

Detailed Notes for Issues

[Adding Duration to Note Lists](#)

[Add Measure + Beat References to Notes and nGrams](#)

[Build EMA Reference](#)

[Durational Ratios in nGrams](#)

[Classifier: Add Z Scores and Hidden PEN/ID Finder](#)

[Finding Similar Intervals Across Corpus](#)

Adding Duration to Note Lists

For Lists of individual notes, we currently need to get_Notes, then get_Durations, and concatenate them to get one table. Could we create tuples of Notes + Durations?

- [Gould code](#) to get Duration for individual note
- [Morgan code](#) for getDurations

Add Measure + Beat References to Notes and nGrams

For Notes and nGrams, can we calculate the Measure/Beat references and add them to output DFs?

Build EMA Reference

Gould's Interval system builds the EMA references for any 'matched' soggetto (which is basically an ngram of intervals).

Could we build EMA for Morgan nGrams of various kinds?

Could we build EMA for Classified Results (which are collections of Gould nGrams that make a PEn, ID, etc? Note that at some point we might want to Classify from Morgan intervals, too.

- [Gould code](#) to build EMA for one nGram

Durational Ratios in nGrams

Another feature we added to Gould intervals was the capacity to make ‘rhythmic vectors’ alongside the melodic ones. We did this by taking the ratio between successive durations. So half note followed by whole note would be 0.5. Whole followed by half note would be 2.0.

Could we add this to Morgan nGrams?

- [See Cell 39 of this Notebook](#) for the code that Richard and Daniel made for this.

Classifier: Add Z Scores and Hidden PEN/ID Finder

The [Freedman-Gould Classifier](#) does a pretty good job of:

- **Finding exact or closely matched soggetti** (ngrams) in a piece
- It then assembles these with **GroupBy** into nearby sets, via various measurements:
 - Measures **“this offset”** and **“next offset”** of the same soggetto
 - It looks for instances that happen at the **same time**--these are what we call **‘parallel’** entries, and are interesting but don’t count in the classified pattern
 - If there is a **long gap before**, or a **long gap after** a given instance, then it **breaks the group into two (or more) subgroups**, then runs the classifier.
 - It finds:
 - singletons (entries all on their own, with long before/after gaps)
 - PENS, IDS, and Fugas
 - Cannot yet find NIMs, but perhaps we will be able to do this with Morgan’s system.
- The Notebook then saves the results as CSV, and includes information about:
 - Piece
 - Soggetto (ngram), with columns for the reference pattern and the one it actually finds (since it’s cross-matching things that are ‘close’)
 - Voice in which the instance appears
 - Measure/Beat
 - Starting Offset for each ngram
 - Total Duration of each ngram
 - See examples of output

- [Model 0017](#)
- [Mass 0015_1](#)

However! I would like to add the following features:

- **use Z scores** to remove instances of a given ngram soggetto from the final groupby objects--just before classification. So if the total durations of four soggetti are (for instance): (10, 10, 7, 10) we would discard the "7".
 - Read [here](#).
- **use "combinations"** on these same final sub-groups in order to find "hidden" PEnS and IDS.
 - In Model 0017, see green highlights at rows 79, 96, and 98; in these instances there are in fact PEnS "hidden" within what the Classifier calls long Fugas. Could we 'find' them?
 - In Mass_0015_1, see green highlight at row 37, which is a _very_ long fuga but contains several PEnS
 - Pseudo code:
 - from itertools import *
 - Get the offsets as a list, which would be something like this:
sample_offsets = [0, 2, 4, 5, 7, 9]
 - x = len(sample_offsets)-1
 - a = list(combinations(sample_offsets, x))
 - print(a):
 - [(0, 2, 4, 5, 7),
 - (0, 2, 4, 5, 9),
 - (0, 2, 4, 7, 9),
 - (0, 2, 5, 7, 9),
 - (0, 4, 5, 7, 9),
 - (2, 4, 5, 7, 9)]
 - Check them for PEn, ID using something like Cell 13 of [this](#).
 - Report Sub-sub group as appropriate

Finding Similar Intervals Across Corpus

There are three ways to do this currently:

- The **Gould Intervals System** can search in any number of pieces at once, and can report **Melodic Matches** with adjustable thresholds for the following:

- **Diatonic** (which Gould calls Generic) or **Chromatic** (which he calls Semitone)
 - Note that these are “1” based rather than “0” based, which is possible with Morgan’s system
- The **Length of the ngrams** for the matched soggetti (which he calls vectors)
- The **Minimum Number of Matches** to report
- Whether the matches are **Exact** copies, or **Close**
- If Close, then **how close** (measured in side-by-side differences of the intervals)
- See this [Notebook](#) (for the latest version of the Classifier by Freedman see [here](#))
- **Freedman and Russo-Batterham** also added **Durational Vectors** to this system
 - Creates a list of the **ratios between successive durations**
 - Checks the **differences between these ratios**
 - **Groups** the soggetti according to the melodic match AND any threshold of **differences between the durational ratios**
 - See this [Notebook](#)
- **Gould/Freedman/Russo-Batterham Classifier**
 - Can produce a series of ‘classified’ outputs for any number of pieces (with options to set variables as noted above. See [this NB](#))
 - Merge the results of the previous to a single file, to allow comparison across works. [See this NB](#)
- **Morgan Intervals with Corpus**
 - [This Colab Notebook](#) will produce a series of outputs (with any Morgan method) for any corpus of files.
 - The results will be separate DFs, but they could be concatenated or combined for subsequent analysis.
 - Note that for some reason this Colab version does not load the latest version of Morgan. Because of problem with Pip? But it works.