As one can see, K-Medians seems to be the best algorithm to use in that case since it provides the most balanced clusters, and Spectral Clustering fails to extract relevant information since it basically splits the dataset into one giant cluster and 19 leftover clusters (in fact 17 of these clusters are singleton and do not show in Figure C9). The general statistics of the clusters in the second experiment are shown in Figure C10 and are less readable than in the previous case. However, it is worth noting that 419 out of 1047 clusters (40%) are singleton in the case of K-Medians, while only 48 out of 527 clusters (9%) are singleton in the case of Agglomerative Clustering. Spectral Clustering outputs 37 singletons out of a total of 516 clusters (7%).

Given the above results, the choice of Agglomerative Clustering came as a desire to balance the output of the two previous experiments and further made sense with regard to the tree-like structure of the songs as shown in Figure 11.

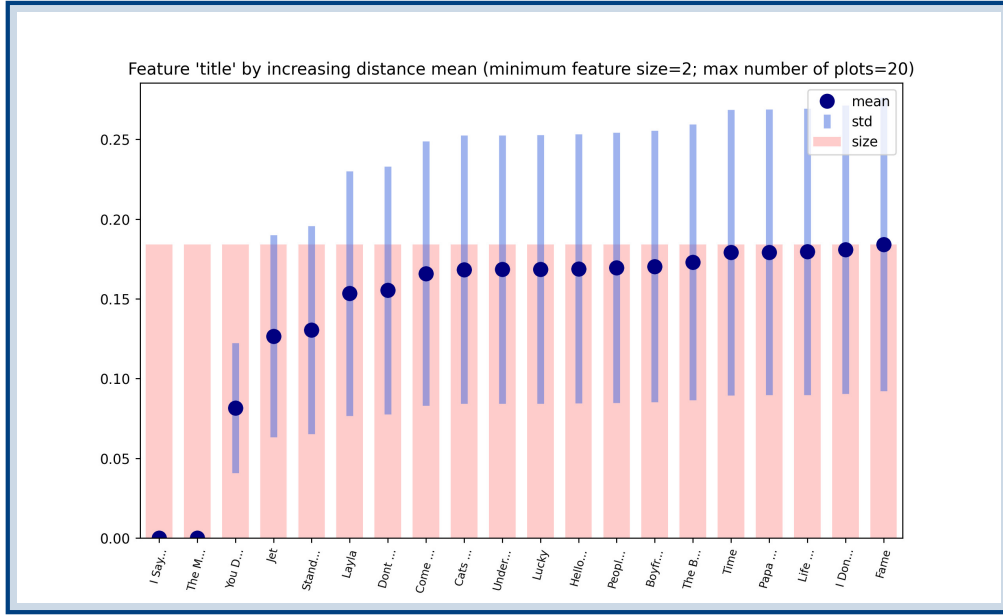


Figure C1. The titles ordered according to their $\mathcal{V}^{\mathcal{P}}$ score (only the first 20 lowest ones are depicted here). The blue dots correspond to the $\mathcal{V}^{\mathcal{P}}$ scores, which are obtained by computing the average distance between songs from a given decade. The blue bars show the standard deviation around this average distance. The bars in red represent the number of songs who appeared for the first time in the corresponding decade. One can see that all the lowest scores correspond to song covers, as to be expected.

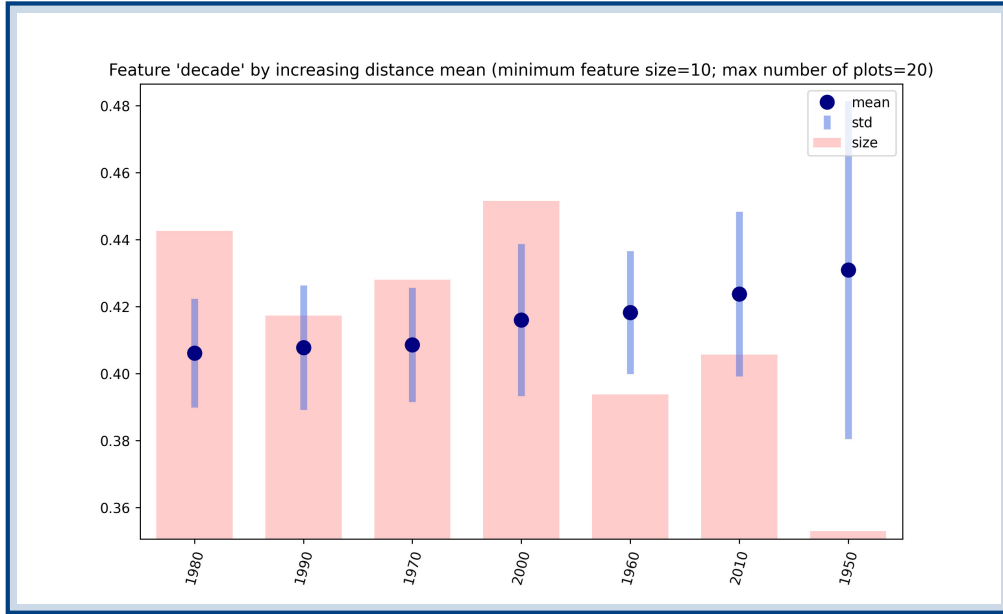


Figure C2. The seven decades available in the dataset ordered according to their $\mathcal{V}^{\mathcal{P}}$ score. The blue dots correspond to the $\mathcal{V}^{\mathcal{P}}$ scores, which are obtained by computing the average distance between songs from a given decade. The blue bars show the standard deviation around this average distance. The bars in red represent the number of songs who appeared for the first time in the corresponding decade. Although no decade has a notably lower score than the next one, the decades past the year 2000 have higher score, meaning that they have a (slightly) richer variety of patterns.

Types	List of corresponding genres (Total number)
rock	acid rock, acoustic rock, alternative rock, arena rock, art rock, baroque rock, blues rock, boogie rock, cello rock, celtic rock, chicano rock, christian alternative rock, christian rock, college rock, comedy rock, country rock, dance rock, disco rock, electro rock, electronic rock, experimental rock, folk rock, funk rock, garage rock, glam rock, gothic rock, grunge rock, hard rock, heartland rock, hot rod rock, indie rock, industrial rock, instrumental rock, jam rock, jazz rock, latin rock, neo progressive rock, new wave rock, noise rock, novelty rock, occult rock, piano rock, pop rock, power rock, progressive rock, psychedelic rock, pub rock, punk rock, raga rock, rap rock, reggae rock, rock, rock americana, rock and roll, rockabilly, rocksteady, roots rock, samba rock, shock rock, soft rock, soul rock, southern rock, space rock, stadium rock, stoner rock, surf rock, swamp rock, symphonic rock, synth rock, teen rock, texican rock and roll, western rockabilly, yacht rock (73)
pop	acoustic pop, adult pop, alternative pop, art pop, avant pop, baroque pop, bitpop, brit pop, chamber pop, christmas synth pop, contemporary pop, country pop, dance pop, disco pop, dream pop, electro pop, electronic pop, emo pop, euro pop, experimental pop, folk pop, funk pop, indie pop, j pop, jangle pop, jazz pop, k pop, latin pop, mod pop, neon pop, operatic pop, orchestral pop, piano pop, pop, pop beat, pop dance, pop disco, pop doo wop, pop funk, pop house, pop metal, pop punk, pop r&b, pop rap, pop reggae, pop rock, pop soul, pop trap, post brit pop, power pop, power pop lo fi, progressive pop, psychedelic pop, punk pop, r&b pop, rave pop, reggae pop, sophisti pop, soul pop, space age pop, sunshine pop, surf pop, swamp pop, symphonic pop, synth pop, techno pop, teen pop, traditional pop, trap pop, twee pop (70)
metal	alternative metal, cello metal, christian metal, comedy metal, doom metal, funk metal, glam metal, gothic metal, groove metal, hair metal, heavy metal, industrial metal, metalcore, nu metal, pop metal, power metal, progressive metal, proto metal, rap metal, sludge metal, speed metal, symphonic metal, thrash metal, viking metal (24)
rap	alternative rap, chicano rap, cloud rap, comedy rap, country rap, dirty rap, emo rap, gangsta rap, jazz rap, pop rap, pop trap, prog rap, punk rap, rap, rap folk, rap metal, rap rock, soundcloud rap (18)
soul	blue eyed soul, chicago soul, country soul, hip hop soul, memphis soul, neo soul, northern soul, philadelphia soul, pop soul, psychedelic soul, r&b soul, soul, soul blues, soul jazz, soul pop, soul r&b, soul rock, southern soul (18)
punk	art punk, dance punk, garage punk, hardcore punk, pop punk, post punk, progressive punk, proto punk, punk, punk blues, punk funk, punk pop, punk rap, punk rock, ska punk, skate punk, surf punk (17)
country	alternative country, bluegrass country, christian country, contemporary country, country, country blues, country folk, country pop, country rap, country rhythm and blues, country rock, country soul, country talking blues comedy, outlaw country, progressive country, tex mex country (16)
electro	electro, electro dance, electro funk, electro hop, electro house, electro pop, electro r&b, electro rock, electro swing, electronic, electronic dance, electronic pop, electronic rock, electronica, indie electronic (15)
funk	avant funk, disco funk, electro funk, funk, funk metal, funk pop, funk rock, funky house, g funk, jazz funk, latin funk, pop funk, psychedelic funk, punk funk, synth funk (15)
hip hop	acoustic hip hop, alternative hip hop, comedy hip hop, conscious hip hop, east coast hip hop, emo hip hop, hardcore hip hop, hip hop, hip hop soul, latin hip hop, old school hip hop, political hip hop, r&b hip hop, southern hip hop, west coast hip hop (15)

Table A3. The 10 types corresponding to the most genres in decreasing order. For a more complete list of types and genres correspondence, see [Corsini, , dataset/types.json].

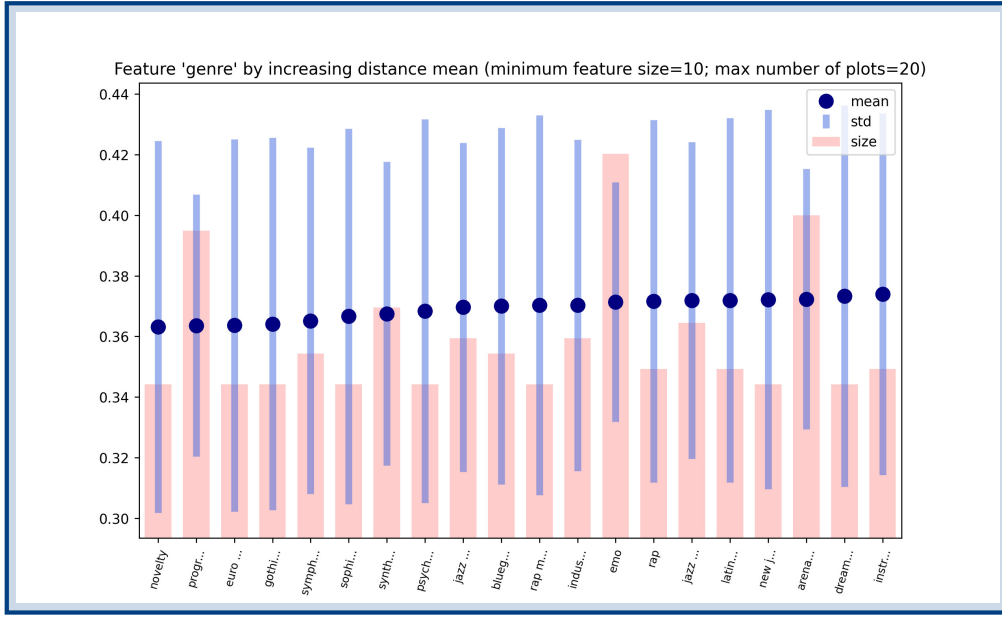


Figure C3. The genres ordered according to their \mathcal{V}^P score (only the first 20 lowest ones are depicted here). The blue dots correspond to the \mathcal{V}^P scores, which are obtained by computing the average distance between songs with a given genre. The blue bars show the standard deviation around this average distance. The bars in red represent the number of songs with the corresponding genre. It is worth observing that the second lower \mathcal{V}^P score, corresponding to the genre ‘*progressive pop*’, has a rather large size compared to the rest of the other genres with low score.

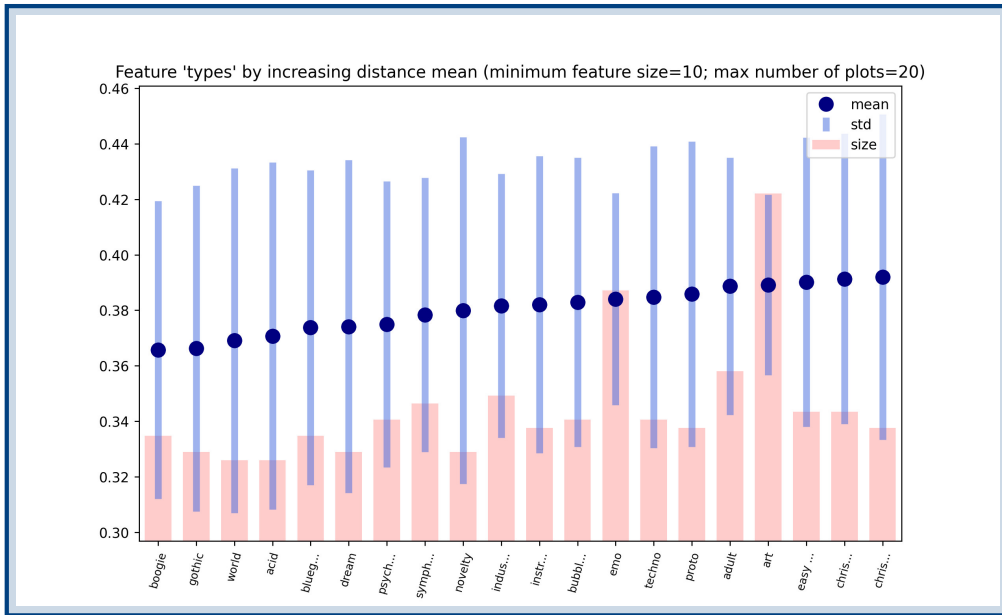


Figure C4. The types ordered according to their \mathcal{V}^P score (only the first 20 lowest ones are depicted here). The blue dots correspond to the \mathcal{V}^P scores, which are obtained by computing the average distance between songs with a given type. The blue bars show the standard deviation around this average distance. The bars in red represent the number of songs with the corresponding specific type. Interestingly, most types refer to genre adjectives (such as ‘*gothic*’, ‘*acid*’, ‘*novelty*’) rather than genre categories (such as ‘*rock*’ or ‘*rap*’) and two types appear to have a low score but a rather large size (compared to the rest): ‘*emo*’ and ‘*art*’.

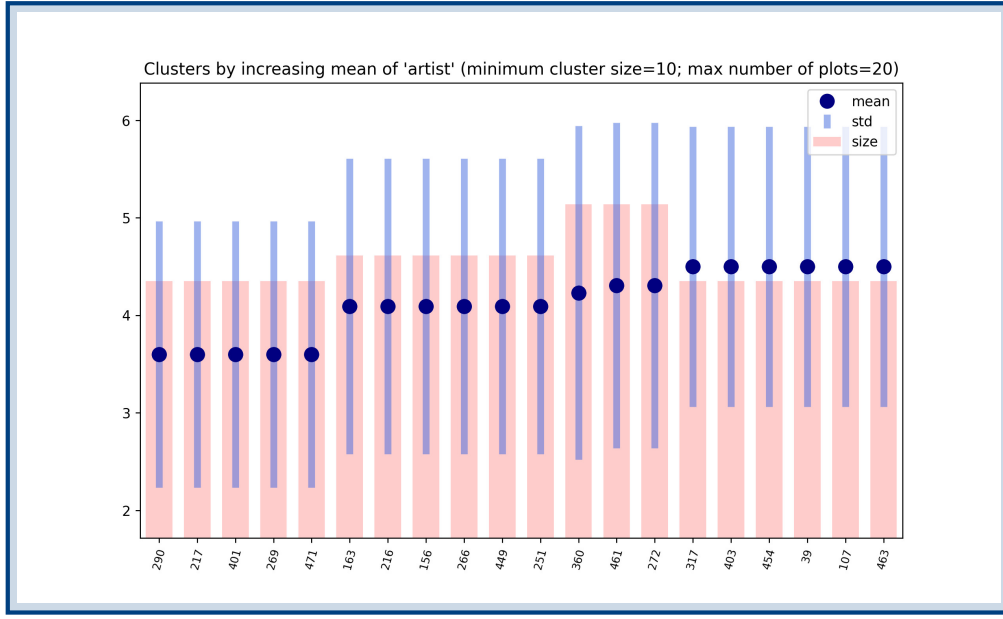


Figure C5. Clusters ordered according to their $\mathcal{V}^{\mathcal{F}}$ score with regards to the feature **artist** (only depicting the 20 lowest scores). The blue dots represent the average score of the cluster, the blue bars represent the standard deviation around this score and the red bars represent the sizes of the clusters. This figure needs to be combined with a representation of the clusters in order to appreciate its results. The cluster numbered 471 (thus one of the five with the lowest score) is represented in Figure C8.

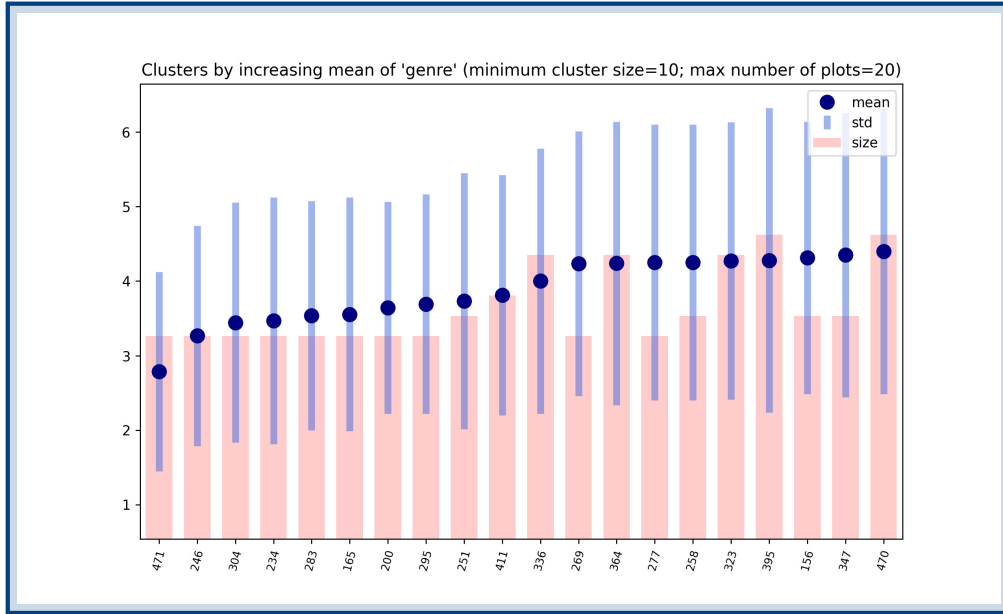


Figure C6. Clusters ordered according to their $\mathcal{V}^{\mathcal{F}}$ score with regards to the feature **genre** (only depicting the 20 lowest scores). The blue dots represent the average score of the cluster, the blue bars represent the standard deviation around this score and the red bars represent the sizes of the clusters. This figure needs to be combined with a representation of the clusters in order to appreciate its results. The cluster numbered 471 (thus the one with the lowest score) is represented in Figure C8.

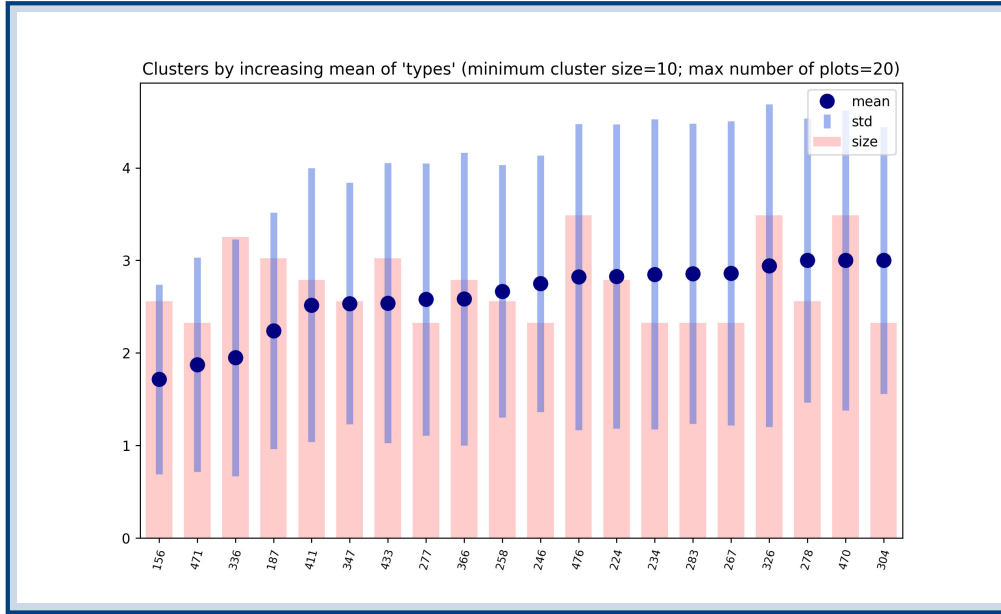


Figure C7. Clusters ordered according to their $\mathcal{V}^{\mathcal{F}}$ score with regards to the feature **type** (only depicting the 20 lowest scores). The blue dots represent the average score of the cluster, the blue bars represent the standard deviation around this score and the red bars represent the sizes of the clusters. This figure needs to be combined with a representation of the clusters in order to appreciate its results. The cluster numbered 471 (thus the one with the second lowest score) is represented in Figure C8.

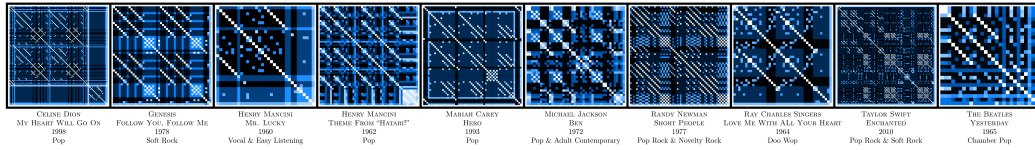


Figure C8. A representation of the cluster numbered 471, having a low $\mathcal{V}^{\mathcal{F}}$ score in Figure C5, C7, and C6. As one can see, all songs have a similar genre (somewhere between ‘pop’ and ‘rock’), however the artist similarity only arises from two of the songs being from HENRY MANCINI. This means that to find finer similarities between songs of the same artist, we need to lower the minimal size of a cluster to less than 10, as it was done in Figure C5; in particular, this is how we managed to find the similarity between songs of LINKIN PARK from Figure 14 (note that there are less than 10 songs in the top two clusters).

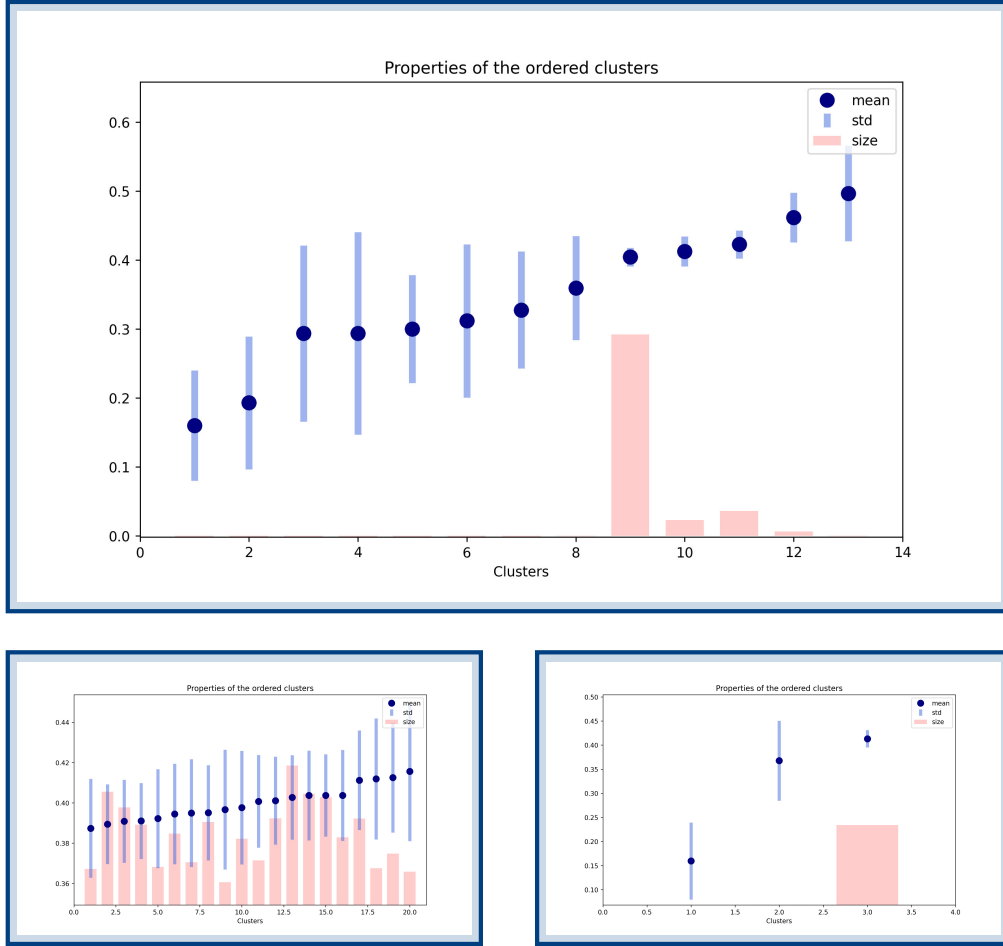


Figure C9. A representation of the statistics of the clusters when applying the three algorithms and outputting 20 clusters. The blue dots represent the average distance between songs of a cluster, the blue bars the standard deviation of these distances, and the red bars the sizes of the clusters. At the top is the output when applying the Agglomerative Clustering, the bottom left figure corresponds to the K-Medians clustering and the bottom right figure corresponds to the Spectral Clustering. As one can see, K-Medians provides a balanced set of clusters, none of them being a singleton, while Spectral Clustering provides only three non-singleton clusters and one with almost all the songs. These figures seem to show that K-Medians is the best algorithm, however, when applied recursively to reduce the sizes of the clusters, it tends to create a lot of singleton clusters and provides less interesting results than Agglomerative Clustering (see Figure C10).

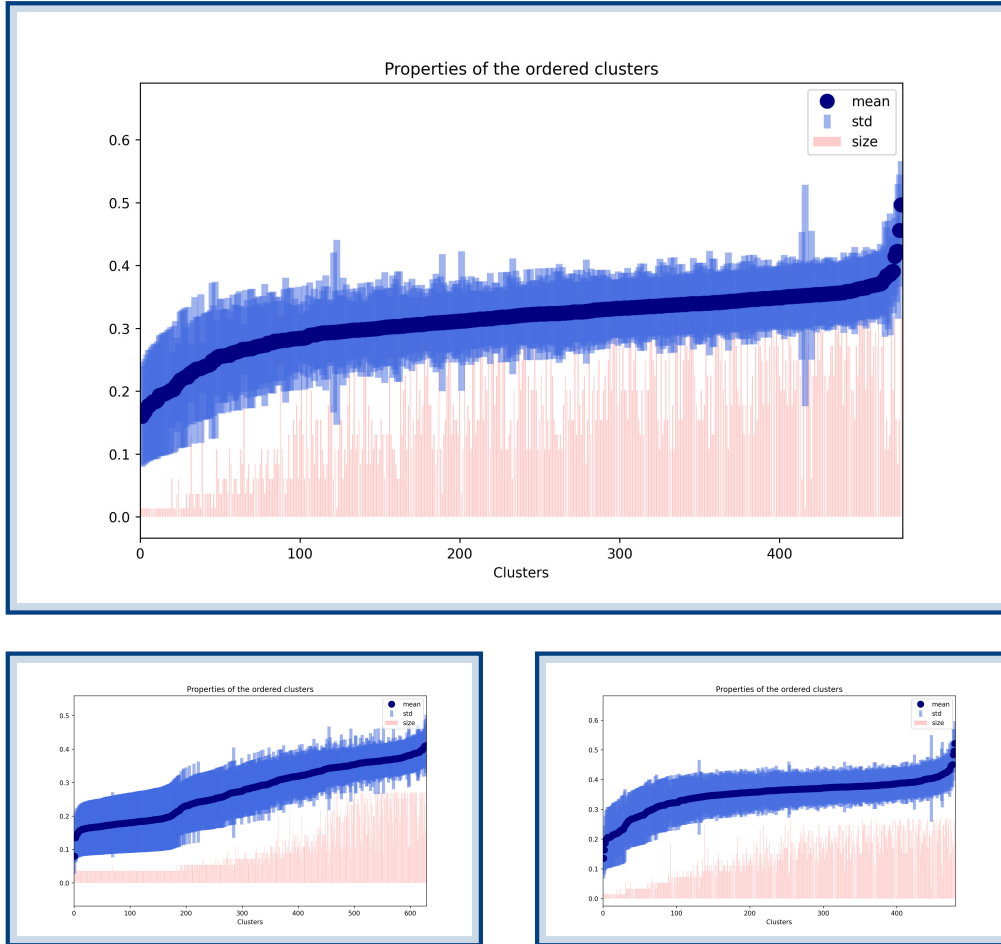


Figure C10. A representation of the statistics of the clusters when applying the three algorithms until all clusters have size at most 15. The blue dots represent the average distance between songs of a cluster, the blue bars the standard deviation of these distances, and the red bars the sizes of the clusters. At the top is the output when applying the Agglomerative Clustering, the bottom left figure corresponds to the K-Medians clustering and the bottom right figure corresponds to the Spectral Clustering. These figures are less readable than those of Figure C9 and it is worth stating another output of the algorithm for each cases, counting the total number of clusters and singletons: 527 clusters including 48 singletons for Agglomerative Clustering, 1047 clusters including 419 singletons for K-Medians, and 516 clusters including 37 singletons for Spectral Clustering. Combining the above figure with Figure C9 shows that Agglomerative Clustering is the most balanced algorithm to consider here and thus explains our choice in Section 4.2.