Pythagoras

Using Python to find patterns in music score data

Brandon Bellanti
MLIS Candidate '20
Simmons University
Boston, MA

Introduction

Pythagoras is a digital humanities project to discover and visualize relationships between similar music works and composers through computer analysis. The variables considered for similarity include composers, eras, key signatures, time signatures, tempos, instrumentation, distributions of notes, and patterns of notes. The primary focus is finding musical patterns used in multiple works and used by multiple composers.

XML-encoded music data

The first step is obtaining music score data in a format that can be parsed by Python. MusicXML is a standardized encoding format for exchanging sheet music digitally; files are found online through the IMSLP library or Project Gutenberg, generated manually with a music notation software like Finale or Sibelius, or generated automatically from optical music recognition (OMR) software like Capella Scan (Figure 1).

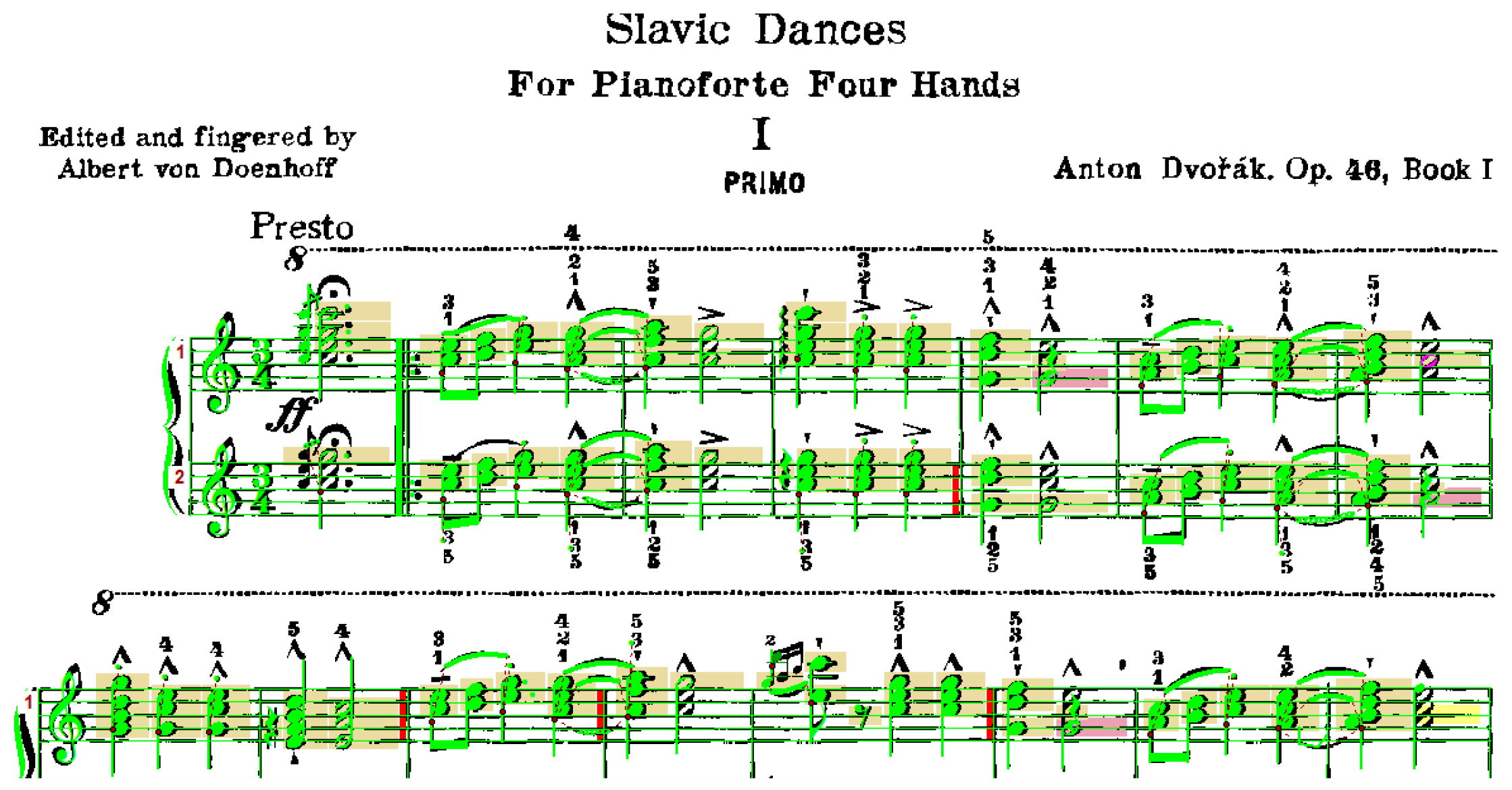
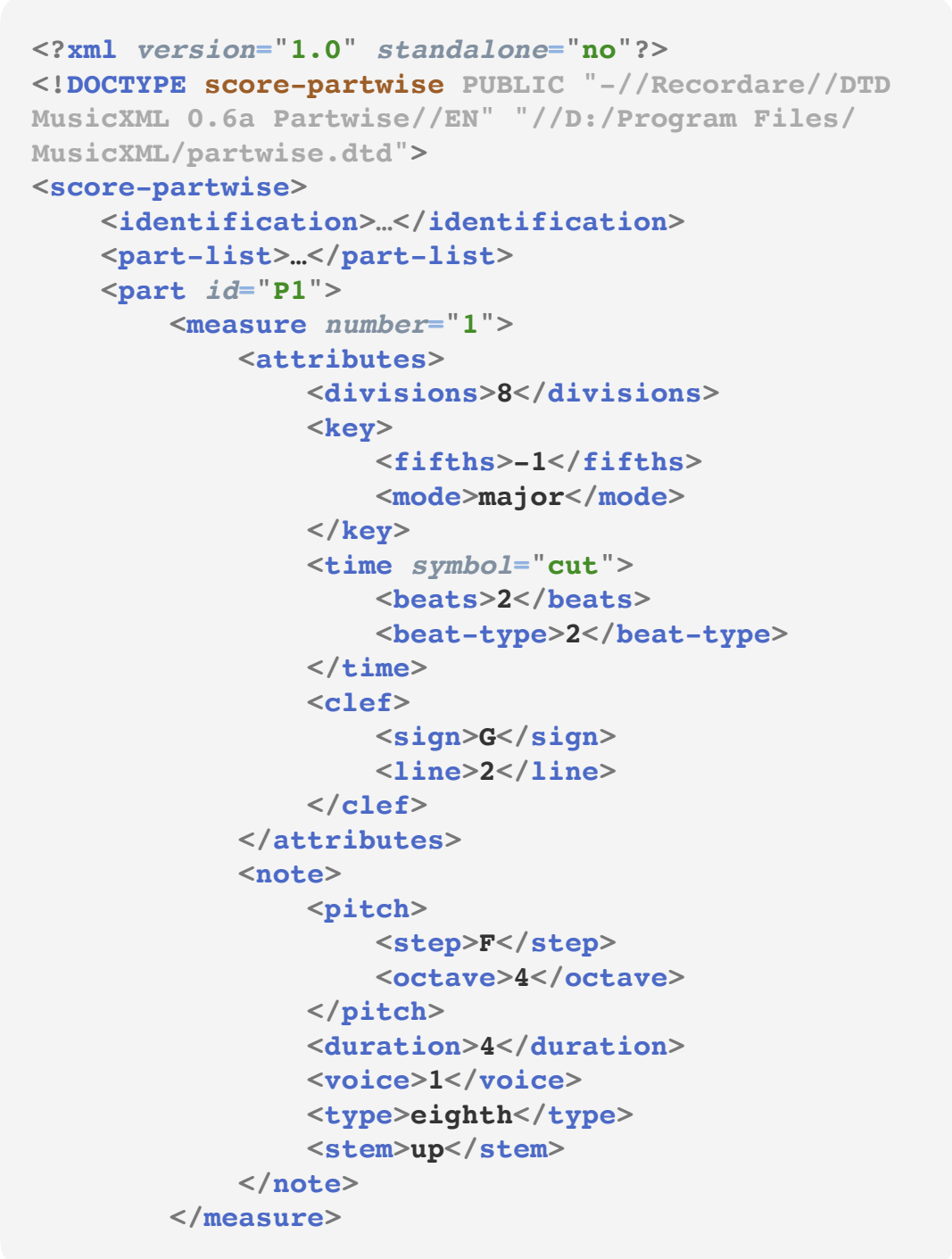

Slavic Dances
For Pianoforte Four Hands
Edited and fingered by Albert von Doenhoff
I
PRIMO
Anton Dvořák. Op. 46, Book I
Presto

Figure 1. Optical music recognition with Capella Scan

MusicXML files (Figure 2) are parsed with Python's lxml library. Information about the key and time signatures, instrumentation, and number of measures is extracted, and the notes are converted from XML to a Pandas dataframe for further analyses.



```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD
MusicXML 0.6a Partwise//EN" "//D:/Program Files/
MusicXML/partwise.dtd">
<score-partwise>
    <identification>…</identification>
    <part-list>…</part-list>
    <part id="P1">
        <measure number="1">
            <attributes>
                <divisions>8</divisions>
                <key>
                    <fifths>-1</fifths>
                    <mode>major</mode>
                </key>
                <time symbol="cut">
                    <beats>2</beats>
                    <beat-type>2</beat-type>
                </time>
                <clef>
                    <sign>G</sign>
                    <line>2</line>
                </clef>
            </attributes>
            <note>
                <pitch>
                    <step>F</step>
                    <octave>4</octave>
                </pitch>
                <duration>4</duration>
                <voice>1</voice>
                <type>eighth</type>
                <stem>up</stem>
            </note>
        </measure>
```

Figure 2. MusicXML file

Visualizations

With the note data stored in a Pandas dataframe, the frequencies of notes and parts are calculated and visualized with the NumPy, Matplotlib, and Seaborn libraries or exported to a CSV file and analyzed in Tableau (Figure 3).

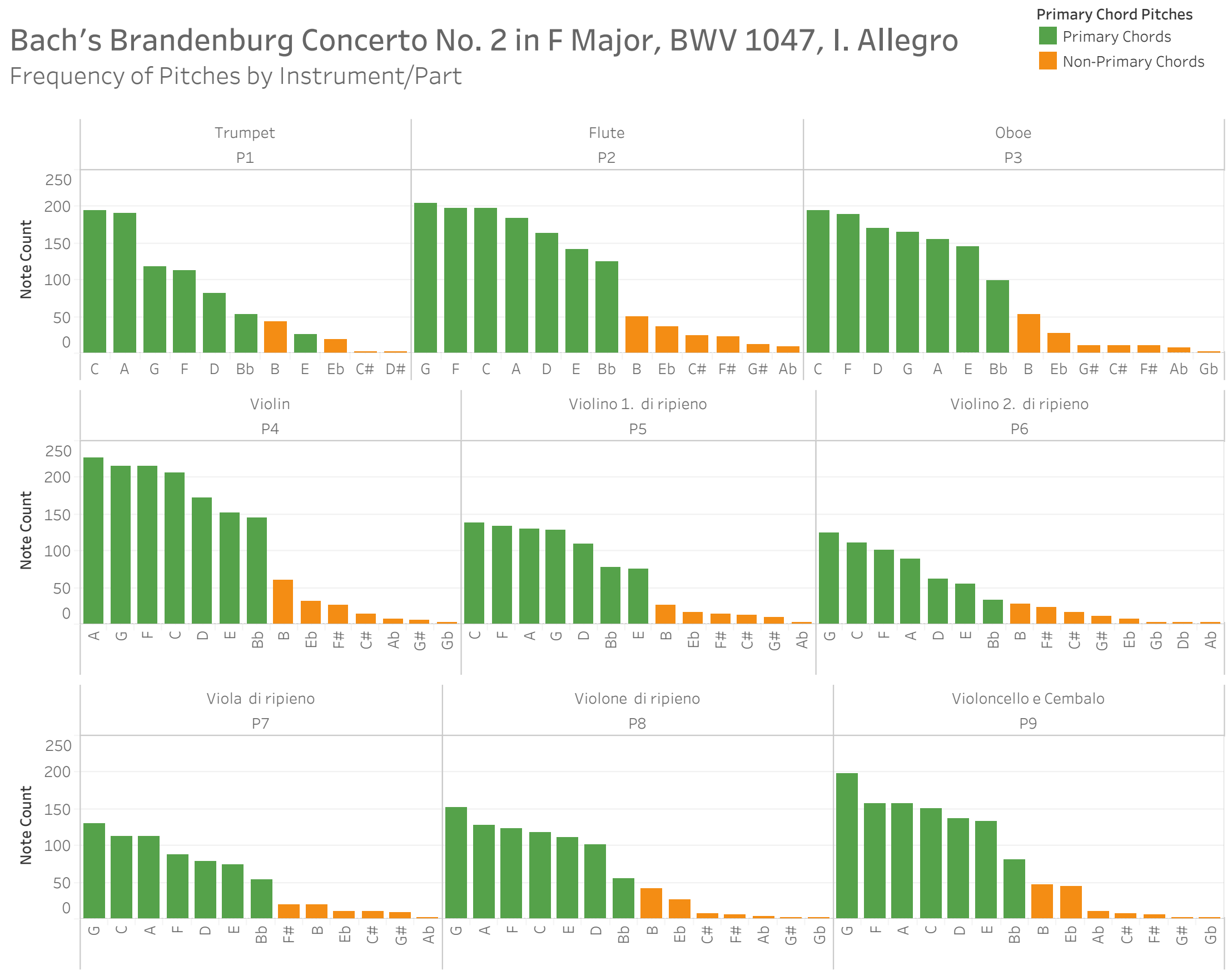


Figure 3. Visualization of note frequencies created in Tableau

Entity-relationship model and metadata

An entity-relationship model (Figure 6) outlines the major entities and attributes used to determine similarity between works and composers. This model is implemented in MySQL and partially implemented in SQLite3 for testing.

An important part of this model and the project overall is providing links to external sources for context and authority. URLs are included for VIAF, IMSLP, and Wikipedia records whenever possible. Python scripts automatically generate HTML files that include microdata for works and composers according to schema.org vocabularies.

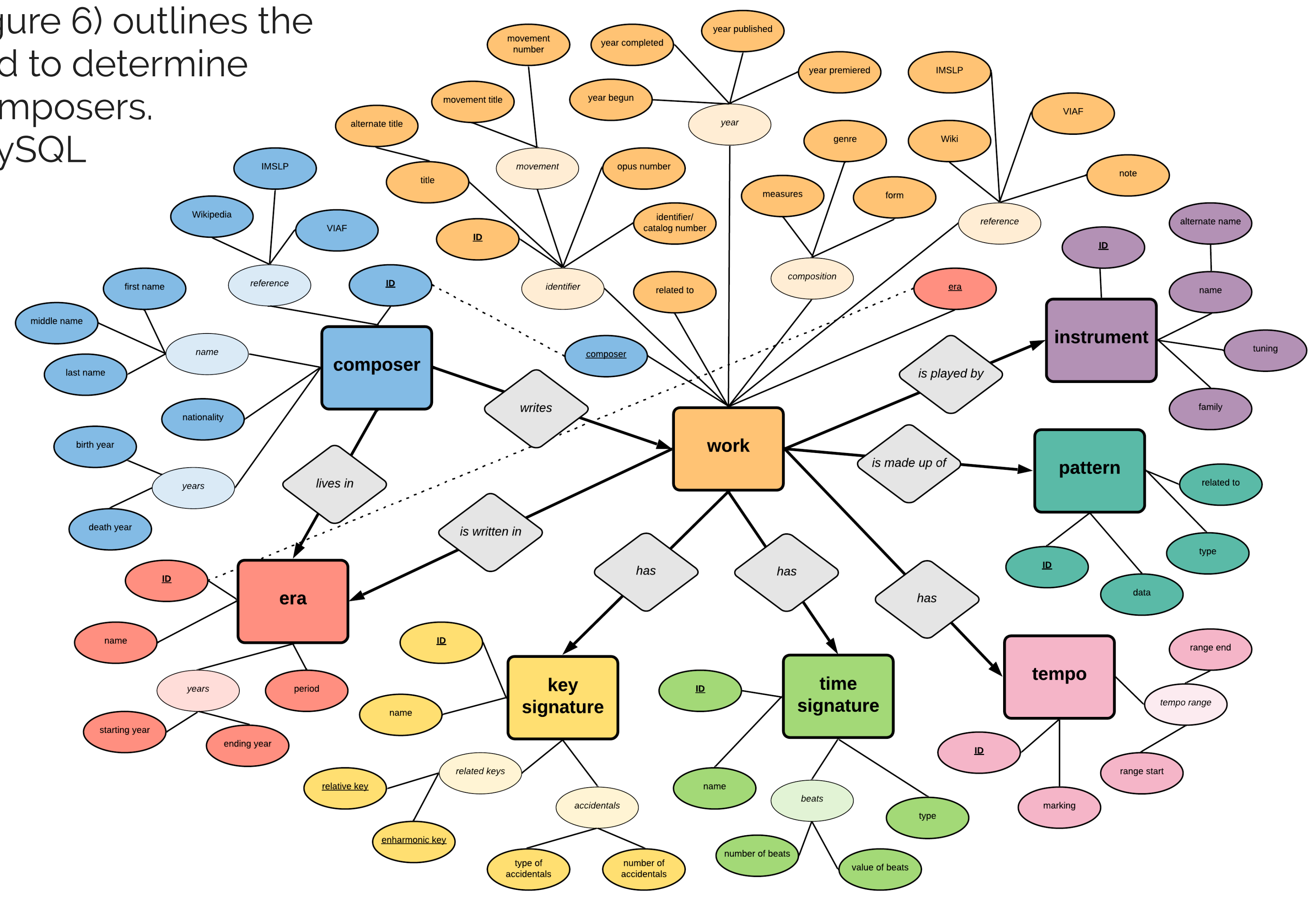


Figure 6. Entity-relationship model for Pythagoras

Pattern recognition with regular expressions

Using the *Series.to_string()* method in Pandas, a series of note pitches or durations are converted to a string, which is then searched using regular expressions. The following code (Figure 4) finds all regex matches for a pattern of a specified length that recurs a specified number of times.

```
pitches = df['pitch'].to_string()

min_length = 4
min_occur = 4

rex = r'(.{%d,})(?:.*\1){%d,}' % (min_length, min_occur-1)

patterns = re.findall(rex,pitches,overlapped=True)
```

Figure 4. Example of regular expression search for patterns

Figure 5 shows the output of the example code when given strings that represent the pitches of three well-known children's songs. Each of these melodies is based on Mozart's work "Ah! vous, dirai-je, maman," and the patterns found in this example are also returned when the MusicXML file for that work is analyzed.

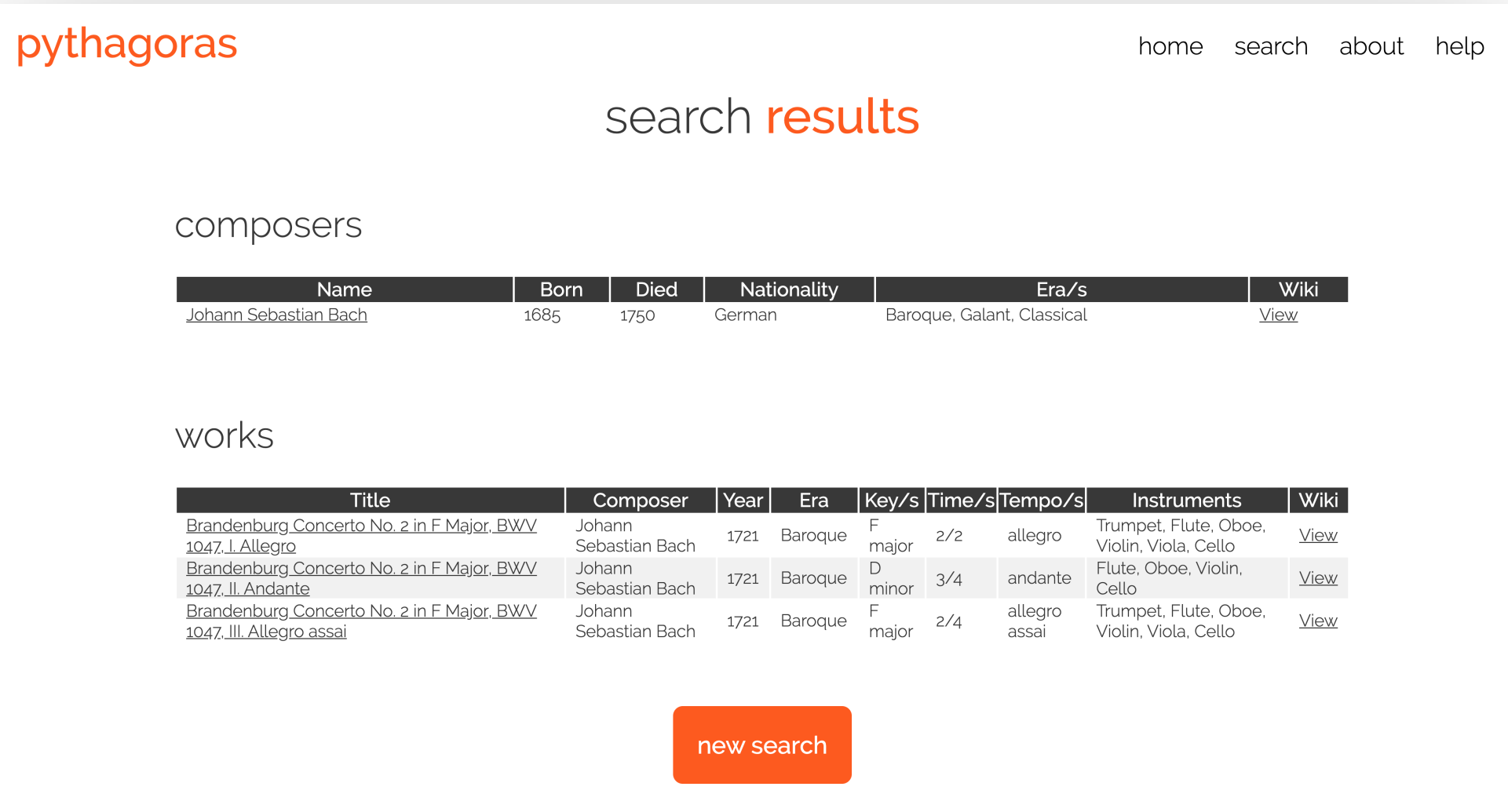
```
childrens_songs = {
  "Twinkle, Twinkle, Little Star":{
    "pitches":"CCGGAAGFFEEDDCGGFFEEDGGFFEEDCCGGAAGFFEEDDC",
    "patterns":["GFFEED","GFFEE","FFEED","GFFE","FFEE","FEED"]
  },
  "Bah, Bah, Black Sheep":{
    "pitches":"CCGGAAAAGFFEEDDCGGGFFFEEEDGGGFFFFEEEDCCGGAAAAGFFEEDDC",
    "patterns":["FFEE"]
    },
  "The ABC Song":{
    "pitches":"CCGGAAGFFEEDDDDCGGFEEDGGGFEEDCCGGAAGFFEEDDC",
    "patterns":["FEED"]
  }
}
```

Figure 5. Example output of code in Figure 4

While this type of search functions for musical patterns in the same key signature written with the same pitches, it must be adapted to make it more flexible. The MusicXML scheme includes pitch and octave elements, so each note is given a numeric value matching a corresponding piano key. The *DataFrame.diff()* method calculates the difference between note values (intervals), and the relative integers are converted to a string and searched. Pitch is only one variable out of many that form musical patterns, but it is helpful here in demonstrating methodology.

Web interface

The ultimate aim of Pythagoras is to create a web interface (Figure 7) that will allow users to search the database for a composer or work, view basic visualizations, and explore related composers and works based on similarity.



search results

composers

Name	Born	Died	Nationality	Era/s	Wiki
Johann Sebastian Bach	1685	1750	German	Baroque, Galant, Classical	View

works

Title	Composer	Year	Era	Key/s	Time/s	Tempo/s	Instruments	Wiki
Brandenburg Concerto No. 2 in F Major, BWV 1047, I. Allegro	Johann Sebastian Bach	1721	Baroque	F major	2/2	allegro	Trumpet, Flute, Oboe, Violin, Viola, Cello	View
Brandenburg Concerto No. 2 in F Major, BWV 1047, II. Andante	Johann Sebastian Bach	1721	Baroque	D minor	3/4	andante	Flute, Oboe, Violin, Cello	View
Brandenburg Concerto No. 2 in F Major, BWV 1047, III. Allegro assai	Johann Sebastian Bach	1721	Baroque	F major	2/4	allegro assai	Trumpet, Flute, Oboe, Violin, Viola, Cello	View

Figure 7. Search result page for the web interface

SCAN ME

Scan the code to view the prototype (best on desktop) and stay up to date with the project!