5-2024

Plumbing the Depths of the Shallow End: Exploring Persistent Homology Using Small Data

R. Anne Flynn
Northern Michigan University, rflynn@nmu.edu

PLUMBING THE DEPTHS OF THE SHALLOW END: EXPLORING PERSISTENT HOMOLOGY USING SMALL DATA

By

R. Anne Flynn

THESIS

Submitted to
Northern Michigan University
In partial fulfillment of the requirements
For the degree of

MASTER OF SCIENCE

College of Graduate Studies and Research

May 2024

SIGNATURE APPROVAL FORM

PLUMBING THE DEPTHS OF THE SHALLOW END:
EXPLORING PERSISTENT HOMOLOGY USING SMALL DATA

This thesis by R. Anne Flynn is recommended for approval by the student's Thesis Committee, the Department Head of the Department of Mathematics and Computer Science, and the Dean of Graduate Education and Research.

_____ Date

Committee Chair: Dr. Joshua J. Thompson, Assistant Professor

_____ Date

First Reader: Dr. Daniel Rowe, Assistant Professor

_____ Date

Second Reader: Dr. J.D. Phillips, Professor

_____ Date

Department Head: Dr. J.D. Phillips

_____ Date

Dean of Graduate Education and Research: Dr. Lisa Eckert

ABSTRACT

PLUMBING THE DEPTHS OF THE SHALLOW END:
EXPLORING PERSISTENT HOMOLOGY USING SMALL DATA

By

R. Anne Flynn

Persistent homology is a prominent tool in topological data analysis. This thesis is designed to be an introduction and guide to a beginner in persistent homology. This comprehensive overview discusses the math used behind it, the code needed to apply it, and its current place in the field. We explain and demonstrate the algebraic topology which fuels persistent homology. Homotopies inspire homology groups, which are able to determine how many holes a shape has. By visualizing data as a shape, persistent homology determines what type of holes are present.

We demonstrate this by using the package `TDA` in the manipulation software R on controlled datasets. A kernel density estimate diagram presents the results. We showcase applying `TDA` to an external, uncontrolled dataset. The limits on memory allowed us to process no more than four columns of data at a time. To more thoroughly explore the dataset, we analyzed several four-column subsets, but found no special features aside from a base level of closeness.

DEDICATION

To my ever supportive husband Matt, without whom this never would have happened. Your insight and imagination consistently expand mine.

And to the Kalamazoo Area Mathematics and Science Center, who instilled in me a great love of mathematics and its cosmic beauty.

ACKNOWLEDGMENTS

The author would like to thank Dr. Joshua Thompson for his steady guidance and instruction. She also acknowledges Dr. Randy Appleton and Jason Haskell for their assistance and the university's Department of Mathematics for its passion of its subject.

All citations are formatted in Chicago style.

TABLE OF CONTENTS

LIST OF FIGURES

SYMBOLS AND ABBREVIATIONS

TDA	topological data analysis
$\mathbb{R}^n$	n-dimensional real space
$\mathbb{Z}^n$	n-dimensional space restricted to integers
$\mathbb{S}^n$	n-dimensional spherical space
$\pi_n(X)$	nth homotopy group of space X
Δ_n	maximum dimension of a simplex
C_n	n-dimensional chain group
δ	boundary function
H_n	n-dimensional homology group
F_iK	ith subcomplex in a filtration of complex K
ε	distance parameter
KDE	kernel density estimator
R	data manipulation software
`TDA`	a package for R
`h`	smoothing parameter
`B`	iteration parameter
`by`	step parameter
`Y`	dataset

1 Introduction

Big Data is here, exploding into the industrial scene and bringing with it the new field of data science. As with most infrastructures, mathematics has something to offer to improve it. Topological data analysis (TDA) elevates data science by marrying abstract math with computer science to find hidden insights. The goal of this thesis is to serve as a sweeping introduction to TDA by exploring a popular tool called persistent homology.

A student with any background in topology and group theory should be well-enough equipped to read this paper, and some familiarity with the program R and its IDE R Studio would be helpful. We cover every other necessary concept in the first few sections. By the time we reach topological data analysis, the reader should be prepared to see those concepts translated into an actionable program.

We begin with the topics one finds on a first swim into algebraic topology, namely homotopy in section two and homology in section three. These bridge the abstract with the concrete and allow topology to be applied to data. Special care is taken toward the beginner, with several examples presented to help these abstract concepts be more intuitively understood. From homotopy groups we proceed to triangulation, boundary, chain groups, and finally homology groups.

Section four takes a slight tangent to briefly discuss TDA as a field. This is meant to be an overview and we briefly introduce a variety of tools and forms of presenting their findings. This gives some context to both the usefulness and approachability of persistent homology. Within

persistence we focus on a method of presentation called the kernel density estimate diagram. The aim of section five is to make reading those diagrams more comfortable, as decoding their results is not as straight-forward as other traditional infographics. We spend some time demonstrating the effects of different parameters in a persistence algorithm before analyzing anything unexpected.

The penultimate section is dedicated to observing persistence's performance on both controlled and real world low-dimensional data. This section contains a collection of warnings and advice, as the current limits of persistence stalled many attempts to analyze the real data. To finish, we consider the possibilities of this sort of technology moving forward. It is a rare gift to be able to explore a new field with a new tool, and the rush of discovery sends readers off to their own adventures in math, data science, and beyond.

Without further ado, let us begin. In order to breach the waters of persistent homology, one must first ground their feet in the sands of algebraic topology. Many mathmaticians encounter point-set topology first: they relearn the meanings of open, continuous, connected, compactness, and so forth. It builds to a natural belief in the concrete structure of what it means to be a point in a set, with other points with which to interact. It eases a student into topology by seeing known concepts in new ways.

Algebraic topology, on the other hand, leaps in from the other direction. It uses abstract algebraic notions to describe topology, flying through concrete examples with barely a touch down to earth. Without an implicit understanding of a group's structure, of homomorphisms, of a proper map, algebraic topology seems almost alien. And yet, it describes exactly the same things point-set topology does. Why use it?

Point-set topology is useful, but doesn't distinguish holes in objects. Consider a line

segment and a circle; algebraic topology is able to discern the hole enclosed by the circle, and thus distinguish the objects as definitively different. Point-set topology is unable to find the hole, so to speak. It can discern the two objects are distinct, as a mapping from one to the other is not continuous across all its subsets, but it cannot tell *how*.

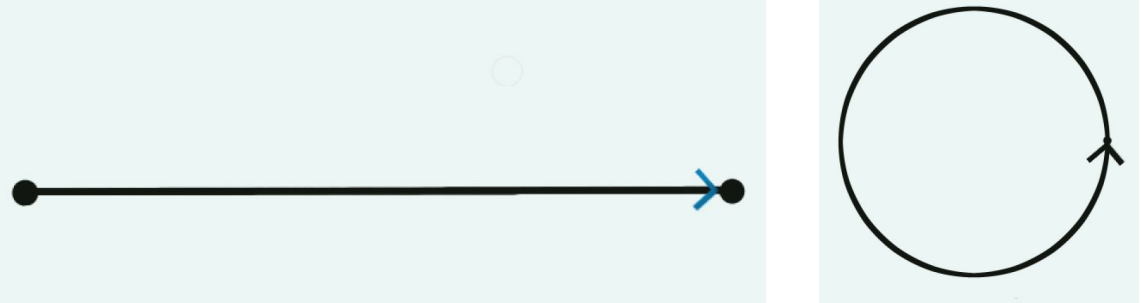

Figure 1: Point-set topology acknowledges a difference in these figures, but not the difference in their number of holes.

Historically, holes have great significance in applying topology, and a significant gap in data of any sort is worth exploring. Consider the bridges of Konigsberg, which helped bring about applied topology as a field. This famous puzzle's solution involves seeing a city as nothing more than a cluster of holes whose edges are available paths to travel. In fact, the entire purpose of persistence is tracking these holes and their strength of presence. Therefore, with its naturally superior hole detection, algebraic topology is the better choice when it comes to data analysis. We see this in homotopies and homologies, the necessary shallows we wade through toward TDA itself.

Homotopies, or homotopy groups, track paths through a shape. These homotopies are the conceptual basis for persistent homology; a cyclic homotopy which encloses nothing has found a hole. As we move up through dimensions, the homotopies in each reveal topologically significant voids. Homotopy groups are exceptionally difficult to compute in higher dimensions

though, so we must look elsewhere for our way forward.

For that, we turn to homology groups. These still find all the desired topological invariants, but are no longer reliant on a visual representation. They are the bridge between the abstract and the application. Even better, they can be algorithmically calculated. They are not quite as comprehensive as homotopies, but it is a necessary trade for progress to continue.

Finally, these deep waters of data can be navigated with topology. If a data set is viewed as a matrix, one can consider each column as its own free dimension. Thus any dataframe, no matter how large, can be represented as a shape inhabiting that many dimensions.

While the original goal of processing a fifty column dataset is impossible at this stage of persistence's development, the experience gained from the attempt is invaluable. We discovered an exponential increase in memory and processing power as one increases the columns of the dataset. Before we get too far ahead of ourselves, however, we must learn the math by beginning with some pure topology.

2 Homotopy

Now that we have decided upon algebraic topology, let us dig in the sand. The first topic to confront is that of homotopy. Put simply, homotopy is a set of paths with the same endpoints which can all be related to each other via some continuous deformation. Like a school of fish, a homotopy of paths is a group noun. The formal definition follows.[1]

A **homotopy** of paths in a topological space X is a family $f_t : I \to X$, $0 \leq t \leq 1$, such that

1) The endpoints $f_t(0) = x_0$ and $f_t(1) = x_1$ are independent of t.

2) The associated map $F : I \times I \to X$ defined by $F(s,t) = f_t(s)$ is continuous.

(It should be noted the term homotopy can also refer to the map which takes one path to another homotopic path, but in this paper the previous definition is primarily used.)

With some intuition about the 'rubber-sheet geometry' behavior of topology, one realizes homotopic paths are topologically equivalent, and so we can consider the entire homotopy as the important object to consider.

2.1 The Fundamental Group

The most notable homotopy group is the first one, denoted $\pi_1(X)$ and known as the fundamental group. It considers all one-dimensional paths on a surface which start and end at the same chosen point, classifying them based on which unavoidable holes they encircle. Figure 2 displays some surfaces with different fundamental groups, reflected in their variety and number of holes. Formally, the **fundamental group** $\pi_1(X, x_0)$ of a topological space X is the set of all homotopy classes $[f]$ of loops $f : I \leftarrow X$ at the basepoint x_0.[2]

1. Allen Hatcher, *Algebraic Topology* (New York, New York: Cambridge University Press, 2002).
2. Hatcher.

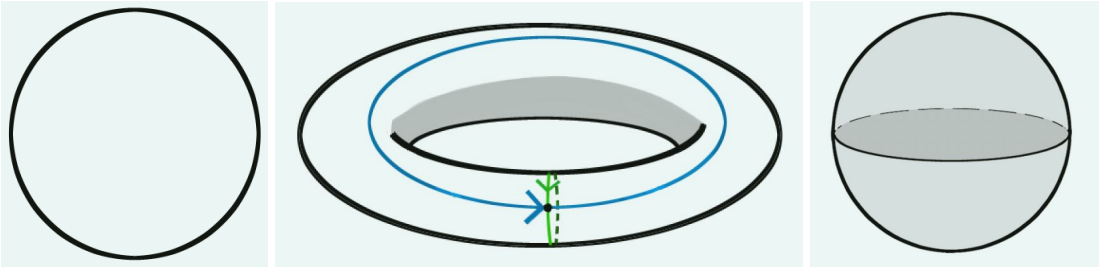

Figure 2: There is one hole in the circle, two in the torus, and zero in the sphere.

It can be daunting to calculate one's first fundamental group, no matter how simple the figure is. It may help to consider a path as a length of rope, and the point of origin as a peg to which it is anchored. The rope can wind and wander around the surface in whatever way one pleases, as long as it finishes at the original point again. Whatever path one draws can be pulled taught after it has finished. Pulling all the slack out of the rope will contract the path to the most generalized path which is homotopic to it.

Let us consider something approachable first, the fundamental group of a circle. It can be expressed as the group generated by a single element, 1, since all paths either complete a certain number of cycles around the circle ($n \cdot 1$ for some $n \in \mathbb{Z}^+$), complete less than a full cycle and can be contracted to the point (0), or complete a certain number of cycles in the opposite orientation ($n \cdot 1$ for some $n \in \mathbb{Z}^-$). Try to find a different kind of path, and one will see it can be contracted to one of these three.

For example, consider a path going one time clockwise about the circle, then three the opposite way (Figure 3). Let clockwise be the positive orientation in this case. We then pull on the rope until no more slack remains. The counter-clockwise cycles will undo all of the clockwise ones and more, ending back at the beginning point. The path will contract to twice about the circle backward and thus, this path is homotopic to the element -2 in the fundamental

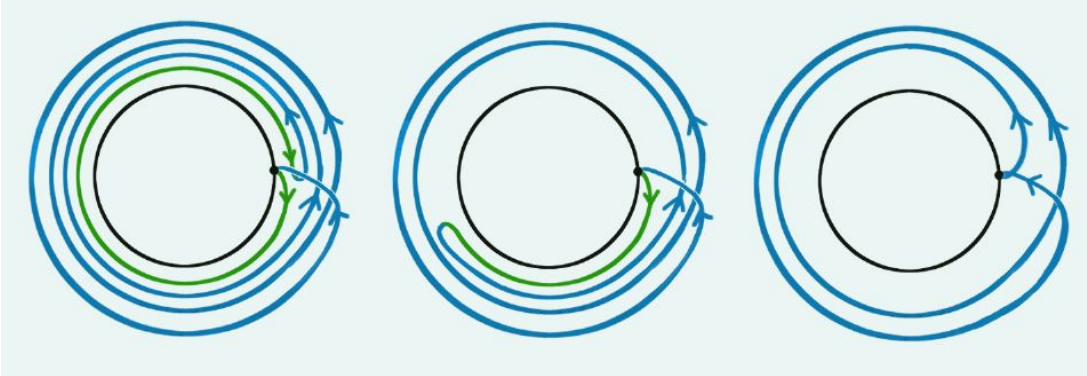

Figure 3: If a path includes a segment and its inverse, those cancel each other.

group.

An annulus has the same fundamental group, since it can itself be path-contracted to a circle (compare the surfaces in Figure 4). Even wandering paths will be pulled tight along the inside circle. This gives rise to another insight.

Theorem 1 *If two spaces are topologically equivalent, their homotopy groups are also equivalent.*

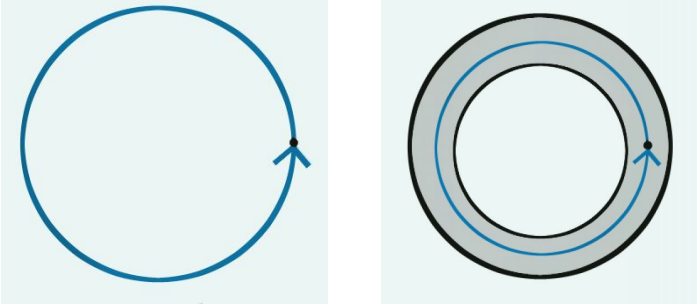

Figure 4: $\pi_1(X)$ of a circle and an annulus, topologically equivalent.

A mobius strip can also be contracted to a circle and so has the same fundamental group. Figure 5 shows how the circle's generator runs directly through the Mobius strip, and one can path-contract it to that circle to show topological equivalence. A disk is even less interesting:

since *every* path can be contracted to a point, the fundamental group is 0.

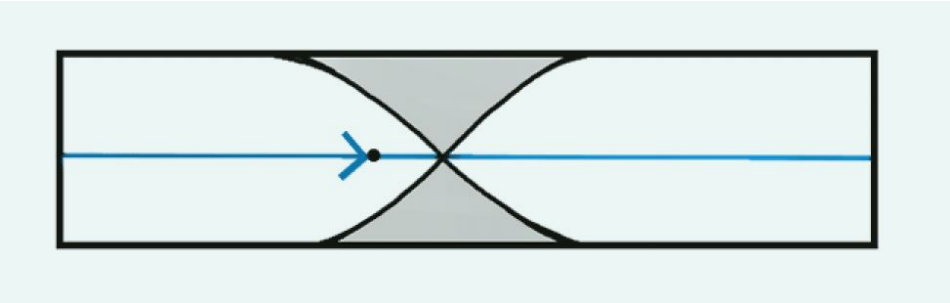

Figure 5: $\pi_1(X)$ of a mobius strip, viewed directly onward.

We move on to a more interesting space: a wedge of two circles. Since a cycle round each circle cannot be homotopied into a cycle about the other, its fundamental group is generated by two elements, denoted by the differently colored paths (Figure 6). One might write this as $\pi_1(X) = \langle a, b \rangle$, calling the blue path a and the green path b. Another way to see this figure is as the intersection of two circles at a single point, and so deduce its fundamental group is the product of the fundamental group of two circles. This is an example of the Van Kampen theorem in action as we use it to decompose a surface into simpler shapes, using their fundamental groups to build that of the original surface. Hatcher's version of the theorem follows.[3]

Theorem 2 (Van Kampen's Theorem) *If X is the union of path-connected open sets A_α each containing the basepoint $x_0 \in X$ and if each intersection $A_\alpha \cap A_\beta$ is path-connected, then the homomorphism $\Phi : \star_\alpha \pi_1(A_\alpha) \to \pi_1(X)$ is surjective. If in addition each intersection $A_\alpha \cap A_\beta \cap A_\gamma$ is path-connected, then the kernel of Φ is the normal subgroup N generated by all elements of the form $i_{\alpha\beta}(\omega) i_{\beta\alpha}(\omega)^{-1}$, and so Φ induces an isomorphism $\pi_1(X) \approx \star_\alpha \pi_1(A_\alpha)/N$.*

3. Hatcher, *Algebraic Topology*.

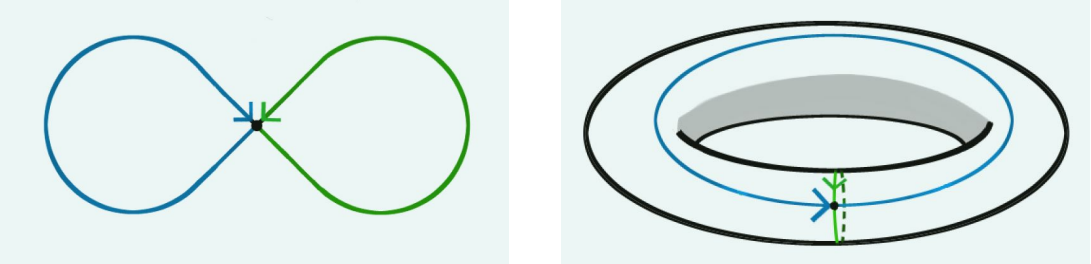

Figure 6: The wedge and the torus share generators in $\pi_1(X)$, but they are topologically different.

Like a wedge, a torus is also generated by two elements, but it has a group relation $aba^{-1}b^{-1} = 1$. This relation comes from its two-dimensional surface. One can form a wedge by identifying opposite edges of a hollow square. If the square is thought to span a topological disc, this identification yields a torus (see Figure 13). Here we see the converse of Theorem 1 is not always true. Sharing the set of fundamental group generators does not confer equivalence of shape. A torus does not embed in $\mathbb{R}^2$ and has two unique features: one tunnel through the center and one 'bubble' within its walls. The wedge of two circles can be embedded in $\mathbb{R}^2$ and has two tunnels, a stark difference from a topological perspective.

A two-holed torus takes things to another level, literally. Its fundamental group is generated by four elements due to the two unique ways one could walk around each hole in the torus (Figure 7). If one assigns each of the four unique loops a letter a, b, c, d, the fundamental group can formally written as $\pi_1(X) = \langle a, b, c, d | aba^{-1}b^{-1}cdc^{-1}d^{-1} = 1 \rangle$. A common question many have at this point is, "How would one represent the path going between the 'donut holes'?" If we call the blue homotopy a and the green one b, their composition is topologically equivalent to the path in question. We invite the reader to generate any path they like across this figure, then determine to which homotopy it belongs.

As a final example, we consider the tetrahedron. Although the stately tetrahedron is, like

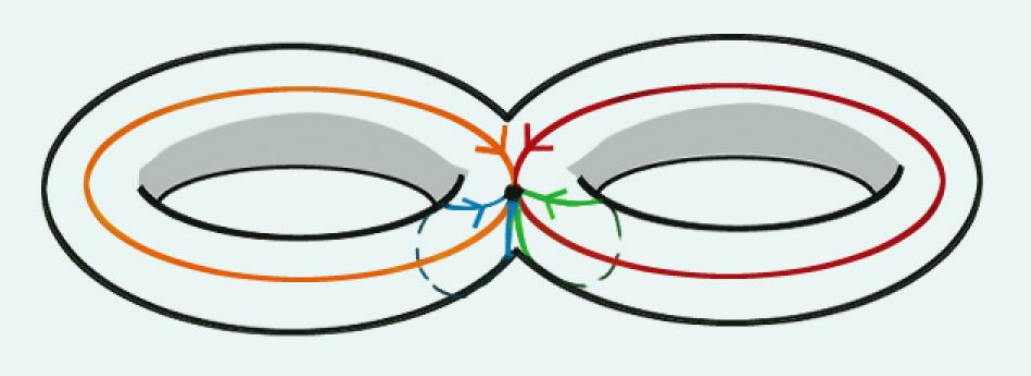

Figure 7: $\pi_1(X)$ of a two-holed torus.

the torus, embedded in $\mathbb{R}^3$, it is isomorphic to the sphere $\mathbb{S}^2$. Any path about $\mathbb{S}^2$ is contractible to a point, as it is a closed, connected two-dimensional surface embedded in a three-dimensional space. It has neither obstacle nor boundary to run into anywhere. Therefore the fundamental group of a tetrahedron is the same as a sphere, simply 0.

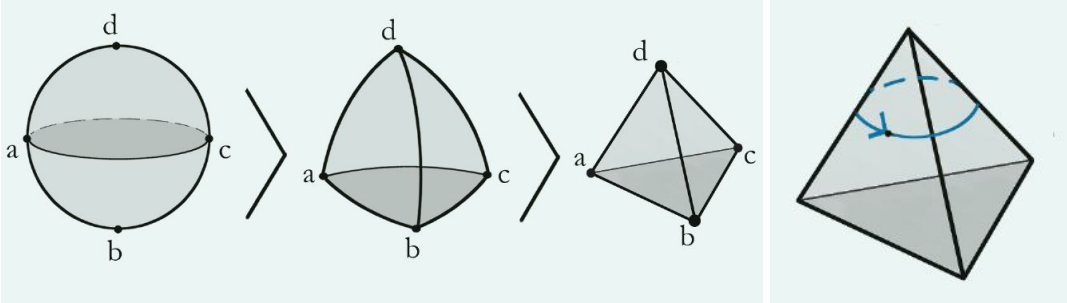

Figure 8: $\pi_1(X)$ of a tetrahedron, topologically equivalent to a sphere.

2.2 Larger Dimensions

The fundamental group, while powerful, does have limitations. It is not inherently abelian, as the orientation of its paths and the order of composition both help generate all the group's elements. Homotopy groups *are* commutative past a certain point– when $n \geq 2$, $\pi_n(X)$ will always be abelian[4]– but this is only useful if we can somehow observe or compute those higher

4. Hatcher, *Algebraic Topology*.

homotopy groups.

Therein lies the biggest problem for using these in application. While in theory we could draw significant insights from a surface's full span of homotopy groups, even computers have a difficult time determining higher-level homotopy groups. Recent theories of how to improve their performance have been released as recently as 2022.[5] Van Kampen's Theorem only applies to the fundamental group, and other such hopeful resources are similarly rousted.

Although we might like to utilize these powerful tools, higher homotopy groups aren't feasible for any sort of high-dimensional surface. In fact, it is infeasible to calculate them directly past our own relatable space of $\mathbb{R}^3$ for anything beyond the simplest of shapes.[6] To further convince any skeptics, Figure 9 contains the homotopy groups of a very simple space, $\mathbb{S}^n$, up to $\mathbb{S}^8$ ($\pi_i(\mathbb{S}^n)$).[7]

Homotopies hold the key to our understanding, but we find them mostly inaccessible. To apply this knowledge to something like data, which comes in notoriously large arrays sometimes, we must find a way to look into those larger spaces without needing diagrams to get by. Algorithms are capable of finding peripheral, oft directly related information like topological invariants, but not finding the homotopy groups themselves. In fact, identifying features in low dimensions affect the higher-dimensional homotopic invariants themselves.[8] For now, we must seek a different method of finding these voids.

5. Ronald Brown, "Modelling and Computing Homotopy Types: I," Hosted by Cornell University, *ArXiv*, September 2022,

6. Hiroshi Toda, *Composition Methods in Homotopy Groups of Spheres*, ISBN 0-691-09586-8 (Princeton University Press, 1962).

7. Toda.

8. Brown, "Modelling and Computing Homotopy Types: I."

	S^0	S^1	S^2	S^3	S^4	S^5	S^6	S^7	S^8
π_1	0	$\mathbb{Z}$	0	0	0	0	0	0	0
π_2	0	0	$\mathbb{Z}$	0	0	0	0	0	0
π_3	0	0	$\mathbb{Z}$	$\mathbb{Z}$	0	0	0	0	0
π_4	0	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$	0	0	0	0
π_5	0	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$	0	0	0
π_6	0	0	$\mathbb{Z}_{12}$	$\mathbb{Z}_{12}$	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$	0	0
π_7	0	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}\times\mathbb{Z}_{12}$	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$	0
π_8	0	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}_2^2$	$\mathbb{Z}_{24}$	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$
π_9	0	0	$\mathbb{Z}_3$	$\mathbb{Z}_3$	$\mathbb{Z}_2^2$	$\mathbb{Z}_2$	$\mathbb{Z}_{24}$	$\mathbb{Z}_2$	$\mathbb{Z}_2$
π_{10}	0	0	$\mathbb{Z}_{15}$	$\mathbb{Z}_{15}$	$\mathbb{Z}_{24}\times\mathbb{Z}_3$	$\mathbb{Z}_2$	0	$\mathbb{Z}_{24}$	$\mathbb{Z}_2$
π_{11}	0	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}_{15}$	$\mathbb{Z}_2$	$\mathbb{Z}$	0	$\mathbb{Z}_{24}$
π_{12}	0	0	$\mathbb{Z}_2^2$	$\mathbb{Z}_2^2$	$\mathbb{Z}_2$	$\mathbb{Z}_{30}$	$\mathbb{Z}_2$	0	0
π_{13}	0	0	$\mathbb{Z}_{12}\times\mathbb{Z}_2$	$\mathbb{Z}_{12}\times\mathbb{Z}_2$	$\mathbb{Z}_2^3$	$\mathbb{Z}_2$	$\mathbb{Z}_{60}$	$\mathbb{Z}_2$	0

Figure 9: Homotopy groups tabulated by Toda Hiroshi.

3 Homology

Homology groups are similar to homotopy groups in how they assess features like voids, although homology groups are a bit less comprehensive. They are far easier to compute for higher dimensions, despite requiring some setup.

It begins with the concept of triangulating a shape. Triangulating is when a surface in $\mathbb{R}^n$ is decomposed into a 'derendered' version of itself, made of a sum of triangles drawn between vertices sampled across the surface.[9] The $\mathbb{R}^0$ elements are vertices, the $\mathbb{R}^1$ elements are edges between those vertices, the $\mathbb{R}^2$ elements are triangles filled in between those vertices, and so on. The term **face** may refer to any dimension's elements (an n-face would live in $\mathbb{R}^n$), but is generically used to describe the triangles in $\mathbb{R}^2$.

With triangulation we have some direction on how to leave the dry sands of abstract homotopy behind. If the goal is to find holes, one can map out an entire surface by labeling its vertices and tracking paths one draws in this triangulation. It allows a topologist to play cartographer, in a sense. One could decompose a surface into any number of shapes, but triangles are the most simple shape and give the least complications moving forward. In addition, triangulation allows one to travel by linear combinations of the faces (classing all paths through these faces as topologically the same), so all coefficients are now in $\mathbb{Z}$.

9. Herbert Edelsbrunner and John Harer, *Computational Topology: An Introduction*, Departments of Computer Science and Mathematics (Durham, North Carolina: Duke University, 2010), 177–208.

3.1 Simplicial Homology

When a surface is triangulated, it can be algebraically described in the form of a **simplicial complex** X, also called a Δ-complex. A simplicial complex is a collection of sets, called simplices, which each contain this surface up to a particular dimension. Its elements are the n-dimensional faces, which are themselves simplices containing their lower-dimensional components. From the other direction, any face of a simplex Δ_n is also contained in all higher-dimensional simplices $\Delta_{m>n}$ which contain that simplex.[10] Consider Figure 10 below. Its simplicial complex would consist of $0, 1$ and 2-simplices, each containing the figure's n-faces represented as sets: the 0-simplex, or Δ_0, contains the vertices $[a, b, c, d]$; the 1-simplex Δ_1 contains the edges $[(ab), (bc), (ca), (ad), (db), (cd)]$; and the 2-simplex Δ_2 contains the faces $[(abc), (acd), (adb), (bdc)]$. Notice there is no 3-simplex, as this is a hollow polygon equivalent to $\mathbb{S}^2$ and there is no 3-face to speak of.

This notation has some readily available properties which lend themselves nicely to higher dimensions. Firstly, it is a nested structure by design. This is a purely combinatorial approach[11].[12] Even if dimensions cannot be visualized they are assuredly included and covered, since every higher dimensional simplex is built by using the shape's skeleton of vertices.

The Δ-complex also captures paths in expressions, not diagrams. This frees us from needing any sort of visualization to work with them. These algebraic records of our paths take the form of **chains**. Chains form groups in their respective dimensions denoted C_n, generated by the unique n-dimensional faces of the structure. To prove these are groups is left to the reader;

10. Larry Wasserman, *Topological Data Analysis*, Department of Statistics (Pittsburgh, Pennsylvania: Carnegie Mellon University, 2016).
11. Robert Ghrist, *Foundations of Topological Data Analysis*, YouTube, July 2023.
12. Hatcher, *Algebraic Topology*.

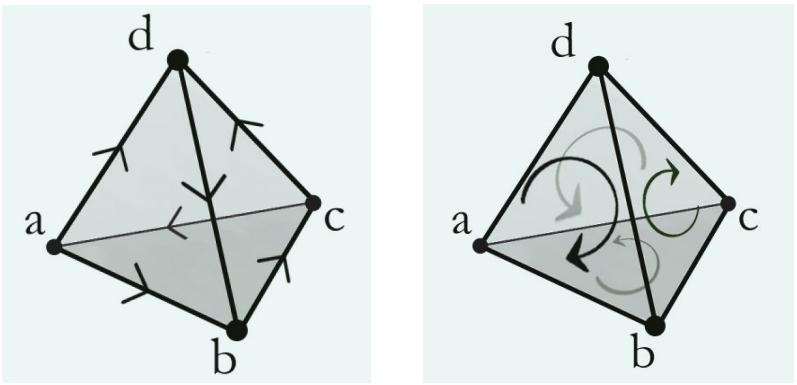

Figure 10: A hollow tetrahedron with orientation demonstrated on edges, then faces.

the identity is the 0 element, and it is clear to see associativity and inverses hold with a little wandering through the figure. To illustrate, our stolid tetrahedron assistant has its vertices as the basis of the C_0 group, its edges as the basis of the C_1 group, and its triangular faces as the basis for the C_2 group. (a,d) and $3(a,b)+(b,c)-2(c,d)+(d,b)$ are both 1-chains in C_1. Any n-chain in an n-simplex can be composed of any linear combination of the faces contained in that simplex. A chain does not have to end at the same point, nor does it have to follow any particular orientation.[13]

When a chain does end at its beginning point, it is called a cyclic chain, or **cycle**. $(a,d)+(d,b)+(b,a)$ is a cycle in Figure 10, while (a,d) alone is not. It is now time to fully reveal the plan: if we can somehow find cycles which walk around *nothing*, we will have found a hole.[14] How do we determine when nothing is inside a cycle? This finally leads us to the single most important function in our presentation of algebraic topology: the boundary function.

13. NJ Wildberger, *Algebraic Topology: a Beginner's Course*, YouTube, University of New South Wales, Sydney, Australia, 2012.

14. Ravi Jagadeesan and Luke Sciarappa, *Simplicial Homology*, MIT Mathematics, Fourth Annual MIT Primes Conference, May 2014.

3.2 The Boundary Function

The **boundary function** δ is a homomorphism $\delta_n : C_n(X) \to C_{n-1}(X)$ where if Δ_n is an n-simplex $\Delta_n = (v_0 v_1 ... \hat{v}_i ... v_n)$, then:

$$\delta_n(\Delta_n) = \Sigma_i (-1)^i (v_0 v_1 ... \hat{v}_i ... v_n) \tag{1}$$

In words, δ_n for a simplex in simplicial complex X is the set of all n-1 cycles found by taking the boundaries of each of its n-faces. These boundaries are an $(n-1)$ simplex created by taking the formal alternating sum of its face's edges. The elements are generated by omitting a different vertex v_i of the face at a time (denoted by the hat over v_i above). If one considers the 2-simplex (abc) in Figure 10, $\delta_2(abc) = bc - ac + ab$. The boundary $\delta_n(X)$ clearly makes a subgroup in the chain group $\mathrm{C}_{n-1}(X)$.

This function does require a little extra structure to operate smoothly, namely the orientation mentioned before. While a valid n-chain may be any linear combination of the elements of its simplex, the boundary function returns only cycles of one dimension lower. It clearly has an order imposed on the list of elements v_j above, which affects the sign based on whatever entry of ordered value i is currently removed. This is the orientation, determined by the vertices' order. Any order may be imposed on the vertices, as long as one holds to its orientation faithfully throughout the calculations. This is expanded on the right in Figure 10; all faces of the tetrahedron have been oriented clockwise when viewed from outside the figure, due to its vertices being ordered from first to last (a,b,c,d). A positively-oriented path, say from a to b, is simply represented as (ab), and going the opposite way is shown as either $-(ab)$ or (ba). This

showcases an ability to permute the order of entries: two adjacent entries in a boundary element may be swapped as long as its sign changes.

It is prudent to note some properties of the boundary function, but they will be brief; the best way to see how the boundary function works is to see it in action. The boundary of a cycle is 0,[15] as the formal sum will result in two of each $n-1$ face, one in each direction, and they will wholly cancel. This is demonstrated in (2) below. This leads to an extremely useful quick fact about the δ function: the boundary of a boundary is 0. If one wants the boundary δ_2 of a 2-dimensional face, the result is a cycle and the boundary of a cycle is 0. This is so important it bears extra mention.

Theorem 3 $\delta^2 = 0$*, i.e. the boundary of a boundary is 0.*

$$\delta_2(adb) = (db - ab + ad)$$

(δ is a homomorphism and we can extend by linearity.)

$$\begin{aligned} \delta_1(db-ab+ad) &= \delta_1(db) - \delta_1(ab) + \delta_1(ad) \\ &= (b-d) - (b-a) + (d-a) \\ &= b - d - b + a + d - a \\ &= 0 \end{aligned} \tag{2}$$

Additionally, the 0-simplex (the vertices) will always have a boundary result of 0. This is

15. Wildberger, *Algebraic Topology: a Beginner's Course*.

demonstrated directly in (3) with the tetrahedron. Let $\Delta_0 = a + b + c + d$ be the 0-chain.

$$
\begin{aligned}
\delta_0(\Delta_0) &= \delta_0(a) + \delta_0(b) + \delta_0(c) + \delta_0(d) \\
\delta_0(a) &= 0 \\
\delta_0(b) &= 0 \\
\delta_0(c) &= 0 \\
\delta_0(d) &= 0
\end{aligned}
\tag{3}
$$

This may seem cheap, but there is no formal sum to take for zero-dimensional elements. This concurs with the intuition one has about the boundary of the lowest dimensional elements in a set. If the boundary of a face is represented in elements one dimensional lower than it, the lowest dimensional elements will have nothing 'below' them (in Δ_{-1}, the trivial group) and so will have no representation. The entire structure is formed from building up from the vertices. The vertices themselves cannot have something which preceeds them, otherwise they would not be the progenitors of the structure. Therefore the boundary of the Δ_0 simplex is the identity 0.

Now we compute δ on other simplices of the tetrahedron. The one-dimensional chain group C_1 for the tetrahedron is generated by the six edges (ab), (bc), (ca), (ad), (db), and (cd).

$$\begin{aligned}
\delta_1(ab) &= b-a \\
\delta_1(bc) &= c-b \\
\delta_1(ca) &= a-c \\
\delta_1(ad) &= d-a \\
\delta_1(db) &= b-d \\
\delta_1(cd) &= d-c
\end{aligned} \tag{4}$$

These six relationships list out the boundaries of all edges in the tetrahedron and generate the subgroup of 1-dimensional boundaries for the entire surface. Any boundary of a collection of 2-dimensional faces can be expressed using these. Next, the boundary of the faces in Δ_2 can be found by $\delta(C_2)$, the second chain group generate by the four faces (abc), (acd), (adb), and (bdc). We compute δ_2 on these generators and collect the results here to be used later.

$$\begin{aligned}
\delta_2(abc) &= (bc)-(ac)+(ab) \\
\delta_2(acd) &= (cd)-(ad)+(ac) \\
\delta_2(adb) &= (db)-(ab)+(ad) \\
\delta_2(bdc) &= (dc)-(bc)+(bd)
\end{aligned} \tag{5}$$

It is of utmost importance to understand boundary before moving on to homology groups. The boundary of some face is the cycle of elements enclosing that face, and these elements are naturally embedded one dimension lower than the face they enclose. Thus, the image of a

particular simplex Δ_n's boundary will be the boundaries of all its existing faces. In addition, the boundary of all cycles always have a formal sum of zero. Their later steps undo their earlier ones. If we can gather all the cycles together and remove the ones which are boundaries of some area, the ones left over will be cycles which do not enclose area, i.e. holes. We accomplish this by taking a quotient of the group of cycles by the group of boundaries in a single dimension. We call these quotients homology groups.

3.3 Homology Groups

A homology group H_n is defined[16] as:

$$H_n = \ker(\delta_n)/\mathrm{im}(\delta_{n+1}) \tag{6}$$

H_n represents the nth homology group, made by taking the quotient of the boundary's kernel in one dimension by the image of boundary from the dimension above it. Recall a kernel of a function is the set of inputs which have the output 0, and the image is the set of all outputs of some function. By taking the kernel of the boundary in a dimension, it ensures all cycles are captured, as they are the chains whose boundaries will go to 0. The image of the dimension above it ensures all boundaries– and only boundaries, which enclose an area– are captured in the form of cycles in dimension n.

Recall the image of the $n+1$ boundary is a subgroup of the C_n chain group. Therefore, the quotient group created in this fashion considers all n-cycles except the ones which enclose an

16. Hatcher, *Algebraic Topology*.

area. The result will be those cycles which enclose nothing: holes. The easiest way to calculate these will be to consider the basis elements m of each group, written in terms of $\mathbb{Z}^m$; once we mod the kernel by the image, what remains will give us our number of voids.

Similar to the boundary function, homology groups are more digestible when taken with examples. For our final backstroke through homology in its own right, we now revisit our examples from Section 2 and instead compute their first homology groups. One will notice how the first homology groups are exactly the abelianized fundamental groups of their respective spaces. In fact, all homology groups are abelian. Neither direction nor starting point matter when the end result is a net distance of travel, measured in number of times a void is unavoidably encircled.

It seems almost insulting to work off the tetrahedron so much and *not* compute at least one homology group; for its excellent support, why not calculate all three? Call the tetrahedron X. The 0th homology group, H_0, is formed by taking the kernel of δ_0 and modding out by the image of δ_1. $\ker(\delta_0)$ contains all vertices a, b, c, and d. $\text{im}(\delta_1)$ contains the boundaries $(b-a)$, $(a-c)$, $(d-b)$, and $(d-a)$ of each edge.

Recall our goal here is to compare the number of basis elements between the kernel and the image. The $\text{im}(\delta_1)$ is algebraically redundant in this sense. The boundary of (bd), for example, is $(d-b)$. By taking the boundary of two other elements, $\delta_1(ad)-\delta_1(ab)$, we arrive at $d-a-b+a$, or $d-b$. Because $\delta(bd)$ can be made by a linear combination of the other elements, it is not a unique basis element. With some linear algebra, we see there are only three

generators needed for all of im(δ_2). This is reflected below in (7).

$$
\begin{aligned}
\mathrm{H}_0 &= \ker(\delta_0)/\mathrm{im}(\delta_1) \\
\ker(\delta_0) &= \langle a,b,c,d\rangle && \textit{(Express as } \mathbb{Z}^4\textit{.)} \\
\mathrm{im}(\delta_1) &= \langle b-a, a-c, d-a\rangle && \textit{(Express as } \mathbb{Z}^3\textit{.)} \\
\implies \mathrm{H}_0 &= \mathbb{Z}^4/\mathbb{Z}^3 \\
&= \mathbb{Z}
\end{aligned} \tag{7}
$$

The H_0 group has a single generator, indicating one can get to any point in the surface from this beginning. What this means in a broad sense is the surface is one connected piece.

Evidence of holes appears in the H_1 group. The kernel of δ_1 will be all 1-dimensional cycles in X, as all cycles go to 0. The basis of these cycles are the unique ones found in the edges of X: $ab+bc+ca$, $bc+cd+db$, $ad-cd+ca$, and $ad+db-ab$. The image of δ_2 is the boundary of the 2-faces abc, acd, adb, and bdc.

$$
\begin{aligned}
\mathrm{H}_1 &= \ker(\delta_1)/\mathrm{im}(\delta_2) \\
\ker(\delta_1) &= \langle ab+bc+ca, bc+cd+db, ad-cd+ca, ad+db-ab\rangle \\
&= \mathbb{Z}^4 \\
\mathrm{im}(\delta_2) &= \langle bc-ac+ab, cd-ad+ac, db-ab+ad, dc-bc+bd\rangle \\
&= \mathbb{Z}^4 \\
\implies \mathrm{H}_1 &= \mathbb{Z}^4/\mathbb{Z}^4 \\
&= 0
\end{aligned} \tag{8}
$$

Recall the fundamental group $\pi_1(X)$ of a sphere is also 0. This first homology group says the same, but what H_1 reveals is there are no 1-dimensional holes. This is easily confirmed by referencing the figure: the only void it has is the one enclosed by its entire self. To find this void, one must find H_2. The δ_2 will find one unique cycle of 2-faces: $abc + acd + adb + bdc$. The second homology group, having a single generator (as shown in (9) below), suggests there is a single 2-dimensional hole.

$$
\begin{aligned}
H_2 &= \ker(\delta_2)/\mathrm{im}(\delta_3) \\
\ker(\delta_2) &= \langle abc + acd + adb + bdc \rangle \\
&= \mathbb{Z} \\
\mathrm{im}(\delta_3) &= 0 \qquad\qquad \textit{(It has no 3-face.)} \\
\implies H_2 &= \mathbb{Z}/0 \\
&= \mathbb{Z}
\end{aligned}
\tag{9}
$$

There are no more non-zero homology groups for this shape. This is a final benefit of homologies as opposed to homotopy groups (refer back to Figure 10 to see non-trivial homotopies existing above a shape's inhabited dimensions). A homology group's necessary pieces are created from a face existing in some dimension. Thus, when a shape no longer has a presence at a sufficiently high dimension, it will not have any elements in the boundary's kernel, nor its image, and its homotopy group there will always be 0. This is the power of the boundary function. It allows a shape, once triangulated, to be broken down into its fills and voids– and nothing else. Before celebrating and swimming ahead into larger swells, however, it is useful to re-examine the homotopy examples and see them in this new light.

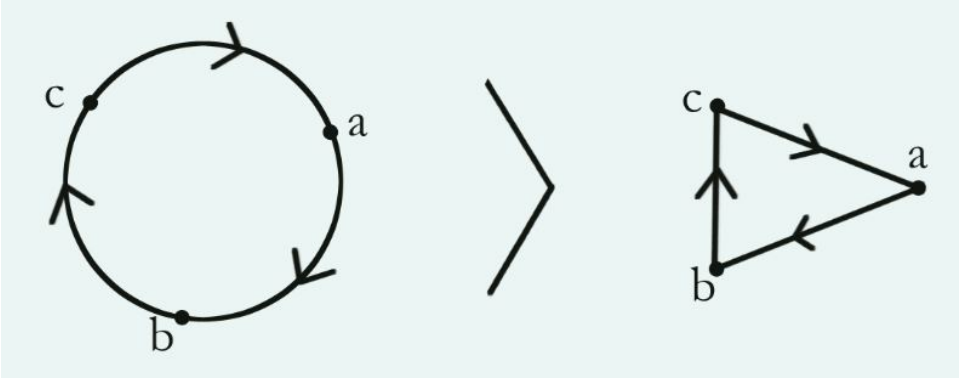

Figure 11: A triangulation of a circle S.

Revisiting the circle may feel elementary, but it is an excellent place to start and confirm any implicit expectations. If one remembers the circle's fundamental group, those expectations are likely correct. To triangulate the circle is a simple matter of noticing it is topologically equivalent to a hollow triangle S with vertices a, b, c and the edges between them (Figure 11). The kernel of δ_1 is all unique cycles created by edges: the circle itself. The image of δ_2 is empty, as there are no 2-faces to give a boundary. As displayed below, the first homology group of a circle ends up being $\mathbb{Z}$. In this case, it is the same as its fundamental group ($\mathbb{Z}$ is already abelian). This confirms what the figure itself presents: in a circle, the one-dimensional cycles contain one cycle which is not a boundary, i.e. the hole.

$$
\begin{aligned}
\mathrm{H}_1 &= \ker(\delta_1)/\mathrm{im}(\delta_2) \\
\ker(\delta_1) &= \langle ab + bc + ca \rangle \\
&= \mathbb{Z} \\
\mathrm{im}(\delta_2) &= 0 \\
\implies \mathrm{H}_1 &= \mathbb{Z}/0 \\
&= \mathbb{Z}
\end{aligned}
\tag{10}
$$

We also briefly reconsider the annulus and mobius band. Recall each of them is topologically equivalent to a circle. By Theorem 1, each of their first homology groups are also $\mathbb{Z}$. A disk is even less interesting: since every path can be contracted to a point, the first homology group is 0.

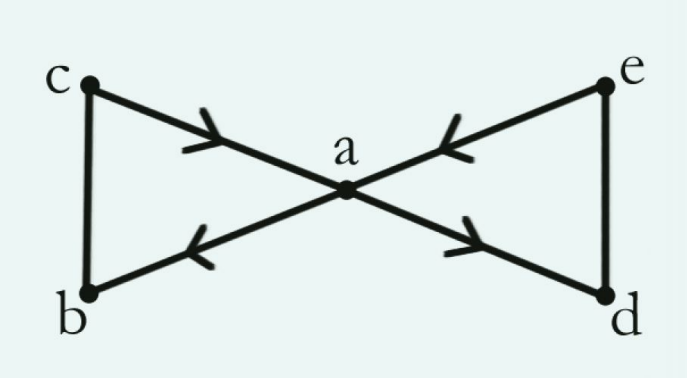

Figure 12: A triangulated wedge W.

With a wedge of two circles W, one can use its fundamental group to predict the first homology group. $\pi_1(W) = \mathbb{Z} \times \mathbb{Z}$, so $\mathrm{H}_1(W)$ is the abelianized version of that: $\mathbb{Z}^2$. In calculating it directly, we see $\ker(\delta_1(W))$ contains a pair of elements (the two unique 1-dimensional cycles in the shape) and $im(\delta_2(W))$ is, again, 0.

$$
\begin{aligned}
\mathrm{H}_1 &= \ker(\delta_1)/\mathrm{im}(\delta_2) \\
\ker(\delta_1) &= \langle ab+bc+ca, ad+de+ea \rangle \\
&= \mathbb{Z}^2 \\
\mathrm{im}(\delta_2) &= 0 \\
\implies \mathrm{H}_1 &= \mathbb{Z}^2/0 \\
&= \mathbb{Z}^2
\end{aligned}
\tag{11}
$$

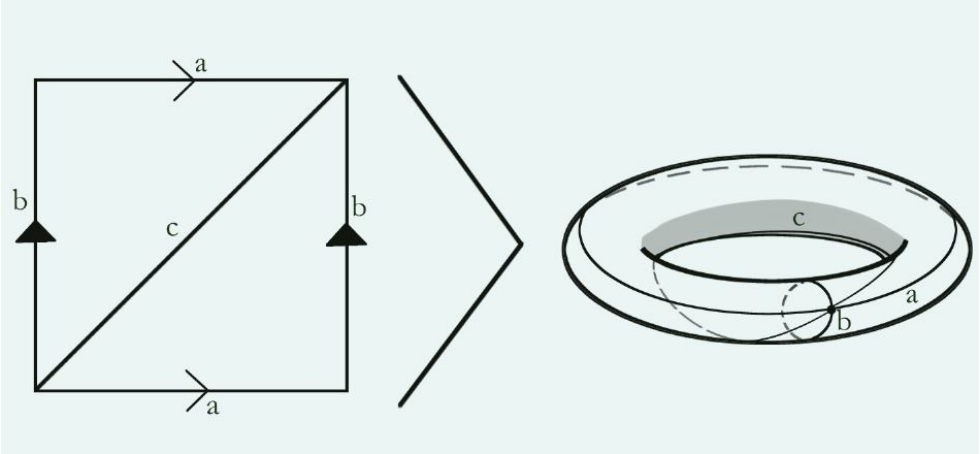

Figure 13: A triangulated torus T and its beginnings as a topologist's square.

Recall the torus T's fundamental group, $\langle a,b|aba^{-1}b^{-1}=1\rangle$. Its first homology group will be the abelianized $\pi_1(T)$, also $\mathbb{Z}^2$. To triangulate the torus, one can utilize a topologist's square as pictured in Figure 13; by connecting (often called gluing) the matching edges together, one will create a torus exactly. Notice how the edges a, b, and c are labeled instead of vertices, as all the vertices end up identifying to each other.

$\ker(\delta_1(T))$ contains a trio of elements: the two unique cycles a and b, corresponding to one path around the tunnel through the center and another around the bubble within its walls, and the cycle $a+b+c$. Also notice how the triangulating edge c is contractible to a point, so it alone cannot be a non-trivial element. The $\text{im}(\delta_2(T))$ is generated by the identical boundary of the two faces made by the path abc.

$$
\begin{aligned}
\mathrm{H}_1 &= \ker(\delta_1)/\mathrm{im}(\delta_2) \\
\ker(\delta_1) &= \langle a, b, a+b+c \rangle \\
&= \mathbb{Z}^3 \\
\mathrm{im}(\delta_2) &= \langle c-b+a \rangle \\
&= \mathbb{Z} \\
\implies \mathrm{H}_1 &= \mathbb{Z}^3/\mathbb{Z} \\
&= \mathbb{Z}^2
\end{aligned}
\tag{12}
$$

A two-holed torus H is our final proving ground for homology, shown in Figure 14 with its triangulation on the left. Recall its $\pi_1(X) = \langle a, b, c, d | aba^{-1}b^{-1}cdc^{-1}d^{-1} = 1 \rangle$, which condenses to an abelianized $\langle a, b, c, d \rangle$. We therefore expect its H_1 to have four generators (this is demonstrated directly in (13)). The kernel of $\delta_1(H)$ consists of nine generators; all nine unique edges have boundaries of 0, since all points get connected into one vertex like the torus. The image of $\delta_2(H)$ contains five generators, as the boundaries of five faces can also generate the last face's boundary.

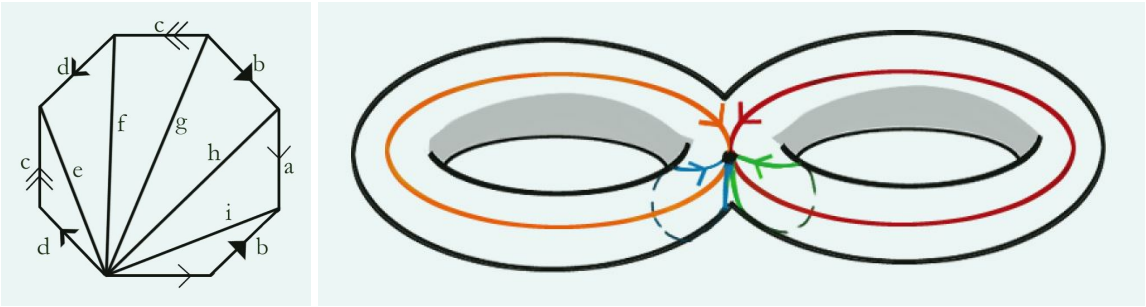

Figure 14: A topologist's octagon and topological equivalent of a two-holed torus H.

$$
\begin{aligned}
H_1 &= \ker(\delta_1)/im(\delta_2) \\
\ker(\delta_1) &= \langle a,b,c,d,e,f,g,h,i \rangle \\
&= \mathbb{Z}^9 \\
im(\delta_2) &= \langle a+b-i, i-a-h, h-b-g, g+c-f, f+d-e \rangle \\
&= \mathbb{Z}^5 \\
\implies H_1 &= \mathbb{Z}^9/\mathbb{Z}^5 \\
&= \mathbb{Z}^4
\end{aligned} \tag{13}
$$

Homology is the missing link between the theoretical heights of homotopy and the concrete application of data science. The ideas homotopy presented– finding holes, mapping a figure by paths– can still be done in homology, provided one is meticulous with notation. Simplicial complexes take care of that, housing the details of a shape within its dimension-bound simplices. Can the same thing could be done to data? It's a curious suggestion: datasets are already numbers, aside from qualitative entries. Why view them as a shape just to reconvert them into chains and groups? This current of thought directs us swiftly toward the goal of topological data analysis.

4 Topology in Data Analysis

Topological data analysis first appeared as a discipline just over thirty years ago. The ideas of singular homology helped Patrizio Frosini look at data in a new light: one where they have volume, curves, and holes. Singular homology is the broader field which spawned simplicial homology, and it is still mainly concerned with finding topological invariants in high-dimensional shapes. Frosini was interested in classifying shapes by their size function, which essentially found the 'difference in shape'[17] between them. A few peripherally related papers separately concerning new ways to organize data, homology, and simplicial complexes trickled out in the late 1990s before the term 'topological persistence' appeared in a paper by the esteemed Edelsbrunner, Letscher, and Zomorodian in 2000. Thus was topological data analysis born, applying homology theory and size function to computer science and the burgeoning field of data science. The next decade saw an explosion of interest as fresh minds jumped on the wave of this new tool.

4.1 Topologizing Data

But the question remains: how *exactly* is one able to apply these topological investigations to data and draw useful results? The secret lies in a shift in perpective. Data, even qualitative data, is organized. Even when it's dirty, cluttered, inefficient, and half-complete, it is possible to impose an order and therefore, an orientation.

Consider any dataset, from anywhere: with some cleaning and manipulation, one can

17. Patrizio Frosini, "Measuring Shapes by Size Function," DOI: https://doi.org/10.1117/12.57059, *Intelligent Robots and Computer Vision X: Algorithms and Techniques* (Boston, Massachusetts) 1607 (February 1992).

structure it into a dataframe of rows and columns. What remains is a complex relationship between the information grouped by the columns and stored in the rows. One could view the k columns as distinct areas of variation, like axes on a graph i.e. dimensions of a figure. It is the same as an elementary x-y table holding the points needed to plot a graph in $\mathbb{R}^2$, simply scaled up. The rows can be treated as points on a graph embedded in $\mathbb{R}^n$ space and each column's entry is the row's value in that corresponding dimension k of $\mathbb{R}^n$. Thus, the data could feasibly be plotted in this n-dimensional graph as a cloud of points called, with great inspiration, a point cloud.

Seeing this point cloud as a collection of vertices goes hand in hand with using them to construct a simplicial complex. One can achieve exactly this by choosing an appropriate metric by which to measure the points' relative closeness. Metrics are critical in TDA, as they can be constructed for any $\mathbb{R}^n$ space and are not bound by any dimension in particular. A fitting metric is as important as choosing how to present the data, as it allows one to tailor what is highlighted as important or not. In many cases the usual distance metric is sufficient to begin, but some methods automatically rely on statistically-supported metrics, using probability distribution to prune outliers or find especially connected points before applying topological tools.[18]

Building a simplicial complex from the data points naturally builds up a figure around those points. Now one can feasibly study the multi-dimensional relationships the entries might have by studying the shape created by them. This shape can be considered a mathematical object created by data, an object with its own unique structure. It may have gaps, holes, clusters, or other notable behaviors which group data in relationships spanning the columns of its dataframe. Relationships between data in one or two columns is a common target in general data analysis, but

18. Brittany T Fasy et al., "Introduction to the R Package TDA," CMU TopStat Group, January 2024,

relationships between different categories can be much harder to find using traditional methods. TDA can find these relationships where other methods cannot.

If there is a dense cluster in the point cloud, those data entries must be close to each other in several ways. If there is a gap inhabiting many dimensions, that gap is present in several different columns of data. Analyzing its shape in this sense is independent of the columns' scales or proximity in range. It allows one to view all these relationships more efficiently than comparing pairs of columns to each other in the hope of finding something useful. Successful attempts to apply TDA have been made in a heavy handful of STEM fields[19].[20] The potential to apply it to *any* field which produces quantitative data is a driving motivation for its development.

TDA is currently in this tinglingly new exploratory stage. There are several approaches to the question of what aspects of the data's topology to focus on: finding and studying clusters of data; using those clusters to find and study the holes in the data instead; reducing the dimension of the data; and more, highlighting all sorts of other features in the created shape like ridges and peaks. Each method has multiple ways to present findings as well, but all methods require a robust and well-chosen metric by which to relate the points. Studying clusters in data is generally the most straightforward approach. We present a couple of methods in this category to demonstrate, namely the attractive Mapper algorithm and more pragmatic clustering trees.

19. Jose A. Perea, "A Brief History of Persistence," DOI: https://doi.org/10.48550/arXiv.1809.03624, *ArXiv*, October 2018,

20. Ziyad Oulhaj, Mathieu Carriere, and Bertrand Michel, "Differentiable Mapper for Topological Optimization of Data Representation," DOI: https://doi.org/10.48550/arXiv.2402.12854, *ArXiv*, February 2024,

4.2 Mapper

The Mapper algorithm is one of the oldest tools in TDA, being presented in 2007.[21] It is a beautiful example of simple clustering in TDA. It ignores much of the underlying subtlety of the point cloud unless one focuses on an aspect specifically, and often ignores the sizes of holes, the density of where the points are gathered, and even the overall shape they create. Mapper instead utilizes a topological theorem called the Nerve Theorem[22] to make a cover of the point cloud.

Discussion is brief here, or we may get *too* distracted on our way to persistence, but

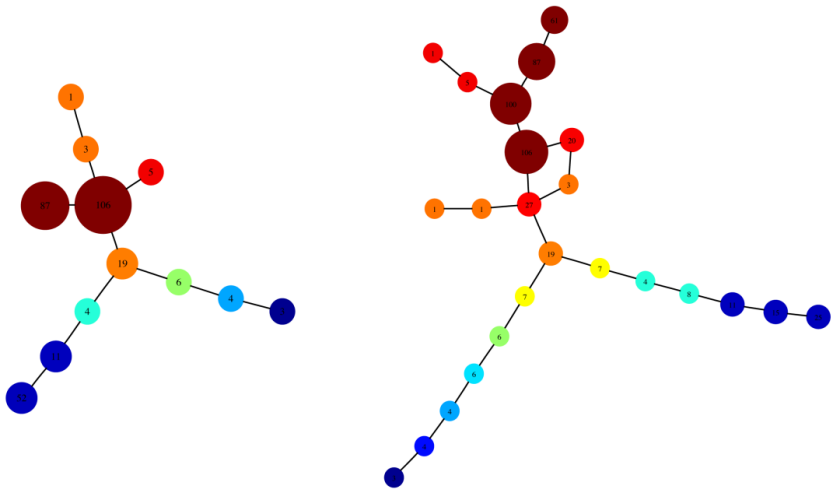

Figure 15: Mapper's analysis of diabetes predictors in medical data.

basically Mapper groups a point cloud based on any clustering algorithm one desires. It does not even require a direction (i.e. orientation) to collect up the data, provided distance between the points can be measured. By calculating a particular open cover called the nerve of a space X, one receives a collection of sets which each have overlap to at least one other set in the nerve. This allows Mapper to treat each set as its own nodule and draw connections between the ones

21. Gurjeet Singh, Facundo Memoli, and Gunnar Carlsson, "Topological Methods for the Analysis of high Dimensional Data Sets and 3D Object Recognition" (Prague, Czech Republic), September 2007,

22. Singh, Memoli, and Carlsson.

which overlap.

What results is a vivid display of nodes and edges as in Figure 15, which displays the Mapper result of processing diabetes data[23] with six dimensions, including risk factors and diagnoses. The two diverging branches correspond to diabetes I and II, and was even able to help predict the severity of diabetes as well as type. Mapper has even been used to present data in an interactive map, where nodules will expand to their own smaller webs when clicked. In this age of screens instead of paper, such engaging presentations are a classy option when the situation allows. Currently Mapper is still being refined, and projects as recent as this year have been published on how to improve the tuning of its parameters and its analytical prowess.[24]

4.3 Cluster Trees

Clustering trees are an example of focusing on clusters of points, but can also analyze them based on their statistical probability. To analyze data by its densest clusters, a cluster algorithm is used for the metric to find which groups measure up to the degree of closeness needed. This is commonly called the λ-tree approach and the data is split up by its clusters (within clusters, as the algorithm measures at finer and finer thresholds) from one major 'stem' into branches.[25] To analyze the data by a different method, α- and κ-trees consider the statistical approach.[26] The data is reverse-fit into a probability distribution which might reasonably lead to them as a result by using a statistically-driven metric. That probability distribution guides splitting the data.

23. Singh, Memoli, and Carlsson, "Topological Methods for the Analysis of high Dimensional Data Sets and 3D Object Recognition."
24. Oulhaj, Carriere, and Michel, "Differentiable Mapper for Topological Optimization of Data Representation."
25. Fasy et al., "Introduction to the R Package TDA."
26. Fasy et al.

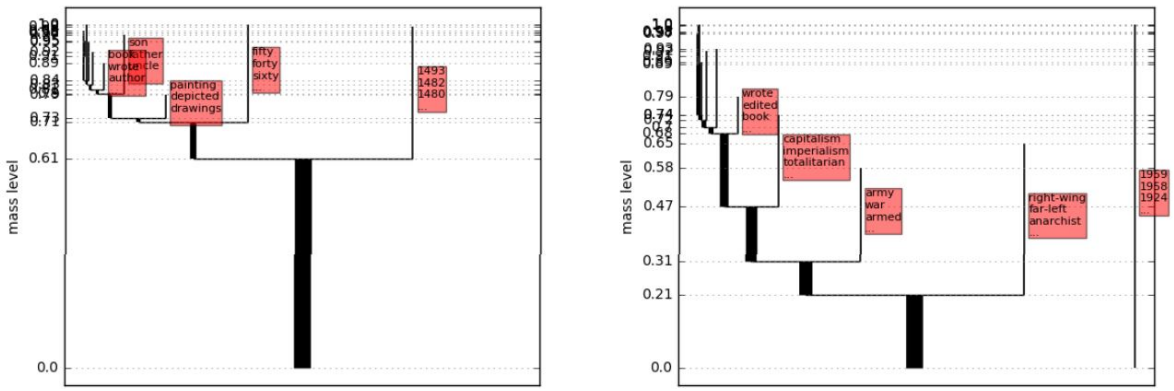

Figure 16: Cluster tree presenting common words in an author's works. The left tree analyzes Leonardo da Vinci; the right, Noam Chomsky.

Whichever method one chooses, the branches in a cluster tree are always those disjoint subsets of its stem's group which are different enough to be determined unique. Now the name cluster tree becomes clear as assessment (of either density or probability) across dimensions drives the partitioning of the data. This process continues, splitting up the subsequent groups of data as long as there is reason to do so. Once the whole dataset has been split up in this way, the data is then assessed for skewing outliers and those 'leaves' of the tree diagram are then pruned off. What is left is an organized display of the data, either by density or by likelihood.

Figure 16 demonstrates this approach processing the words of two very different authors' works.[27] One can see at a glance what keywords are most connected to da Vinci's works, how different they are from Chomsky's, and how the words appear in relation to each other. The statistics behind clustering are imperative to how the findings will be presented, but this method has the capability to easily sort data any number of ways.

27. Bhuwan Dhingra and Chaitanya Ahuja, "Statistical Topological Data Analysis," 30th Conference on Neural Information Processing Systems, December 2016,

4.4 Towards Persistence

We see TDA can clearly be used to analyze data by which points are related in many ways. It seems one is only limited by one's imagination: what metric, what threshold, what presentation? But these still focus on one side of closeness. How about analyzing data by finding which points are *not* related in many ways? This leads to persistent homology, the tool we have been chasing after through these very pages. As one of the primary approaches to TDA[28],[29] persistent homology utilizes the algebraic topology presented earlier to track down voids in each dimension.

In addition, it tracks how many consecutive dimensions these voids occupy. A gap in many dimensions at once between points is a much more significant hole than a gap within only a couple columns of information. As the specific focus of interest, we will spend more time with persistent homology than any other TDA tool discussed thus far– so much time, in fact, it deserves its own section. Before we leave the general study of TDA behind, let us take stock of its progress so far.

TDA as a whole has yet to become a widespread tool by those working in data science. Like any other new technology, there are obstacles to maneuver past to refine it to a widely usable status. One of the first was how to present findings: a brilliant new school of thought is unlikely to be applied to much if its results are obtuse, or worse, inaccessible. That problem was solved rather quickly; these days, one has a happy selection from which to choose. Some are stripped-down, bare bones presentations of dense insights like barcodes or trees, some are interactive and glitzy like the Mapper algorithm or neural clusters.

28. Frederic Chazal and Bertrand Michel, "Persistent Homology in TDA," Geometrica, *Inria*, June 2016,
29. Dmitriy Morozov, *A Practical Guide to Persistent Homology*, Lawrence Berkeley National Lab, March 2023.

The next issue is that of its high learning curve. At the moment, TDA mostly exists as code one can integrate into a program. That implies one must be familiar with coding, a skill which has yet to become widespread. In addition, the parameters of these tools can be a bit overwhelming, and self-directed learning is a challenge indeed (though certainly not impossible). This has been improved by helpful readme files included with the TDA packages and tutorials for the more popular tools, which can now be found with a simple search online. It has yet to become truly user-friendly, although several programs now have TDA packages or bindings to use them, including C++, R, and Python[30].[31]

TDA also struggles with its outputs: without background in statistics or a similar field, the results one gets can seem oversimplistic or unremarkable. One must often draw insights from it and then restructure the results in a different presentation to be clear to a less familiar audience. This additional processing must accompany TDA for it to be accessible enough to spread to a wider audience.

TDA's final major obstacle is its newness. Much of its use has yet to be discovered, and explaining this exploration to another often feels underwhelming when the other asks, "but how is it used?" The beginnings of a thing are often small, and growth takes work. TDA is still a nebulous possibility, finding connections but not the implicit reason of connectedness. When it becomes more sophisticated, TDA could feasibly replace our current approach to data science, but it still has a long way to go before the average person will be using it.

30. Fasy et al., "Introduction to the R Package TDA."
31. Morozov, *A Practical Guide to Persistent Homology*.

5 Persistent Homology

We have waded through all sorts of related knowledge to rest, for a moment, upon this sandbar called persistent homology, or **persistence** for short. Even in its most abstract realizations, persistence is less than 80 years old and has been theorized far more than utilized. The thought process is this: if homology groups measure voids, is there insight to be drawn from voids which punch through many dimensions rather than just one or two? If a void persists through many dimensions, it must affect the points in multiple dimensions as well, making it more impactful than a flat, one-dimensional hole.

Persistence takes everything a reader had to swim through to get to this point and takes it one step farther, integrating it together in a sophisticated TDA approach capable of not just finding holes, but measuring the relative sizes of those holes. Homotopy theory is how one articulates this idea, homology groups translate that articulation into possibility, and simplicial homology changes that possibility into a concrete plan of attack. Now we bring persistence into reality by translating the steps into code and utilizing the computational power of our modern technology.

5.1 Filtration

To write a program to achieve homology group computations up to any dimension we desire, we must structure our data impeccably. The higher dimensional faces must be build from its lower ones, relying on a robust metric to determine which lower faces are within the distance threshold needed to connect. A specific type of simplicial complex, called a filtration,

will accomplish this. The succinct definition from Ghrist follows.[32]

A **filtration** of a complex K is a nested sequence of subcomplexes:

$\emptyset \subset F_1K \subset F_2K... \subset F_{n-1}K \subset F_nK = K$ where F_iK is nonempty and distinct.

Each F_iK will be strictly contained in the $i+1$ complex, F_1K always contains all data points in the point cloud– the vertices of the shape being built– and the largest subcomplex F_nK must be identical to the final complex K itself. This filtration can be inductively built as demonstrated in the popular Vietoris-Rips complex, which is naturally a filtration by its structure if one repeatedly increases the distance parameter ε. The Vietoris-Rips complex approaches a dataset with the Euclidean distance metric and builds the shape by connecting faces which are within the distance ε to each other. The formal definition of a Vietoris-Rips complex follows.[33]

A **Vietoris-Rips complex R(X, ε)** consists of simplices with vertices in

$X = x_1, x_2, ..., x_n \subset \mathbb{R}^d$ and diameter at most ε.

This ε starts very small, and the resulting 'bare-bones' complex is just the set of points themselves: F_1K. This process is then repeated after increasing ε until the complex changes; now some edges are possible, perhaps even some faces. This resulting complex is F_2K. The process repeats, growing ε with each iteration, and one can intuitively tell the resulting complexes are coarser and coarser representations of the shape being built, as shown below. Each complex will be contained in the simplicial complexes which came before it until ε is large enough to contain every possible face: this is F_nK, or K itself. Structuring a data set as a filtration makes it possible to build the shape inductively and to collect more detailed 'pictures' of it in the form of these larger complexes.

32. Ghrist, *Foundations of Topological Data Analysis*.
33. Fasy et al., "Introduction to the R Package TDA."

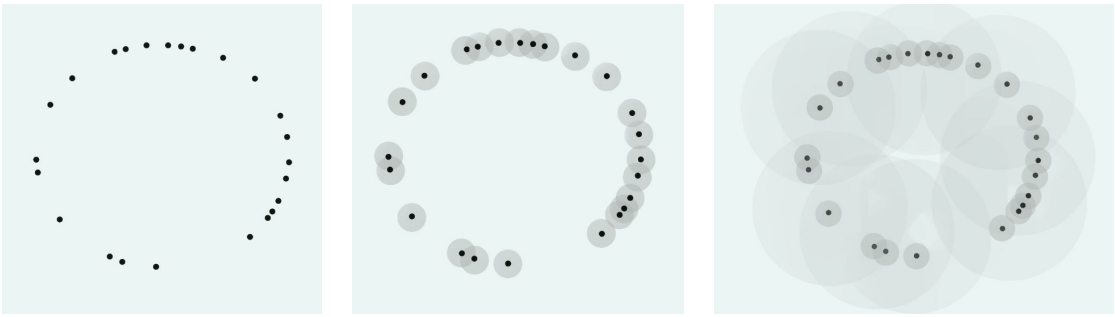

Figure 17: Persistence in action: the distance threshold creates and kills holes as it grows.

Building a filtration allows one to find the data's homology groups algorithmically by calculating them in the final resultant complex K. However, the real uniqueness of persistence is in the ability to compare homology groups of the entire filtration. By calculating homology groups in each subcomplex $F_i(K)$, one can see when the structure is coarse enough for a void to appear, and when it becomes too coarse for that void to continue to exist. Figure 17 illustrates this simply: in the first subcomplex when the distance threshold is very low, the central hole has yet to be measured. All we can 'see' in the homology groups is a bunch of disconnected pieces. As the distance threshold grows, the points are connected and the central hole can be measured and found. The gaps between points also disappear. As the threshold continues to increase, there will be a complex where the central hole is filled in, and the homology groups will no longer 'see' it either.

Calculating homology groups of the entire filtration tells a story about the shape. By comparing corresponding homology groups between subcomplexes, one can watch voids being born, only to die out later in coarser subcomplexes. This birth-and-death relationship is at the core of persistence. It highlights the biggest gaps in data, ones which persist through many iterations of the shape (hence the name) and over great lengths, not just blips.

Now the way to code this behavior becomes clear. A program can inductively build a filtration from a dataset, then calculate the homology groups at each level. The first subcomplex which shows a gap between data– a new generator in a homology group– gets marked as the birth of that gap, and when that distance between points is closed in a subsequent subcomplex, it marks it as a death of the gap. Gaps which are born early in the filtration and die late, like the central hole in Figure 17, persist much longer than those with a shorter lifespan, like the many gaps between the points.

5.2 Presentations

There are as many ways to represent one's findings as it is to investigate to find them and persistence is no exception. We feature three different presentations for the findings of persistence here for context.

A barcode is the initial, and most rudimentary, way to present persistent homology's findings. This is not to say it is not effective. There is only one axis of information, displayed horizontally: time. This keeps the diagram simple and immediately legible. Each void is represented as a line color-coded to the homology group in which it is contained; each begins at the time code of its birth and ends when it dies. This simple presentation allows the most significant voids to jump out, as the longest lines are the voids which live the longest through the filtration. Consider Figure 18.[34] The longest black line is the single generator of the zeroth homology group (indicating it is one broadly connected piece) and the red line is the first homology void, i.e. the hole in the center. It can sometimes be difficult to determine which

34. Fasy et al., "Introduction to the R Package TDA."

voids are significant, though, as it is left completely to the viewer. As the data's shape gets more complicated, one can see how several voids with similar lifespans yet differering time codes or dimensions could be challenging to analyze. To correct for this, the data can be further processed and presented in a KDE diagram.

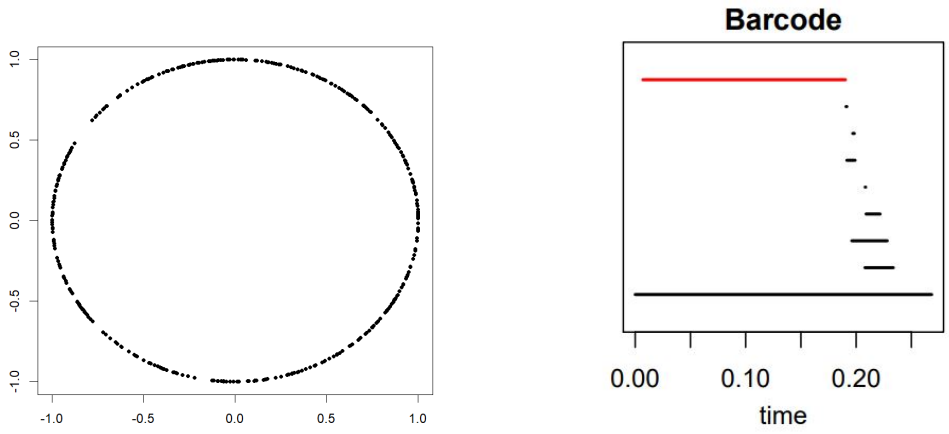

Figure 18: A circle of points and its corresponding barcode.

A *kernel density estimate*, or KDE diagram, is a standard presentation for persistence. It includes a statistical confidence band which helps eliminate short-lived gaps. Any points or symbols outside that confidence band are the significant ones, born early and dying late. To clearly present this, the birth and death times of a void are each considered a separate axis and each void is plotted as a point in $\mathbb{R}^2$. This is more sophisticated, but less efficient than the barcode, as half the graph's area goes unused by design. Notice in Figure 19 how the symbols representing voids vary based on which homology group contains the void: here, black dots are generators of the zeroth homology group and red triangles belong to the first homology group. There is one significant element in the zeroth group (one connected piece) and one significant element in the first group (the hole in the center). This matches perfectly with the insights we drew from the barcode (as one would hope, analyzing the same figure).

Despite its popularity, the KDE diagram is not without its drawbacks. In this presentation, it is impossible for a void to appear in the lower right. Since both axes use the same scale, a void appearing there would imply it is born after it dies. Sometimes researchers choose to present an altered form of the KDE diagram where the graph is rotated. The central $x = y$ line is adjusted to be the x-axis, and therefore the tallest points on the graph will be the longest-lived voids. However, the classic KDE diagram is the most popular choice by far. The confidence band included allows one to simply find the symbols outside the band and as the most likely significant voids. Of course, one must choose an appropriate confidence interval to make the best use of this. We will revisit the KDE diagram in greater detail in the following section.

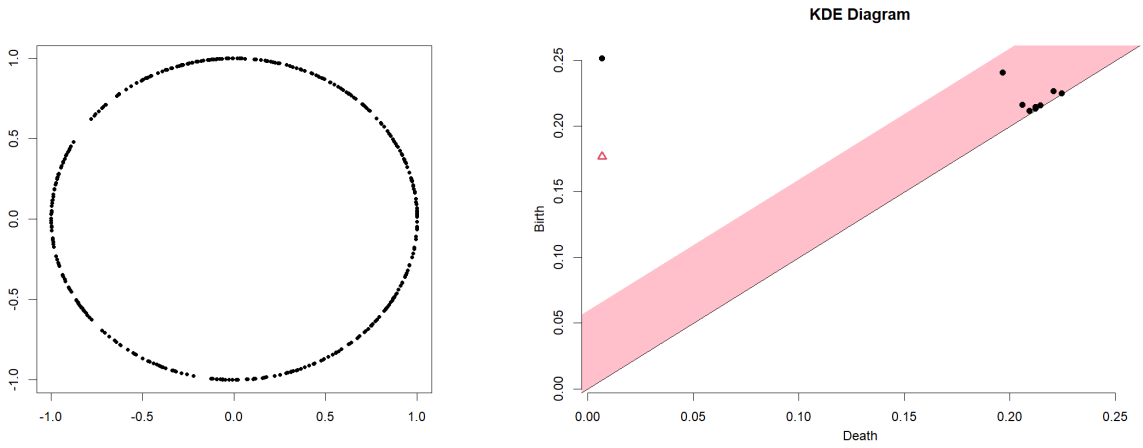

Figure 19: A circle of points has only a couple significant features in a KDE diagram.

A persistence silhouette (and its landscape) displays the same information, further processed. Most researchers are satisfied with the KDE diagram, so these are less popular, but in the right hands they have a niche use. One makes a persistence landscape from a rotated KDE diagram by doing away with the confidence interval and treating the points of one homology group as the unique vertices in isoceles triangles, all grounded on the x-axis. This series of little mountain peaks are then pruned, with all lesser triangles (the ones completely engulfed by larger

ones) omitted. What remains is a single jagged line which creates the landscape we see. This removes any voids which are born and die in the shadow of a looming, larger void, and is a simple way to remove insignificant gaps.

To make a silhouette, as in Figure 20,[35] these remaining points (the peaks of the mountains) are each modeled with a pair of linear equations, combined characteristically in a piecewise function. These functions are then processed through a "power-weighted silhouette"[36] function. This silhouette furthers the distance between significant and insignificant points using a smoothing parameter– if there are several similar lifespans of gaps, the silhouette will be fairly level and show many 'most significant' points, but if there are few points of significance, they will be inflated as large peaks in a more dampened skyline.

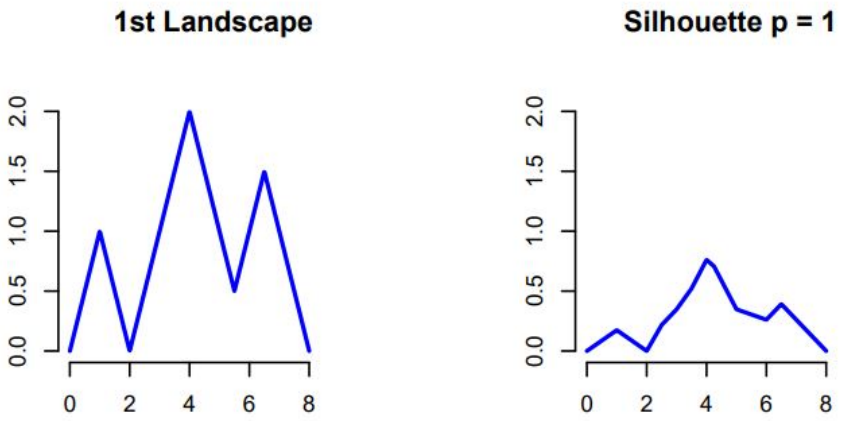

Figure 20: A persistent landscape and its corresponding silhouette.

There are other ways to present findings in persistence, and the number of choices is growing by the year. Different parameters for metrics and statistical significances, like the k Nearest Neighbor density estimator, exist depending on one's objectives. Variations on presenting

35. Fasy et al., "Introduction to the R Package TDA."
36. Fasy et al.

the data abound, from simple cluster labels to even further processed representations like mean landscapes with confidence bands.[37] There are too many to showcase here, as our focus is on choosing a reliable method with which to discuss our findings, and it is time to leave the many options TDA offers us behind. The KDE diagram with a confidence band is a tried-and-true choice to present persistence, so the following section uses this exclusively. Any reader ought to be well equipped now to venture farther into the ocean of abounding data, including the few samples offered in this exploration.

37. Fasy et al., "Introduction to the R Package TDA."

6 Datasets

We are ready to swim on our own: the knowledge gained thus far buoys us with confidence to explore data and see what persistence might show us. The data manipulation software R offers a package specifically for topological data analysis, neatly called `TDA`.

6.1 The TDA Package

Developed in 2021[38] and updated as recently as January of this year, the package `TDA` allows one to calculate persistent homology and density clustering, then represent them in a variety of ways. Density clustering can be represented in a cluster tree and is possible with only the `TDA` package, but persistent homology requires the use of an additional package. Thankfully, it includes an interface for some C++ libraries which contain the function for persistent homology, and of these we elected to use the `Dionysus` library.

Thus armed, one can direct R to analyze data– after some basic cleaning. It is prudent to normalize the data to begin; the underlying features will still be present and it is easier to fit parameters within functions to the data itself. This is not a necessary step and many of the following examples are not normalized, but it can help tame the unpredictable nature of real world data and is suggested in such cases. Once the data is prepared and parameters set, it is ready to be put through three small blocks of code, illustrated in Figure 21: the `gridDiag` function, the `bootstrapBand` function, and a tailored `plot` function.

38. Fasy et al., "Introduction to the R Package TDA."

```
10 Y <- df
11 lim <- c(0, 100)
12 by = 1
13
14 margin <- seq(from = lim[1], to = lim[2], by = by)
15 Grid <- expand.grid(margin, margin, margin, ...)
16
17 h <- .3
18 by <- 1
19 Xlim < c(-1, 1.1);
20 Ylim <- c(-1, 1.1) ;
21 Zlim <- c(-1.5, 1. 5) ...
22
23 DiagGrid <- gridDiag(
24   X = Y, FUN = kde, h = 0.3, lim = cbind(Xlim, Ylim, Zlim, ...), by = by,
25   sublevel = FALSE, library = "Dionysus", location = TRUE,
26   printProgress = FALSE)
27
28 band <- bootstrapBand(
29   X = Y, FUN = kde, Grid = Grid,
30   B = 75, parallel = FALSE, alpha = 0.05, h = h)
31
32 plot(
33   DiagGrid[["diagram"]],
34   band = 1 * band[["width"]],
35   main = "KDE Diagram")
```

Figure 21: R code for generating KDE diagrams of datasets.

After assigning the intended data to the oft-more easily referenced variable `Y`, one can see several parameters are needed to be carefully chosen before being able to run the functions. Lines 14 and 15 set a grid of points as a framework to hold the data. Thus, the repeated `margins` within `expand.grid` must match the dimension of the data. Line 17's `h` is a smoothing parameter which controls how granular the findings will be treated and is set to the suggested value of 0.3.[39] The `by` below it is a step parameter and the limits through Line 21 set the domain for each column of data (hence why it is helpful to normalize data, especially if one may need to run many different sets through).

39. Fasy et al., "Introduction to the R Package TDA."

The first proper algorithm `gridDiag` follows. After setting both the data (`Y`) and the function (constructing a `KDE` diagram) to be used, the `lim` argument's dimension must necessarily match that of the data as it sets the range for each dimension using the referenced limits. This `gridDiag` algorithm is what calculates the persistent homology and compiles every void in every filtration of the data set. The `by` determines the fineness of the filtration itself– how much the distance threshold increases per iteration– and the library argument calls whichever supplemental C++ library is being used. The rest of the inputs are options one can include, such as recording the location of where voids are born and die, and whether to communicate the progress toward completion while the function is running or not.

The next algorithm `bootstrapBand` calculates an extremely important confidence band which, in the KDE diagram, rests atop the Birth = Death line of $x = y$ (see Fig. 19, 22, etc). It does this by calculating the distance between the KDE results from a random subsampled collection of points in `Y` and those of the entire set. By doing this many times, `bootstrapBand` collects a large bank of points it can trim to the desired level of confidence; the statistically-determined outliers are then used to determine the height of the band it constructs. This pink band, when plotted, cuts off the relatively insignificant homology points. Its parameter field begins the same way, with the data to be used, the function being evaluated, and the space the data inhabits (`Y`, `kde`, and `Grid` respectively), and the tunable parameters follow. `B` determines the number of iterations `bootstrapBand` runs, which determines the sample size of this confidence calculation. The higher `B` is, the longer this algorithm takes, but it gains a larger sample size from which to determine outliers. The `alpha` sets the confidence level; with an `alpha = 0.05`, this will calculate at 95% confidence. This `bootstrapBand` algorithm is the most computationally expensive of the three and takes exponentially more memory to run as one increases the dimensions of data

being processed.

The final algorithm, `plot`, graphs a desired set of points in a two-dimensional space and is an intrinsic feature of R. We instruct it to plot the results of `gridDiag` (named `DiagGrid` on Line 23 for easier referencing) and include a band the width `bootstrapBand` suggests (named `band` on Line 28 for the same reason). The last parameter, `main`, merely gives the graph a label. Thus, `plot` will graph the marked homology group generators, or voids, by when they were born and when they died. This natural comparison displays nicely on an x-y graph, and finally we see our completed KDE diagram.

6.2 Deciphering KDE Diagrams

Now more rigorous investigation of persistence in action is warranted. Reason instructs one to trust it, but without personal experience, faith in something abstract can be a tall order. Instead, let us apply it to some known examples to gain some familiarity. Once it has revealed its mannerisms to us, perhaps real-world data will be an easier fish to catch.

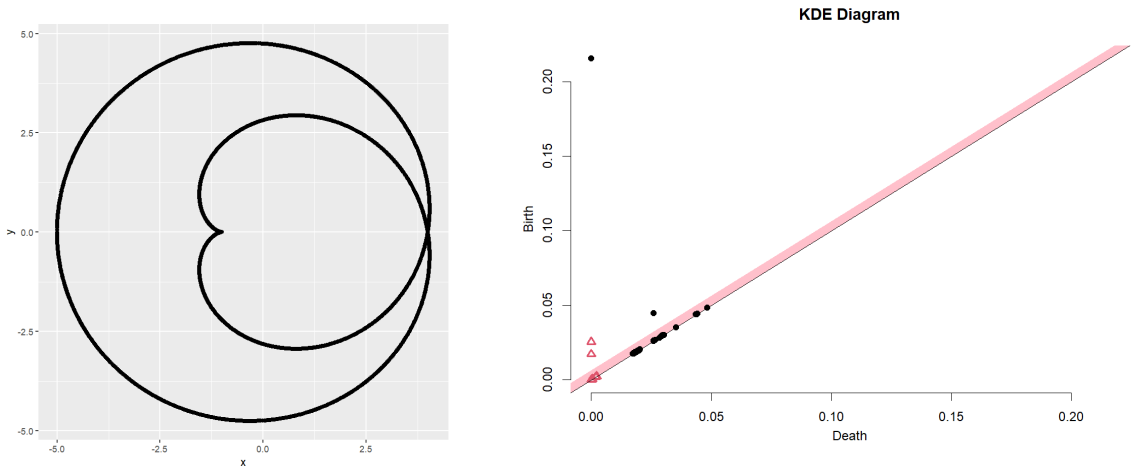

Figure 22: A classic cardioid and its KDE diagram.

Consider the cardioid shown in Figure 22 and its corresponding KDE diagram; the diagram suggests there are two major pieces to the figure and it holds two holes in $\mathbb{R}^2$. Notice how much closer one H_1 point (denoted by black dots) is to the confidence band than the other; this suggests the separation detected between these two pieces is nearly insignificant. By comparing it to the graph of the cardioid itself, we see it is likely picking up the outside circle and inside heart, and the connection between them is missed– as in, the general subsampling of points in the `bootstrapBand` algorithm did not sample the connecting point between the pieces enough to determine the void read by the filtration there to be insignificant.

This KDE diagram had its iteration parameter `B` set to only 1, so its mistake may be easily remedied. If the `B` is increased, that second H_1 point is much closer to the confidence band, but still technically gets missed– and the calculation takes more time and processing power. By adjusting this parameter a bit more, one discovers a 'sweet spot' between speed and accuracy for `B`.

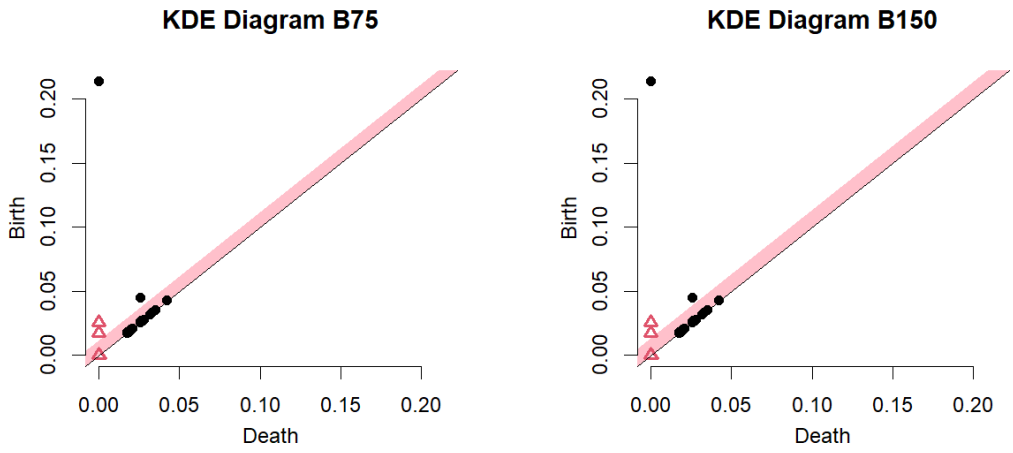

Figure 23: Time Versus Accuracy. B = 75 took 25 seconds to generate. B = 150 took 45 seconds.

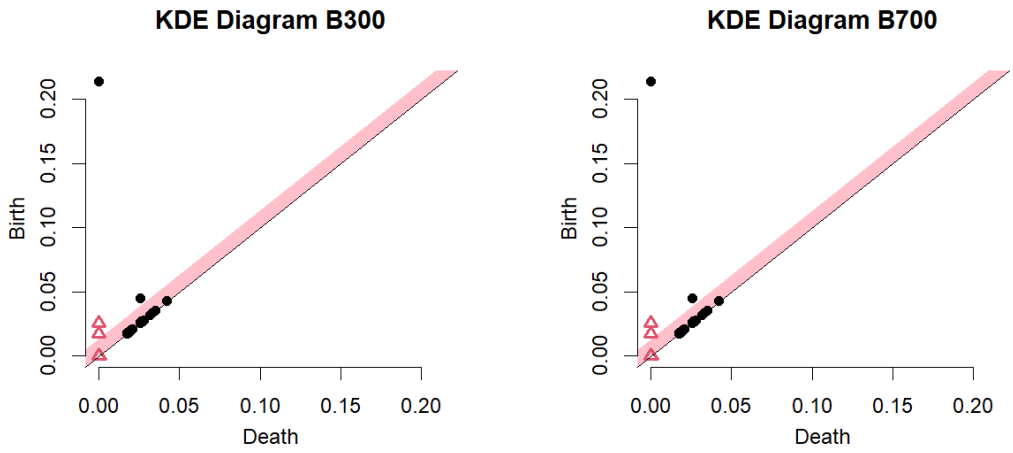

Figure 24: B = 300 took close to two minutes to generate; B = 700 took nearly five minutes.

Figures 23 and 24 display KDE diagrams of the same cardioid, but with `B` values of 75, 150, 300, and 700. As `B` increases, the diagrams do become more accurate, but not enough to completely overcome that oversight, and each needs more time to compute than the last. In fact, there is little discernable difference, if any, between the the last three of these plots. When the shape being analyzed is completely visible, it is easy enough to recognize points close to the border as belonging on the other side of the line. This skepticism ought to be carried forward; each point's distance from the confidence band is as important as the number of points outside of it. Otherwise, this analysis seems to communicate its findings rather accurately on a simple model.

Now, the process of persistence is supposedly a robust manner of analyzing data, but *how* is it robust? A reliable place to begin is determining if it is independent of affine transformations in $\mathbb{R}^n$, where any data might be plotted and analyzed. If so, the patterns and underlying topological features should be picked up despite the perspective from which one views them. Recall an affine transformation, or motion, is any transformation which necessarily preserves distances.

Any motion can be described as a composition of four basic ones: translations and rotations (the direct motions), and reflections and glides (indirect, or orientation-reversing motions). The core of persistence, building a filtration between the given points, derives directly from distance between points. This is enough to show any motions on a dataset should result in the same filtrations from a given distance threshold, and so the same diagrams, as the original dataset.

Let us examine the KDE diagrams of the cardioid when it has been so transformed to reinforce our confidence. A reflection reverses the shape's orientation, which causes significant change, so it seems as good a place to start as any. Figure 25 displays a reflected cardioid and its resultant KDE diagram. The diagrams are exactly the same. Figure 26 takes this one step further by reflecting, then rotating, the cardioid. Again, the diagrams match.

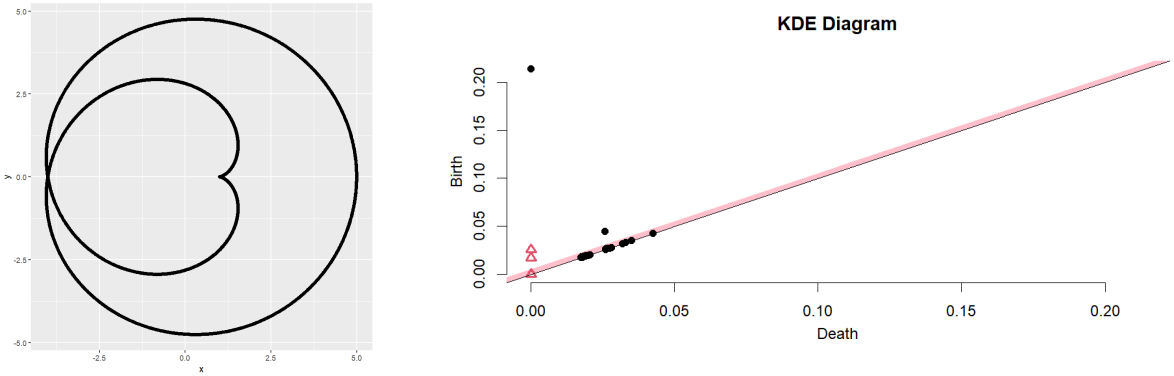

Figure 25: The reflected cardioid. Its KDE diagram is identical to the one in Figure 22.

Just changing the orientation of a shape (by changing the perspective on the points beneath it) does nothing to alter the underlying features like connectedness and holes. The other two types of motions have the same lack of effect: translations obviously do not change the distance between the points, only the relative placement the entire set is from the origin. Glides are a special composition of a translation and a reflection, both of which have no impact on this

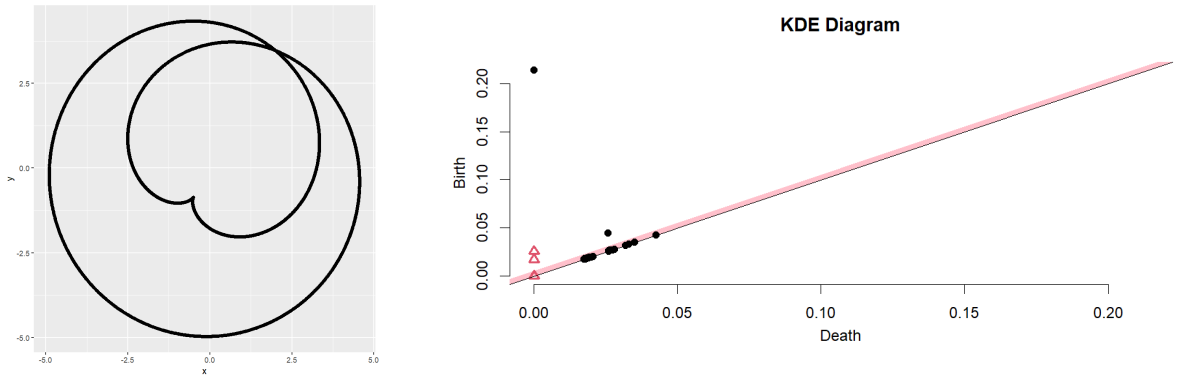

Figure 26: The rotated and reflected cardioid and its matching KDE diagram.

analysis. Thus we confirm motions do nothing to change persistence, so any direction from which we approach this analysis is viable.

This enigmatic tool is becoming clearer. What if we analyze a new shape, one which is visually different but has the same types of holes? Perhaps epicycloids of other sizes can further inform us on how persistence speaks. Figures 27, 28, and 29 display three, five, and seven petaled epicycloids with their correspoinding KDE diagrams. These were all processed with `B` = 100 and `h` = 0.3. Unlike the cardioid, these need a bit more scrutiny before revealing their secrets.

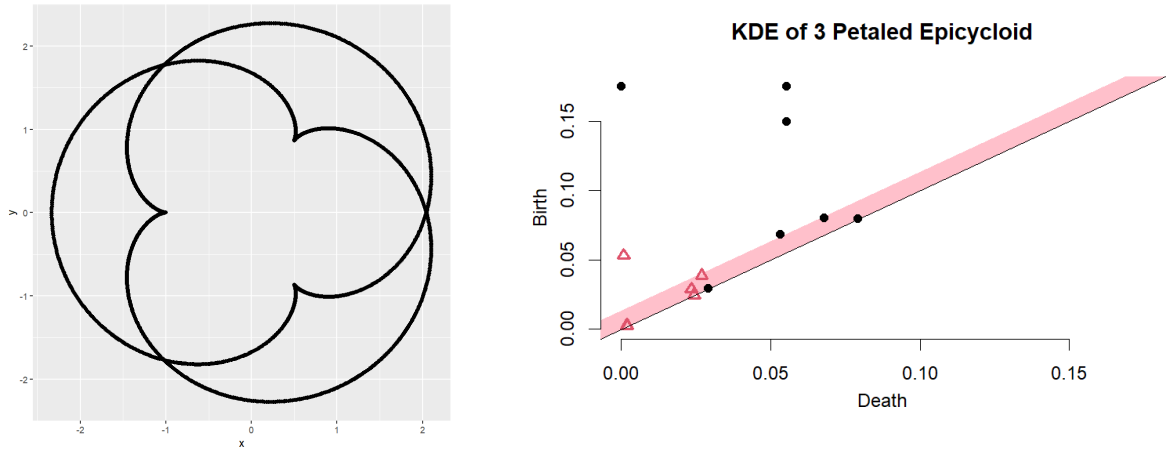

Figure 27: A three-petaled epicycloid and its KDE diagram.

The three petaled cycloid, for example, clearly has one connected piece with four two-dimensional holes in it, but its KDE diagram seems to think it is in multiple pieces and only has one two-dimensional hole. The cardioid's KDE had a difficult time sampling that single connecting point between the two major pieces; this may have the same obstacle, but three-fold. It may read this shape as three separated pieces, whose corresponding generators all die at the same time– making it one shape represented by the first clear H_1 point. It is a difficult thing to tell for certain, but the single H_2 point certainly represents the large hole in the middle of the figure. The rest, just beneath the border of the bootstrap band, likely track the smaller holes and were ultimately deemed insignificant by the confidence test. Again, points close to the boundary (on whichever side) may be background players in the structure of their shape.

Thus intrigued by the teachings a new shape had to give, we proceed to the five petaled

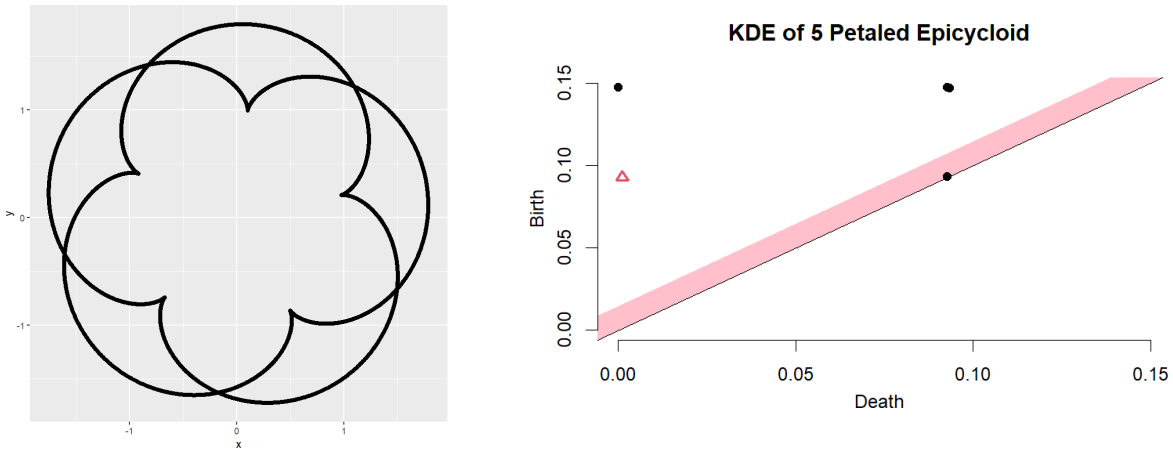

Figure 28: A five-petaled epicycloid and its KDE diagram.

cardioid in earnest. It looks at first to be disappointingly simple, but a careful eye will notice the odd shape of the highest-valued H_1 point. It is, in fact, two H_1 points on top of one another. It brings another useful insight to light– the idea of multiplicity. Multiple holes could be born and die at near exactly the same times, resulting in more holes hiding in plain sight. In Figure 27,

there are three visible H_1 points above the confidence band, but perhaps that is the minimum number of significant points. There could be more of the same type, all being born and dying at the same time. This KDE diagram hints at the symmetry of these epicycloids: clearly the corresponding shape has one connected piece (as the most removed H_1 point suggests), but is made up of several radially-symmetric pieces. One can conclude the layered H_1 points in its KDE diagram denote the many petals.

The single H_2 triangle is perhaps the more interesting feature– or, more notably, its lack of cohorts. If one expects a similar result to the three-petaled epicycloid, these many insignificant, yet observed, points are missing. By comparing figures, one notices the corresponding holes in the five-petaled figure are smaller with a shorter radial width. It is likely the distance threshold was not fine enough to pick these up as the neighborhoods around each sampled point grew; by the time the points might be connected to show a hole, the neighborhoods were big enough to connect across the shorter width as well. Perhaps we know enough of KDE's mannerisms now to loosely predict the result of a seven-petaled epicycloid.

The KDE of the seven-petaled cycloid is not wildly unusual– a good sign, as it shows

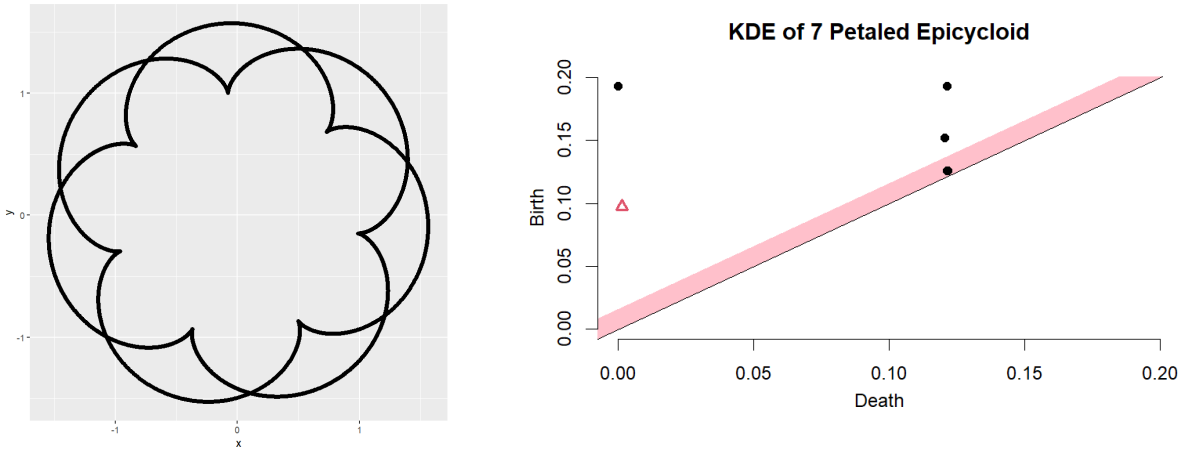

Figure 29: A seven-petaled epicycloid and its KDE diagram.

some predictions *can* be made for it. The single H_2 point is expected at this point; the other holes in this figure are even smaller than the last and were similarly missed. Like the rest, the single removed H_1 point informs on its connectedness and the linear H_1 points suggest it has at least two pieces which fade away into that lasting, removed point. The H_1 point within the confidence band is actually at least two, only visible if the figure is enlarged significantly. While one can understand the result when comparing it with the figure, this is not terribly useful when exploring unknown data. It is possible adjusting the smoothing parameter `h` may improve the accuracy. Figure 30 displays two more KDE diagrams of the seven-petaled cycloid with `h` values lower than the usual 0.3, keeping `B` and other parameters the same.

The results seem significantly different at first, until all three diagrams are compared.

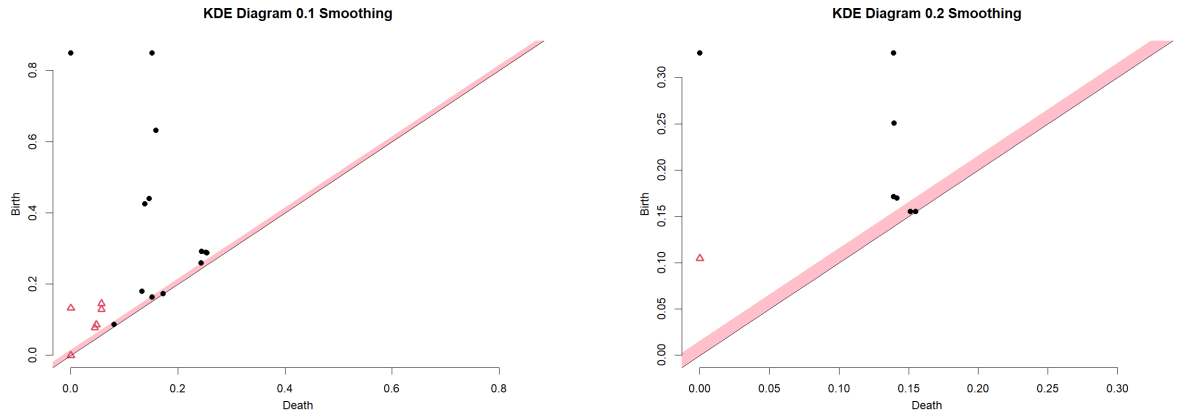

Figure 30: Lowering the h parameter expands what the program considers significant.

The major features are consistent between them: the left-most H_1 and H_2 points remain the same, and a string of H_1 points around the Death = 0.12 mark hint at the many symmetric pieces of the cycloid. More noticeable are the extra features included in results using lower `h` values. They have not sprung from nowhere, but were always there, lurking. The `gridDiag` function just determined them too insignificant to include.

$\texttt{h} = 0.1$ displays the most features– perhaps it displays too many. It claims there are at least 8 significant H_1 points aside from the most enduring one, which is clearly too many (even if by just a small amount). It claims there are at least four extra H_2 holes as well. While being admittedly more accurate there, it is poor practice to trade one type of accuracy for another unless it is directly more useful. $\texttt{h} = 0.2$ is closer to the mark, but not by much. The only difference is a couple more H_1 points which eventually disappear. Noting two or four temporary pieces in a seven-petaled cycloid makes little difference, which adds support to why $\texttt{h} = 0.3$ is the standard smoothing parameter. This may make a difference in other data, but that is left to a case-by-case basis.

6.3 Noise

We have considered how to read KDE diagrams and its hidden nuances; it is often not easy, nor straightforward, but it is enough in these early stages to know it is reporting *something* of use. How resistant is it to noise? All the marks which fall within the confidence band are considered topologically insignificant; in other words, noise relative to the other features which stand out more. But what about analyzing intentionally noisy data to test its resistance?

Let us compare the results after processing a three column dataset of a cleanly-plotted trefoil (Figure 31) and a noisy trefoil (Figure 32) in $\mathbb{R}^3$. R Studio plots many two-dimensional perspectives of a three-dimensional image when it has no three-dimensional visualization libraries installed. For example, the top center and left center images display the trefoil from the xy-plane at different orientations: first on the xy-plane, then on the yx-plane. On the right are the

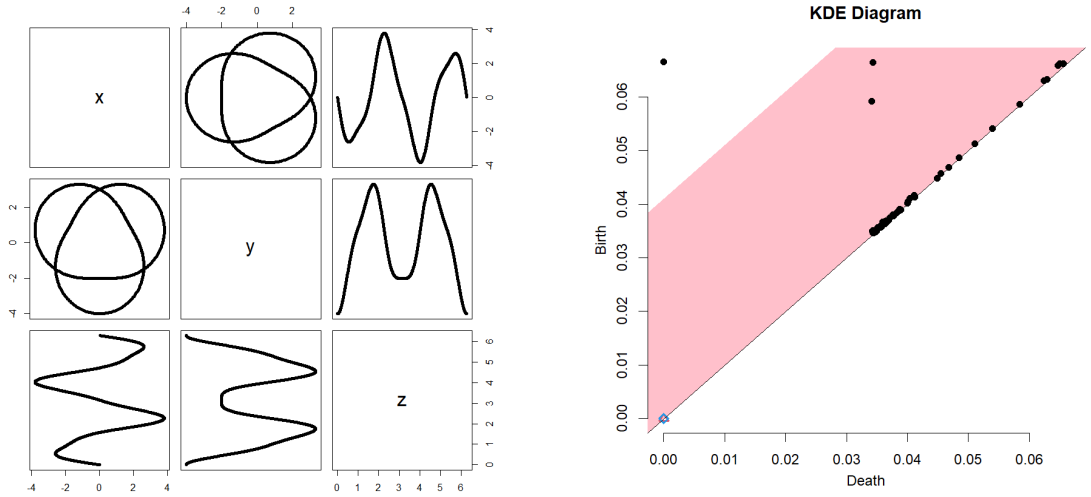

Figure 31: A clean trefoil and its KDE diagram.

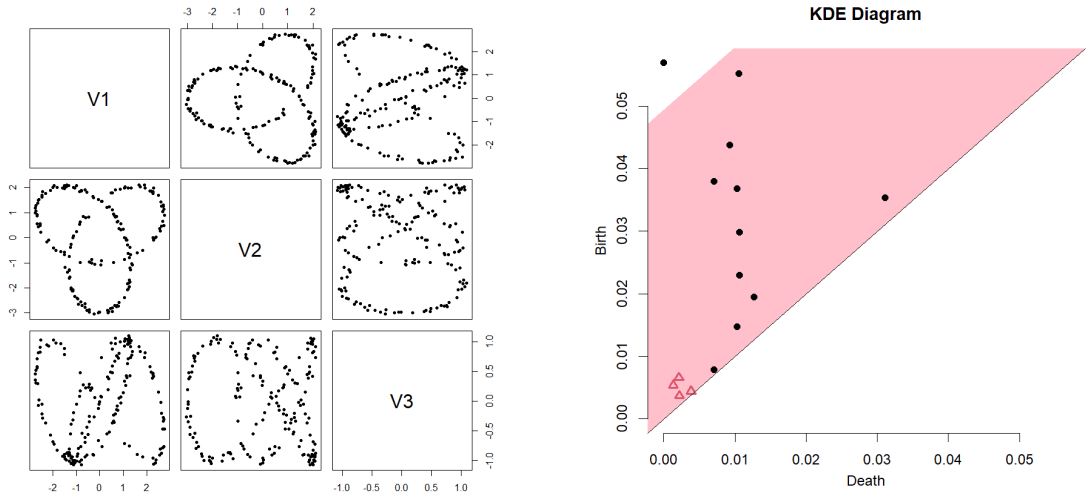

Figure 32: A noisy trefoil and its KDE diagram.

now-familiar KDE diagrams; one can see the noise cluttering up the integrity of the shape in Figure 32. Notice the representation here takes the names of each column– V1, V2, and V3– as the names of the dimensions. When implementing this tool on real-world data, one could easily display the names of their columns in the center for reference.

Both KDE diagrams give the same end result, which is heartening, but the outcast outliers

reveal how close they came to failing. In the clean trefoil, there are a couple H_1 points which stand out, and very close to the origin there is an H_3 hole. In the noisy trefoil, no H_3 holes are found, but several H_2 holes are. Also, the H_1 points are much livelier than their clean-cut counterparts in Figure 31. In this simple example the end result is the same, but the noise had a clear impact of the quality of the diagram. It reinforces the good habit of cleaning one's data well, especially using real-world data (which is always more chaotic than a generated dataset).

6.4 Data in the Wild

Now we leave any safety of grounded, prescient knowledge and search for a real-world dataset to explore. We kick off toward the deeper waters where all sorts of beastly datasets lurk in repositories across the internet, waiting to be found and scrutinized. The website Kaggle, in particular, proudly hosts many kinds of datasets suitable for projects.[40] At the beginning of this endeavor, a dataset concerning Turkish music and its statistics was chosen as the real-world set to explore.[41] The rest of this section discusses our findings after applying `TDA` to this dataset.

The Turkish music dataset, now referred to as the Turkish set, contains 400 pieces of music grouped by emotion with fifty statistics to compare, including energy, fluctuation, brightness, and roughness means. The emotions– happy, sad, relaxed, and angry– offer a potentially excellent structure to test persistence's strengths in categorization and noting anomalous clusters in data. Surely different moods of music have some difference in their statistics. With fifty parameters, it would be difficult to check except by tedious traditional means, but `TDA` may be able to find

40. Joakim Arvidsson, *Turkish Music Emotion*, Kaggle, Attribution 4.0 International (CC BY 4.0), October 2023.
41. Mehmet Bilal Er, *Turkish Music Emotion*, UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C5JG93, 2023.

those differences for us.

The original aim for this dataset was to introduce a curated subset which contained only the relaxed and angry pieces. By selecting the most polarized moods of music, perhaps `TDA` could separate them by mood if given a random selection. Additionally, if it received a subset containing mostly one mood, we predicted it would be able to notice which points were the other mood (the anomalies). Perhaps we could even dip our toes into the machine learning space. If we feed our program a curated training subset of the Turkish set and teach it the responses we expect to see by adjusting parameters, we could then feed it a randomized subset of the Turkish set to see if it has learned enough to accurately analyze the subset without adjustment.

Before we were able to test this, we first had to clean the data. Each column's range of values differed wildly, but otherwise the Turkish set was clean and organized. The only tidying we did was normalizing it so each column's range was contained in the interval [0,1]; this made it much easier to adjust parameters in the code. After selecting only the angry and relaxed music pieces, we stripped the mood column to remove `TDA`'s ability to classify the entries directly by that parameter. If it could classify data by finding deeper gaps and patterns, it would not need to be told their moods in advance.

Once we had a tidy dataframe (by using several commands in the popular `tidyverse` library), we downsampled it to seventy percent. The remaining thirty percent would be reserved as a blind test, if we had the time. The resultant 140 x 50 dataframe, called `train` for ease, was the subset we explored. The remaining hopeful test data was not used, and from here on the `train` set is what is referenced when we say 'the data.'

We began by testing just two columns of the data. After troubleshooting the usual beginning errors in any new program, we received results. They were uninteresting, but confirmed

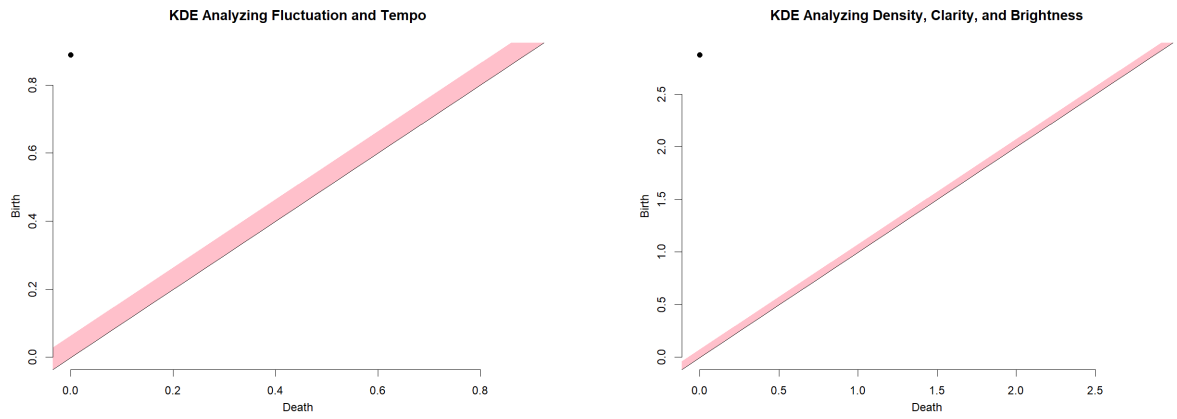

Figure 33: KDE diagrams of curated two-column and three-column subsets, respectively.

`TDA`'s readiness to handle more. Three columns took longer to compute, but had no further issues and was similarly dull. Feeling confident, we jumped ahead to ten columns and were promptly met with a session-terminating error. Truncating the sample size (number of rows) did nothing to fix it. By running several tests at lower dimensions, we discovered an obstacle which put our original plan far out of reach: the increasing computational price. Three columns could be processed by less than 1 gigabyte of RAM, but four columns took about 11 gigabytes. Trying to process five columns showed us the limit of our abilities: it asked for nearly 4 terabytes of RAM. This price did not change, no matter how few points we directed it to process. We were unable to accomodate such a vast demand and decided to pivot our aims.

The entirety of the data held a full fifty columns of data, which we now know to be laughably expensive to compute all at once. Instead, we explored this music data as much as possible with the time and tools we had. With four columns as our ceiling, we processed the data in sections to see if any consecutive statistics held relationships worth noting. The results were inconclusive; all were nearly featureless, with the only difference between them being the size of their confidence bands. This may speak to the level of uniformity in the density of each section,

but gives no significant feedback.

The preliminary tests were run at the usual `h` = 0.3 parameter. A sample of tests were repeated at `h` = 0.1 to ensure features were not being smoothed over, with the same results. Unfortunately, we were unable to find any subset, curated or random, which returned any significant features aside from the single H_0 point. Recall that point suggests the filtration process found one 'connected' point cloud, an evenness in proximity which has no notable gaps or holes.

Perhaps the Turkish set was a poor choice. Perhaps there really are no holes, or they are hiding in higher dimensions than we can reach. The exploration was valuable in its own right. We had no reference for the limit of this tool, hence the naive choice of a fifty column dataset, and we explored that on both a personal machine and a cluster server. The size limit was the same for both due to the exponential nature of the processing demand. The set remains an interesting data science challenge with many other possible approaches. One could curate pieces with extreme values in a particular column, artificially creating a gap, and see if `TDA` finds any other related gaps or features. One could add a for loop to test all possible combinations of four column subsets, giving every possible chance to find significant features. One could plot linear regressions of columns and select those which have more or less correlation to each other. Any of these may coax some more answers out of the data, if there are any to be found.

We came to a somewhat anticlimactic end, but it was not without educational merit. `TDA` can not only be used by any aspiring data explorer, it is changing as we speak. Its last update came eight weeks ago as of this date and its creators have been active in the TDA sphere for a decade with no sign of stopping. Resources abound for persistent homology, including similar

TDA libraries available in other languages like Python and C++.[42] The Turkish set is not a failure. It is merely an unfinished project.

42. Morozov, *A Practical Guide to Persistent Homology*.

7 Conclusion

As a practice, TDA has several different tools in development and has been applied successfully to the fields of medicine, multivariate time series analysis, sensor networks, genomic studies, and more[43] in its short history. Using it in tandem with machine learning to help train artificial intelligence models is still a hotbed of research activity.[44] In the age of 'Big Data,' such a tool is very useful indeed. Nearly everything can be recorded in data at this point. If a sort of order can be formed upon it, TDA can filtrate and analyze it, and that includes qualitative data. Indexing the qualitative traits gives an order to them and they can be treated like any other dimension of space. Even continuous material like sound files can be stratified by large-scale sampling or extracting features, like the Turkish set exhibited.

TDA's close link to machine learning offers another ocean of possibility. Artificial intelligence has radically changed the landscape of our society and has become the biggest technological advancement since the internet. Pure math got little recognition for its contributions compared to computer and data sciences until TDA came along. Now, previous obstacles for artificial intelligence like object recognition (both overhead[45] and in 3D[46][47][48]) are seeing rapid progress thanks to the workings of TDA.

Communities and research groups have sprung up all over within the last decade to jump on this bandwagon and make use of its more abstract (and often more generalizeable) concepts.

43. Perea, "A Brief History of Persistence."
44. Frederic Chazal and Bertrand Michel, "An Introduction to Topological Data Analysis: Fundamental and Practical Aspects for Data Scientists," Hosted by Cornell University, *ArXiv*, February 2021,
45. John Roach, "Topology, Algebra, and Geometry Give Math Respect in Data Science," November 2023,
46. Chazal and Michel, "Persistent Homology in TDA."
47. Singh, Memoli, and Carlsson, "Topological Methods for the Analysis of high Dimensional Data Sets and 3D Object Recognition."
48. Wasserman, *Topological Data Analysis*.

The Topology, Algebra, and Geometry in Data Science community, for example,[49] encourages newcomers and mathematicians alike to explore TDA and is going strong after being founded in 2022. The Banff International Research Station is hosting a week-long workshop in Alberta on Representation Theory and Topological Data Analysis starting in April.[50] This trend is only gaining momentum as artificial intelligence becomes more sophisticated and demands a similar increase in its education.

There are some who think TDA is a flash in the pan.[51] Its practical application has yet to be proven and the obsacles may prove too much for it. Some avenues are already losing support, as seen in Mapper's removal from The Comprehensive R Archive Network due to lack of maintenance. But for as many naysayers, there are two more TDA enthusiasts ready to help it succeed. Though warranted, skepticism ought to serve as further motivation to prove TDA's possibilities as a useful technological advancement. What the future holds for TDA is murky to all but the most informed, but with its rapidly changing landscape, things could be completely different in as little as a decade from now.

49. Roach, "Topology, Algebra, and Geometry Give Math Respect in Data Science."
50. Banff International Research Station, *Representation Theory and Topological Data Analysis (24w5241)*, Cornell University, Attribution 4.0 International (CC BY 4.0), 2023.
51. Wasserman, *Topological Data Analysis*.

REFERENCES

Arvidsson, Joakim. *Turkish Music Emotion*. Kaggle. Attribution 4.0 International (CC BY 4.0), October 2023.

Brown, Ronald. "Modelling and Computing Homotopy Types: I." Hosted by Cornell University, *ArXiv*, September 2022.

Chazal, Frederic, and Bertrand Michel. "An Introduction to Topological Data Analysis: Fundamental and Practical Aspects for Data Scientists." Hosted by Cornell University, *ArXiv*, February 2021.

———. "Persistent Homology in TDA." Geometrica, *Inria*, June 2016.

Dhingra, Bhuwan, and Chaitanya Ahuja. "Statistical Topological Data Analysis." 30th Conference on Neural Information Processing Systems, December 2016.

Edelsbrunner, Herbert, and John Harer. *Computational Topology: An Introduction*. 177–208. Departments of Computer Science and Mathematics. Durham, North Carolina: Duke University, 2010.

Er, Mehmet Bilal. *Turkish Music Emotion*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5JG93, 2023.

Fasy, Brittany T, Jisu Kim, Fabrizio Lecci, Clement Maria, David L. Millman, and Vincent Rouvreau. "Introduction to the R Package TDA." CMU TopStat Group, January 2024.

Frosini, Patrizio. "Measuring Shapes by Size Function." DOI: https://doi.org/10.1117/12.57059, *Intelligent Robots and Computer Vision X: Algorithms and Techniques* (Boston, Massachusetts) 1607 (February 1992).

Ghrist, Robert. *Foundations of Topological Data Analysis*. YouTube, July 2023.

Hatcher, Allen. *Algebraic Topology*. New York, New York: Cambridge University Press, 2002.

Jagadeesan, Ravi, and Luke Sciarappa. *Simplicial Homology*. MIT Mathematics. Fourth Annual MIT Primes Conference, May 2014.

Morozov, Dmitriy. *A Practical Guide to Persistent Homology*. Lawrence Berkeley National Lab, March 2023.

Oulhaj, Ziyad, Mathieu Carriere, and Bertrand Michel. "Differentiable Mapper for Topological Optimization of Data Representation." DOI: https://doi.org/10.48550/arXiv.2402.12854, *ArXiv*, February 2024.

Perea, Jose A. "A Brief History of Persistence." DOI: https://doi.org/10.48550/arXiv.1809.03624, *ArXiv*, October 2018.

Roach, John. "Topology, Algebra, and Geometry Give Math Respect in Data Science," November 2023.

Singh, Gurjeet, Facundo Memoli, and Gunnar Carlsson. "Topological Methods for the Analysis of high Dimensional Data Sets and 3D Object Recognition" (Prague, Czech Republic), September 2007.

Station, Banff International Research. *Representation Theory and Topological Data Analysis (24w5241)*. Cornell University. Attribution 4.0 International (CC BY 4.0), 2023.

Toda, Hiroshi. *Composition Methods in Homotopy Groups of Spheres*. ISBN 0-691-09586-8. Princeton University Press, 1962.

Wasserman, Larry. *Topological Data Analysis*. Department of Statistics. Pittsburgh, Pennsylvania: Carnegie Mellon University, 2016.

Wildberger, NJ. *Algebraic Topology: a Beginner's Course*. YouTube. University of New South Wales. Sydney, Australia, 2012.