Benchmarking R packages for Calculation of Persistent Homology

by Eashwar V. Somasundaram, Shael E. Brown, Adam Litzler, Jacob G. Scott, and Raoul R. Wadhwa

Abstract Several persistent homology software libraries have been implemented in R. Specifically, the Dionysus, GUDHI, and Ripser libraries have been wrapped by the **TDA** and **TDAstats** CRAN packages. These software represent powerful analysis tools that are computationally expensive and, to our knowledge, have not been formally benchmarked. Here, we analyze runtime and memory growth for the 2 R packages and the 3 underlying libraries. We find that datasets with less than 3 dimensions can be evaluated with persistent homology fastest by the GUDHI library in the **TDA** package. For higher-dimensional datasets, the Ripser library in the TDAstats package is the fastest. Ripser and **TDAstats** are also the most memory-efficient tools to calculate persistent homology.

Introduction

Topological data analysis (TDA) is a broad set of methodologies that characterizes structural features of datasets inspired by topological principles. It has a broad range of usage, from viral evolution to physical chemistry (Chan et al., 2013; Offroy and Duponchel, 2016). Within the umbrella of TDA, persistent homology represents an algebraic approach to understanding the number, characteristics, and persistence of structural features in an n-dimensional point cloud. In the basic workflow of persistent homology, a series of simplicial complexes are generated on point clouds to characterize topological features. There are several methods to generate these complexes on point clouds. In this paper, we focus on persistent homology of the Vietoris-Rips and alpha complexes, which use simplicial complexes to approximate topologic relationships in point clouds. The exact method of constructing these complexes is described in the Mathematics section. Essentially, we measure features that are discovered by the algorithm at a particular stage and disappear at a later stage. The difference between these stages is persistence. Features with larger persistence more likely represent real geometric patterns rather than noise.

There are several C++ libraries available to researchers that calculate alpha and Vietoris-Rips complexes, such as Dionysus, GUDHI, and Ripser (Morozov, 2018; Maria et al., 2016; Bauer, 2019). These libraries have been wrapped in R by the **TDA** and **TDAstats** packages (Fasy et al., 2019; Wadhwa et al., 2018). Although useful, calculating persistent homology for large datasets is often limited due to computational complexity (Otter et al., 2017). As a result, researchers often limit persistent homology analysis to lower dimensions. However, ignoring features in higher dimensions may cause significant information loss, underutilizing persistent homology's capabilities. Here, we aim to benchmark two R packages - **TDA** and **TDAstats** - and enable researchers to most efficiently calculate persistent homology in R.

Mathematics of Persistent Homology

An n-dimensional simplex is the convex hull of $n+1$ points in a Euclidean space. More intuitively, an n-dimensional simplex is the simplest n-dimensional object (e.g., a 0-simplex is a point, a 1-simplex is a line, a 2-simplex is a triangle, 3-simplex is a tetrahedron). These simplices can be glued together on common sub-simplices to form a simplicial complex (e.g., two triangles sharing a common side). In a simplicial complex, topological features will arise that can be characterized by Betti numbers. Each Betti number, denoted by B_k, k counts the number of features in dimension k. B_0 counts the number of connected components, B_1 counts refer to loops, B_2 counts the number of voids, and so on (Edelsbrunner and Harer, 2008).

There are several different methods to construct a simplicial complex on a given point cloud S, but this paper focuses on the Vietoris-Rips and alpha complexes. The Vietoris-Rips complex is perhaps the most common method for constructing a simplicial complex to calculate persistent homology (Hausmann, 1996). In a point cloud of k points in 2 dimensions, a distance parameter, $\delta > 0$, can be used to draw a circle of diameter δ around every point in S. For point clouds in 3 dimensions, spheres of diameter δ are drawn around each point. For dimensions k greater than 3, a k-dimensional hypersphere is drawn around each point. The remainder of this explanation will focus on the 2-dimensional case. If δ is sufficiently large, then some of the resulting circles may intersect. In this case, a line is drawn to connect the points at the center of the intersecting circles. When a triple of points is connected, we add a triangle (2-simplex). When a quadruple of points are connected, we add a

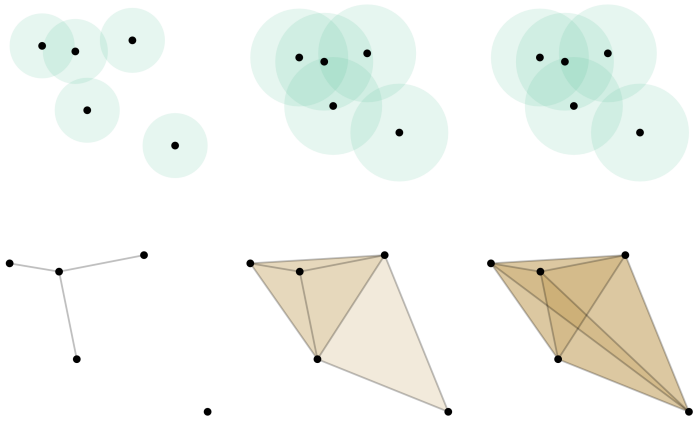

Figure 1: Basic Visualization of the Vietoris-Rips Complex. For a given parameter, δ, δ-diameter circles are drawn around each point. If two circles intersect, a point is drawn between their centers. As δ continues to grow, more circles intersect, filling out the simplicial complex. Features on the simplicial complex appear and die as δ increases. These features' dimensions, birth, and death are recorded in an nx3 matrix. Eventually, the full convex hull is drawn, ending the "filtration" process.

tetrahedron (3-simplex) and so forth. However, we only add simplices at most of the dimension of the space of the point clouds (e.g., only up to 3-simplices are added in a 3-dimensional point cloud). This group of points and lines form the skeleton of a simplicial complex. For each distance parameter, δ, there will be a single simplicial complex associated with it. As δ increases, different topological features may appear, persist, and eventually disappear.

Once δ reaches the maximum Euclidean distance between any pair of points in the point cloud, a convex hull will form around all k points creating a $(k-1)$-dimensional simplex. A 3-column matrix can be created recording the dimension of each feature, the δ at which that feature appeared, and the δ at which it disappeared. This matrix characterizes the persistent homology of that point cloud.

Alpha complexes provide another method to generate simplicial complexes on the point cloud S. For alpha complexes, we partition the whole space in which the data resides into cells such that each cell contains exactly one data point x, and the cell of that data point is the set of all points closer to x than any other data points. Such a partition is also known as a Voronoi diagram. The nerve of a Voronoi diagram is equivalent to the Delaunay Triangulation (Edelsbrunner and Mücke, 1994). Alpha complexes are simplicial complexes that are subsets of the Delaunay Triangulation. The parameter, α, can describe the radius of a ball (dimension matches dimension of the space) of each point in the point cloud S much, like δ describes the diameter of a circle in the Vietoris-Rips complex. We first intersect the α radius balls with their own Voronoi cell and then search for intersections of these subsetted balls to form simplices. Once α is large enough, the full Delaunay Triangulation is formed. In between these stages, the birth and death of features at certain values of α can be captured in a 3-column persistent homology matrix much like the Vietoris-Rips complex. One key difference from the Vietoris-Rips complex is that edges can only form between neighboring points in the alpha complex.

In both methods, the boundary matrix records all simplicial complexes for each parameter value (δ for Vietoris-Rips complexes and α for alpha complexes). Calculating persistent homology is divided

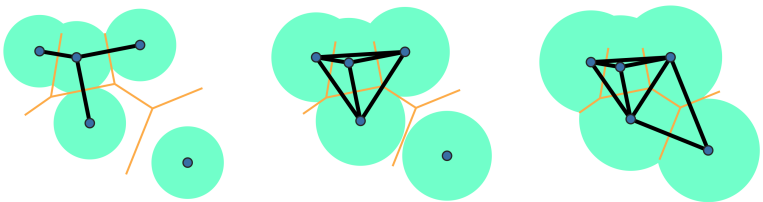

Figure 2: Basic visualization of the Alpha complex. For a given α, α-radius balls are drawn around each point, and the union of the balls is taken. Then, an intersection between this union of α-balls and the Vornoi diagram is taken. A connecting segment is drawn between points in adjacent Voronoi cells once the α-ball fills out the Voronoi diagram. As α grows, more circles fill out the Voronoi cells. Once α is large enough, the Delaunay Triangulation is formed.

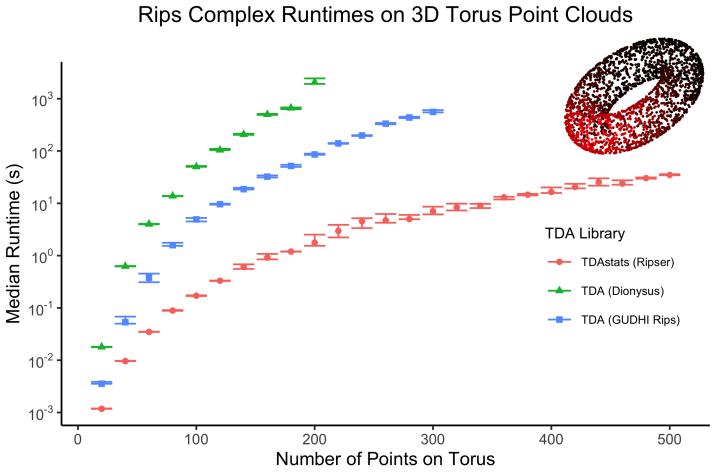

Figure 3: Calculating persistent homology of a torus with three TDA libraries. Median runtime (min to max, $n = 10$ iterations per data point) for each TDA library (denoted by color) is plotted against point cloud size. Homological features of up to 2 dimensions were calculated. Time complexity follows a power law for all three libraries (see GitHub repo for regression details). Although the libraries have similar runtimes for smaller point clouds, Dionysus has a clear disadvantage when the number of points exceeds 100. When the number of points exceeds 200, Ripser has a clear advantage over GUDHI, which maintains its advantage over Dionysus.

into two steps: (1) forming the boundary matrix and (2) reducing the boundary matrix to be able to read off the topological features of each dimension and their birth/death values of the parameter. The second step can be computed in at most O(k^3) steps, where k is the number of rows (and columns) of the boundary matrix. The size of the boundary matrix can describe the memory complexity on the random access memory (RAM) for persistent homology calculations. We compare memory complexity between alpha and Vietoris-Rips complexes in this paper.

Alpha complex calculations have a run time complexity of $O(n^{d/2})$, and Vietoris-Rips complex calculations have a run time complexity of $O(2^n)$, where n is the number of points and d is the point cloud dimension (Otter et al., 2017). Vietoris-Rips's run time and memory are exponential with regards to point number (but constant with data dimension) in contrast to alpha complexes where run time and memory are polynomial with point number (but exponential with data dimension). Therefore, we can predict that low dimensional point clouds favor alpha complexes, but fewer points in higher dimension favor Vietoris-Rips complexes.

Methods

We use **readr** v1.3.1 to read rectangular data (Wickham et al., 2018), **ggplot2** v3.2.1 (Wickham, 2016), **scatterplot3d** v0.3-41 (Ligges and Mächler, 2003), **recexcavAAR** v0.3.0 (Schmid and Serbe, 2017), **deldir** v0.1 (Turner, 2020), ggtda v0.1 (Brunson et al., 2020), and **magick** v2.2 (Ooms, 2019) to visualize data, **bench** v1.0.4 to collect benchmark data (Hester, 2019), **TDA** v1.6.9 (Fasy et al., 2019) and **TDAstats** v0.4.1 (Wadhwa et al., 2018) to calculate persistent homology of Vietoris-Rips and alpha simplicial complices, and **pryr** v0.1.4 for calculations involving R objects (Wickham, 2018). Median runtime calculations are shown along with the minimum and maximum of 10 iterations per benchmark. Datasets were generated by sampling functions in base R to generate points uniformly distributed over circles (dimension = 2), spheres (dimension = 3), filled squares (dimension = 2), filled cubes (dimension = 3, 4), and tori (dimension = 3). The number of points per point cloud varied from 25 to 500 along with intervals of 25 points, which were empirical limits chosen after considering available computational resources. For consistency between software libraries, the minimum and maximum simplicial complex radii were predetermined for each point cloud and provided as parameters to the **TDA** and **TDAstats** R packages. Within the **TDA** package, benchmark data was collected for the GUDHI (Maria et al., 2016) and Dionysus (Morozov, 2018) libraries; within the **TDAstats** package, benchmark data was collected for the Ripser (Bauer, 2019) library. As alpha complex calculation was only implemented in GUDHI, alpha complex benchmark data was naturally only collected for the single library. Measuring memory usage proved challenging since all the libraries calculating persistent homology were implemented in either C++ or Java and then wrapped in R as part of a CRAN package. Thus, memory burden was indirectly measured by using boundary matrix size as a

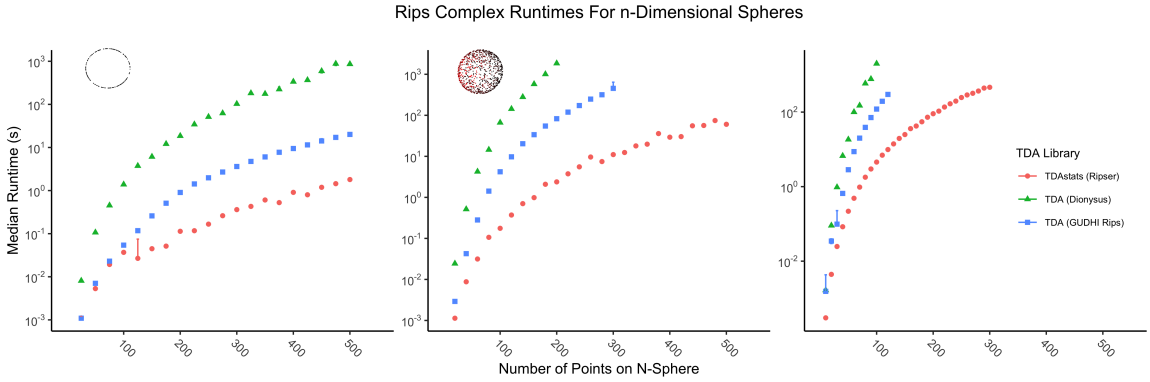

Figure 4: Calculating persistent homology of round point clouds of varying dimensions with three TDA libraries. Median runtime (min to max, $n = 10$ iterations per data point) for each TDA library (denoted by color) is plotted against point cloud size and faceted by data dimension. The left panel compares library performance for a 2-dimensional circular point cloud, the center panel for a 3-dimensional spherical point cloud, and the right panel for a 4-dimensional hyperspherical point cloud. Maximum feature dimensions (one less than the data dimension) were calculated in each case.

proxy. Given that Ripser optimizes computation of persistent homology by avoiding calculation of a boundary matrix, memory use benchmarks are not provided for Ripser and, consequently, **TDAstats**.

Benchmark data were collected twice - once on a local machine and once on a remote computing node, each of which featured 16 GB RAM. Both datasets were compared for consistency and are publicly available at the repository linked below. Data from the remote computing node is visualized in this report. The larger point clouds required more than 16 GB of RAM to calculate persistent homology using a subset of the libraries; attempts to compute results resulted in runtime errors, and the corresponding output is missing from the corresponding figures and tables. Fully reproducible code for all numerical results and figures can be found at `https://github.com/eashwarsoma/TDA-benchmark`. This GitHub repository also contains instructions for generating the Supplement referenced in this report's results. Video explanations of TDA concepts and reproducing all results in this report can be found at `https://tinyurl.com/TDABench`.

Results

Computing persistent homology of a canonical torus grants quick insight into efficiency of each library (Figure 3). Dionysus exhibits the longest median runtime, and, although Ripser and GUDHI have similar runtimes for smaller point clouds, as the number of points increases Ripser eventually has a significant lead. Next, we compare library performance with multiple canonical datasets to ensure that the noted pattern generalizes.

Tori do not trivially generalize to other dimensions, but circles do. Benchmarking on a circular point cloud permits confirmation of the pattern in Figure 3 while also revealing how the libraries compare as the dataset dimension increases. Figure 4 exhibits the resulting data for a 2-dimensional circle (left panel), 3-dimensional sphere (center panel), and 4-dimensional hypersphere (right panel). When the dataset dimension equals 2, GUDHI practically matches Ripser's performance in outpacing Dionysus. However, in the case of the 3-dimensional sphere, the pattern visualized in Figure 3 for the 3-dimensional torus is again present. By the 4th dimension, the gap between Ripser and GUDHI widens. Of note, missing points for larger datasets in Figure 4 are not plotted if and only if calculating persistent homology caused an error due to insufficient RAM. Thus, for the hypersphere, Ripser was able to calculate persistent homology for a dataset with approximately 3 times as many points as Dionysus and over 2 times as many as GUDHI. Interestingly, all curves plotted in Figure 4 grow polynomially with respect to the number of points (see Supplement for regression details).

Large data and feature dimensionality often restrict persistent homology calculations to small point clouds due to computational limits. When calculating persistent homology on a high-dimensional point cloud, as Vietoris-Rips feature dimension increases, there is a corresponding increase in runtime (Figure 5). Dionysus is clearly outmatched by GUDHI and Ripser as feature dimension increases, with the difference being clearest for larger point clouds; by feature dimension 5, Ripser outpaces GUDHI as well (Figure 5). It is unclear whether runtime for each library grows polynomially or exponentially (see Supplement for regression details).

Even with a constant feature dimension, the underlying data dimension could play a role in

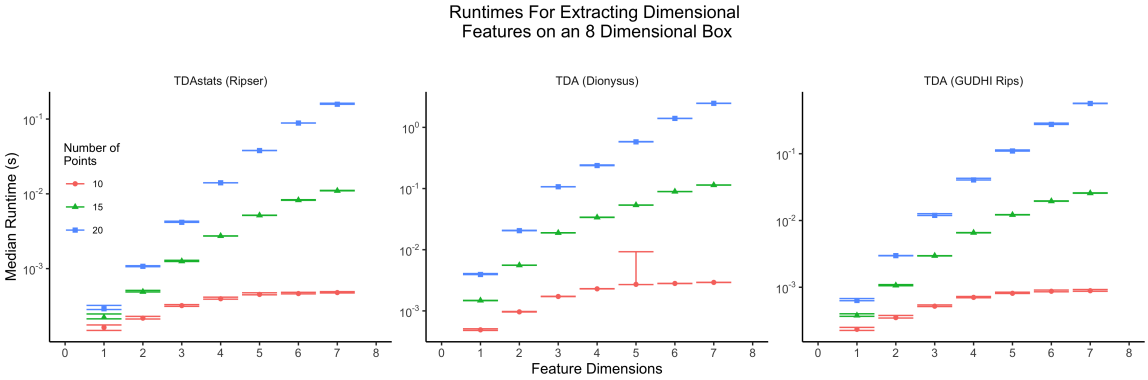

Figure 5: Comparison of Vietoris-Rips complex persistent homology calculation as a function of feature dimension. Median runtime (min to max, $n = 10$ iterations per data point) for various point cloud sizes (denoted by color) is plotted against the calculated feature dimension and faceted by TDA library. Persistent homology was calculated on a uniformly distributed random sample of points contained within a 1 unit, 8-dimensional cube. Computational limitations of calculating persistent homology for a large number of feature dimensions restricted point clouds to relatively small sizes.

the runtime of persistent homology calculation. Figure 6 compares the handling of this issue by the Vietoris-Rips complex and the alpha complex. Since GUDHI is the only library implementing functionality with an alpha complex, we compare its implementations of the Vietoris-Rips and alpha complices. Due to computational limitations, an alpha complex could not be calculated for any point clouds with data dimensions exceeding 3. Two notable aspects of Figure 6 stand out. First, the alpha complex calculation clearly runs faster than the Vietoris-Rips complex calculation, a trend that becomes clearer as point cloud size increases. Second, although the Vietoris-Rips complex calculation runtime appears to be independent of the underlying data dimension, the alpha complex calculation is dependent on it. Figure 6 shows a subtle difference between data dimensions 2 and 3 as point cloud size increases. Although unconcerning for a data dimension up to 3, failure to run any alpha complex calculations with a data dimension of 4 could be cause for concern.

In addition to runtime differences, the three Vietoris-Rips homology engines differ in memory use. All three engines appeared to follow power law growth, with a linear trend on log-log plots (Figure 7). However, for nearly all combinations of point cloud dimension and shape, TDAstats used the least memory, and Dionysus used the most, with TDAstats also growing with the smallest power law exponent as the number of points increased for most point clouds. For most point clouds, runtime and memory complexity for TDAstats (Ripser) grew with a power function at least one degree less than the other engines (Figure 8).

Discussion

As persistent homology calculations continue to become a more popular tool to analyze complex multidimensional data, it will be important to understand from a computational perspective which method to use. In this paper, we examined two forms of persistent homology complexes: Vietoris-Rips and alpha complexes. Both algorithms describe topological features through the generation of simplicial complexes. The advantage in saving computational time by choosing a particular algorithm depends on point cloud characteristics.

Figure 9 shows that at high point cloud sizes, GUDHI's alpha complex outperforms Ripser. Theoretically, alpha complexes gain polynomial run time complexity as the number of points increases, whereas Vietoris-Rips complexes gain exponential run time complexity (Otter et al., 2017). Specifically, alpha complexes are $O(n^{d/2})$, and Vietoris-Rips complexes are $O(2^n)$, where n is the number of points on a point cloud and d is the dimensions on the point cloud. For the conditions in our paper, Vietoris-Rips and alpha complexes both performed better than their theoretical maximums. Vietoris-Rips complex calculations consistently had a polynomial growth for both runtime and memory, while alpha complexes had linear runtime growth.

Based on the theoretical complexity and our results, alpha complexes are superior for point clouds with 3 or fewer dimensions. This advantage becomes especially clear at a high number of points. This difference in performance is clear in both runtime and memory use. Interestingly, while alpha complexes had overall less memory use, the memory use varied depending on the shape. Alpha complexes seem to require more memory for noisier data sets such as the annulus when compared to

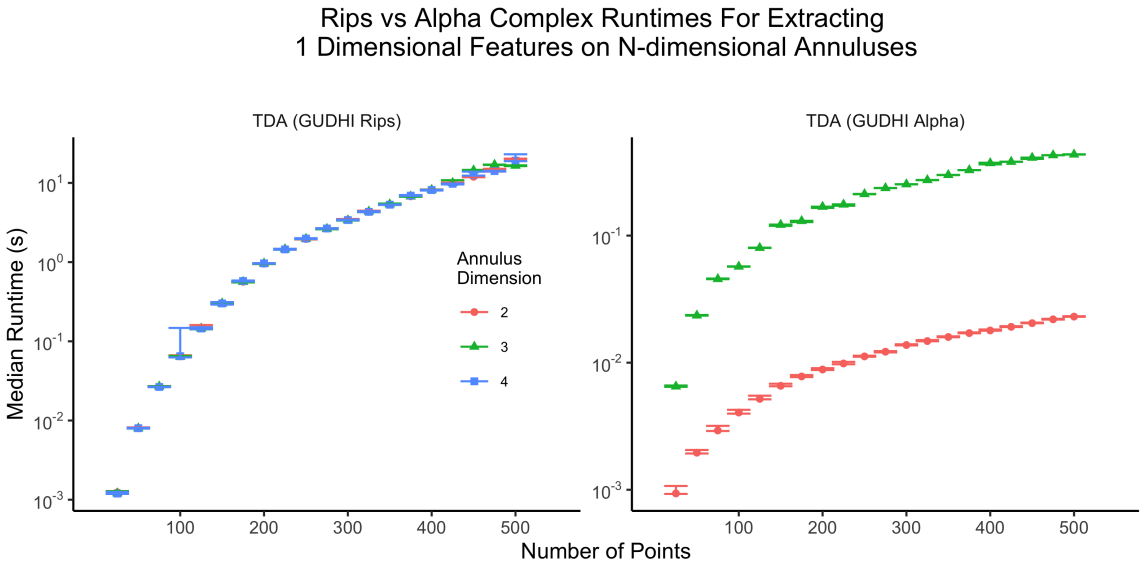

Figure 6: Comparing persistent homology calculation between Vietoris-Rips and alpha complices. Median runtime (min to max, $n = 10$ iterations per data point) for various data dimensions (denoted by color) are plotted against point cloud size and faceted by type of simplicial complex. Maximum of feature dimension was kept constant at 1. Alpha complex runtimes are linear, in contrast to polynomial Vietoris-Rips runtimes (see Supplement for regression details).

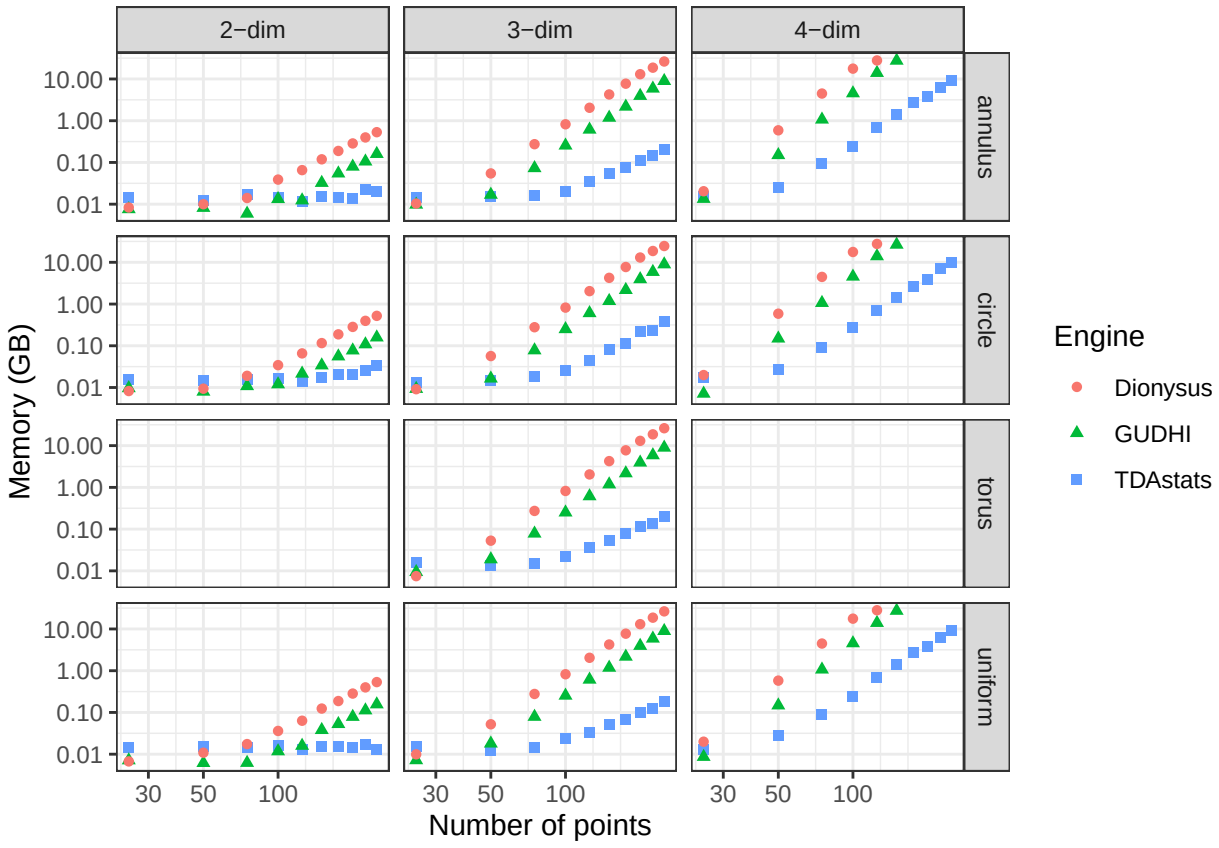

Figure 7: Comparing memory use of Vietoris-Rips persistent homology engines. Each column title corresponds to point cloud dimension; each row title lists point cloud shape; each persistent homology engine is represented by points of a distinct shape and color. For point clouds containing more than 50 points, there appears to be a linear trend on the log-log axes. Data for 2- and 4-dimensional tori were not collected because a torus does not trivially generalize to dimensions other than 3. Missing points for GUDHI and Dionysus in the 4-dim plots indicate that persistent homology calculation was terminated since memory requirement exceeded available RAM (32 GB).

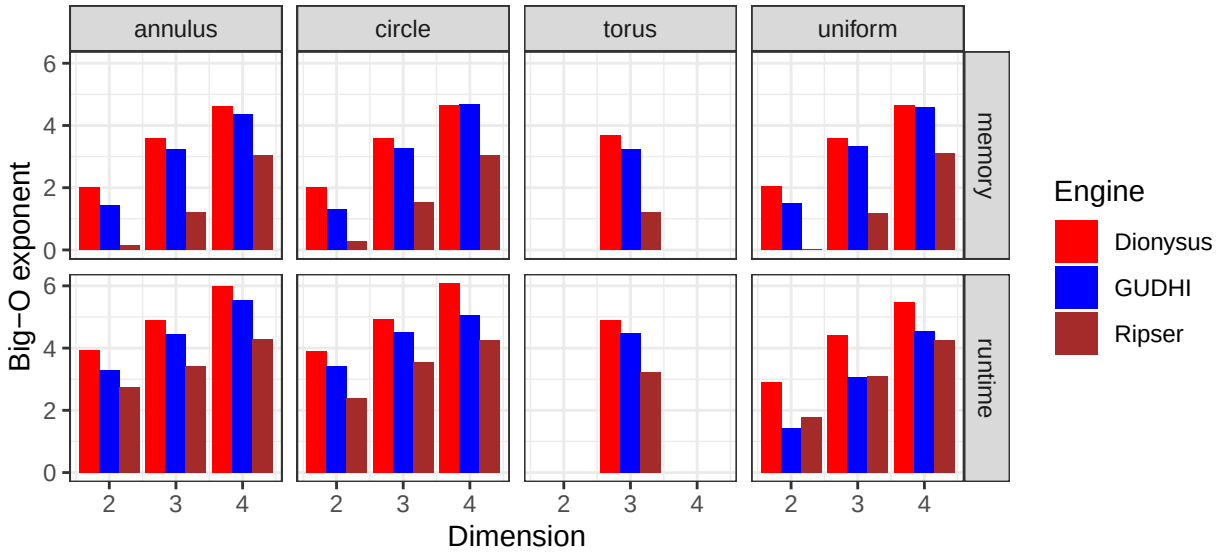

Figure 8: Big-O exponents for runtime and memory complexity as point cloud size varies. In the majority of cases, Ripser (and consequently, TDAstats) has the lowest exponent, indicating the slowest growth in complexity as point cloud size increases. Due to shape constraints, torus only has data available in the third dimension.

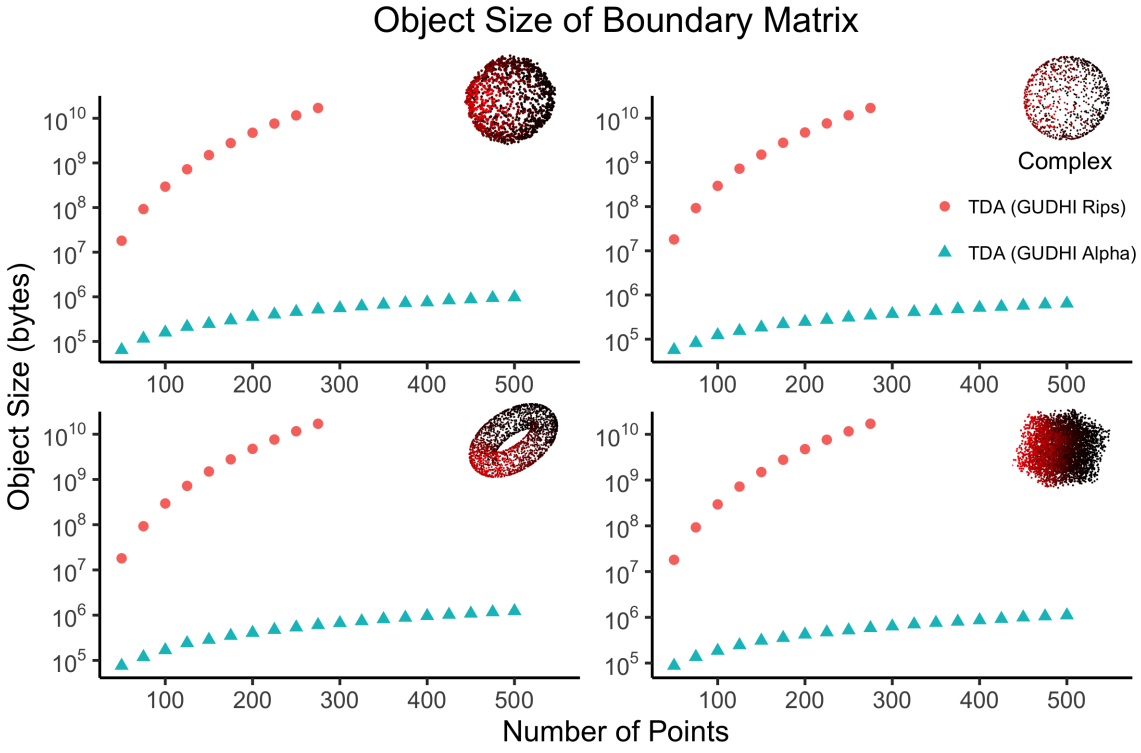

Figure 9: Runtime comparison of persistent homology calculation between Ripser's Vietoris-Rips and GUDHI's alpha complex functionality. Median runtime (min to max $n = 10$ iterations per data point) for various 3-dimensional point cloud structures (facet) plotted against point cloud size for each library (color). Benchmarking was conducted on an annulus (top-left), a sphere (top-right), a torus (bottom-left), and a cube (bottom-right). Data was not collected for data dimensions greater than 3 due to computational limitations of calculating alpha complices.

the sphere.

However, without sufficient computational resources, alpha complexes were not usable for point cloud dimensions greater than 3. If a point cloud has more than 3 dimensions, then it could undergo pre-processing with dimension reduction before using alpha complexes. Note, it is possible an algorithm will eventually be developed to enable alpha complex computation of higher-dimensional data. However, if data dimension cannot be reduced without significant information loss, then Vietoris-Rips complexes should be used. It should be stated that if a point cloud is compatible with both complexes, both analyses should be performed as there may be a variation in the persistent homology matrix. This is because alpha complexes satisfy the Nerve Theorem (Edelsbrunner and Mücke, 1994), which implies that they are topologically equivalent to the true underlying topology of the dataset; in contrast, Vietoris-Rips complexes only approximate the underlying topology (Hausmann, 1996). Among the tested Vietoris-Rips engines, Ripser (wrapped by **TDAstats**) has the fastest calculation time. GUDHI and Dionysus (wrapped by **TDA**) significantly fall behind as feature dimension and number of points increase.

On average, Ripser computed the persistent homology of a Vietoris-Rips complex with less memory than either GUDHI or Dionysus. Thus, when efficiency is critical, useRs would likely find **TDAstats** the appropriate library. However, **TDA** contains an entire library of features not available in **TDAstats**. Specifically, **TDA** allows kNN density estimation, kernel density estimation, density-based clustering, and dendrogram visualization, among other useful functionality. When computational resources are plenty, when point clouds are small and low-dimensional, or when the aforementioned functionality is required, **TDA** will likely be more appropriate than **TDAstats**. Both packages are hosted on CRAN.

While Vietoris-Rips complexes can handle high-dimensional data well, the calculation still significantly slows down when looking for higher dimension features. This is evidenced by the big-O polynomial growth for runtime and memory that have degree less than 4 for most 2-dimensional point clouds, but degree between 4 and 6 for most 4-dimensional point clouds (Figure 8); even higher degree complexities should be expected as point cloud dimension increases. Thus, finding high-dimensional topological features in high-dimensional point clouds remains a challenge. Methods to calculate persistent homology do exist for other simplicial complexes, such as the Delaunay complex and the Witness complex, but, to our knowledge, they are not currently implemented in CRAN packages. Future challenges would be creating and implementing algorithms that reduce the computational complexity of higher-dimensional topological feature calculations for R.

Acknowledgments

The authors thank Emily Dolson, PhD for assistance with using a high-performance computing node for data collection. JGS was supported by NIH grant R37 CA244613.

Bibliography

U. Bauer. Ripser: efficient computation of vietoris-rips persistence barcodes. *arXiv*, math.AT (1908.02518), 2019. [p184, 186]

J. C. Brunson, R. R. Wadhwa, and J. G. Scott. *ggtda: ggplot2-Compatible Visualization of Persistent Homology*, 2020. URL https://github.com/rrrlw/ggtda. R package version 0.1.0. [p186]

J. M. Chan, G. Carlsson, and R. Rabadan. Topology of viral evolution. *Proceedings of the National Academy of Sciences*, 110(46):18566—-18571, 2013. doi: 10.1073/pnas.1313480110. [p184]

H. Edelsbrunner and J. Harer. Persistent homology — a survey. *Discrete & Computational Geometry*, 453, 1 2008. doi: 10.1090/conm/453/08802. [p184]

H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13 (1):43–72, Jan 1994. doi: 10.1145/174462.156635. [p185, 191]

B. T. Fasy, J. Kim, F. Lecci, C. Maria, D. L. Millman, and V. Rouvreau. *TDA: Statistical Tools for Topological Data Analysis*, 2019. URL https://CRAN.R-project.org/package=TDA. R package version 1.6.5. [p184, 186]

J.-C. Hausmann. On the vietoris-rips complexes and a cohomology theory for metric spaces. In F. Quinn, editor, *Prospects in Topology (AM-138): Proceedings of a Conference in Honor of William Browder*, pages 175–188. Princeton University Press, Princeton, NJ, 1996. doi: 10.1515/9781400882588-013. [p184, 191]

J. Hester. *bench: High Precision Timing of R Expressions*, 2019. URL https://CRAN.R-project.org/package=bench. R package version 1.0.4. [p186]

U. Ligges and M. Mächler. Scatterplot3d - an r package for visualizing multivariate data. *Journal of Statistical Software*, 8(11):1–20, 2003. URL http://www.jstatsoft.org. [p186]

C. Maria, P. Dlotko, V. Rouvreau, and M. Glisse. Rips complex. In *GUDHI User and Reference Manual*. GUDHI Editorial Board, 2016. URL http://gudhi.gforge.inria.fr/doc/latest/. [p184, 186]

D. Morozov. *Dionysus*, 2018. URL http://mrzv.org/software/dionysus2. [p184, 186]

M. Offroy and L. Duponchel. Topological data analysis: A promising big data exploration tool in biology, analytical chemistry and physical chemistry. *Analytica Chimica Acta*, 910:1–11, 2016. doi: 10.1016/j.aca.2015.12.037. [p184]

J. Ooms. *magick: Advanced Graphics and Image-Processing in R*, 2019. URL https://CRAN.R-project.org/package=magick. R package version 2.2. [p186]

N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6(1), Sep 2017. doi: 10.1140/epjds/s13688-017-0109-5. [p184, 186, 188]

C. Schmid and B. Serbe. *recexcavAAR: 3D Reconstruction of Archaeological Excavations*, 2017. URL https://CRAN.R-project.org/package=recexcavAAR. R package version 0.3.0. [p186]

R. Turner. *deldir: Delaunay Triangulation and Dirichlet (Voronoi) Tessellation*, 2020. URL https://CRAN.R-project.org/package=deldir. R package version 0.1-25. [p186]

R. R. Wadhwa, D. F. Williamson, A. Dhawan, and J. G. Scott. TDAstats: R pipeline for computing persistent homology in topological data analysis. *Journal of Open Source Software*, 3(28):860, 2018. doi: 10.21105/joss.00860. URL https://doi.org/10.21105/joss.00860. [p184, 186]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p186]

H. Wickham. *pryr: Tools for Computing on the Language*, 2018. URL https://CRAN.R-project.org/package=pryr. R package version 0.1.4. [p186]

H. Wickham, J. Hester, and R. Francois. *readr: Read Rectangular Text Data*, 2018. URL https://CRAN.R-project.org/package=readr. R package version 1.3.1. [p186]

Eashwar V. Somasundaram
School of Medicine
Case Western Reserve University
Cleveland, OH 44106, United States

eashwar.somasundaram@case.edu

Shael E. Brown
Department of Quantitative Life Sciences
McGill University
Montreal, QC H3A 0G4, Canada

shael.brown@mail.mcgill.ca

Adam Litzler
Department of Translational Hematology and Oncology Research
Cleveland Clinic Foundation
Cleveland, OH 44195, United States

adli4611@colorado.edu

Jacob G. Scott
Department of Translational Hematology and Oncology Research
Cleveland Clinic Foundation
Cleveland, OH 44195, United States

ScottJ10@ccf.org

Raoul R. Wadhwa
Cleveland Clinic Lerner College of Medicine
Case Western Reserve University
Cleveland, OH 44195, United States

raoul.wadhwa@case.edu