The Journal of Open Source Software

Teaspoon: A Python Package for Topological Signal Processing

Firas A. Khasawneh [1*], **Elizabeth Munch** [1*¶], **Danielle Barnes**[1], **Max M. Chumley** [1], **İsmail Güzel** [4,5], **Audun D. Myers** [2], **Sunia Tanweer** [1], **Sarah Tymochko** [3], **and Melih Yesilli** [1]

1 Michigan State University, East Lansing, MI, USA **2** Pacific Northwest National Lab (PNNL), USA **3** University of California, Los Angeles, USA **4** Network Technologies Department, TÜBİTAK ULAKBİM, Ankara, TÜRKİYE **5** Institute of Applied Mathematics, Middle East Technical University, Ankara, TÜRKİYE ¶ Corresponding author * These authors contributed equally.

DOI: 10.21105/joss.07243

Software

- Review
- Repository
- Archive

Editor: Hugo Ledoux

Reviewers:

- @yossibokorbleile
- @EduPH

Submitted: 26 August 2024
Published: 11 March 2025

Summary

The field of topological data analysis (TDA) has risen in the last two decades to become a mature field of research providing a unique perspective on data analysis. TDA consists of a suite of tools inspired by the field of Algebraic Topology, which encodes shape and structure in data in a quantitative manner. One particular subfield of work has focused on using TDA tools for the analysis of time series, colloquially known as Topological Signal Processing, or TSP. The python package teaspoon has been built specifically to cater to the needs of researchers working in the field of TSP, with the added benefit that the code can be used for other forms of input data beyond signal.

Recent work has largely focused on the use of persistent homology and its variants for this context, thus this has been the main tool utilized in teaspoon. While a full discussion of the specifics of persistence is outside the scope of this brief paper, we give a brief introduction here and direct the interested reader to Dey & Wang (2021) and Munch (2017) for more information. Standard homology (see, e.g., Hatcher (2002)) is a construction which builds a vector space for any input topological space X and a given dimension p of structure to be studied. Denoted $H_p(X)$, the $p = 0$ dimensional homology encodes the structure of connected components; $p = 1$ encodes loops; $p = 2$ encodes voids; and higher dimensional versions exist without the interpretability of these lower dimensional versions. While homology is defined for a fixed topological space, persistent homology studies the changing homology for a changing topological space. Indeed, through a fundamental theorem of persistence (Crawley-Boevey, 2015), we can use this sequence of vector spaces to determine when p-dimensional structures appear (or are born) and disappear (or die). The persistence diagram is a 2D scatter plot of points, each one giving the (birth, death) coordinates for a particular p-dimensional structure.

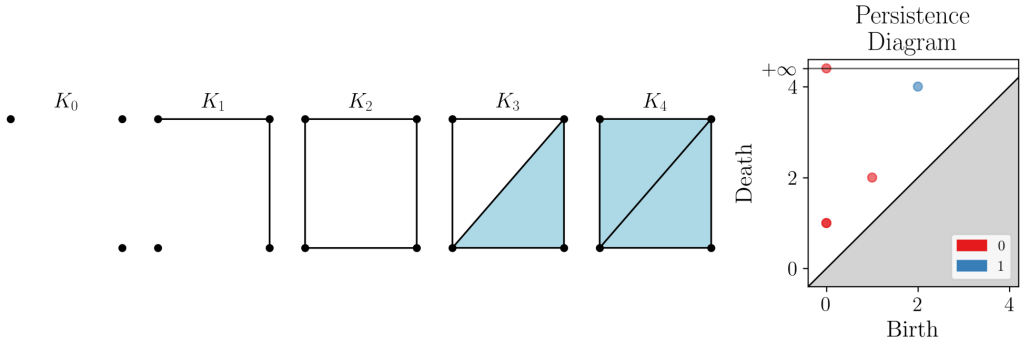

Figure 1: An example filtration of simplicial complexes and the corresponding persistence diagram.

As an example, in Figure 1 we have a sequence of topological spaces in which we wish to track how the homology is changing, along with the corresponding persistence diagram (right). Each red point in the persistence diagram at right represents a connected component (an H_0 feature), and each blue point represents a cycle (a H_1 feature). For example, three vertices from K_0 are connected by edges in K_1, thus there are two persistence points at $(0, 1)$ in dimension 0 corresponding to the fact that three connected components have merged into one. Similarly, a cycle first appears in K_2 and is filled in at K_4, so there is a persistence point in dimension 1 at (2,4).

In the following section, we show how persistence can be used in a pipeline for signal processing by aligning each of the steps with the modules of the `teaspoon` package.

Package Modules

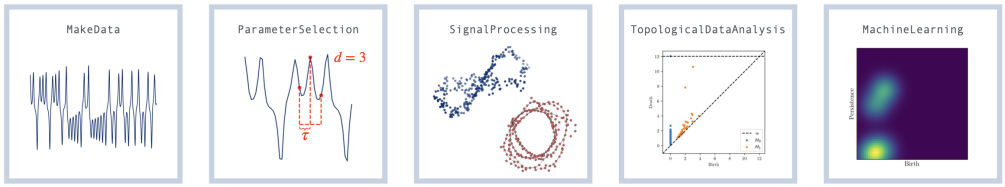

Figure 2: The basic pipeline of a TSP project aligned with the modules of `teaspoon`.

Many topological signal processing projects can be fit into the pipeline shown in Figure 2. This pipeline aligns with the five submodules of `teaspoon`, which we describe in more detail in the following subsections.

Make Data

First, we assume that we are given a collection of time series, perhaps simulated from varying a parameter in a known dynamical system. For testing purposes, this can be done by generating synthetic data directly from the *MakeData* module. The main feature of the module is the *Dynamic Systems Library (DynSysLib)* submodule, which includes a wide variety of dynamical systems. As of writing, this includes maps, autonomous and driven dissipative flows, conservative flows, periodic functions, noise models, and delayed flows. In addition, there is also synthetic data generation available for other types of data often used in TDA pipelines, including point clouds (such as those drawn from an annulus or torus) and functions (such as Gaussian fields).

Parameter Selection

Once we have a time series, we can apply transforms to prepare the data for different versions of persistence, however these often require parameter choices. One of the most commonly used transforms in the signal processing literature is the Takens embedding (sometimes called delay coordinate embedding) of the data into $\mathbb{R}^d$ for given dimension d with a delay parameter τ. The choices of d and τ for the embedding are subtle, however the *Parameter Selection* module has an extensive array tools for the automated computation of the parameters. These include the standard options such as mutual information, auto-correlation, and false nearest neighbors; along with newly developed options using TDA and persistent homology to estimate embedding delays (Audun D. Myers et al., 2024) by finding the delay that maximizes the 1D persistence lifetime or loop size of the attractor. We continue to implement algorithms to make this module more comprehensive such as generalizations to the false nearest neighbors method in (Chelidze, 2017) which uses strands of points to more reliably estimate the embedding dimension when noise is present and similarly, the 'Cao method' (Cao, 1997) has the added benefit of distinguishing deterministic signals from stochastic signals.

Signal Processing

The transforms converting a time series into a mathematical structure available for TDA analysis are contained in the *Signal Processing (SP)* module. This includes both standard tools as well as newly developed techniques that incorporate topological information. For instance, the Takens embedding is included, which converts an input time series into a point cloud. However, there are also more recent techniques which convert a time series into a network, such as the Ordinal Partition Network (OPN) (McCullough et al., 2015; Audun D. Myers, Khasawneh, et al., 2023) or the Coarse Grained State Space (CGSS) network (Audun D. Myers, Chumley, et al., 2023; Wang & Tian, 2016). Similarly, standard entropy computations are included, as well as persistent (as in persistent homology) entropy. For more conventional time series analysis, a noise robust zero-crossing detection tool (Tanweer, Khasawneh, & Munch, 2024) is included which detects all crossings of a discrete signal at once. This module also includes the *Texture Analysis* submodule, which provides techniques for comparing experimental and nominal surface textures in manufacturing/machining applications (Chumley et al., 2023). The *Stochastic P-Bifurcation Detection* provides homological techniques for automatic and unbiased detection of Phenomenological Bifurcations in stochastic dynamical systems (Tanweer, Khasawneh, Munch, & Tempelman, 2024; Tanweer & Khasawneh, 2024).

Topological Data Analysis

After performing any necessary transformations, the *Topological Data Analysis (TDA)* module has tools for computing topological signatures of data persistence on input data. This module is largely wrappers for externally available code since much work has already been done to optimize this aspect of the pipeline. Point cloud persistence, for instance when taking the Takens embedding as input, is computed using the external *ripser* Python package (Bauer, 2021) in Scikit-TDA (Saul & Tralie, 2019). Zero dimensional sublevel set persistence is computed with entirely internal code (Audun D. Myers et al., 2022). The module also offers code for computing bottleneck distance—which relies on the *scikit-tda persim* package (Saul & Tralie, 2019)—and Wasserstein distance based on Optimal Transport Theory. There is also code which makes it easier to use the fast zigzag software (Dey & Hou, 2022) by providing a wrapper for generating the input file given a list of point clouds as well as filtering the resultant persistence diagram. As newly available code is released and maintained, this means that all internal functions can be switched to other external packages as needed without a great deal of update to the remainder of the code.

Machine Learning

Finally, once the time series has been converted into a persistence diagram representation, the *Machine Learning (ML)* module gives a variety of featurization methods to convert the persistence diagram into a vector based representation amenable to regression and classification tasks. These include persistence landscapes (Bubenik, 2015), persistence images (Adams et al., 2017), Carlsson coordinates (Adcock et al., 2016), template functions (Perea et al., 2022), path signatures (Chevyrev et al., 2020), and kernel methods (Reininghaus et al., 2015).

Statement of need

The teaspoon package is focused on applications of TDA to time series with an emphasis on ease of usability in a Python environment. Optimization of the computation of persistence itself has been well studied by others and excellent code already exists for this aspect of the pipeline (Otter et al., 2017). Where applicable, teaspoon uses these codebanks, particularly for persistent homology computations. Existing code banks include Ripser (Bauer, 2021), GUDHI (Boissonnat et al., 2016), giotto-tda (Tauzin et al., 2020), dionysus2 (Morozov, 2019), scikit-tda (Saul & Tralie, 2019), R-TDA (Fasy et al., 2014), and the Topology Toolkit (TTK) (Bin Masood et al., 2019). However, persistence in these codebanks is often provided in a very general context. So, teaspoon fills the gap by providing tailored, well-documented tools for time series that can be used with a lower barrier to entry. This is not covered in other packages which are meant for broad applicability without specialization.

Representative Publications Using Teaspoon

The teaspoon package was started in 2017 as a GitLab repository, and was ported to GitHub in 2018. A previous but now outdated paper outlined the basic functionality of teaspoon at the time (Audun D. Myers et al., 2020). Because of its longevity, a non-exhaustive but extensive list of papers (Chumley et al., 2023; Elchesen et al., 2022; Gilpin, 2021; Güzel et al., 2022; Jones & Wei, 2023; Audun D. Myers, Kvinge, et al., 2023; Audun D. Myers et al., 2022; Audun D. Myers, Khasawneh, et al., 2023; Audun D. Myers, Muñoz, et al., 2023, 2023; Audun D. Myers, Chumley, et al., 2023; Perea et al., 2022; Tymochko et al., 2019) as well as theses (Collins, 2022; Tymochko, 2022; Yi, 2022) have utilized teaspoon.

Acknowledgements

This material is based in part upon work supported by the Air Force Office of Scientific Research under Award No. FA9550-22-1-0007. It was additionally supported in part by the National Science Foundation through grants CCF-1907591, CCF-2106578, and CCF-2142713.

References

Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Chepushtanova, S., Hanson, E., Motta, F., & Ziegelmeier, L. (2017). Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, *18*(8), 1–35. http://jmlr.org/papers/v18/16-337.html

Adcock, A., Carlsson, E., & Carlsson, G. (2016). The ring of algebraic functions on persistence bar codes. *Homology, Homotopy and Applications*, *18*(1), 381–402. https://doi.org/10.4310/HHA.2016.v18.n1.a21

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

Bin Masood, T., Budin, J., Falk, M., Favelier, G., Garth, C., Gueunet, C., Guillou, P., Hofmann, L., Hristov, P., Kamakshidasan, A., Kappe, C., Klacansky, P., Laurin, P., Levine, J., Lukasczyk, J., Sakurai, D., Soler, M., Steneteg, P., Tierny, J., … Wozniak, M. (2019). An Overview of the Topology ToolKit. *TopoInVis*.

Boissonnat, J.-D., Glisse, M., Kramar, M., Maria, C., & Rouvreau, V. (2016). *GUDHI: Geometry understanding in higher dimensions*. http://gudhi.gforge.inria.fr/. http://gudhi.gforge.inria.fr/

Bubenik, P. (2015). Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, *16*, 77–102. http://jmlr.org/papers/v16/bubenik15a.html

Cao, L. (1997). Practical method for determining the minimum embedding dimension of a scalar time series. *Physica D: Nonlinear Phenomena*, *110*(1-2), 43–50. https://doi.org/10.1016/s0167-2789(97)00118-8

Chelidze, D. (2017). Reliable estimation of minimum embedding dimension through statistical analysis of nearest neighbors. *Journal of Computational and Nonlinear Dynamics*, *12*(5), 051024. https://doi.org/10.1115/1.4036814

Chevyrev, I., Nanda, V., & Oberhauser, H. (2020). Persistence paths and signature features in topological data analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(1), 192–202. https://doi.org/10.1109/TPAMI.2018.2885516

Chumley, M. M., Yesilli, M. C., Chen, J., Khasawneh, F. A., & Guo, Y. (2023). Pattern characterization using topological data analysis: Application to piezo vibration striking treatment. *Precision Engineering*, *83*, 42–57. https://doi.org/10.1016/j.precisioneng.2023.05.005

Collins, J. R. (2022). Topological time-series classification [PhD thesis]. In *ProQuest Dissertations and Theses* (p. 105). ISBN: 9798802715796

Crawley-Boevey, W. (2015). Decomposition of pointwise finite-dimensional persistence modules. *Journal of Algebra and Its Applications*, *14*(05), 1550066. https://doi.org/10.1142/s0219498815500668

Dey, T. K., & Hou, T. (2022). *Fast computation of zigzag persistence*. https://doi.org/10.48550/arxiv.2204.11080

Dey, T. K., & Wang, Y. (2021). *Computational topology for data analysis*. Cambridge University Press.

Elchesen, A., Hartsock, I., Perea, J. A., & Rask, T. (2022). *Learning on persistence diagrams as radon measures*. https://doi.org/10.48550/ARXIV.2212.08295

Fasy, B. T., Kim, J., Lecci, F., & Maria, C. (2014). *Introduction to the r package TDA*. https://arxiv.org/abs/http://arxiv.org/abs/1411.1830v2

Gilpin, W. (2021). Chaos as an interpretable benchmark for forecasting and data-driven modelling. *NeurIPS (Neural Information Processing Systems) 2021*. https://doi.org/10.48550/ARXIV.2110.05266

Güzel, İ., Munch, E., & Khasawneh, F. A. (2022). Detecting bifurcations in dynamical systems with CROCKER plots. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *32*(9). https://doi.org/10.1063/5.0102421

Hatcher, A. (2002). *Algebraic topology*. Cambridge University Press.

Jones, B., & Wei, G. (2023). *Persistent directed flag laplacian*. https://doi.org/10.48550/ARXIV.2312.02099

McCullough, M., Small, M., Stemler, T., & Iu, H. H.-C. (2015). Time lagged ordinal partition networks for capturing dynamics of continuous dynamical systems. *Chaos: An*

Interdisciplinary Journal of Nonlinear Science, *25*(5), 053101. https://doi.org/10.1063/1.4919075

Morozov, D. (2019). *Dionysus2*. http://www.mrzv.org/software/dionysus2/.

Munch, E. (2017). A user's guide to topological data analysis. *Journal of Learning Analytics*, *4*(2). https://doi.org/10.18608/jla.2017.42.6

Myers, Audun D., Chumley, M. M., & Khasawneh, F. A. (2024). Delay parameter selection in permutation entropy using topological data analysis. *La Matematica*, 1–34. https://doi.org/10.1007/s44007-024-00110-4

Myers, Audun D., Chumley, M. M., Khasawneh, F. A., & Munch, E. (2023). Persistent homology of coarse-grained state-space networks. *Physical Review E*, *107*(3), 034303. https://doi.org/10.1103/physreve.107.034303

Myers, Audun D., Khasawneh, F. A., & Fasy, B. T. (2022). ANAPT: Additive noise analysis for persistence thresholding. *Foundations of Data Science*, *4*(2), 243. https://doi.org/10.3934/fods.2022005

Myers, Audun D., Khasawneh, F. A., & Munch, E. (2023). Persistence of weighted ordinal partition networks for dynamic state detection. *SIAM Journal on Applied Dynamical Systems*, *22*(1), 65–89. https://doi.org/10.1137/22m1476848

Myers, Audun D., Kvinge, H., & Emerson, T. (2023). TopFusion: Using topological feature space for fusion and imputation in multi-modal data. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 600–609. https://doi.org/10.1109/cvprw59228.2023.00067

Myers, Audun D., Muñoz, D., Khasawneh, F. A., & Munch, E. (2023). Temporal network analysis using zigzag persistence. *EPJ Data Science*, *12*(1). https://doi.org/10.1140/epjds/s13688-023-00379-5

Myers, Audun D., Yesilli, M., Tymochko, S., Khasawneh, F., & Munch, E. (2020). *Teaspoon: A comprehensive python package for topological signal processing*. https://openreview.net/pdf?id=qUoVqrIcy2P

Otter, N., Porter, M. A., Tillmann, U., Grindrod, P., & Harrington, H. A. (2017). A roadmap for the computation of persistent homology. *EPJ Data Science*, *6*(1). https://doi.org/10.1140/epjds/s13688-017-0109-5

Perea, J. A., Munch, E., & Khasawneh, F. A. (2022). Approximating continuous functions on persistence diagrams using template functions. *Foundations of Computational Mathematics*, *23*(4), 1215–1272. https://doi.org/10.1007/s10208-022-09567-7

Reininghaus, J., Huber, S., Bauer, U., & Kwitt, R. (2015). A stable multi-scale kernel for topological machine learning. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4741–4748. https://doi.org/10.1109/CVPR.2015.7299106

Saul, N., & Tralie, C. (2019). *Scikit-TDA: Topological data analysis for python*. https://doi.org/10.5281/zenodo.2533369

Tanweer, S., & Khasawneh, F. A. (2024). Topological detection of phenomenological bifurcations with unreliable kernel density estimates. *Probabilistic Engineering Mechanics*, *76*, 103634. https://doi.org/10.1016/j.probengmech.2024.103634

Tanweer, S., Khasawneh, F. A., & Munch, E. (2024). Robust crossings detection in noisy signals using topological signal processing. *Foundations of Data Science*, *6*(2), 154–171. https://doi.org/10.3934/fods.2024006

Tanweer, S., Khasawneh, F. A., Munch, E., & Tempelman, J. R. (2024). A topological framework for identifying phenomenological bifurcations in stochastic dynamical systems. *Nonlinear Dynamics*, *112*(6), 4687–4703. https://doi.org/10.1007/s11071-024-09289-1

Tauzin, G., Lupo, U., Tunstall, L., Pérez, J. B., Caorsi, M., Medina-Mardones, A., Dassatti, A., & Hess, K. (2020). *Giotto-tda: A topological data analysis toolkit for machine learning and data exploration*. https://arxiv.org/abs/2004.02551

Tymochko, S. (2022). Topological approaches for quantifying the shape of time series data [PhD thesis]. In *ProQuest Dissertations and Theses* (p. 123). ISBN: 979-8-209-98467-2

Tymochko, S., Munch, E., & Khasawneh, F. A. (2019, December). Adaptive partitioning for template functions on persistence diagrams. *2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA)*. https://doi.org/10.1109/icmla.2019.00202

Wang, M., & Tian, L. (2016). From time series to complex networks: The phase space coarse graining. *Physica A: Statistical Mechanics and Its Applications*, *461*, 456–468. https://doi.org/10.1016/j.physa.2016.06.028

Yi, W. (2022). *When hearts beat as one – cardiac dynamics and synchrony in string quartet performances* [Master's thesis, University of Oslo]. https://www.duo.uio.no/handle/10852/96059