Assignment 3: Dimensionality Reduction

Spyridon Alvanakis-Apostolou

June 2, 2023

1 Library

I have chosen to utilize the `Scanpy` library, which is widely used for single-cell analysis. Scanpy is a highly prevalent library for analyzing gene expression data at the single-cell level. It offers a comprehensive toolkit in collaboration with anndata, encompassing preprocessing, visualization, clustering, trajectory inference, and differential expression testing. This Python-based implementation effectively handles datasets comprising over one million cells.

2 Preprocess

The entire report is focused on the initial dataset provided to us, although the methods employed are also applicable to the remaining datasets. Each dataset consists of 200 genes and 200 cells. During the preprocessing stage, I determined the gene count per cell and the cell count per gene. This information proved useful in categorizing and visualizing the expression levels of genes and cells.

The next step involved removing certain cells based on their gene count to ensure quality control. Three cells exhibited minimal gene expression and were consequently excluded from further analysis. It is a common practice to also reduce the number of genes by selecting the "highly variable genes" criterion based on their variance. However, in this particular dataset, I chose not to utilize this approach due to the small number of genes. It is important to note that a small number of genes can introduce bias to the dataset. Therefore, reducing the number of genes in such cases may have a detrimental impact on the analysis. Eventually, none of the genes were excluded.

After that, the dataset underwent normalization and logarithmic transformation to mitigate scaling differences among its values. This step was crucial because dimensionality reduction techniques like PCA are highly sensitive to variations in data scaling. Therefore, it was imperative to ensure that the data was uniformly scaled to facilitate accurate and meaningful results during the application of dimensionality reduction methods.

3 Dimentionality Reduction

3.1 PCA

Principal Component Analysis (PCA) is a linear approach used for dimensionality reduction, which improve the data visualization while preserving the maximum of information of original data. The loss of information is investigated by the variance between original data and the compressed (projected) data, which aim is to maximize the variance.

Basicaly, the eigenvectors are sorted based on their corresponding eigenvalues. This sorting process allows to identify the most significant principal components. To determine the number of principal components to retain, a cutoff point is chosen, which represents the desired level of variance explained. We can select the eigenvectors that contribute to the cumulative variance reaching or exceeding the chosen threshold. Finally, the dimensionality reduction is applied by retaining only these selected principal components, effectively reducing the dimensionality of the dataset.

There are several approaches to selecting the cutoff point in PCA. The first method1 involves us-

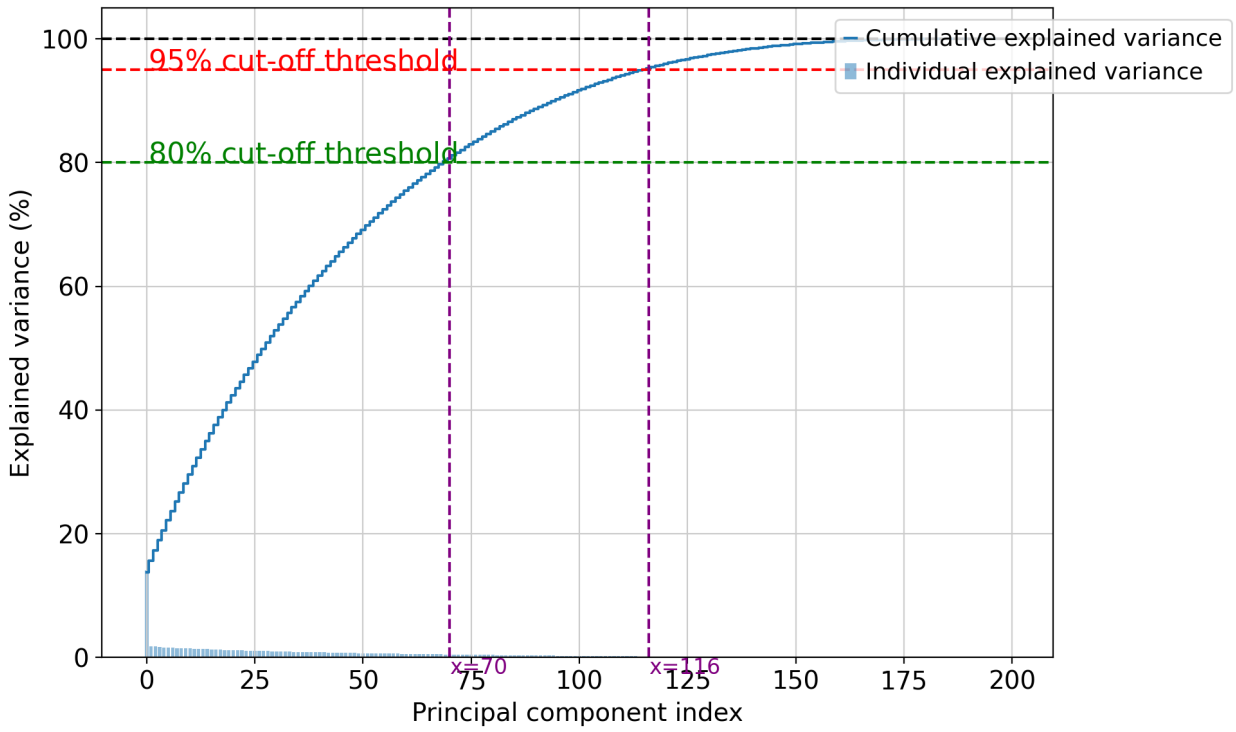

Figure 1: Upon observing the cumulative variance plot, we find that in order to capture 80% of the cumulative variance, we need to retain 70 components. For a more comprehensive coverage, retaining 95% of the cumulative variance would require 116 components. Additionally, examining the eigenvalues, we notice that the first eigenvalue appears significantly larger compared to the remaining eigenvalues. This suggests that the first principal component contains a substantial amount of information when compared to the other components individually.

ing cumulative variance to retain a specific percentage of information, such as 95% or 80%, which may vary depending on the dataset. This approach ensures that a substantial portion of the dataset's information is preserved. The second method involves utilizing a scree plot9, which plots the eigenvalues against the corresponding principal components. By visually inspecting the plot and applying the elbow rule10, a cutoff point can be determined, retaining only a few of the most significant principal components. However, it's important to note that this approach may result in a loss of substantial information. The third approach, known as the Kaiser rule11, involves keeping only the dimensions or principal components with eigenvalues greater than one. This criterion is based on the statistical significance of the components and aims to retain those that explain a substantial amount of variance in the dataset. It is important to consider the specific characteristics of the dataset and the goals of the analysis when choosing the appropriate method for determining the cutoff point in PCA.

I made the decision to utilize the minimum number of components among the four approaches. This choice was motivated by the potential drawbacks associated with using excessive information that could introduce bias. Each of the methods resulted in a considerable number of principal components, which

would not contribute significantly to the visualization and exploration of clusters in the dataset.

Taking these factors into consideration, I reached the conclusion that retaining the minimum number of components for the first dataset struck the most optimal balance between the visualization aspect and the expression of information.

3.2 T-SNE

T-distributed Stochastic Neighbor Embedding, commonly referred to as t-SNE, is a manifold learning algorithm that operates by creating a probability distribution over the dataset. It subsequently constructs probability distributions in a lower-dimensional space, aiming to minimize the divergence between the distributions.

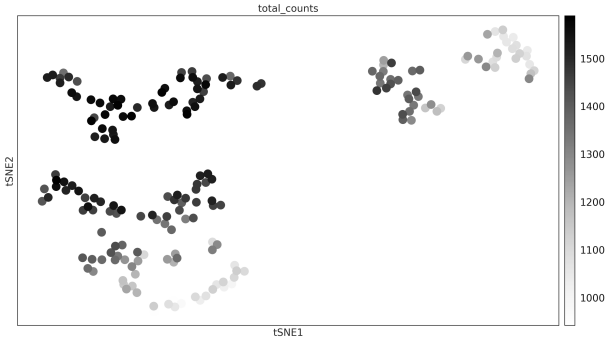

Figure 2: Final 2D illustration using t-SNE

For the t-SNE representation, I specifically selected the 'X_pca,' as it is recommended when dealing with datasets containing more than 50 genes. To achieve a satisfactory 2D visualization, several parameters needed to be set, including the perplexity and the learning rate.

I experimented with different perplexity values, using the default (30), 10, 20, 40 and 50. Additionally, I explored various learning rates, considering that excessively small or large values might hinder clear visualization of data clusters. T-SNE only produces a 2D representation of the data, which means that the number of components is automatically set to 2. This limitation arises from the nature of the t-SNE algorithm, which focuses on visualizing high-dimensional data in a lower-dimensional space.

Considering that t-SNE is primarily used for 2D visualization, my objective was to create a plot that clearly illustrates distinguishable clusters of data points. I also took into account the time-consuming nature of t-SNE and aimed to minimize the computational burden by selecting the smallest possible number of dimensions, perplexity, and learning rate while still achieving the desired visualization result2.

To do that, I generated multiple plots using different parameter combinations to assess how t-SNE visualized the data. By experimenting with various parameter settings, I was able to identify the optimal parameters that produced the desired visualization outcome with well-separated clusters. This iterative approach allowed me to fine-tune the parameters and select the most suitable configuration for generating an informative and visually appealing t-SNE plot.

The parameters that successfully produced the desired visualization were a learning rate of 400, n_pcs (number of principal components) set to 2, and a perplexity value of 10. These parameter selections were made taking into consideration the specific characteristics of the dataset. Given that the dataset had a relatively small number of genes and cells, it was not a complex dataset in terms of dimensionality. Hence, the chosen parameters were tailored to the dataset's simplicity, allowing for an effective visualization of the data in a 2D space. Also, considering the dataset's size, it was logical to select a relatively small perplexity value to avoid overfitting or excessive complexity.

3.3 UMAP

UMAP (Uniform Manifold Approximation and Projection) offers faster and more scalable performance in terms of both dataset size and dimensionality. It also tends to better preserve the global structure of the data. UMAP achieves this by constructing a high-dimensional graph representation of the data and optimizing a low-dimensional graph to closely resemble it in terms of structural similarity.

Among the parameters used in UMAP, two of the most commonly employed are n_neighbors and

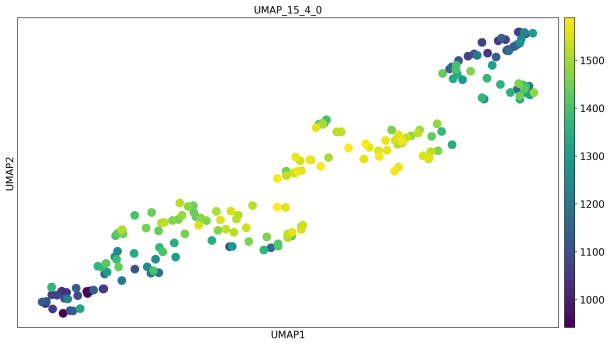

Figure 3: Final 2D illustration using UMAP

min_dist, which comparing with the number of components play a crucial role in controlling the balance between local and global structure in the final projection.

The primary parameter, n_neighbors, determines the number of approximate nearest neighbors used in constructing the initial high-dimensional graph. It effectively governs how UMAP balances between local and global structure. Lower values prioritize local structure by restricting the number of neighboring points considered during analysis in high dimensions, while higher values lean toward capturing the overall structure at the expense of finer details.

The second parameter we will examine is min_dist, which defines the minimum distance between points in the low-dimensional space. This parameter influences the degree of clustering in UMAP embeddings. Lower values result in tightly packed embeddings, while larger values cause points to be more loosely grouped together, with a greater emphasis on preserving the broad topological structure.

To visualize the results of UMAP in a 2D space, I experimented with different combinations of the number of components, the number of neighbors, and the minimum distances. Among these combinations, the configuration that effectively groups the data points together and forms clusters was achieved with 4 components, 15 neighbors, and a minimum distance of 0.

By using 4 components, the UMAP projection was able to capture a sufficient amount of information from the original data while reducing the dimensionality. Setting 15 neighbors allowed for a balance between local and global structure, providing a meaningful representation of the relationships between data points. The minimum distance of 0 ensured that the embeddings were overly compact, resulting in a visually informative visualization.

4 Clustering

A Gaussian mixture model (GMM) is a probabilistic model that assumes the data points are generated from a mixture of multiple Gaussian distributions, each with its own set of unknown parameters. It can be seen as an extension of k-means clustering, incorporating information about both the covariance structure and the centers of the latent Gaussians.

In a GMM, each data point is associated with a particular Gaussian component, and the model aims to estimate the parameters of these Gaussian components to best fit the observed data. The parameters typically include the mean and covariance matrix of each Gaussian component, representing their location and shape in the feature space.

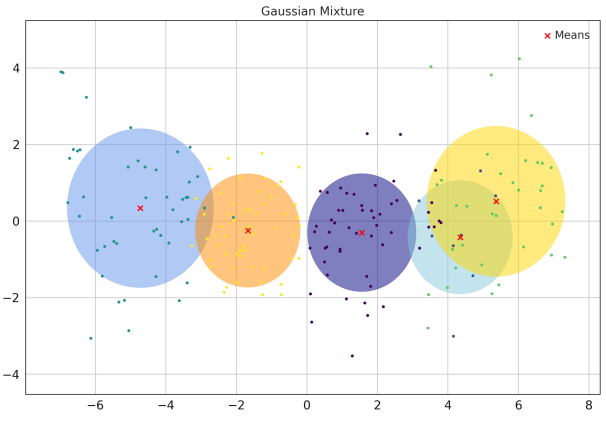

Figure 4: PCA cluster with 5 components

By modeling the data as a mixture of Gaussians, GMMs allow for more flexibility in capturing complex data patterns. They can capture clusters of varying shapes and sizes, as well as the inherent uncertainty

in assigning data points to different clusters.

In order to determine the optimal parameters for the GMM algorithm, I experimented with different settings. I explored the use of 8-10 components, as they seemed enough to separate the visualized clusters. Additionally, I varied the values of the reg_covar and the covariance type parameters to capture different shapes of clusters and explored different numbers of initializations to gain a comprehensive understanding of the results. Additionally, I conducted experiments with different numbers of initializations to assess the stability and robustness of the GMM results.

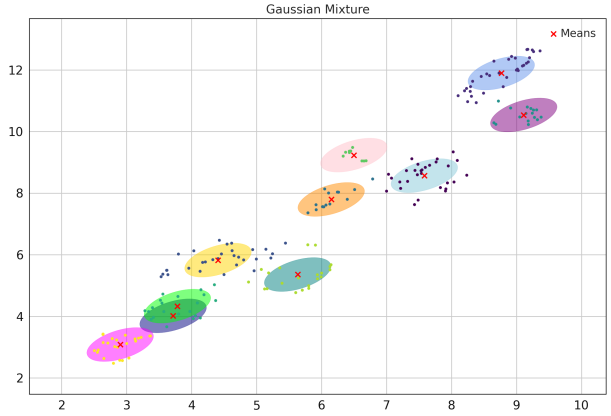

Figure 5: UMAP clusters with 10 components

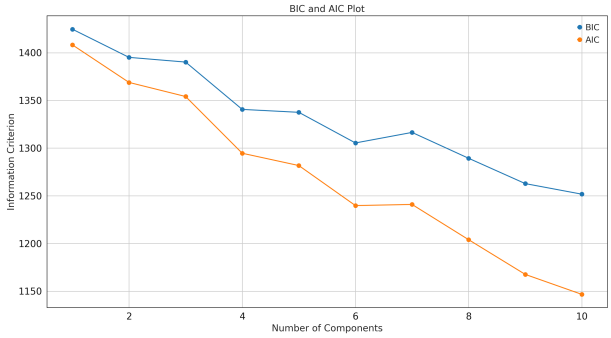

Figure 6: UMAP BIC and AIC score through with 10 components

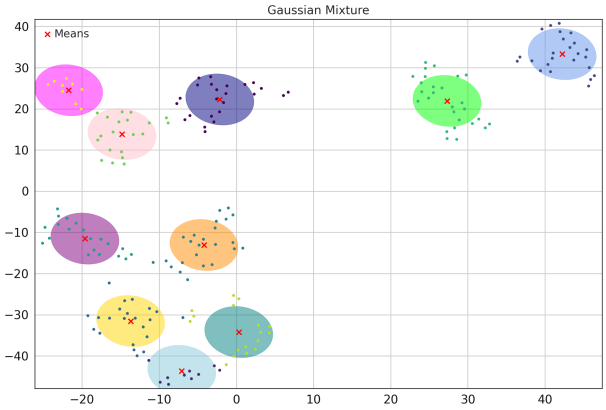

Figure 7: T-SNE clusters with 10 components

The criterion used to evaluate the best GMM parameters was the Bayesian Information Criterion (BIC) score. The BIC score penalizes models for their complexity, meaning that more complex models receive higher scores, making them less likely to be selected. The goal was to find the GMM configuration with the lowest BIC score, indicating a good balance between model complexity.

As anticipated, the performance of the GMM based on Principal Component Analysis (PCA) was not satisfactory4 in terms of the BIC score. This is primarily because PCA1 and PCA2 space did not exhibit distinct clusters, making it challenging for the GMM to identify meaningful patterns in the data.

On the other hand, t-SNE7 resulted in clear clusters. However, it was surpassed by UMAP in terms of the BIC score. Despite UMAP having a big BIC score6, it outperformed other dimensionality reduction methods by effectively visualizing the clusters. UMAP managed to capture and represent the underlying structures5 of the data more accurately, even with its higher complexity.

5 Visualization

I created two visualization functions for the GMM results. The first function12 generates ellipses to represent the covariances of the GMM clusters. Each

n_components	covariance_type	reg_covar	n_init	BIC _score	dim_red_method
5	spherical	0.001	10	7043.673104	PCA
10	tied	0.0001	1	1238.260093	UMAP
10	tied	0.0001	10	3173.414380	T-SNE

Table 1: Best parameters for GMM and BIC scores for every dimensionality reduction method

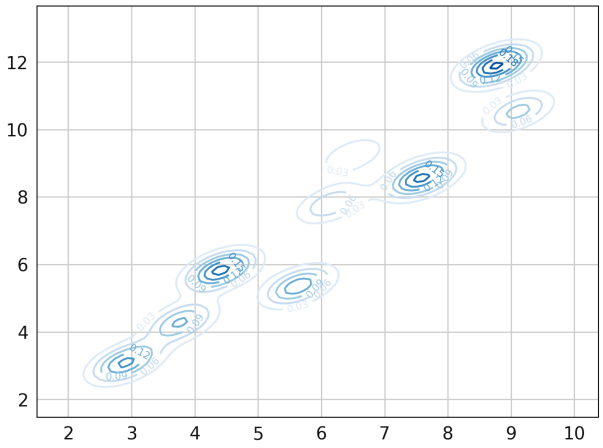

Figure 8: Contour plot

cluster is assigned a different color, and when clusters overlap, the colors are mixed to indicate the overlap. Additionally, there is an optional command to add contour lines to the distributions.

The second function focuses on visualizing the density contours8 of the GMM distributions in a more minimalistic manner. It also provides a 3D plot13 showing the joint distributions, allowing the viewer to select the desired angle for observing the plot.

6 Different Datasets

In general, the same set of parameters was tested on all the datasets, and the results were consistent across them. The UMAP dimensionality reduction method, when applied with these specific parameters, consistently outperformed all other methods in terms of the BIC score. The t-SNE method consistently obtained the second-best BIC score, while PCA consistently had the worst performance.

It is worth noting that the differences in the BIC scores between the datasets were not substantial.

If the reader decides to apply the analysis to a different dataset, it would be advisable to adjust the `Quality Control` parameters first (following suggestions that are in the code file), to ensure the best possible results.

7 Plots

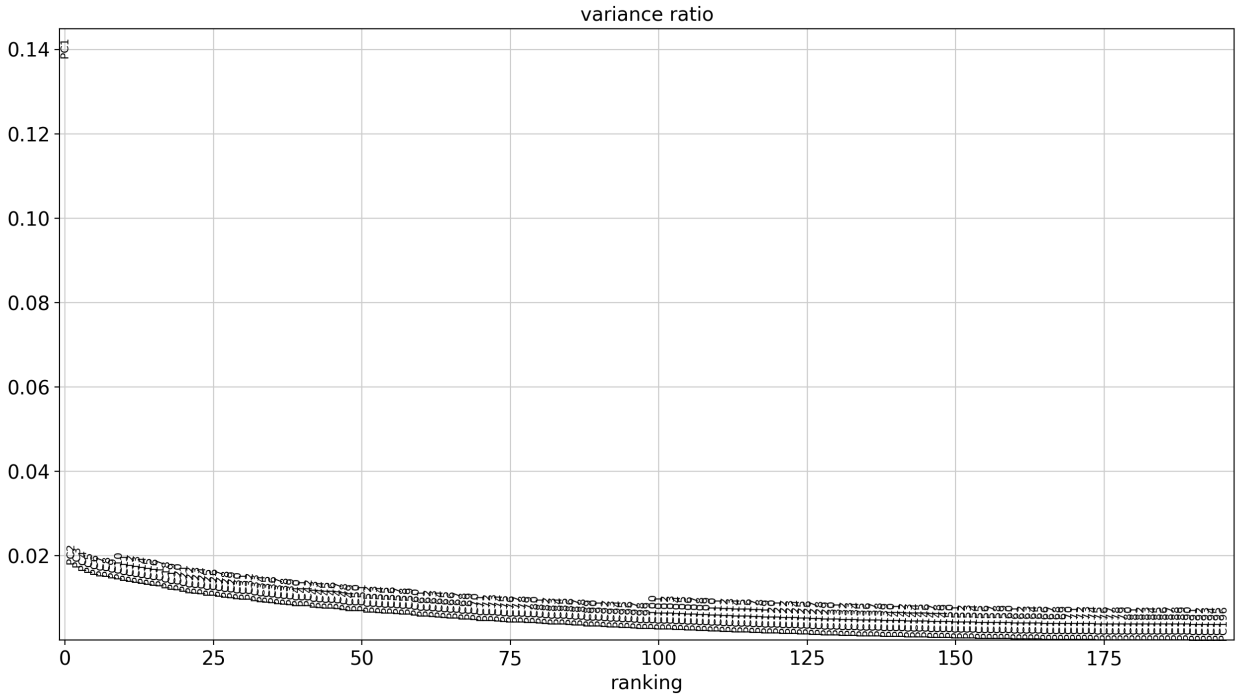

Figure 9: Scree plot with PCA components

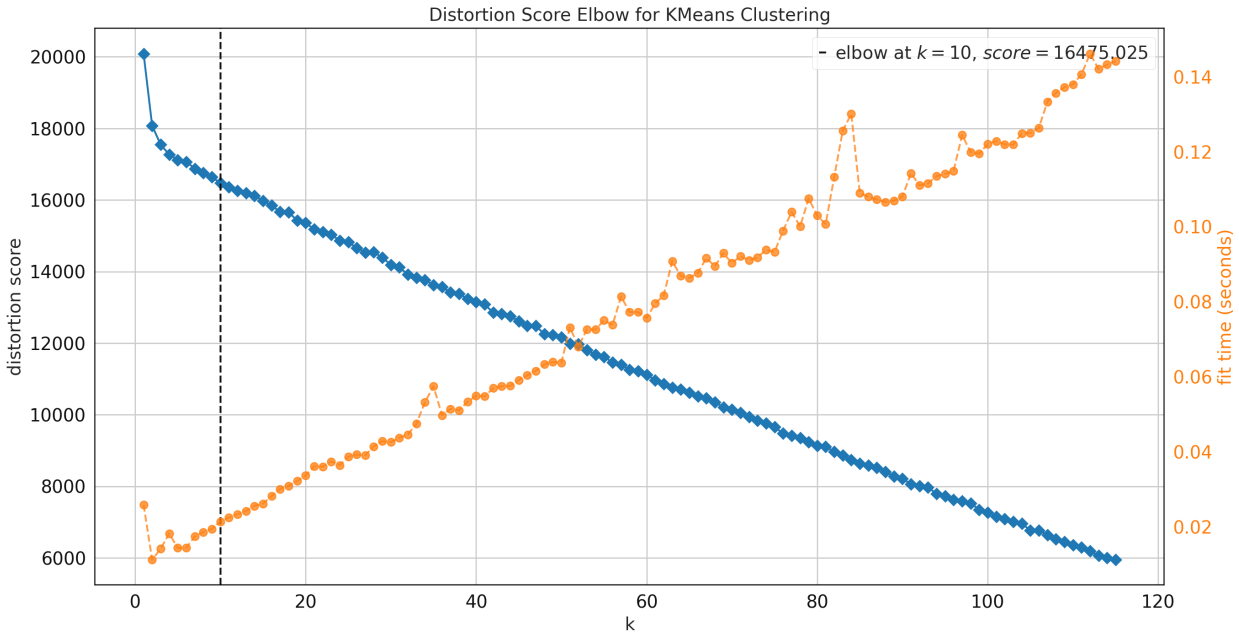

Figure 10: Elbow Rule

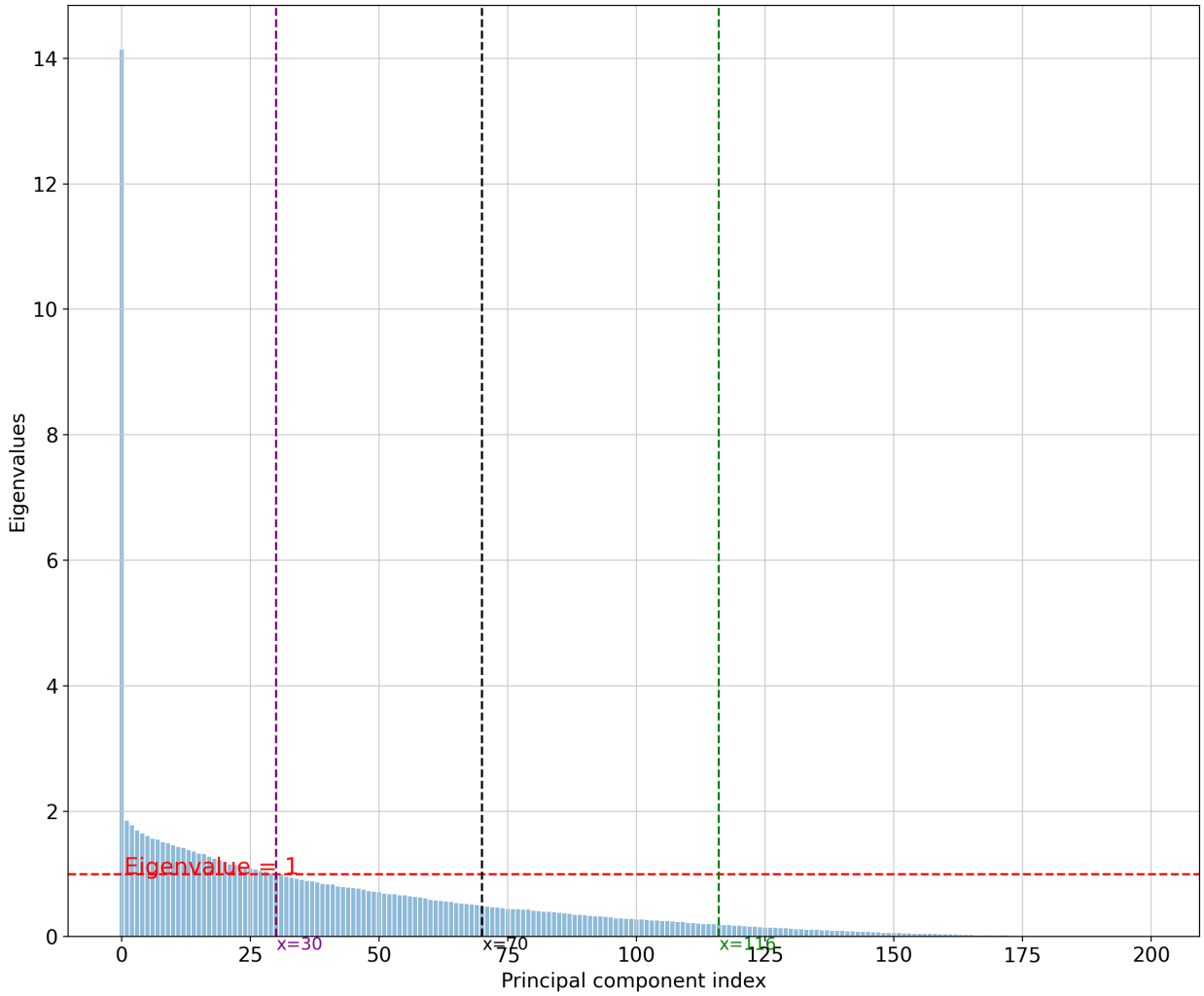

Figure 11: Based on the Kaiser rule, which suggests keeping only the principal components with eigenvalues greater than one, you have determined that the first 30 principal components should be retained for dimensionality reduction.

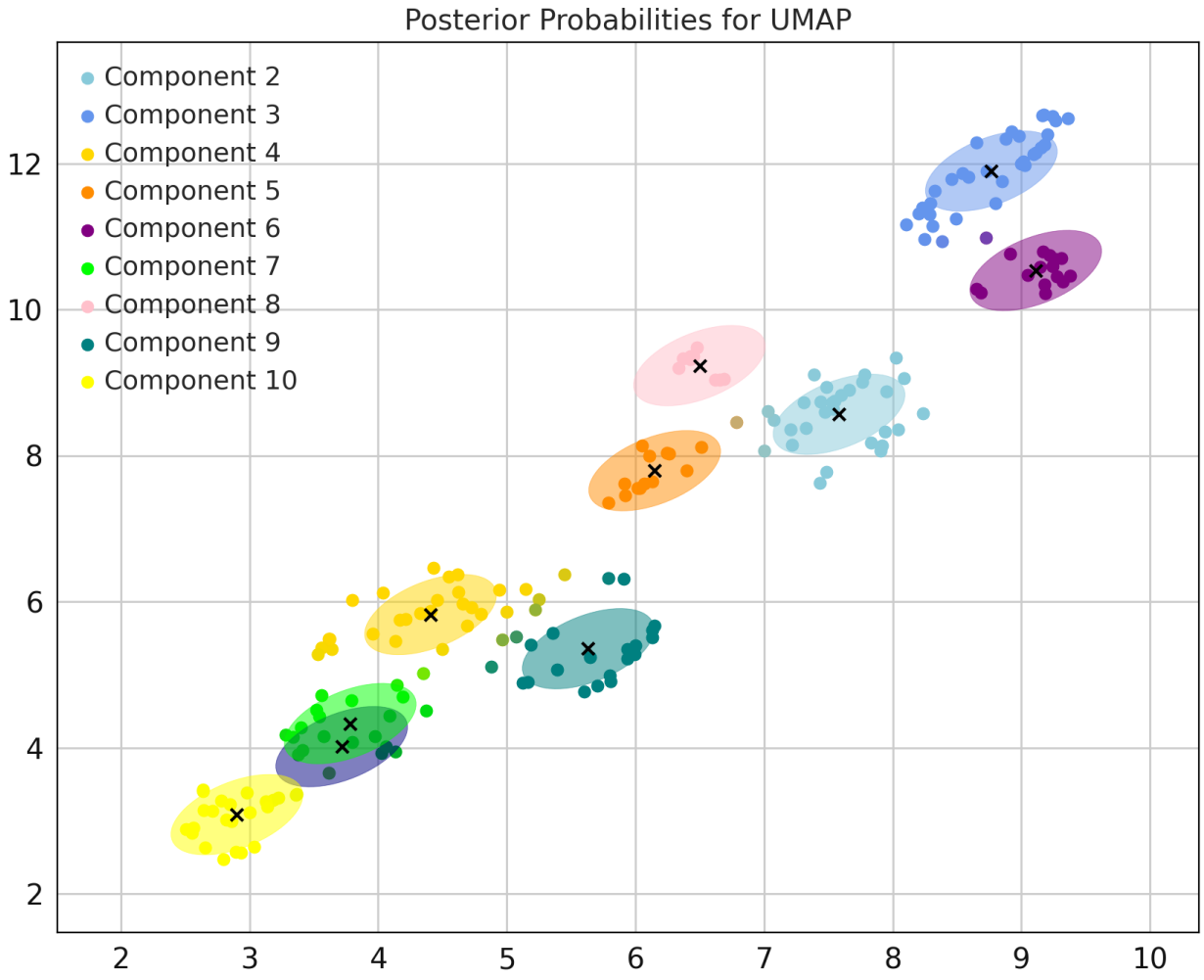

Figure 12: Mixed colors plot

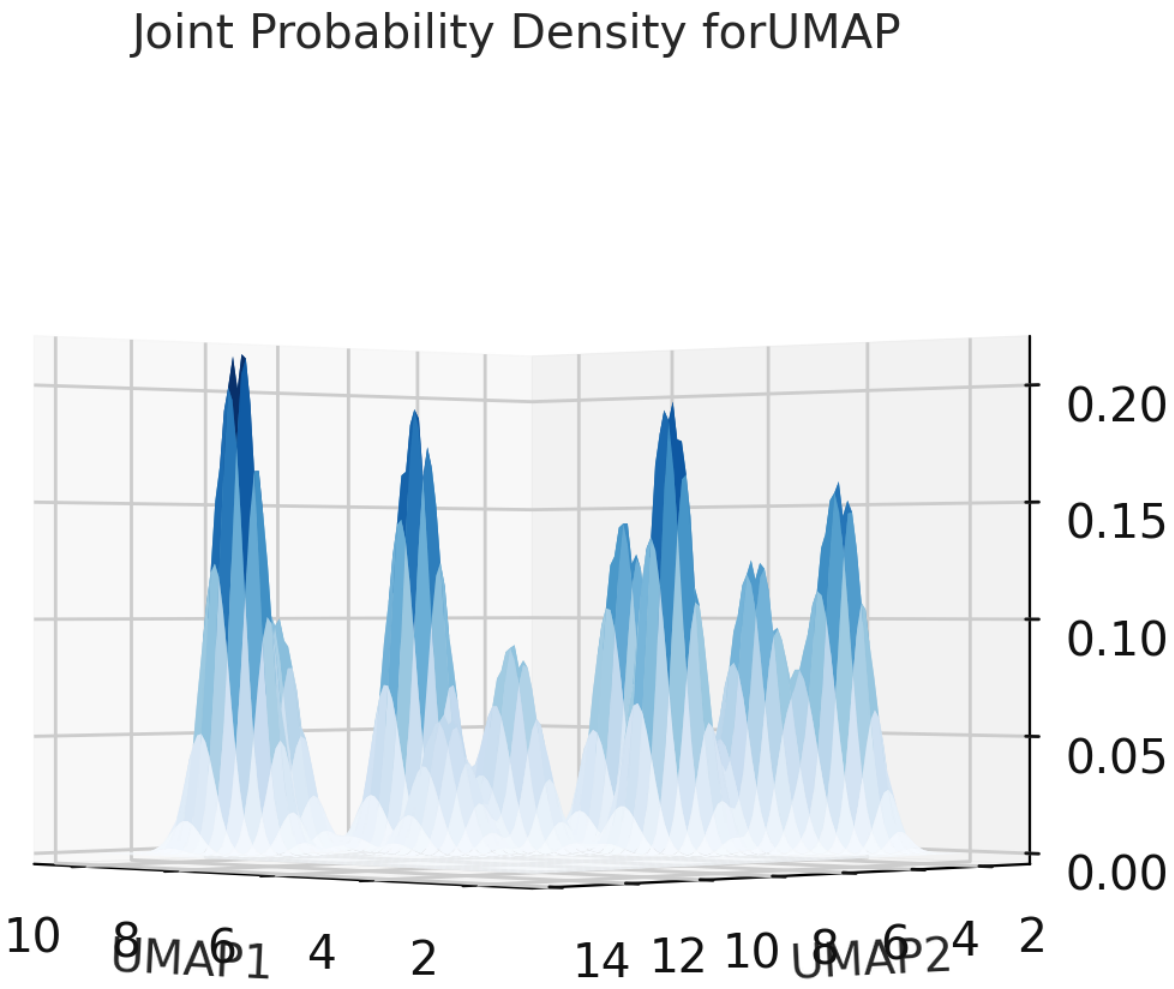

Figure 13: 3D plot