Markov Chains in Generative Music

Supervisor: Sean Bechhofer

Christopher Norman
Final Year Project Report
BSc Computer Science and Mathematics

The University of Manchester
Department of Computer Science

April 2022

Abstract

Markov Chains in Generative Music

Author: Christopher Norman

Generative music describes constantly changing music created by a system using a set of rules. In this project we describe and implement Markov chains as a generative method for simple single line melodies and rhythm. We discuss the limitations, effectiveness and future applications of the model.

Supervisor: Sean Bechhofer

Acknowledgements

I would like to thank Sean Bechhofer and my tutor group members Rowan and Tom for their encouragement, discussions, feedback and excellent song recommendations throughout this project.

Contents

1 Introduction **6**
1.1 What is Generative Music? 6
1.2 Motivation 8
1.3 Musical Definitions 8
1.4 Aim 8

2 Markov Chains **9**
2.1 First Order Markov Chains 9
2.2 Higher-Order Markov Chains 11
2.3 Random Walk 12
2.4 Application of Markov Chains in a Musical Context 12
2.4.1 First-Order Chains 12
2.4.2 Higher-Order Chains 13

3 Ethical Considerations **14**
3.1 Markov Chains 14
3.2 Input Data 15
3.3 Data Collection 15

4 Design and Implementation **16**
4.1 Software and Tools 16
4.2 Architecture 17
4.3 Simplification of Music Input 17
4.4 Implementation 18
4.4.1 Markov Chain Object 18
4.4.2 Input Data Conversion 18
4.4.3 Markov Model Creation 19
4.4.4 Other Generation Methods 21
4.4.5 Sequence Generation 21
4.4.6 User Parameters and Model Selection 21
4.4.7 Output 22

5 Results Analysis **24**
5.1 Analysis Procedure 24
5.1.1 Observations During Testing 24
5.1.2 Survey 25
5.1.3 Limitations of survey analysis 26
5.1.4 Multiple Linear Regression 27
5.1.5 Multiple R^2 correlation coefficient 28
5.1.6 Nested Models 28
5.1.7 Analysis of Variance 29
5.1.8 Hypothesis 29
5.2 Survey Results 30
5.2.1 Data Processing 30
5.2.2 Survey Section 1 - Combined Melody and Rhythm Generation . . . 32
5.2.3 Survey Section 2 - Rhythm Generation 34
5.3 Survey Conclusions 37

6 Project Evaluation **39**
6.1 Limitations 39
6.2 Future work 39
6.2.1 Application to ambient music 39
6.2.2 Musical Dynamics/ Polyphony 40
6.2.3 Non-homogeneous Markov chains 40
6.2.4 Graphical Interface/ Web page 40
6.2.5 Hidden Markov Models 40
6.3 Project Planning and Reflection 41
6.4 Achievements and Conclusion 42

Bibliography **43**

A Glossary **46**

List of Figures

2.1 Example Markov chain diagram 9
2.2 Example Markov transition matrix 10
2.3 Sheet music for ‘Mary Had A Little Lamb’ 12
2.4 Transition matrix and diagram for ’Mary Had A Little Lamb’ 12
2.5 Sheet music for ‘Mary Had A Little Lamb’ where the note `G` is highlighted in red 13

4.1 Markov chain music generation diagram 17
4.2 Sequence of notes in the song ‘Mary Had A Little Lamb’ 19
4.3 Python output: transition frequency matrix for the song ‘Mary Had A Little Lamb’ 20
4.4 Python output: transition probability matrix for the song ‘Mary Had A Little Lamb’ 20
4.5 Original piece: “Mary Had A Little Lamb” 22
4.6 Generated piece for “Mary Had A Little Lamb” 22
4.7 Generated melody from “Nocturne in E-flat major, Op. 9, No. 2” by F. Chopin 23

5.1 Extract from CSV data generated by Google forms output 30
5.2 Section from updated survey sample data CSV file 30
5.3 Distribution of musical abilities among respondents in the survey 31
5.4 Bar charts for survey section 1 rating distributions 31
5.5 Output from R `summary()` function for multiple linear regression model with continuous ‘generation method’ variable 32
5.6 Output from R `summary()` function for multiple linear regression model with discrete ‘generation method’ variable and uncombined A and D samples. . . 33
5.7 Output from R `summary()` function for multiple linear regression model with discrete ‘generation method’ variable without ‘ability_level’ variable and uncombined A and D samples. 35
5.8 ANOVA table comparing the nested model created by removing the `ability_level` variable. 35
5.9 Output from R `summary()` function for multiple linear regression model with discrete ‘generation method’. The music samples using the same generation method are combined in this model. 36

5.10 Output from R `summary()` function of multiple linear regression model for comparing rhythm generation methods. 38

6.1 Jira Development Timeline 41
6.2 Jira kanban board 41

Chapter 1

Introduction

In this chapter, we introduce the definition of generative music and describe the motivation and aims of the project.

1.1 What is Generative Music?

Generative music describes constantly changing music produced by a system restricted by a set of rules or processes. Brian Eno, a musician and producer famous for his pioneering work in ambient music and has collaborated with David Bowie, U2, Talking Heads and Paul Simon, popularised the term 'generative music' in 1996 with the release of *Generative Music 1* [1]. Instead of being released as a traditional recording Generative Music 1 was released on floppy disk and used SSEYO's Koan generative music system. The software used probabilistic rules to determine which notes will be played which means each performance is likely to be unique.

Whilst the term 'Generative music' is relatively recent, examples of generated or algorithmic music are not new. Musikalisches Würfelspiel (German for 'musical dice game') was a method of randomly generating music using dice and is possibly attributed to Wolfgang Amadeus Mozart in the 18th century [2]. The general idea is to combine small sections of music (e.g. notes or single bars) at random to produce a piece that is made up from these musical fragments.

Another example of algorithmic composition that also includes an element of chance is *Reunion* by John Cage in 1968. Cage used a chessboard containing photo-receptors on each of the squares, when a player moved a piece a different note would be played. This meant a unique piece of music would be produced each time a game is played [2].

The first piece of music composed by a computer is generally considered to be the *Illiac Suite* [3] in 1957. Before this computers had only aided a human composer, Lejaren Hiller and Leonard Isaacson used the Illiac computer to simulate a music composition process. They first generated materials using various methods then applied transformation rules to manipulate the materials and finally, used rules to select some of the generated material to combine into a final composition [2]. One of the techniques used to generate these initial materials was Markov chains (A random process such that the state of a certain system at a certain time only depends on the previous state of the system [4]). This is also the first documented use of Markov chains to generate music [3]. The uses of Markov chains for music composition are discussed in depth in the 1989 article: *The Markov Process as a Compositional Model: A Survey and Tutorial* by Charles Ames [5]. Ames states that he once thought Markov chains were "only of peripheral interest to users of composing programs" but then reconsidered how well Markov chain models work when he was hired to implement Markov processes that evolve gradually over a piece of work. A more recent paper, *Algorithmic music composition using dynamic Markov chains and genetic algorithms* by Chip Bell [6] in 2011, describes a method for combining Markov chains and genetic algorithms ("heuristic optimization algorithms, which are inspired by natural selection and evolution in biological systems" [7]). Markov chains are used for choosing the next pitch, rhythm, or chord and genetic algorithms are used to select the Markov chains that produce the "most pleasant-sounding music".

Music composition algorithms that are based on mathematical equations and random events generally use stochastic/random processes. Stochastic models contain random variables and are therefore non-deterministic, this means that it is not possible to predict the output (a piece of music) for a given input, even if the input is the same. Markov chains are examples of stochastic processes and are discussed in the next chapter.

More recent breakthroughs in generative music have used machine learning models; In 2016 DeepMind, an artificial intelligence and research laboratory created WaveNet which uses deep neural networks for generating raw audio waveforms. The initial task was aimed at text-to-speech and achieved state-of-the-art performance. As a further experiment, WaveNet was then trained to model music and the researchers found that "it generates novel and often highly realistic musical fragments" [8]. The two data sets used consisted of 200 hours of music audio and 60 hours of solo piano audio. Due to higher accessibility to large data sets and more computation power, deep learning methods have grown in popularity over the last 30 years [9].

The *Handbook of Artificial Intelligence for Music* (2021) contains various papers from leading experts in the field. The chapters range from sociological and philosophical issues [10] to state-of-the-art implementations of models for generating music. In particular, Artemi-Maria Gioti discusses *Artificial Intelligence for Music Composition* [11] and states "As "one-fits-all" approaches are rarely possible, the design of AI tools for music should be regarded as artistic in nature.". This idea extends to all generative music methods and hence these tools are not a replacement for the artist or composer but are used to aid a creative process.

1.2 Motivation

Brian Eno's book *A Year With Swollen Appendices* [12] encapsulates the potential of generative music in the quote:

> "I too think it's possible that our grandchildren will look at us in wonder and say, 'You mean you used to listen to exactly the same thing over and over again?'".

Today's generative music methods are unlikely to produce a number one hit single by themselves but provide avenues for exploration and allow people with very little musical experience to become 'composers' themselves. For example, instead of a traditional music model, where an artist or composer creates a piece of music and the listener plays a recording of the created music, generative music methods can allow the listener to have control over the music by modifying various parameters set by the composer. The music becomes an interactive art piece and provides artists with a new dimension in how their music can connect with an audience. Generative music models are also used as tools for artists and producers to use to enhance and build new musical compositions. Rather than viewing generative music tools as competition they should be viewed as tools for collaboration with computers, the paper *"Modes for Creative Human-Computer Collaboration: Alternating and Task-Divided Co-Creativity"* describes a number of possible ways for music co-creation with a computer [13].

Markov chains are a key concept within probability theory and have a variety of applications as statistical models in industries such as finance and operational research [14]. They also form the basis for more complex models such as hidden Markov models [15] and for statistical computing in Markov chain Monte Carlo methods [16]. Markov chains have been applied in music since 1957 [17] and are still used as the basis for more recent applications such as hidden Markov models for music composition [6].

1.3 Musical Definitions

Useful musical definitions are found in Appendix A.

1.4 Aim

The aim of the project is to implement and evaluate different methods of generating simple musical melodies based on an input using Markov chains. A goal is to find an "optimal" balance between complete randomness and a previous composer/artist's work. We will state our hypotheses about Markov chain music generation and a questionnaire will be carried out to evaluate these methods.

Chapter 2

Markov Chains

In this chapter, we introduce Markov chains and related mathematical definitions. We then show how Markov chains can be applied in a musical context for music generation.

2.1 First Order Markov Chains

A Markov chain is a stochastic (random) process such that the state of a certain system at a certain time only depends on the previous state of the system [4]. Figure 2.1 is an example of a Markov chain as a directed graph.

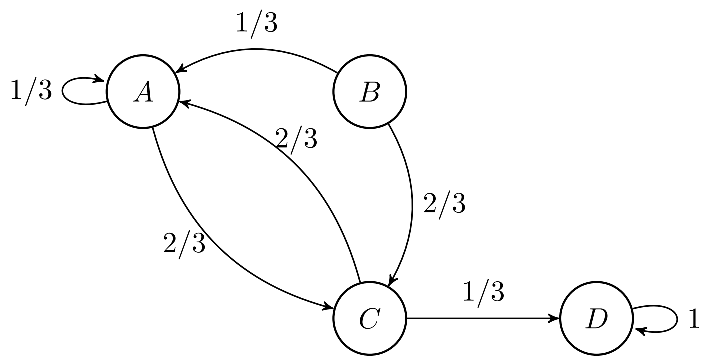


Figure 2.1: Example Markov chain diagram

In this example, we have 4 states represented as nodes and edges representing the probabilities of transitioning between these states. Note that these edges are directed. This graph can be written in matrix form as shown in Figure 2.2. Where position (i, j) in the matrix represents the probability of the transition from node i to node j. This is called a transition matrix. The rows indicate the current state and the columns indicate the next state. The values in each of the rows must sum to 1.

$$\begin{array}{c|cccc} & A & B & C & D \\ A & \frac{1}{3} & 0 & \frac{2}{3} & 0 \\ B & \frac{1}{3} & 0 & \frac{2}{3} & 0 \\ C & \frac{2}{3} & 0 & 0 & \frac{1}{3} \\ D & 0 & 0 & 0 & 1 \end{array}$$

Figure 2.2: Example Markov transition matrix

We define Markov chains as in [18] and [14]:

Definition 1 (Markov chain definition) *A Markov chain is a sequence of random variables $\boldsymbol{X} = \{X_k : k = 0, 1, 2, ...\}$ with the following properties. X_k is defined on the probability space $(\Omega, \mathcal{F}, P)$, where Ω is the sample space (a set of all outcomes), $\mathcal{F}$ is the events space which is a set of events (an event is a set of outcomes in the sample space) and P is a probability function. Each X_k take values in a finite set S where S is the state space (the set of all possible values of any X_k). For any $\{i, j, i_{k-1}, i_{k-2}, ..., i_0\} \subseteq S$ and any $k \geq 1$, we have the Markov property:*

$$P\{X_{k+1} = j | X_k = i, X_{k-1} = i_{k-1}, X_{k-2} = i_{k-2}, ..., X_0 = i_0\} = P\{X_{k+1} = j | X_k = i\} \tag{2.1}$$

and the transition probabilities

$$P\{X_{k+1} = j | X_k = i\} = p_{i,j} \tag{2.2}$$

are independent of k. We can abbreviate Equation 2.1 as,

$$P\{X_{k+1} | X_k, X_{k-1}, X_{k-2}, ..., X_0\} = P\{X_{k+1} | X_k\}$$

Each X_k in $\boldsymbol{X}$ takes the value of a single state in the set S.

This describes a system that changes from one state to another over time. The states of the system are contained in the set S. The value of X_k is the state of the system at time k and X_0 is the initial state of the system. If the system is in the state i at time k, then the probability that it changes to state j at time $k + 1$ does not depend on the earlier times $k - 1, k - 2, ...0$ [18]. Hence the next state of the system only depends on the current state of the system. The above definition is for *stationary* Markov chains. This means the probabilities of transitioning from one state to another ($p_{i,j}$) do not change with time k. There exist Markov chains with probabilities that depend on time but in this project we will only consider Markov chains with the stationary condition.

Some other useful definitions from [19] are stated below:

Definition 2 (Absorbing state) *A state $a \in S$ is said to be absorbing if $p_{a,a} = 1$.*

This means from state a the only possible transition is to itself.

Definition 3 (Accessible state) *A state $b \in S$ is said to be accessible from $a \in S$ if there exists a finite integer $n \geq 0$ such that*

$$P\{X_n = b | X_0 = a\} > 0 \tag{2.3}$$

In other words, it is possible to travel from a to b with a non-zero probability in a certain number of steps.

Definition 4 (Recurrent state) *A state $i \in S$ is said to be recurrent if, starting from state i, the chain will return to state i within a finite time, with probability 1.*

Definition 5 (Diagonal matrix) *A square matrix of size $n \times n$, where n is a positive integer, is **diagonal** if all non-zero elements are only in the diagonal running from the upper left to the lower right.*

$$\begin{pmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{pmatrix}$$

2.2 Higher-Order Markov Chains

Second order Markov chains consider not only the current state of the system but also the previous state. The transition probabilities from Definition 2.1 can therefore be rewritten as

$$P\{X_{k+1} | X_k = i, X_{k-1} = i_{k-1}, ..., X_0 = i_0\} = P\{X_{k+1} = j | X_k = i, X_{k-1} = i_{k-1}\} \tag{2.4}$$

We can generalise this to an n^{th} order Markov chain where we consider n previous states of the system.

$$P\{X_k | X_{k-1}, X_{k-2}, ..., X_0\} = P\{X_k | X_{k-1}, X_{k-2}, ..., X_{k-n}\} \tag{2.5}$$

A conventional model with m possible states and an order of k has a total of $(m-1)m^k$ model parameters [4]. This means the number of parameters increases exponentially with respect to the order.

2.3 Random Walk

A random walk on a graph can be defined as follows,

Let G be a finite graph with at least two vertices and is connected (each vertex is 'reachable' from any other vertex). Starting at a vertex u from all the edges connected to u choose one uniformly at random. Move to the vertex v connected to u by the chosen edge and continue this process from v [20].

We can also perform a random walk on weighted graphs, known as a *biased random walk*. Instead of selecting the next vertex v uniformly at random, we can choose an edge at random with a probability proportional to its weight. The idea of a random walk is useful for generating a sequence of nodes visited in a Markov chain where the weights of the edges are equal to the transition probabilities.

2.4 Application of Markov Chains in a Musical Context

2.4.1 First-Order Chains

In a first-order Markov chain we can simply assign a single node to each note. For example

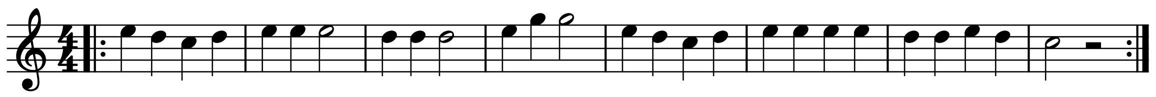


Figure 2.3: Sheet music for 'Mary Had A Little Lamb'

the piece 'Mary Had A Little Lamb' shown in Figure 2.3 contains the notes `E, D, C` and `G`. The transition matrix and diagram are shown in Figure 2.4

$$\begin{array}{c|cccc} & E & D & C & G \\ \hline E & \frac{1}{2} & \frac{5}{12} & 0 & \frac{1}{12} \\ D & \frac{4}{10} & \frac{3}{10} & \frac{3}{10} & 0 \\ C & 0 & 1 & 0 & 0 \\ G & \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{array}$$

$$\begin{tikzcd}
E \arrow[loop left, "\frac{1}{2}"] \arrow[r, bend right, "\frac{5}{12}"'] \arrow[d, bend right, "\frac{1}{12}"'] & D \arrow[l, bend right, "\frac{4}{10}"'] \arrow[loop right, "\frac{3}{10}"] \arrow[d, bend left, "\frac{3}{10}"] \\
G \arrow[loop left, "\frac{1}{2}"] \arrow[u, bend right, "\frac{1}{2}"'] & C \arrow[u, bend left, "1"]
\end{tikzcd}$$

Figure 2.4: Transition matrix and diagram for 'Mary Had A Little Lamb'

To create the transition matrix we first calculate the frequencies of each note following the single previous note and place these in a matrix (ignoring the starting note). We then divide each row by its row sum to create probabilities that sum to 1. In Figure 2.5 we have highlighted the `G` notes.

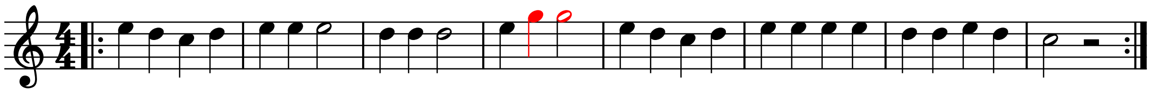


Figure 2.5: Sheet music for 'Mary Had A Little Lamb' where the note `G` is highlighted in red

We see that only `E` and itself follow it, therefore the probabilities of a transition from `G` to `[E,D,C,G]` are $[\frac{1}{2}, 0, 0, \frac{1}{2}]$. This process is continued for every note in the piece/-pieces of music. We can also adapt this model to include the rhythm/duration of notes very simply, instead of each note being a node we can take note-rhythm pairs such as ($A\#$, `quarter`) or ($E\flat$, `16th`) and set these as the nodes. This creates an approximate probability distribution for the Markov chain which we can then perform a random walk on.

Whilst this process does not always create a Markov chain that satisfies all properties as defined earlier we will aim to use this model to approximate the underlying distribution of a particular piece/pieces of music. Hence we will assume that music can be represented in this way and perform (subjective) experiments later to see if this is an effective method for generating music.

2.4.2 Higher-Order Chains

To create higher-order Markov chains each node will contain a list of n notes, durations or note-duration pairs where n is a positive integer and represents the order of the Markov chain. Therefore, the nodes are every unique subsequence of notes of length n within the song.

Considering again the piece 'Mary Had A Little Lamb' (Figure 2.3), for a second-order Markov chain the nodes can be written as `(E,D)`, `(D,C)`, `(C,D)`, `(D,E)`, `(E,E)`, `(D,D)`, `(E,G)`, `(G,G)`, `(G,E)`. Note that the order in which notes appear is important e.g. `(C,D)`$\neq$`(D,C)`. As we increase the order the number of unique sequences m exponentially and therefore the transition matrix will be of size m by m.

To handle a wider range of pitches we can include a number next to each note to represent its octave using scientific pitch notation [21] e.g. `D4` means a `D` note in the fourth octave.

Chapter 3

Ethical Considerations

In this chapter, we discuss some decisions we have made relating to various ethical challenges within the project.

3.1 Markov Chains

Markov chains require input data to create a transition matrix and therefore the sequence they generate depends on the data. As we increase the order of the Markov chain the closer the generated sequence gets to the original piece of music (transition probabilities tend to either one or zero). If we assumed the order to be equal to the number of notes in the input piece we would end up copying it exactly. A meaningful question is, therefore, who owns the music generated?

Even when we use a first-order Markov chain (the lowest possible order of a Markov chain) there is a possibility of generating the original piece as a sequence, this is due to the way we construct the transition matrix. We can generalise this idea to every piece of music: A bag contains the names of every possible note that can be played on a piano, we then randomly draw an infinite sequence of these notes replacing them back into the bag each time. Will the note sequence of a piece such as Nocturne by Chopin be found within this sequence?

This problem is analogous to the 'Infinite monkey theorem' [22] which says that a monkey hitting keys at random on a typewriter for an infinite amount of time will type any text. Therefore by drawing an infinite sequence of notes, every finite melody that has or will be created (simplified into single notes and without stylistic features) could be considered a subsequence of this infinite sequence.

The probability of randomly drawing the exact notes of an existing song is incredibly low but with the constraints on the probabilities of choosing notes due to the Markov model this probability is increased. In the analysis section of this project, we will look at the effect that increasing the order of the Markov chain has on the similarity of a randomly generated piece to the original piece.

Not only can whole songs be copied but also phrases within songs, a famous example of this is when representatives of Queen and David Bowie threatened Vanilla Ice with a copyright infringement lawsuit for the bassline in the song ‘Ice Ice Baby’ [23]. The bassline in ‘Under Pressure’ by Queen and David Bowie contains only seven notes and two different pitches but is very recognisable in the song ‘Ice Ice Baby’ even though the two songs are in different keys and have different tempos. Jazz songs are often transposed and played in different keys, even though all notes in a transposed song are different from the original piece it is still essentially the same song. In lawsuits about music copyright infringement the decision is made on a case by case basis and is almost always subjective.

Sampling is defined as “The extraction of portions of sound—‘samples’—from recorded media, and their reuse as material for new recordings.” [24]. To sample legally you must obtain permission from the copyright holder of the music [25]. Assuming music generation from an input piece is a form of sampling then the copyright holder will own the generated piece. If we wanted to use a copyrighted piece of music we must first get permission to use it. Therefore, in this project, we do not claim to own any of the music that the program generates.

3.2 Input Data

All training data used in this project will be from the public domain and copy-right free or under Creative Commons or other similar licenses.

3.3 Data Collection

As this project will rely on data collected from surveys it is important to follow ethical conduct. A risk assessment will be carried out before deploying any surveys. Personal data will not be collected and anonymisation will be used wherever possible.

Chapter 4

Design and Implementation

In this chapter, we discuss the design and implementation of Markov chain music generation and justify the decisions we made during development.

4.1 Software and Tools

Python: We will implement Markov chain music generation in Python. Python has been chosen due to its flexibility and range of libraries with extensive documentation.

MusicXML: MusicXML [26] is a standardised format for digital sheet music. It will be used in this project as input data and output data. MusicXML has support for many applications and a large amount of documentation.

ABC Notation: ABC is a standard text-based music notation used mainly for folk and traditional music. This format will be used as input data.

MIDI: MIDI is a standardised format to interconnect computers with musical instruments. This format is used as input data and output data.

music21: music21 [27] is a Python library used to analyse and manipulate musical data. This will be the main library used to import music data from MusicXML, ABC notation or Midi formats into Python. We can then use music21 to manipulate this data and output the generated music as MusicXML or Midi files.

NumPy: A comprehensive mathematical function library. NumPy will be used for creating the matrices required to hold the transition probabilities and any other matrices/vectors required. It will also be used for random number generation for determining a random walk.

MuseScore: MuseScore is a free music composition and notation software. MuseScore can be used to display and play audio from MusicXML files, with music21 we can use a single command to show and listen to a piece of music in MuseScore.

R: R is a programming language with extensive libraries for statistical computation. It will be used in our evaluation to fit a model to some collected data and perform statistical tests.

4.2 Architecture

In Figure 4.1, we show an architecture diagram for implementing Markov chain music generation in Python using the above tools. The red section describes processes that use the Markov chain object and the blue rectangles describe the sections in which the `music21` library is used.

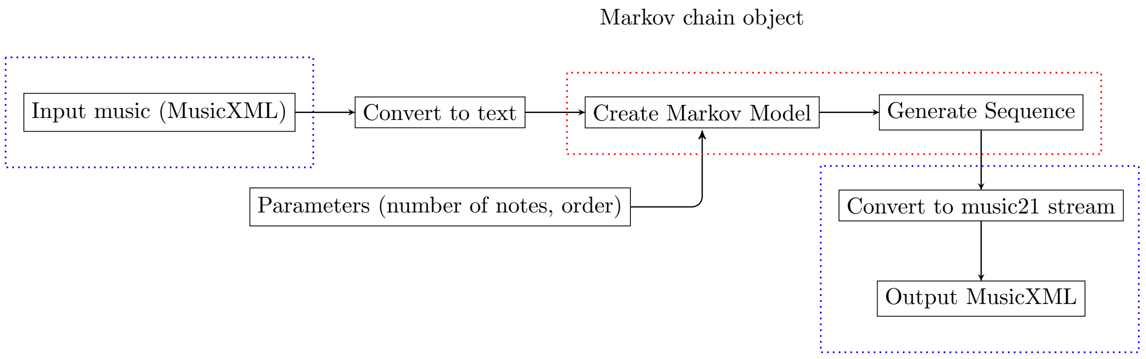


Figure 4.1: Markov chain music generation diagram

4.3 Simplification of Music Input

To process the data we will lose some information about the original piece, for this project we will only look at the pitch and duration of notes (including rests). While processing the input data we will remove:

1. Stylistic features (tempo changes and dynamics)
2. Chords (replace with chord root note)
3. Multiple clefs/ polyphony (only retrieve the information from the treble clef/highest clef)
4. Complex note lengths (the output will only contain full, half, quarter, eighth, 16$^{\text{th}}$, 32$^{\text{nd}}$ and 64$^{\text{th}}$ notes)

The features above are very important devices in music to convey emotion; quantised music often sounds ‘robotic’. Unfortunately, their removal is necessary to create a Markov chain model. As many other musical features are not tied to single notes but groups of notes or bars it is not possible to include them in the Markov chain model. It may be possible to add more general features later using other random generation techniques without using a Markov chain. This idea will be discussed further in the analysis chapter.

4.4 Implementation

4.4.1 Markov Chain Object

The first step of implementation was to create an object class in Python which handles the creation and processing of a Markov chain. This class is generalised so that it can be used in other applications and is not specific to music generation. It contains the following class attributes:

1. `transition_matrix` - A 2-dimensional NumPy matrix used to hold transition probabilities.
2. `order` - Integer to represent the Markov chain length/ order.
3. `data` - Ordered list of strings (e.g. musical notes) to be used to create the transition matrix.
4. `states` - List of node names as strings.
5. `index_dict` - Dictionary to retrieve a states string name from an integer index ID.
6. `state_dict` Dictionary to retrieve a state's integer ID (position in state's list) from a state's name.
7. `initial_probability` - Vector containing probabilities proportional to the frequencies of each node in the data list. Each value corresponds to a single node/state in `states`.

The object's constructor requires `data` and `order` to instantiate the Markov chain object.

4.4.2 Input Data Conversion

To train the Markov model we need an ordered list of notes with their pitch and or duration values. Music21 handles most of this, the input MusicXML, MIDI or ABC file is converted to a music21 `Stream` object which is effectively a container or list for notes and can represent a whole song. Each note in the song is represented as a `Note` object which has attributes such as duration and pitch. The `Stream` contains ordered `Note` objects. To extract the note values we simply iterate through the stream and add the `Note` duration and pitch attributes as a string into a list. When a rest (a pause in written music) occurs we append a value of `"R"`. We have three generation methods which are as follows,

1. Melody Markov chain.
2. Combined melody and rhythm Markov chain.
3. Rhythm Markov chain.

For the melody Markov chain, we retrieve the pitch value e.g. '`E5`' and add this to the list. To deal with rhythm we put a text representation such as '`whole`', '`half`', '`quarter`', '`eighth`' or '`sixteenth`'. When we use the combined melody and rhythm Markov chain we concatenate the pitch value with the rhythm value and a comma between e.g. '`E5,quarter`'.

Applying this method to 'Mary had a little lamb' (Figure 2.3) for the melody only Markov chain produces the lists of notes shown in Figure 4.2. The program produces the unique

```
['E5', 'D5', 'C5', 'D5', 'E5', 'E5', 'E5', 'D5', 'D5', 'D5', 'E5', 'G5',
'G5', 'E5', 'D5', 'C5', 'D5', 'E5','E5', 'E5', 'E5', 'D5', 'D5', 'E5',
'D5', 'C5', 'R']
```

Figure 4.2: Sequence of notes in the song 'Mary Had A Little Lamb'

states for each generation method respectively for a first order Markov chain:

1. `['E5', 'D5', 'C5', 'G5']`
2. `['E5,quarter', 'D5,quarter', 'C5,quarter', 'E5,half', 'D5,half', 'G5,quarter', 'G5,half', 'C5,half']`
3. `['quarter', 'half']`

For chains with order 2 or higher we concatenate state names with a pipe symbol `'|'`, the outputs shown below are from a second order Markov chain for each generation method respectively:

1. `['E5|D5', 'D5|C5', 'C5|D5', 'D5|E5', 'E5|E5', 'D5|D5', 'E5|G5', 'G5|G5', 'G5|E5']`
2. `['E5,quarter|D5,quarter', 'D5,quarter|C5,quarter', 'C5,quarter|D5,quarter','D5,quarter|E5,quarter', 'E5,quarter|E5,quarter', 'E5,quarter|E5,half', 'E5,half|D5,quarter', 'D5,quarter|D5,quarter', 'D5,quarter|D5,half', 'D5,half|E5,quarter','E5,quarter|G5,quarter', 'G5,quarter|G5,half', 'G5,half|E5,quarter', 'D5,quarter|C5,half']`
3. `['quarter|quarter', 'quarter|half', 'half|quarter']`

Even though the piece is quite small (contains 26 notes in total) we can see that the number of unique states increases by a large amount as we change the order from one to two which follows the statement in Section 2.2 about the increase in model parameters with respect to the order.

4.4.3 Markov Model Creation

To create the Markov model we assume that the input music is generated by some underlying probability distribution. We aim to estimate the conditional probabilities by first calculating a transition frequency matrix, this is done by iterating through the list of notes/nodes and adding 1 to the respective position in the frequency matrix. An example transition frequency matrix calculated on the 26 notes from ‘Mary Had A Little Lamb’ is shown in Figure 4.3. To estimate the transition probabilities we simply divide each row

```
['E5','D5','C5','G5']

[[6. 5. 0. 1.]
 [4. 3. 3. 0.]
 [0. 2. 0. 0.]
 [1. 0. 0. 1.]]
```

Figure 4.3: Python output: transition frequency matrix for the song ‘Mary Had A Little Lamb’

```
['E5', 'D5', 'C5', 'G5']

[[0.5        0.41666667 0.         0.08333333]
 [0.4        0.3        0.3        0.        ]
 [0.         1.         0.         0.        ]
 [0.5        0.         0.         0.5       ]]
```

Figure 4.4: Python output: transition probability matrix for the song ‘Mary Had A Little Lamb’

in the frequency matrix by the sum to ‘normalise’ the frequencies into probabilities (range $[0, 1]$). The calculated transition probability matrix is shown in Figure 4.4, which is equivalent to the matrix shown in Figure 2.4. When a piece ends on a unique state/note, one or more rows in the transition frequency matrix can contain zero in every entry, in this case, we cannot normalise the rows (due to division by zero) so instead, we replace the values of the row with the `initial_probability` vector. This occurs more frequently with higher order chains and chains that include rhythm with the melody.

Preventing Absorbing States

Definition 2 states that if a state is *absorbing* the only possible transition is back to itself. This means that the transition probability matrix row corresponding to the state will have all zeroes except for a one at the transition to itself (on a diagonal element). When this happens the music will constantly repeat the same note over and over. To avoid this we can check each row for probabilities of one on the diagonal of the matrix and zeroes everywhere else. When this occurs we can replace the row in the transition matrix with the `initial_probabilities` vector to make sure the generation does not repeat a note many times and 'stagnate'.

4.4.4 Other Generation Methods

We implemented two other methods for generating melody and rhythm to compare the model. These are:

1. Uniform random notes - Randomly chosen notes and lengths were sampled uniformly (with equal probability) from those found in the training pieces.
2. Original rhythm - The rhythm from the training pieces was directly copied into the newly generated sequence. If the original piece has fewer notes than the generated piece we re-use the original rhythm as many times as needed.
3. Constant rhythm - Each note is given a specified constant value e.g. "all notes are quarter notes".

4.4.5 Sequence Generation

After creating the Markov chain object and calculating the transition matrix we can use it to perform a random walk across the nodes to generate a new sequence of nodes. This is implemented using NumPy `random.choice()` function which chooses a value from `states` for a given row of probabilities from `transition_matrix`. We take the probabilities from the row of the current node to produce a random next node, the next node is then set as the current node and the process is repeated until the number of generated notes n is equal to the required amount (this is set by the user).

4.4.6 User Parameters and Model Selection

When running the application the user has the ability to choose:

1. **Melody generation method:**
   1.1. Independent melody Markov chain.
   1.2. Combined melody and rhythm.
2. **Rhythm generation method:**
   2.1. Independent rhythm Markov chain.
   2.2. Constant rhythm.
   2.3. Original rhythm.
3. **Markov chain order (for melody, rhythm or combined).**
4. **Number of notes to generate.**
5. **File containing training samples.**

6. **Option to save the generated piece.**

These are are options in the command line interface.

4.4.7 Output

After returning the sequence of notes, a new output music21 `Stream` object is created. We iterate through the generated sequence and retrieve each notes pitch and length values, assigning them each to a `music21 Note` object. These notes are then added to the output `Stream` which can then be saved as a MIDI file or viewed as sheet music (musicXML format) in the MuseScore application.

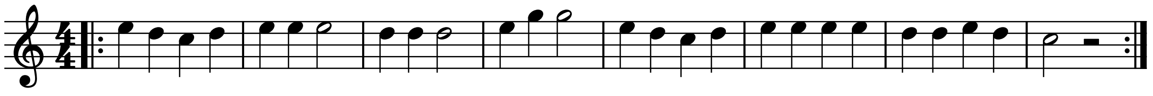


Figure 4.5: Original piece: "Mary Had A Little Lamb"

In Figure 4.6 we show a sample output of the program for the input data in Figure 4.5. A single first-order Markov chain for melody and rhythm was used to generate this sample. Figure 4.7 contains a section of generated music for a more complex example - "Nocturne in E-flat major, Op. 9, No. 2" by Frédéric Chopin.

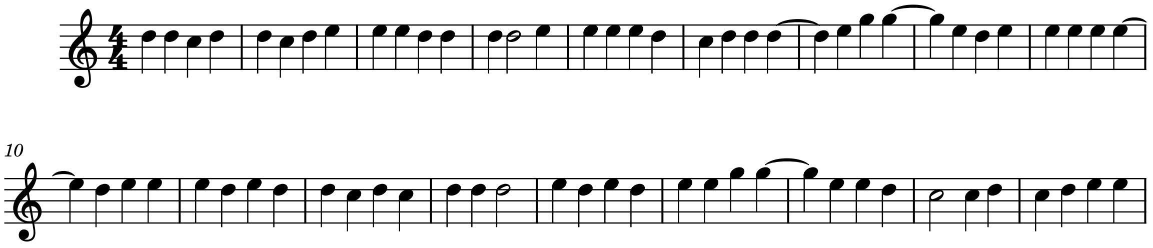


Figure 4.6: Generated piece for "Mary Had A Little Lamb"

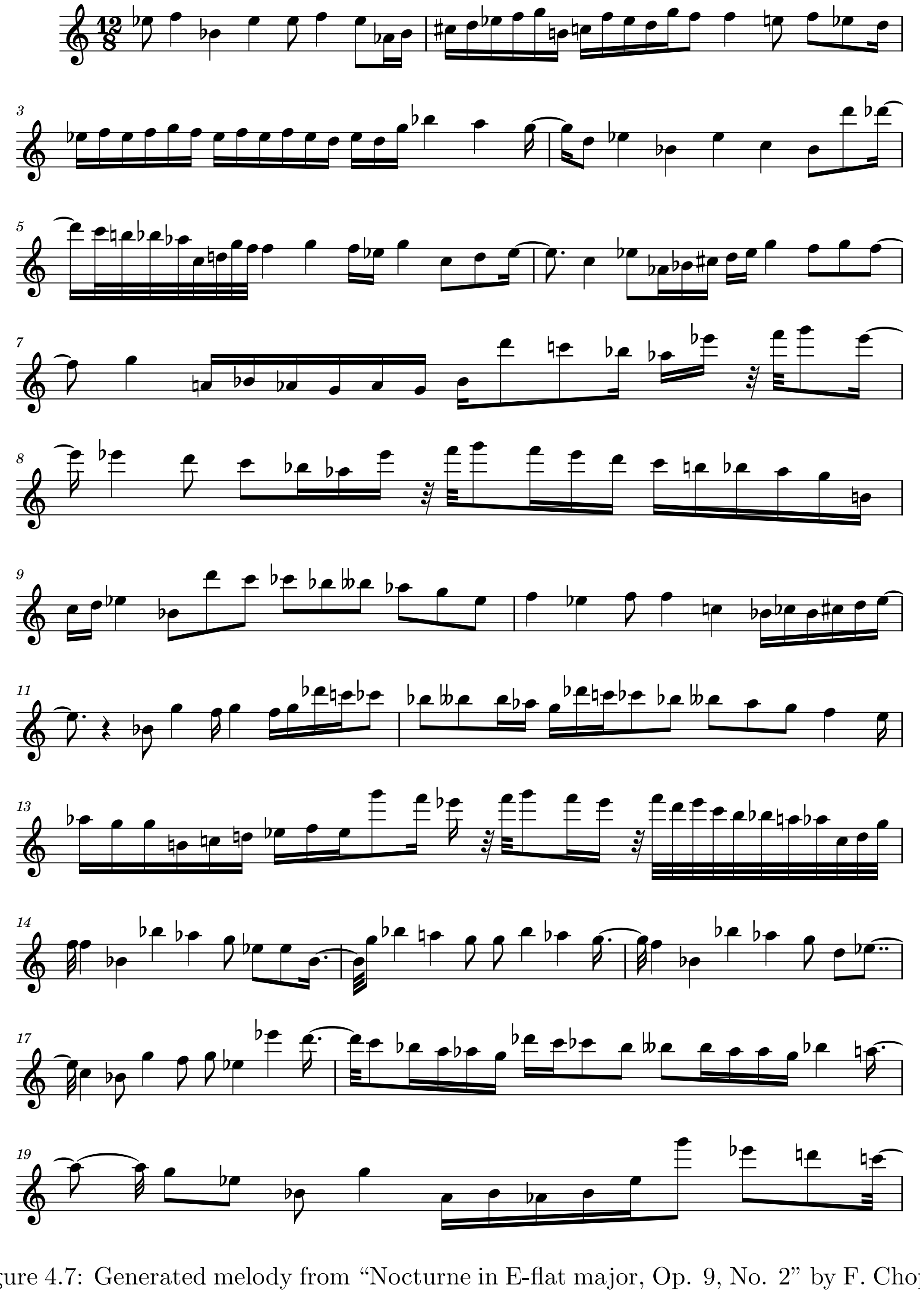


Figure 4.7: Generated melody from “Nocturne in E-flat major, Op. 9, No. 2” by F. Chopin

Chapter 5

Results Analysis

In this chapter, we will discuss and carry out the process of evaluating if the aims of the project were met. The procedure, tools and methods used for statistical evaluations are described.

5.1 Analysis Procedure

The aim of this project was to generate new music from existing pieces of music using various methods including Markov chains. It is not possible to objectively determine whether the generated music is 'enjoyable' so we will perform an analysis of data gathered using a survey to draw conclusions about the outcome of this project. The survey questions have been designed to compare different methods of music generation directly. Some useful statistical definitions can be found in Appendix A.

5.1.1 Observations During Testing

Multiple Training Pieces

Using more than one piece of music to create the transition matrix produced some interesting results. When two pieces using different scales were used (depending on the order of the Markov chain) sections from both pieces were reproduced and combined into a single piece. In the case where the scales used were the same or very similar the piece merged more and produced music that was much more unique.
The pieces generated when the inputs have the same scale were more 'pleasing'/ sounded a lot less dissonant compared to using input pieces with different scales. If the scales used are significantly different (share very few notes) and the Markov chain order is greater than the number of notes the scales share then nodes created from other pieces are rarely/never reached. This is because the nodes generated from one piece have a very small probability of transitioning to nodes from the other piece. This is especially true when the Markov chain order is high as the pieces will share little or no musical note subsequences.

High order Markov chains

As the order of the Markov chain increases all transition probabilities tend to 1 or 0. When a transition matrix only contains binary values the system is therefore no longer probabilistic and is deterministic. Therefore the sequence can be determined before it is generated. In this case, the random walk returns the original training piece. During testing we observed that, depending on the complexity of the piece every transition probability can be close to 1 or 0 at orders as low as $n = 6$.

5.1.2 Survey

We will carry out the survey using Google forms. To allow participants to listen to extracts generated by our program we will generate the music first, export each piece in MIDI, convert MIDI to MP3 files, upload unlisted videos on YouTube using the MP3 file as music and link them within the Google form. The survey is entirely anonymous and contains no data collection of personal information.

Questions

The first section of the survey aims to analyse the effect of changing the Markov chain's order with a combined melody and rhythm model. Each participant will be asked to listen to 4 extracts with the generation methods: first-order Markov chain, uniformly random, sixth-order Markov chain (high order almost exactly reproduces original song), a second sample of the first-order Markov chain. They will then be asked to rate each of the samples on a scale from 1-5 where 1 is 'not enjoyable to listen to' and 5 is 'most enjoyable and pleasant to listen to'. We will aim to fit a linear regression model to the resulting data. Ideally, the order in which extracts are displayed in the quiz would be randomised to reduce bias but this is not possible in Google forms. Each sample will be given a random letter (A-D) to make sure there is no bias from reading the name of the sample.
The second section of the survey aims to compare three different methods of rhythm generation whilst maintaining the same Markov chain order of 2. The methods of generating rhythm are as follows:

1. Constant rhythm: each note is an eighth note.
2. Original rhythm: rhythm values are directly copied from the training piece.
3. Independent rhythm Markov chain order 2: rhythm is generated independently from the melody using its own second-order Markov chain.

This is rated on the same scale from 1-5 as the first section.
Finally, the participants will be asked to state their level of musical ability from:

1. Beginner - Very little or no knowledge of music theory.
2. Intermediate - Some knowledge of music theory or can play an instrument casually.

3. Advanced - Large knowledge of music theory and can play an instrument at a high level.

4. Professional - Working/professional musician.

We will place this question at the end of the survey to avoid some question order bias.

Sources of Bias

Bias can cause skewed results leading to incorrect conclusions. In our survey, we will consider the following forms of bias [28]:

1. **Question order bias** - respondents may react differently to questions based on the order the questions appear. To avoid this bias we will group related questions, randomly select the order of questions within these groups and place more specific questions later on in the survey. The questions should be randomly ordered within sections for each participant, with a large enough sample size the bias should average out.

2. **Participant bias** - occurs when individuals involved in an experiment act or respond in ways they believe correspond with what the researchers are looking for. To avoid this we will try to not shape people's beliefs before they take the survey.

3. **Selection bias** - occurs when individuals in a study differ systematically from the population of interest. To avoid this we will aim to get survey respondents that are representative of the general population.

4. **Non-response bias** - occurs when individuals that do not respond to a survey are systematically different to those who participate. To avoid this we will design the survey to be easy to understand, short enough that people complete it and anonymous. This should increase completion rates among potential participants.

Many of these biases require expensive methods and much larger sample sizes than we have access to in order to minimise. Therefore before carrying out any evaluation we must acknowledge the existence of this bias and not draw definite conclusions from our survey.

5.1.3 Limitations of survey analysis

As we have to compare many different methods it was not possible to include a large number of musical extracts without the survey being too long. Long surveys result in lower completion rates and therefore smaller sample sizes [29]. We created all musical extracts from a single training piece - 'Nocturne Op 9 No 2 (E Flat Major) - Frédéric Chopin'. A downside of this is that the output depends heavily on the quality of the input, hence it may not be possible to extrapolate the results from this survey for every piece of music. The positive of using a single piece for all generated extracts is that we can directly compare the generation methods without having to account for differences in training data. The length of each extract was limited to roughly 30 seconds, to reduce bias in selecting

musical extracts we generated multiple 30-second extracts for each method and selected one at random to use in the survey.
Furthermore, the 30-second extracts are nowhere near fully representative of a much longer generated piece and hence there will be high variability in the results. The results depend heavily on the training data and section of generated music. Ideally, we would have multiple extracts for each generation method and repeat this for different input data (eg. 3 different input data sets, 3 samples for each generation method, a total of 9 samples for each method). This would allow us to average over the ratings to obtain a more reliable average rating. However, this was not possible with the scale of this project. To check for this bias a second sample for the 'combined melody and rhythm generation' method with order 1 was included in the survey to compare to itself.
Finally, it may be hard to find people to complete the survey that have a high level of musical knowledge or theory (e.g professional musicians) and therefore their sample will not represent the population exactly. As this analysis only aims to find a correlation using people's subjective opinions and ratings it should not be a large problem, this is because there will be noise in the data due to how different people approach ranking/rating each extract. We will check models including and excluding the musical ability variables to see if this has any effect on how people rank the pieces.

5.1.4 Multiple Linear Regression

One aim of this project is to find an 'optimal' balance between a completely random sequence of notes (restricted to a single key) and an existing piece using the same notes. To do this we will try to find a relationship between Markov chain order and the rating/ranking of each method from the survey using a multiple linear regression model.

Regression analysis is the process of estimating the relationship between a dependent variable and one or more independent variables [30]. These models are in the form

$$Y = f(\mathbf{X}, \boldsymbol{\beta}) + \epsilon \tag{5.1}$$

where Y is the dependent variable, also known as the response, $\mathbf{X}$ is a vector of independent/explanatory variables, $\boldsymbol{\beta}$ is a vector of unknown parameters that we wish to estimate and ϵ represents an additive error term. A specific instance of this model is called *multiple linear regression* which assumes the relationship between the explanatory variables and the response is linear. This means the function $f(\mathbf{X}, \boldsymbol{\beta})$ in Equation 5.1 is a linear function of the explanatory variables $X_1, X_2, ..., X_n$. From our survey, we will have two explanatory variables which are the order of the Markov chain/method of generating music and the level of musical ability. With the data obtained from the survey, we will test multiple models and use statistical tests to compare them. Hence, an example model will be of the form,

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_{12} X_1 X_2 + \epsilon. \tag{5.2}$$

In linear regression we assume that the errors ϵ are normally distributed with expectation $\mathbb{E}[\epsilon] = 0$ and variance $Var(\epsilon) = \sigma^2$ where σ^2 is an unknown constant. We can write,

$$\epsilon \sim N(0, \sigma^2). \tag{5.3}$$

Using sample data we can find the estimates $\hat{\boldsymbol{\beta}}$ for $\boldsymbol{\beta}$ using least squares estimation. Least squares estimation aims to minimise the sum of square errors. This means that the estimated responses for each sample are as close to the true value as possible. We can also estimate the value of σ^2 which is the variation in the errors.
From Part 1 in the survey, the models we will test are

1. X_1 is the order of the model. We will use $X_1 = 0$ to represent the uniform random extract. This model assumes the Markov chain order (X_1) is a continuous variable, this has the drawback of needing to assign numeric variables to the uniform random extracts and original piece and therefore may not fit the data well.

2. Use categorical variables for both the 'method of generation' and 'level of musical ability'. We show this equation in Equation 5.5.

In Part 1 of the survey the three categories for 'method of generation' are: 'uniform random', 'order 1', 'order 6' and the three categories for musical ability are: 'beginner', 'intermediate' and 'advanced'. Dummy variables can be used to include categorical variables in models, they are defined as,

$$X_1 = \begin{cases} 1, & \text{if sample belongs to 'order 1' class,} \\ 0, & \text{otherwise.} \end{cases} \tag{5.4}$$

Hence we can write a categorical model for Part 1 of the survey using dummy variables as,

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 Z_1 + \beta_4 Z_2 \tag{5.5}$$

Where, Y (the response) is the rating out of 5, β_0 is the mean response (rating) for a sample with the variables 'uniform random' and 'beginner', β_1 is the difference in mean between 'uniform random' and 'order 1' samples, β_2 is the difference in mean between 'uniform random' and 'order 6' samples, β_3 is the difference in mean between 'beginner' and 'intermediate' samples, β_4 is the difference in mean between 'beginner' and 'advanced' samples.

5.1.5 Multiple R^2 correlation coefficient

Pearson correlation coefficient is a measure of the linear correlation between two variables and is in the range $[-1, 1]$. A value of 1 means there is a perfect positive linear correlation and a value of -1 means there is a perfect negative linear correlation. In the multivariate case, we use the multiple R^2 correlation coefficient which represents the percentage of the variation in the data that is explained by the fitted model, it is in the range $[0, 1]$. A value of 1 means all the variation is explained by the model [31].

5.1.6 Nested Models

A nested model is a regression model that contains a subset of the predictor variables in another regression model. Below are two models `MODEL1` and `MODEL2`:

$$\texttt{MODEL1} : Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon. \tag{5.6}$$

$$\texttt{MODEL2} : Y = \beta_0 + \epsilon. \tag{5.7}$$

MODEL2 only contains an intercept variable and is therefore nested in MODEL1. An intercept variable is the mean value of the response Y when all other variables in the model equal zero. Hence, if we assume some data fits MODEL2 we are saying that the response is a constant value and does not change when the independent variables are changed. This means we assume that the response is not affected by any explanatory variables and all variance in the data is random error.

5.1.7 Analysis of Variance

To test if categorical explanatory variables are significant we can use analysis of variance (ANOVA). ANOVA uses the law of total variance (Equation 5.8) to decompose the total variance into components attributable to different sources of variation [32].

$$Var(Y) = \underbrace{E[Var(Y|X)]}_{\text{Unexplained Variance}} + \underbrace{Var(E[Y|X])}_{\text{Explained Variance}}. \tag{5.8}$$

This allows us to compare the mean values for each category and check if we should keep them in the model. R contains a function `anova()` which takes an input of a model created in R and outputs an 'ANOVA' table which can be used to draw conclusions about the model.
This function can also be used to compare nested models using hypothesis testing.

5.1.8 Hypothesis

We wish to test that the rating/response (on a scale from 1-5) depends on the method of generating music and the musical ability of a person. This means our hypothesis can be written as,

$$H_0 : \boldsymbol{\beta} = \mathbf{0},$$
$$\text{against,}$$
$$H_1 : \text{at least 1 } \beta_i \neq 0, i \in \{1, ..., n\},$$

where $\boldsymbol{\beta}$ represents the vector of model parameters and β_i is the i^{th} parameter $\in \{\beta_1, ..., \beta_n\}$. H_0 is the null hypothesis and H_1 is the alternative hypothesis. This is the same as comparing the nested model that only includes an intercept to the model that includes the parameters for generation method and musical ability. All statistical tests will be carried out at a 5% significance level.

We will also compare the models using only categorical explanatory variables against a continuous 'generation method' variable.

In words, our hypothesis is that as you increase the Markov chain length the more pleasing/enjoyable the generated music should sound. We also hypothesise that using Markov chains to generate rhythm creates more pleasing/enjoyable pieces than using a constant note length but not as pleasing/enjoyable as the rhythm from the original piece for a Markov chain generated melody.

5.2 Survey Results

The generated music samples used within the survey can be found on YouTube by clicking **here**. The labels for each piece are explained later in this section.

5.2.1 Data Processing

To use the data in R we transform the data from Google forms output as a CSV file into a new CSV file where each row contains one sample and the columns have the labels: rating (response), generation method, musical ability.

```
Rate Extract A, Rate Extract B,What level is your musical ability?

3        1        Beginner (Very little/no knowledge of music theory)
1        2        Beginner (Very little/no knowledge of music theory)
5        1        Intermediate (Some knowledge of music theory)
2        1        Beginner (Very little/no knowledge of music theory)
```

Figure 5.1: Extract from CSV data generated by Google forms output

```
rating,sample,ability_level
3,A,beginner
1,A,beginner
5,A,intermediate
2,A,beginner
4,A,intermediate
4,A,intermediate
3,A,intermediate
```

Figure 5.2: Section from updated survey sample data CSV file

In total there were 23 respondents who completed the whole survey. Figure 5.3 shows the distribution of musical abilities among respondents. Unfortunately, only 1 respondent was from the ‘advanced’ category and no respondents were professional musicians.

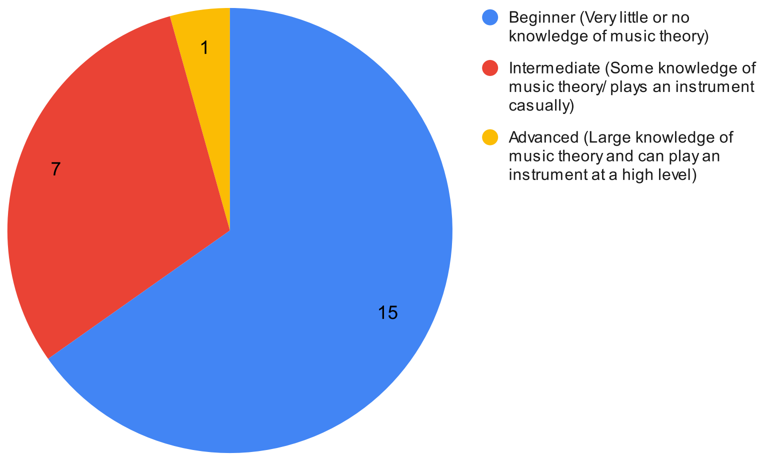


Figure 5.3: Distribution of musical abilities among respondents in the survey

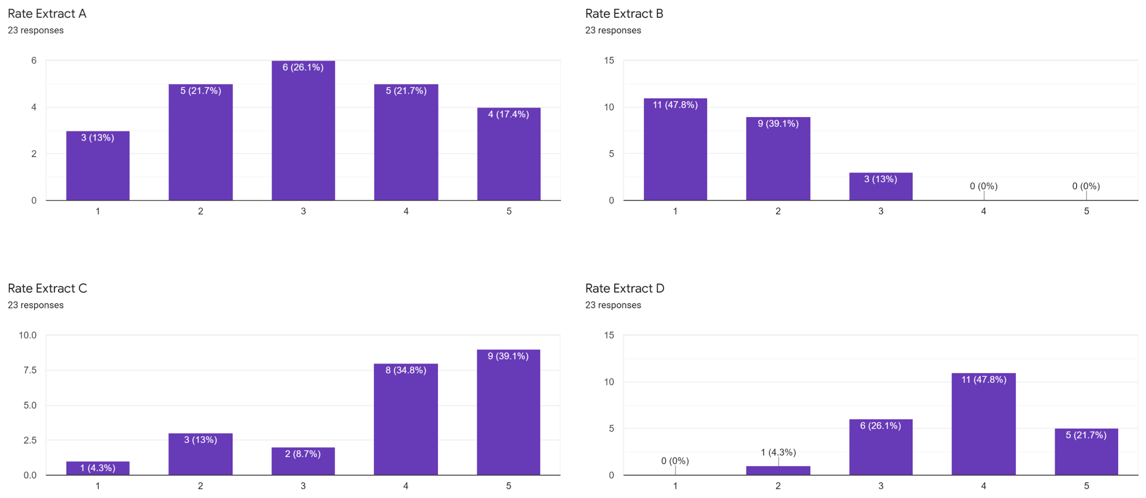


Figure 5.4: Bar charts for survey section 1 rating distributions

```
Call:
lm(formula = rating ~ sample + ability_level)

Residuals:
     Min       1Q   Median       3Q      Max
-2.98024 -0.86702  0.01976  0.88792  2.24506

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                3.00988    0.63835   4.715  9.0e-06 ***
sample                     0.24506    0.05588   4.385  3.2e-05 ***
ability_levelbeginner     -0.50000    0.64910  -0.770    0.443
ability_levelintermediate -0.14286    0.67189  -0.213    0.832
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.257 on 88 degrees of freedom
Multiple R-squared:  0.1937,        Adjusted R-squared:  0.1662
F-statistic: 7.045 on 3 and 88 DF,  p-value: 0.0002673
```

Figure 5.5: Output from R `summary()` function for multiple linear regression model with continuous ‘generation method’ variable

5.2.2 Survey Section 1 - Combined Melody and Rhythm Generation

Before fitting any models to our data we will look at the distribution of ratings for each extract. Figure 5.4 contains a bar chart of the ratings given by respondents for each extract. By inspection we can see that most of the samples have a somewhat normal distribution around their most frequent value, when the mean value is close to 1 or 5 this distribution is skewed due to the ratings being discrete and over a limited range of (1,5). Each extract seems to have a unique distribution hence we should be able to fit regression models to this data. We will first consider a model that contains a continuous value for ‘generation method’. The *R* output summary for this model is shown in Figure 5.5. From this output, we can see that the `(Intercept)` and `sample` (this is the extract and relates to the generation method/Markov chain order) variables are significant as they have ‘`***`’ on the right-hand side of the table. The p-value = 0.0002673 relates to the statistical test of comparing the created model to the model without any parameters. Hence at a 5% significance level there is significant evidence to say there exists some linear relationship between the explanatory variables and the rating. However, the correlation coefficient (Multiple R-squared value) = 0.1937 which implies there exists a weak positive correlation. The Markov chain order/ method of generation is probably not a linear relation to the rating of an extract which explains why the correlation is quite weak.

```
Call:
lm(formula = rating ~ sample + ability_level)

Residuals:
    Min      1Q  Median      3Q     Max
-2.7826 -0.7391  0.1211  0.6863  2.0435

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                 3.4565     0.5513   6.270 1.39e-08 ***
sampleB                    -1.4348     0.3058  -4.692 1.01e-05 ***
sampleC                     0.8261     0.3058   2.702  0.00831 **
sampleD                     0.7826     0.3058   2.559  0.01223 *
ability_levelbeginner      -0.5000     0.5355  -0.934  0.35305
ability_levelintermediate  -0.1429     0.5543  -0.258  0.79722
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.037 on 86 degrees of freedom
Multiple R-squared:  0.4637,	Adjusted R-squared:  0.4325
F-statistic: 14.87 on 5 and 86 DF,  p-value: 1.68e-10
```

Figure 5.6: Output from R `summary()` function for multiple linear regression model with discrete ‘generation method’ variable and uncombined A and D samples.

Next, we will consider the model where the sample/extract is categorical and we are comparing the means for each category. The samples are as follows:

- Sample A: Markov chain order 1.
- Sample B: Uniform random generation.
- Sample C: Markov chain order 6.
- Sample D: Markov chain order 1.

Figure 5.6 shows the *R* output for this model. Compared to the model in Figure 5.5, we see that the correlation coefficient is much larger with a value of 0.4637 and a much smaller p-value in the new model. This indicates that our new model with categorical variables fits the data much better.

Now, we will compare the model in Figure 5.6 with a new model in Figure 5.7. This new model removes the `ability_level` variable and is therefore nested in the previous model. We can compare the two models directly using the `anova()` command, the result is shown in Figure 5.8. This carries out an F test to check if the `ability_level` variable is

significant in the model. The resulting p-value for the test is 0.2527 which is greater than 0.05 (5% significance level) hence the more complex model is not significantly better and we can remove the `ability_level` explanatory variable from the model.
This model (Figure 5.7) only uses categorical variables for the generation method and seems to fit the data well with a slightly higher proportion of the variation being explained by the model (multiple $R^2 = 0.4463$) and a higher F-statistic. Looking at the estimated regression parameters for this model we can read the estimated response for each sample:

- sampleA - The expected response for sample A is given by the (Intercept) estimate term in the table. This is 3.087. The generation method used in sample A was a first-order Markov chain.
- sampleB - The expected response for sample B = (Intercept) value + sampleB estimate from table = 3.0870 - 1.4348 = 1.6522. The generation method used for sample B was uniform random music generation.
- sampleC - The expected response for sample C = (Intercept) value + sampleC estimate from table = 3.0870 + 0.8261 = 3.9131. This sample was generated using an order 6 Markov chain and closely resembled the original piece.
- sampleD - The expected response for sample D = (Intercept) value + sampleD estimate from table = 3.0870 + 0.7826 = 3.8696. This sample was generated using a first order Markov chain, the same as sample A. The difference between the two samples (0.7826) could be explained by the variation in quality in selecting the samples. It could also be due to being the final question in the survey's first section compared with sample A being the first (question order bias) - it can be difficult to rate an extract without having anything to compare it to.

Looking at these results we can see a clear trend that supports our hypothesis that the higher the Markov chain (and further away from complete randomness) the higher the rating will be. If we combine the results from sample A and sample D and replace each extract name with the generation method we obtain the result in Figure 5.9. We see that the multiple R^2 value has decreased but overall the model still fits well as all variables are highly significant. As the multiple R^2 value has decreased after combining the samples using the same generation method we can conclude that there is a lot of variation between samples of the same generation method.

5.2.3 Survey Section 2 - Rhythm Generation

We will use the model with categorical sample variables for the rhythm generation method. Figure 5.10 shows the R summary output for this model. Each of the 3 regression parameters in this model are highly significant as their rows contain the significance code `'***'`. Hence we can conclude that the mean ratings from the survey were significantly different from each other. We can read the estimated regression parameters for the model and find the expected mean response/rating for each sample as follows:

```
Call:
lm(formula = rating ~ sample)

Residuals:
    Min      1Q  Median      3Q     Max
-2.9130 -0.6522  0.1087  0.9130  1.9130

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0870     0.2172  14.213  < 2e-16 ***
sampleB      -1.4348     0.3072  -4.671 1.07e-05 ***
sampleC       0.8261     0.3072   2.689  0.00856 **
sampleD       0.7826     0.3072   2.548  0.01257 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.042 on 88 degrees of freedom
Multiple R-squared:  0.4463,	Adjusted R-squared:  0.4274
F-statistic: 23.64 on 3 and 88 DF,  p-value: 2.587e-11
```

Figure 5.7: Output from R `summary()` function for multiple linear regression model with discrete ‘generation method’ variable without ‘ability_level’ variable and uncombined A and D samples.

```
Analysis of Variance Table

Model 1: rating ~ sample + ability_level
Model 2: rating ~ sample
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     86 92.472
2     88 95.478 -2   -3.0062 1.3979 0.2527
```

Figure 5.8: ANOVA table comparing the nested model created by removing the `ability_level` variable.

```
Call:
lm(formula = rating ~ factor(sample))

Residuals:
    Min      1Q  Median      3Q     Max 
-2.9130 -0.6522  0.2174  0.5217  1.5217 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)       1.6522     0.2238   7.383 7.88e-11 ***
factor(sample)1   1.8261     0.2741   6.662 2.17e-09 ***
factor(sample)6   2.2609     0.3165   7.144 2.39e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1 

Residual standard error: 1.073 on 89 degrees of freedom
Multiple R-squared:  0.4054,	Adjusted R-squared:  0.3921 
F-statistic: 30.35 on 2 and 89 DF,  p-value: 8.943e-11
```

Figure 5.9: Output from R `summary()` function for multiple linear regression model with discrete ‘generation method’. The music samples using the same generation method are combined in this model.

- `sampleE` - This is the baseline variable for the model so the mean response is equal to the `(Intercept)` estimate which is equal to 2.2609. For this piece, each note is an eighth note so the rhythm is constant, as expected this received the lowest rating out of the samples E, F and G.

- `sampleF` - The expected response for sample F is given by `(Intercept)` + `factor(sample)F` = 2.2609 + 1.0870 = 3.3479. The rhythm for this sample is copied directly from the original training piece.

- `sampleG` - The expected response for sample G is the highest out of the three with a value of 2.2609 + 1.4783 = 3.7392. This sample's rhythm was generated from a separate Markov chain independent of the melody.

These results were not as hypothesised, in theory, the highest-rated piece should be from sample F which used the original piece's rhythm, then sample G (Markov chain generated rhythm) and finally, sample E with a constant rhythm. This could be explained by the variation in the quality of generated pieces found in the first section of the survey but also due to the simplification of the music. The training piece Nocturne Op.9 No.2 in E Flat Major by Chopin contains large sections of trills (rapid alternation between two notes) combined with tempo changes such as ritardando (gradual slowing of the piece), when these features are removed the rhythm generated often contained fast tempo sections of 16th notes which did not sound pleasing. By creating an independent Markov chain for rhythm generation the chance of generating long sections of 16th notes was reduced which could partly explain the difference in ratings. As the survey length was very limited and only 3 samples were compared these results are highly likely to be biased and any conclusions we make must consider this bias.

5.3 Survey Conclusions

From our survey there is some evidence to suggest that as we increase the Markov chain order, listeners find the music generated more enjoyable. There is also evidence to suggest that the use of Markov chains to generate a rhythm for a sequence of notes is more effective than using a uniformly random rhythm across a number of note lengths or using a constant rhythm throughout a piece. These results depend heavily on the quality and complexity of the input piece/pieces that the model is trained on. In general a Markov chain order from 1-3 is a good trade-off between uniform random note sampling and the original piece. If the order is greater than 3, depending on the input, a large number of musical phrases are directly copied and repeated from the original piece.

```
Call:
lm(formula = rating ~ factor(sample))

Residuals:
    Min      1Q  Median      3Q     Max 
-2.7391 -0.7391 -0.2609  0.7391  1.7391 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)       2.2609     0.2209  10.234 2.96e-15 ***
factor(sample)F   1.0870     0.3124   3.479 0.000896 ***
factor(sample)G   1.4783     0.3124   4.732 1.22e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.059 on 66 degrees of freedom
Multiple R-squared:  0.267,	Adjusted R-squared:  0.2448 
F-statistic: 12.02 on 2 and 66 DF,  p-value: 3.537e-05
```

Figure 5.10: Output from R `summary()` function of multiple linear regression model for comparing rhythm generation methods.

Chapter 6

Project Evaluation

In this chapter, we discuss the achievements, limitations, future work and conclusions of the project.

6.1 Limitations

Overall the music generated, in our opinion, was not perfect and usually not enjoyable to listen to for long periods of time. This is especially due to the lack of expression and polyphony (such as chords, a bass line or multiple melody lines) in the produced pieces. Unfortunately, it was not possible to effectively test how enjoyable long pieces were (longer than 10 minutes). More in-depth testing could be carried out in the future including obtaining opinions from professional musicians and using a larger sample size in the survey.

6.2 Future work

In this section, we will discuss possible future developments and applications of Markov chains within generative music and other generative music methods.

6.2.1 Application to ambient music

Some ambient music producers such as Aphex Twin use modular electronic synthesizers (Eurorack) which are generally monophonic. The generated sequences of notes from a Markov chain model could be used as an input for a modular synthesizer or a module could be created that implements Markov chain music generation. Similar modules already exist such as *mutable instruments 'Marbles'* random sampler which provides many ways of imposing structure on the random events generated by the module [33].

Rather than implementing Markov chain melody generation in hardware (which may not be possible) there are other modules such as the Eurorack module: *EuroPi* by *Allen Synthesis* [34] which contains a Raspberry Pi Pico. A Python script on the Raspberry Pi Pico can be written/edited to change the functionality of the module. Using this module we could implement a lightweight version of the code produced in this project on the Raspberry Pi to allow Markov chain melody generation and be used with other modules for live music or music writing.

Ambient music is defined as “a style of gentle, largely electronic instrumental music with no persistent beat, used to create or enhance a mood or atmosphere” [35]. Much of Brian Eno’s generative music work is considered to be ambient music [36]. The lack of a persistent beat and strong structure makes it an ideal candidate for the application of a Markov model for melody generation.

6.2.2 Musical Dynamics/ Polyphony

During this project, we felt that the melodies generated were often a bit ‘empty’. We could adapt our model to generate an accompanying chord sequence or bass line using separate Markov chain models. A 2017 paper, “*Rhythm Transcription of Polyphonic Piano Music Based on Merged-Output HMM for Multiple Voices*”, uses the concept of hidden Markov models (see Section 6.2.5) to synchronise multiple lines or voices. This is especially useful when the music is polyrhythmic (several different rhythms performed simultaneously [24]) or more complex note lengths such as triplets are used. This method could be implemented alongside multiple voices of music generation to create a coherent piece of music.

6.2.3 Non-homogeneous Markov chains

Non-homogeneous Markov chains include transition probabilities that depend on time. This idea could be used to transform the transmission probability matrix to vary over the course of a piece by including variables in the transition matrix and re-calculating the transition matrix for each time step or note. We could also explore random transformations to the transmission matrix such as multiplying by a rotation matrix.

6.2.4 Graphical Interface/ Web page

This project focused on the exploration and evaluation of simple Markov chain music generation. To allow people that may not be familiar with coding to explore and test this project’s implementation, a graphical interface and or web page could be a useful tool. The current command-line interface is not user friendly so integration of the Markov chain music generation on a web page would allow users to modify parameters and generate their own music.

6.2.5 Hidden Markov Models

Hidden Markov models (HMMs) have been shown to be an effective method for capturing the musical style of some training data and composing new music [37]. HMMs contain hidden states, their transmission and output probabilities can be estimated from observed output sequences generated by the Markov process by using dynamic programming [15]. In music generation, we use dynamic programming to predict the probabilities that result in a given song instead of estimating from note occurrence frequencies. The estimated model parameters can then be used to generate new sequences of notes.

6.3 Project Planning and Reflection

Whilst undertaking this project we used Jira, an issue tracking project management tool, to plan tasks for each week and maintain a development timeline. Figure 6.1 shows the general development timeline. Within each of these main tasks smaller tasks were set and assigned before the start of the next week. Jira has the added benefit of being able to link tasks with GitHub issues which was useful for keeping on track with the code development. The kanban board (Figure 6.2) is used to visualise which tasks are to be completed, in progress or completed. Each sub-task was assigned to a 'super-task' and assigned a deadline.

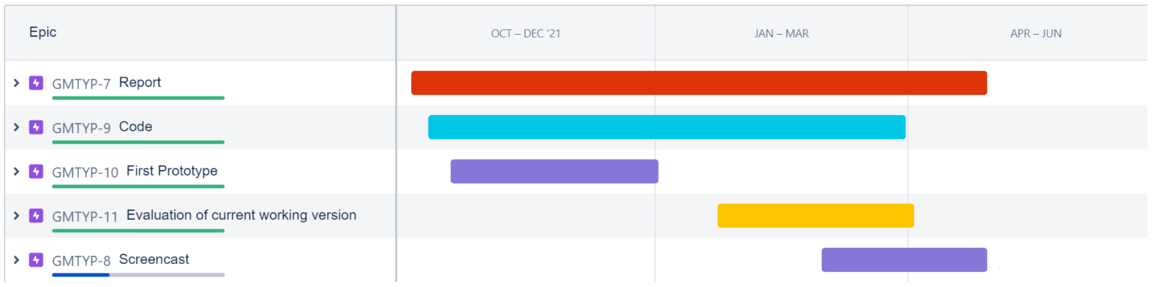
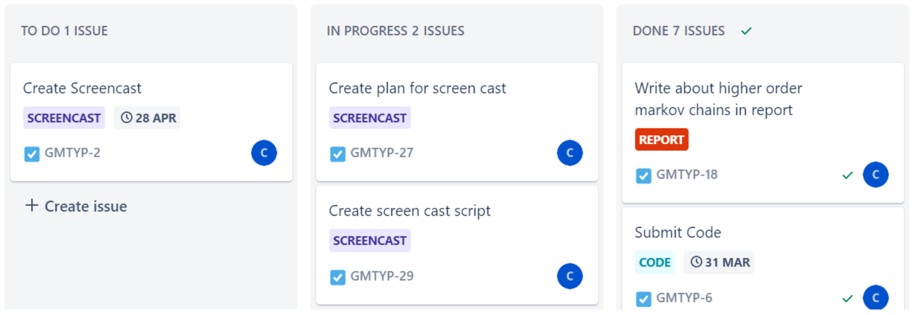


Figure 6.1: Jira Development Timeline

Overall this project could be considered a success, we have implemented and evaluated Markov chain music generation. However, the evaluation was not as rigorous as we had first wanted and the survey was extremely prone to bias. To improve, a longer survey with many different trained samples should be used and we should gather more data from experienced musicians and producers. We could have also tried to get written feedback on the samples from experienced musicians to further explore the limitations of the implemented model.


TO DO 1 ISSUE

Create Screencast
SCREENCAST 28 APR
GMTYP-2 C

+ Create issue

IN PROGRESS 2 ISSUES

Create plan for screen cast
SCREENCAST
GMTYP-27 C

Create screen cast script
SCREENCAST
GMTYP-29 C

DONE 7 ISSUES ✓

Write about higher order markov chains in report
REPORT
GMTYP-18 ✓ C

Submit Code
CODE 31 MAR
GMTYP-6 ✓ C

Figure 6.2: Jira kanban board

6.4 Achievements and Conclusion

In this project we have successfully generated simple melodies and their rhythm using a Markov chain model. We have briefly compared models using different parameters and methods for generating both rhythm and melody. The outputs from these models were evaluated, from this evaluation we conclude that there is evidence to suggest that Markov chains are an effective method for producing 'pleasing' melodies. The advantages of a Markov chain model over a machine learning model is that we require far less processing power [8] and can produce music based on a very small sample size , for example, 'Mary Had A Little Lamb' (Figure 2.3) has 8 bars of music and was able to produce long, albeit boring, pieces of music.

Bibliography

[1] R. Loydell and K. Marshall, “Sound mirrors: Brian eno & touchscreen generative music,” *Musicology Research*, vol. 3, pp. 27–50, 2017.

[2] A. Alpern, “Techniques for algorithmic composition of music,” *On the web: http://hamp. hampshire. edu/adaF92/algocomp/algocomp*, vol. 95, no. 1995, p. 120, 1995.

[3] O. Sandred, M. Laurson, and M. Kuuskankare, “Revisiting the illiac suite–a rule-based approach to stochastic processes,” *Sonic Ideas/Ideas Sonicas*, vol. 2, pp. 42–46, 2009.

[4] W. K. Ching and M. K. Ng, *Markov Chains: Models, Algorithms and Applications.* Springer, 2006. pg. 112.

[5] C. Ames, “The markov process as a compositional model: A survey and tutorial,” *Leonardo*, vol. 22, no. 2, pp. 175–187, 1989.

[6] C. Bell, “Algorithmic music composition using dynamic markov chains and genetic algorithms,” *Journal of Computing Sciences in Colleges*, vol. 27, no. 2, pp. 99–107, 2011.

[7] W. Dubitzky, O. Wolkenhauer, K.-H. Cho, and H. Yokota, eds., *Genetic Algorithm*, pp. 819–819. New York, NY: Springer New York, 2013.

[8] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” *CoRR*, vol. abs/1609.03499, 2016.

[9] A. L. Fradkov, “Early history of machine learning,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1385–1390, 2020. 21st IFAC World Congress.

[10] O. Bown, *Sociocultural and Design Perspectives on AI-Based Music Production: Why Do We Make Music and What Changes if AI Makes It for Us?*, pp. 1–20. Cham: Springer International Publishing, 2021.

[11] A. M. Gioti, *Artificial Intelligence for Music Composition*, pp. 53–73. Cham: Springer International Publishing, 2021.

[12] B. Eno, *A Year with Swollen Appendices.* Faber & Faber, 1996.

[13] A. Kantosalo and H. T. Toivonen, "Modes for creative human-computer collaboration: Alternating and task-divided co-creativity," in *ICCC*, 2016.

[14] R. Douc, E. Moulines, P. Priouret, and P. Soulier, *Examples of Markov Chains*, pp. 27–52. Cham: Springer International Publishing, 2018.

[15] A. van den Bosch, *Hidden Markov Models*, pp. 609–611. Boston, MA: Springer US, 2017.

[16] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, *Handbook of markov chain monte carlo.* CRC press, 2011.

[17] L. Hiller, L. Isaacson, *et al.*, "Musical composition with a high-speed digital computer," *Machine models of music*, pp. 9–21, 1993.

[18] K. Chan, C. Lenard, and T. Mills, *An Introduction to Markov Chains. Conference: The MAV 49th Annual Conference.* The Mathematical Association of Victoria, 2012.

[19] N. Privault, *Discrete-Time Markov Chains. In: Understanding Markov Chains. Springer Undergraduate Mathematics Series.* Springer, Singapore, 2018. pg. 89-113.

[20] R. P. Stanley, *Random Walks. In: Algebraic Combinatorics. Undergraduate Texts in Mathematics.* Springer, Cham, 2018.

[21] R. W. Young, "Terminology for logarithmic frequency units," *The Journal of the Acoustical Society of America*, vol. 11, no. 1, pp. 134–139, 1939.

[22] Wikipedia contributors, "Infinite monkey theorem — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/wiki/Infinite_monkey_theorem`, 2021. [Online; accessed 05-December-2021].

[23] Wikipedia contributors, "Ice ice baby — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/wiki/Ice_Ice_Baby`, 2021. [Online; accessed 05-December-2021].

[24] J. Kennedy, M. Kennedy, and T. Rutherford-Johnson, *The Oxford Dictionary of Music.* Oxford University Press, 2013.

[25] B. Bergman, "Into the grey: The unclear laws of digital sampling," *Hastings Communications and Entertainment Law Journal (Comm/Ent)*, vol. 27, p. 619, 2004-2005.

[26] MusicXML contributors, "Musicxml for exchanging digital sheet music." `https://www.musicxml.com/`, September 2021. [Online; accessed 05-December-2021].

[27] Music21 contributors, "Music21." `http://web.mit.edu/music21/`. [Online; accessed 05-December-2021].

[28] K. Bogner and U. Landrock, "Response biases in standardised surveys," *GESIS survey guidelines*, 2016.

[29] R. G. Kost and J. C. de Rosa, "Impact of survey length and compensation on validity, reliability, and sample characteristics for Ultrashort-, Short-, and Long-Research Participant Perception Surveys," *J Clin Transl Sci*, vol. 2, pp. 31–37, Feb 2018.

[30] A. Sen and M. Srivastava, *Multiple Regression*, pp. 28–59. New York, NY: Springer New York, 1990.

[31] R. Johnston, *Correlation Coefficient*, pp. 1304–1305. Dordrecht: Springer Netherlands, 2014.

[32] M. Saisana, *Analysis of Variance*, pp. 162–165. Dordrecht: Springer Netherlands, 2014.

[33] Mutable Instruments Authors, "Mutable instruments: Marbles." `https://mutable-instruments.net/modules/marbles/`. [Online; accessed 26-March-2022].

[34] Allen Synthesis, "Europi." `https://github.com/Allen-Synthesis/EuroPi`, 2022.

[35] A. Stevenson, "ambient music," 2010.

[36] B. Eno, "Ambient music," *Audio culture: Readings in modern music*, pp. 94–97, 2004.

[37] A. Van Der Merwe and W. Schulze, "Music Generation with Markov Models," *IEEE MultiMedia*, vol. 18, no. 3, pp. 78–85, 2011.

[38] G. Upton and I. Cook, *A Dictionary of Statistics*. Oxford University Press, 2008.

Appendix A

Glossary

Musical definitions

All definitions in this section are from *The Oxford Dictionary of Music* [24].

dynamics.	The variation in loudness between notes or phrases.
interval.	The difference in pitch between two sounds.
key.	A subset of pitches that form the basis of a musical composition.
melody.	A sequence of single notes arranged in a definite pattern of pitch and rhythm.
monophonic.	Music with a single melodic line.
octave.	The interval between a musical pitch and another with double its frequency.
polyphonic.	Music with two or more simultaneous melodic lines.
polyrhythm	Several different rhythms performed simultaneously.
quantisation.	The process of transforming performed musical notes, which may have some imprecision due to expressive performance, to an underlying musical representation that eliminates the imprecision.
rest.	Used to mark pauses where no notes are played in written music.
rhythm.	The placement of notes or sounds in time.
time signature.	A sign placed after the clef and at the beginning of a piece of music, or during the course of it, to indicate the time or metre of the music. Normally it comprises two numbers, one above the other, the lower defining the unit of measurement in relation to the whole-note, the upper indicating the number of those units in each measure (bar).
triplets.	Group of 3 notes, or notes and rests, equal in time-value, written where a group of 2 notes is suggested by time signature.

Probability and statistics definitions

All definitions in this section are from *A Dictionary of Statistics* [38].

alternative hypothesis. An opposing theory in relation to the null hypothesis.

categorical variable. A variable whose values are not numerical. Examples include gender (male, female), paint colour (red, white, blue), type of bird (duck, goose, owl).

continuous variable. A variable whose set of possible values is a continuous interval of real numbers.

statistical test. A method of statistical inference used to decide whether the data at hand sufficiently support a particular hypothesis.

null hypothesis. The hypothesis that there is no significant difference between specified populations, any observed difference being due to sampling or experimental error.

significance level. A measure of the strength of the evidence that must be present in your sample before you will reject the null hypothesis and conclude that the effect is statistically significant.

test statistic. A numerical summary of a data-set that reduces the data to one value that can be used to perform the hypothesis test.

p-value. The probability of obtaining a value for the test statistic that is as extreme, or more extreme (taking account of the alternative hypothesis).