The Journal of Open Source Software

repytah: An Open-Source Python Package for Building Aligned Hierarchies for Sequential Data

Chenhui Jia[1], Lizette Carpenter[1], Thu Tran[1], Amanda Y. Liu[1], Sasha Yeutseyeva[1], Marium Tapal[1], Yingke Wang[2], Zoie Kexin Zhao[1], Jordan Moody[1], Denise Nava[1], Eleanor Donaher[1], Lillian Yushu Jiang[1], Ben Bruncati[1], and Katherine M. Kinnaird[1¶]

1 Smith College, USA **2** Columbia University, USA ¶ Corresponding author

DOI: 10.21105/joss.05213

Software

- Review
- Repository
- Archive

Editor: Mehmet Hakan Satman

Reviewers:

- @Rocsg
- @ranzhengcode

Submitted: 17 February 2023
Published: 13 May 2023

Summary

We introduce repytah, a Python package that constructs the aligned hierarchies representation that contains all possible structure-based hierarchical decompositions for a finite length piece of sequential data aligned on a common time axis. In particular, this representation–introduced by Kinnaird (2016) with music-based data (like musical recordings or scores) as the primary motivation–is intended for sequential data where repetitions have particular meaning (such as a verse, chorus, motif, or theme). Although the original motivation for the aligned hierarchies representation was finding structure for music-based data streams, there is nothing inherent in the construction of these representations that limits repytah to only being used on sequential data that is music-based.

The repytah package builds these aligned hierarchies by first extracting repeated structures (of all meaningful lengths) from the self-dissimilarity matrix (SDM) for a piece of sequential data. Intentionally repytah uses the SDM as the starting point for constructing the aligned hierarchies, as an SDM cannot be reversed-engineered back to the original signal and allows for researchers to collaborate with signals that are protected either by copyright or under privacy considerations. This package is a Python translation of the original MATLAB code by Kinnaird (2014) with additional documentation, and the code has been updated to leverage efficiencies in Python.

Statement of Need

Broadly, Music Information Retrieval (MIR) seeks to capture information about music in pursuit of various music-related tasks, such as playlist recommendation, cover song identification, and beat detection. Content-based methods work directly on musical artifacts such as audio recordings or musical scores, while other approaches leverage information about musical artifacts such as metadata (like the artist or album name), tags (like genre), or listener surveys. Approaches to MIR tasks can also include a focus on repeated (or novel) elements.

Music-based sequenced data, like scores and recordings, often have repeated elements that build on each other, creating structure-based hierarchies. The aligned hierarchies representation by Kinnaird (2016) is a structure-based representation that places all possible structural-hierarchical decompositions of a data stream onto one common time axis. These aligned hierarchies when applied to music-based data sequences can be used for visualization or for similarity tasks within MIR like the fingerprint task (that seeks to find all copies of a query song that are the same recording of the same piece performed by the same musical group) (Kinnaird, 2016). The aligned hierarchies can also be post-processed to address additional

tasks such as the cover song task (that seeks to find different recordings of the same piece of music either by the original artist or another one) (Kinnaird, 2018; McGuirl et al., 2018; Savard et al., 2020). A drawback of Kinnaird's code, however, is that the original code was written in MATLAB, which can only be used with a license and hence is not broadly accessible.

There has been a recent trend of creating Python packages for MIR research. Examples include the `AMEN` package (AMEN, 2016), the `mir_eval` library (Raffel et al., 2014) , the `mirdata` library (Bittner et al., 2019), and the more recent `libfmp` package (Müller & Zalkow, 2021). As MIR has grown as a discipline, there has been an increased focus on reproducibility, accessibility, and open-source development (Bittner et al., 2019; McFee et al., 2019). As such, there have been several examples of code being translated from MATLAB to Python. The most notable example is `librosa`, a package that provides a number of powerful tools for MIR work (McFee et al., 2015).

Following suit, the presented package `repytah` forms the aligned hierarchies for a given pieces of sequential data, giving MIR users broader access to these tools through the open-source Python language. Similar to the original code, `repytah` extracts structural repetitions within a data stream and leverages their relationships to each other to build the aligned hierarchies representation. In addition to translating the code from MATLAB by cross-referencing the desired output of the package with the outputs from the original code on several examples, `repytah` improves on the original code, streamlining various computations and further modularizing functions. Additionally, this package provides more complete documentation, examples, and test files than the original code.

Functionality

There are four modules in the `repytah` Python package that work together to form the aligned hierarchies: `search`, `assemble`, `transform`, and `utilities`:

- The `search` module finds and records information about repeated structures, represented as diagonals in a song's thresholded self-dissimilarity matrix.
- Once found, these repeated structures are later transformed and assembled into the aligned hierarchies using the `assemble` module, which finds the essential structure components from the repeated structures found with the `search` module, and then uses those essential structure components to build the aligned hierarchies.
- Throughout the process of building the aligned hierarchies, the found structure can be represented as either lists or as matrices. Functions in the `transform` module transform inputs between the different types, either from lists of indices into matrices or vice versa.
- Lastly, the `utilities` module contains functions that are frequently called by functions in the other three modules.

Each module has an associated Jupyter notebook file that summarizes the module's functions. There are also test files for each module.

Additionally, the package includes `example.py` which runs a complete example building aligned hierarchies from a single example CSV file for the score of Chopin's Mazurka Op. 6, No. 1. This example CSV file stores the musical information about each beat of the Mazurka as Chroma Features (or the twelve Western tones: {C, C#, D, D#, etc}). These Chroma features are extracted from the associated files on the kern Scores data base (Sapp, 2005).

In `example.py`, the first steps include transforming the sequential data into the SDM by first shingling the features (M. Casey & Slaney, 2006a, 2006b, 2007) and then using the cosine-dissimilarity measure on the resulting shingles. Again, we note that while `example.py` does compute the SDM, the `repytah` package's main contribution is the construction of the aligned hierarchies from a given SDM.

Acknowledgements

repytah has been partially funded by Smith College's Summer Undergraduate Research Fellowship (SURF) in 2019 - 2022 and by Smith College's CFCD funding mechanism. Additionally, as Kinnaird is the Clare Boothe Luce Assistant Professor of Computer Science and Statistical & Data Sciences at Smith College, this work has also been partially supported by Henry Luce Foundation's Clare Boothe Luce Program. Additionally, as repytah was developed in the TInKER lab (Katherine M. Kinnaird, Founder and PI), we would like to acknowlege other members of the TInKER lab, including Tasha Adler, for being part of our lab and for listening to presentations about earlier versions of this work.

Finally, we would like to acknowledge and give thanks to Brian McFee and the librosa team. We significantly referenced the Python package librosa in our development process.

References

AMEN, dev team. (2016). Amen: A toolbox for algorithmic remixing, after echo nest remix. In *GitHub repository*. GitHub. https://github.com/algorithmic-music-exploration/amen

Bittner, R. M., Fuentes, M., Rubinstein, D., Jansson, A., Choi, K., & Kell, T. (2019). Mirdata: Software for reproducible usage of datasets. *In Proceedings of the 20th International Society for Music Information Retrieval (ISMIR) Conference*.

Kinnaird, K. M. (2014). Thesis code. In *GitHub repository*. GitHub. https://github.com/kmkinnaird/ThesisCode

Kinnaird, K. M. (2016). Aligned hierarchies: A multi-scale structure-based representation for music-based data streams. *In Proceedings of the 17th International Society for Music Information Retrieval (ISMIR) Conference*.

Kinnaird, K. M. (2018). Aligned sub-hierarchies: A structure-based approach to the cover song task. *In Proceedings of the 19th International Society for Music Information Retrieval (ISMIR) Conference*.

M. Casey, & Slaney, M. (2006a). Song intersection by approximate nearest neighbor search. *Ismir07*, 144–149.

M. Casey, & Slaney, M. (2006b). The importance of sequences in music similarity. *Assp6*, V-5 - V-8. https://doi.org/10.1109/icassp.2006.1661198

M. Casey, & Slaney, M. (2007). Fast recognition of remixed audio. *Assp7*, IV-1425 - IV-1428.

McFee, B., Kim, J. W., Cartwright, M., Salamon, J., Bittner, R. M., & Bello, J. P. (2019). Open source practices for music signal processing research. *IEEE Signal Processing Magazine*, *36*(1), 128–137. https://doi.org/10.1109/MSP.2018.2875349

McFee, B., Raffel, C., Liang, D., Ellis, D. P. W., McVicar, M., Battenberg, E., & Nieto, O. (2015). Librosa: Audio and music signal analysis in Python. In K. Huff & J. Bergstra (Eds.), *In proceedings of the 14th Python in science conference* (pp. 18–24). https://doi.org/10.25080/Majora-7b98e3ed-003

McGuirl, M. R., Kinnaird, K. M., Savard, C., & Bugbee, E. H. (2018). SE and SNL diagrams: Flexible data structures for MIR. *In Proceedings of the 19th International Society for Music Information Retrieval (ISMIR) Conference*.

Müller, M., & Zalkow, F. (2021). Libfmp: A Python package for fundamentals of music processing. *The Journal of Open Source Software*, *6*(63), 3326. https://doi.org/10.21105/joss.03326

Raffel, C., McFee, B., Humphrey, E. J., Salamon, J., Nieto, O., Liang, D., & Ellis, D. P. W. (2014). Mir_eval: A transparent implementation of common MIR metrics. *In Proceedings of the 15th International Society for Music Information Retrieval (ISMIR) Conference*.

Sapp, C. S. (2005). Online database of scores in the humdrum file format. *Ismir06*, 664–665.

Savard, C., Bugbee, E. H., McGuirl, M. R., & Kinnaird, K. M. (2020). SuPP & MaPP: Adaptable structure-based representations for MIR tasks. *In Proceedings of the 21th International Society for Music Information Retrieval (ISMIR) Conference*.