Clustering in High-Dimension – Tools and Challenges

Lucy Liu & Haim Bar*

Department of Statistics, University of Connecticut, Storrs, CT

https://doi.org/10.33697/ajur.2026.173

*Students: lucy.liu@uconn.edu**
Mentor: haim.bar@uconn.edu

ABSTRACT
Dimensionality reduction methods such as Multidimensional scaling (MDS) or t-Distributed Stochastic Neighbor Embedding (t-SNE) are often followed by clustering in the reduced plot. To examine whether or to what extent these methods affect clustering, we simulate several data structures and apply clustering methods. We first perform clustering using the data in the original space, where we know the true clusters, then perform MDS and t-SNE to scale the data down to two dimensions, cluster on this projected data, and compare differences in the results. We find that MDS and t-SNE can either increase or decrease clustering performance, and are unable to correctly represent data structures with certain shape structures, or in the presence of noise. We examine several clustering methods and show that their performance depends to a large extent on the structure of the data, original dimension, and the noise level, even before we perform dimensionality reduction via MDS and t-SNE. No method among the ones considered here dominates the others in terms of clustering accuracy.

KEYWORDS
K-means; Hierarchical clustering; Ward linkage; Complete linkage; Average linkage; Single linkage; Persistence Diagram; Simulation

1. INTRODUCTION

Clustering plays a vital role in extracting meaningful patterns and structures from unlabeled data. The challenge of identifying coherent groupings within data spans diverse disciplines, including biology, finance, and social sciences. For example, clustering algorithms are often used in medical imaging, where the objective is to perform segmentation and identify tumors. However, the quality of clustering outcomes can be influenced by the latent characteristics of the data, such as the shape, dimensionality, and distribution of clusters.

The central problem addressed in this paper is: given data embedded in a high-dimensional space with unknown cluster geometry, which combination of clustering algorithm and dimensionality reduction method yields the most reliable results? This question has direct practical importance because selecting an inappropriate clustering method can lead to misleading interpretations—from misidentifying cell types in genomics to incorrectly segmenting populations in social science research. To address this problem, we conduct a systematic simulation-based study that highlights the behavior of existing clustering methods under controlled, restricted conditions. We vary three key factors—cluster shape (convex vs. non-convex), ambient dimensionality, and noise level—to provide empirical guidance for practitioners who must choose among these methods without knowing the true underlying structure of their data.

When presented with a dataset to cluster, is there an optimal clustering algorithm that consistently outperforms others, or does the quality of clustering of a certain method depend on the structural layout of the data? Different algorithms, such as K-means and hierarchical clustering, each have unique assumptions and mechanisms. A comparative understanding of these methods in relation to varying data configurations is crucial for guiding methodological choices.

Do certain algorithms demonstrate robustness, or do they require significant adaptation to handle such complexities? As datasets grow increasingly complex, high-dimensional data presents a unique set of challenges for clustering algorithms. The "curse of dimensionality" can obscure meaningful groupings, which makes it necessary to investigate how classical and modern clustering techniques perform in high-dimensional spaces. Understanding algorithmic behavior in these scenarios helps practitioners anticipate potential pitfalls and select more reliable clustering strategies.

Does dimensionality reduction invariably enhance clustering by removing noise, or does it introduce distortions that compromise results? Projecting high-dimensional data to a lower dimension offers several advantages, such as enabling visualization in two dimensions and improving computational efficiency. However, there is also a risk of misrepresentation, potentially introducing distortions that affect the outcome of clustering. Exploring these trade-offs is essential for determining when and how dimensionality reduction should be applied.

Furthermore, how does the spatial arrangement of data influence the effectiveness of these combined strategies? Dimensionality reduction techniques, such as Multidimensional Scaling (MDS)[1] and t-Distributed Stochastic Neighbor Embedding (t-SNE)[2], are often employed to mitigate high-dimensional challenges. However, their impact on clustering quality remains an open question, particularly for non-convex or complex data structures.

The process of dimensionality reduction may lead to misplacement of points. Namely, points that are close in the original data may end up far apart in the projected data, or vice versa. The impact of such misplacements on clustering quality has not been fully addressed.

This paper is organized as follows. In Section 2, we review related work and the application areas that motivate this study. In Section 3, we establish the necessary notation, justify our selection of clustering and dimensionality reduction methods, and describe each method in detail. In Section 4, we present results from an extensive simulation study. In Section 5, we summarize our main observations, and Section 6 contains practical advice and suggestions for future research.

2. RELATED WORK

The methods studied in this paper—K-means, hierarchical clustering, topological data analysis (TDA), MDS, and t-SNE—are well-established tools that have been applied across a wide range of domains. Understanding how they have been used in practice motivates the need for the systematic comparative study presented here.

In computational biology, dimensionality reduction followed by clustering is standard practice for analyzing single-cell RNA sequencing (scRNA-seq) data[3]. Researchers reduce the gene-expression space (often thousands of dimensions) to two or three dimensions using MDS or t-SNE before inferring cell lineages and cluster identities. The implicit assumption is that the dimensionality reduction step faithfully preserves cluster separability—an assumption our study directly examines under controlled simulation conditions.

In political science, Diaconis et al.[4] applied MDS to the 2005 U.S. House of Representatives roll call votes, producing a 3-D mapping that reveals voting patterns. Their work illustrates MDS as an exploratory visualization tool, but does not evaluate how subsequent clustering is affected by the projection step, nor how this changes as the original dimension increases. Our study fills this gap by measuring the effect of MDS and t-SNE on clustering accuracy across a range of dimensions.

Concept mapping, as used in policy evaluation[5], applies MDS to "pile sorting" data to identify thematic clusters in survey responses. Again, the quality of the final clustering depends on whether the low-dimensional embedding preserves the true groupings—a question we address empirically for multiple data configurations and cluster shapes.

These applications collectively demonstrate that the pipeline of dimensionality reduction followed by clustering is widely

practiced yet its reliability is rarely evaluated in a controlled manner. This is the precise gap our simulation study addresses.

3. METHODS AND PROCEDURES

Cluster analysis is the task of grouping a set of objects so that objects in the same group are more similar to each other than to those in other groups. These groups are called *clusters*. The effectiveness of clustering algorithms is highly dependent on the type of data used.

In this section, we describe three approaches to clustering. The first is called K-means, the second is a family of clustering algorithms called hierarchical, and the third approach is called topological data analysis (TDA). Regardless of the approach, in this paper we will assume that the data consists of N points in an n-dimensional Euclidean space, and the distance between two points, $x, y \in \mathbb{R}^n$ is computed in the usual way, by summing the squared differences in all the coordinates:

$$d(x, y) = \|x - y\| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}\,.$$

In the following, when we refer to distance, we mean this Euclidean norm. However, other distance metrics may be used for clustering.

We now elaborate on some well-known clustering methods, all of which require the number of clusters to be specified. The following clustering algorithms are included in scikit-learn[6], a Python machine-learning package. We selected K-means and hierarchical clustering (with all four linkage criteria available in scikit-learn) because they are among the most widely used clustering algorithms in practice, and because they span a broad spectrum of geometric assumptions—from the spherical cluster assumption of K-means to the flexible, shape-sensitive behavior of single linkage. We include TDA (specifically, the ToMATo algorithm) as a representative topology-based approach that is theoretically capable of capturing non-convex cluster shapes without relying on distance-to-centroid measures. This selection directly reflects the research questions posed above, enabling a direct comparison of centroid-based, linkage-based, and topology-based clustering strategies.

3.1. *K-Means*

A popular algorithm for clustering is K-means. It has been used across a wide range of application areas in many different disciplines. The K-means algorithm works by assigning data points to K clusters, each represented by a centroid, usually the mean of the points in that cluster.

The algorithm follows these steps:

1. Initialization: Select K initial centroids from the dataset.
2. Assignment: Assign each data point to the nearest centroid.
3. Update: Compute new centroids by averaging the points in each cluster.
4. Iteration: Repeat the assignment and update steps until the centroids stabilize or the change is below a defined threshold.

The K-means algorithm aims to choose centroids that minimize a criterion known as the *inertia* or within-cluster sum of squares. That is, if $x_i \in C_j$, the set of all data points assigned to the j-th cluster, and μ_j is the centroid of the j-th cluster, then the K-means algorithm aims to find $\mu_1, \ldots, \mu_K$ such that

$$I = \min_{\mu_1,\ldots,\mu_K} \sum_{j=1}^{K} \sum_{i:x_i \in C_j} ||x_i - \mu_j||^2.$$

Inertia measures cluster compactness, with lower values indicating more cohesive clusters.

K-means has a few limitations. First, it performs poorly in the presence of elongated clusters and manifolds with irregular shapes due to its reliance on the Euclidean distance. Another limitation of K-means is that different random initializations may give different results. A third limitation is that, if the data we use for clustering is given in a high-dimensional space, $\mathbb{R}^n$, but the true clusters are defined in terms of just m features, where $m < n$, then each additional feature (coordinate) adds some noise which inflates the distances, and this has a negative impact on clustering performance. (This applies to other methods, and not just K-means). Suppose that the noise in each coordinate that has no role in defining the true clusters has a standard normal distribution, $N(0, 1)$. The sum of squares of such superfluous coordinates has a chi-square distribution with $n - m$ degrees of freedom and, importantly, a mean of $n - m$. Thus, when n is much greater than m, the distances are dominated by noise rather than by the actual cluster configuration. This makes distance calculations less meaningful. It is often recommended to run a dimensionality reduction algorithm prior to K-means clustering to alleviate this problem and speed up computations.

3.2. *Hierarchical Clustering*

Hierarchical clustering is a method in which clusters are formed sequentially – either *top-down* (divisive clustering), where in the first step all points are put in one cluster, and then in each step one cluster is split into two until each point is its own cluster; or *bottom-up* (agglomerative clustering), where each observation starts in its own cluster, and in each step two clusters are merged together until all the points are in one cluster.

The hierarchy resulting from this process is represented as a tree, or dendrogram. Hierarchical clustering is deterministic and will give the same result each time.

We use agglomerative clustering in this paper. With this method, joining the next two closest clusters in each step depends on how one defines 'closest', because, even when one decides the distance metric between points (the Euclidean, in our case), there is more than one way to define the distance between *sets*. The literature also uses the term *linkage criteria* to refer to the chosen definition of distance between sets, since this choice determines which two clusters will be linked (merged) in the next step.

Scikit-learn's AgglomerativeClustering object offers four possible linkage criteria, which we describe now. We use the notation A and B to define sets (clusters), and i and j to denote points in these sets.

Single linkage minimizes the distance between the closest pair of observations from different clusters. In other words, the distance between two clusters is defined as the smallest distance between *any* pair of points from disjoint clusters. Using mathematical notation, the single linkage distance between sets is defined as:

$$D(A, B) = \min_{i \in A, j \in B} d(i, j)\,,$$

where $d(i, j)$ is the Euclidean distance between elements i and j in clusters A and B, respectively.

Complete linkage minimizes the maximum distance between pairs of points from different clusters. Complete linkage is thus based on the following definition of distance between sets:

$$D(A, B) = \max_{i \in A, j \in B} d(i, j)\,.$$

Average linkage minimizes the *average* of the distances between all pairs of points from two disjoint clusters.

$$D(A,B) = \frac{1}{|A| \cdot |B|} \sum_{i \in A} \sum_{j \in B} d(i,j)$$

where $|A|$ and $|B|$ are the number of elements in clusters A and B, respectively.

Ward linkage minimizes the sum of squared differences within all clusters[7]. It minimizes variance similar to the K-means objective function. The distance between sets for Ward linkage is expressed as:

$$D(A,B) = \sqrt{\frac{|A| \cdot |B|}{|A| + |B|}} \cdot \|\bar{a} - \bar{b}\|,$$

where $|A|$ and $|B|$ are the number of elements in clusters A and B, respectively, $\bar{a}$ and $\bar{b}$ are the centroids (mean points) of clusters A and B respectively, and $\|\bar{a} - \bar{b}\|$ is the Euclidean distance between the centroids of clusters A and B.

3.3. *Topological Data Analysis*

Topological data analysis (TDA) is a technique for identifying structural patterns in complex datasets using persistent homology, a method that captures the underlying shape of data across multiple dimensions. TDA can quantify shape and structure in data, making it a topology-based method which is in theory resistant to noise, scalable to larger data sets, and capable of capturing complex geometric shapes. We find, however, that it is not necessarily so, and TDA may be severely affected by the dimensionality of the data and the noise level.

We will use the Topological Mode Analysis Tool (ToMATo) Algorithm for TDA clustering[8]. ToMATo operates in three main steps:

1. Density Estimation and Neighborhood Graph Construction: The algorithm begins by estimating the density of points in the dataset. It then constructs a neighborhood graph, where each data point is connected to its nearest neighbors based on a chosen similarity measure.
2. Mode-Seeking and Initial Cluster Formation: ToMATo assigns each point to a local density maximum by following the paths of connected points in the neighborhood graph. This process groups points into initial clusters, each centered around a mode, a local maximum of the estimated data density that serves as the center of an initial cluster, representing a dense and potentially meaningful region of the data. A persistence diagram is created to measure the persistence of each mode before it gets merged with another. The birth time of a mode is when it first appears as an independent cluster. The death time is when it merges into a more dominant cluster. The difference (persistence = death - birth) quantifies the cluster's significance.
3. Hierarchical Merging of Clusters: Clusters are merged based on their persistence. Modes with low persistence, short-lived peaks, are considered less important and merged first. High-persistence modes, stable clusters, remain in the final result. Since merging follows a hierarchical structure, each new cluster is formed by combining two existing ones, producing a nested sequence of cluster formations.

3.4. *Evaluating Clustering Performance*

When evaluating the performance of a clustering algorithm, one must first determine which pairs of points were clustered correctly, and which were not. Since labels are assigned to clusters arbitrarily by a clustering algorithm, these labels may not match the true labels directly. Instead of relying on the arbitrary labels assigned by an algorithm, pairwise relationships between points are used. In other words, if two points that are assigned to the same cluster (regardless of the label) are also in the same cluster in the true clustering configuration, then this pair is considered correctly assigned. This leads to four possible outcomes for each pair, and we count the number of occurrences of each outcome over all the pairs:

- a is the number of pairs of points that are in the same cluster in both the predicted clusters and true ones (true positives).
- b is the number of pairs that are in the same cluster in the predicted configuration but not in the true ones (false positives).
- c is the number of pairs that are in the same cluster in the true configuration but not in the predicted one (false negatives).
- d is the number of pairs that are not in the same cluster, neither in the true configuration nor in the predicted one (true negatives).

One way to evaluate clustering performance is the *accuracy*, defined as the number of correct predictions over total predictions, or $(a+d)/(a+b+c+d)$. There are two problems with accuracy. One is that it may be inflated when the data is not balanced. That is, when some clusters are much larger than others. In such unbalanced cases, a simple clustering rule that puts all points in one cluster will exhibit high accuracy. The other problem is that when the number of true negatives (d) is large, it dominates both the numerator and the denominator and yields an accuracy value close to 1. In clustering tasks, it is often the case that d is indeed large, since most pairs are not in the same cluster.

Thus, in the following we use the Jaccard index, which is defined as:

$$J = \frac{a}{a+b+c}\,.$$

Note that it is similar to the accuracy measure except that the true negatives are dropped from the calculation. The Jaccard Index[9] considers all pairs of elements and determines whether they are clustered together in both clusterings (intersection, the numerator) or in at least one of the clusterings (union, the denominator), which is why it is also referred to as the intersection over union (IoU) in the literature. The Jaccard index ranges from 0 to 1, where 0 means no agreement (completely different clusterings), and 1 indicates perfect agreement.

3.5. *Dimensionality Reduction Methods*

As mentioned above, clustering is especially challenging when the data is embedded in a high-dimensional space, but the true clusters are defined by lower-dimensional shapes. Thus, before clustering is performed, the first step is to try and reduce the dimensionality of the data. Because clustering is especially useful when the clusters can be visualized, it is often the case that the data is projected into a 2-D Euclidean space.

Although dimensionality reduction is an essential first step when performing clustering, it is important to note that it results in some loss of information. To illustrate, consider projecting the surface of the earth onto a 2-D map. In this process, distances are inevitably distorted. For example, on a standard world map Alaska and Russia appear on opposite sides, suggesting they are far apart, when in fact they are relatively close. This shows that when high-dimensional data are reduced to lower dimensions, distances between points can be distorted. Since there is already distortion from three dimensions to two, it stands to reason that there would be even greater distortion when reducing from much higher dimensions. Thus, measuring the distortion caused by the projection step is essential to evaluate the degree of misplacement of data points, and its impact on the ultimate task of clustering.

In this paper we will use two dimensionality reduction methods: Multidimensional scaling (MDS) and t-Distributed Stochastic Neighbor Embedding (t-SNE).

We begin with a brief description of MDS. Given a dissimilarity or distance matrix D representing the differences between all pairs of points, such that D_{ij} denotes the distance between data points i and j in the high-dimensional space,

multidimensional scaling (MDS) seeks a configuration of points in a low-dimensional space such that the corresponding Euclidean distances d_{ij} are as close as possible to D_{ij}, across all pairs i, j. Metric MDS achieves this by minimizing a loss function known as stress. We use Kruskal's[1] Stress-1, defined as

$$\text{Stress-1} = \sqrt{\frac{\sum_{i<j} \left(d_{ij} - D_{ij}\right)^2}{\sum_{i<j} D_{ij}^2}}.$$

The summation is taken over all unordered pairs $i < j$. The Stress-1 criterion measures the normalized discrepancy between interpoint distances in the original and embedded spaces, with smaller stress values indicating a more faithful low-dimensional representation of the original distance structure.

Scikit-learn provides an MDS algorithm that is implemented in Python.

Next, we briefly describe t-SNE[2]. t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear dimensionality reduction method widely used for visualizing high-dimensional data in two or three dimensions. This method is based on converting pairwise similarities between data points into probability distributions and minimizing the divergence between these distributions in the high- and low-dimensional spaces.

In the high-dimensional space, similarities between points are modeled using conditional probabilities that reflect the likelihood of one point being a neighbor of another under a Gaussian distribution centered at the point of interest. In the low-dimensional embedding, similarities are modeled using a Student's t-distribution with one degree of freedom, which has heavier tails than a Gaussian and thus better preserves local structures while avoiding the "crowding problem." The optimization objective is to minimize the Kullback-Leibler (KL) divergence between the two distributions. This minimization problem is typically solved via gradient descent.

The scikit-learn library provides an efficient implementation of t-SNE through the class sklearn.manifold.TSNE. This implementation supports multiple optimization algorithms, enabling scalability to moderately large datasets.

We restrict our study to MDS and t-SNE because they are the dimensionality reduction methods most commonly encountered in the application domains described in Section 2, and because they represent two fundamentally different methodological families: metric (distance-preserving) methods exemplified by MDS, and distribution-based (neighbor-embedding) methods exemplified by t-SNE. This contrast is sufficient to illustrate the full range of behaviors relevant to our research questions. Other methods, most notably Uniform Manifold Approximation and Projection (UMAP)[10], have gained increasing popularity and typically produce high-quality embeddings; evaluating UMAP alongside MDS and t-SNE is a natural direction for future work.

4. RESULTS

We present simulations that were performed in order to assess whether or to what extent clustering algorithms are affected by three factors:

1. Noise level – the dispersion of points belonging to a distinct cluster. The larger the dispersion, the more likely it is that points will be found in the neighborhood, or even within, another cluster.
2. The dimensionality – as stated above, if the true dimension of the clusters, m, is smaller than the dimension of the data, n, then the superfluous dimensions contribute to noise and when $n - m$ is large the noise overwhelms the signal, making clustering impossible.
3. The shapes of the clusters – in particular, we use both convex and non-convex examples. Several clustering algorithms rely on convexity (or even on sphericity) and are known to perform poorly when some clusters are non-convex.

We use three configurations in our simulations. In the first scenario we generate five clusters as 'Gaussian clouds' (or balls) in $\mathbb{R}^4$. In the second scenario, we use hyperspheres, which are similar to the clouds, except that the data is concentrated close to the surface of the five balls. The third configuration consists of two horseshoes in $\mathbb{R}^2$ such that they do not overlap, but their convex hulls do.

Each scenario is repeated with varying degrees of noise. Furthermore, the data from each scenario is embedded in a higher dimensional space $\mathbb{R}^n$, with $n = 4, 8, 16, 32, 64, 128, 256$.

We compare clustering results in these configurations (in terms of the IOU, or the Jaccard index) before and after MDS and t-SNE are applied.

4.1. *Gaussian Clouds*

We simulate five Gaussian clouds, each with 600 points, for a total of 3,000 points. The centers of the clouds span a four-dimensional space.

The centers are embedded in an n-dimensional space, and the points are sampled from an n-dimensional multivariate normal distribution around each center, with the cluster standard deviation set to $\sqrt{n}/4$. A 2-D depiction of the data after applying MDS is shown in Figure 1.

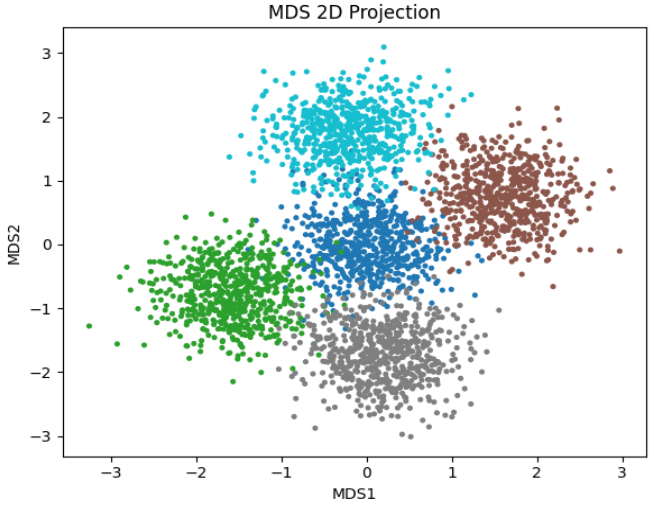

Figure 1. Visualization of the simulated dataset consisting of five Gaussian clusters, each with 600 points (3,000 points total). 2-D embedding obtained using MDS.

For each iteration in our simulation, we randomly draw 300 points from each cluster (out of its 600) to form a sample of 1,500 points. Here, and in the following scenarios, when using the cluster.KMeans function, we set the parameter max_iter = 1 to allow a valid comparison with the other clustering methods.

The results of six clustering methods are shown in Figure 2, for different values of n (the dimension of the space). The left figure shows boxplots of the Jaccard index obtained from 100 simulations when the six clustering methods were applied to the original data in $\mathbb{R}^n$. The other two figures show the results when clustering is applied after reducing the dimension to $\mathbb{R}^2$ with MDS (middle figure) and t-SNE (right figure). Table 1 shows only the median Jaccard index for the six methods for $n = 4$ and $n = 64$.

Figure 2 shows that all of the clustering methods decreased in performance as the dimension increased. This holds true both with the original (unscaled) data, and after dimensionality reduction. The rate at which the performance deteriorates is smallest when using t-SNE to reduce the dimension. The plots show that with this configuration TDA had a

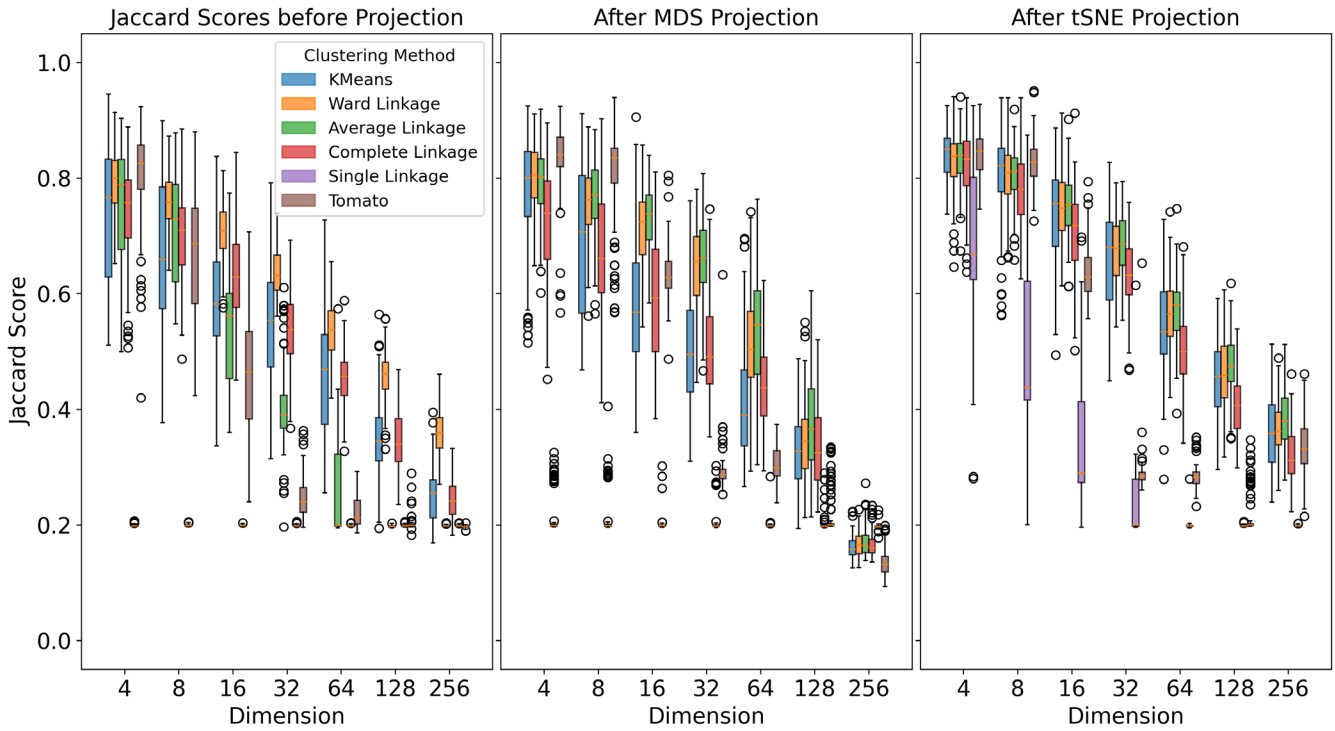

Figure 2. Comparison of six clustering methods applied to the simulated clouds dataset across varying data dimensions n (from 4 to 256). The left panel shows boxplots of the Jaccard index from 100 simulations on the original $\mathbb{R}^n$ data. The middle and right panels show performance after reducing the data to $\mathbb{R}^2$ using MDS and t-SNE, respectively. For all methods, the KMeans algorithm was limited to a single iteration per sample to ensure a fair comparison.

Clouds	Dim 4			Dim 64		
	In $\mathbb{R}^n$	After MDS	After t-SNE	In $\mathbb{R}^n$	After MDS	After t-SNE
K-means	0.766	0.801	**0.850***	0.469	0.390	0.534
Ward linkage	0.800	0.805	0.838	0.537*	0.503	0.565
Average linkage	0.789	0.801	0.839	0.199	0.546*	**0.580***
Complete linkage	0.757	0.739	0.833	0.457	0.437	0.499
Single linkage	0.198_*	0.199_*	0.668_*	0.198_*	0.198_*	0.198_*
TDA	0.826*	0.840*	0.847	0.213	0.298	0.282

Table 1. Median Clustering Performance – Gaussian clouds. A superscript/subscript asterisk denotes the best/worst performance in a column, respectively. Bold-face denotes the best performance overall, for a fixed n.

slightly better performance than the other methods when $n = 4$, and also when $n = 8$ but only after dimensionality reduction. However, in higher dimensions the performance of TDA (with and without dimensionality reduction) quickly deteriorates.

The worst performance among all clustering methods in this scenario was that of single linkage. Although with $n = 4$ t-SNE greatly improved the median Jaccard index (from 0.198 to 0.668, see Table 1), it is still far below all other methods, and in higher dimension it performed very poorly.

Without dimensionality reduction, Ward linkage had the best performance in higher dimensions. With MDS as the dimensionality reduction method, average linkage was as good as Ward's. K-means and complete linkage were not as good, especially as n increased.

In dimension 4, MDS offered a slight improvement over clustering without scaling, except for complete linkage, where the median was decreased a little, and single linkage which performed poorly in this configuration. Running t-SNE prior to clustering improved the median Jaccard index for all methods. In dimension 64, running t-SNE prior to clus-

tering improved the median Jaccard index for all methods except for single linkage, and the most dramatic improvement was for average linkage. MDS improved the average linkage and TDA performance, but for the other four methods the median score decreased.

4.2. *Hyperspheres*

The second scenario is similar to the previous one, except that the points in each cluster are concentrated near the surface, as opposed to being scattered uniformly in the ball. Note that unlike the previous example, hyperspheres are non-convex sets. This is easy to see from the triangle inequality. As before, we simulate five spheres, each with 600 points, for a total of 3,000 points. Points are drawn uniformly on a sphere with a fixed radius of $\sqrt{n}/2$.

In our simulation, we increase the dimension in which the clusters are embedded from 4 to 256. We run the simulation 100 times, each time drawing a sample size of 300 points out of the 600 from each cluster. Figure 3 shows the boxplot of the Jaccard index for the six clustering methods, over a range of dimensions. The left figure shows the results without a dimensionality reduction step. The middle/right figures show the results when clustering is performed after MDS/t-SNE, respectively. Table 2 shows the medians for the cases of $n = 4$ and $n = 64$.

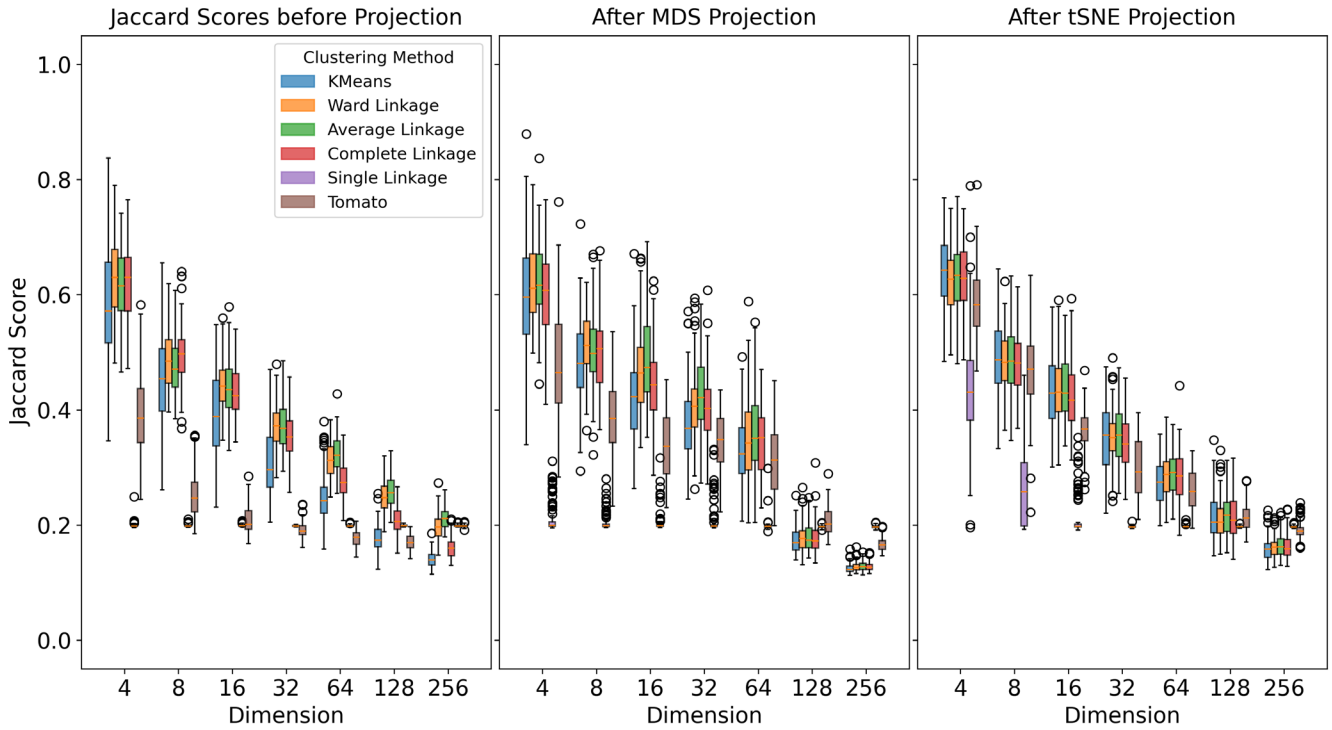

Figure 3. Comparison of clustering methods applied to the simulated hyperspheres dataset for varying dimensions n (from 4 to 256). The left panel shows boxplots of the Jaccard index from 100 simulations on the original $\mathbb{R}^n$ data. The middle and right panels show performance after reducing the data to $\mathbb{R}^2$ using MDS and t-SNE, respectively.

Hyperspheres	Dim 4			Dim 64		
	In $\mathbb{R}^n$	After MDS	After t-SNE	In $\mathbb{R}^n$	After MDS	After t-SNE
K-means	0.572	0.596	**0.642***	0.242	0.323	0.274
Ward linkage	0.630*	0.611	0.627	0.312	0.342	0.287
Average linkage	0.615	0.616*	0.633	0.321*	0.350	0.291*
Complete linkage	0.630*	0.608	0.628	0.273	**0.351***	0.285
Single linkage	0.198$_*$	0.199$_*$	0.431$_*$	0.198	0.197$_*$	0.197$_*$
TDA	0.385	0.464	0.582	0.179$_*$	0.313	0.258

Table 2. Median Clustering Performance – hyperspheres. A superscript/subscript asterisk denotes the best/worst performance in a column, respectively. Bold-face denotes the best performance overall, for a fixed n.

In some regards the results are similar to the previous scenario: all of the clustering methods decreased in performance

as the dimension increased, and the worst performer was single linkage. Also, the performance of TDA deteriorates most rapidly, followed by K-means, which is also negatively affected by the increase in n.

We see that although this scenario is similar to the previous one, the fact that the clusters are not uniformly distributed in a ball, but are concentrated near the surface, presents a challenge to all clustering methods. The overall scores are lower than in the previous case, and deteriorate rather quickly. In this scenario, MDS was somewhat effective in improving the clustering in high dimension, while t-SNE resulted in lower scores when compared with no dimensionality reduction (except for a slight improvement for complete linkage; see Table 2).

In dimension 4, Ward linkage had the highest median Jaccard index before dimensional scaling, followed closely by complete linkage. Average linkage had the highest Jaccard index after MDS. K-means had the highest median Jaccard index after t-SNE. In dimension 64, average linkage had the highest median Jaccard index before dimensional scaling, and also after t-SNE. Complete linkage was best after MDS (0.351), and overall in this scenario with $n = 64$.

4.3. *Two Horseshoes*

For another non-convex scenario, we simulate two horseshoes, each with 2,000 points, for a total of 4,000 points. Figure 4 shows the data in 2-D, after performing MDS.

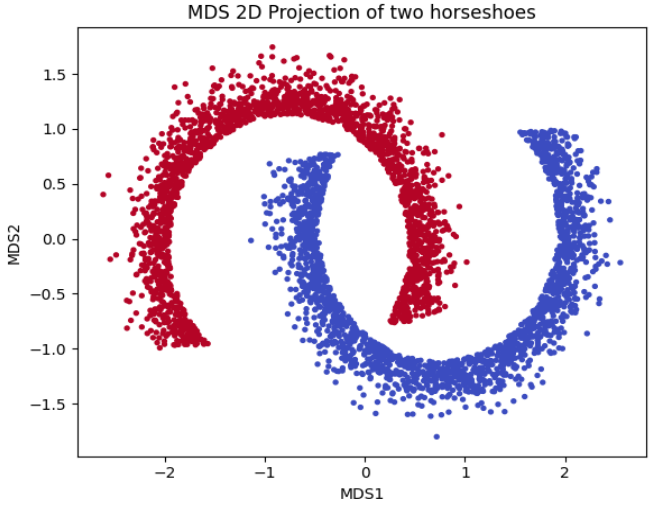

Figure 4. Visualization of the simulated dataset consisting of two non-convex clusters, each with 2,000 points (4,000 points total). 2-D embedding obtained using MDS.

In our simulations, we increase the dimension from 4 to 64. In each of 100 iterations we draw a random sample of size of 200 points from each cluster. The medians from each method for $n = 4$ and $n = 64$ are shown in Table 3, and the boxplots from 100 iterations with each n are shown in Figure 5.

Two Horseshoes	Dim 4			Dim 64		
	In $\mathbb{R}^n$	After MDS	After t-SNE	In $\mathbb{R}^n$	After MDS	After t-SNE
K-means	0.440_*	0.447_*	0.645	0.441_*	0.438_*	0.643
Ward linkage	0.537	0.536	0.558	0.525	0.546^*	0.576
Average linkage	0.524	0.524	0.525	0.528	0.526	0.539
Complete linkage	0.538^*	0.550^*	0.581	0.543^*	0.551	0.586
Single linkage	0.498	0.498	$\mathbf{1.000}^*$	0.498	0.500	$\mathbf{1.000}^*$
TDA	0.476	0.486	0.500	0.461	0.487	0.483

Table 3. Median Clustering Performance – two horseshoes. A superscript/subscript asterisk denotes the best/worst performance in a column, respectively. Bold-face denotes the best performance overall, for a fixed n.

The results in this scenario are strikingly different from the previous examples. First, regardless of n, all methods struggle to find the correct clusters, as indicated by a Jaccard index which is approximately 0.5 for most methods. The one

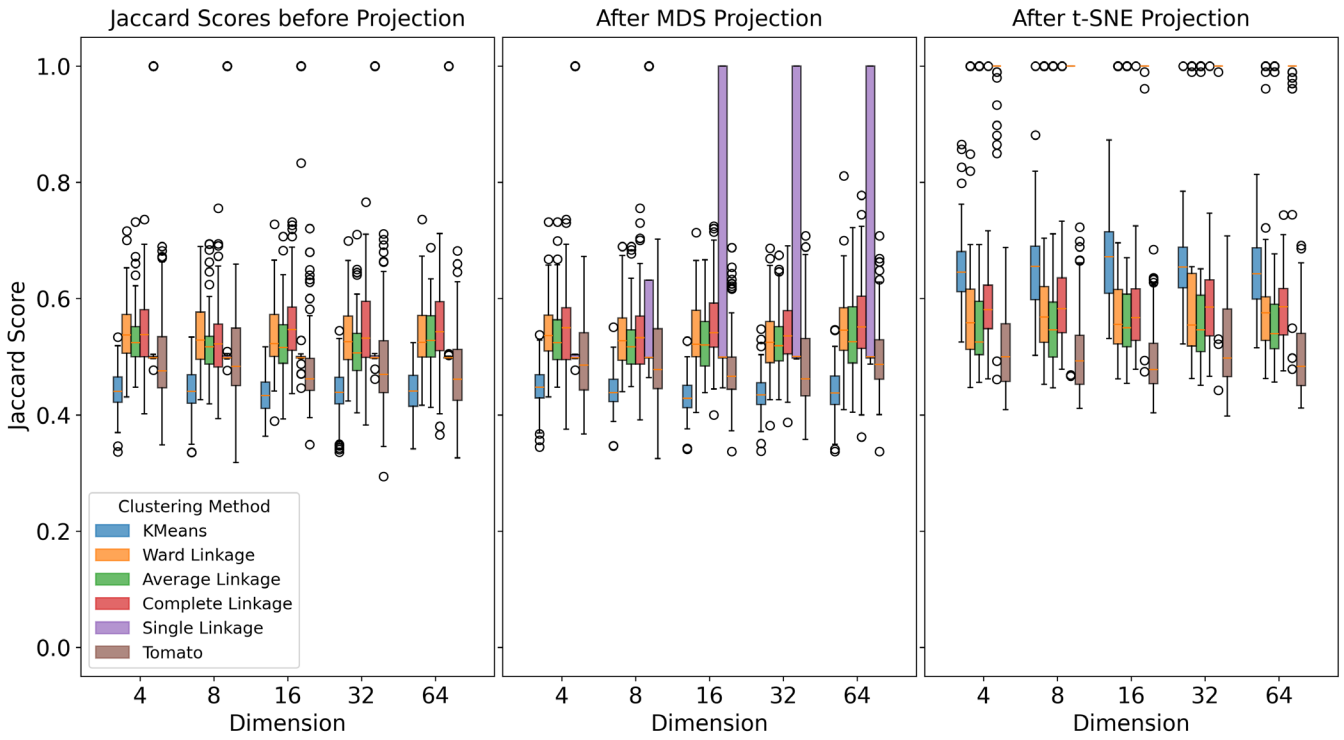

Figure 5. Comparison of clustering methods applied to the two-horseshoes dataset for varying dimensions n (from 4 to 64). The left panel shows boxplots of the Jaccard index from 100 simulations on the original $\mathbb{R}^n$ data. The middle and right panels show performance after reducing the data to $\mathbb{R}^2$ using MDS and t-SNE, respectively.

method that appears to perform better in some of the simulations is the single linkage, which was the worst method in the previous examples. However, to achieve this improved performance one must start with t-SNE. In this scenario K-means has the worst performance when no dimensionality reduction is performed, or if MDS is performed. On the other hand, when t-SNE is performed, K-means performs better (increasing from 0.44 to 0.64), but not nearly as good as single linkage with t-SNE.

5. DISCUSSION

The purpose of this study was to evaluate how different clustering algorithms perform across a variety of data structures and to determine whether or to what extent clustering is affected by the dimension of the space containing the clusters. We also investigated in what way dimensionality reduction methods influence the accuracy of these clustering outcomes. We used the Jaccard index (also known as intersection over union, or IoU) to assess the quality of each clustering algorithm.

First, we observe that the simulations demonstrate clustering performance depends strongly on the geometric structure of the data and the presence of noise, even before dimensionality reduction is applied. We used three configurations, and found that when the clusters are convex, the performance was better, overall, when compared with non-convex shapes. Clustering in the Gaussian clouds scenario (4.1) had better IOUs than the similar, but non-convex setting of hyperspheres (4.2), and both had much better results than the third scenario (4.3), in which not only the shapes are non-convex, but their convex hulls intersect.

Second, we find that no single clustering method consistently performs well across all data configurations or outperforms the other methods. For example, K-means and Ward linkage tended to perform well in the first two scenarios when the underlying clusters were approximately spherical. However, their performance was poor in the third scenario. In contrast, single linkage showed sensitivity to noise and performed the worst among all methods for both the clouds

and spheres scenarios, yet had the best performance for the non-convex horseshoe data after t-SNE. Topological data analysis (TDA), using the ToMATo algorithm, did not have the best clustering performance in any of the three scenarios we have tested.

Third, we find that the higher the dimension in which the clusters are embedded, the harder the clustering task, regardless of the method. In some cases, performing a dimensionality reduction before clustering does improve the result. However, this is not always the case (e.g., the two horseshoes) and the positive effect of the dimensionality reduction step depends on the shapes. In general, between the two methods that we used, namely, MDS and t-SNE, the latter is preferred, but that is not always the case. For example, with the hyperspheres in $\mathbb{R}^{64}$, complete linkage after MDS yielded the best result (albeit, a rather poor one), while in the clouds scenario, the better results were achieved if t-SNE was used. See Figures 2 and 3.

6. CONCLUSIONS

The results highlighted here have important practical consequences. It is important to keep in mind that with real data we do not know whether the clusters are convex or not, or whether their convex hulls intersect, nor do we know the actual intrinsic dimension of the data, nor the magnitude or distribution of noise. As we have demonstrated, there is no way to choose the 'best' clustering algorithm, as each one depends on the (unknown) shapes in different ways. The effectiveness of each method depends fundamentally on the geometry and noise characteristics of the data. Methods like K-means, Ward linkage, average linkage, and complete linkage have better clustering performance in settings with compact, spherical clusters, while single linkage offers advantages for clustering well-separated, non-convex data.

Furthermore, if low-dimensional clusters are embedded in a high-dimensional space, the contribution of superfluous coordinates to the distance between points is substantial and obscures meaningful geometric relationships as the dimension of the space becomes much larger than the intrinsic dimension of the clusters. Dimensionality reduction techniques may alleviate this problem in some cases, but not always. In some cases, clustering in the original high-dimensional space provided more accurate results than after projection to lower dimensions. Our findings suggest that there is no clear winner between the dimensionality reduction methods we have used.

Finally, we point out that in our analysis we provided the correct number of clusters, but in general this is another unknown and has to be estimated by the user. The general approach is to vary k and choose the one that gives the best information criterion (e.g., AIC, BIC). This is beyond the scope of this study, but it is yet another challenge facing practitioners who perform clustering.

We recommend that analysts perform cross-validation; splitting the data into training and testing data, and then evaluate multiple clustering methods and dimensionality reduction methods for any given dataset, rather than relying on a preselected clustering method or on MDS or t-SNE plots.

Future work includes determining the intrinsic dimension of datasets for better clustering performance (and possibly a better way to select the correct number of clusters) and leveraging AI to improve current clustering methods. Such improvements have the potential to improve deep-learning-based segmentation methods such as nnU-Net[11] to detect tumors in medical imaging.

ACKNOWLEDGEMENTS

The authors thank the Holster Scholar Grant for supporting this project.

REFERENCES

1. Kruskal, Joseph B. (1964) Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis, *Psychometrika* 29.1 : 1-27. *https://doi.org/10.1007/BF02289565*
2. Maaten, L.V., and Hinton, G.E. (2008) Visualizing Data using t-SNE, *Journal of Machine Learning Research* 9, 2579-

2605.
3. Van den Berge, K., Roux de Bézieux, H., and Street, K. (2020) Trajectory-based differential expression analysis for single-cell sequencing data, *Nature Communications* 11, 1201. *https://doi.org/10.1038/s41467-020-14766-3*
4. Diaconis, P., Goel, S., and Holmes, S. (2008) Horseshoes in multidimensional scaling and local kernel methods, *The Annals of Applied Statistics* 2(3), 777–807. *https://doi.org/10.1214/08-AOAS165*
5. Trochim, W. M., and McLinden, D. (2017) Introduction to a Special Issue on Concept Mapping, *Evaluation and Program Planning* 60, 166–175. *https://doi.org/10.1016/j.evalprogplan.2016.10.006*
6. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011) Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research* 12, 2825–2830. *http://jmlr.org/papers/v12/pedregosa11a.html*
7. Ward, J. H. (1963). Hierarchical Grouping to Optimize an Objective Function, *Journal of the American Statistical Association* 58(301), 236–244. *https://doi.org/10.1080/01621459.1963.10500845*
8. Chazal, F., Guibas, L. J., Oudot, S. Y., and Skraba, P. (2013) Persistence-Based Clustering in Riemannian Manifolds, *Journal of the ACM* 60(6), Article 41. *https://doi.org/10.1145/2535927*
9. Jaccard, P. (1901) Étude comparative de la distribution florale dans une portion des Alpes et des Jura, *Bulletin de la Société Vaudoise des Sciences Naturelles* 37, 547–579.
10. McInnes, L., Healy, J., and Melville, J. (2018) UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, *arXiv preprint* arXiv:1802.03426. *https://arxiv.org/abs/1802.03426*
11. Isensee, F., Jaeger, P.F., Kohl, S.A.A. et al. (2021) nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation, *Nat Methods* 18, 203–211. *https://doi.org/10.1038/s41592-020-01008-z*
12. Dexter, E., Rollwagen-Bollens, G., and Bollens, S. M. (2018) The trouble with stress: A flexible method for the evaluation of nonmetric multidimensional scaling, *Limnology and Oceanography: Methods* 16(7), 434–443. *https://doi.org/10.1002/lom3.10257*
13. Cohen-Steiner, D., Edelsbrunner, H., and Harer, J. (2007) Stability of Persistence Diagrams, *Discrete and Computational Geometry* 37, 103–120. *https://doi.org/10.1007/s00454-006-1276-5*

ABOUT THE STUDENT AUTHOR

Lucy Liu is a junior majoring in both Statistics and Applied Mathematics, and she plans to graduate in the Spring of 2027.

PRESS SUMMARY

Clustering is a widely used technique for finding patterns or groups within data. This simulation study investigates the performance of several clustering methods when the clusters are embedded in high dimensional spaces, and whether clustering performance is improved if one first performs dimensionality reduction. By testing various data scenarios and dimensions, and applying several clustering methods, we observed that (i) dimensionality reduction methods such as MDS and t-SNE sometimes weaken clustering accuracy, (ii) higher dimensions of data reduce clustering performance, and (iii) no single clustering method works best in all settings.