Topological Data Analysis of Open-Ended Responses

Bright Effah

A thesis presented in fulfilment of the requirements for the degree of Master of Science

Supervisor: Dr. Alexandre Karassev

Department of Mathematics and Computer Science

School of Graduate Studies

Nipissing University

North Bay, ON−Canada

November, 2022

Table of Contents

Acknowledgements

Firstly, I would like to thank God Almighty for His Grace this far and also express my sincere gratitude to my thesis supervisor, Dr. Alexandre Karassev. His guidance and support in all aspect helped me to finish this research work. This thesis would not be possible without his constant motivation, understanding and patience.

Secondly, I would like to thank the rest of my committee members: Dr. Murat Tuncali, Dr. Tzvetalin Vassilev and Dr. Henry Adams. I appreciate their insightful comments and encouragement throughout my thesis and stay for my masters. My sincere thanks also goes to my friends: Luke Cooper, Mohammedamin Sabzevari and Megan Pulford. They supported and inspired me in the past semesters.

Finaly, I would like to thank my parents Charles Effah and Comfort Saah, for supporting me throughout my life. Not forgetting of my dear friend Josephine Nunoo, Maria Cooper and the Ankomah family for their support as well.

Abstract

Topological data analysis is a new and emerging field of mathematics in which topology and geometry are applied to problems in data analysis. In this research, we will study two powerful tools from topological data analysis called Persistent Homology and the Mapper algorithm.

We discuss the theory of simplicial complexes from group theory, a particular type of homology theory called simplicial homology that captures n-dimensional 'holes', and also walk through the Mapper algorithm pipeline for visualizing and comparing features in datasets.

We explored the responses of age "16-99" on how they define "Loneliness". Each response was considered as a point to form a dataset in 2D. The age group category was analysed with Persistent Homology using the Vietoris Rips and bottleneck distance. The computation was visualised on the Persistent Diagrams which leverage on the bottleneck distance to compare the Persistence Diagrams of different datasets from the age category.

Lastly, those who are lonely and people living alone datasets were extracted separately and applied on the Mapper algorithm to reduce the dimensionality and clustered the datasets to show similarities and outliers in the responses. We also retrieved some responses from the clusters to compare with others clusters.

Keywords*: Simplicial Complex, Persistent Homology, Persistent Diagram, Bottleneck Distance, Mapper, Filter function, Clusters(nodes) and Edges*

1 Chapter 1: Introduction

In our everyday life, we produce information anytime, anywhere and to anyone. Thinking about this scenario, you could realise the volume of information or data one produces daily. Specifically, with text documents or information in text form, we have seen the growing rate of data and increase in technology with the overload of information through the internet and other digital channels around the globe. In order to make our life easier and having a good sense of what is happening in our world, what the threats and opportunities are, we need to collect and analyse this huge amount of data [1]. In collecting and analysing these data, many of these information or text data goes through computer coding but the coding is performed by human experts in the field of such documents. Research on clustering texts and other non-structured data is sparse and we have been able to find only a few research papers in which open responses from education-related questionnaires were clustered [2]

In 2018, the BBC Loneliness Team conducted a survey on how older people talk about loneliness. These were open-ended responses from 52,302 participants. Due to the volume and the presentation of information in this survey, many statistical algorithms failed to analysis the data. Considering the use of clustering techniques and other ideas from areas such as computer science, machine learning, and uncertainty quantification — along with mathematical and statistical models — are often very useful for data analysis [3]. In general, classification algorithms are either supervised or unsupervised. A supervised classification algorithm learns from a training dataset that has been manually classified into a finite number of categories or classes, and the goal is to define rules that can classify our responses data points. Due to the size of our data, we could not do any manual classification. There are several approaches that could be used for supervised classification, for example linear or quadratic discriminant analysis. However, we will look at the unsupervised classification since we do not have any categories for our data. In unsupervised classification or clustering, no labelled data is available; instead the goal is to organize

observations into clusters based on a notion of similarity. There are many hierarchical clustering algorithms including single linkage, complete linkage and average linkage. After analysing various types of clustering algorithms, KMeans clustering was chosen since it had minimum number of clusters for big data.

The KMeans algorithm provided four common words among the five clusters we selected which is feel, share, people and talk. These result was not obvious, since the Yet Another Keyword Extractor(YAKE) showed more common words among the responses. The similarities or differences in their responses could not be captured by the KMeans clustering as desired. According to [2] clustering open-ended questionnaire responses and other short educational text is important but a little researched problem. This is true because clustering open ended responses is a difficult task in general.

In the past two decades, several algorithms were proposed that can be used to capture shapes and features of large data sets. These algorithms have been an interesting development from the field of Pure Mathematics which has given rise to Topological Data Analysis (TDA). Because TDA has its fundamental concept stemming from geometric approach to pattern or shape recognition within the dataset, we can trace its root in numerous findings from algebraic topology and computational geometry. Over the last 24 years, there has been a concerted effort to adapt topological methods to various applied problems, one of which is the study of large and high dimensional datasets [4]. High dimensionality of data poses as one of the main challenges confronting data. As high dimensional datasets are usually represented by data clouds, data mining methods, especially distance-based algorithms, are applied to study the data cloud and reveal, analyse, and summarize the relations among the points in the cloud [5]. TDA algorithms can in particular, reduce dimensionality to analyse these datasets .

TDA has been used in many fields of research to capture shapes and understand relationship between datasets, but we are going to look at the papers related to text analysis: “The shape of an image: A study of mapper on images” [6], “TDA

in text classification and face detection" [7], "Topological Data Analysis in text processing" [5], "An introduction to a new text classification and visualization for natural language processing using topological data analysis" [1], and "Persistent Homology: An introduction and a new text representation for natural language processing" [8]. The papers mentioned above have been worked on using TDA but were just text documents slightly different from open ended responses where we seek to capture individual ideas. We later found a paper "In search of distinct graduate admission strategies in physics: An exploratory study using topological data analysis" [9] which was analysed with responses but less than 200 participants. The work also focused on 30 sets of question and [9] noted that they performed their clustering based on the numerical responses given to a single question.

This shows minimum research in the area of TDA on open ended responses, although [9] has indicated in future directions that further analysis of the results from Mapper should involve analysing the non-numerical responses on the survey using other methods. The same method could be used if only we are able to transform these words into some form of point cloud data or matrix. We will use their age, how they live and also question on whether they are lonely or not to examine the shape and relationships between their responses. We select Persistent Homology and Mapper from TDA to analysis this data. The process will be well illustrated after we learn about the algorithms. As a branch of topological data analysis, persistent homology has the advantage of capturing novel invariant structural features of documents [8]. Mapper was proposed by [10] as a method to understand the shape of data using a topology-inspired construction.

This thesis is divided into five chapters. Chapter one contains necessary background information and outlines the problem statement as well as objectives of the research. The second and third chapters focus on the detailed explanation of the two algorithms, with all necessary definitions and statements. The fourth chapter describes the process that the text data (responses) will go through to output numbers as a representation of the words in the text data. The fifth chapter interprets

and discusses the results.

2 Chapter 2: Persistent Homology

Having introduced the notion of TDA, we now look at the development of systems we need for characterizing topological spaces. Group theory on the other hand provides us with tools to establish equivalence relations using homomorphisms and factor groups. Unlike homeomorphy and homotopy, homology is discrete by nature. As such, it is the basis for this section in serving as a tool to define homology as a topological classification system.

Homology groups present a mathematical language for the loops/holes in a topological space. They may capture holes surprisingly well, by focusing on what is around them. Persistent homology incorporates both approximation approach from computational geometry and homology groups from algebraic topology to detect features of a space at different scales. In algebraic aspect, we turn the function into measurements and in geometry, we define a function on a topological space, and measurements make sense only if the function does.

2.1 Group Theory

Definition 2.1.1 *A group is a nonempty set G on which there is a binary operation η on G which is a function $\eta : G \times G \longrightarrow G$, denoted by $\eta(a, b) = a * b$, such that the following properties are satisfied:*

1. *If a and b belong to G, then $a * b$ is also in G(closure);*
2. *$(a * b) * c = a * (b * c)$ for all $a, b, c, \in G$ (associativity);*
3. *there exists an element $e \in G$ (often denoted by 1 if it does not cause ambiguity) such that $e * a = a * a = a$ for all $a \in G$ (identity);*
4. *For every $a \in G$, there is an element $a^{-1} \in G$ such that $aa^{-1} = a^{-1}a = e$ (inverse).*

We consider example under the operational sign + (in place of $*$) to be a binary operation defined on the integers $\mathbb{Z}$. Instead of writing $+(2, 7) = 9$ we write $2 +$

$7 = 9$. Actually, the binary operation η is usually thought of as multiplication (or addition for commutative groups) and instead of $\eta(a, b)$, we use notation such as $ab, a + b, a \cdot b$ and $a * b$. If the set G is a finite set of n elements, we can present the binary operation, say $*$, by an n by n array called the multiplication table. If $a, b \in G$, then the (a, b)–entry of this table is $a * b$.

Definition 2.1.2 *A group $(G, *)$ is abelian if the binary operation $*$ is commutative, ie., $a * b = b * a$ for all elements $a, b \in G$.*

In this work, we only consider abelian groups. Some common examples of abelian groups under addition are the integer $\mathbb{Z}$, the real number $\mathbb{R}$, the complex number $\mathbb{C}$, the rationals $\mathbb{Q}$, and the integers modulo m denoted $\mathbb{Z}_m$. A classical example of a non-abelian group is the group of $n \times n$ invertible matrices under matrix multiplication.

Definition 2.1.3 *A subset $H \subseteq G$ of a group $(G, *)$ is a subgroup of G if $(H, *)$ is closed under $*$ and taking inverses.*

The subgroup consisting of the identity element of G, $\{e\}$ is the trivial subgroup of group G (all other subgroups are nontrivial). The set of positive reals$(\mathbb{R}_+, \times)$ is a subgroup of $(\mathbb{R}\backslash 0, \times)$ by restricting multiplication to positive numbers. Note however that the set of negative reals $(\mathbb{R}_-, \times)$ is not a subgroup because it is not closed with respect to $\times$.

Definition 2.1.4 *Let H be a subgroup of group G. For $g \in G$, the subset $gH = \{gh \mid h \in H\}$ of G is the left coset of H containing g and $Hg = \{hg \mid h \in H\}$ is the right coset of H containing g.*

For an Abelian subgroup H of G, $gh = hg, \forall g \in G$, $h \in H$, so the left and right cosets match. We may easily show that every left coset and every right coset has the same cordinality by constructing a 1-1 map of H onto a left coset gH of H for a fixed element g of G. For example, $\{0, 2\}$ is a subgroup of $\mathbb{Z}_4$. The coset of 1 is $1 + \{0, 2\} = \{1, 3\}$. The sets $\{0, 2\}$ and $\{1, 3\}$ exhaust all of $\mathbb{Z}_4$. As $\mathbb{Z}_4$ is Abelian, the sets are both the left and right cosets.

2.1.1 Group Characterization

Our goal for this chapter is to fully understand the structure of certain finite groups that are generated in our study of homology. Due to our goal and interest in characterizing the structure of groups, we define maps between groups to relate their structures.

Definition 2.1.5 *Two groups G and H are said to be isomorphic if there is bijective function $\theta : H \longrightarrow G$ such that:*

$$\theta(g_1 g_2) = \theta(g_1)\theta(g_2) \tag{1}$$

for all $g_1, g_2 \in G$. The function θ is called an isomorphism.

If θ satisfies only the equation1, then θ is named a homomorphism, as defined below.

Definition 2.1.6 *A map ϕ of a group G into a group G' is a homomorphism if $\phi(ab) = \phi(a)\phi(b)$ for all $a, b \in G$. For any groups G and G', there is always at least one homomorphism $\phi : G \longrightarrow G'$, namely, the trivial homomorphism defined by $\phi(g) = e'$ for all $g \in G$, where e' is the identity in G'.*

Definition 2.1.7 *Let $\phi : G \longrightarrow G'$ be a homomorphism. ker $\phi = \{g \in G \mid \phi(g) = e'\}$. In other words, the kernel is the set of elements that map to identity.*

Theorem 2.1.1 *Let ϕ be a homomorphism of a group G into a group G'*

1. *If e is the identity in G, then $\phi(e)$ is the identity e' in G'.*
2. *If $g \in G$, then $\phi(g^{-1}) = \phi(g)^{-1}$.*
3. *If H is a subgroup of G, then $\phi(H)$ is a subgroup of G'.*
4. *If K' is a subgroup of G', then $\phi^{-1}(K')$ is a subgroup of G.*

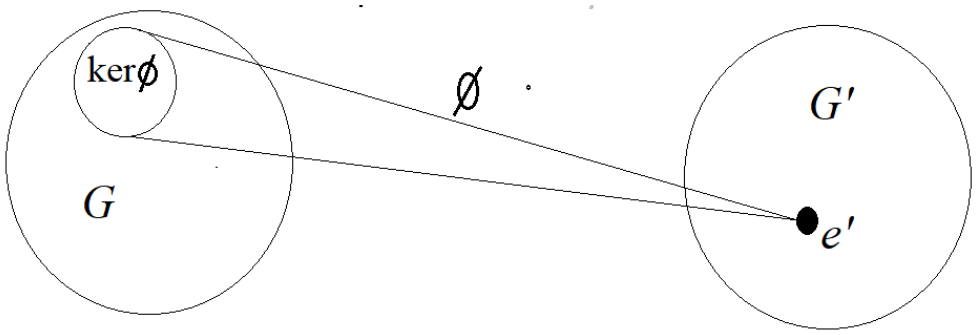


Figure 2.1.1: A homomorphism $\phi : G \longrightarrow G'$ and its kernel.

Homomorphisms in another way define a special subgroup in their domain. Note that kerϕ is a subgroup by an application of theorem 2.1.1 to the fact that $\{e'\}$ is the trivial subgroup of G'. So, we may use it to partition G into cosets.

When we consider $(H, *)$ to be a subgroup of an abelian group $(G, *)$, we can introduce a new binary operation $\star$ not on the elements of G but on the cosets of H : $(a * H) \star (b * H) = (a * b) * H$, $\forall a, b \in G$. The operation $\star$ is well-defined and does not depend on the particular choice of representer.

Definition 2.1.8 *The cosets* $\{g * H \mid g \in G\}$ *under the operation* $\star$ *form a group, called the quotient group* G/H.

It is necessary to consider quotient groups as "higher level" groups defined on the squares in the previous picture kerϕ which is the smaller circle inside G of Figure 2.1.1 as a subgroup of G. The elements forming the quotient group $G/ker\phi$ are the cosets of ker ϕ, that is all the other equally smaller circles we could draw in G but outside of the the circle containing ker ϕ. For example, let ϕ: $(R \setminus \{0\}, \times) \to (\{1, -1\}, \times)$ be a homomorphism of the group of non-zero reals to $\mathbb{Z}_2$ (in multiplicative notation), sending all positive reals to 1 and all negative integers to -1. Then ker$\phi = R_+$, and there are two cosets: R_+ and R_-. It is easy to see that the quotient group $(R\{0\})/R_+$ is isomorphic to $(\{-1, 1\}\times)$. This is a special case of the following standard theorem.

Theorem 2.1.2 *Let $\phi : G \to G'$ be a group homomorphism with kernel H'. Then $\phi(G)$ is a group and the map $\varphi : G/H \to \phi(G)$ given by $\varphi(aH) = \phi(a)$ is an isomorphism. Let $\gamma : G \to G/H$ be the canonical map given by $\gamma(g) = gH$, then for each $g \in G$ and $\phi(g) = \varphi\gamma(g)$. γ is also the corresponding homomorphism.*

Definition 2.1.9 *Let $S \subset G$. The subgroup generated by S, $\langle S \rangle$, is the subgroup of all elements of G that can expressed as the finite products of elements from S and their inverses.*

It is easy to check that $< S >$ is indeed a subgroup.

For example, $\mathbb{Z}$ is itself the subgroup generated by 1, the group of even integers is the subgroup of $\mathbb{Z}$ generated by 2.

Definition 2.1.10 *The rank of a group G is the size of the smallest subset that generates G.*

For example, $\text{rank}(\mathbb{Z}) = 1$ since $\mathbb{Z} =< 1 >$, and $\text{rank}(\mathbb{Z} \times \mathbb{Z}) = 2$ since $\mathbb{Z} \times \mathbb{Z} =< \{(0,1),(1,0)\} >$. Here, there is no one-element generating set for $\mathbb{Z} \times \mathbb{Z}$. Group theory is important because when counting "holes/loops" in homology, G will be the group of cycles. The first thing we do to compute homology groups of a space is to approximate the space by a simpler structure called simplicial complex which will help us to derive some meaningful computation from it.

2.2 Simplicial Complex

In general, we are not able to represent topological spaces in our computer systems precisely due to the finite storage for the data structures. We therefore sample finite set of points of the space to represent the topological space by a triangulation. A triangulation gives rise to a simplicial complex, a combinatorial structure that can represent a surface of a space.

We begin with the definitions of simplicial complexes that capture both geometry and topology as it was mentioned in the introduction of this work. We start the

definitions with linear combinations since it allows us to represent regions of space with very few points.

Definition 2.2.1 *Let* $S = \{p_0, p_1, ..., p_k\} \subseteq \mathbb{R}^d$. *A linear combination is* $x = \sum_{i=0}^{k} \lambda_i p_i$, *for some* $\lambda_i \in \mathbb{R}$. *An affine combination is a linear combination with* $\sum_{i=0}^{k} \lambda_i = 1$. *A convex combination is a an affine combination with* $\lambda_i \geq 0$, *for all* i. *The set of all convex combinations is the convex hull of* S *and is denoted by* $conv(S)$.

Definition 2.2.2 *A* k*-simplex* σ *is the convex hull of* $p + 1$ *affinely independent points* $\{x_0, x_1, ..., x_p\} \in \mathbb{R}^d$. *We denote* $\sigma = conv\{x_0, x_1, ..., x_p\}$. *The dimension of* σ *is* p.

In the above definition, affinely independent means the p vectors $x_i - x_0$ for $i = 1, ..., p$ are linearly independent, i.e., they are in general position. The convex hull is simply the solid polyhedron determined by the p+1 vertices. For a k-simplex σ, we also write $\dim\sigma = k$. We show low-dimensional simplices with their names where 0-simplex is a node(vertex)$\{A\}$, 1-simplex an edge$\{A, B\}$, 2-simplex a triangle$\{A, B, C\}$, and 3-simplex a tetrahedron$\{A, B, C, D\}$ as shown in the Figure 2.2.1 below.

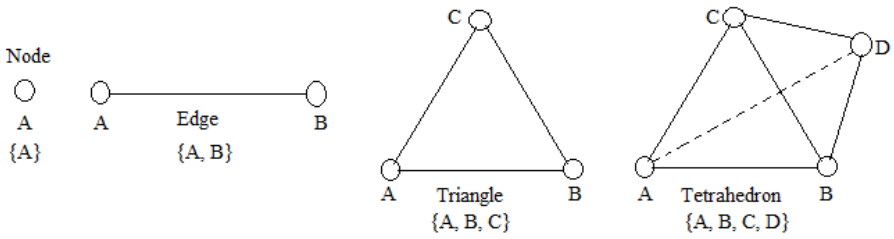


Figure 2.2.1: From left to right: a node(vertex), an edge, a triangle, and a tetrahedron. Also an edge has two vertices, a triangle has three edges and tetrahedron has four triangles as faces

.

Definition 2.2.3 *A face of* σ *is conv*S *where* $S \subset \{x_0, ..., x_p\}$ *is a subset of the* $p +$ *1 vertices.*

Looking at tetrahedron, it has four triangle faces corresponding to the four subsets of S obtained by removing one vertex at a time from σ. These four triangle faces are also 2-simplices themselves. It also has six edge faces and four singleton vertex faces.

A k-simplex has $\binom{k+1}{l+1}$ faces of dimension l and $\sum_{l=-1}^{k}\binom{k+1}{l+1} = 2^{k+1}$ faces in total. A simplex, therefore can be large depending on the kind of points you are visualizing but very uniform and simple combinatorial object.

Definition 2.2.4 *A simplicial complex K is a finite collection of simplices such that $\sigma \in K$ and τ being a face of σ implies $\tau \in K$, and σ, $\sigma' \in K$ implies $\sigma \cap \sigma'$ is either empty or a face of both σ and σ'.*

Definition 2.2.5 *If τ is a face of σ we call σ a coface of τ. A simplex in a simplicial complex K is principal if it has no proper coface in K.*

In the above definition, *proper* has the same meaning as for sets.

In simple terms, a simplicial complex is a collection of simplices that fit together nicely, as shown in the left side of Figure 2.2.2 below, as opposed to simplices in the right picture.

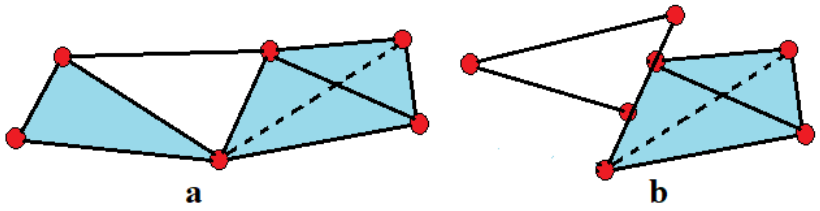


Figure 2.2.2: The figure labeled a is a simplicial complex while the figure labeled b is not

.

In Definition 2.2.4, the first condition says that if a simplex, e.g. a triangle, is in K, then its faces, that is, its edges and vertices, need to be in K as well. The second condition says that we can only glue simplices by their common faces. For example, we can glue two triangles by a common vertex or a common edge but cannot glue a vertex of a triangle on one of the edges of the other triangle. The Figure 2.2.2 on

the left is an example of a simplicial complex whereas the Figure 2.2.2 on the right is not a simplicial complex since it violates the condition in Definition 2.2.2.

Simplicial complexes are specifically well-suited for practical computations, since they can easily be represented for example by listing their vertices, edges, triangles, tetrahedra etc. Note that the underlying space of a simplicial complex can be viewed as a subspace of $\mathbb{R}^d$ for sufficiently large d (indeed, if n is the total number of 0-simplices, i.e. vertices, of the simplicial complex, take $d = n + 1$.) Therefore any simplicial complex can be viewed as a topological space.

Definition 2.2.6 *A representation of a simplicial complex K as a subspace of $\mathbb{R}^d$ is called geometric realization of K and is often denoted by $|K|$.*

In what follows we often will not make significant distinction between K and its geometric realization, as soon as it does not cause ambiguities.

Many topological spaces are homeomorphic to the underlying space of a simplicial complex, so we may be able to compute properties of a space by computing it on such a complex. Such a homeomorphism is called a triangulation.

Definition 2.2.7 *A triangulation of a topological space X is a simplicial complex K, with a homeomorphism $|K| \to X$. If such a triangulation exists, X is said to be triangulable, and we say that K triangulates X.*

The concept of triangulations will help motivate the use of simplicial homology, and indeed of persistent homology. While we will define homology using simplicial complexes, in order to exemplify how data analysis is often done we need the concept of an abstract simplicial complex [11].

Applying homology calculation to a single, static simplicial complex yields information about clusters, holes, cavities, etc., and a count of the homology classes in each dimension for the Betti numbers (defined later in the text) precisely summarizes those features in the simplicial complex. In a simplicial complex, we consider the holes as voids bounded by simplices of different dimensions. In dimension 0, they are connected components, in dimension 1, they are loops bounded by edges

(1-simplices), in dimension 2, they are holes bounded by triangles (2-simplices) and in general, in dimension p, they are the holes bounded by $p-$simplices.

Notation 2.2.1 *$(H, *)$ is the homology functor from the cluster (category) of simplicial complexes and simplicial maps to the cluster of abelian groups and homomorphisms.*

2.3 Simplicial Homology

In simplicial homology, we focus on finding the holes in a simplicial complex. To understand what simplicial homology is, we need to define the chains, and two special types of chains, namely cycles and boundaries. In calculating homology groups, the steps involved rely on defining a boundary map ∂_p that goes from chain groups $C_p \to C_{p-1}$. The definitions below will clarify these components.

Definition 2.3.1 *The p-dimensional chains of simplicial complex K are formal linear combinations of p-dimensional simplices in K. The notation for this is*

$$c = \sum \alpha_i \sigma_i \tag{2}$$

the above Definition 2.3.1, the σ_i's are the p-simplices and the α_i are the coefficients. With our computation of homology, we will be working with α_i that take values of either 0 or 1, called modulo 2 coefficients. The choice of coefficient to define homologies can be more complicated. For example, one can use integers $\mathbb{Z}$, rational numbers $\mathbb{Q}$, real numbers $\mathbb{R}$, or in general elements of a field, a ring, or an arbitrary abelian group. For our purposes, it will be sufficient to consider coeffcients modulo 2 (i.e. elements of $\mathbb{Z}_2$). Thus, our chain above can be identified with the set of those p-simplices σ_i for which $\alpha_i = 1$.

For example, consider a tetrahedron to be K. By definition the triangle with four faces (i.e., 2-simplices) are in K. A 2-chain is a subset of these four triangles, e.g., all four triangles, the bottom triangle face only, or the empty set. There are 2^4 distinct 2-chains. Similarly, by definition all six edges of the tetrahedron are in K,

too. Thus, there are 2^6 distinct 1-chains. Despite the name "chain", a p-chain does not have to be connected. The figure below shows a 1-chain (the red edges) on the left and a 2-chain on the right:

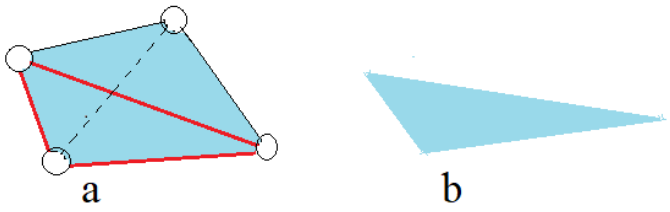


Figure 2.3.1: shows a 1-chain (the red edges) in **a** on the left and a 2-chain (triangle as the face)in **b** on the right.

Definition 2.3.2 *A chain group C_p is the set of p-chains of the simplicial complex K. That is, p-chains together with the addition operation form the group of p-chains denoted as $C_p = C_p(K)$. Namely, if $c = \sum \alpha_i \sigma_i$ and $c' = \sum \beta_i \sigma_i$, we let $c + c' = \sum(\alpha_i + \beta_i)\sigma_i$, where the coefficients satisfy $1 + 1 = 0$.*

In set notation, the sum of two p-chains is their symmetric difference. The neutral element is 0 the empty sum. The inverse of c is $-c = c$ since $c + c = 0$. Obviously, C_p is abelian. We can now assign a group of p-chains for each integer p. For $\dim K = p < 0$ this group is trivial, consisting only of the neutral element.

In the example below, duplicate 1-simpicies cancel out, and when adding two p-chains we get another p-chain.

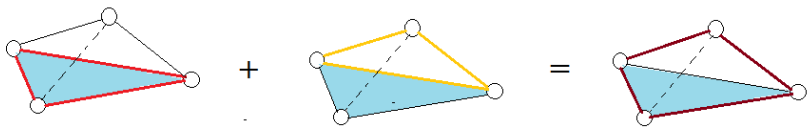


Figure 2.3.2: shows a 1-chain addition

The map ∂_p between the chain groups, that we will define next, and the kernel and image is the linear transformation that describe cycles and boundaries, respectively, which can be used to identify the homology classes. Denote the cycles as $\ker \partial_p = Z_p$ and the boundaries as $\operatorname{im} \partial_{p+1} = B_p$.

Definition 2.3.3 *The boundary of a p-simplex σ is the formal sum of $(p-1)$-dimensional faces of σ.*

We go back to Figure 2.2.1: in terms of sets, the boundary of a tetrahedron $\{A, B, C, D\}$ is the set of four triangle faces which is also shown as $\{\{ABC\}, \{ABD\}, \{ACD\}, \{BCD\}\}$; the boundary of triangle $\{ABC\}$ are its three edges $\{\{AB\}, \{BC\}, \{AC\}\}$; the boundary of an edge $\{AB\}$ are its two vertices $\{\{A\}, \{B\}\}$.

Definition 2.3.4 *The boundary of a p-chain is the sum of the boundaries of the simplices in the chain, taking modulo 2 (i.e. duplicate simplices are cancelled out).*

It is easy to verify the following observation.

Fact 2.3.1 *Taking the boundary generates a group homomorphism ∂_p from C_p to C_{p-1}.*

Referring to Figure 2.3.2, the boundary of a sum of faces formed by the red edges and the yellow edges will have one edge cancel out since this edge is common to both triangular faces.

Note that our definition of the boundary map can be restated as follows.

Definition 2.3.5 *Consider a simplicial complex K and $\sigma \in K$, $\sigma = [v_0, v_1, \ldots, v_k]$. The boundary homomorphism ∂_k is defined as follows:*

$$\partial_k \sigma = \sum_{i=0}^{k} [v_0, v_1, ..., \hat{v}_i, ..., v_p] \tag{3}$$

where $\hat{v}$ indicates that a vertex v omitted from the sequence.

It is easy to check that ∂_k is well defined, that is, the result of ∂_k the same for every ordering of the vertices.

Now, we can differentiate between two special types of chains using the boundary map that will be useful to define homology.

Definition 2.3.6 *The p-th cycle group is* $Z_p = \operatorname{Ker} \partial_p$. *A chain that is an element of* Z_p *named a p-cycle. The p-th boundary group is* $B_p = \operatorname{Im} \partial_{p+1}$. *A chain that is an element of* B_p *is called a p-boundary.*

Fact 2.3.2 *It is easy to see that both* Z_p *and* B_p *are subgroups of* C_p *for all positive integers p.*

Lemma 2.3.1 (Fundamental lemma of homology) *For every* $(p+1)$*-chain c we have* $\partial_p \partial_{p+1}(c) = 0$

The proof of this can be found below Lemma 2.2.7 in [12].

Considering the boundary homomorphism, we have the following diagram for a p-dimensional complex K:

$$0 \to \mathrm{C}_p \xrightarrow{\partial_p} \mathrm{C}_{p-1} \xrightarrow{\partial_{p-1}} \cdots \to \mathrm{C}_1 \xrightarrow{\partial_1} \mathrm{C}_0 \xrightarrow{\partial_0} 0$$

with $\partial_p \partial_{p+1} = 0$ for all p. Note that the sequence is augmented on the right by a 0, with $\partial_0 = 0$. On the left, $C_{p+1} = 0$, as there are no (p+1)-simplices in K. Such a sequence is called a chain complex. Chain complexes are common in homology, but this is the only one we will see here. The images and kernels of these maps are subgroups of C_p. From Lemma 2.3.1, we notice the group of boundaries form a subgroups of the cycles group so we can take quotients. This means we can partition each cycle group into classes of each cycles that differ from each other by boundaries. This leads to the concept of homology groups and their ranks, which we will define and discuss in the coming section.

To really drive this home, it will be advisable to take a look at maps between chain complexes of two simplicial complexes K and L which is induced by some maps between K and L themselves. We consider simplicial maps in the definition below.

Definition 2.3.7 *Let K and L be simplicial complexes. Then an admissible vertex map from K to L is a function* $\phi\colon V(K) \to V(L)$ *such that, if* $\{v_0, v_1, \ldots, v_r\}$ *is a simplex of K, then* $\{\phi(v_0), \phi(v_1), \ldots, \phi(v_r)\}$ *is a simplex of L.*

A simplicial map $\phi\colon K \to L$ is an isomorphism if it is bijective; in which case we say that K and L are isomorphic. It is easy to check that a simplicial map ϕ induces a continuous function $|\phi|\colon |K| \to |L|$ such that if, in barycentric coordinates, $x = \sum \alpha_i v_i$ then $|\phi|(x) = \alpha_i \phi(v_i)$. That is, $|\phi|$ is linear when restricted to each simplex of K.

Definition 2.3.8 *A continuous function $f\colon |K| \to |L|$ between the underlying spaces of two simplicial complexes is a simplicial map (with respect to K and L) if $f = |\phi|$ for an admissible vertex map from K to L.*

Definition 2.3.9 *Let K and L be two simplicial complexes, and $f\colon K \to L$ a simplicial map. Then we define the induced homomorphism of f as the map:*

$$f_{\#}\colon C_p(K) \to C_p(L)\colon \sum_i p_i \sigma_i \mapsto \sum_i p_i f(\sigma_i)$$

The Definition 2.3.9 helps us to get the following as lemma.

Lemma 2.3.2 *For an induced map $f_{\#}\colon C_p(K) \to C_p(L)$, the following diagram commutes:*

$$\begin{array}{ccc} C_p(K), & \xrightarrow{\partial} & C_{p-1}(K) \\ \downarrow f_{\#} & & \downarrow f_{\#} \\ C_p(L) & \xrightarrow{\partial} & C_{p-1}(L) \end{array}$$

This is, $\partial f_{\#} = f_{\#} \partial$.

2.4 Homology

Our choice of homology theory will be simplicial homology, which complements our representation of data clouds in simplicial form. Another important feature of simplicial homology is the existence of efficient computation algorithms. Homology utilizes finitely generated Abelian groups for describing the topology of spaces. Luckily, we fully understand the structure of these groups from our previous subsection 2.1.

Definition 2.4.1 *The p-th homology group is the quotient group $H_p = Z_p/B_p$. The p-th Betti number is its rank: $\beta_p = rank(H_p)$.*

The Definition 2.4.1 shows the important features at each dimension by the clear demonstration of the algebraic relation among simplices of every dimension that appear in K. Homology differentiates between trivial and nontrivial features, revealing how the simplices of K are connected.

For example, the first Betti number $\beta_1 = \text{rank}(\text{Z}_1) = 1$ indicates one independent 1-dimensional hole not filled in by triangles. In general, β_p can be interpreted as the number of independent p-dimensional holes. Using tetrahedron as example, it has $\beta_0 = 1$ since the shape is connected and $\beta_1 = \beta_2 = 0$ since there are no holes or voids or loops.

A hollow tetrahedron (or the boundary of tetrahedron) has $\beta_0 = 1$, $\beta_1 = 0$, $\beta_2 = 1$ because of the "void". When we keep edges and remove the four triangle faces, the skeleton has $\beta_0 = 1$, $\beta_1 = 3$ (we get four triangular holes but one is the sum of the other three), $\beta_2 = 0$ (no more void). Lastly removing the edges but keeping the four vertices, $\beta_0 = 4$ (4 connected components, each a single vertex) and $\beta_1 = \beta_2 = 0$.

Calculating the homology of a simplicial complex from a topological space, it provides powerful insight into the defining features of the space. For complexes built from a set of data points, numerous constructions, such as Vietoris-Rips and Čech complexes, provide various methods for adding and connecting simplices. These complexes rely on a specified proximity parameter — a choice of distance between vertices in the original set V — that determines whether and how simplices are added to the complex K. But for simplicial complexes that derive from point-cloud data, many nontrivial topological features may not appear at just one parameter value, which generates just one simplicial complex from the data. This limitation highlights the challenge of topologically analyzing raw data in a meaningful way: often no single parameter value captures at once all the significant features underlying the dataset.[13]

2.5 Persistent Homology

Persistent homology attempts to search and trace homological groups through filtrations. A filtered simplicial complex with its boundary functions is called a persistent complex. For an increasing sequence of positive real numbers we attain a persistent complex. Our goal is to analyze topological invariants of a point cloud dataset by analyzing its persistent complex. Some standard representations for persistent homology are barcode, persistent diagram, and landscape. A barcode represents each persistent generator(classes that generate p-th homology group) with a horizontal line beginning at the first filtration level where it appears, and ending at the filtration level where it disappears, while a persistence diagram plots a point for each generator with its x-coordinate the birth time and its y-coordinate the death time. A visual representation of persistent homology is persistent diagram.

In what follows, we provide detailed explanation of the above mentioned concepts. Our ultimate goal is to apply the methods of persistent homologies to open ended responses. We define persistence homologies by considering how the simplicial homology changes as we move from one choice of the parameter to another. This process is proven to be stable with respect to small perturbations in the point cloud data X.

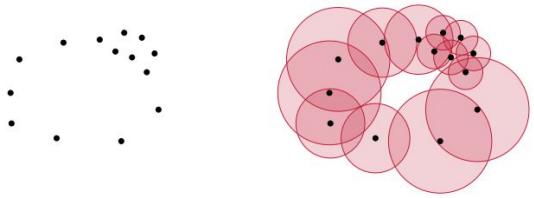
Data from the real-world is not commonly endowed with a simplicial structure. We receive it as a point cloud which will then be converted into the space suitable to be studied by the methods of algebraic topology. The conversions goes through a process of construction and we shall take a look at a very general concept first, the nerve of a covering. After that, we will describe some constructions that permit us to approximate data by simplicial complexes.

Definition 2.5.1 *Given a topological space X, an indexed family of sets $\mathcal{U} = \{U_i | i \in \mathcal{I}\}$ is a covering of X if*

$$X \subseteq \bigcup_{i \in \mathcal{I}} U_i,$$

Thus, the covering in the above definition described the situation when a topo-

logical space (point cloud data) is contained inside the union of sets. In practice, we assume that the number of sets required for a covering is finite. We will later on see how to obtain coverings automatically for data from $\mathbb{R}^n$



A point cloud data Covering

Figure 2.5.1: A point cloud data and a covering. Each set of the covering is shown as a ball. Some of the points are covered more densely than others[14].

In the Figure 2.5.1 we show a covering of a point cloud data. Considering a covering of a topological space, we may obtain a simplicial complex from it. The idea is to define a simplex for each set of elements of a covering that have non-empty intersection. This leads to the Definition 2.5.3 of a nerve. A 1-skeleton of the nerve complex is given by all pairs of covering sets that intersect, while the 2-skeleton contains all triples of covering sets that intersect, etc. Figure 2.5.2 depicts the 1-skeleton and the 2-skeleton of a covering.

2.5.1 The Nerve Theorem

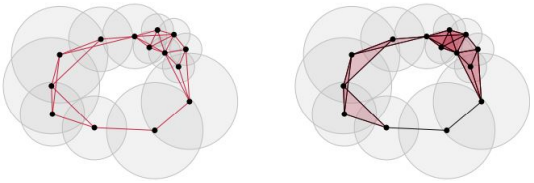
The nerve theorem is one of the most important theorems for both persistent homology and the Mapper algorithm which we will talk about in the next chapter. The theorem helps to identify non-empty intersections among subsets of some topological space X. These subsets end up forming simplicial complex which will act as a topological space X for our point cloud dataset.

Definition 2.5.2 *For a topological space X, we consider a finite collection $\mathcal{U}$ of subsets $U \subseteq X$. The nerve $\mathcal{U}$, is the collection of all subsets of $\mathcal{P}(\mathcal{U})$ whose sets have non-empty intersections:*

$$N(\mathcal{U}) = \left\{ X \subseteq \mathcal{P}(\mathcal{U}) \bigcap X \neq \emptyset \right\} \tag{4}$$

Definition 2.5.3 *Given a covering $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ of a topological space X, the nerve of the covering $\mathcal{U}$ is the simplicial complex $N(\mathcal{U})$, known as the Čech complex, whose vertex set is the index set A, and where a subset $\{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_n\} \subseteq A$ spans a n-simplex in $N(\mathcal{U})$ if and only if $U_{\alpha_0} \cap U_{\alpha_1} \cap U_{\alpha_2} \cap \cdots \cap U_{\alpha_n} \neq \emptyset$.*

The simplices correspond to subcollections of sets in $\mathcal{U}$ that have at least one point in common (non-empty intersection). This clearly defines a simplicial complex. The nerve construction is useful in the case of discrete point cloud as well as in general case of arbitrary topological space.



The 1-skeleton of the nerve The 2-skeleton of the nerve

Figure 2.5.2: The nerve of a covering. For k-simplex as a member of the nerve, we need to evaluate intersections between subsets of cardinality k. When calculating the different skeletons of the nerve, we can see that the upper right part of the point cloud gives rise to many 2-simplices. [14]

Definition 2.5.4 *Let $f, g \colon X \to Y$ be two continuous maps be a map between two topological spaces. We say f and g are homotopic, and write $f \simeq g$, if there is a continuous map $H \colon X \times [0,1] \to Y$ with $f(x) = H(x,0)$ and $g(x) = H(x,1)$ for all $x \in X$.*

Definition 2.5.5 *We say that a continuous map $f \colon X \to Y$ between two topological spaces is a homotopy equivalence, if there exists a map $g \colon Y \to X$ such that $gf \simeq \mathrm{id}_X$ and $fg \simeq \mathrm{id}_Y$.*

Lemma 2.5.1 *For any finite collection of sets $\mathcal{U}$, its nerve $\mathcal{N}(\mathcal{U})$ defines a simplicial complex.*

Theorem 2.5.1 *We let $\mathcal{U}$ be a finite collection of closed, convex sets in $\mathbb{R}^N$. Then the geometric realization of the nerve of $\mathcal{U}$ and the union of all sets contained in $\mathcal{U}$ are homotopy equivalent*:

$$\bigcup \mathcal{U} \simeq |N(\mathcal{U})|$$

After showing the definition of the nerve complex and how it can be obtained, we get an idea of the formation of the nerve and we will turn to look at how to obtain a covering of that element in the set. For example, in an Euclidean space, how can we construct a closed geometric ball around the element in the set with the same radius? This gives us an idea of filtrations.

2.5.2 Čech complex

Consider our point cloud data $X \subseteq \mathbb{R}^n$, with a set of data points as Figure 2.5.1, each with n variables. To find the underlying structure of the point cloud data X, we make an assumption that X is sampled from some specific subspace $E \subseteq \mathbb{R}^n$ and we would like to show some topological properties of this specific subspace E. The most interesting aim would be to recover the homology of E. Since X is a discrete set, it does not have any interesting topology on its own, but by constructing balls around the points in X and looking at their union, we hope to obtain more structure. The collection of these balls induce a simplicial complex called the Čech complex.

Definition 2.5.6 *Letting X be a finite set of points in $\mathbb{R}^n$, and $B_x(r)$ denote a closed ball with centre $x \in \mathbb{R}^n$ and radius $r > 0$, the Čech complex of $n+1$ ponts and r is the nerve of this family of balls:*

$$C(r) := \left\{ \sigma \subseteq n+1 \mid \bigcap_{x \in \sigma} B_x(r) \neq \varnothing \right\} \tag{5}$$

Thus, complex is the nerve of the covering of the point cloud that consists of closed balls of radius r. Since the Euclidean balls are convex objects, it follows from Theorem 2.5.1 that, the Čech complex on $n+1$ at scale r is homotopy equivalent to the union of balls of radius r centered at the points of $n+1$

Fact 2.5.1 *A Čech complex has the property that increasing radii result in nested complexes.*

Fact 2.5.2 *[15] For a radius r, a subset of vertices σ forms a simplex if and only if the corresponding set of points can be enclosed within a ball of radius r*

Checking whether a set of points can be enclosed by a ball of a given radius is a standard problem in computational geometry[16, 17]. Its solution becomes more complex as the dimensions increase with increase in data, making the Čech complex currently infeasible to calculate for most applications.

Since Čech complex construction is more faithful to the topology (by the Nerve theorem), there have been recent attempts at an improved construction algorithm [18], but the results are still somewhat preliminary. The Čech complex require precise distances between points, resulting in a computational burden. Thus we have to choose alternative type of complex that is more computationally easy to deal with. The drawback of the subsequent construction is that we lose the homotopy type preservation property to some extent. However, the combinatorial construction turns out to be less complicated[14].

Definition 2.5.7 *Let X be a finite topological space with an associated metric $dist_X$. The diameter of X is the upper bound of the set of all pairwise distances, i.e.*

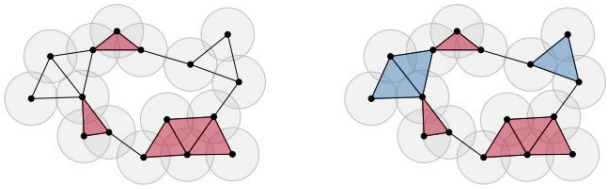
$$\text{diam } X = sup\Big\{ dis_X(x, y) \,\Big|\, x, y \in X \Big\} \tag{6}$$

Since our X, being point cloud, is a finite topological space, hence, the supremum sometimes exists and is attained by some pair of points in the space.

We show in Figure 2.5.4 how to obtain geometric complexes from a point cloud data X. The Čech and Vietoris-Rips complexes are demonstrated on the same value of the scale parameter. The Čech complex contains a triangle for each subset of three balls or disks with a non-empty intersection. By contrast, the Vietoris–Rips complex contains a triangle whenever three balls have a pairwise intersection.

This is why we said early that subsequent construction of Vietoris-Rips complex always loses the homotopy type preservation property, but the combinatorial construction of Vietoris-Rips complex turns out to be less complicated.

From Figure 2.5.4 we can now begin to look more into Vietoris-Rips for our computation. Depending on the application, other constructions for creating simplicial



Čech complex Vietoris-Rips complex

Figure 2.5.3: The red triangles shows Čech complex while the blue triangles shows Vietoris-Rips complex

complexes are available as well. For example flow complex, alpha complexes, which are subcomplexes of the Delaunay triangulation etc.

2.5.3 Vietoris-Rips complex

In contrast to the Čech complex, since Vietoris-Rips complexes need only coarse proximity information on pairs, their geometric representations might be just an approximation. Both methods often produce simplices of dimensions higher than that of the space containing the point cloud. We introduce the Vietoris-Rips complex as a filtration function for our computation of homology. Let $X \subset \mathbb{R}^n$ be a finite collection of points $x_1, ..., x_{n+1}$, where $|X| = n + 1$.

Definition 2.5.8 *Let X be a finite set of points in a metric space. Consider $r \geq 0$, that will be referred to as a scale parameter. The Vietoris–Rips simplicial complex $VR(X; r)$ is defined to have as its n-simplices those collections of $n+1$ points in X that have diameter at most r, for all non-negative integers n.*

Figure 2.5.8 shows a point cloud data with disk around the points but no ideal choice of scale parameter. Concentrating on the right hand diagram labelled Vietoris-Rips complex, we can say the Vietoris–Rips complex captures the desired features in the data and closes more holes due to its properties. The perspective behind persistence is to allow the scale parameter to increase in order to observe the corresponding appearances and disappearances of topological features. To be more precise, each hole appears at a certain scale (birth) and disappears at a larger scale (death). Those holes that persist across a wide range of scales often reflect

topological features in the shape underlying the data, whereas the holes that do not persist for long are often considered to be noise.

In contrast, while Vietoris-Rips complexes need only coarse proximity information on pairs, their geometric representations might be just an approximation. Both Čech and Vietoris-Rips often produce simplices of dimensions higher than that of the space. A third approach called the Witness Complexes addresses this last problem, but its faithfulness to the space is uncertain.

2.5.4 Persistence Modules

Notation 2.5.1 *A map $in : X \to Y$ between topological spaces X and Y is called an inclusion map, if $X \subseteq Y$ and $in(x) = x, \forall x \in X$. We denote this map as $X \hookrightarrow Y$*

Definition 2.5.9 *Let $I \subseteq \mathbb{R}$ be a set of real values. A filtration $(X)_{\alpha \in I}$ is a collection of topological spaces X_α such that for each pair $\alpha_1 \leq \alpha_2$ there is an inclusion $X_{\alpha_1} \hookrightarrow X_{\alpha_2}$. The elements of the set I are called the scales of the filtration.*

A filtration can be obtain using concepts from Definitions in 2.5.8. For any two scales $0 \leq \alpha \leq \beta$, the Vietoris-Rips complexes at those scales satisfy $R_\alpha \subseteq R_\beta$, so there is a natural inclusion from R_α to R_β. This gives a Vietoris-Rips filtration $(R_\alpha)_{\alpha \geq 0}$ over the range of scales $[0, \infty)$.

An increasing sequence $0 \leq \epsilon_0 < \epsilon_1 < \epsilon_2 < \dots$ produces a filtration

$$VR_{\epsilon_0} \subseteq VR_{\epsilon_1} \subseteq VR_{\epsilon_2} \subseteq \dots, \tag{7}$$

with the property that a simplex enters the sequence no earlier than all its faces. The associated inclusion maps induce linear maps between the corresponding homology groups $H_k(VR_{\epsilon_i})$ where $i \in \{1, \dots, n\}$, which are algebraic structures whose ranks count the number of independent k-dimensional holes in the Vietoris–Rips complex. A technical remark is that homology depends on the choice of a group of coefficients; it is simplest to use field coefficients (for example $\mathbb{R}, \mathbb{Q}, \text{or} \mathbb{Z}/p\mathbb{Z}$ for p prime), in which case the homology groups are furthermore vector spaces[19]. In this text, we restrict

ourselves with coefficients from $\mathbb{Z}/2\mathbb{Z}$. The corresponding collection of vector spaces and linear maps is called a persistent homology module.

Definition 2.5.10 *Let $J \subset R$ be a discrete index set. A (simplicial) tower over J is a sequence of simplicial complexes $(K_\alpha)_{\alpha \in J}$, along with simplicial maps $f_\alpha \colon K_\alpha \to K_{\alpha'}$ for every pair of consecutive scales $\alpha \leq \alpha'$ in J.*

With the collection of consecutive parameter scales $\{\alpha_1 \leq \cdots \leq \alpha_m\}$, $\alpha_i \in J$ for all i, we can create simplicial maps to get $f_{\alpha_1,\alpha_m} \colon K_{\alpha_1} \to K_{\alpha_m}$ by defining $f_{\alpha_1,\alpha_m} = f_{\alpha_{m-1}} \circ \cdots \circ f_{\alpha_1}$. This turns the tower into a category, where the objects are the simplicial complexes at different parameter scales and the morphisms are simplicial maps between them. The simplest example of a tower is a filtration such as the Vietoris-Rips filtration restricted to a discrete number of parameter scales, where the simplicial maps are simply inclusions.

The shape of a tower is the number of simplices which are included over all parameter scales. For the special case of filtrations, the shape of the tower is simply the largest complex in the sequence. Note that this is not true in general for simplicial towers, since simplices can collapse in the tower, and the size of the complex at a given scale may not take into account the collapsed simplices which were included at earlier scales in the tower[20].

Definition 2.5.11 *A persistence module $\mathbb{V}$ over an index set $I \subseteq \mathbb{R}$ is a collection of vector spaces $(V_\alpha)_{\alpha \in I}$ connected with homomorphisms of the form $\phi^{\alpha,\beta} : V_\alpha \to V_\beta$ for $\alpha \leq \beta$, that satisfy:*

- *$\phi^{\alpha,\alpha}$ is the identity map on V_α for each $\alpha \in I$;*
- *$\phi^{\alpha,\gamma} = \phi^{\beta,\gamma} \circ \phi^{\alpha,\beta}$ for all $\alpha \leq \beta \leq \gamma$.*

We can now build a persistence module from the concept defined in 2.5.10. Let $(K_\alpha)_{\alpha \in I}$ be a tower connected with maps $\{f_\alpha\}_{\alpha \in I}$. As mentioned earlier, $(K_\alpha)_{\alpha \in I}$ along with the composition of simplicial maps $\{f_{\alpha,\beta}\}_{\alpha \leq \beta \in I}$ forms a cluster. Applying the homology functor from our Definition 2.2.1 gives a collection of vector spaces

$(H_p(K_\alpha))_{\alpha\in I}$ connected with linear maps $f_*^{\alpha,\beta}\colon H_p(K_\alpha) \to H_p(K_\beta)$ which satisfy the conditions for being a persistence module, for all p [21]. Therefore, $(H_p(K_\alpha))_{\alpha\in I}$ is the $p-th$ persistence module of $(K_\alpha)_{\alpha\in I}$. We denote by $H(K_\alpha)$ the direct sum of $H_p(K_\alpha)$ for all p, and in short say that $(H(K_\alpha))\alpha \in I$ is the persistence module of the tower.

Definition 2.5.12 *Homology classes $c' \in H_p(K_\alpha)$ and $c \in H_p(K_\beta)$, where $\alpha < \beta$, merge at $H_p(K_\gamma)$ for $\gamma > \beta$ if $f_{\alpha,\gamma}(c') = f_{\beta,\gamma}(c)$. We say that c' is an elder of c.*

Definition 2.5.13 *A homology class $c \in H_p(K_\beta)$ dies (in the sense of the elder rule) at $H_p(K_\gamma)$ if either $f_{\beta,\gamma}(c) = 0$ or c merges with an elder at $H_p(K_\alpha)$. We denote by* death(c) *the smallest index γ for which c dies in $H_p(K_\gamma)$.*

Note that we can consider the existence of two simultaneously born homology classes c' and c in $H_p(K_\beta)$ which merge at $H_p(K_\gamma)$ but do not have elders. In that case, we do not have a formal rule to decide which dies in $H_p(K_\gamma)$. In our application, we make an arbitrary choice at this point.

The useful tools of calculating and visualising persistent homology are persistence diagrams and barcodes. A persistence diagram contains all pairs (birth(c), death(c)) for some p-th persistent homology group. The barcodes contain the same information that the persistence diagram but instead of pairs we present lifetimes, $[\text{birth}(c), \text{death}(c))$[22].

Definition 2.5.14 *A barcode is a graphical representation of $H_p(V;F)$ as a collection of horizontal line segments in a plane whose horizontal axis corresponds to the parameter and whose vertical axis represents an (arbitrary) ordering of homology generators. A barcode is best thought of as the persistence analogue of a Betti number which has it root in homology.*

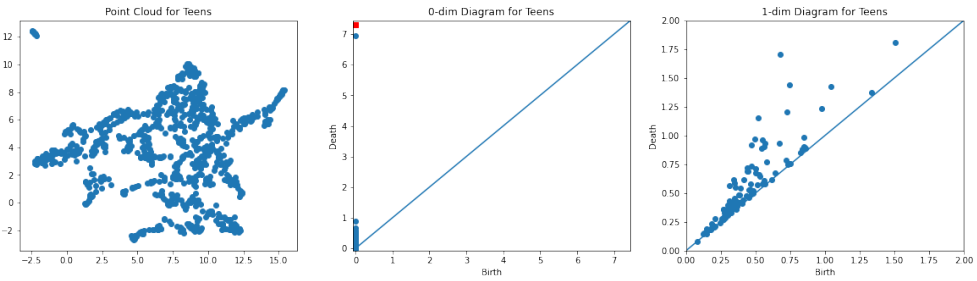
For a finite persistence module V with field F coefficients in the representation of barcodes shows the ability to qualitatively filter out topological noise and capture significant features.

Definition 2.5.15 *Let* $\mathbb{V}$ *be a persistence module generated by persistent homologies with barcode* $\{[b_1, d_1), \ldots, [b_m, d_m)\}$. *The persistence diagram* (dgm) *of* $\mathbb{V}$ *is the multi-set of points*

$$\mathrm{dgm}(\mathbb{V}) = \{(b_i, d_i)\}_{1 \leq i \leq m} \tag{8}$$

in the extended Euclidean plane $[-\infty, \infty]^2$. *We say that the value* $(d_i - b_i)$ *is the persistence of the interval* $[b_i, d_i)$.

Since $d_i \geq b_i$ for each interval $[b_i, d_i)]$ in the barcode, all the points in the persistence diagram lie on or above the diagonal $y = x$ for $x =$ birth and $y =$ death of a hole. The idea translates to characterizing the connected components, holes, voids, loops, and higher dimensional analogues that persist for the Vietoris Rips complexes, VR_j , where $0 \leq j \leq p$. This is also recorded using a data visualization tool called the Persistence Diagram [23]. This is a scatter plot of points above the diagonal, or line $y = x$, where the y-axis is the axis labeled "birth" and x-axis is labeled "death" as shown in Figure 2.5.4. A topological signal is detected if it deviates away from the diagonal. Otherwise, it is treated as noise. Referring back to Figure 2.5.8, we can see a circular figure persisting throughout the filtration. Here, the topological signal is the loop that characterizes the circle.



Point cloud data Persistent diagram (H_0) Persistent diagram (H_1)

Figure 2.5.4: The pictorial representation of H_0 and H_1

2.6 Bottleneck distance

In this subsection we discuss the notion of closeness of persistence modules. That is a way to compare two barcodes. Let $\mathbb{V}, \mathbb{W}$ be two persistence modules. We

discuss a way to compare the persistence diagrams (dgm$\mathbb{V}$) and (dgm$\mathbb{W}$). The standard metrics for measuring the distance between two persistence diagrams are the bottleneck distance and the p^{th} Wasserstein distance [24]. In this paper, we use the bottleneck distance.

Remark 2.6.1 *In many applications, each edge of a graph has an associated numerical value, called a weight. Usually, the edge weights are nonnegative integers. Weighted graphs may be either directed or undirected. For example, the weight of edges can be defined by meas of weight of vertices as follows. Let $\omega(u)$ and $\omega(v)$ be weight of a vertex u and v respectively. For an edge $e = (u, v)$ in E, assign its weight as $\omega(e) = |\omega(u) - \omega(v)| = |g(u) - g(v)|$. Note that $\omega(u) \geq 0 \, \forall e \in G$. In real world applications, the weight may be a measure of the length of a route, the capacity of a line, the energy required to move between locations along a route, etc.*

Definition 2.6.1 *A matching on the vertex set V of a graph G is a subgraph of G in which each vertex is contained in at most one edge. If $G = (V, E)$ is a graph then the collection of all subgraphs (V, E) of G which are matchings determines a simplicial complex $M(G)$.*

Definition 2.6.2 *A bottleneck matching $M \subseteq \mathrm{dgm}(\mathbb{V}) \times \mathrm{dgm}(\mathbb{W})$ between points of $\mathrm{dgm}(\mathbb{V})$ and $\mathrm{dgm}(\mathbb{W})$ with cost μ is a matching which satisfies:*

- *each point of $\mathrm{dgm}(\mathbb{V})$ is matched to at most one point of $\mathrm{dgm}(\mathbb{W})$ and vice-versa, with $||v - w||_\infty \leq \mu$ for each $(v, w) \in M$.*
- *for each unmatched point of $u \in \mathrm{dgm}(\mathbb{V}) \cup \mathrm{dgm}(\mathbb{W})$, the l_∞-distance from u to the diagonal $y = x$ is at most μ.*

The bottleneck distance between $\mathrm{dgm}\mathbb{V}$ and $\mathrm{dgm}\mathbb{W}$ is defined as

$$\mathrm{dB}(\mathrm{dgm}(\mathbb{V}), dgm(\mathbb{W})) = \inf\{\mathrm{cost}(M) \colon M \subseteq dgm(\mathbb{V}) \times dgm(\mathbb{W})\},$$

where $\mathrm{cost}(M)$ is the cost of the matching M.

In the above definition, we define l_∞-distance and respective norm in $\mathbb{R}^2$ in the standard way, i.e. for $v = (x, y)$ we let $||v|| = \max\{|x|, |y|\}$.

3 Chapter 3: Mapper Algorithm

Methods of the previous section 2 could only handle a limited amount of data to process due to computational difficulty. For our intended application to loneliness survey, we cannot use these methods to analyze category for whether they live alone or not and category for self ratings on loneliness. Hence we consider another powerful tool of TDA that is mostly known for capturing important insights on shapes and features in big data. We discuss the Mapper algorithm, one of the key tools in TDA that is used for reducing and interpreting high dimensionality set of data by capturing its shape. The Mapper algorithm has gained significant traction across various application domains [4]. Mapper was proposed in [10] as a method to understand the shape of data using a topology-inspired construction. The whole algorithm is primarily based on the idea of partial clustering of the data guided by a set of functions defined on the data.

Definition 3.0.1 *Let X and Z be topological spaces and let $f\colon X \to Z$ be a continuous map. Let $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ be a finite open covering of Z. Consider the covering $f^*(\mathcal{U}) = \{f^{-1}(U_\alpha)\}_{\alpha \in A}$ of X, which is called the pull-back of $\mathcal{U}$ along f. The Mapper construction is defined to be the nerve complex(see Definition 2.5.3) of the pullback covering $M(\mathcal{U}, f) = N(f^*(\mathcal{U}))$.*

To apply the Mapper algorithm, we need the following:

1. dataset (X)
2. Filter function $f\colon X \to Z$ (dimentionality reduction)
3. Covering generated by $g\colon X \to Z$ (Put data into overlapping bin; a standard choice for the covering in the case $Z = \mathbb{R}$ is a finite sequence of overlapping intervals)
4. Cluster each bin and create a graph where vertex represent a cluster of bin and edge represent nonempty intersection between clusters

The pipeline will be explained below according to how it works. We begin with representing our data as input and the output is a simplicial complex.

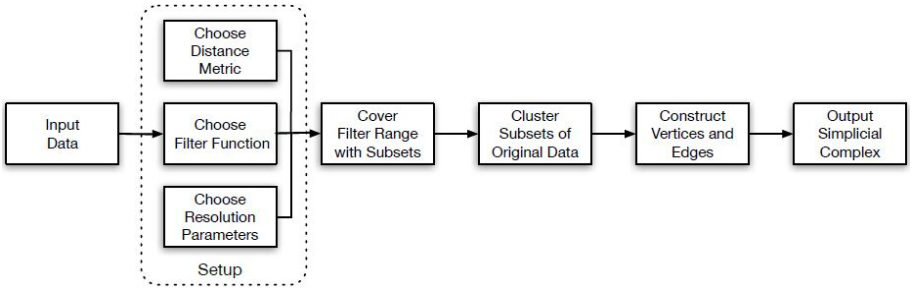


Figure 3.0.1: Framework of the Mapper algorithm for generating topological networks [25]

Since the Mapper complex is high dimensional in general, applications have typically used its 1-skeleton (referred to as the Mapper graph) for numerical and qualitative analysing, simplifying, visualizing and interpreting results. Features such as paths, flares, and loops, in the selected Mapper graph often convey meaningful insights on the dataset. In many cases, data coming from real applications is big and it is not possible to visualize and discern structure even in low dimensional projections. The loneliness dataset also has millions of points and is known to have a simple structure which is obscured due to its immense size.

3.1 Metric Space and Point Cloud Data

Mapper was originally constructed for numerical data [10]. The construction uses points in a metric space just like most cluster analysis algorithms. If a coherent definition of pairwise distance between every point can be constructed, then Mapper can construct a map of the topology of that space, but interpreting the resulting map may be more difficult than if the data were embedded in a high-dimensional vector space.

One possible alternative distance function involves using the correlation between two sets of responses, which would allow a mixture of quantitative and qualitative responses to be used in the mapping[26]. Formally, whichever function is used to calculate distances must satisfy certain criteria. We define Mapper and outline the

algorithm from input dataset (X) to output simplicial complex.

Since the algorithm requires a metric space when applying it to survey data, responses from the survey need to be in the form of a point cloud data or some points existing in a metric space. People who did not respond to the survey were taken out of our study to deal with the issue of missing values, since they pose a problem to further analysis. Although these participants with no responses must exist somewhere in the space and have a measurable distance to each other point in order to be clustered, the algorithm requires there be no missing values hence they are represented by their responses to other questions.

This approach helped us to analyze the complete dataset. The distance matrix is used to capture the similarity among the data points. There are a number of Topological Methods in Machine Learning and Artificial Intelligence.

3.2 Filter functions

The next step is computing filter functions, also known as lenses in TDA. The filter functions are one of the most important steps in the Mapper algorithm. The choice of a filter functions is very important since a good function can help us to reveal some interesting geometrical information about the dataset. In Section 2 we discussed a filtration called the Vietoris-Rips complex that took a point cloud data from lower dimension to higher dimension in order to detect some topological features in the data but we will be doing the opposite in this section.

We consider the data in a high dimension matrix and then reduce dimensionality while preserving some important features of the data. In the default setting, each dimension represents a filter function, or equivalently, all of them could be considered together as a high-dimensional filter function. In our implementation, we consider individual filter functions $f_i \colon X \to \mathbb{R}$ for $i = 1, \ldots, h$. Since we may have sparse data, considering each dimension as a filter function may bring problems to analysis. The sensitivity of our text dataset affect the majority of words in the responses in terms of representation, since the dimensions in our case will be generated as a result of

the total responses by the total vocabulary.

Looking at the unsupervised data structure problem associated with our data, we employ machine learning algorithms to handle the task of dimensionality reduction. There are many algorithms that we tried for the purposes of finding a good approximation for the representation of our sparse matrix, such as Principal Component Analysis (PCA), Truncated Singular Value Decomposition (T-SVD), t–Stochastic Neighbourhood Embedding (t-SNE) and Uniform Manifold Approximation and Projection(UMAP) [1][27] etc. We chose T-SVD and UMAP for filter function in this work. Note, in particular, that T-SVD is a popular method for dimensionality reduction when it comes to doing better work on sparse data. Sparse data here means a data with many zero values. PCA is used on a covariance matrix.

3.2.1 Truncated Singular Value Decomposition (T-SVD)

The singular value decomposition of a matrix A is the factorization of A into the product of three matrices $A = UDV^T$ where the columns of U and V are orthonormal and the matrix D is diagonal with positive real entries. The SVD is useful in many tasks. The idea here is to obtain a low rank matrix that would be "close" to the matrix A, i.e. could serve as a good approximation to the data matrix A. The right singular vectors of A which are the columns of V in the singular value decomposition, always form an orthogonal set with no assumptions on A. The columns of U are called the left singular vectors and they also form an orthogonal set. A simple consequence of the orthogonality is that for a square and invertible matrix A, the inverse of A is $VD^{-1}U^T$.

To gain insight into the SVD, treat the rows of an $n \times d$ matrix A as n points in a d-dimensional space and consider the problem of finding the best k-dimensional subspace with respect to these points. The "best" here means the one that minimizes the sum of the squares of the perpendicular distances of the points to the subspace [28].

The following is a classical theorem from linear algebra, whose proof can be found

in most standard textbooks. Within this theorem, we also included the definitions of singular values and singular vectors (these definitions are equivalent to defining these concepts using the eigenvalue approach).

Theorem 3.2.1 *For any $n \times d$ matrix with real coefficients A there exist an $n \times r$ matrix U, $d \times r$ matrix V , and $r \times r$ diagonal matrix D such that $A = UDV^T$, and such that the columns of U form an orthonormal set, and the columns of V form an orthonormal set. The diagonal entries of D, denoted $\sigma_1, \sigma_2, \ldots, \sigma_r$, are called the singular values of A. The columns $u_1, \ldots, u_m$ of U are called the left singular vectors; the columns $v_1, \ldots, v_m$ of V are called the right singular vectors.*

The figure below illustrate the above theorem. A can be decomposed into a sum of rank one matrices as Figure 3.2.1.

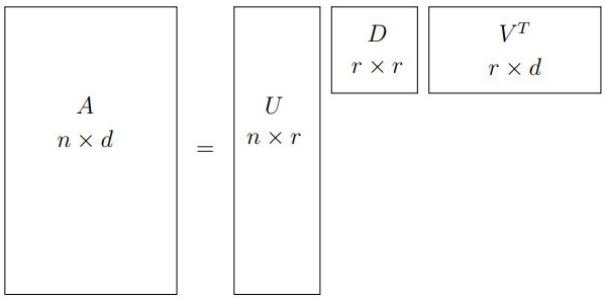


Figure 3.2.1: The SVD decomposition of an $n \times d$ matrix

As a direct consequence of Theorem 3.2.1 we note that A can be decomposed into a sum of rank one matrices as follows.

Theorem 3.2.2 *Let A be an $n \times d$ matrix with right singular vectors $v_1, v_2, \ldots, v_r$, left singular vectors $u_1, u_2, \ldots, u_r$, and corresponding singular values $\sigma_1, \sigma_2, \ldots, \sigma_r$. Then*

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{9}$$

Note that in the above theorems the number r also represent the number of singular values (σ) that are nonzero.

For any matrix A, the sequence of singular values is unique and if the singular values are all distinct, then the sequence of singular vectors is also unique. Nevertheless, when some set of singular values are equal, the corresponding singular vectors

span some subspace. Any set of orthonormal vectors spanning this subspace can be used as the singular vectors [28]. Singular values are nonnegative, and usually listed in decreasing order: $\sigma_1 > \sigma_2 > \cdots > \sigma_r > 0$.

We can form an approximation to A by truncating the sum in the Equation 9. Instead of taking all the singular values and their corresponding left and right singular vectors, we only take the k largest singular values and their corresponding vectors:

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T \tag{10}$$

This is called the truncated SVD.

Note that this truncation allows for a significant savings in terms of storing the elements of A. It also reduces the amount of necessary computations we may want to perform in our analysis of A viewed as a point cloud representation. On the other hand, careful choice of k may allow us to retain the most important features of the data encoded in A.

3.2.2 Uniform Manifold Approximation and Projection(UMAP)

One of the most famous algorithms for visualization is t-SNE, but its performance suffers with large data and using it correctly can be challenging. UMAP is a new algorithm developed in 2018 [27]. It offers a number of advantages over t-SNE, but more importantly it allows one to increase speed and to better preserve global structures in data. The UMAP outperform t-SNE when the data is embedded into more than $2-$dimensions.

We specifically chose this algorithm since we will be using low dimensional representation for further work in build our Mapper graph. The computation performance of UMAP is better and more efficient than t-SNE. This is the result of the fact that UMAP does not require global normalisation (since it represents data as a fuzzy topological structure rather than as a probability distribution). It aslo allows for data-dense regions to be "stretched out" in the representation. This can help reduce

overcrowding of the low dimensional representation, but comes at the cost of a more challenging interpretation of distances.

The theoretical foundations for UMAP are largely based in manifold theory and TDA. Much of the theory is most easily explained in the language of topology and category theory. Readers may consult [29] for background to UMAP.

To start UMAP algorithm, we first approximate the manifold that we assume the data (approximately) lies on. The manifold may be known apriori as M or may need to be inferred from the dataset X. Suppose the manifold is not known in advance and we wish to approximate geodesic distance on it. In practice, finite real world data is rarely so nicely behaved. However, if we assume that the manifold has a Riemannian metric not necessarily inherited from the ambient space, we can find a metric such that the data is approximately uniformly distributed with regard to that metric [27]. The data points in dataset X are considered to be from some Riemannian manifold M, then mapped into $\mathbb{R}^n$ by some embedding $\phi : M \hookrightarrow \mathbb{R}$.

Now we reconstruct manifold M to find a good embedding from $M \hookrightarrow \mathbb{R}^m$. In achieving this, we assume X uniformly drawn from M where $X \subset \mathbb{R}^n$. Note that in practice we may also have stretched out or compressed M under the embedding into $\mathbb{R}^n$, hence this does not imply that the data is uniformly distributed in $\mathbb{R}^n$. This assumption only implies that X approximates M well. The algorithm also relies on a connectivity assumption which assumes that every data point in X is connected in M to its nearest neighbour in X, considered as a subset of $\mathbb{R}^n$.

Now, we consider the metric on M as locally constant, this property then gives us the following Lemma 3.2.1, which allows us to approximate distances in M between data points in X that are close enough in $\mathbb{R}^n$. Since M is embedded in $\mathbb{R}^n$, we can measure distance and volume within M in two ways: that of the intrinsic metric on M, and that of the metric induced by $\mathbb{R}^n$ [30]. A proof of the following lemma can be found in Appendix A of the supplementary materials in [27].

Lemma 3.2.1 *Let (M, g) be a Riemannian manifold embedded in R^n (here g is the metric tensor). Let $p \in M$ be a point. Suppose that g is locally constant about p*

in an open neighbourhood U such that g is a constant diagonal matrix in ambient coordinates. Let B be a ball in M, containing p with volume $\frac{\pi^{n/2}}{\Gamma(n/2+1)}$ with respect to the metric on M. Then the distance of the shortest path in M from p to any point $q \in B$ is $\frac{1}{r} d_{\mathbb{R}^n}(p,q)$, where r is the radius of B in $\mathbb{R}^n$ and $d_{\mathbb{R}^n}(p,q)$ is the distance from p to q in $\mathbb{R}^n$.

Assume X is uniformly distributed on M. Then, away from any boundaries, any ball of fixed volume should contain approximately the same number of points of X regardless of where on the manifold it is centered. We now have the opportunity of approximating distances around M close to the data points through scaling the distance in $\mathbb{R}^n$. Conversely, a ball centered at x_i that contains exactly the k-nearest-neighbors of x_i should have approximately fixed volume regardless of the choice of $x_i \in X$ [27]. Looking at Lemma 3.2.1 we see that the geodesic distance from x_i to its neighbors can be approximated by normalising distances with respect to the distance to the k^{th} nearest-neighbor of x_i.

By Lemma 3.2.1, for k small enough, we can approximate distances in M from x_i to one of its k nearest neighbours x_j as follows. We represent k as a hyperparameter and write $\{x_{i_1}, ..., x_{i_k}\}$ for the k nearest neighbours of x_i.

Notation 3.2.1 *A Hyperparameter is a parameter whose value is used to control the learning process. These values are chosen and can be fine tuned at a moment to attain different results from a dataset.*

We can tell from Lemma 3.2.1 that the distance in M from x_i to x_j is approximately $\frac{1}{r_i} d_{\mathbb{R}^n}(x_i, x_j)$, where r_i is the distance to the k^{th} nearest neighbour of x_i. To smooth this value, and reduce a potential impact of having the k^{th} nearest neighbour very far away while the $(k-1)^{th}$ nearest neighbours clustered close to x_i, we take r_i to be the value such that

$$\sum_{j=i}^{k} exp\Big(\frac{-|x_i - x_j|}{r_i}\Big) = \log_2(k) \tag{11}$$

This setup is problematic since the distance we get from x_i to x_j using the above

method will in general be different to that from x_j to x_i, as $r_j \neq r_i$. This means while the x_i are uniformly drawn from the manifold, they are not all the same distance apart as discussed previously. We need a technique for combining a family of locally-defined finite metric spaces to get a global structure, where we have some idea of uncertainty on the metric spaces. For this, we will use fuzzy simplicial sets which are defined in page 8 of the the paper [27].

Note that, although we have used the $\mathbb{R}^n$ metric to develop this theory, it still holds with any other metric. UMAP can be used with custom distance measures and so can handle categorical data and other measures of distance between data points [30].

For the purposes of UMAP, category theory gives us an adjunction between fuzzy simplicial sets and finite extended-pseudo-metric spaces. An adjunction is a translation between different domains of discourse.

The above introduction, definitions, and lemmas give us a brief idea of what UMAP is and why we chose it as one of our filters. It is now clear that there are a number of advantages it offer over t-SNE. While both UMAP and t-SNE produce somewhat similar output, the increased speed, better preservation of global structure, and more understandable parameters make UMAP a more effective tool for visualizing high dimensional data. It is important to remember that no dimensionality reduction technique is perfect – by necessity, we distort the data to fit it into lower dimensions and UMAP is no exception [31]. More importantly, by framing an intuitive understanding of how the algorithm works and understanding how to tune its parameters, we can more effectively use this powerful tool to reduce, visualize and understand large, high-dimensional datasets.

3.3 Covering

The covering is one of the most important part of the pipeline of the Mapper algorithm that determines the range of subsets (group of points) to choose over a space or dataset X. This take place right after the dimensionality reduction of the

dataset X. The background to this Subsection 3.3 is mainly motivated by the book [21] on topological spaces. Given a finite covering $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ of a space X, the Nerve Theorem 2.5.3 can help us to obtain a map from X to $N(\mathcal{U})$. A partition of unity subordinate to the finite open covering $\mathcal{U}$ is a family of real valued functions $\{\varphi_{\alpha \in A}\}_{\alpha \in A}$ with the following properties.

- $0 \leq \varphi_\alpha(x) \leq 1$ for all $\alpha \in A$ and $x \in X$.
- $\sum_{\alpha \in A} \varphi_\alpha(x) = 1$ for all $x \in X$.
- The closure of the set $\{x \in X | \varphi_\alpha(x) > 0\}$ is contained in the open set U_α.

Now assume that we are given a space equipped with a continuous map $f \colon X \to Z$ to a parameter space Z, and that the space Z is equipped with a covering $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$, again for some finite indexing set A. Since f is continuous, the sets $f^{-1}(U_\alpha)$ also form an open covering of X. For each α, we can now consider the decomposition of $f^{-1}(U_\alpha)$ into its path connected components, so we write $f^{-1}(U_\alpha) = \bigcup_{i=1}^{j_\alpha} V(\alpha, i)$, where j_α is the number of connected components in $f^{-1}(U_\alpha)$ [10]. We write $\tilde{\mathcal{U}}$ for the covering of X obtained this way from the covering $\mathcal{U}$ of Z.

Definition 3.3.1 *Consider two coverings $\mathcal{U} = \{U_\alpha\}_\alpha \in A$ and $\mathcal{V} = \{V_\beta\}_\beta \in A$ of a space X. A map of coverings from $\mathcal{U}$ to $\mathcal{V}$ is a function $f : A \to B$ so that for all $\alpha \in A$, we have $U_\alpha \subseteq V_{f(\alpha)}$ for all $\alpha \in A$.*

Example 3.3.1 *Let $X = [0, N] \times [0, N] \subseteq \mathbb{R}^2$. Given $\epsilon > 0$, we let $B_\epsilon(i, j)$ be the set $(i-\epsilon, i+1+\epsilon) \times (j-\epsilon, j+1+\epsilon)$. The collection $\{B_\epsilon(i, j)\}$ for $0 \leq i, j \leq N-1$ provides a covering $\mathcal{B}_\epsilon$ of X, and the identity map on the indexing set $\{(i, j) | 0 \leq i, j \leq N-1\}$ is a map of coverings $\mathcal{B}_\epsilon \to \mathcal{B}'_\epsilon$ whenever $\epsilon \leq \epsilon'$.*

From the Figure 3.3.1 we see the interval associated with the filter function. A filter function is any function on data cloud which is constructed with a purpose to reduce dimensionality while preserving some important features of the data. An example of a filter function is a projection from higher dimensional data cloud onto of the coordinate axis.

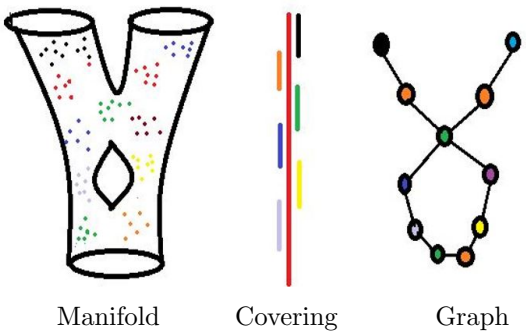


Figure 3.3.1: The pictorial representation of how a covering overlap and

These coverings are the colours that lies beside the filter line in the figure labeled Covering. In this diagram, we can think of f as a map (a projection) from the object labeled Manifold to the object labeled Covering. The open covering for the $Im(f)$ consisting of six overlapping intervals is marked with six different colours. Observe that the coloring highlights the overlap of the pullbacks for each interval, allowing us to see which points in the original point cloud is mapped to which interval.

It is clear that the length of the interval covering over a specific range or area contribute to the size of node in the graph in the Figure 3.3.1. Tuning this parameter is very significant to the results of the output from the Mapper algorithm.

3.4 Clustering

The idea of clustering is a natural continuation of coverings considered in the previous Section 3.3. After we pull back the cover using filter function, the dataset X is organized into bins which identifies the cluster for the Mapper algorithm. There are many clustering algorithms that have been embedded with an interface for the user to choose the range of ϵ(the radius of the cluster) and minimum points to be represented in a clustering $MinPts$. Local clustering is achieved based on points which are mapped by the filter function to the same element of a cover.

The clusters correspond to the nodes in the Mapper graph as shown in the A and C part of Figure 3.4.1. Clusters (nodes) with non-empty intersection of points mapping to overlapping elements of a cover are then joined by an edge in the graph as demonstrated in diagram C. If three or more clusters of points have non-empty

intersection mapping to three or more overlapping elements of a cover, we get higher simplices in the Mapper complex. The Mapper construction thus creates a simplicial complex of clusters representing the structure of data under the chosen filtering function plus their covering intervals. This modification to standard clustering gives more insight into the global structure of data through simplicial constructions.

In Figure 3.4.1 the original cloud dataset X (labeled A) is filtered using the projection to the vertical axis to reduce dimensionality. We put a cover consisting of three intervals on the vertical axis and pull this cover back to the space X. The next step in Mapper is to apply a clustering algorithm to the points in each bin. For each cluster found in a bin, a node of the resulting graph is created. Figure 3.4.1 shows the clustering with circles on the points and these areas are then represented as nodes (a node corresponding to each cluster) in C while the intersections of the clusters are represented by the edges connecting respective nodes.

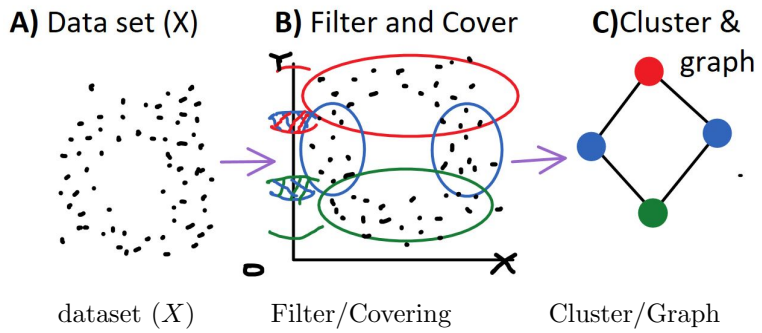


Figure 3.4.1: A) Mapper algorithm on point cloud data (X). B) We project the data to embedded space $\mathbb{R}$. We put data into overlapping bins. C) Clusters as the nodes of the graph are connected if there is a nonempty intersection.

Currently, no specific clustering algorithm has been assigned to the Mapper. The prospective clustering algorithms give different clustering results; the clustering of X varies depending on the choice of the clustering algorithm. There are clustering algorithms that do not produce a filtration of coverings as bin size increases. For example, as explained in Chapter 1 (Introduction), KMeans is a clustering algorithm used previously on the data from the Loneliness survey but it does not produce filtration of coverings as bin size increases.

For some other clustering algorithms like Single-linkage, it was proved [32] that

they produce filtration of coverings as bin size increases when the same scale (i.e. cutting height for the dendrogram [32]) is used in each bin and clustered independently. Hence, single-linkage algorithm in each bin, however, this does not give a filtration of coverings. Also the same paper demonstrate that complete-linkage does not give a filtration of coverings as bin size increases when the same scale is used in each bin.

We rely on the most widely used clustering algorithm in Mapper called Density-Based Spatial Clustering of Applications with Noise, or DBSCAN for short. When DBSCAN is used and bin size increases, there is a filtration of coverings. This algorithm is described in the next subsection.

3.4.1 Density-Based Spatial Clustering of Applications with Noise

DBSCAN is a density-based clustering algorithm that identifies regions with adequately high density into clusters and discovers clusters of arbitrary structure in spatial datasets by taking care of the noise in the background of where the data lies. Background noise points may give misleading information about the underlying space of the dataset unless they are identified as noise. One can think of noise points as a low density set of points. We noticed that our loneliness data contains some noise, which is mostly the scattered points that are neither a core point nor a border point. We would like to identify those points as noise and not include them in any of the clusters. In both KMeans and hierarchical clustering, those noise points are not identified, but with DBSCAN clustering noise points are identified [32].

DBSCAN groups points together as clusters on the basis of so-called density-connectedness, which is defined below. It also defines a cluster as a maximum set of density-connected points from the point cloud. A density-based cluster (node) is a set of density-connected objects that is maximal regarding density-reachability. This is an important feature of DBSCAN distinguishing it from other clustering algorithms. For example when using KMeans clustering on our data with $k = 4$ (where k is the number of clusters), we obtain four clusters where each cluster

contains points from two density connected subsets resembling lines, which are not density connected to each other. Therefore a KMeans cluster contains points from two different components. Even if the number of clusters is increased to $k = 5$, similar results are obtained.

Everywhere below, $N_\epsilon(x)$ denotes a closed ϵ-neighbourhood of a point x in the point cloud (a closed ball of radius ϵ centered at x.)

Definition 3.4.1 *Let $\epsilon > 0$ and $MinPts$ be two parameters. We define core points, border points, and noise with respect to ϵ and $MinPts$ as follows.*

- *A point p is a core point if $|N_\epsilon(p)| \geq MinPts$.*
- *A point s is a border point if $|N_\epsilon(s)| < MinPts$, and $s \in N_\epsilon(p)$ for p a core point.*
- *A point r is noise if r is neither a core point nor a border point.*

Definition 3.4.2 *A point s is directly density-reachable from a point p with respect to ϵ and $MinPts$ if s is in an ϵ-ball centered at p and there are at least $MinPts$ points in the ϵ ball centered at p. Note that p needs to be a core point, and $s \in N_\epsilon(p)$.*

Definition 3.4.3 *A point s is density-reachable from a point p with respect to ϵ and $MinPts$ if there is a sequence of points $p = \alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n = q$ such that α_{i+1} is directly density-reachable form α_i for all i.*

Definition 3.4.4 *A point p is density-connected to a point s with respect to ϵ and $MinPts$ if there is a point o such that both p and s are density-reachable from o with respect to ϵ and $MinPts$.*

Definition 3.4.5 *Let p,q, and s be points in a given dataset. We say s is a free-border-point with respect to $\epsilon > 0$ and a parameter MinPts if*

- *$s \in N_\epsilon(p)$ and $|N_\epsilon(p)| \geq MinPts$,*
- *$s \in N_\epsilon(s)$ and $|N_\epsilon(s)| \geq MinPts$,*

- $|N_\epsilon(s)| < MinPts$, *and*
- p *is not density connected to* q.

Definition 3.4.6 *Let X be a dataset of points. A cluster C with respect to ϵ and MinPts is a non-empty subset of X satisfying:*

- *(Maximality) $\forall p, s$ if $p \in C$ and q is density-reachable from p with respect to ϵ and $MinPts$, then $s \in C$.*
- *(Connectivity) $\forall p, s \in C$ we have p is density-connected to q with respect to ϵ and MinPts.*

In DBSCAN, there is a filtration of coverings in the absence of free-border points. It also shows a $MinPts$ decrease when DBSCAN is used, there is a filtration of coverings in the absence of free-border points [32].

Suppose $MinPts_0 \geq MinPts_1$. An ϵ-neighborhood, $N_\epsilon(p)$, of a core point p with respect to ϵ and $MinPts_0$ contains at least $MinPts_0$ points. That is, $|N_\epsilon(p)| \geq MinPts_0$. Hence $|N_\epsilon(p)| \geq MinPts_0 \geq MinPts_1$. Therefore, if p is a core point determining the cluster $C_p^{MinPts_0}$ with respect to ϵ and $MinPts_0$ points, then p is also a core point determining the cluster $C_p^{MinPts_1}$ with respect to ϵ and $MinPts_1$.

Points not contained in some cluster (node) are considered to be noise. DBSCAN checks for clusters by checking the $\epsilon-$neighborhood of every point in the dataset. If the $\epsilon-$neighborhood of a point p contains more than $MinPts$ points, a new cluster with p as a core element is produced. DBSCAN iteratively assembles precisely density-reachable objects from these essential elements, which can include the merger of a few density-reachable clusters. The process terminates when no new point can be added to any cluster.

Simply put, the DBSCAN algorithm detects whether or not the first point, $x_1 \in X$ is a core point. Then the next data point, x_2, is evaluated, until for all the points it is determined whether or not they are core points. If x_1 is a core point, DBSCAN returns a cluster (we name this cluster/node C_1) based on x_1. That is, DBSCAN

returns a cluster containing the core point x_1. Then, the data points that are in C_1 are "removed". We let the remaining set be denoted by $X - C_1$. Next, the DBSCAN clustering algorithm determines a cluster based on a core point in $X - C_1$. This process continues until every point is clustered, and points that do not belong to any cluster are labeled as noise points.

Now, suppose our $x_1 = q$ where q is a arbitrary point far away from a core point p (or x_1 be any of the points in the ball centered at q with the radius ϵ). Then points that are density-reachable from q with respect to ϵ and $MinPts$ are clustered together. Assume s is a point directly density reachable from both points q and p with respect to ϵ and $Minpts$. Since the border point s and all the points in the ϵ-ball centered at q are density-reachable from q with respect to ϵ and $MinPts$, this cluster includes specifically those points. Hence, point q, the points in the ball centered at q, and point s are "removed". The first element in the remaining data points is either p or any of the points in the ball centered at point p. The algorithm then clusters those points together because any of those points is density-reachable from p (or from any of those points) [32].

The cases bellow summarize the preceding discussion showing how a core point and a border point can be considered as part of a cluster.

1. Case $A : x_1$ is q or x_1 is any of the points in the ball centered at q,
2. Case $B : x_1$ is p or x_1 is any of the points in the ball centered at p,
3. Case $C : x_1 = s$, and x_2 is q or x_2 is any of the points in the ball centered at q,
4. Case $D : x_1 = s$, and x_2 is p or x_2 is any of the points in the ball centered at p,

For each value of $x_1 \in X$, the Python Scikit Learn clustering algorithm DBSCAN was used. Note that if we ran the DBSCAN clustering algorithm multiple times on the same dataset, we will find that in cases A and C, the same point s will be

clustered with point q, and in cases B and D, the same point s will be clustered with point p. This demonstrate the consistency of DBSCAN. Once the data entry is fixed, DBSCAN will continue to output identical clusters no matter how many times the algorithm is run. That is, every time the dataset is clustered with DBSCAN, we will achieve the same results as long as the order of the dataset is the same[32].

Again, if there is a border point s, density-reachable from two (or more) core points, p and q, such that neither p nor q is density-reachable from s, q is not density-reachable from p, and p is not density-reachable from q, then s belongs to a cluster containing p or q (but not both) depending on which core point in these clusters is listed first.

DBSCAN takes its input as an ordered set of points, and two parameters, ϵ and $MinPts$. It also identifies the core points with respect to ϵ and $MinPts$. Based on how the dataset is ordered, clusters are formed with respect to ϵ and $MinPts$, and any point that is not found in any of the clusters is labeled as a noise point. We summarize below the output of DBSCAN.

Notation 3.4.1 *The DBSCAN algorithm represents the dataset as* $X = C_1 \cup C_2 \cup C_3 \cup ... \cup C_n \cup N$ *such that* N *is a set of noise points and* C_i*, a non-empty subset of* X*, is a cluster with respect to* ϵ *and* $MinPts$ *for each* i*, satisfying:*

- *(Maximality)* $\forall p, q$: *if* $p \in C_i, q \notin \bigcup_{j=1}^{i-1} C_j$*, and* q *is density-reachable from* p *with respect to* ϵ *and MinPts,then* $q \in C_i$*.*

- *(Connectivity)* $\forall p, q \in C_i$: p *is density-connected to* q *with respect to* ϵ *and MinPts.*

We illustrate how DBSCAN clustering algorithm works. Assume a dataset of an arbitrary 70 points in 2D where the distance between two consecutive points is 0.5 (for reference and to visualize the process, the reader may look at Figure3.4.1). Consider $MinPts = 20$ and $\epsilon = 0.8$. A cluster, according to the Definition 3.4.6, satisfies the two conditions, namely, Maximality and Connectivity. We assume again that there are 20 points in $N_\epsilon(p)$, hence $|N_\epsilon(p)| \geq MinPts$, and p is a core point with

respect to $\epsilon = 0.8$ and $MinPts = 20$. DBSCAN gives one cluster which contains all of the 20 points. In this case, the maximality condition is satisfied because every point is density-reachable from p, and every point is in the same cluster as the point p. And, the connectivity condition is satisfied because any two points in the cluster are density-connected. Note that we would get the same cluster for $MinPts \leq 20$ with respect to the same ϵ. However, if $MinPts \geq 21$, then no cluster exists with respect to ϵ and $MinPts$ since for any point x in the dataset, $|N(x)| \leq 20$. Hence, there will just be noise because the number $MinPts$ determines how dense an $\epsilon-$ neighborhood of a point should be for there to be a cluster.

One issue with the DBSCAN clustering algorithm is determining the parameters that define density, i.e. ϵ and $MinPts$. If the right ϵ and $MinPts$ are not selected, we might have no clusters or too many clusters since the distance may not be known by the user. It takes multiple runs to get satisfactory ϵ and $MinPts$ for clustering with DBSCAN.

3.4.2 Graphing

This subsection only describes the cluster formed by the DBSCAN clustering algorithm and the intersections between these clusters. The clusters are referred as the nodes or vertices while the intersections are represented by the edges between the nodes. The part C of Figure 3.4.1 demonstrates a clear sample of the formation of the graph after the clustering in Part B.

4 Chapter 4: Methodology

The BBC Loneliness Experiment team conducted an international study for loneliness in 2018 indicating that open ended responses were obtained online. The participants comprised a large number of people that agreed to respond to the survey and it had the total number of responses of 55,203 participants. Participants were asked to define loneliness in their own words. Of these, about 71% (39444) completed the free text question ''What does loneliness mean to you?". The participants' age ranged from 16 to 99.

Most studies under text analysis do not preprocess the raw texts because these texts are cautiously edited formal documents. Conversely, our textual data needs some preprocessing because open-ended responses are informal and commonly filled with errors. Overall, the preprocessing is technically challenging especially when the size of the data is large. Also, these operations cannot be carried out with just a simple training data since it will demand expert to code the trained data.

4.1 Natural Language Processing (NLP)

Many applications of TDA takes in numbers that form points or matrix in the space or point cloud. Applying TDA to Natural Language Processing is still challenging no matter what form of text you may be dealing with. As TDA exploits the shape of data, it is not clear how one should interpret the geometry of text, document or language.

Many NLP problems start with the transformation of natural language into some sort of numerical representation. The motivation is to map text data to a form more amenable to computation. Primarily, this form is represented as a real-valued vector that attempts to encode the "meaning" or "importance" of a vocabulary. Moreover, this embedding is usually constructed in such a manner that vocabularies with similar meaning are expected to be "close" together.

4.2 Preprocessing

We visualize the data with the wordcloud and frequency algorithm in natural language tool kits (NLTK) Python to see the most frequent words. We then focus on dealing with the major typos in the open-ended responses since it will take us days to manually identify all errors. The words with highest frequency help us to see if there is any major spelling mistakes by looking at the whole or most of the words according to frequency. This helps in enhancing the coding performance of computerized content analysis. The cleaning process is explained in the next paragraph.

We first tokenize our text. Tokenisation refers to splitting of sentences into words, characters, punctuation marks, all of which are called tokens. The splitting happens mainly at the occurrence of a space or a punctuation. This step helps in filtering out unwanted words in further processing steps. Consider, for example, a response *"Loneliness is when there is no one to understand you and people ignoring you all the time !"* This sentence will be tokenized as *"Loneliness", "is", "when", "there", "is", "no", "one", "to", "understand", " you", "and", "people", "ignoring", "you", "all", "the", "time", "!"*

Secondly, symbols (e.g., @, $, &), punctuation marks (e.g.!, ?, /), emojis and numerals (e.g., -99) were deleted in open-ended responses because our main aim is to get a bag of words for our analysis. The algorithm we used to convert our text data to numerical data only takes text. Also, all upper-case words (e.g., "SAD", "WHO") were replaced by lower-case words (e.g., "sad", "who") to keep consistency of the notation across survey respondents.

Thirdly, stopword removal is very important for our process since, the most frequent words often do not carry much meaning. Examples: "the, a, of, for, in, . . . ". You can also create your own stopword list for your application domain if you realise there are some specific words that score high frequency but do not contribute to your analysis. In our case, we added "loneliness" to the stopwords since most people used this word in their responses.

Finally, we applied stemming (lemmatization) procedures, such as replacing "lov-

ing" with "love", "travelled" with "travel", and "wrote" or "written" with "write" in order to reduce irrelevant variance in considering the most important vocabulary in the entire textual data. The Natural Language toolkit (NLTK) is a suite for programs that provides us with libraries to do all the preprocessing in Python. This is a complex process, since there can be many exceptional cases (e.g., department vs depart, be vs were). The most commonly used stemmer is the Porter Stemmer. Notwithstanding, there are many others but they are not perfect and they do make mistakes. Many are not designed for special domains like social work terms. Some other languages (e.g., Spanish) are particularly hard.

4.3 Feature Extraction

Feature extraction simply means the representation of features in vector or matrix forms to make it understandable or interpretable to the machine in terms of computation. This preparation is fed to the filter algorithm (T-SVD).

4.3.1 Term Frequency and Inverse Document Frequency (TF-IDF)

The TF-IDF is a standard algorithm for converting texts to vector representation, that is used to penalize the vocabularies that are frequent but hold less value in the context of a document. As the name suggests, it has two parts, namely Term Frequency and Inverse Document Frequency. Term Frequency (TF) refers to the frequency of a word in a document without biasing the size of the document. This is because it is divided by the length of document as a normalization technique, thus giving fair accuracy to both small and huge data sets [33]. TF is calculated as:

$$TF(d,t) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{12}$$

where $f_{t,d}$ is a count of a particular term or word in a document that is, the number of times that the term t occurs in document d. Note the denominator is simply the total number of terms t' in document d (counting each occurrence of the same term

separately).

Its counterpart Inverse Document Frequency (IDF) finds the relevance of a word in the document. It basically scales down the words of less relevance by computing log of ratio of total documents to the total occurrences of the word in that document. This also address the issue of IDF not being a good discriminator by considering the importance of terms in relation to the total number of documents and to the number of documents in which the term is contained. IDF is calculated as:

$$IDF(t) = \log \frac{1 + |d|}{|d_t|} \tag{13}$$

where d is the total number of documents and d_t is the number of documents in which the term t is contained.

The Term Frequency - Inverse Document Frequency (TF-IDF) statistic weights terms by combining how frequent a term is in a document (TF) with how rare the term is with respect to the entire document set (IDF) [34]. TF-IDF is calculated as:

$$TF - IDF(d, t) = TF(d, t) \times IDF(t) \tag{14}$$

where d represents a document, t represents a term. We apply this to our clean responses to produce a matrix which is then fed into our two algorithms from TDA, Persistent Homology and Mapper, for computation. We will see the outcome of TF-IDF in the next chapter.

Some software packages available to implement Mapper are Ayasdi, giotto-tda, Mapper Interactive, TDA Mapper for R package, Scikit-TDA etc. We will be working with Scikit-TDA: Kepler Mapper which has been made flexible for implementing the Mapper algorithm. KeplerMapper can be used for visualization of high-dimensional data and 3D point cloud data. KeplerMapper can make use of Scikit-Learn API compatible cluster and scaling algorithms[35]. We shall refer to this library as KMappper.

5 Chapter 5: Results and Discussion

In this final section, we present the result from our experiment using the Persistent Homology and Mapper algorithm to capture some insight from responses on "What does loneliness mean to you?". After receiving the data in CSV format, we took the data through preprocessing where punctuation signs, stopwords and characters were removed.

We passed our clean text to the TF-IDF algorithm for feature extraction where vocabularies are scored according to it importance. This process creates the transformation of text to vectors in a $n \times m$ matrix form. The exact matrix dimensions were 39444×18677, where 39444 is the number of rows of the matrix (the total number of valid responses from the survey) and 18677 is the number of columns of the matrix (the total number of vocabularies). In this matrix, the high score in a cell signifies the importance of that particular vocabulary in the survey and vice versa. This matrix now presents itself as a point cloud.

We chose T-SVD and UMAP for filter function in this work following our discussion from Subsection 3.3.1. Note, in particular, that T-SVD is a popular method for dimensionality reduction when it comes to doing better work on sparse data but not securing a global structure and UMAP algorithm on the other hand performs well on capturing the global structure hence we combine the two to produce a better structure for our analysis.

The T-SVD reduced the number of columns from 18677 to 100. This was possible due to the sparsity of the matrix. We now obtain a matrix of size 39444×100. This was done with T-SVD due to it robustness on sparse matrices. The UMAP algorithm was later employed to reduce the same columns from 100 to 12, hence we now present each row of responses as a vector in a $12-$dimensional space. The matrix of size 39444×12 was the result of reduction of our dataset to compute persistent homologies.

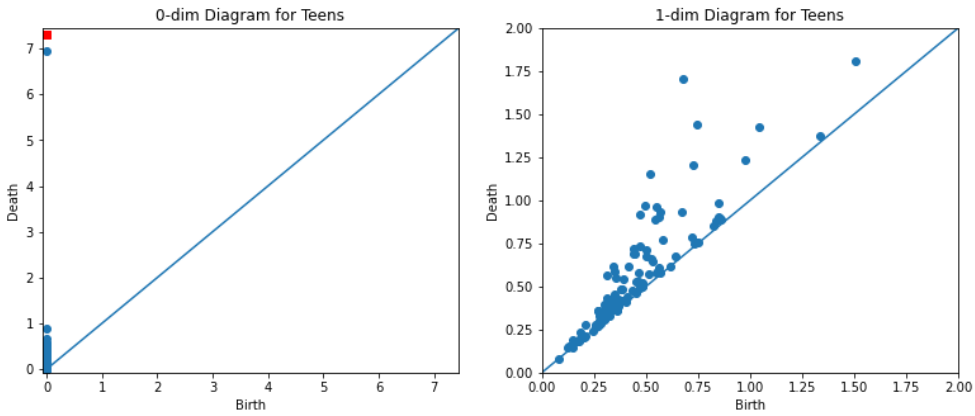
We summarize topological information of age categories, those who identify themselves as lonely and people living alone. We use barcodes and persistence

diagrams to get information leveraging on the theory of persistent homology from Section 2. Also, we use Mapper Algorithm outlined in Section 3 to analysis those respondents who identify themselves as lonely and people living alone.

5.1 Persistent Homology

Let us start by examining the data set to generate the categorical data based on the age range assigned to each category. Respondents' ages ranged from 16 to 99 where "16-19" was named Teen category, "20-39" as Youth category, "40-49" as Matured-Youth, "50-59" as Matured, "60-69" as Retiring, "70-99" as Pension and those with no response to age were represented as BlankAge. Each group of age category is presented and plotted in the point cloud. We show a computed birth and death of a connected component and a loop in a persistence diagram of the Rips filtration constructed on the Teen category point cloud, see Figure 5.1.1.

As the point cloud moves through the Rips filtration, we go ahead to draw an arbitrarily small circle around each point in our connect-the-dot principle explained in Section 2. Then we slowly start to draw larger and larger circles around each point until two or more of those circles intersect. When two points' circles intersect, we connect them. As we begin to connect more points (a nested sequences of simplicial complex), loops will start to appear (and eventually disappear) in our Teen category and individual components will start to disappear. For each component or loop, we plot a point (x, y) in our persistence diagram so that x represents the birth of the hole or loop (radius of the circles when the loop first appeared) and y represents the death of that hole (radius of the circles when the loop disappeared). This creates H_0 and H_1 persistence diagrams shown in Figure 5.1.1. First, the blue points along the death axis ($y-$ axis) in the Persistent diagram (H_0) 5.1.1 represent the 0-dimensional cycles or vertices. This means cycles in H_0 and all those blue points and the red point start at time zero but can end at different times during the time evolution of the Rips filtration. In $dgm\ H_1$ 5.1.1, we observe many points in the diagram H_1 are clustered around the line $y = x$(diagonal). The points on the line $x = y$



Persistent diagram (H_0) Persistent diagram (H_1)

Figure 5.1.1: Teenage category representation of H_0 and H_1

are loops that easily die after they are created. Those points further away from the line $y = x$(diagonal) are considered persistent, and this is where our interest lies (these are features that show more characteristics of the global structure of the point cloud). The ones considered persistent record the holes or loops and indicate how long these loops persist by taking into consideration the birth and death time of the loop.

Our aim in this work is to use persistence homology to examine the shape and relationships between the age groups by pairing the categories to measure distance from the loops created by the age group point cloud. All the categories shown above has been analyzed in the Figure 5.1.3 and 5.1.4. We used the bottleneck distance to compute a statistic for each relationship among the data sets. The Figures 5.1.3 and 5.1.4 below show persistence diagrams that captures holes in each data sets at H_1 (here H_1 stands for 1-dimensional persistent homologies).

The Figure 5.1.2 shows a combination of all the datasets illustrating the category with the highest persisted hole. We can have an earlier conclusion of close responses due to how fast holes created dies off in the Persistent Diagram 5.1.2

The points in the diagrams 5.1.2 demonstrate the time holes appear (birth) in the dataset and when they disappear (death). The points close to the diagonals are the ones that disappear quickly. The further the point from the diagonal, the longer the hole persists. The red lines in the persistence diagrams show the distance between

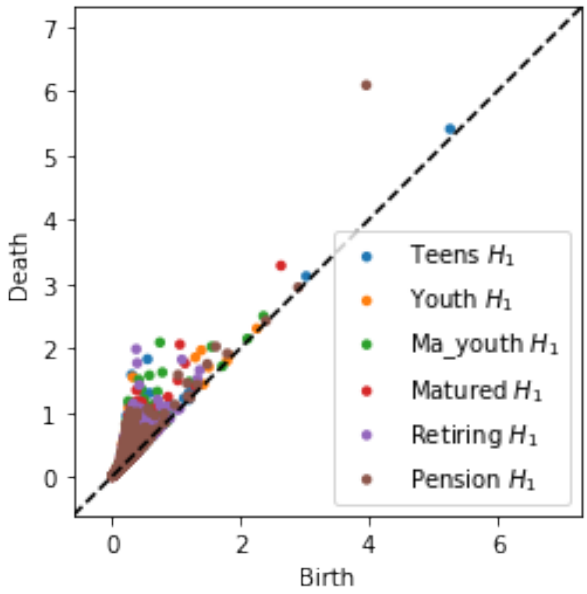
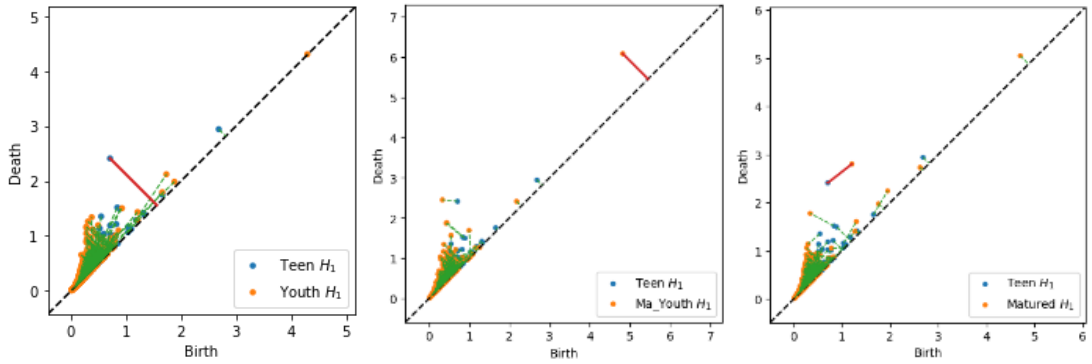


Figure 5.1.2: All category representation of H_1

any two datasets compared analogous to the previous paragraph in Figure 5.1.3 but we consider these cases in H_1 only.

We found the "Pension" age category to have the a hole persisted for a longer time than the rest of the categories. The holes signify a distance or variation in their responses. The longer the red line, the higher the distance score between the two data set measured. The bottleneck distance algorithm measures it's distance by choosing a point far from one data set and matching it to the diagonal, when it appears as the minimal matching, or else it chooses a point in the other data set closer to it. To understand this matching and measuring of similarities between the two persistent diagrams, let us consider $dgm(b)$ and $dgm(c)$ in Figure 5.1.3.

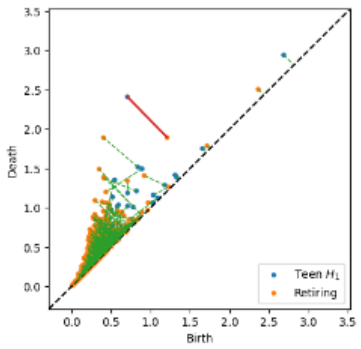
The orange point in $dgm(b)$ representing "Mature-Youth" is connected to the diagonal line($x = y$) but not the blue point since that is further away than connecting the diagonal line. For the $dgm(c)$, it is clear that the persisted loop from the blue point representing the "Teen" gets a minimal matching by connecting with the closest orange point representing the "Matured" group. This examples can be seen across the multiple persistent diagrams presented in Figures 5.1.3 and 5.1.4. The actual statistic values were computed and has been presented in the next subsection using bottleneck distance.



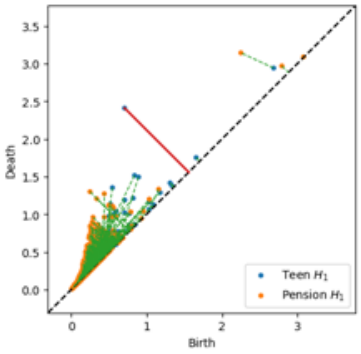
(a) Teenage and Youth

(b) Teenage and Matured-Youth

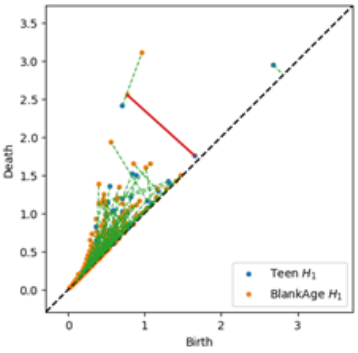
(c) Teenage and Matured



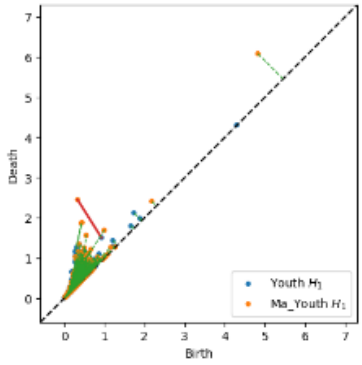
(d) Teenage and Retiring



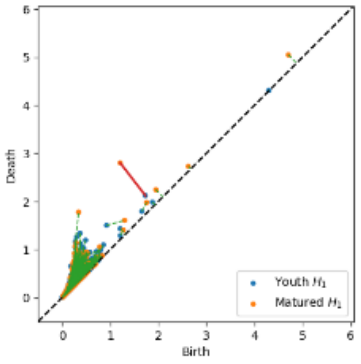
(e) Teenage and Pension



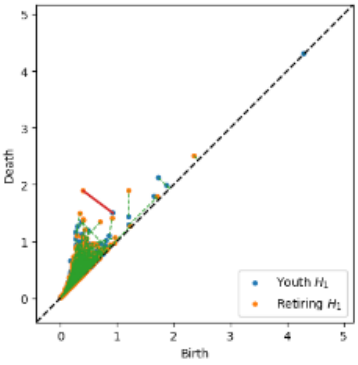
(f) Teenage and No response



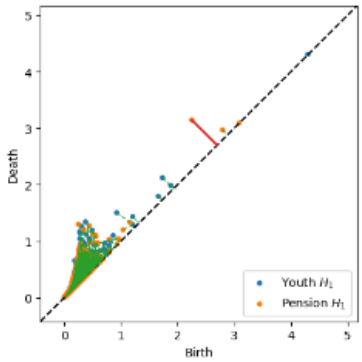
(g) Youth and Matured-Youth



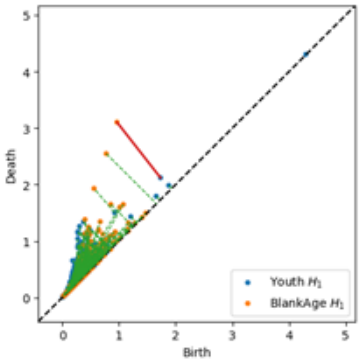
(h) Youth and Matured



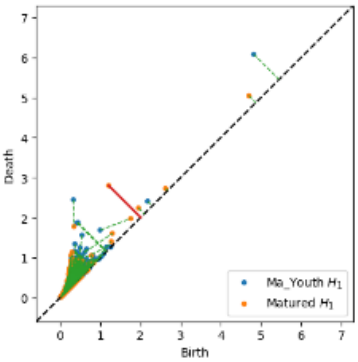
(i) Youth and Retiring



(j) Youth and Pension

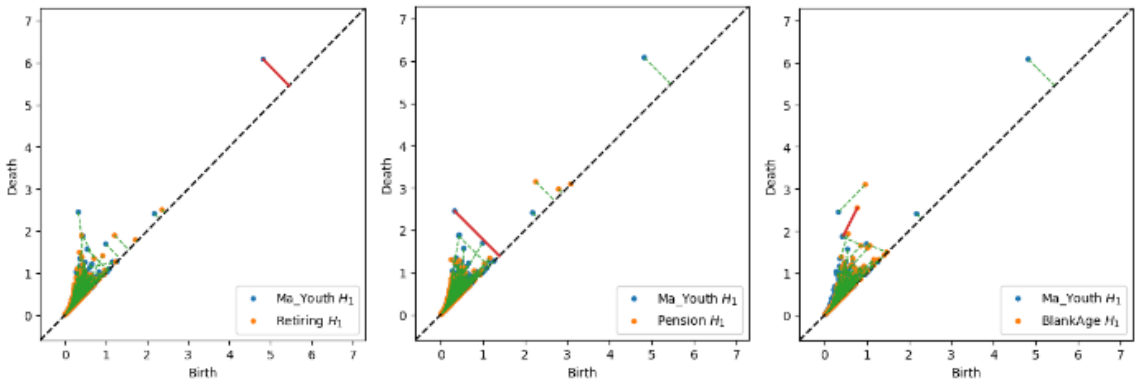


(k) Youth and No Response

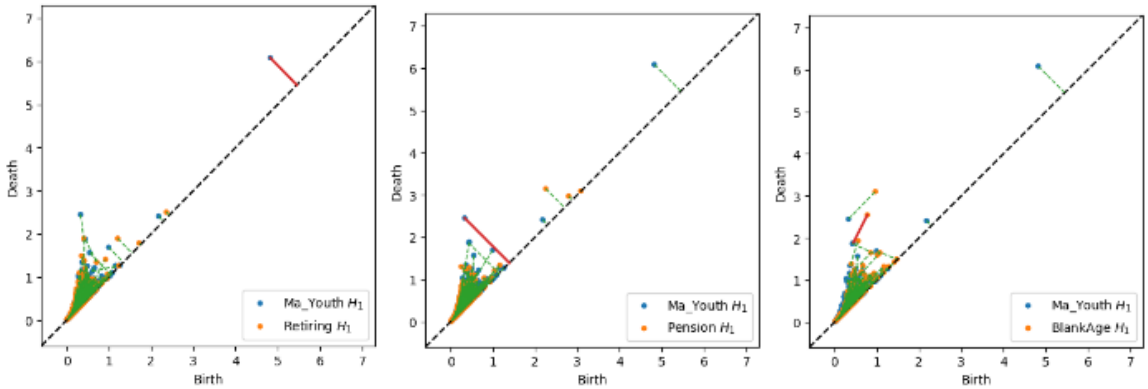


(l) Matured-Youth and Matured

Figure 5.1.3: Matchings between pairs persistent diagrams for age group categories in H_1 settings



(a) Matured-Youth and Retiring (b) Matured-Youth and Pension (c) Matured-Youth and No Response



(d) Matured and Retiring (e) Matured and Pension (f) Matured & No Response

Figure 5.1.4: Matchings between pairs persistent diagrams for age group categories in H_1 settings

5.1.1 Bottleneck Distance

The bottleneck distance gives a statistic to show how close the two data set paired are similar. It is the smallest distance for which there exists a perfect matching between the points of the two persistence diagrams as described in Section 2.6. We generate H_1 diagrams for each of the data sets to measure the similarity and the shortest distance was between the Youth and Pension groups which recorded 0.446. The Teens distance with Matured and Retiring followed as short distances of 0.501 and 0.516, respectively. The Retiring and Pension also showed close response with a distance of 0.592. Matured youth and Pension groups distance was slightly higher with 1.059, the Blank age group was also around the same number with retiring and pension scoring 1.069 and 1.069, respectively. Overall, the distances between the data sets from various groups were very close to each other.

In summary, persistent homologies captured the holes in our data which was visualised by the persistent diagram and the bottleneck distance computed the relations between each category.

	Teenage	Youth	Matured-Youth	Matured	Retiring	Pension	No response
Teenage	0.000						
Youth	0.851	0.000					
Matured-Youth	0.629	0.945	0.000				
Matured	0.501	0.680	0.796	0.000			
Retiring	0.516	0.522	0.629	0.796	0.000		
Pension	0.851	0.446	1.059	0.796	0.592	0.000	
No response	0.881	0.983	0.668	0.770	1.069	1.069	0.000

5.2 Mapper Algorithm

The mapper algorithm was used to show the relationship of words used by the living alone category. We dive deep into some clusters and call out some responses for readers to have some idea about how these responses were received and how similarities occurred between categories. The figures below gives us insight to the connection between responses, among the categories. In this subsection, we used Kepler Mapper [36] as the Mapper Python software package to implement all our graphs and plots.

Some of the questions in the survey inquired whether a participant lives alone and asked to identify themselves as lonely (or not). We used these two categorical features to examine the relationship between those that answered "yes" or "no". This may give us some information whether the definition of loneliness is influenced by those indicators or not. It is clear from the graph that more than 50% do not live alone. Also, more than 50% indicated they were not lonely. Although one third of the respondent live alone, we recorded less than one third as respondent being lonely. About 19% of the people could not respond to whether they are lonely or not.

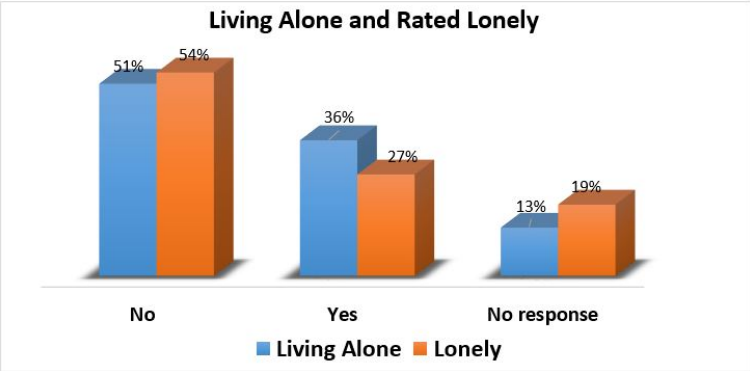


Figure 5.2.1: Living Alone and Lonely

Following the dimensionality reduction for the Persistent Homology, we do something similar, but for this time we reduce the exact matrix dimensions which is 39444×18677, to a matrix of size 39444×100 by using the T-SVD algorithm. We then use the UMAP to further reduce the dimension of the columns from 100 to

2 so we can present each row of responses as a vector in a 2−dimensional plane (x, y plane) while keeping the number of respondent constant. The matrix of size 39444×2 was the resulting dataset to apply our covering.

We passed our clean text to the TF-IDF algorithm for feature extraction where vocabularies are scored according to it importance. This process creates the transformation of text to vectors in a $n \times m$ matrix form. and 18677 is the number of columns of the matrix (the total number of vocabularies). In this matrix, the high score in a cell signifies the importance of that particular vocabulary in the survey and vice versa. This matrix now presents itself as a point cloud.

The T-SVD reduced the number of columns from 18677 to 100. This was possible due to the sparsity of the matrix. . This was done with T-SVD due to it robustness on sparse matrices. The UMAP algorithm was later employed to reduce the same columns from 100 to 12, hence we now present each row of responses as a vector in a 12−dimensional space. The matrix of size 39444×12 was the result of reduction of our dataset to compute persistent homologies.

The Figure 5.2.2 is a representation of participants that rated lonely. The first graph 5.2.2 on the left hand side shows a node (a group of members that formed a cluster based on their similar vocabularies). The connection of two nodes (vertices) by an edge can be interpreted as a strong similarity from one node to another. The nodes with the highest members turn out to be the ones with the highest connections. For example, we select one node from the highest connection and it is evident under the cluster statistics, that the size of members recorded is 1082. With the same analysis we turn to the right hand side, and select a node not connected to any node and under the cluster statistics. We record 18 members. The selection was shown with a marker around the node.

A sample of responses taken from the node with the size 1082 under the lonely category is shown in raw data below (As how it was received without any preprocessing). The first number is the identification number (ID) followed by gender category where ‘M’ is male, ‘F’ for female, ‘O’ for others and ‘P’ is prefer not to say. The

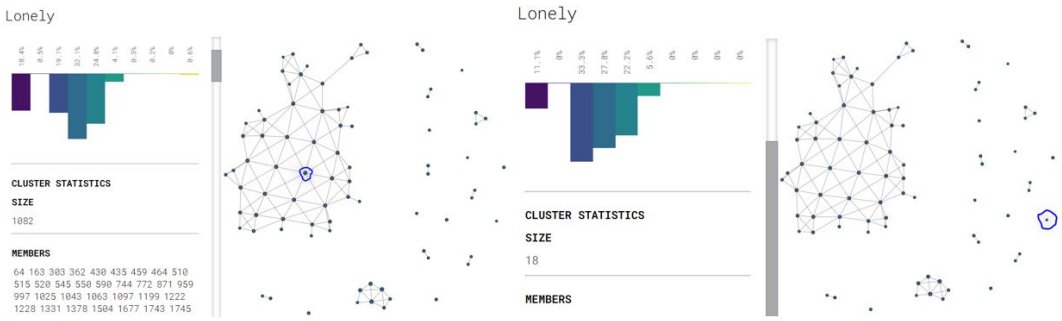


Figure 5.2.2: Mapper graph for people who rated themselves as Lonely

third number is a variable for age between the range of 16-99 and the last item is the response to what loneliness meant to them.

Sample 5.2.1 ***Lonely Category***

ID, Gender, Age, ***Response***

- ***163, M, 60****, "Having no-one to turn to at 4 in the afternoon or 4 in the morning",*
- ***303, M, 53****, " that you have nobody to speak to honestly who listens",*
- ***1745, F, 69****, "Lack of meaningful communication with other people",*
- ***430, F, 31****, "It means being or feeling alone. Like I currently am.",*
- ***435, F, 61****, "No one caring if I live or die.",*
- ***1677, M, 28****, "Not having someone to do nothing with. No one to share day to day news with. No one to mull over decisions with."*

Not lonely is a category that was of substantial size with a lot of members sharing their view on loneliness. The nodes in that category show some more colour variations from the highly connected nodes to the non connected nodes. The colours of these nodes demonstrate the density of each node (cluster) in higher dimensions according to the member distribution. Since we performed dimensionality reduction, the colours help to identify nodes with large number of members from the member distribution on the left hand of the Figure 5.2.3. The node distribution on the right

hand side of Figure 5.2.3 depict a clear size of nodes in percentages across the nodes. The colour of a node tells us about the frequency in terms of members available in each node as across the entire node.

It is also obvious that the clusters present in the Not Lonely category is scattered and disjointed more than the lonely category in visual terms. It is not of a high interest to recall responses from connected nodes and non connected nodes but we will show some responses from the selected node highlighted in the middle of the highly dense connected group of nodes. It has a member size of 1438 and is a dominated group as this is evident from the node distribution on the right hand side of the Figure 5.2.3. On top of the node distribution, we record a basic statistics of our graph with 111 nodes and 168 edges as well as the total and unique samples. These samples are the total number of points our algorithm produced.

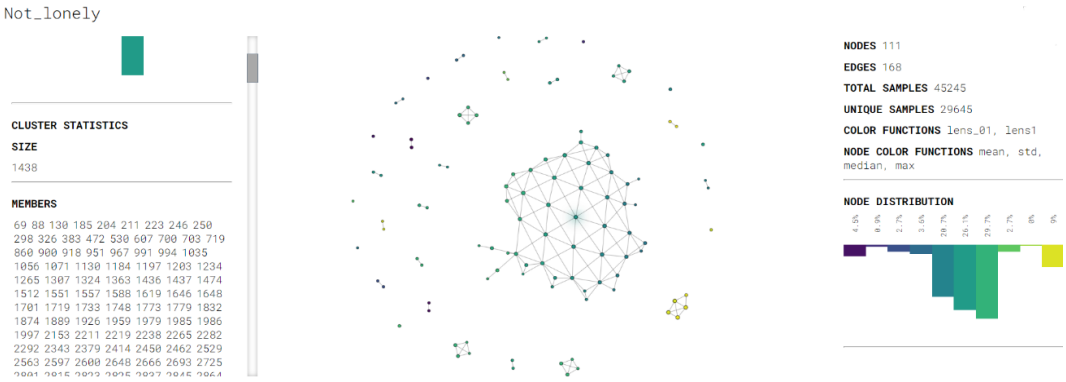


Figure 5.2.3: Mapper graph for people who rated themselves as Not lonely

A random selection of 6 responses out of 1438 members was selected to give the reader an idea of raw responses from particular age group and gender as well as the cluster they represent as being populated and shares similarities with other clusters. The rest of the information on the guide lines to read the raw text still remains the same from the previous paragraph on raw data samples.

Sample 5.2.2 *Not Lonely Category*

[ID, Gender, Age, 'Response']

- ***900, M, 70****, "Being unable to communicate, lacking reassurance, feeling empty and helpless",*
- ***951, F, 44****, "Lack of comfort, though also the time to do my own thing",*
- ***3990, F, 66****, "Feeling like no one wants to be with you or thinks you? re interesting",*
- ***4290, F, 60****, "no one to talk to, no one to turn to for comfort, encouragement",*
- ***4304, M, 24****, "not being able to share the good times as well as the not so good",*
- ***4359, F, 22****, "Despair. no one to confide in. Not being settled".*

Living alone or not living alone may or may not have any influence on the meaning of loneliness to a person. We analyzed the group of category that live alone and call out some samples of responses. The basic statistics of the Figure 5.2.4 is similar to that of Figure 5.2.2. In this analysis, we will concentrate on the node circle in the Figure 5.2.4 below to extract some insight. The first observation is the connectivity of the node to it neighbours which signifies a strong relationship between the responses not just in the same node (cluster) but different nodes (clusters). The second observation is the node distribution where the colour of the node shows 90% of the total node associated with the chosen cluster. Although members in the cluster

show a different distribution from the left histogram, we however record the highest percentage which is in line with the selected node. This simply means that a cluster has a various points and those points are in a higher dimension hence the colours demonstrating the relationship in the space.

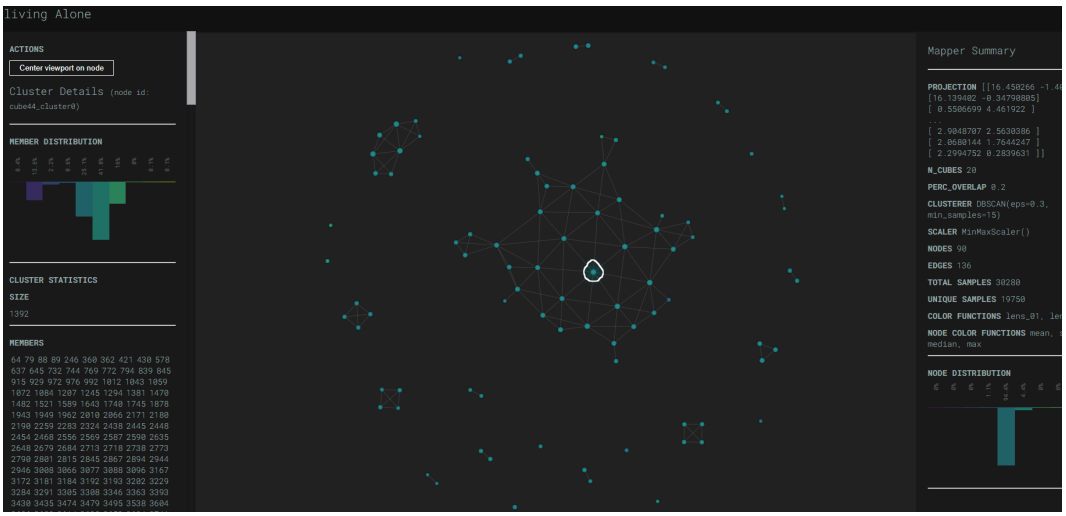


Figure 5.2.4: Mapper graph for people Living Alone

The circled node has a size of 1392 members across the responses which will be very hard to visual in this paper. We will show a few raw responses from respondents that live alone. The format for the samples remain the same from the previous ones.

Sample 5.2.3 *Living Alone Category*

[ID, Gender, Age, 'Response']

- ***5516, F, 58****, "empty house. being on my own. fear of the future",*
- ***5546, F, 54****, "If I'm not there I will not be missed. I always feel on the fringe",*
- ***36466, F, 61****, "Not having someone to do something with",*
- ***45523, M, 35****, "Being lonely is hurting on the inside with no way to ease the pain",*
- ***45620, F, 24****, "Be detached from main life course",*

- ***45369, M, 69***, *"Being alone in a crowd. Being ignored. Silence in an empty house".*

In the Figure 5.2.5 below, we consider three cases to conclude our understanding of node (vertex or clusters as we referred to) and edges (connection or relationships between two clusters). A selected node with highest edges connecting its neighbours is circled and labelled 'A' in Figure 5.2.5, a second highest connection of nodes but with a highest member size among the three nodes is labelled 'B', and the 'C' label is for a non-connected node with the lowest node distribution. The three selected nodes or clusters are in different sets of domain in terms of their association and member size. The 'A' has a node size of 1850, 'B' has a node size of 4550, and the 'C' has a node size of 61. Notwithstanding, our earlier statement of node distribution giving us the idea of relationship between colours and cluster size, cannot be possible in this scenario. This is due to the spread of that distribution. Although node 'B' contain the highest size as compared to node 'A' with the highest node distribution, colour nodes collectively will be the nodes with the largest size of members. The most connected nodes having the highest cluster size is a good sign for us to base our analysis on the majority but we should be careful when faced with a node distribution like what we have below.

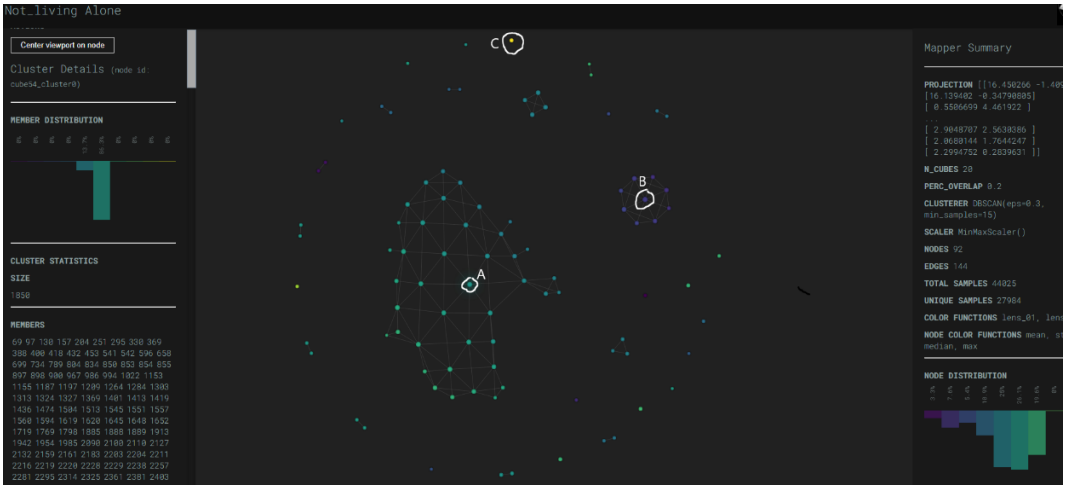


Figure 5.2.5: Mapper graph for people who Not-Living Alone

Since the Figure 5.2.5 demonstrate a busy colour node with more than half of the respondents in this category, we will concentrate on the selected nodes. Node 'A' holds a very vital position in the graph output in Figure 5.2.5 due to the colour, size and edges. With reference to the previous guide on how to read and under the raw data, we will implement that same understanding in the subsequent samples.

Sample 5.2.4 *Not Living Alone Category*

[ID, Gender, Age, 'Response']

- ***204, F, 28****, "Not having anyone to talk to",*
- ***699, F, 58****, "Loneliness is the lack of a companion. It is the lack of another to share your space, whatever that space may be",*
- ***2610, M, 44****, 'Feeling isolated despite having friends, feeling imprisoned by oneself, stuck",*
- ***4855, M, 60****, 'Loneliness means a feeling. Sadness. Aching. Physical sensations in the body"*

Node 'B' had the second highest edges connected to it neighbours but the highest member size among the 3 nodes selected. We will call some samples from this node to understand the language used in the cluster. Holding all guidelines to raw data constant, below are 6 randomly chosen responses from node 'B'.

Sample 5.2.5 *Not Living Alone Category*

[ID, Gender, Age, 'Response']

- ***9378, F, 22****, 'Internal silence with sounds happing around me that I am not part of",*
- ***13474, M, 52****, 'Loneliness I imagine means feeling isolated and cut off having no one to connect with",*
- ***9397, P, 50****, 'Isolation. Boredom. No meaning. No help. Feel Worthless. Nothing to look forward to. Pointless existence. Rejection",*

- ***13476, F, 33****, 'Being disconnected and missing out. Feeling an absence without a sense of when it will be over"*

We have witnessed a few responses from the previous raw data giving us an idea of what might have connected the participants together in a particular cluster. The last node labelled 'C' is a separated node with lowest size and no connection to any cluster. But the vocabulary used is interesting for our observation on similarities of words across the entire data. With the same guideline in reading the raw data, we show a few sample from the cluster labelled 'C'.

Sample 5.2.6 *Not Living Alone Category*

[ID, Gender, Age, 'Response']

- ***45967, M, 46****, 'Feeling alone. Social isolation. Unable to connect with others",*
- ***10854, M, 47****, 'Needing someone either physically or emotionally and not having those needs met",*
- ***14372, M, 18****, 'Not having anyone to talk to or trust. Feeling like an outsider",*
- ***37406, F, 23****, 'not having someone to talk to.not be able to share my sadness"*

Bibliography

[1] Elyasi, N.; Moghadam, M. H. An introduction to a new text classification and visualization for natural language processing using topological data analysis. *arXiv preprint arXiv:1906.01726* **2019**,

[2] Hämäläinen, W.; Joy, M.; Berger, F.; Huttunen, S. Clustering students' open-ended questionnaire answers. *arXiv preprint arXiv:1809.07306* **2018**,

[3] Otter, N.; Porter, M. A.; Tillmann, U.; Grindrod, P.; Harrington, H. A. A roadmap for the computation of persistent homology. *EPJ Data Science* **2017**, *6*, 1–38.

[4] Lum, P. Y.; Singh, G.; Lehman, A.; Ishkanov, T.; Vejdemo-Johansson, M.; Alagappan, M.; Carlsson, J.; Carlsson, G. Extracting insights from the shape of complex data using topology. *Scientific reports* **2013**, *3*, 1–8.

[5] Gholizadeh, S. Topological data analysis in text processing. Ph.D. thesis, The University of North Carolina at Charlotte, 2020.

[6] Robles, A.; Hajij, M.; Rosen, P. The shape of an image: A study of mapper on images. *arXiv preprint arXiv:1710.09008* **2017**,

[7] Lei, Y. Topological Methods for the Analysis of Applications. International Conference on Modern Educational Technology and Innovation and Entrepreneurship (ICMETIE 2020). 2020; pp 6–9.

[8] Zhu, X. Persistent Homology: An Introduction and a New Text Representation for Natural Language Processing. IJCAI. 2013; pp 1953–1959.

[9] Doyle, J.; Potvin, G. In search of distinct graduate admission strategies in physics: An exploratory study using topological data analysis. 2015.

[10] Singh, G.; M'Emoli, F.; Carlsson, G. Topological methods for the analysis of high dimensionaldata sets and 3d object recognition, p 91–100. *Prague, Czech Republic, Eurographics Association* **2007**,

[11] von Brömssen, E. Computing persistent homology in parallel with a functional language. **2021**,

[12] Griend, R., *et al.* Simplicial Complexes and Persistent Homology. B.S. thesis, 2020.

[13] Bender, K. J., *et al.* The Sense of Shape: Discovering Spatial Features of Data with Applied Topology. **2016**,

[14] Rieck, B. Persistent homology in multivariate data visualization. Ph.D. thesis, Ruprecht-Karls-Universität Heidelberg, 2017.

[15] Edelsbrunner, H.; Harer, J.; Natarajan, V.; Pascucci, V. Morse complexes for piecewise linear 3-manifolds. Proc. 19th ACM Symp. Computational Geometry. 2006; pp 361–370.

[16] Fischer, K.; Gärtner, B.; Kutz, M. Fast smallest-enclosing-ball computation in high dimensions. European Symposium on Algorithms. 2003; pp 630–641.

[17] Gärtner, B. Fast and robust smallest enclosing balls. European symposium on algorithms. 1999; pp 325–338.

[18] Dantchev, S.; Ivrissimtzis, I. Efficient construction of the Čech complex. *Computers & Graphics* **2012**, *36*, 708–713.

[19] Adams, H.; Aminian, M.; Farnell, E.; Kirby, M.; Mirth, J.; Neville, R.; Peterson, C.; Shonkwiler, C. *Topological Data Analysis*; Springer, 2020; pp 1–31.

[20] Choudhary, A. Approximation Algorithms for Vietoris-Rips and˘Cech Filtrations. **2017**,

[21] Munkres, J. R. *Elements of algebraic topology*; CRC press, 2018.

[22] Karvonen, E., *et al.* Persistent Homology and Its Applications. **2021**,

[23] Chung, Y.-M.; Lawson, A. Persistence curves: A canonical framework for summarizing persistence diagrams. *Advances in Computational Mathematics* **2022**, *48*, 1–42.

[24] Cohen-Steiner, D.; Edelsbrunner, H.; Harer, J.; Mileyko, Y. Lipschitz functions have L p-stable persistence. *Foundations of computational mathematics* **2010**, *10*, 127–139.

[25] Guo, W.; Banerjee, A. G. Identification of key features using topological data analysis for accurate prediction of manufacturing system outputs. *Journal of Manufacturing Systems* **2017**, *43*, 225–234.

[26] Doyle, J. Describing and mapping the interactions between student affective factors related to persistence in science, physics, and engineering. **2017**,

[27] McInnes, L.; Healy, J.; Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* **2018**,

[28] Hopcroft, J.; Kannan, R. Computer science theory for the information age. **2012**,

[29] Mac Lane, S. *Categories for the working mathematician*; Springer Science & Business Media, 2013; Vol. 5.

[30] Jackson, A. The mathematics of UMAP. **2019**,

[31] Coenen, A.; Pearce, A. Understanding UMAP; 2019. *URL https://pair-code. github. io/understanding-umap*

[32] Bungula, W. T. Bi-filtration and stability of TDA mapper for point cloud data. Ph.D. thesis, University of Iowa, 2019.

[33] Garcia Constantino, M. On the use of text classification methods for text summarisation. Ph.D. thesis, University of Liverpool, 2013.

[34] Bramer, M. Clustering. *Principles of Data Mining* **2007**, 221–238.

[35] van Veen, H. J.; Saul, N.; Eargle, D.; Mangham, S. W. Kepler Mapper: A flexible Python implementation of the Mapper algorithm. *Journal of Open Source Software* **2019**, *4*, 1315.

[36] Van Veen, H. J.; Saul, N.; Eargle, D.; Mangham, S. W. Kepler Mapper: A flexible Python implementation of the Mapper algorithm. *Journal of Open Source Software* **2019**, *4*, 1315.