Reproducing Pitch Experiments in “Measuring the Evolution of Contemporary Western Popular Music”

Colin Raffel, Dan Ellis

1 Introduction

The recent work “Measuring the Evolution of Contemporary Western Popular Music” [1] focuses on measuring large-scale trends in music over time. The authors analyzed acoustic features calculated from about 500,000 pieces of popular music spanning across 54 years. Their approach first subsamples the dataset so that the same number of tracks will be analyzed for each year. They then use various quantization techniques to represent each feature vector as a codeword. Rank-frequency distributions and large weighted networks are constructed out of these generated codewords. Based on the codeword distributions and various statistics of the networks, they draw conclusions about the characteristics of the music from each year.

Unfortunately, a software implementation of the experiments in [1] is not publicly available. Furthermore, the use of network analysis is uncommon in the field of Music Information Retrieval. As a result, the reasoning behind the conclusions made about the evolution of popular music is not clear. Motivated by these issues, we recreated the experiments which involved measuring large-scale changes in pitch, harmony, and melody. Our implementation gives further insight into the techniques used and allows for future comparative research to be carried out.

The rest of this report is structured as follows: First, we describe the sampling and quantization steps used to create the dataset. In the second section, we summarize the statistics and techniques used to draw conclusions about musical evolution. In Section 3, we describe the specifics of our reimplementation of these experiments. Finally, we compare our results and give further insight into the techniques used.

2 Setup

The analysis in [1] makes use of the Million Song Dataset [2], a large-scale collection of metadata and acoustic features calculated for a million pieces of western popular music. The Million Song Dataset includes a year label for about 500,000 tracks. Each of these tracks includes “segment-level” chroma and timbre vectors, which give a compact representation of the pitches present and perceptual quality over short, musically relevant time intervals. Each track is also accompanied by estimated beat times. An overview of these features can be found in [3] and the details of the algorithms used to generate this data can be found in [4].

In order to use the Million Song Dataset, three preprocessing steps are carried out. First, segment-level features are combined into beat-level features so that they may be compared at the same timescale. Second, because the number of tracks for each year varies dramatically, the data is subsampled so that the same number of feature vectors correspond to each year. Finally, because the features available in the Million Song Dataset are continuously valued, they are quantized into the "codeword" form needed for distribution and network analysis.

2.1 Beat-level Integration

The segment-level feature extraction used to generate the Million Song Dataset can be seen as an approximation to critically sampling the feature vectors over time. That is, the segment boundaries are meant to be placed where the feature vectors exhibit dramatic changes so that they represent intervals where the pitch and timbre is relatively constant. The size of segments can vary based on the properties of an audio signal and may not correspond to a musically relevant interval. As a result, the segments are integrated across beats using the approximated beat locations available in the Million Song Dataset. Integration is performed by averaging feature vectors for each segment falling within each beat interval.

2.2 Subsampling

Because the Million Song Dataset is so large and because the songs are not evenly distributed across years, subsampling of the dataset is performed. In particular, one million feature vectors are extracted for each year from 1955 to 2009 from songs labeled with years within a five year window. For example, for the year 2005, one million feature vectors are extracted from songs from 2003-2007 (inclusive). The feature vectors are extracted on a song-by-song basis - that is, songs are chosen at random from the year range and their feature vectors are added to the sample together so that they appear consecutively in the sample. In this way, the million feature vectors for a year range will be a concatenation of feature vectors for a few thousand songs. This subsampling is performed ten times, and all ten subsamplings are used to generate the statistics described in the paper.

2.3 Quantization

Most of the analysis performed is motivated by techniques used to analyze text corpuses and networks. When computing statistics about a collection of text, there is a discrete set of possible items (words) which may appear. Similarly, networks consist of a collection of discrete nodes which are interconnected. Because both of these domains rely on a finite set of codewords or nodes, the feature vectors are discretized such that only a finite number of feature vectors is possible. The discretization process is different for each feature type. Although we only replicate those experiments pertaining to the pitch codewords, we implemented the quantization for each feature type and describe each process below.

2.3.1 Pitch

In the Million Song Dataset, the harmonic characteristics of a song are summarized as a sequence of twelve-dimensional vectors which denote the relative amount of each pitch in the twelve-tone equal temperament scale. Pieces of music typically only use a small subset (corresponding to the pieces' key) of the possible combinations of all semitones. To perform analysis in a key-independent manner, the pitch vectors are circularly shifted to a common tonality using the approach described in [5]. This technique correlates all possible shifts of the chroma vectors in each song with an experimentally derived pitch template [6] and chooses the shift amount corresponding to the highest correlation.

Each pitch vector is initially given in normalized form, such that the values range from 0 to 1. After circular shifting, a simple binary quantization is performed which uses a fixed threshold of .5. This results in $2^{12} = 4096$ possible pitch codewords.

2.3.2 Timbre and Loudness

Timbre describes the pitch- and loudness-independent qualities of different sounds which allow them to be distinguished from another. The timbre of each track in the Million Song Dataset is represented as a sequence of twelve dimensional continuously valued vectors. Each unbounded value in each timbre vector corresponds to high-level abstractions of the spectral surface with the first value corresponding to each segment's perceptual loudness. In [1], this loudness value is first constrained to a range of 0 to 60 and is then quantized to one of 300 discrete levels. The remaining timbre values are discretized using ternary quantization. The three quantiles are set according to a sampling of 1,000,000 timbre vectors across all years, such that at most 8000 timbre vectors are extracted for each year. In this sampling, tracks with no year annotation are included.

3 Experiments

Once collections of codewords have been sampled for each year, various techniques are used to try to encapsulate large-scale changes in musical discourse. Specifically, the frequency of each codeword and the distribution of codeword frequencies are computed to capture the evolution of the probability of encountering different codewords. In order to study sequences of codewords, networks are constructed for each year where each node corresponds to a codeword and an edge between nodes denotes that the codewords appear in adjacent beats. Various statistics are computed on these networks to infer changes in musical structure.

3.1 Codeword Distributions

To determine how overrepresented different codewords are, the rank-frequency distribution is plotted for each decade from 1955 to 2005. These curves represent the relative frequency of the most common, second most common, etc. codewords. The rank-frequency distribution for pitch codewords computed in [1] can be seen in Figure 1. Given the number of times each codeword appears each year, the probability density of codeword counts can be computed. Figure 2 from [1] shows the probability that a codeword appears z times in each of three different years.

3.2 Network Analysis

The rank-frequency analysis described above does not capture the sequence in which codewords appear in the data. To address this, weighted networks were constructed for each year such that the weight of an edge between two nodes denotes the frequency of occurrence of a transition between one codeword and another. For pitch networks, it was found that the number of transitions from one codeword to another was approximately equal in both directions, so undirected networks ware used.

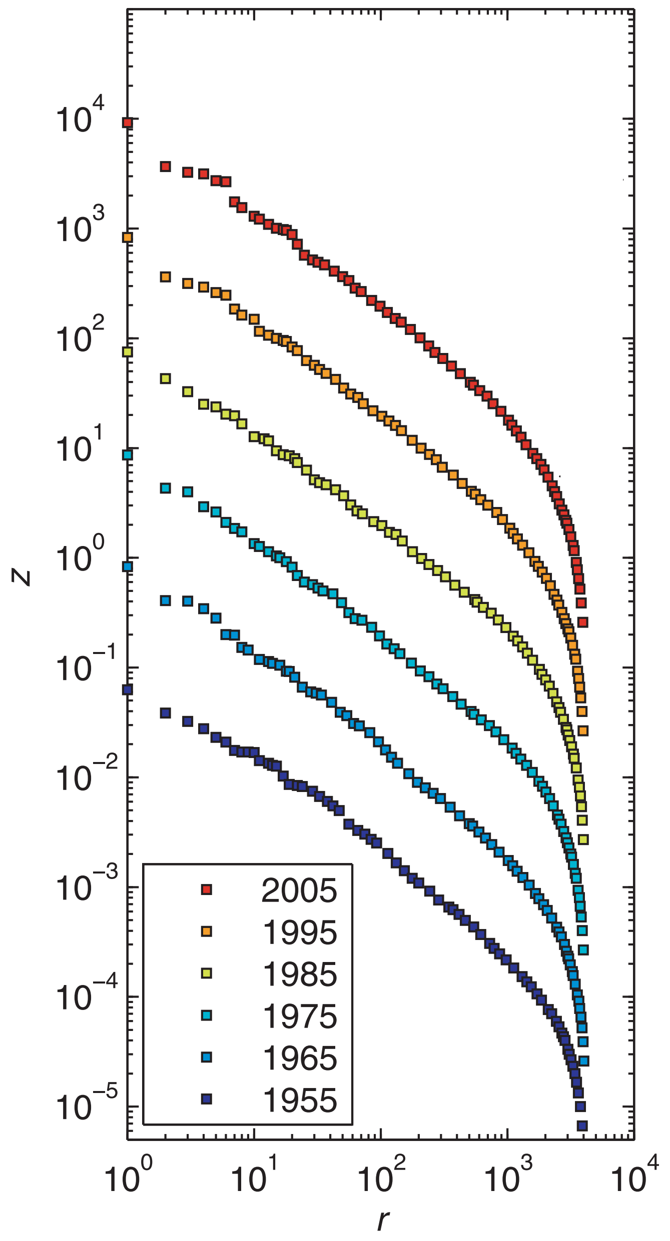

Figure 1: Rank-frequency distribution of pitch codewords from the original experiment. Curves are shifted vertically by a factor of 10 for clarity. r denotes the codeword rank, while z' denotes the relative frequency of the codeword.

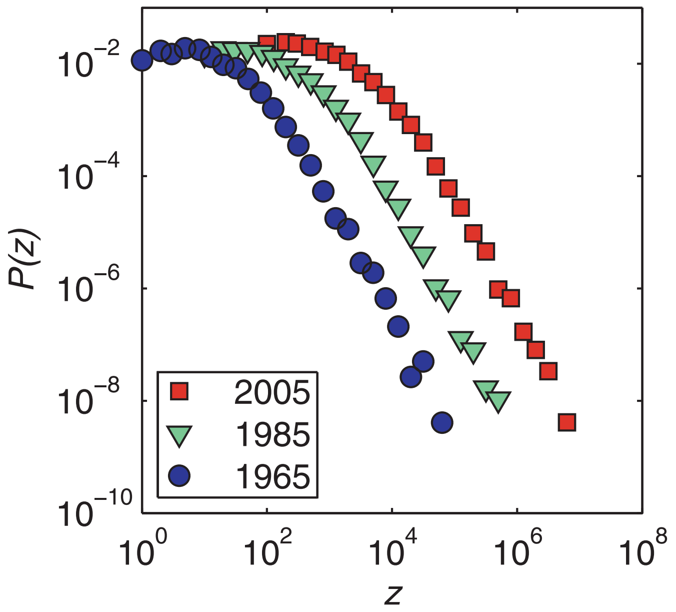

Figure 2: Distribution $P(z)$ of codeword frequencies z from the original experiment. Curves are shifted horizontally by a factor of 10 for clarity.

3.2.1 Preprocessing

It was found that the pitch networks were highly dense, such that most nodes were interconnected. However, most of the edges contained only a small amount of the total weight of the network. As a result, a filtering procedure is used which assumes a null model where the weights of the edges connected to a given node are approximately equal. Any edges which cannot be explained by the null model are removed.

More specifically, given a node with k connected edges, the probability that one edge has a fraction $u \in [0,1]$ of the total weight of all edges connected to the node is assumed to be $(k-1)(1-u)^{k-2}$. An edge with weight w connected to some node with total weight s is then preserved if and only if $(s-w)^{k-1} < \alpha$. The parameter α controls to what extent the network will be filtered and is set to .99 in all experiments. This filtering step dramatically reduces the number of nodes, with the intention of only keeping a relevant backbone of the network.

After filtering, it was found that a few nodes in each network are extremely highly connected. This characteristic may obscure some of the properties of the networks which are to be calculated as part of the experiment. As a result, the ten nodes which are the most connected are removed prior to calculating the statistics described below. This step intends to reveal the finer-grain structure in each network by ignoring the characteristics of the networks which do not change year to year.

The network statistics were also computed for a randomized network constructed by randomly swapping pairs of links as described in [7]. This scheme takes two edges connecting nodes A, B and C, D respectively and randomly rewires them to instead connect A, C and B, D or A, D and B, C. Provisions are added to avoid self-links and parallel edges. Link swapping is performed $4M$ times, where M is the number of edges in the network. This randomization process maintains the number of edges connected to each node but changes the network topology.

3.2.2 Statistics

Once weighted undirected networks have been constructed and preprocessed as described above for each year, various common statistics are computed to quantify their structure. The clustering coefficient C is used to measure the extent to which nodes are locally interconnected and is computed as

$$C = \frac{1}{N} \sum_{i}^{N} \frac{2T_i}{k_i(k_i - 1)}$$

where T_i is the number of triangles and k_i is the number of nodes connected to node i, and N is the total number of nodes in the network. Random networks tend to have small clustering coefficients, while real-world networks often exhibit some level of clustering. The average shortest path length is also used to measure how densely interconnected a network is, and is computed as the average of the lengths of the paths consisting of smallest number of edges between each pair of nodes. In [1], the trend of the clustering coefficient and average shortest path length is computed for the preprocessed and randomized networks, producing the scatterplot shown in Figure 3.

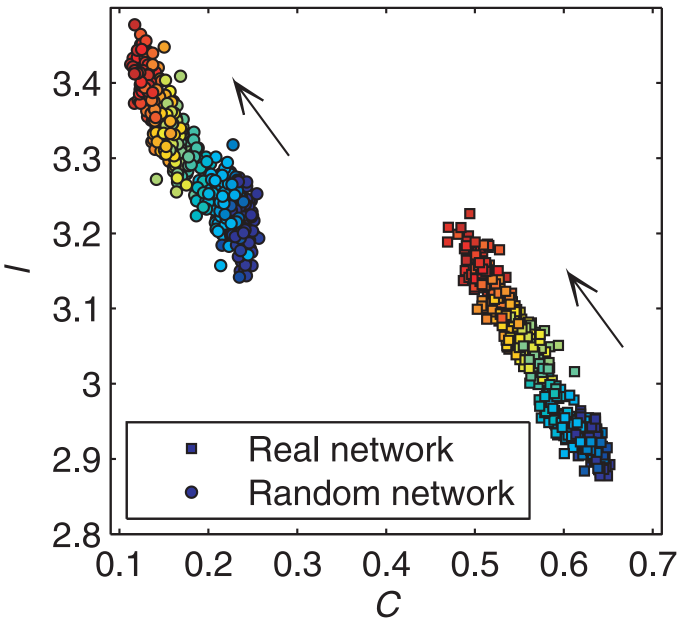

Figure 3: Average shortest path length l and clustering coefficient C for the pitch networks from the original experiment. The networks were constructed and randomized as described in Section 3.2. The point color and arrows indicate chronology, with dark blue corresponding to early years and red to recent ones.

4 Implementation

To reproduce the results in [1], we reimplemented each of the steps described above in a cohesive package of Python scripts available online.[1] We made extensive use of the Numpy and Scipy [8] libraries for various numerical calculations. We also used Matplotlib library [9] to produce the various figures shown in Section 5. Our Python scripts are directly importable as IPython Notebook files [10] which allows for intuitive, step-by-step reproduction of our results. We also included extensive references to relevant passages in [1] to make it clear which part of the experiment each function corresponds to.

Beat-based features were extracted using code distributed with the Million Song Dataset. We then randomly chose tracks from each year window until 1,000,000 beat-level feature vectors were extracted. This random sampling was performed ten times. Notably, only 626,803, 777,324, and 949,293 beats are available for the year windows around 1955, 1956, and 1957 respectively. As a result, we used a smaller number of codewords for those years and the same set of vectors were necessarily used for each of the ten random samplings. When shifting the pitch features to a common key, we used values for the pitch template from [11].

To compute the rank-frequency distribution, we simply sorted the number of times each codeword appeared in each year. The probability of each codeword count was then computed in the maximum-likelihood sense - that is, we computed the number of times each codeword frequency appeared in each year and normalized by the total number of unique codeword counts.

Networks for the pitch codewords were constructed by first mapping each twelve-dimensional binarized vector to a unique integer value between 0 and 4095. We utilized the igraph software package [12], which is implemented in C++ with Python bindings. Igraph includes functions for creating the networks and producing the network statistics described above. For preprocessing the pitch networks, we produced our own implementation of the “disparity filter”. We compared a publicly-available implementation of the random edge-swapping technique from the author of [7] to the one used in [1] which was obtained through private correspondence with the authors. Both implementations were translated to work with Python and igraph.

All of our experiments were run on a 2011 Macbook Pro running Mac OS X 10.7.4 with an eight-core 2.2 GHz Intel Core i7 processor and 8 GB of RAM. Storing and processing the Million Song Dataset takes hundreds of gigabytes of digital storage space, so we used an external hard drive. The necessary processing time to subsample the data and generate the figures shown in Section 5 was about eight hours.

5 Results

Our reproduction of the rank-frequency curves from the original experiment is shown in Figure 4. This plot is qualitatively similar to Figure 1 and exhibits the same consistency of rank-frequency distribution across decades. The curves also indicate that the relatively frequency of codewords falls off roughly linearly with rank. The close similarity of these figures indicates that our implementation of the subsampling and codeword generation steps is likely correct.

We note that each curve in Figure 4 has approximately 4096 points, corresponding to the total number of possible codewords, while those in Figure 1 have substantially fewer. This could imply that the rank-frequency counts from the original experiment do not include every codeword or that

[1]`https://github.com/craffel/music-evolution`

some kind of histogram binning was used. Furthermore, we note that our curves are shifted down by a factor of roughly three relative to those from the original experiment. Because these plots show the relative frequency, the sum of the y-value of all points on each curve should be 1. This difference could be an artifact of the per-decade factor of 10 shifting or the histogram binning technique, if one was used.

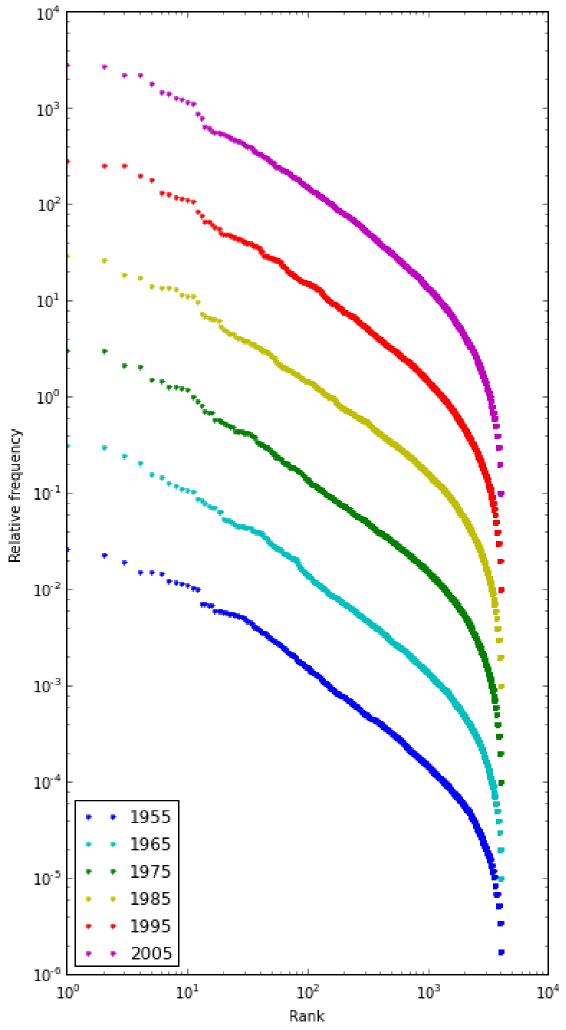

Figure 4: Reproduction of rank-frequency distributions for pitch codewords. Curves are shifted vertically by a factor of 10 per decade.

When reproducing the codeword frequency distributions shown in Figure 2, we first plotted the individual codeword counts against and the relative frequency of each count. This produced the the scatterplot in Figure 5. These curves have the intuitive explanation that many codeword counts (particularly large ones, such as 978, 1032, etc.) only appear once, while small codeword counts

(1, 2, 3, etc) correspond to many codewords. As a result, the curves fall off quickly from high probability values and flatten out into discrete lines for large codeword counts.

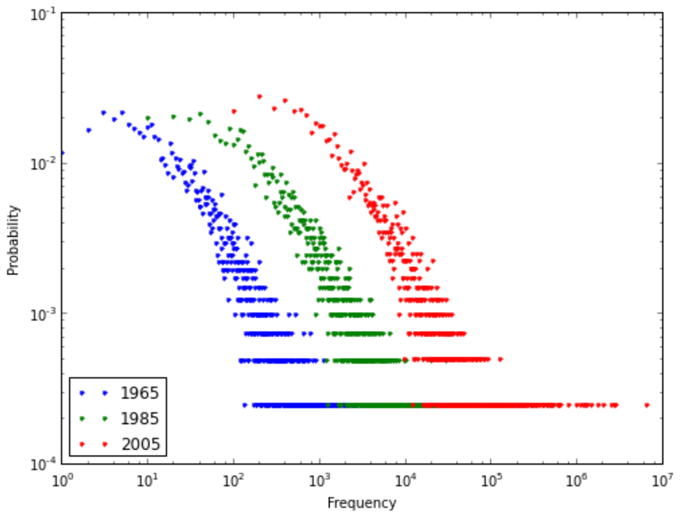

Figure 5: Scatterplot of codeword frequencies against the probability of each frequency. Curves are shifted horizontally by a factor of 10 for clarity.

While intuitive, this plot clearly does not match the original experiment. In particular, due to the small number of points plotted per curve, it is clear that some kind of histogram binning is being performed. We found that the original plots show 25 discrete points, so we computed a histogram with 25 log-spaced bins to group the data points shown in Figure 5. This produced the curves shown in Figure 6.

While qualitatively much closer to the original diagram, we note that these curves exhibit a dip in the low counts not present in those from the original experiment. We reasoned that this could be due to a normalization by histogram bin width. After normalizing the bin counts by the bin width, we obtained the curves shown in Figure 7. This plot is qualitatively very close to Figure 2, up to a normalization in the counts. We therefore conclude that the additional step of computing a width-normalized, log-binned histogram was undertaken in the original experiment.

Our first attempt at reproducing the plot showing the evolution of network statistics for randomized and unrandomized networks is shown in Figure 8. For this plot, we used publicly available code from the original author of the algorithm [7]. The statistics corresponding to the unrandomized networks match up very closely to those in Figure 3. This indicates that our network construction and preprocessing steps match those from the original experiment.

Importantly, the randomized networks produced using the code from [7] do not exhibit a characteristic increase in average shortest path length as seen in Figure 3. Through correspondence with the authors of [1], we obtained the code used to produce the randomized networks in the original experiment. Using this implementation produced Figure 9. The statistics of the randomized networks in this plot very closely match those in in Figure 3.

After comparing the two techniques used for network randomization, we found that they differed in two aspects. First, the approach used in [1] chooses random edges by selecting nodes at random

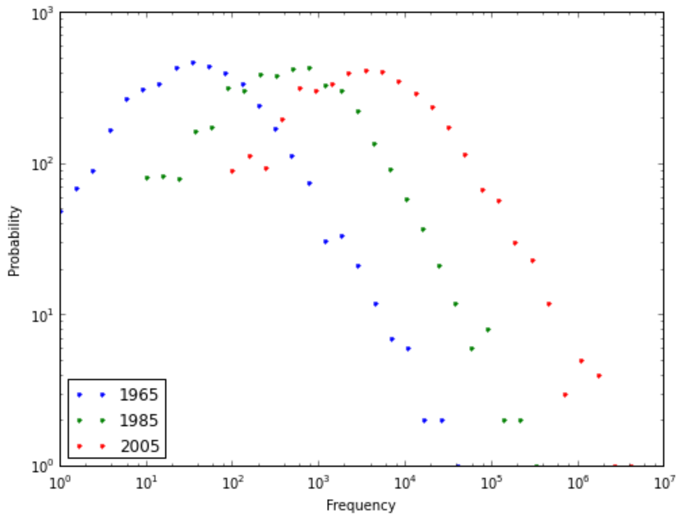

Figure 6: Scatterplot of log-binned histogram counts of codeword frequencies. Curves are shifted horizontally by a factor of 10 for clarity.

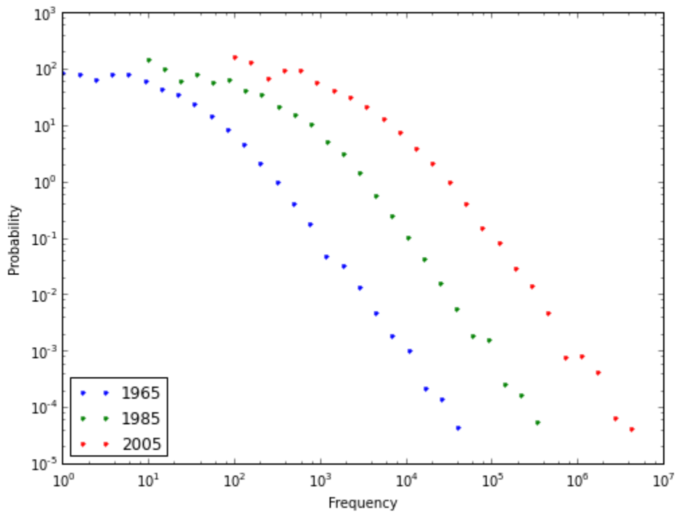

Figure 7: Scatterplot of log-binned histogram counts of codeword frequencies, normalized by bin width. Curves are shifted horizontally by a factor of 10 for clarity.

and choosing a random edge from the chosen node. This preferentially chooses edges which are connected to nodes with many edges, and as a result does not uniformly randomly choose edges.

The second difference involves the order in which the edges are rewired - that is, whether to rewire the edges A, B and C, D to A, C and B, D or A, D and B, C. In the approach used in [1], the edges are always rewired as A, C and B, D while in [7] the rewiring order is chosen uniformly at

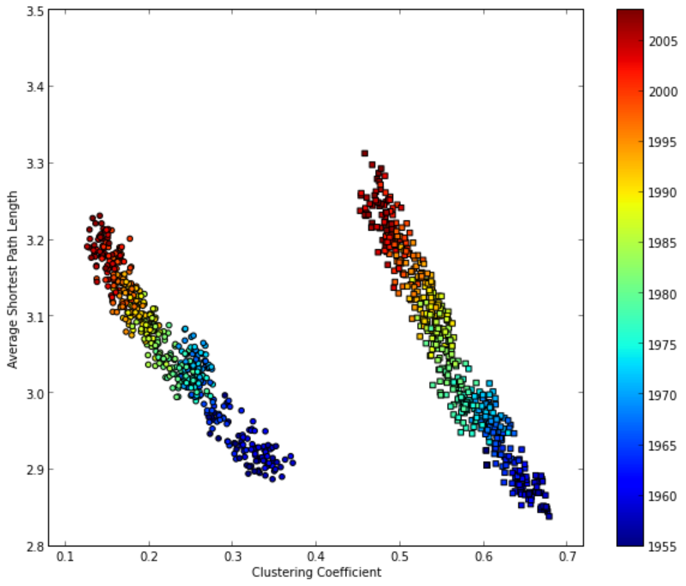

Figure 8: Network statistics for randomized and unrandomized networks using random edge-swapping code from [7].

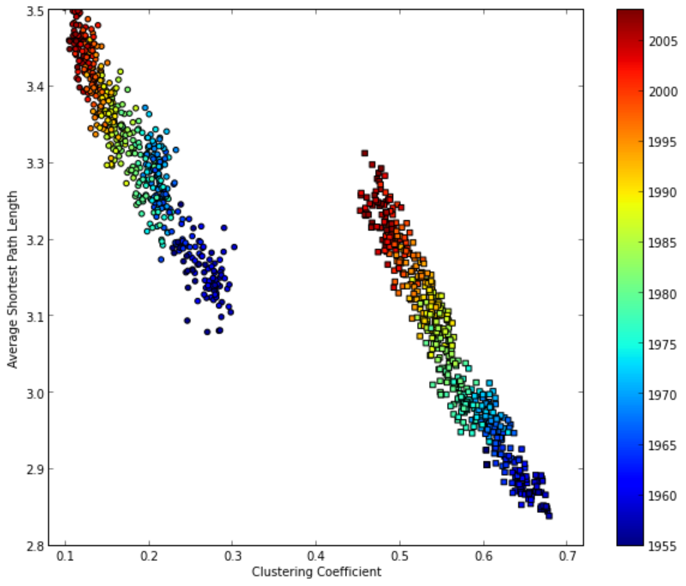

Figure 9: Network statistics for randomized and unrandomized networks using random edge-swapping code from [1].

random. To determine the effect of this latter difference, we added functionality to the code from [1] to randomly choose the rewiring order. This produced the plot shown in Figure 10. From this figure, it is clear that the randomization of the edge rewiring order has a small but noticeable effect on the average shortest path length of the randomized networks.

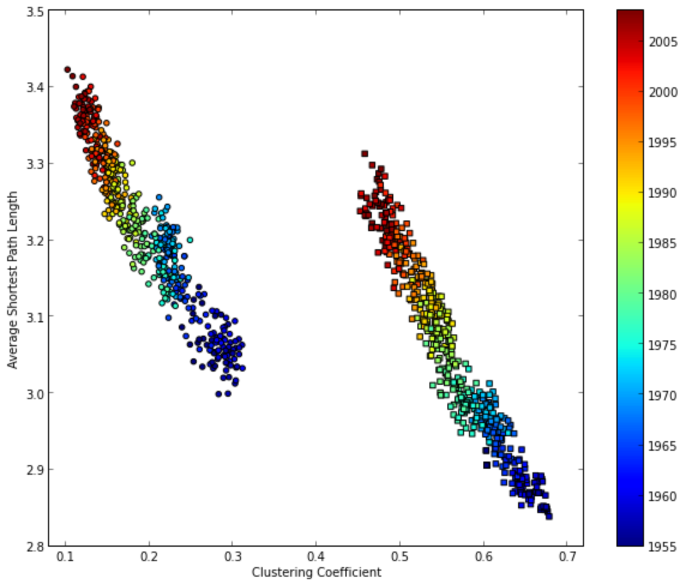

Figure 10: Network statistics for randomized and unrandomized networks using random edge-swapping code from [1], modified to randomly choose the edge rewiring order.

6 Conclusion

In this report, we have described the reimplementation of the pitch-related experiments carried out in [1]. In the course of reproducing these experiments, we found various details which were omitted from the original work but are helpful for obtaining the presented results. We have made our implementation publicly available so that other researchers can modify, expand, and compare work focused on measuring musical evolution.

References

[1] Joan Serrà, Álvaro Corral, Marián Boguñá, Martín Haro, and Josep Ll Arcos, "Measuring the evolution of contemporary western popular music," *Scientific Reports*, vol. 2, 2012.

[2] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere, "The million song dataset," in *ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida*. University of Miami, 2011, pp. 591–596.

[3] Tristan Jehan and David DesRoches, "Analyzer documentation," Tech. Rep., The Echonest, 2011.

[4] Tristan Jehan, *Creating music by listening*, Ph.D. thesis, Massachusetts Institute of Technology, 2005.

[5] Joan Serra, Emilia Gómez, and Perfecto Herrera, "Transposing chroma representations to a common key," in *IEEE CS Conference on The Use of Symbols to Represent Music and Multimedia Objects*. Citeseer, 2008, pp. 45–48.

[6] Carol L Krumhansl, *Cognitive foundations of musical pitch*, Number 17. Oxford University Press, USA, 1990.

[7] Sergei Maslov and Kim Sneppen, "Specificity and stability in topology of protein networks," *Science Signaling*, vol. 296, no. 5569, pp. 910, 2002.

[8] Eric Jones, Travis Oliphant, Pearu Peterson, et al., "SciPy: Open source scientific tools for Python," 2001–.

[9] John D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, May-Jun 2007.

[10] Fernando Pérez and Brian E. Granger, "IPython: a System for Interactive Scientific Computing," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 21–29, May 2007.

[11] Tuomas Eerola and Petri Toiviainen, *MIDI Toolbox: MATLAB Tools for Music Research*, University of Jyväskylä, Jyväskylä, Finland, 2004.

[12] Gabor Csardi and Tamas Nepusz, "The igraph software package for complex network research," *InterJournal, Complex Systems*, vol. 1695, pp. 38, 2006.