Introducing the *Digital Musicology Corpus Analysis Toolkit* for processing and analyzing notated music on a very large scale

Johannes Hentschel[1], Yannis Rammos[1], Andrew McLeod[2], Martin Rohrmeier[1]

✉ johannes.hentschel@epfl.ch

[1] Digital and Cognitive Musicology Lab, École Polytechnique Fédérale de Lausanne, Switzerland [2] Fraunhofer IDMT, Ilmenau, Germany

Swiss National Science Foundation

Data Pre-processing

`DiMCAT` is able to analyze heterogeneous symbolic music corpora by converting them into five dataframes, one per **Facet**:

events: notes and rests (incl. ties, tremolos, grace notes, etc.);
control events: performance details (e.g., dynamics, lyrics, articulation);
structure: time line and flow control (e.g., time signatures, repeat signs);
annotations: analytical labels (e.g., chord labels, algorithmic outputs);
metadata: piece information (e.g., creation date, composer, score origin).

Frictionless Serialization

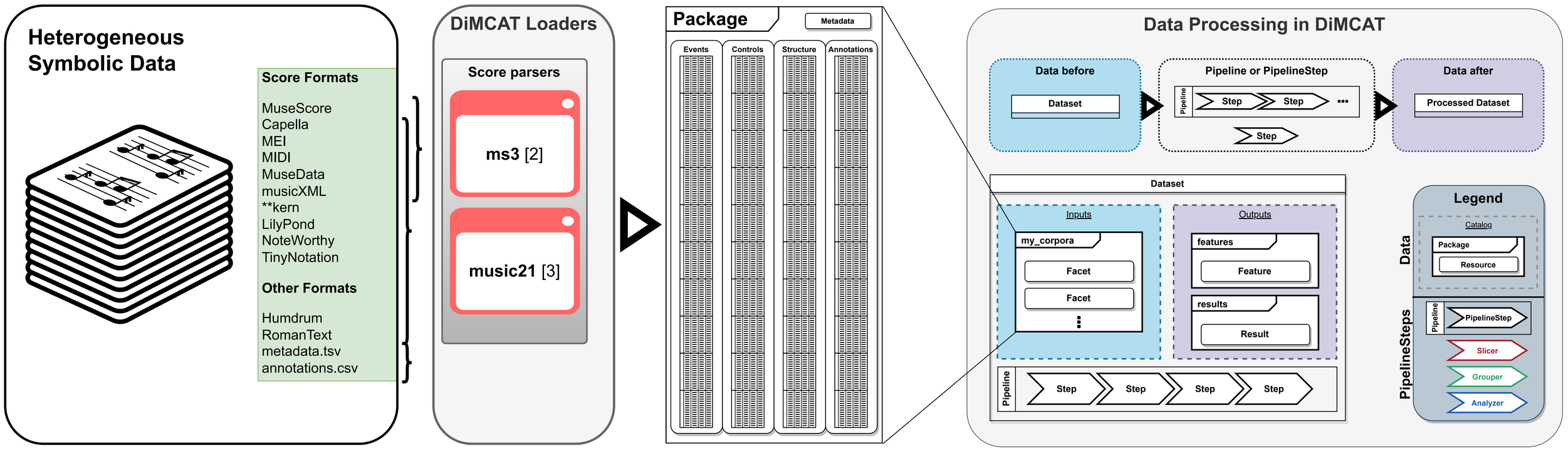
All `DiMCAT` objects can be serialized to **JSON**. For all data objects, this is achieved using the `Frictionless` standard [1].

- Each dataframe in `DiMCAT` is called a **Resource** and stored as *tab-separated values (TSV)* together with a JSON descriptor.
- Multiple resources are stored together as a **Package**, that is, a JSON descriptor and a ZIP file containing compressed TSVs.
- The JSON descriptors fully specify the column schemas. This makes the **Resources** interoperable and allows for lazy loading.

Basic Principle

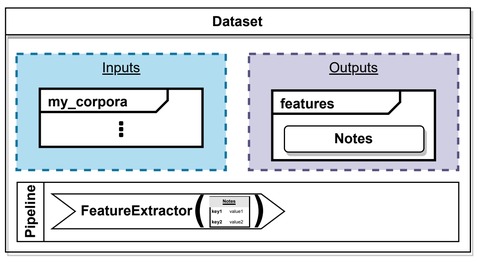
Data objects are processed by **PipelineStep** objects, yielding new, processed Data objects.

- The **DiMCAT** workflow starts with initializing a **Dataset** object:
  - either an empty one, loading a Package into the `Inputs` **Catalog**;
  - or one that had been serialized together with its `Outputs`.
- Having copied a given **Dataset** object, a **PipelineStep** may
  - add **Feature** or **Result** resources to the `Outputs`;
  - perform transformations on **Resources** added by previous Steps.



Extracting and Visualizing Features

`DiMCAT` extracts **Feature** resources in declarative fashion. A **PipelineStep** or a user specify the required feature and the **Dataset** object extracts it or returns a previously cached resource. In the remaining examples, the corresponding **FeatureExtractor** steps are not explicitly shown in the pipelines.



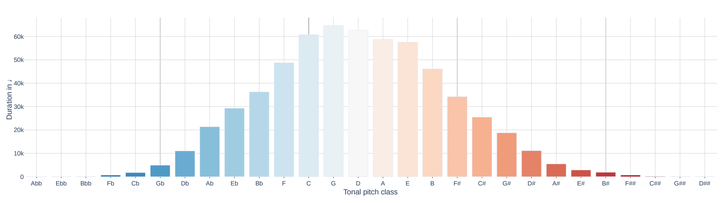
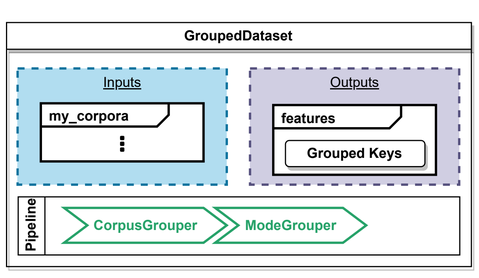


Fig. 1: Pitch-class distribution over *my_corpora* in quarter notes, generated by a plotting method of the extracted **Notes** feature.

Grouping

Applying a **Grouper** to a dataset is tantamount to binning chunks of data based on a membership criterion. Depending on the type of **Grouper**, it may transform all **Resource** objects in the `Outputs` (e.g., the **CorpusGrouper**) or request a specific **Feature** (e.g., the **ModeGrouper**).



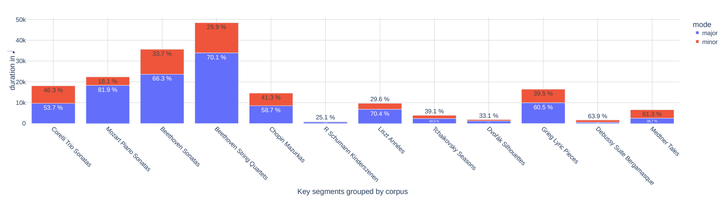
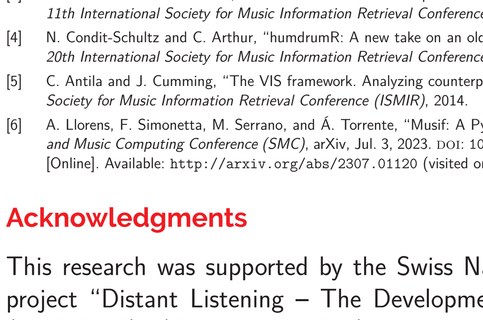


Fig. 3: Bar chart showing fractions of corpus duration with local keys in major (blue) vs. minor (red), produced by plotting **Keys** after applying two Groupers.

Slicing

Slicer objects segment **Resources** according to the boundaries gained from one particular **Feature**. In a `SlicedDataset`, these segments, rather than entire pieces, serve as the basic unit that **Analyzer** objects operate on. Slicing a **Feature** based on another one may enable the use of additional **Groupers**, for instance, when **Notes** are sliced based on **HarmonyLabels**.



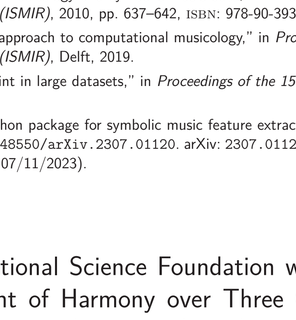
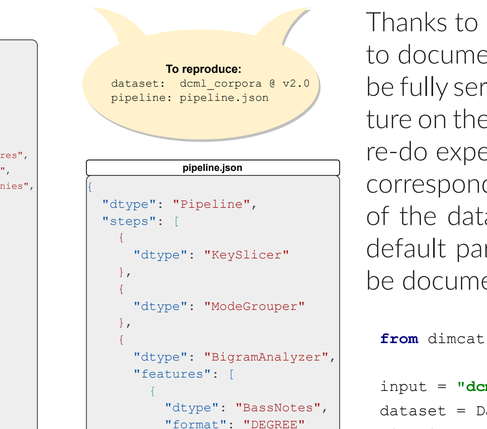


Fig. 3: This truncated heatmap shows the relative frequencies (in %) of transitions between the ten most frequent local keys in *my_corpora*. Keys are given as Roman numerals pertaining to a major scale. Percentages following y-axis labels correspond to the proportion of key segments in the given key. Black bars represent the relative entropy of each (complete) row.

Analyzing & Transforming



Analyzer objects perform an analysis or transformation on one or several Features and add the resulting **Resource** (**Feature** or **Result**) to the `Outputs`. Analyzers my depend on prior application of other **PipelineSteps**.

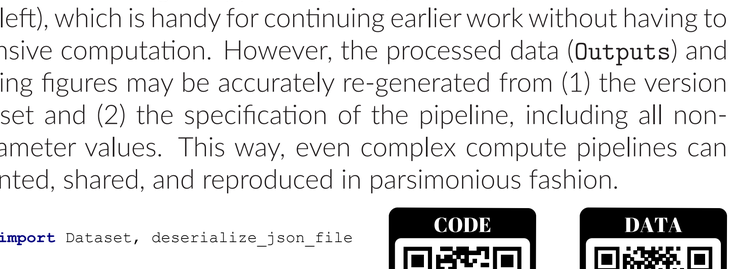


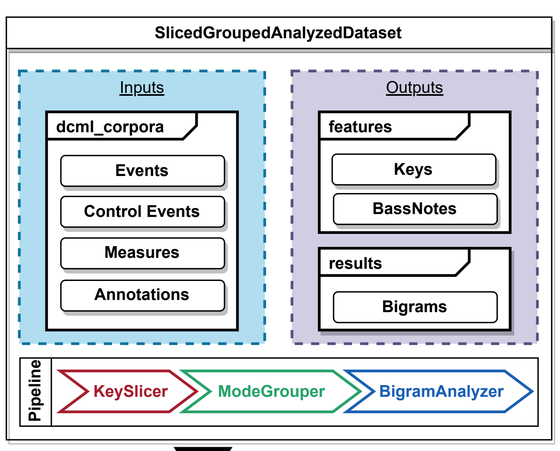
Fig. 4: Sankey plot displaying the transition masses between chordal roots of all 4-grams terminating on a Perfect Authentic Cadence.

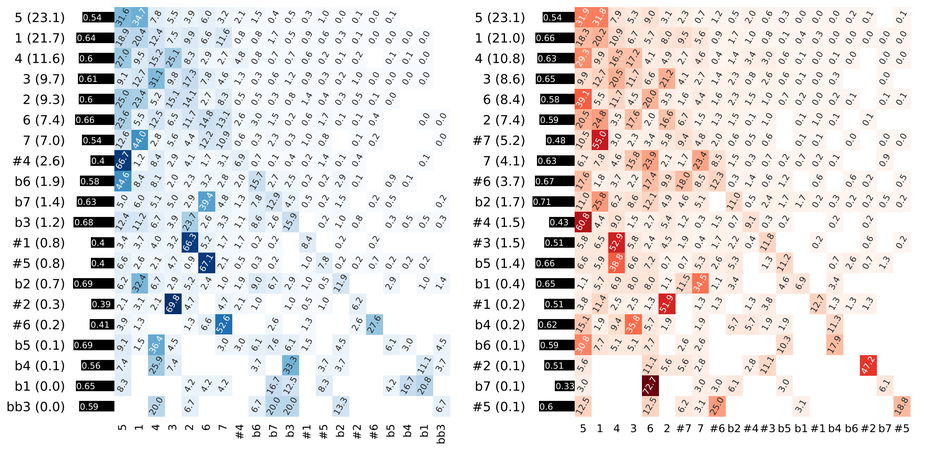
Statement of Need

As corpora of digital musical scores continue to grow, we introduce the *Digital Musicology Corpus Analysis Toolkit* (`DiMCAT`), a **Python library** for processing large corpora of digitally encoded **scores and annotations**. Equally aimed at music-analytical **corpus studies**, MIR, and machine-learning research, the library performs common **data transformations and analyses** using dataframes. In comparison to other libraries that achieve similar goals [3–6], `DiMCAT` specializes in flexibly scaling the operational level (whole datasets, pieces, segments) and offering appropriate visualizations. The **pipeline API design** enables the flexible combination of commonplace operations (slicing, grouping, analyzing) and facilitates the **communication of reproducible research results**. It prioritizes **computational speed**, **extensibility**, and **ease of use**, thus aiming to cater to machine-learning practitioners and musicologists alike.

The Reproducible Slice-Group-Analyze Pipeline

Fig. 5: Transitions between the 20 most frequent chordal bass notes (as scale degrees) for all local key segments in major (left, blue) and in minor (right, red).





Thanks to its comprehensive serialization capabilities, `DiMCAT` makes it easy to document and communicate **reproducible results**. A **Dataset** object can be fully serialized including its zipped outputs (see the simplified JSON structure on the left), which is handy for continuing earlier work without having to re-do expensive computation. However, the processed data (`Outputs`) and corresponding figures may be accurately re-generated from (1) the version of the dataset and (2) the specification of the pipeline, including all non-default parameter values. This way, even complex compute pipelines can be documented, shared, and reproduced in parsimonious fashion.

```python
from dimcat import Dataset, deserialize_json_file

input = "dcml_corpora.datapackage.json"
dataset = Dataset.from_package(input)
pipeline = deserialize_json_file("pipeline.json")
processed_dataset = pipeline.process(dataset)
bigrams = processed_dataset.get_result()
bigrams.plot()
```

CODE DATA

References

[1] D. Fowler, J. Barratt, and P. Walsh, "Frictionless data: Making research data quality visible," *International Journal of Digital Curation*, vol. 12, no. 2, pp. 274–285, May 13, 2018, ISSN: 1746-8256. DOI: 10.2218/ijdc.v12i2.577. [Online]. Available: http://www.ijdc.net/article/view/577 (visited on 07/10/2023).

[2] J. Hentschel and M. Rohrmeier, "ms3: A parser for MuseScore files, serving as data factory for annotated music corpora," *Journal of Open Source Software*, vol. 8, no. 88, p. 5195, 2023. DOI: 10.21105/joss.05195.

[3] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data," in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, 2010, pp. 637–642, ISBN: 978-90-393-5381-3.

[4] N. Condit-Schultz and C. Arthur, "humdrumR: A new take on an old approach to computational musicology," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, 2019.

[5] C. Antila and J. Cumming, "The VIS framework. Analyzing counterpoint in large datasets," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[6] A. Llorens, F. Simonetta, M. Serrano, and Á. Torrente, "Musif: A Python package for symbolic music feature extraction," in *Sound and Music Computing Conference (SMC)*, arXiv, Jul. 3, 2023. DOI: 10.48550/arXiv.2307.01120. arXiv: 2307.01120 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2307.01120 (visited on 07/11/2023).

Acknowledgments

This research was supported by the Swiss National Science Foundation within the project "Distant Listening – The Development of Harmony over Three Centuries (1700–2000)" (Grant no. 182811). This project was being conducted at the Latour Chair in Digital and Cognitive Musicology, generously funded by Mr. Claude Latour.