The Journal of Open Source Software

DREiMac: Dimensionality Reduction with Eilenberg-MacLane Coordinates

Jose A. Perea[1], **Luis Scoccola**[2], **and Christopher J. Tralie**[3]

1 Northeastern University **2** University of Oxford **3** Ursinus College

DOI: 10.21105/joss.05791

Software

- Review
- Repository
- Archive

Editor: Fabian Scheipl

Reviewers:

- @mtsch
- @raphaelreinauer

Submitted: 28 July 2023
Published: 17 November 2023

Summary

DREiMac is a library for topological data coordinatization, visualization, and dimensionality reduction. Currently, DREiMac is able to find topology-preserving representations of point clouds taking values in the circle, in higher dimensional tori, in the real and complex projective spaces, and in lens spaces.

In a few words, DREiMac takes as input a point cloud together with a topological feature of the point cloud (in the form of a persistent cohomology class), and returns a map from the point cloud to a well-understood topological space (a circle, a product of circles, a projective space, or a lens space), which preserves the given topological feature in a precise sense.

DREiMac is based on persistent cohomology (de Silva et al., 2011a), a method from topological data analysis; the theory behind DREiMac is developed in Perea (2020), Perea (2018), Polanco & Perea (2019), and Scoccola et al. (2023). DREiMac is implemented in Python, using Numba (Lam et al., 2015) for the more expensive computations, and Ripser (Bauer, 2021) and its Python wrapper (Tralie et al., 2018) for persistent homology computations. We test DREiMac periodically in Ubuntu, macOS, and Windows.

The documentation for DREiMac can be found here.

Statement of need and main contributions

Topological coordinatization is witnessing increased application in domains such as neuroscience (Gardner et al., 2022; Kang et al., 2021; Rybakken et al., 2019), dynamical systems (Vejdemo-Johansson et al., 2015), and dimensionality reduction (Scoccola & Perea, 2023). The fast implementations and data science integrations provided in DREiMac are aimed at enabling other domain scientists in their pursuits.

To the best of our knowledge, the only publicly available software packages implementing cohomological coordinates based on persistent cohomology are Dionysus (Morozov, 2012) and Ripserer (Čufar, 2020). Dionysus and Ripserer are general purpose libraries for persistent homology, which in particular implement the original circular coordinates algorithm of de Silva et al. (2011b), and the sparse circular coordinates algorithm of Perea (2020), respectively.

DREiMac adds to the current landscape of cohomological coordinates software by implementing various currently missing functionalities; we elaborate on these below. DREiMac also includes functions for generating topologically interesting datasets for testing, various geometrical utilities including functions for manipulating the coordinates returned by the algorithms, and several example notebooks including notebooks illustrating the effect of each of the main parameters of the algorithms.

Sparse algorithms. All of DREiMac's coordinates are sparse, meaning that persistent cohomology computations are carried on a simplicial complex built on a small sample of the full point cloud. This gives a significant speedup when compared to algorithms which use a simplicial complex built on the entire dataset, since the persistent cohomology computation is the most computationally intensive part of the algorithm. For a precise description of the notion of sparseness, see the papers that develop the algorithms that DREiMac implements (Perea, 2018, 2020; Polanco & Perea, 2019; Scoccola et al., 2023).

Improvements to the circular coordinates algorithm. DREiMac implements two new functionalities addressing two issues that can arise when computing circular coordinates for data.

The circular coordinates algorithm turns a cohomology class with coefficients in $\mathbb{Z}$ into a map into the circle. However, since persistent cohomology is computed with coefficients in a field, the cohomology class is obtained by lifting a cohomology class with coefficients in $\mathbb{Z}/q\mathbb{Z}$, with q a prime. This lift can fail to be a cocycle, resulting in discontinuous coordinates, which are arguably not meaningful; see Figure 1 (right). An algebraic procedure for fixing this issue is described in de Silva et al. (2011b), but has thus far not been implemented. DREiMac implements this using integer linear programming.

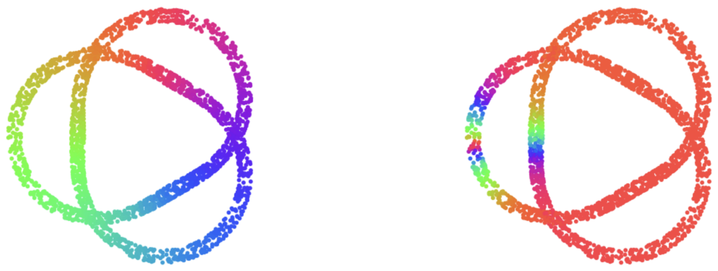

Figure 1: Parametrizing the circularity of a trefoil knot in 3D. Here we display a 2-dimensional representation, but the 3-dimensional point cloud does not have self intersections (in the sense that it is locally 1-dimensional everywhere). On the right, the output of the circular coordinates algorithm without applying the algebraic procedure to fix the lift of the cohomology class. On the left, the ouput of DREiMac, which implements this fix. Details about this example can be found in the documentation.

Another practical issue of the circular coordinates algorithm is its performance in the presence of more than one large scale circular feature (Figures 2 and 3). To address this, DREiMac implements the toroidal coordinates algorithm, introduced in Scoccola et al. (2023), which allows the user to select several 1-dimensional cohomology classes and returns coordinates that parametrize these circular features in a simpler fashion.

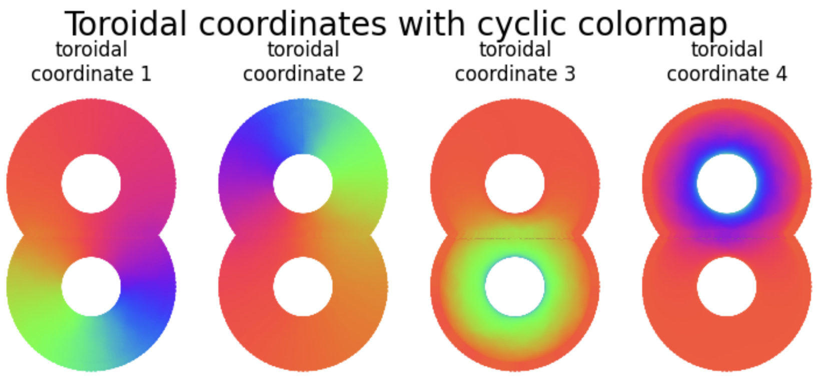

Figure 2: Parametrizing the circularity of a surface of genus two in 3D. Here we display a 2-dimensional representation, but the 3-dimensional point cloud does not have self intersections (in the sense that it is locally 2-dimensional everywhere). This is DREiMac's output obtained by running the toroidal coordinates algorithm. The output of running the circular coordinates algorithm is in Figure 3. Details about this example can be found in the documentation.

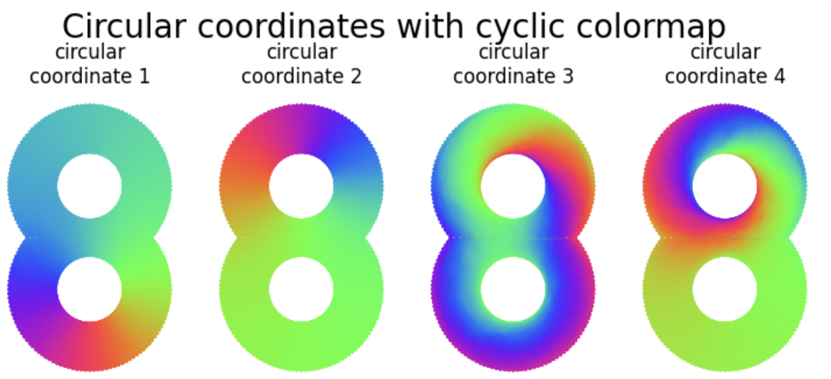

Figure 3: Parametrizing the circularity of a surface of genus two in 3D. This output is obtained by running the circular coordinates algorithm. The parametrization obtained is arguably less interpretable than that obtained by the toroidal coordinates algorithm, shown in Figure 2.

Previously not implemented cohomological coordinates. DREiMac implements real projective, complex projective, and lens coordinates, introduced in Perea (2018) and Polanco & Perea (2019). These allow the user to construct topologically meaningful coordinates for point clouds using cohomology classes with coefficients in $\mathbb{Z}/2\mathbb{Z}$, $\mathbb{Z}$, and $\mathbb{Z}/q\mathbb{Z}$ (q a prime), respectively, and in cohomological dimensions 1, 2, and 1, respectively.

Example

We illustrate DREiMac's capabilities by showing how it parametrizes the large scale circular features of the unprecessed COIL-20 dataset (Nene et al., 1996); details about this example can be found in the documentation. The dataset consists of gray-scale images of 5 objects, photographed from different angles. As such, it consists of 5 clusters, each cluster exhibiting one large scale circular feature; see Figure 4.

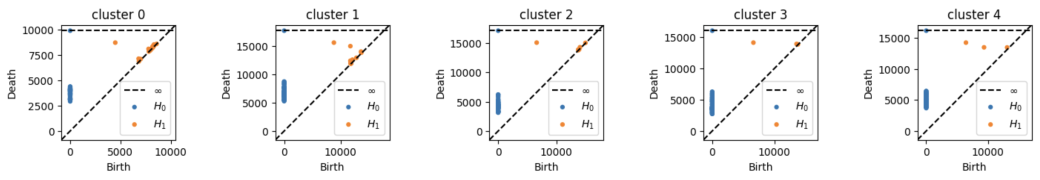

Figure 4: Persistent cohomology of 5 clusters of unprocessed COIL-20 dataset.

We use single-linkage to cluster the data into 5 clusters and compute the persistent cohomology of each cluster. We then run the circular coordinates algorithm on each cluster, using the most prominent cohomology class of each cluster. We display the result in Figure 5.

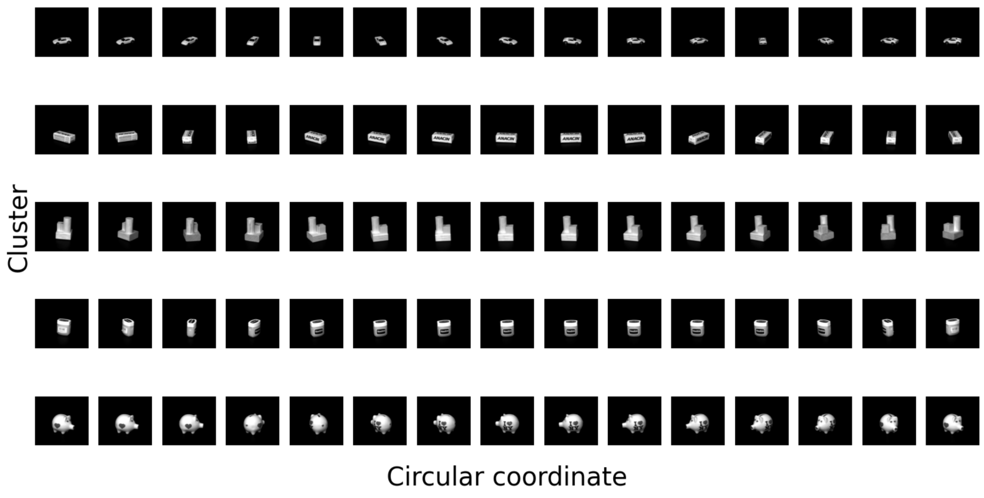

Figure 5: Unprocessed COIL-20 parametrized by clustering and circular coordinates.

Statement of contribution

J.A.P.: initial MATLAB implementation (not part of this software) and design of this software. C.T. and L.S.: design and implementation.

Acknowledgements

We thank Tom Mease for contributions and discussions. J.A.P. and L.S. were partially supported by the National Science Foundation through grants CCF-2006661 and CAREER award DMS-1943758.

References

Bauer, U. (2021). Ripser: Efficient computation of Vietoris-Rips persistence barcodes. *J. Appl. Comput. Topol.*, *5*(3), 391–423. https://doi.org/10.1007/s41468-021-00071-5

Čufar, M. (2020). Ripserer.jl: Flexible and efficient persistent homology computation in Julia. *Journal of Open Source Software*, *5*(54), 2614. https://doi.org/10.21105/joss.02614

de Silva, V., Morozov, D., & Vejdemo-Johansson, M. (2011a). Dualities in persistent (co)homology. *Inverse Problems*, *27*(12), 124003, 17. https://doi.org/10.1088/0266-5611/27/12/124003

de Silva, V., Morozov, D., & Vejdemo-Johansson, M. (2011b). Persistent cohomology and circular coordinates. *Discrete Comput. Geom.*, *45*(4), 737–759. https://doi.org/10.1007/s00454-011-9344-x

Gardner, R. J., Hermansen, E., Pachitariu, M., Burak, Y., Baas, N. A., Dunn, B. A., Moser, M.-B., & Moser, E. I. (2022). Toroidal topology of population activity in grid cells. *Nature*, *602*(7895), 123–128. https://doi.org/10.1038/s41586-021-04268-7

Kang, L., Xu, B., & Morozov, D. (2021). Evaluating state space discovery by persistent cohomology in the spatial representation system. *Frontiers in Computational Neuroscience*, *15*, 616748. https://doi.org/10.3389/fncom.2021.616748

Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A LLVM-based python JIT compiler. *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*.

https://doi.org/10.1145/2833157.2833162

Morozov, D. (2012). *Dionysus*. https://github.com/mrzv/dionysus

Nene, S. A., Nayar, S. K., Murase, H., & others. (1996). *Columbia object image library (coil-20)*.

Perea, J. A. (2018). Multiscale projective coordinates via persistent cohomology of sparse filtrations. *Discrete Comput. Geom.*, *59*(1), 175–225. https://doi.org/10.1007/s00454-017-9927-2

Perea, J. A. (2020). Sparse circular coordinates via principal $\mathbb{Z}$-bundles. In *Topological data analysis—the Abel Symposium 2018* (Vol. 15, pp. 435–458). Springer, Cham. ISBN: 978-3-030-43408-3

Polanco, L., & Perea, J. A. (2019). Coordinatizing data with lens spaces and persistent cohomology. In Z. Friggstad & J.-L. D. Carufel (Eds.), *Proceedings of the 31st Canadian Conference on Computational Geometry, CCCG 2019, August 8-10, 2019, University of Alberta, Edmonton, Alberta, Canada* (pp. 49–58).

Rybakken, E., Baas, N., & Dunn, B. (2019). Decoding of neural data using cohomological feature extraction. *Neural Computation*, *31*(1), 68–93. https://doi.org/10.1162/neco_a_01150

Scoccola, L., Gakhar, H., Bush, J., Schonsheck, N., Rask, T., Zhou, L., & Perea, J. A. (2023). Toroidal Coordinates: Decorrelating Circular Coordinates with Lattice Reduction. In E. W. Chambers & J. Gudmundsson (Eds.), *39th International Symposium on Computational Geometry (SoCG 2023)* (Vol. 258, pp. 57:1–57:20). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.SoCG.2023.57

Scoccola, L., & Perea, J. A. (2023). FibeRed: Fiberwise Dimensionality Reduction of Topologically Complex Data with Vector Bundles. In E. W. Chambers & J. Gudmundsson (Eds.), *39th International Symposium on Computational Geometry (SoCG 2023)* (Vol. 258, pp. 56:1–56:18). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.SoCG.2023.56

Tralie, C., Saul, N., & Bar-On, R. (2018). Ripser.py: A lean persistent homology library for python. *Journal of Open Source Software*, *3*(29), 925. https://doi.org/10.21105/joss.00925

Vejdemo-Johansson, M., Pokorny, F. T., Skraba, P., & Kragic, D. (2015). Cohomological learning of periodic motion. *Applicable Algebra in Engineering, Communication and Computing*, *26*(1), 5–26. https://doi.org/10.1007/s00200-015-0251-x