Applying Multidimensional Scaling to 3D Genomics

A project report presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Amrutha Varshini Alur

May 2024

This project report titled

Applying Multidimensional Scaling to 3D Genomics

by

Amrutha Varshini Alur

has been approved for

the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology by

Lonnie Welch

Professor

Patrick Fox

Dean, Russ College of Engineering and Technology

ABSTRACT

AMRUTHA VARSHINI ALUR., M.S., May 2024, Computer Science

Applying Multidimensional Scaling to 3D Genomics

Director of project report: Lonnie Welch

The study of the 3-dimensional genome and single-cell analyses are important to understand the differences in the structure of chromatin across different cell types. The entire complexity of cellular variations cannot be fully captured by traditional approaches. However, the Genome Architecture Mapping (GAM) technique provides comprehensive single-cell information, ligation-independence, and multi-way contact detection, which overcomes the limitations of other methods that study the 3D genome. The GAM approach generates a tremendous amount of data that is difficult to understand not just for humans but also for machines. This issue is called the curse of dimensionality. To overcome this problem, we employ a dimensionality reduction technique called Multidimensional Scaling (MDS) to reduce high dimensional data to a manageable set of dimensions. Our study evaluates how well this technique performs while identifying features in the GAM segregation table that allows for data clustering, compared to the existing technique called UMAP (Uniform Manifold Approximation and Projection). We utilized evaluation criteria such as the Calinski-Harabasz score and cluster purity to advance our understanding of cellular heterogeneity, and genetic organization and to determine which technique performs better. The results demonstrated that MDS only achieves a cluster purity of 50% on average, indicating poor performance in feature selection for clustering. On the other hand, UMAP achieved significantly higher cluster purity, reaching 93%.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Professor Lonnie Welch, my research advisor, for his patience and encouragement in this research work. I thank my Masters' committee members Professor David Juedes, and Professor Faiz Rahman for their help on this project. I would also like to thank my colleagues Trenton Davis and Srinath Palleboina for their help during my research. I also want to thank my parents for supporting me in my work towards a Masters' Degree.

Page

Table of Contents

1. INTRODUCTION

The field of biology is broad. Problems range from simply understanding the mechanisms that drive certain feature development, to complex diseases that consistently evolve like cancer and the recent pandemic's COVID-19 virus. Understanding the driving factors for these simple and more complex problems is one of the main ideas in the field. Many scientists believe that the 3-dimensional organization of chromatin is something that drives many diseases and differences between cells that are of different types or sub types [1]. The simplest exploration of this topic would be to classify cells according to their types. There are many different techniques that have explored this topic. One of the main issues that many computational biologists currently face is the issue with high- dimensional datasets. Measuring distances becomes less effective in spaces with many dimensions. Techniques like clustering that rely on distance may become less successful as a result. This problem has been deemed the curse of dimensionality [2]. Techniques such as genome architecture mapping (GAM) generate data that can be overwhelming for computers. In this project, I will explore a dimensionality reduction technique and aim to simplify the GAM segregation table data while preserving key details, allowing us to see the larger context and identify distinct cell types.

Within this project I have investigated how well the multidimensional scaling (MDS) algorithm can reduce the dimensions of genome architecture mapping (GAM) data to identify distinct cell types. This project evaluates whether multidimensional scaling (MDS) effectively preserves key information about different cell types within

high dimensional genome architecture mapping (GAM) data. To address this, we built a pipeline comprising of three key steps; distance matrix calculation, projecting high dimensional data into a lower dimensional space while maintaining important relationships between each cell and perform clustering that will separate a collection of heterogeneous cells into homogeneous groups that correspond to cell type. A fellow PhD student (Trenton Davis) developed this architecture, which I am using to apply the MDS technique to determine whether it is a better alternative solution than the existing solution (Uniform Manifold Approximation and Projection (UMAP)).

We claim that MDS dimensionality reduction technique is ineffective as the results indicate that it does not produce meaningful clusters when applied to GAM data. Considering the importance of dimensionality reduction and correct cell type identification, this project attempts to verify this claim by conducting a thorough evaluation of MDS's ability to preserve essential cellular information.

2. BACKGROUND

Understanding cellular heterogeneity requires single-cell analysis because it enables the study of individual cells and yields accurate genetic and biochemical information that is not achievable from bulk RNA/DNA samples [3]. This method has reshaped molecular biology and has potential to help us comprehend biological complexity on a deeper level [4]. On the other hand, initial approaches for single-cell analyses have had a number of limitations. The inability to fully capture the complexity of heterogeneity in cells was one of the primary challenges. Tens of thousands of cells, or even more, were needed for traditional approaches, which frequently resulted in lack of precision while identifying individual cell variations [3]. This resulted from the fact that these approaches were not able to distinguish between individual cell variations, only capturing the overall intensity of signals from a collection of cells [3]. Handling high- dimensional data created another challenge. Systematic noise, biological system characteristics, data sparsity and complexity are some of the inherent difficulties with single-cell data [4].

Despite these drawbacks, single-cell technologies have experienced impressive expansion and ongoing technological advancement, broadening its uses in scientific and medical research [3][4]**.** However, the community at large needs improved tools that are scalable, reasonably priced, and gentle enough to preserve cell integrity to fulfill the scientific and medical needs of single-cell analysis.

The spatial arrangement of a cell's DNA in the nucleus is referred to as the "3D genome," and it is important for cellular function and gene regulation. It is crucial to study the 3-dimensional genome at a single cell level because it provides insight into how the genome is organized differently in each cell, which affects the expression of genes specific to that cell type [6][8]. The 3D genomic features, like chromatin loops, A/B compartments, and topologically associating domains (TADs), are associated with essential genome functions, including DNA replication and gene transcription [9]. The functional importance of the variations in these 3D genomic structures among individual cells directly affects gene expression specific to those cell types [7][9].

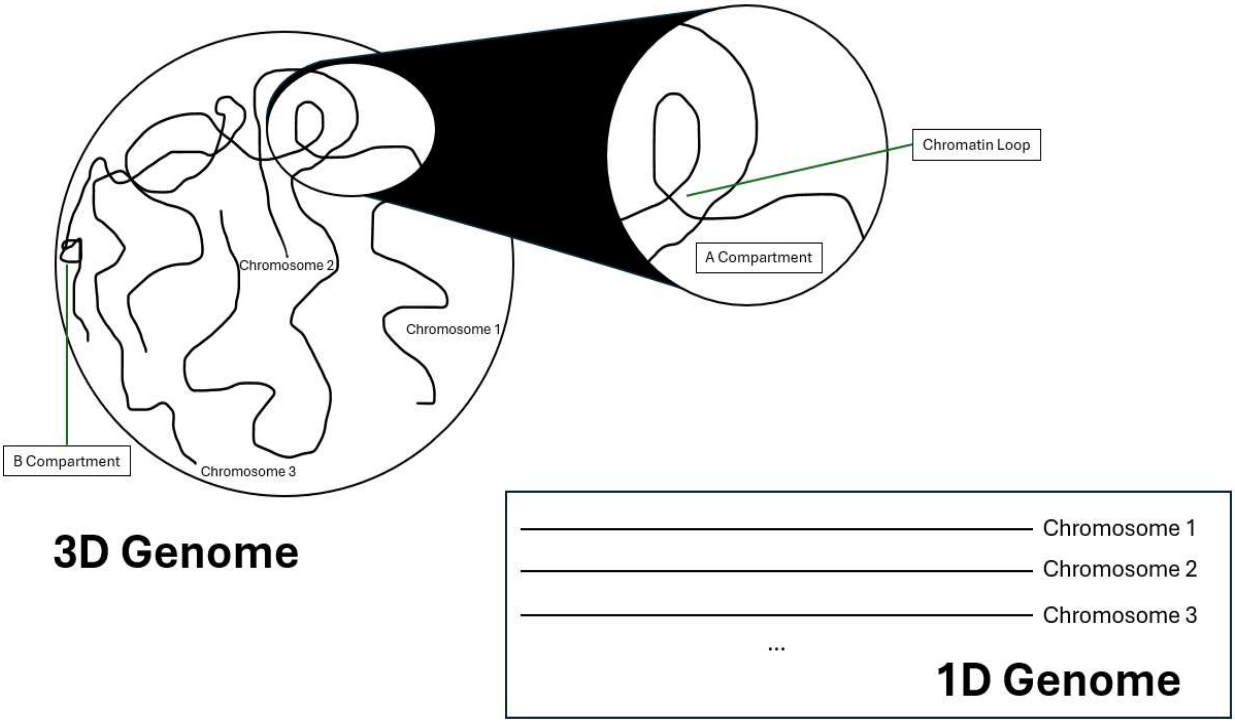


Figure 1. 3D Genome structure.

In summary, single-cell research on the 3D genome is crucial for comprehending how individual cell variations in the genome's spatial organization affect gene expression specific to a given cell type and provide important new insights on cellular heterogeneity and function.

Understanding the spatial structure of DNA in the nucleus and how it is important for the regulation of genes and cellular function has been made possible by the advancement of 3D genome technology. Chromosome Conformation Capture (3C) is the fundamental technique in this field. Its derivatives, such as 4C (Circular Chromosome Conformation Capture), 5C (Chromosome Conformation Capture Carbon Copy), and Hi-C, have gradually improved the scope of 3D genome analysis.

Measuring the interaction between two genomic loci inside the three- dimensional (3D) space of the nucleus was first carried out by the 3C approach. To cross-link associating DNA segments, formaldehyde fixation is used. This is followed by restriction enzyme digestion and ligation under circumstances which favor intramolecular ligation [10][11]. 4C builds on 3C by employing circularization of the ligated and digested DNA to identify every region interacting with a specific locus of interest [10]. By connecting 3C with massively parallel sequencing, 5C improves the process even more and makes it possible to analyze multiple loci interactions at once [10]. Hi-C is a major advancement that makes it possible to analyze chromatin interactions across the entire genome. With high-throughput sequencing, it combines a similar cross-linking, digestion, and ligation method to map interactions throughout the entire genome [10][12].

These methods have limitations, mainly because they rely on enzymatic digestion and ligation procedures to capture interacting DNA segments, despite their revolutionary contributions to the knowledge of the 3D genome. Due to chromatin accessibility and sequence specificity the effectiveness of enzymatic digestion varies throughout the genome, which may result in an inadequate or uneven representation of interactions [10][12]. The precise measurement of interaction frequencies may be impacted by variation in ligation efficiency, which is further influenced by the orientation and spatial proximity of DNA ends [10]. It can be difficult to distinguish biologically significant interactions from noise while handling high-dimensional data, particularly when Hi-C is involved. This is because normalization and interpretation of the data require complex computing tools [10][13].

Genome Architecture Mapping (GAM) is a technique that overcomes several drawbacks in 3C-based approaches, greatly advancing the study of the 3D genome. A more detailed knowledge of chromatin arrangement and how it impacts gene regulation and cellular function is made possible by the invention and use of GAM. Chromatin arrangement is used to understand how different parts of DNA are arranged in three-dimensional space within the nucleus of a cell. The process of GAM includes cutting thin slices of the nucleus, sequencing the DNA, and identifying if each genomic locus is present or absent in a collection of single slices taken from a population of nuclei at random orientations [14].

GAM does not rely on the digestion and ligation process to obtain interacting

DNA segments, in contrary to 3C-based techniques. As a result, a more precise representation of the spatial interactions between various genomic regions can be obtained by GAM [14]. At the single-cell level, GAM offers abundant information, enabling a more thorough comprehension of the differences in genomic architecture between individual cells [14][15]. GAM can be carried out with a minimal quantity of cells, serving as an effective tool for examining samples with a limited cell count [15]. It measures locus co-segregation, extracts radial positions, and infers chromatin compaction. This offers a thorough understanding of the genome's three- dimensional structure [15].

GAM is an important advancement in the field of genome architecture research. GAM provides a more flexible, thorough, and comprehensive way of understanding the 3-dimensional organization of the genome, overcoming multiple drawbacks of 3C-based techniques, like the need for ligation and many cells. GAM is an effective technique for exploring the complex interactions between genome structures and functions because of its capability to capture multi-way contacts and offer extensive single-cell data. This has significant implications for the study of genetics, epigenetics, and disease.

GAM enables the measurement of chromatin contacts and other features of three-dimensional chromatin topology. This method is based on sequencing DNA from a large collection of thin nuclear sections. The researchers identified specific interactions between active genes and enhancers across very large genomic distances [14][15]. The primary output of GAM is segregation table representing the genome wide interactions.

Each row in this table represents a specific genomic window (region of the genome) and each column represents a nuclear profile (NP) which is a slice from a single cell. Each NP provides information about how genomic regions are spatially organized. By combining information from multiple NPs (columns), we create an array of single-cell data. This array allows us to study the chromatin organization within a specific genomic region of interest (GRI). Researchers can analyze these contacts to understand how genes, enhancers, and other regulatory elements interact [14][15]

3. METHODS

Our pipeline for data analysis is a systematic approach for extracting useful data from large and complicated GAM dataset. It converts these data into useful outcomes using complex algorithms. It offers a thorough overview of every phase in this process, starting from the first data input to the final clustering result.

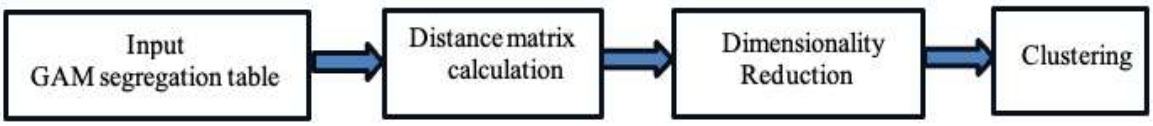


Figure 2. Project pipeline.

The above figure shows the architecture of this project. The pipeline starts with an input of GAM segregation table. Next, the Jaccard distance (see section 3.1) between each pair of samples is calculated. Then, dimensionality reduction is performed using the multidimensional scaling technique, which is a technique that reduces the dimensionality of the data by projecting it to lower dimensional space while preserving the distances between the original observations as much as possible. The final step is to apply a clustering algorithm, which separates the data points into clusters according to how similar or distinct they are.

In this project, we evaluate the performance of multidimensional scaling (MDS) not only in producing well-formed clusters with low intra-cluster distance and high inter-cluster distance, but also in grouping the clusters with high purity.

3.1. GAM Segregation Table Description

GAM enables the measurement of chromatin contacts and other features of three-dimensional chromatin topology. This method is based on sequencing DNA from a large collection of thin nuclear sections. The primary output of GAM is segmentation table representing the genome wide interactions. Each row in this table represents a specific genomic window (specific region of the genome) and each column represents a nuclear profile (NP) which is a slice from a single cell. Each NP provides information about how genomic regions are spatially organized. The entries in the table indicate whether a particular genomic window (row) was captured by a specific NP (column). If the value of the cell in the GAM table is '1' it indicates the genomic window is present in the NP and if the value is '0' it indicates that genomic window is not captured by the NP. These contacts reveal how different parts of the genome physically interact. By combining information from multiple NPs (columns), we create an array of single-cell data. This array allows us to study the chromatin organization within a specific genomic region of interest (GRI). Researchers can analyze these contacts to understand how genes, enhancers, and other regulatory elements interact [14][15].

3.2. Distance Matrix Calculation

Calculating a distance matrix to measure the similarity/dissimilarity between data points is an important step in the pipeline. We have calculated this distance matrix using

Jaccard distance. It can be defined as the measure of dissimilarity between two data sets. One way to quantify how distinct two sets is to use the Jaccard distance, which works especially well with binary data [16]. The data in the GAM segregation table is binary indicating presence or absence of a genomic window in a nuclear profile with 1's and 0's respectively. Moreover, GAM data has a greater number of 0's compared to 1's (the data is sparse) but Jaccard distance concentrates on the existence of elements rather than matching zeros [17][18]. However, it is more meaningful to look at shared contacts (genomic window present in the nuclear profile) rather than shared non-contacts (genomic window not present in the nuclear profile). Therefore, calculating distance matrix with Jaccard distance is the right choice for the GAM data.

Jaccard distance can be calculated by subtracting the Jaccard index from 1. Jaccard index or Jaccard similarity coefficient is a statistic used for accessing the similarity of sample sets. Jaccard distance between every pair of NPs is calculated as shown below.

	Nuclear profile (NP1)	Nuclear profile (NP2)
Window 1	0	0
Window 2	0	1
Window 3	1	0
Window 4	1	1

For a data set like this, Jaccard similarity coefficient (J) can be calculated as

$$J = \frac{M11}{M10 + M10 + M11}$$

Where,

M11 = 1 (Number of rows where both NP1 & NP2 has value of '1'.

M01 = 1 (Number of rows where NP1 has value of '0' & NP2 has value of '1').

M10 = 1 (Number of rows where NP1 has value of '1' & NP2 has value of '0').

M00 = 1 (Number of rows where both NP1 & NP2 has value of '0').

Here, J = $\frac{1}{3}$

Jaccard distance (JD) = 1 – J

Here, JD = 1- $\frac{1}{3}$ = $\frac{2}{3}$

3.3. Dimensionality Reduction

Dimensionality reduction is a process that reduces the number of features under consideration in a dataset. Each feature represents a dimension that partly characterizes data points within the dataset. This technique is crucial in clustering for several reasons.

Firstly, high-dimensional spaces can make distance measures less effective, as all points tend to be equidistant from each other. Dimensionality reduction helps mitigate this issue by reducing the number of dimensions, making distance metrics more meaningful. Secondly, high dimensionality combined with large sample size leads to computational challenges. Algorithms become computationally expensive, requiring substantial processing power and memory. Instability arises because small disturbances in the data can significantly affect results. By reducing the number of dimensions, the data becomes simpler to process, which can lead to faster and more efficient clustering. While

dimensionality reduction can result in some information loss, the proposed solution Multidimensional scaling technique aims to minimize this loss by preserving the pairwise distances between data points. The lost information is less important for the structures or patterns of interest in data. Thirdly, dimensionality reduction can help eliminate irrelevant features from the data, which can improve the accuracy and robustness of clustering algorithms. Additionally, lower- dimensional data is easier to visualize and interpret. This can be particularly helpful for understanding the structure and relationships within the data. Finally, by focusing on the most relevant features, dimensionality reduction can lead to more distinct and meaningful clusters, enhancing the performance of clustering algorithms [21].

3.3.1. Existing Solution

The same data processing pipeline is used in the existing solution, but the technique used in dimensionality reduction step is different. A technique called Uniform Manifold Approximation and Projection (UMAP) has been used to reduce the complexity of high-dimensional GAM data. This algorithm first creates a high-dimensional graph of the GAM data, where each data point is connected to its nearest neighbors, for example, suppose we have a dataset of different types of fruits, where each fruit is described by a set of features such as color, weight, size, shape, taste, etc. This is our high-dimensional data. UMAP starts by creating a graph where each fruit (data point) is connected to its nearest neighbors based on the similarity of their features.

For instance, an apple might be connected to another apple because they have similar color and size, but it might also be connected to a pear because they have a similar shape. After we have this high dimensional graph, it is converted into a simplicial complex, which effectively represents the shape and structure of the data. UMAP then uses Stochastic Gradient Descent (SGD) to optimize the layout of a similar graph in the lower-dimensional space, aiming to achieve the greatest resemblance with high dimensional graph. The final result is a low-dimensional, simpler representation of the data that largely retains its original structure.

The PhD student has used this Uniform Manifold Approximation and Projection (UMAP) technique and explored all the parameter settings. This approach has been proven to be effective. However, the proposed solution explores an alternative technique called Multi-Dimensional Scaling (MDS) for the dimensionality reduction step in the same pipeline to determine whether it can generate better results.

3.3.2. Proposed Solution

The proposed solution for dimensionality reduction step is to utilize multidimensional scaling algorithm. This technique will handle the challenge of high dimensional GAM data by projecting it onto lower dimensional space and help with further analysis.

Multidimensional scaling (MDS)

Multidimensional scaling (MDS) technique is used to visualize dissimilarities or

distances between collection of objects. It is a powerful technique used to project data from higher dimensional space to lower dimensional space while fully preserving distances between every pair of points. Using the idea of distance as its foundation, MDS seeks to locate a data projection that minimizes discrepancies between the distances in the original space and the lower-dimensional space.

It is a widely used technique to view high-dimensional, complex data and to spot relationships and patterns that might not be visible in the original high dimensional space. Numerous data types, such as categorical, numerical, and mixed data, may be used with it [22].

The size of GAM segregation table (as shown in Fig 3.) is 87805 rows (windows) and 668 columns (Nuclear profiles). On this data set, we calculate pairwise Jaccard distances between every pair of nuclear profiles. With this Jaccard distance, we generate a symmetric distance matrix of a size 668 rows (NPs) and 668 columns (NPs). The input for the MDS algorithm will be this distance matrix.

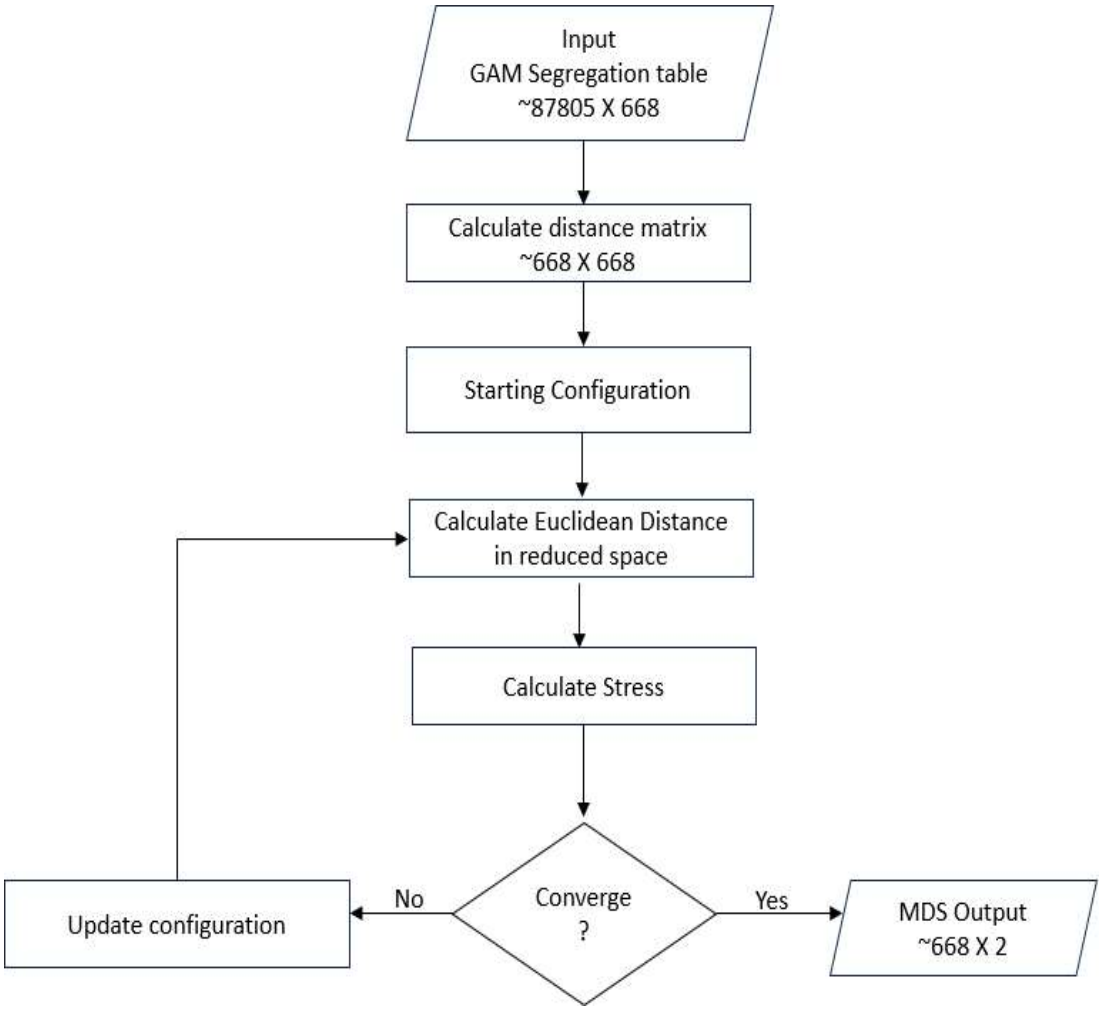


Figure 3. Flow chart of MDS algorithm.

The Scaling by Majorizing a Complicated Function (SMACOF) algorithm is a multidimensional scaling algorithm which aims to minimize the objective function (stress). Stress majorization, also known as the Guttman Transform is a method that monitors stress and guarantees a decrease in this stress throughout each iteration of SMACOF until either the change in stress is close to zero, or the maximum number of

iterations have been achieved. As stress decreases, the algorithm will converge on a lower dimensional space that preserves the distances that were in the original high dimensional space. SMACOF performs the following steps to complete the previously explained computations: Transform guarantees a decrease in stress throughout each iteration until either the change in stress is close to zero, or the maximum number of iterations has been achieved. As stress decreases, the algorithm will converge on a lower dimensional space that preserves the distances that were in the original high dimensional space. SMACOF performs the following steps to complete the previously explained computations:

Step1: Assigns a random configuration in reduced space

The SMACOF algorithm will generate a reduced space (N-dimensional space) and assign a new random start configuration in that reduced space, based on the number of "N-components" parameter that the user defines. Here, the number of dimensions in the reduced space (the MDS output) is represented by the N-components parameter.

Step2: Calculate pairwise Euclidean distance in the reduced space

In this step, the SMACOF algorithm calculates the Euclidean distance between every pair of points in the reduced space. Euclidean distance is the actual straight-line distance between two points in the Euclidean space. This distance is also called the Pythagorean distance. The following formula can be used to compute Euclidean distance.

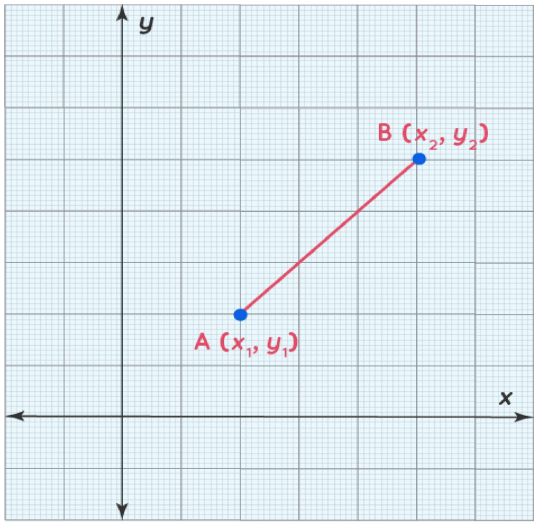


Figure 4. Graphical representation of Euclidean distance.

Euclidean Distance Formula is $d = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}$

where,

d is the Euclidean distance between two points A & B

(x_1, y_1) are the coordinates of point A and

(x_2, y_2) are the coordinates of point B

Step3: Calculate stress by comparing distances in reduced space to distances in original space

In this step, it calculates the stress by comparing the differences between the distances in original space and reduced space (predicted distances). Depending on the differences between original and predicted distances, stress is the measure of goodness of fit. A better fit is achieved with low stress value, which means that the distances between the points in the reduced space correctly represent the original distances.

The following formula can be used to compute Stress.

$$\text{Stress} = \frac{\sum(\text{Original distance} - \text{Euclidean distance})^2}{2}$$

EPS is a parameter in MDS algorithm which represents the maximum allowable relative change in stress that signifies convergence, indicating that more iterations will not considerably enhance the solution. The MDS algorithm's default value for the EPS parameter is 1e-3, which indicates that the process stops when the difference between the stress values from the previous iteration and the current iteration is less than 0.001. This threshold prevents unnecessary calculations while guaranteeing a result that is accurate enough.

Step4: Update configuration in reduced space, to minimize stress

MDS algorithm updates configuration in reduced space by utilizing an optimization technique called Guttman transform. This technique uses an iterative process to update the data points in the lower dimensional space. Based on the difference between the calculated distances in the lower dimensional space and the actual distances in the original space, the positions of data points are refined in each iteration. Regardless of the starting point, this technique guarantees convergence to the solution with the least amount of stress. Guttman transform can be used to compute new configurations as shown below.

$$X' = 1/n\ (B*X), \text{ Where}$$

n = number of dimensions

B = - (original distances/Euclidean distance in reduced space)

X = initial configuration

The algorithm iterates between step2 to step4 until the stopping criteria is met. There are two stopping criteria for this algorithm.

1. max_iter: The maximum number of iterations that the SMACOF algorithm can complete in a single execution is determined by this parameter. By default, this parameter is set to 300 iterations.

2. eps: This parameter is a relative stress tolerance. The algorithm will terminate if the stress difference between the previous iteration and the current iteration falls below this value. Its default value is 0.001 [23].

3.4. Clustering

Following the dimensionality reduction of the GAM data to reduce its complexity, the data is now ready for clustering. In this project we have used Leiden community detection algorithm [24] to find clusters in reduced dimensional space. This method enables the identification of distinct groups within the data depending to their similarities, allowing for more detailed understanding of the GAM data.

3.5. Evaluation Metrics

In this project, the Calinski-Harabasz (CH) index is used as an evaluation metric to determine the cluster quality. Each cluster's quality can be determined by the ratio of inter cluster separation to intra cluster separation, using Euclidean distance. The CH index demonstrates how densely packed and well-separated the clusters are. A greater CH index represents better clustering performance, and we choose the best parameters of MDS algorithm based on this CH index. Formula to calculate CH index is,

$$\text{CH} = \frac{[\sum_{m=1}^{M} K_m ||cc_m - c||^2]/(M-1)}{[\sum_{m=1}^{M} \sum_{j=1}^{K_m} ||f_k^{cm} - cc_m||^2]/(N-M)}$$

Where,

M is the number of clusters,

N is the total number of samples in the dataset (N > M),

c is the centroid of the entire dataset,

K_m is the number of samples in cluster m,

c_m is the centroid of cluster m and

f_k is the individual data sample in cluster m [25].

Another evaluation metric is to determine cluster purity. In this project, "cluster purity" refers to the percentage of particular type of cell allocated to each cluster. We used "Ground truth" data set, meaning each point has already been correctly labeled in advance. By comparing the clusters generated by our model with the labels in this data set, we can determine cluster purity.

4. EXPERIMENTS AND RESULTS

4.1. Experiments

In this section we delve deeper into the details of the project. We concentrate on the MDS technique and its related parameters. These parameters have a significant impact on the outcomes of the MDS technique.

MDS Parameters

n_components:

This parameter specifies how many dimensions to use to incorporate the differences. It is set to 2 by default, which indicates that a two-dimensional space will be used to show the data. If necessary, this value can be changed to explore higher-dimensional embeddings.

metric:

Determines whether to execute non-metric MDS or metric MDS. It accepts Boolean values, if set to True, metric MDS is performed; otherwise, nonmetric MDS is performed. Its default value is set to true.

n_init:

This is the number of times that various initializations will be used to run the SMACOF algorithm. The run with the lowest final stress will determine the best output of the runs, which will determine the final outcome. The default value is 4.

max_iter:

For a single run of the SMACOF algorithm, this is the maximum number of iterations allowed. The default value is 300.

verbose:

This parameter determines the level of verbosity which means it provides the amount of data that is recorded in the logs or console during algorithm execution. The default value is 0.

eps:

The maximum allowable relative change in stress, or it indicates convergence and the point at which more iterations will not significantly enhance the solution.

n_jobs:

For the computation, this is the number of jobs to use. Every run of the algorithm is computed in parallel if multiple initializations (n_init) are used. The default value is none but if it is set to -1 all processors will be used.

random_state:

The random number generator that is used to initialize the centers is determined by this parameter. For results that can be replicated across several function calls, pass an integer.

dissimilarity:

This parameter is the dissimilarity measure that is used in the algorithm. The default value is set to ‘euclidean’, which means that pairwise Euclidean distances between points is calculated in the GAM dataset. If the parameter is set to ‘precomputed’, distances of

our choice that is pre-computed can be passed directly.

normalized stress:

Multidimensional scaling that is non-metric uses the normalized stress parameter (MDS). It determines whether the default stress value calculation should be replaced with a normalized stress value (Stress-1) or not.

The final output of Multidimensional Scaling (MDS) technique is dependent on three parameters that is n_components, max_iter and eps. By determining how many dimensions to project the data onto, n_components basically adjust the number of dimensions in the lower dimensional space. The degree to which the algorithm optimizes the arrangement of points within that selected dimensionality is determined by max_iter and eps. The verbose, n_jobs, and normalized_stress parameters had no effect on how the algorithm solves the problem and hence, have no impact on the final output of the algorithm. These parameters are only used to show statistics within different aspects of the algorithm. The n_init parameter affected the output to some extent, but after experimenting with different parameter settings, it became evident that no pattern was forming in the results. Since the initial configuration is always random, any influence from n_init on the result was not substantial, and therefore, it was eliminated from the list of parameters to explore during the hyperparameter search.

Tuning the max_iter, n_components, and eps parameters with a range of values allowed me to observe a trend in the resulting CH score, giving an optimal set of parameters for each. The range of values I have used is 2 to 100 dimensions for the

n_components parameter, 300 to 20000 iterations for the max_iter parameter, and 1e-14 to 1e-20 for the eps parameter. The trends that I observed while experimenting with these wide range of values are shown below.

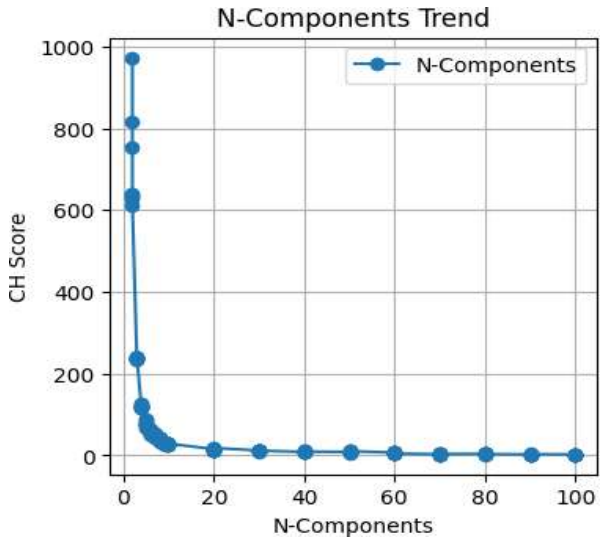


Figure 5. N-components vs CH score.

Figure 5 illustrates the variation in CH score based on the number of components. The x-axis represents a number of components (number of dimensions in reduced space). N- components range from 2 to 100. The y-axis represents CH score. Since the CH score sharply decreases as the N-components rise, a higher CH score will result from fewer components.

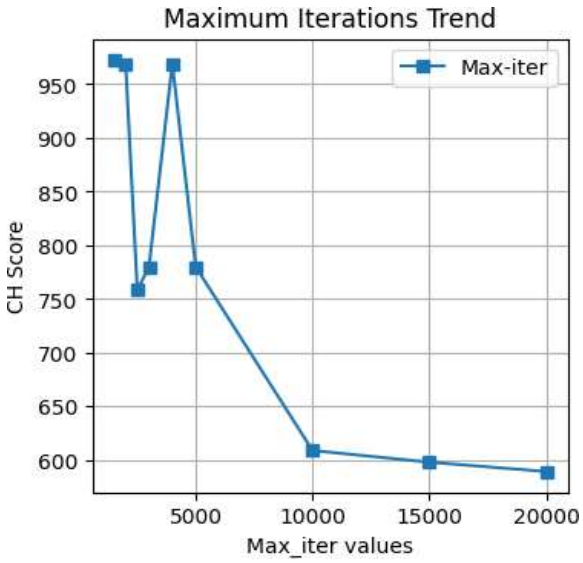


Figure 6. Max_iter values vs CH score.

Figure 6 illustrates the variation in CH score based on the maximum number of Iterations. The x-axis represents the maximum number of iterations. Max_iter ranges from 1500 to 20000. The y-axis represents CH score. Initially, when we increased the maximum number of iterations, CH score decreased and after around 2500 iterations CH score increased sharply and reached a highest point at around 4000 iterations. However, beyond this point, the increase in maximum number of iterations resulted in a steady decrease in the CH score until 20000 iterations.

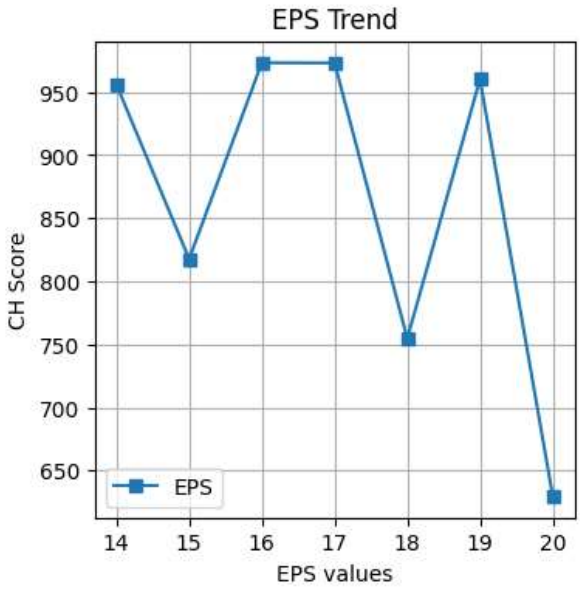


Figure 7. EPS values vs CH score.

Figure 7 illustrates the variation in CH score based on EPS values. The x-axis represents EPS values (Difference in stress between the previous iteration and the current iteration). EPS values range from 1e-14 to 1e-20. The y-axis represents CH score. The highest CH score can be attained at 1E-16 and 1E-17, but the EPS values do not rise or fall linearly with CH score, instead, the trend is inconsistent. The algorithm utilized the maximum number of iterations as the stopping point instead of eps values beyond 1E-20. The ideal values for eps are therefore 1E-16 and 1E-17.

Besides experimenting with various dimensionality reduction technique parameter settings, I also considered the impact of employing various distance measurements to calculate the distance matrix to find out how they affected the results. I experimented with distance measures, such as Braycurtis, Canberra, Chebyshev, Cityblock, Correlation, Cosine, Dice, Euclidean, Hamming, Jaccard, Jensenshannon, Matching, Makowski,

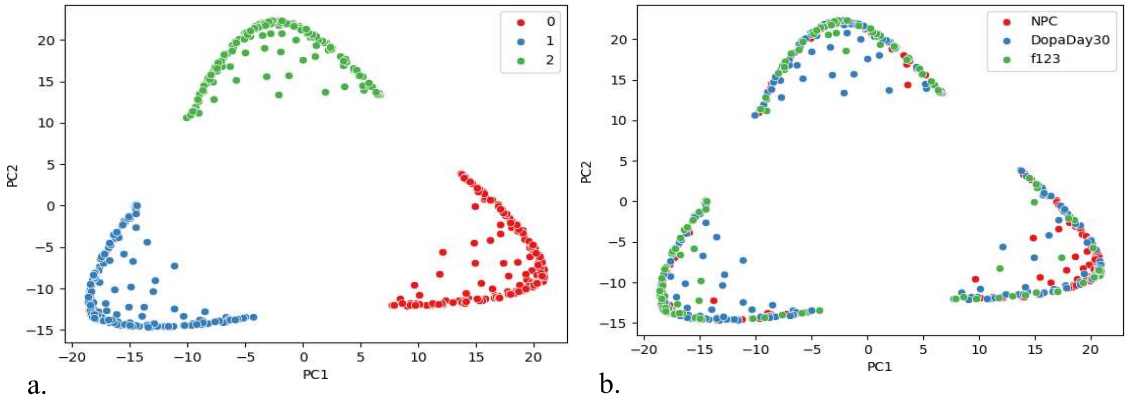
Rogerstanimoto, Russellrao, Seuclidean, Sokalmichener, Sokalsneath, Sqeuclidean and Yule [26], and conducted a thorough parameters search for each of these various distances to see if any of them generated more pure clusters than Jaccard distance and produced the most informative and comprehensible low-dimensional representation of the GAM data.

4.2. Results

In this section we will look at scatter plots, pie charts and the 3Dplot. These will help us understand the maximum cluster purity that we could achieve with MDS algorithm.

4.2.1. PCA Plots

In the below figures (Figure 8a & 8b) x-axis represents the first principle component PC1 and y-axis represents the second principle component PC2. These are the two directions where the data varies the most. Each point on the plot represents a single cell.



a. b.

Figure 8. Showing principle component analysis plot for unlabeled data and labeled data. 8a. Unlabeled data. 8b. Labeled data.

In Figure 8a, we can observe that the data points are grouped into three clusters, but they are not labeled. Although we see clear separation in data points, we are not sure what each cluster represents.

In Figure 8b, each data point is labeled according to their cell type: “NPC”, “DopaDay30”, or “f123”. This plot provides additional information about the data, allowing us to identify the cell type to which each point belongs. It is evident that these clusters do not correspond to pure single cell populations and there is no clear separation of cell types, which means the MDS technique is not effectively separating cell types.

4.2.2. Pie charts

The pie charts in the below figures (Figure 9a, 9b and 9c) represents the distribution of different cell types within each cluster. ‘DopaDay30’ cell type is represented by blue colored segment, ‘f123’ cell type is represented by green colored

segment, and ‘NPC’ cell type is represented by red colored segment.

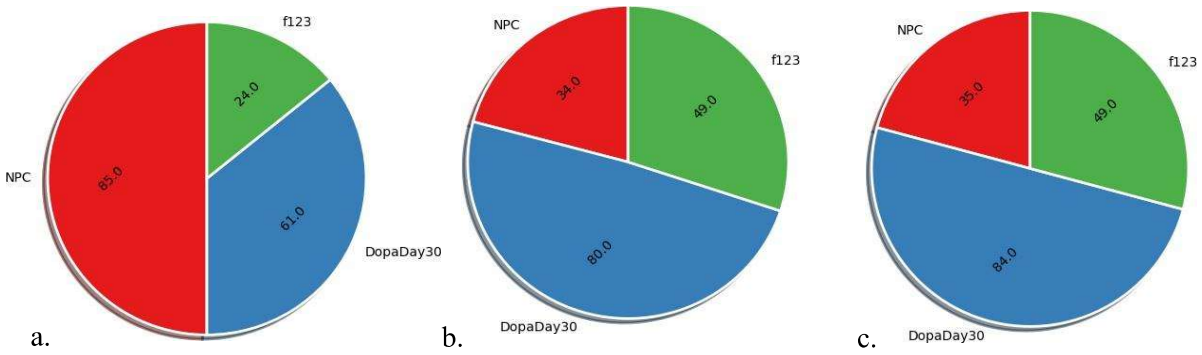


Figure 9. Showing pie charts for three clusters. 9a. Pie Chart for cluster 1. 9b. Pie chart for cluster 2. 9c. Pie chart for clusters 3.

From Figure 9a, we can observe that cluster-1 has 85 NPC cells, 24 f123 cells, and 61 DopaDay30 cells. That is this cluster has 50% NPC cells, 14.1% f123 cells, and 35.9% DopaDay30 cells.

From Figure 9b, we can observe that cluster-2 has 34 NPC cells, 49 f123 cells, and 80 DopaDay30 cells. That is this cluster has 20.9% NPC cells, 30.1% f123 cells, and 49.1% DopaDay30 cells.

From Figure 9c, we can observe that cluster-3 has 35 NPC cells, 49 f123 cells, and 84 DopaDay30 cells. That is this cluster has 20.8% NPC cells, 29.2% f123 cells, and 50% DopaDay30 cells.

In all the pie charts above, the maximum purity of cluster was 50%. This shows that the cluster quality is not good.

4.2.3. 3D Plot

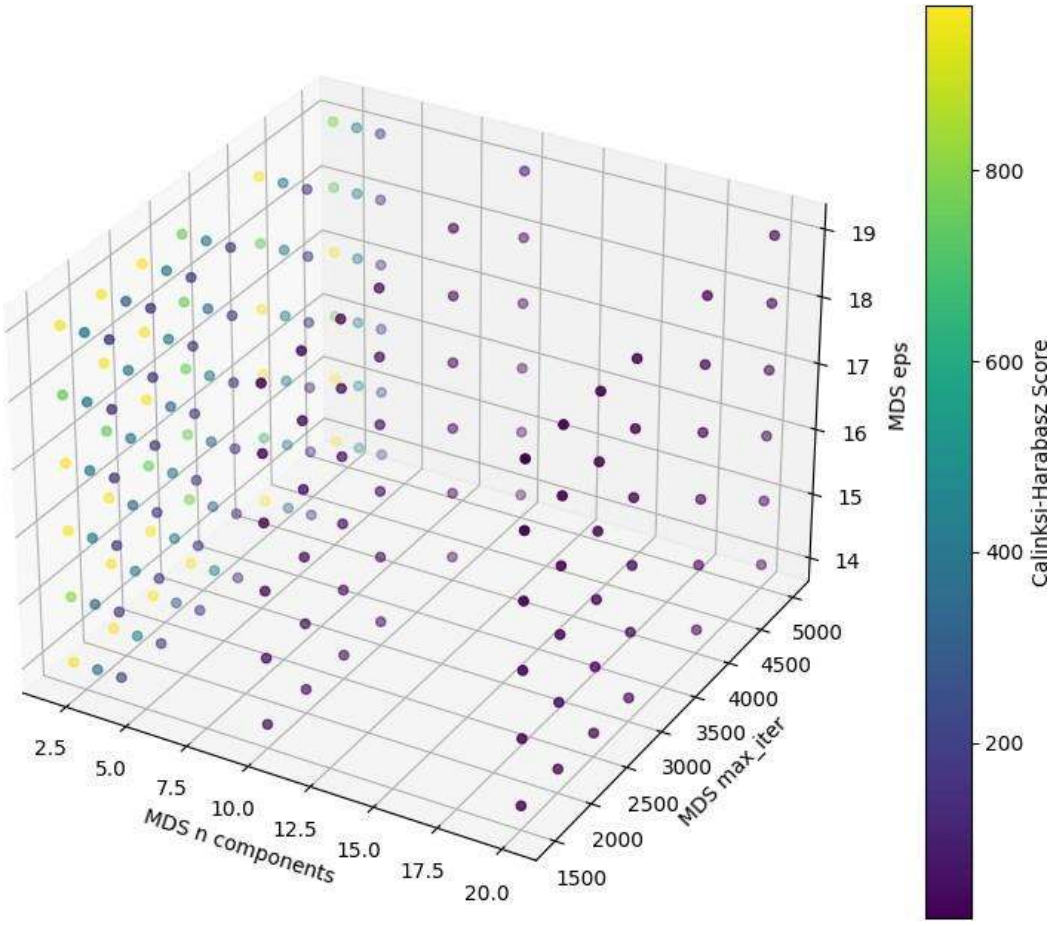
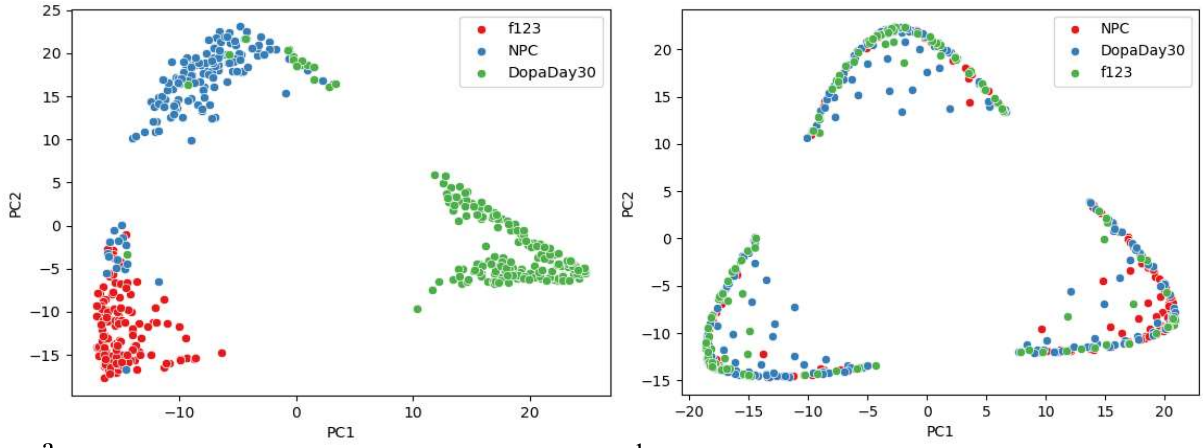


Figure 10. 3D graphical representation of number of components, maximum iterations, and EPS parameters with their respective CH scores.

In the above figure, x-axis represents the number of components in the reduced space (MDS n components) ranging from 2 to 20 dimensions. The y-axis represents maximum number of iterations of the algorithm (MDS max_iter), ranging from 1500 to 5000 iterations. Z-axis represents the difference in stress values of previous iteration and current iteration of the algorithm (MDS eps), ranging from 1e-14 to 1e-19. A distinct set

of MDS parameters (n Components, max_iter, and eps) is represented by each dot. The color scale on the right side of the graph shows how the colors of the dots correlate with CH Scores. The color scale goes from purple (representing a lower CH score) to yellow (representing a higher CH score). It is evident that the components with the lowest n component value have the greatest CH scores. Therefore, 2 is the ideal n-component value. Additionally, we can see that the range of eps from 1e-14 to 1e-19 and the maximum number of iterations between 1500 and 5000 yields the highest CH score.

4.3. Comparison of UMAP and MDS results

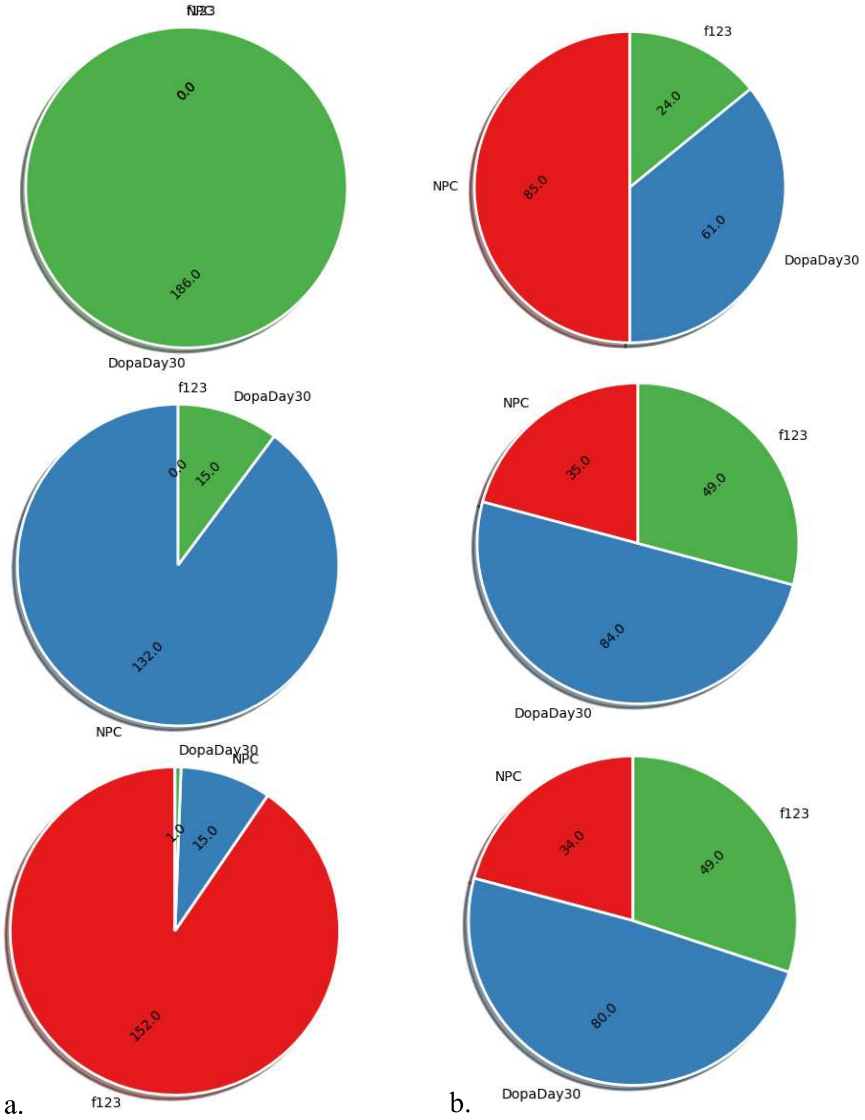


a. b.

Figure 11. Comparison of UMAP and MDS scatter plot results. 11a. UMAP scatter plot. 11b. MDS scatter plot.

The scatter plot produced using UMAP has three distinct clusters, each marked by different color and each cluster represents a different cell type. The well-separated structure of these clusters suggests that UMAP is successfully discriminating between

different cell types. Although, the scatter plot produced using MDS also uses different colors to represent different cell types, each cluster has multiple colors in it. This mixing of colors in clusters indicates that the clusters are not well-separated as they are in UMAP scatter plot.



a. b.

Figure 12. Comparison of UMAP and MDS pie chart results. 12a. UMAP pie chart results. 12b. MDS pie chart results.

Large segments in UMAP pie charts show that the cell types are efficiently being distinguished, as each segment indicates a different cell type. However, compared to UMAP, MDS pie charts have fewer distinct segment sizes, implying that they are less successful in differentiating between the various cell types.

By looking at these results we can observe that UMAP technique performs better at producing clusters with significantly higher purity than the MDS technique. The purity of the clusters generated by UMAP implies that this approach is more successful in capturing the inherent structure of the data, resulting in more significant and unique clusters. Therefore, based on the findings presented in this section, MDS Technique is not as effective as UMAP technique in identifying different cell types.

5. CONCLUSION AND FUTURE WORK

In conclusion, we aimed to group the data from the GAM segregation table into meaningful clusters using a dimensionality reduction technique called multidimensional scaling and evaluate the performance of this technique based on how well the clusters are formed. The degree of separation between the clusters, or the CH Index, is one metric we used to assess the quality of the clusters. Upon closer inspection, however, we discovered that the clusters were not necessarily composed of a single cell type. Thus, our clusters were impure. We also looked at the extensive parameter search of the algorithm by choosing a wide range of values for each parameter and how it affected the quality of clusters. By experimenting with various parameter settings, we observed which ones worked best. Even though we found the optimal parameters after a broad search, we could not achieve good cluster purity. Hence, MDS is not a suitable technique for this project.

As part of future work, we will investigate dimensionality reduction techniques other than MDS. Cluster purity and separation may be improved by using methods such as t-SNE (t-Distributed Stochastic Neighbor Embedding), NMF (Nonnegative Matrix Factorization), Isomap or Auto Encoders.

6. REFERENCES

[1] Ling, X., Liu, X., Jiang, S., Fan, L., & Ding, J. (2022). The dynamics of three-dimensional chromatin organization and phase separation in cell fate transitions and diseases. Cell Regeneration, 11(1), 42.

[2] Liu, J., & Vinck, M. (2022). Improved visualization of high-dimensional data using the distance-of-distance transformation. PLoS computational biology, 18(12), e1010764.

[3] Jovic, D., Liang, X., Zeng, H., Lin, L., Xu, F., & Luo, Y. (2022). Single-cell RNA sequencing technologies and applications: A brief overview. Clinical and Translational Medicine, 12(3), e694.

[4] Yuan, G. C., Cai, L., Elowitz, M., Enver, T., Fan, G., Guo, G., ... & Tirosh, I. (2017). Challenges and emerging directions in single-cell analysis. Genome biology, 18, 1-8.

[5] Heumos, L., Schaar, A. C., Lance, C., Litinetskaya, A., Drost, F., Zappia, L., ... & Theis, F. J. (2023). Best practices for single-cell analysis across modalities. Nature Reviews Genetics, 1-23.

[6] Zhou, T., Zhang, R., & Ma, J. (2021). The 3D genome structure of single cells. Annual review of biomedical data science, 4, 21-41.

[7] Zhang, R., Zhou, T., & Ma, J. (2022). Ultrafast and interpretable single-cell 3D genome analysis with Fast-Higashi. Cell Systems, 13(10), 798-807.

[8] Zhang, R., Zhou, T., & Ma, J. (2022). Multiscale and integrative single-cell Hi-C analysis with Higashi. Nature biotechnology, 40(2), 254-261.

[9] Xiong, K., Zhang, R., & Ma, J. (2023). scGHOST: Identifying single-cell 3D genome subcompartments. bioRxiv, 2023-05.

[10] Ay, F., & Noble, W. S. (2015). Analysis methods for studying the 3D architecture of the genome. Genome biology, 16, 1-15.

[11] Mohanta, T. K., Mishra, A. K., & Al-Harrasi, A. (2021). The 3D genome: from structure to function. International Journal of Molecular Sciences, 22(21), 11585.

[12] Jerkovic, I., & Cavalli, G. (2021). Understanding 3D genome organization by multidisciplinary methods. Nature Reviews Molecular Cell Biology, 22(8), 511-528.

[13] Zhang, Y., Boninsegna, L., Yang, M., Misteli, T., Alber, F., & Ma, J. (2023). Computational methods for analysing multiscale 3D genome organization. Nature Reviews Genetics, 1-19.

[14] Beagrie, R. A., Scialdone, A., Schueler, M., Kraemer, D. C., Chotalia, M., Xie, S. Q., ... & Pombo, A. (2017). Complex multi-enhancer contacts captured by genome architecture mapping. Nature, 543(7646), 519-524.

[15] Welch, L. R., Baugher, C., Zhang, Y., Davis, T., Marzluff, W. F., Welch, J. D., & Pombo, A. (2020). Single-cell analysis of the 3D topologies of genomic loci using genome architecture mapping. bioRxiv, 2020-02.

[16] Chung, N. C., Miasojedow, B., Startek, M., & Gambin, A. (2019). Jaccard/Tanimoto similarity test and estimation methods for biological presence-absence data. BMC bioinformatics, 20(Suppl 15), 644.

[17] Huang, Q., Luo, P., & Tung, A. K. (2023). A New Sparse Data Clustering Method Based on Frequent Items. Proceedings of the ACM on Management of Data, 1(1), 1-28.

[18] Besta, M., Kanakagiri, R., Mustafa, H., Karasikov, M., Rätsch, G., Hoefler, T., & Solomonik, E. (2020, May). Communication-efficient jaccard similarity for high-performance distributed genome comparisons. In 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS) (pp. 1122-1132). IEEE.

[19] McInnes, L. (n.d.). How umap works. How UMAP Works - umap 0.5 documentation. https://umap-learn.readthedocs.io/en/latest/how_umap_works.html#the-umap-algorithm

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

[21] Ros, F., & Riad, R. (2023). Feature and Dimensionality Reduction for Clustering with Deep Learning. Springer Nature.

[22] Kruskal, J. B., & Wish, M. (1978). Multidimensional scaling (No. 11). Sage.

[23] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

[24] Traag, V. A., Waltman, L., & van Eck, N. J. (2019). From Louvain to Leiden: Guaranteeing well-connected communities. Scientific Reports, 9(1). https://doi.org/10.1038/s41598-019-41695-z

[25] Ekemeyong Awong, L. E., & Zielinska, T. (2023). Comparative Analysis of the Clustering Quality in Self-Organizing Maps for Human Posture Classification. Sensors, 23(18), 7925.

[26] Pdist. pdist - SciPy v1.14.0 Manual. (n.d.). https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html