scGmix Machine Learning Pipeline Technical Report

Psallidas Kyriakos

[1]Department of Computer Science and Telecommunications, National and Kapodistrian University of Athens

Abstract

*This technical report presents the **scGmix** (single-cell Gaussian Mixture) pipeline, a tool written in Python and designed for intuitively discovering cell states from scRNA-seq datasets. The pipeline seamlessly integrates multiple functionalities, including data preprocessing with quality control and appropriate normalization, dimensionality reduction techniques such as principal component analysis (PCA), t-distributed stochastic neighbor embedding (t-SNE), or uniform manifold approximation and projection (UMAP), and cell clustering using Gaussian Mixture Models (GMMs). GMM clustering can be performed using either pre-clustering component means computed through the tools offered by **scGmix**, or automatically precomputed component means using the integrated optuna optimization library. While some components of the pipeline require further tuning, **scGmix** achieves a balanced approach between automated processes, user preferences, and interpretability, thus we believe it is a valuable tool for users who wish to identify cell states based on their specific requirements.*

Introduction

scRNA-seq technology has revolutionized our ability to understand the intricate heterogeneity of cell populations and identify distinct cell states. Gaussian Mixture Models (GMMs) are a natural fit for this goal since they excel at capturing normally distributed sub-populations within larger populations especially when considering the central limit theorem. However, the current landscape of user-friendly Python pipelines that seamlessly integrate GMM clustering with single-cell data analysis remains limited, with the most notable pipeline being **scGMAI**[1] which utilized a deep autoencoder to create a Gaussian mixture model for clustering scRNA-seq data. Aiming to enrich the available tools for single-cell GMMs in Python we proceed with showcasing **scGmix**.

Methodologies

Libraries & Pipeline overview

The pipeline revolves around three essential domains: Python-based handling of single-cell datasets, Gaussian mixture models, and hyperparameter optimization. Its primary objective is to seamlessly integrate these tools into a unified framework, effectively bridging them together.

During the evaluation of libraries to serve as the foundation for the pipeline, three specific candidates stood out. Among them, Scanpy[2] emerged as the most promising choice for single-cell data handling, offering a comprehensive range of functionalities and tools tailored specifically for analyzing single-cell data in Python. Notably, Scanpy leverages the capabilities of anndata [3], a robust package that efficiently manages annotated data matrices in both memory and disk. This makes it an ideal candidate for serving as the primary input for **scGMix**, allowing it to build upon the solid foundation provided by anndata.

Regarding Gaussian mixture models, sklearn[4] is the go-to library for various machine-learning tasks in Python. Its *GaussianMixture* class made it a natural choice for this pipeline. Finally, while sklearn also provides tools for hyperparameter optimization using random and grid search, we chose the optuna[5] hyperparameter optimization library for its Bayesian hyperparameter optimization capabilities.

The pipeline consists of four stages, each executed conveniently through a single method call on an instance of the **scGmix** class. The inputs of **scGmix** class are an scRNA-seq expression counts matrix in Anndata format, the chosen method for dimensionality reduction (which will serve as the foundation for downstream analysis), and finally a seed to ensure reproducibility of random processes within the pipeline instance. Once these parameters are provided, one can fit an optimal GMM and retrieve cell states with just the following four lines of code.

```
scGmix.preprocessing()
scGmix.dimreduction()
scGmix.mix()
scGmix.visualize()
```

Naturally, the default configuration is provided in this approach. However, it is worth noting that each method, particularly dimensionality reduction (dimreduction) and GMM fitting (mix), can be further customized to accommodate the specific requirements of the user. Detailed explanations regarding these customizations will be provided in the subsequent sections. Before delving into the finer details of each stage, it is recommended to first review the comprehensive high-level overview of the pipeline, as illustrated in (Figure 5). This will provide a broader perspective for those who wish to have a complete picture before diving into the details of each stage.

For the showcase of all the following stages, we utilize the synthetic data-set: **dataset1** as input in **scGmix**, available in the GitHub repository of the pipeline.

Preprocessing

Quality control & Normalization is the initial step and it involves annotating the given Anndata object by incorporating quality control metrics. These metrics include the count of cells expressing each gene, along with the total and average expression levels of each gene across all cells. Additionally the count of expressed genes and the overall expression level detected in each cell. The distributions of the last two measurements for the synthetic dataset are shown in Figure 6(a). As for cell filtering, we eliminate cells that deviate by more than 5 absolute deviations from the median in the aforementioned distributions.

$$MAD = median(|X_i - median(X)|)$$

By applying this method, cells that deviate significantly from the median in terms of both total gene expression and the number of detected genes can be identified. These cells are flagged for filtering as they are deemed to contain either very few genes with low expression or an exceptionally high number of genes with an intense expression, thereby lacking valuable information. Generally, it is recommended to employ moderately lenient thresholds, such as 5 median absolute deviations (MADs) to 3 MADs [6]. The user can change the default threshold of 5 by giving a new value to *mads_away* when calling the preprocessing method:

```
scGmix.preprocessing(mads_away=5)
```

For gene filtering, we employ a simple removal of genes with zero expression across all cells, since they can affect the next step. Normalization is carried out by first normalizing the total counts for all genes and log transforming the counts' matrix after adding the value of one to stabilize the variance as seen in (Figure 6(b)).

Feature selection is optional, the user can utilize it to keep only highly variable by setting thresholds for the minimum and maximum mean expression, as well as the minimum gene dispersion (variation in gene expression) effectively filtering out genes that are not highly variable. The default setting for feature selection is off since for small datasets it may introduce bias, but the user can easily enable it and tune it with:

```
scGmix.preprocessing(mads_away = 5,
feature_selection = True,
min_mean=0.0125, max_mean=3, min_disp=0.5)
```

Dimensionality reduction

The method of choice for dimensionality reduction is inherited from the input class *method* variable. However regardless of the selected method, a Principal Component Analysis (PCA) representation is always computed. This PCA representation serves as the downstream representation if no other method is chosen. Otherwise, it serves as the initialization for either the t-distributed stochastic neighbor embedding (t-SNE) or the uniform manifold approximation and projection (UMAP) of the dataset.

The optimal number of principal components can be determined with three methods:

```
scGmix.dimreduction(
pc_selection_method = "screeplot")
```

As shown above, the default method is the "Scree plot knee," as depicted in (Figure 7(a), 8(b)), using the synthetic dataset as an example. The scree plot illustrates the number of principal components on the x-axis and their respective eigenvalues on the y-axis. By identifying the "elbow" point on the plot, we can determine the principal component at which the most significant change in explained variance curvature occurs. This approach is based on the research paper titled "Finding a 'Kneedle' in a Haystack: Detecting Knee Points in System Behavior"[7], as we utilize the Python implementation of the algorithm provided by the kneed library. It is important to acknowledge that in certain cases, this approach may underestimate the number of optimal components, particularly when PCA is used as the initialization method for UMAP and t-SNE. To address this issue, we have implemented two more selection methods.

A second popular option we have implemented is the cumulative variance threshold (Figure 8(c)) via:

```
scGmix.dimreduction(
pc_selection_method = "variance",
threshold= 90)
```

Typically, this threshold falls within the range of 70% to 95%. We have defined a default of 90%, but the user can change it as they see fit and remove principal components that pass the threshold. The final option is Kaiser's rule (Figure 8(d)), by setting *pc_selection_method = "kaiser"*. With Kaiser's rule only principal components with eigenvalues greater than 1 are retained. The rationale behind the Kaiser criterion is that any principal component with a variance below 1 carries less information than the original features, and therefore, it is not beneficial to retain them.

The parameters for TSNE, UMAP dimensionality reduction methods are pre-defined in the case of t-SNE following guidelines from the paper "The art of using t-SNE for single-cell transcriptomics"[8]. For perplexity, which defines attention between local and global aspects of the data, similar to the k neighbors we consider for each node in a KNN graph we utilize the following cases:

$$\text{perplexity} = \begin{cases} 30 + \text{cells}/100, & \text{for cells}/100 \gg 30 \\ 30, & \textit{else} \end{cases}$$

As for the learning rate that controls the step size during the t- SNE optimization process, we choose its values based on:

$$l\text{learning rate} = \begin{cases} \text{cells}/12, & \text{for cells}/12 > 200 \\ 200, & \text{for cells} \leq 200 \end{cases}$$

While full automation of every component might be desirable in some cases, we believe that the value lies in the balance between automation and user control, for this reason, we have chosen to provide users with the flexibility to define the nearest neighbors and minimum distance parameters for UMAP, as it is a relatively recent technique but very powerful. This decision allows users to have more control over the analysis process and enables them to observe the results according to their preferences. As a reference Increasing the minimum distance results in clusters that are loosely defined, whereas decreasing it has the opposite effect. Similarly, increasing the number of neighbors, akin to increasing the perplexity in t-SNE, intuitively preserves more information about global structures while decreasing them has the opposite effect. The PCA, t-SNE, and UMAP representation for the example synthetic dataset is illustrated in (Figure 14).

Clustering

Having produced low dimensional representations of our original dataset via PCA, UMAP or TSNE. The goal of this stage is to identify the optimal number of cell states, by identifying the optimal number of components and covariance type for a Gaussian mixture model that describes the data. Our solution involves two main approaches for the user to choose on either the PCA, UMAP or TSNE representations:

Option 1. We Enable users to leverage precomputed component means with their own outside of the pipeline preferred method, in this way, the pipeline can be used for optimizing the covariance structure of the GMM through the utilization of BIC or AIC.

```
scGmix.mix(
enable_preclustering = True,
user_means = input means)
```

Option 2. We allow the automated identification of the ideal number of components and component means for the GMM by employing either automated optimized spectral pre-clustering based on the maximization of the silhouette score, or user-parameterized Leiden clustering for community detection via the *resoluton* parameter. Finding the means with pre-clustering serves the purpose of identifying cell clusters that are often not picked up by Gaussian Mixture Models (GMM) due to their initialization, which may miss irregularly shaped clusters. The results of the example synthetic dataset for this options are illustrated in (Figures 9(a), 9(b)).

```
scGmix.mix(
enable_preclustering = True,
preclustering_method = "spectral" or "leiden"
leiden_resolution = 0.5)
```

Option 3: This approach automatically optimizes the number of components and covariance type by utilizing BIC or AIC scores without precomputed means. The method keeps track of BIC values for different component sizes and determines the number of components that results in the largest drop in BIC/AIC score compared to the previous component. This ensures a suitable trade-off between bias and variance (Figures 9(c), 9(d)).

```
scGmix.mix(enable_preclustering = False)
```

The equation of BIC and AIC are:

$$BIC = -2 \times log(L) + k \times log(n)$$

$$AIC = -2 \times log(L) + 2 \times k$$

The log-likelihood takes into account how well the GMM explains the observed data. And both critérions penalize complex models by including a term that scales with the number of model parameters (k). However, BIC scales the parameters by the *log* of the number of observations in the dataset n, for this reason, it selects less complex models than AIC which tends to overfit the training set, since the model parameters k are multiplied by a scalar $= 2$. Subsequently, BIC is the default metric for the optimization to minimize in **scGmix**.

Visualization

The final step in the scGmix pipeline involves retrieving and visualizing the results. After the clustering stage, where the GMM model is fitted, we obtain a Gaussian mixture consisting of c components. This mixture is represented by the equation:

$$P(\vec{x}) = \sum_{i=1}^{k} w_i \cdot \mathcal{N}(\vec{x}|\vec{\mu}_i, \Sigma_i)$$

To analyze the results, we can retrieve the posterior probability of a cell vector $\vec{x}$ being generated by each of the components. This probability is denoted as:

$$p(\vec{x}|c_i)$$

Additionally, we can consider the joint distribution by incorporating the respective weight of each component, resulting in the expression:

$$p(\vec{x}, c_i) = p(\vec{x}|c_i)p(c_i) = p(\vec{x}|c_i)w_i$$

Finally summing the joint probabilities we can reach the marginal $P(\vec{x})$. The user can retrieve the cluster labels inferred, as well as the posteriors, joint and marginal probabilities by calling:

```
scGmix.visualize(membership_threshold=0.90,
cmap = "bwr")
```

The *membership_threshold* parameter establishes a threshold for the posterior probability required for a cell to be considered part of a cell state. Cells below this threshold are identified as transitioning between states, with a default setting of 90%. In addition to the previously described results, **scGmix** provides informative visual representations. We applied the *visualize* step to the fully automated fitted Gaussian Mixture Model (GMM) from the previous step and present the results. (Figure 11) illustrates the Mixture components, cluster membership, and posterior probabilities with annotated cell IDs, providing a comprehensive view. Furthermore, **scGmix** generates a similar graph for the marginal probabilities, as shown in (Figure 12). For a more minimal representation, cell trajectory heatmaps displaying joint and posterior cell state probabilities are also included in the results (Figure 10(a), 10(b)). Finally **scGmix** also returns the cell counts per state both in the hard clustered sense and based on the probability threshold (Figure 13).

Results & Discussion

After offering a detailed explanation of the methodology underlying **scGmix** and using a synthetic dataset for illustrative purposes, we have set aside an additional four distinct synthetic datasets to evaluate our approach with various parameters and methods. This will enable us to assess the strengths of **scGmix** as well as areas where it may have limitations. As mentioned before all the datasets are available in . the GitHub repository of the pipeline. We will present the cell state posterior scatter-plot for each dataset, as we consider it to be the most informative and independent representation.

Synthetic dataset 2 pipeline example

For the synthetic dataset two, the default scree-plot knee component selection method was utilized and retained 4 out of the 100 assigned PCs. UMAP was selected as the dimensionality reduction method with its default parameters: nearest neighbors = 15 and minimum distance = 0.5. At the clustering stage preclustering was enabled with the Leiden method and a resolution = 0.2, which resulted in the selection of three clusters. The resulting model achieved a BIC score of 1398 after 100 optuna trials and a GMM with diagonal covariance type. Upon analyzing the cell state posterior scatterplot, it becomes apparent that a significant proportion of cells undergo transitions between state 1 and state 2 (Figure 14(a)). In this scenario, all pipeline components work as intended. As its evident the clusters inferred in the UMAP space retain their importance in the PCA space (Figure 1).

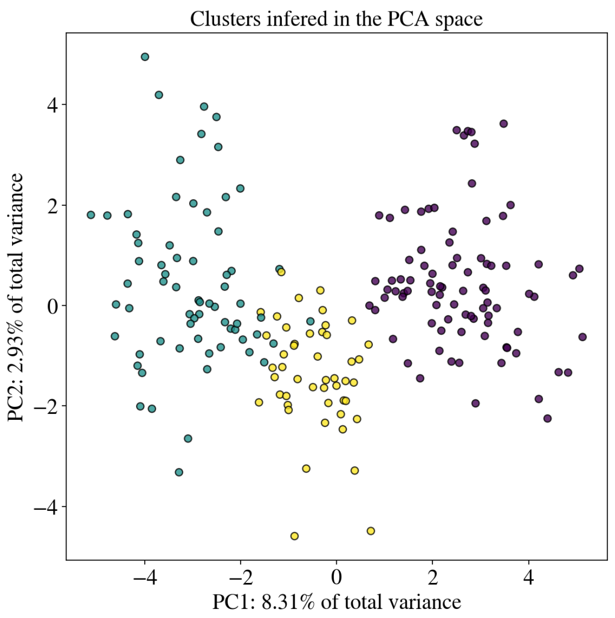


Figure 1: *Representation of the clusters infered in the PCA space for the synthetic dataset 2 pipeline*

Synthetic dataset 3 pipeline example

In synthetic dataset three, we employed the default PCA as a dimensionality reduction technique and used the scree plot knee method to select the optimal number of components. Out of the original 100 components, three were retained. Preclustering was disabled during the clustering stage, and the number of components for the GMM model to be optimized by optuna was chosen based on the largest drop in BIC score compared to the previous component. However, we have discovered a potential weakness in this method. Our goal of enforcing parsimonious models, while achieved can become too strict. When the next component has a slightly smaller drop in BIC score compared to the previous one, it fails to recognize it as the optimal choice, a fact which is evident in (Figure (2) where the optimal components arguably should be four. After optimization, the resulting GMM has two components and a BIC score of 2456 with full covariance matrices, the cell posteriors are illustrated in (Figure 14(b)).

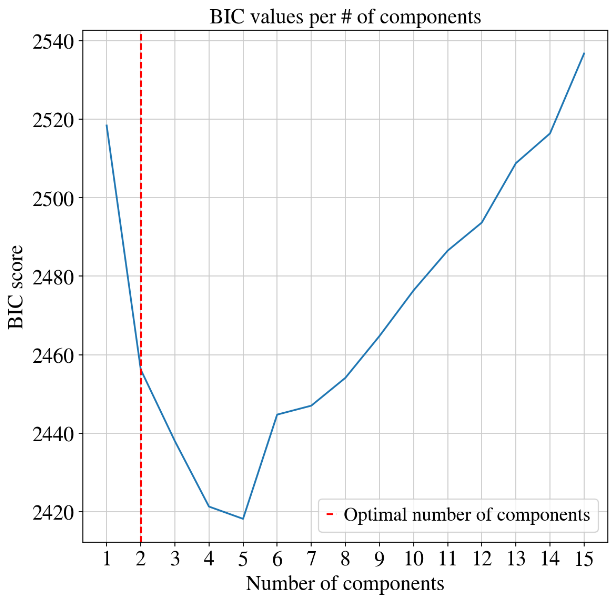


Figure 2: *BIC graph and knee point detection based on the largest drop for synthetic dataset 3*

Synthetic dataset 4 pipeline example

For the fourth synthetic dataset, we utilized the TSNE dimensionality reduction technique. To initialize it, we retained 125 principal components, which accounted for 90% of the total 180 PCs. Our goal was to maximize the silhouette score, so we employed Automated Spectral clustering for clustering. The optimized model achieved a BIC score of 4344, using a spherical covariance type, as shown in Figure 14(c). Upon investigating the TSNE inferred clusters in the first two principal components, depicted in Figure 3, as well as the TSNE representation, it becomes apparent that retaining 90% of the variance threshold for the principal components is excessive. This approach places too much emphasis on the global perspective of TSNE and enforces our selection of the knee method as the default.

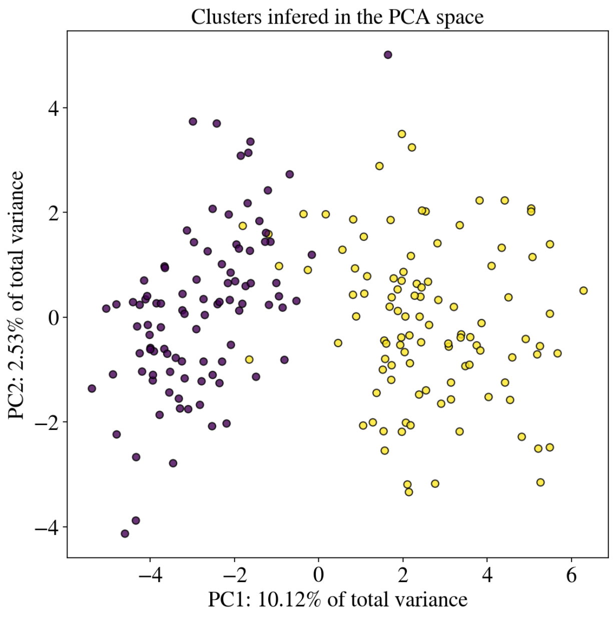


Figure 3: *Representation of the clusters inferred in the PCA space for synthetic dataset 3*

Synthetic dataset 5 pipeline example

In the final data set we utilize the default configuration, as we did for dataset 3. The curvature of the BIC to number of components graph (Figure 4) did not present the error-prone case of dataset 3 and thus 6 components were identified. After a 100 trials of hyperparameter tuning the resulting GMM had a BIC score of 2586 and spherical covariance type showcased in (Figure 14(d)) representing a large number of transitioning cells between multiple states.

Conclusions

ScGmix offers an efficient solution for automating tedious tasks, particularly parameter tuning. It strikes a balance by providing users with a significant level of control, allowing customization of UMAP and Leiden pre-clustering parameters. By seamlessly connecting single-cell datasets and Gaussian mixture models in Python, **scGmix** successfully achieves its goal. It is capable of handling any single-cell dataset and generate satisfactory results when integrated into the appropriate pipeline workflow with just four lines of code. Our recommendation is to start with

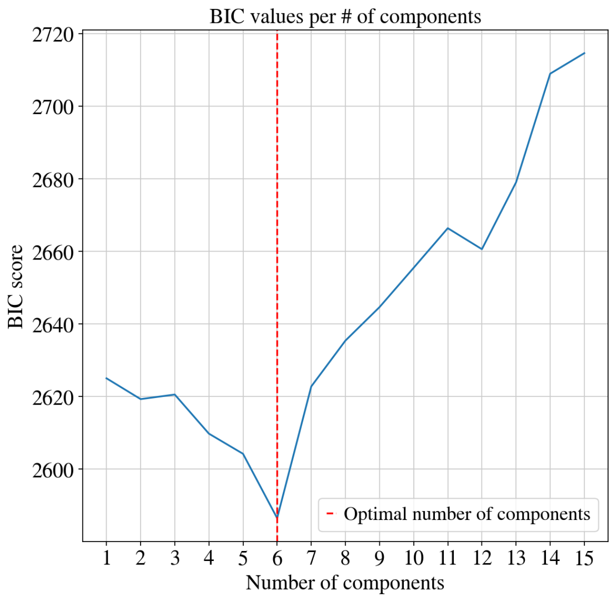


Figure 4: *BIC graph and knee point detection based on the largest drop for synthetic dataset 5*

the default approach, which utilizes PCA for dimensionality reduction, the scree plot method for optimal component selection, and automated component and parameter tuning for the GMM model. Once the results are reviewed, users can delve into the advanced pipeline workflow. This involves selecting UMAP as the dimensionality reduction technique, enabling preclustering with Leiden clustering, and fine-tuning the parameters of both UMAP and Leiden to achieve optimal outcomes. However, it is important to note that one possible flaw of **scGmix** lies in its selection of the optimal number of components when the pipeline is set to its fully automated mode. Our method attempts to enforce parsimonious modeling, but it proves too strict in specific cases related to the curvature of the BIC/AIC score to the component graph as seen in (Figure 2). Therefore, this aspect provides an area for possible improvements.

References

[1] *scGMAI: scGMAI: a Gaussian mixture model for clustering single-cell RNA-seq data based on deep autoencoder*. en.

[2] F Alexander Wolf, Philipp Angerer, and Fabian J Theis. "SCANPY: large-scale single-cell gene expression data analysis". In: *Genome biology* 19 (2018), pp. 1–5.

[3] Isaac Virshup et al. "anndata: Annotated data". In: *BioRxiv* (2021), pp. 2021–12.

[4] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[5] Takuya Akiba et al. "Optuna: A next-generation hyperparameter optimization framework". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 2623–2631.

[6] Mark D Robinson Robinson et al. "PipeComp, a general framework for the evaluation of computational pipelines, reveals performant single cell RNA-seq preprocessing tools". In: (2020).

[7] Ville Satopaa et al. "Finding a" kneedle" in a haystack: Detecting knee points in system behavior". In: *2011 31st international conference on distributed computing systems workshops*. IEEE. 2011, pp. 166–171.

[8] Dmitry Kobak and Philipp Berens. "The art of using t-SNE for single-cell transcriptomics". In: *Nature communications* 10.1 (2019), p. 5416.

Figures

The figures referenced in the methods & results sections of the report are showcased on the following pages: 7 – 15.

Code Availability

The code to reproduce the results of the paper and/or extend the base pipeline is available in this GitHub repository.

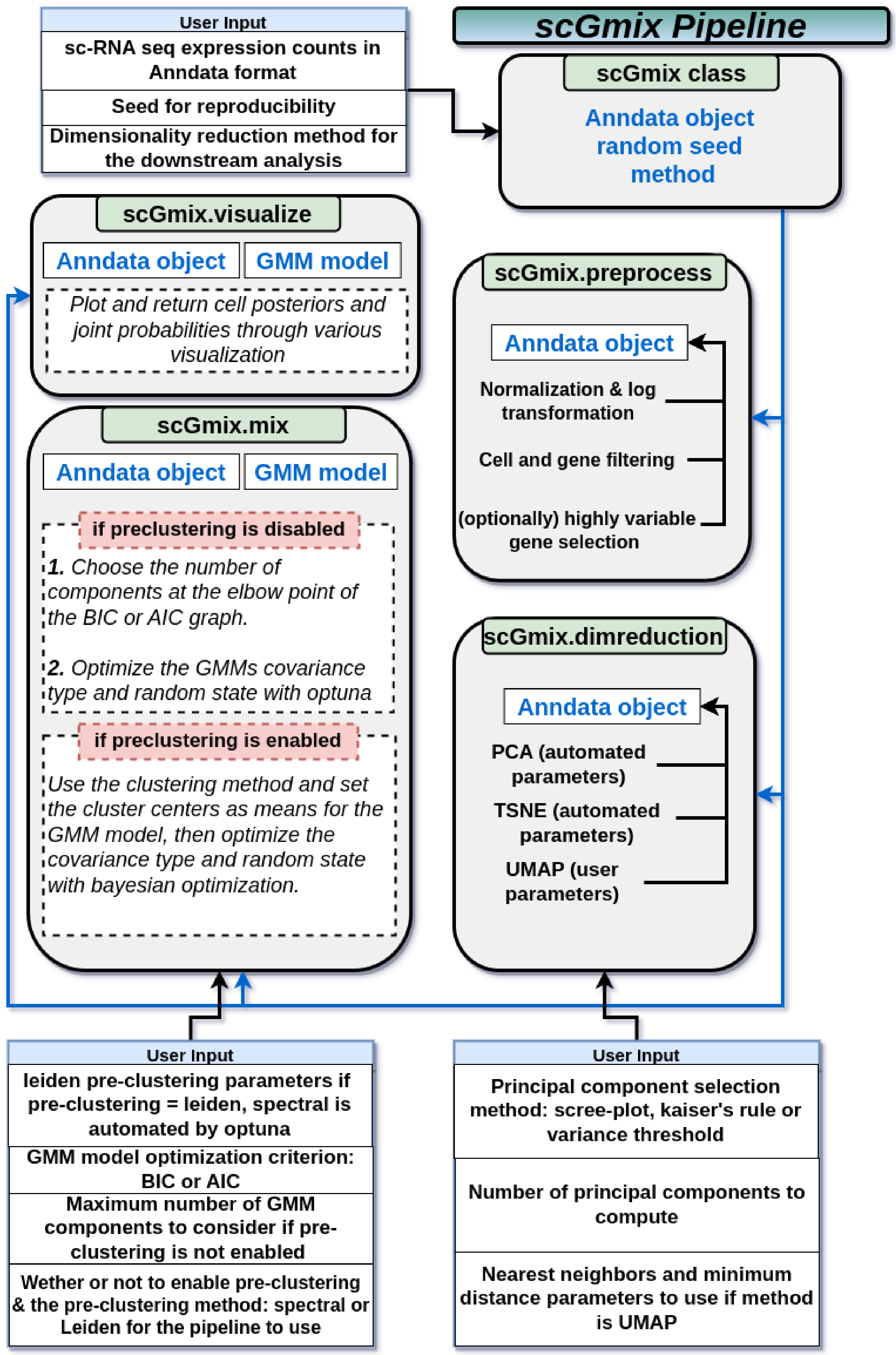
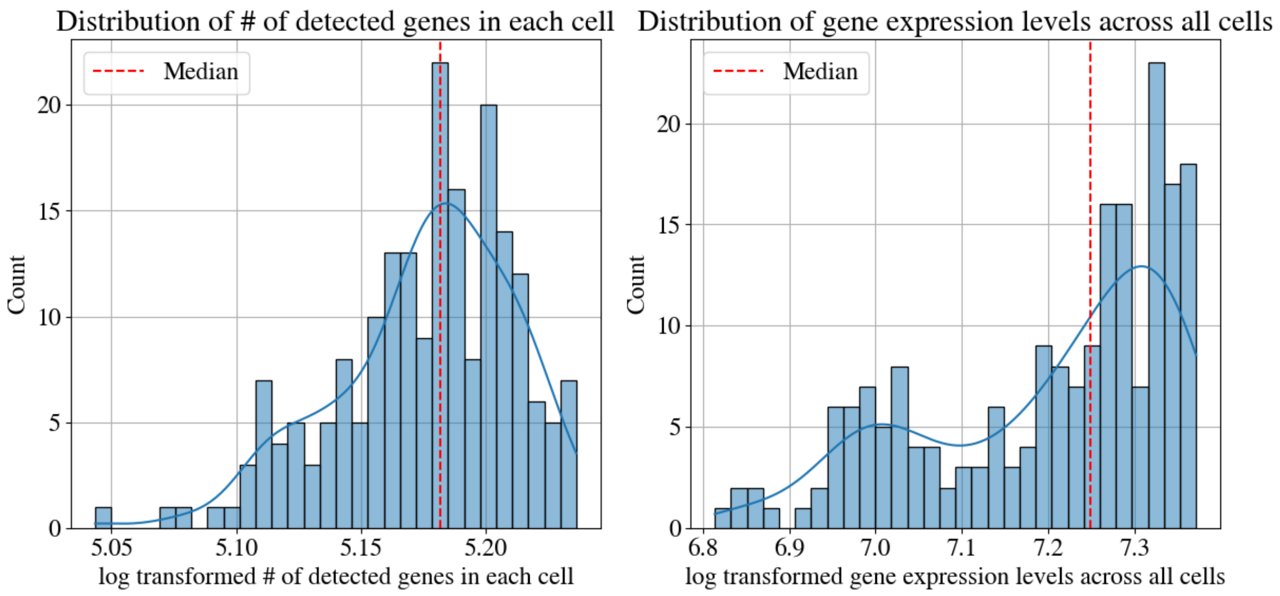
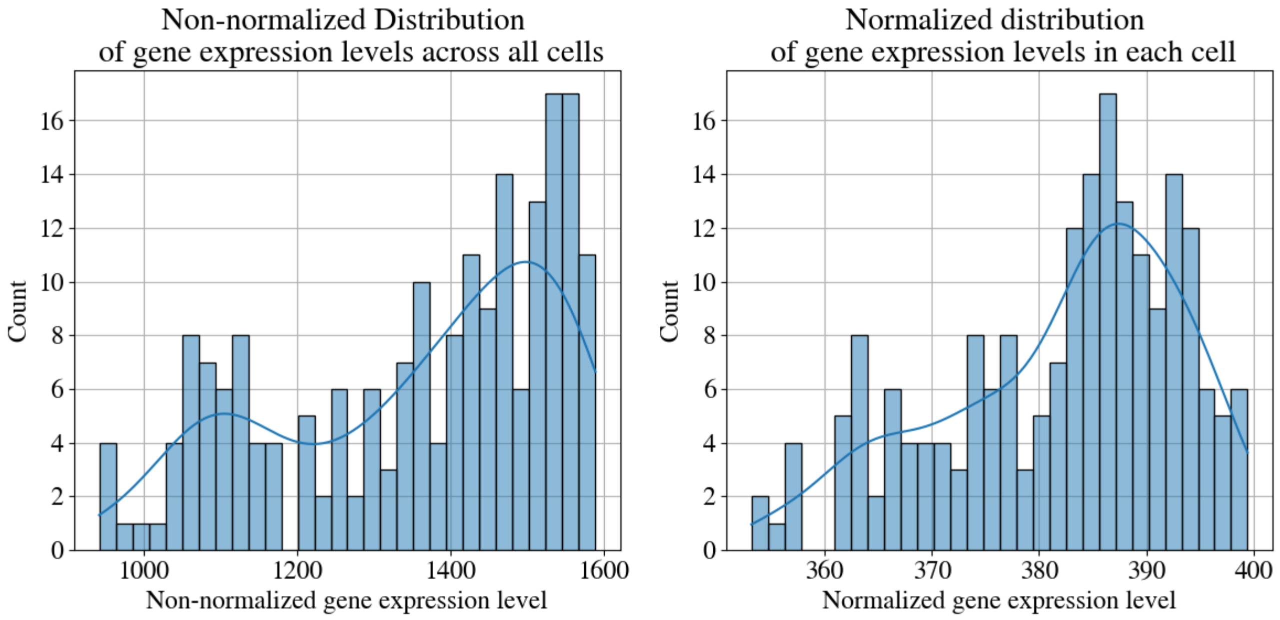

User Input
sc-RNA seq expression counts in Anndata format
Seed for reproducibility
Dimensionality reduction method for the downstream analysis
scGmix Pipeline
scGmix class
Anndata object
random seed
method
scGmix.visualize
Anndata object
GMM model
Plot and return cell posteriors and joint probabilities through various visualization
scGmix.preprocess
Anndata object
Normalization & log transformation
Cell and gene filtering
(optionally) highly variable gene selection
scGmix.mix
Anndata object
GMM model
if preclustering is disabled
1. Choose the number of components at the elbow point of the BIC or AIC graph.
2. Optimize the GMMs covariance type and random state with optuna
if preclustering is enabled
Use the clustering method and set the cluster centers as means for the GMM model, then optimize the covariance type and random state with bayesian optimization.
scGmix.dimreduction
Anndata object
PCA (automated parameters)
TSNE (automated parameters)
UMAP (user parameters)
User Input
leiden pre-clustering parameters if pre-clustering = leiden, spectral is automated by optuna
GMM model optimization criterion: BIC or AIC
Maximum number of GMM components to consider if pre-clustering is not enabled
Wether or not to enable pre-clustering & the pre-clustering method: spectral or Leiden for the pipeline to use
User Input
Principal component selection method: scree-plot, kaiser's rule or variance threshold
Number of principal components to compute
Nearest neighbors and minimum distance parameters to use if method is UMAP

Figure 5: *scGmix high-level overview*

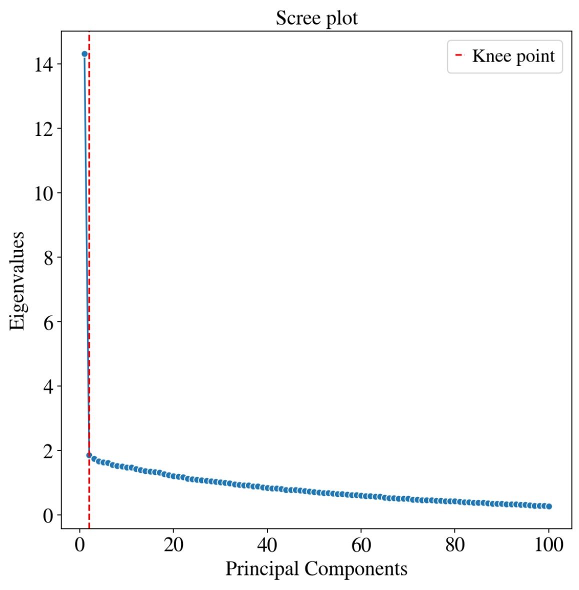


(a) example synthetic dataset QC distributions

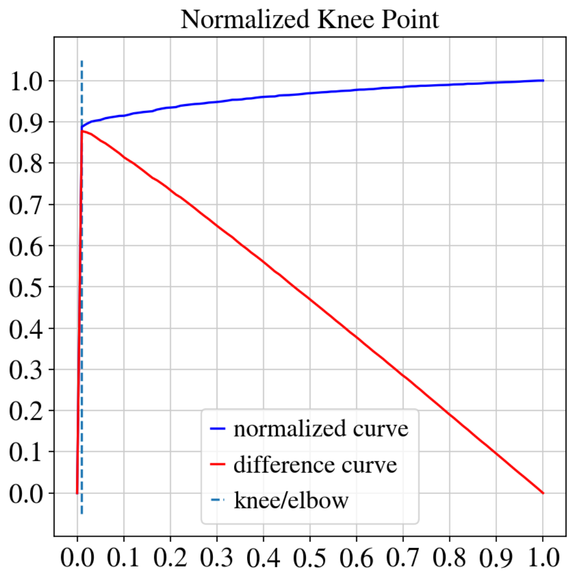


(b) example synthetic dataset normalization & log-transformation effect

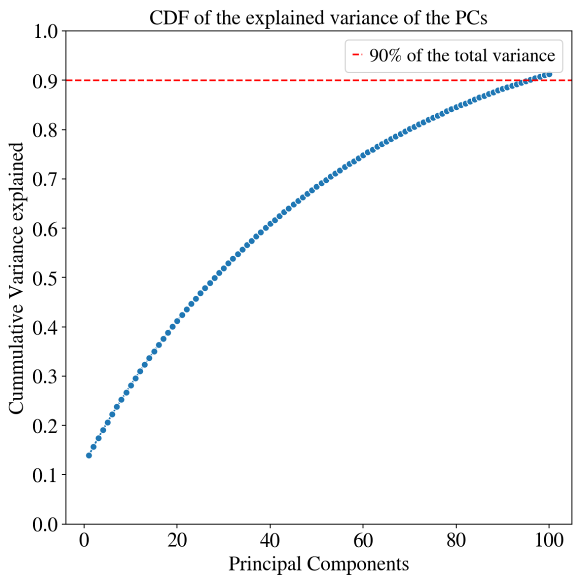
Figure 6: *QC metric distributions in the preprocessing stage for the example synthetic dataset*



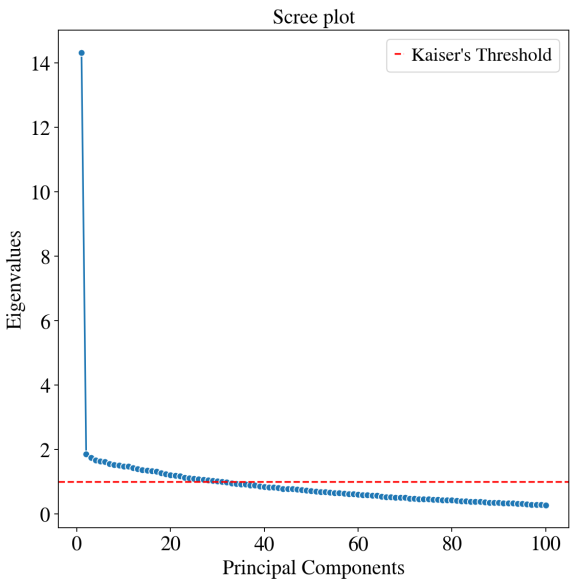
(a) Scree plot knee point method



(b) kneed algorithm result for scree plot knee point method

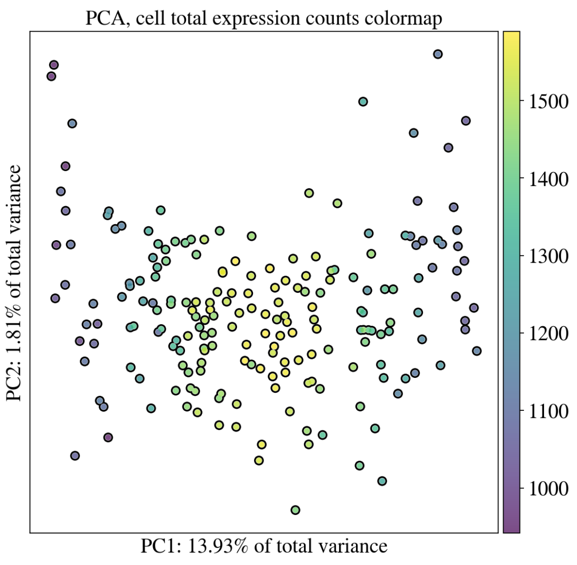


(c) Cumulative variance threshold method, set to 90% of total variance

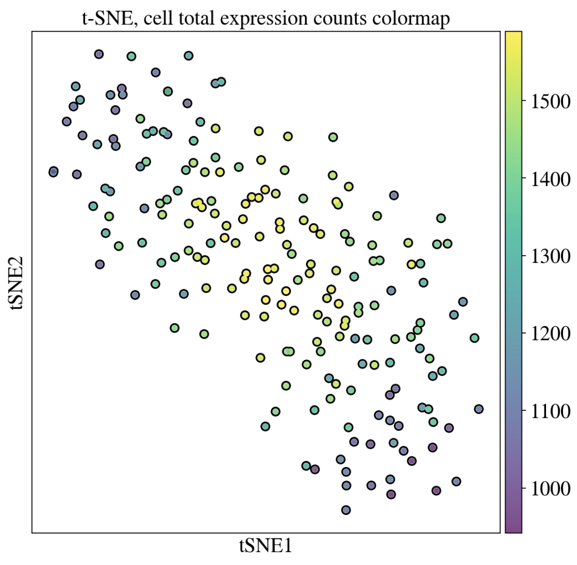


(d) Kaiser's rule method

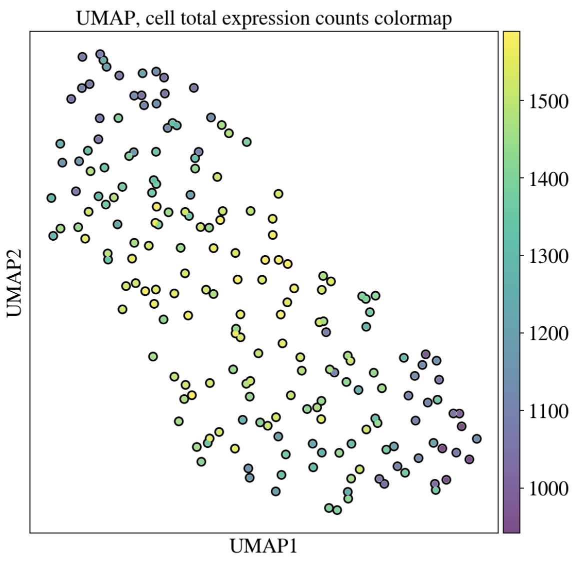
Figure 7: *scGmix optimal principal component selection methods & synthetic dataset results*



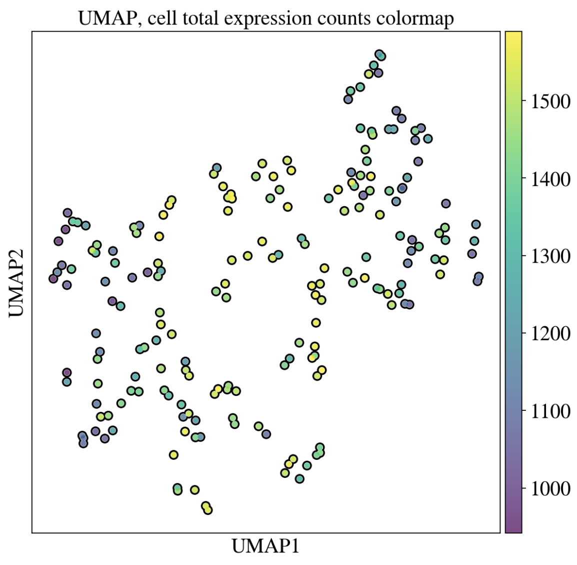
(a) PCA representation of the example synthetic dataset



(b) TSNE representation of the example synthetic dataset, kaiser's rule method for initializing PCA.

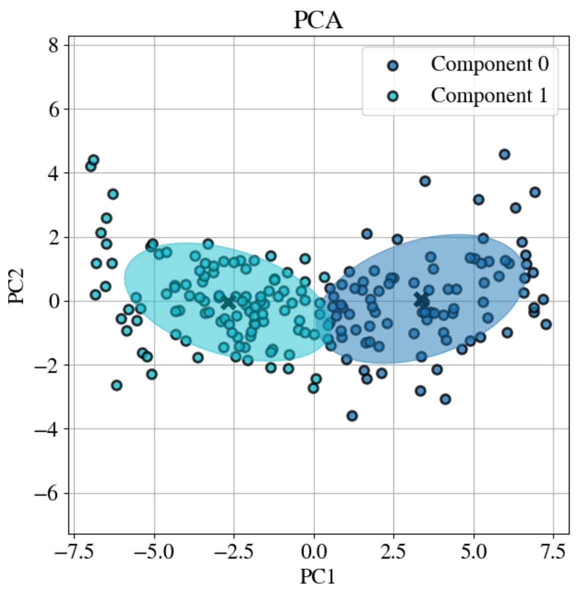


(c) UMAP representation of the example synthetic dataset, kaiser's rule method for initializing PCA,
nearest neighbors = 10, min dist = 0.1

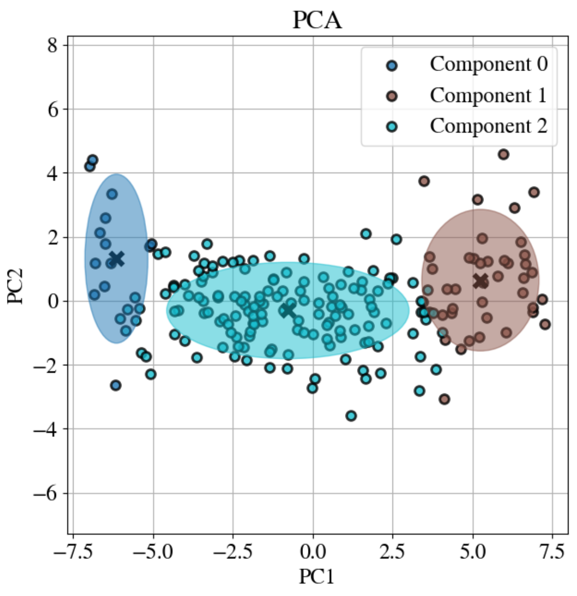


(d) UMAP representation of the example synthetic dataset, kaiser's rule method for initializing PCA,
nearest neighbors = 3, min dist = 0.1

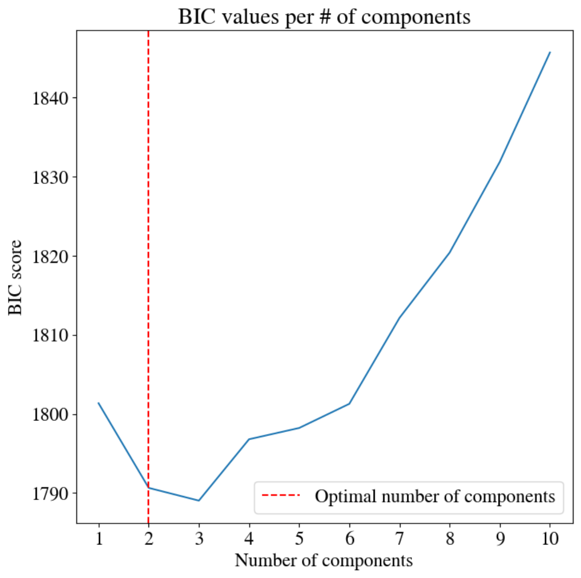
Figure 8: *scGmix dimensionality reduction results for the example synthetic dataset*



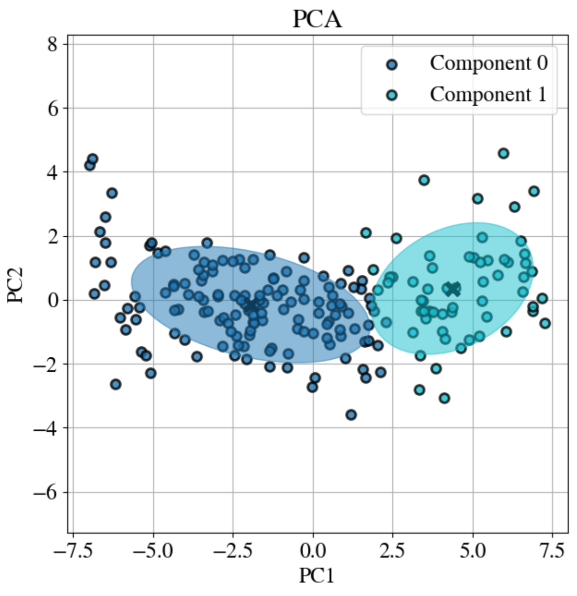
(a) Option 2.Pipeline automated spectral pre-clustering



(b) Option 2. Pipeline Leiden pre-clustering with resolution = 0.9

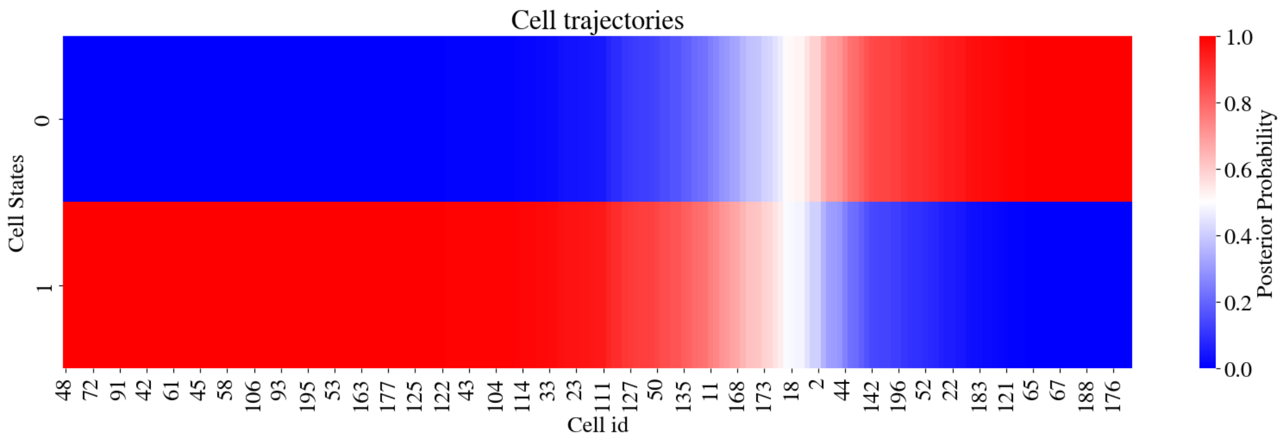


(c) Option 3. Component selection based on the largest drop in BIC value from the previous component

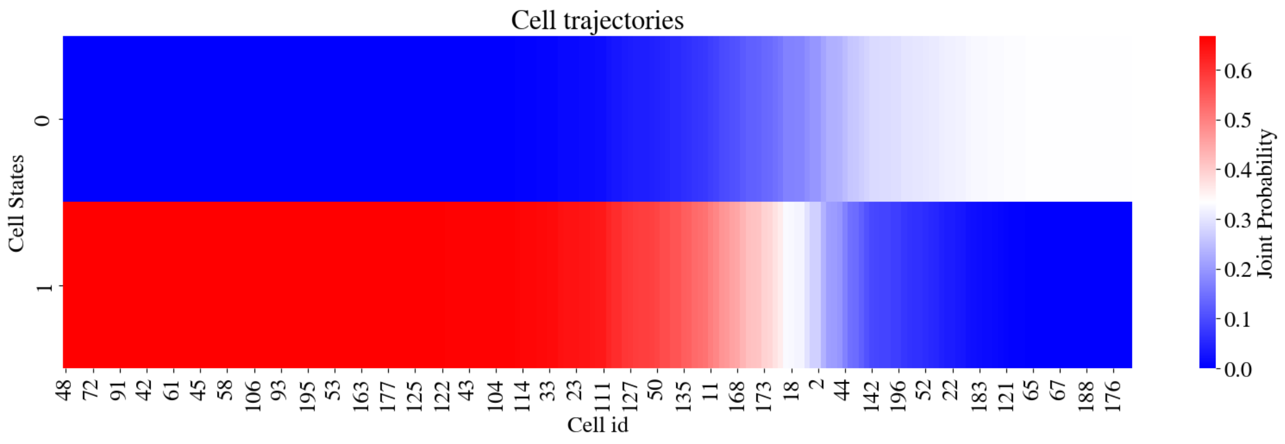


(d) Option 3. Fully automated optimization, without pre-clustering result

Figure 9: *scGmix clustering results with pre-clustering and fully automated methods on the synthetic dataset*



(a) Cell state posterior probabilities heatmap



(b) Cell state joint probabilities heatmap

Figure 10: *heatmap representation of cell trajectories*

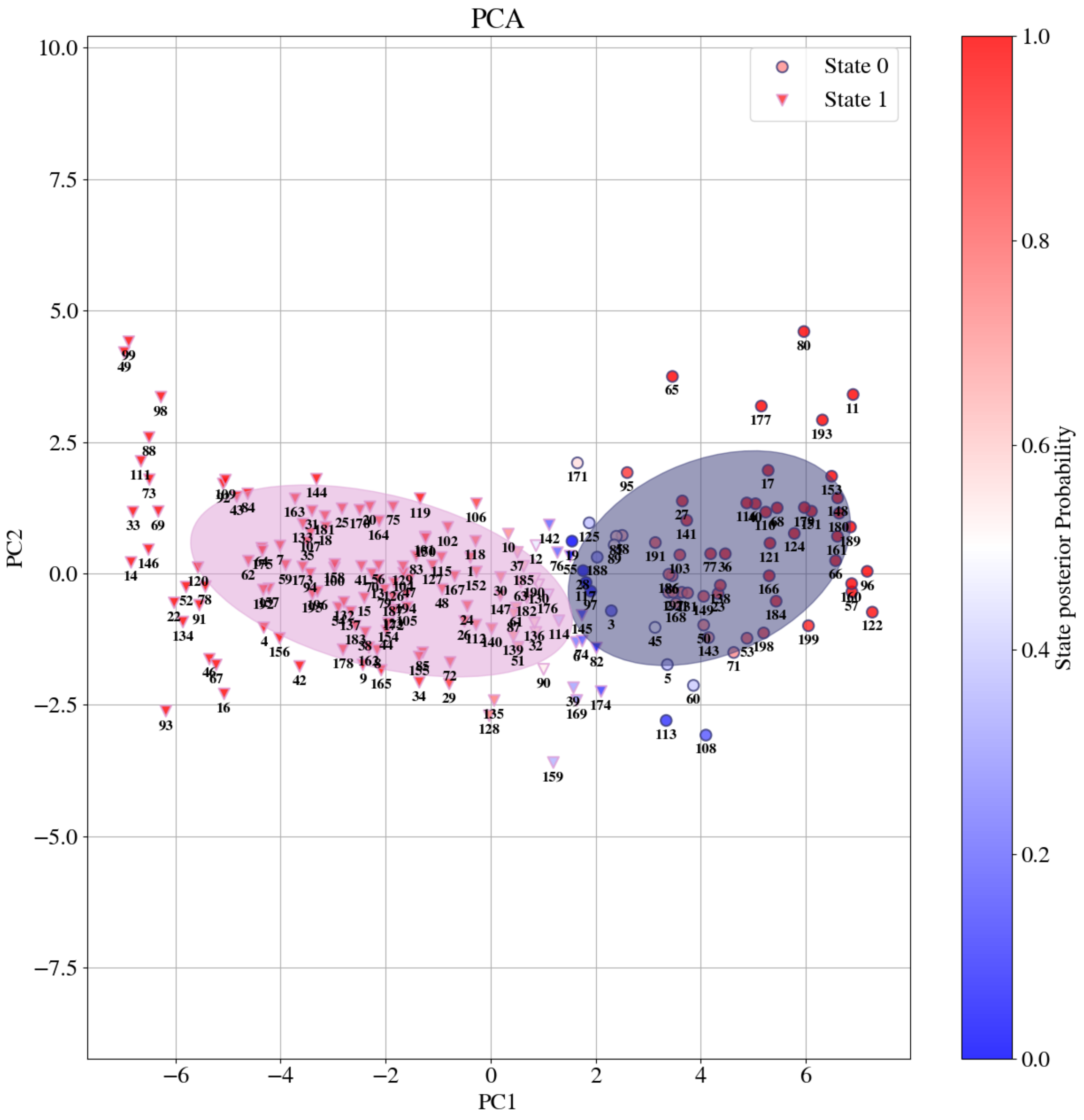


Figure 11: *Cell posterior probability $P(\vec{x}|c_i)$ scatter-plot*

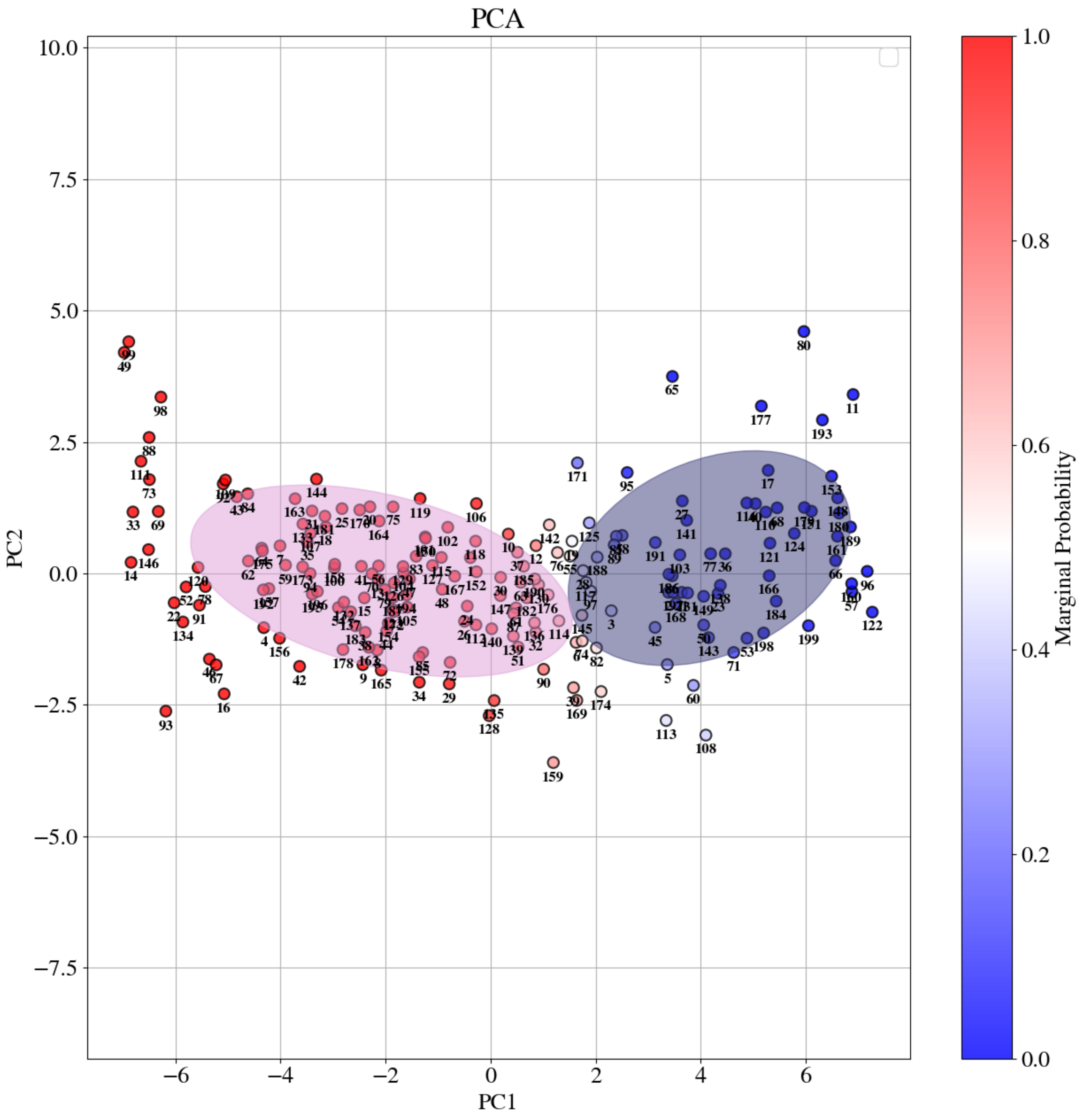


Figure 12: *Cell marginal probability $P(\vec{x})$ scatter-plot*

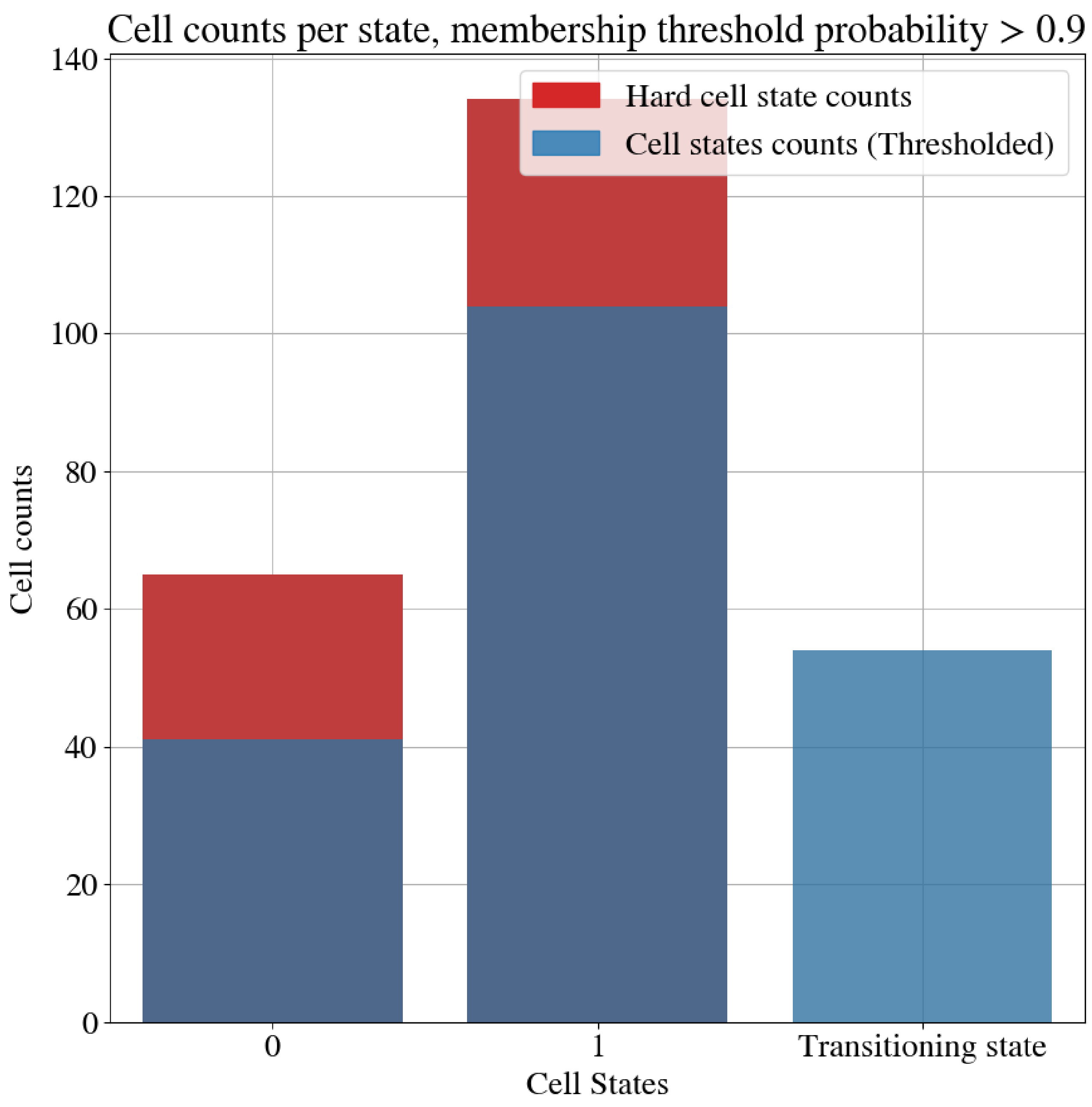
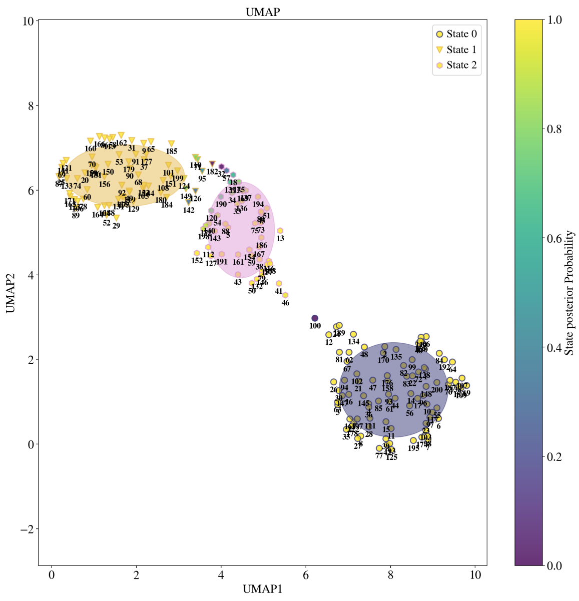
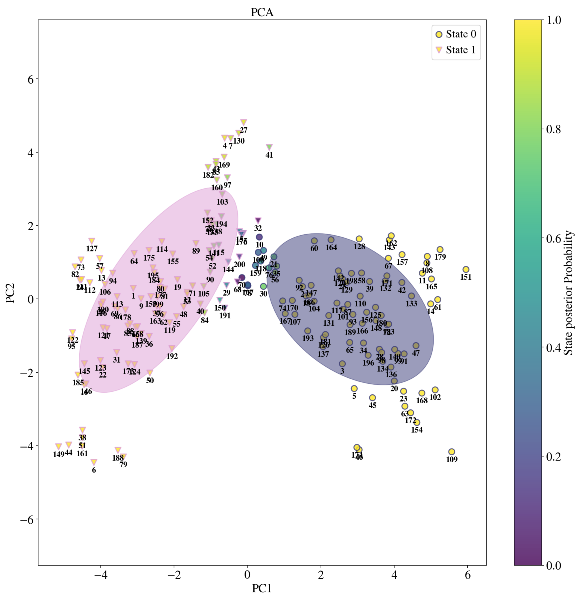


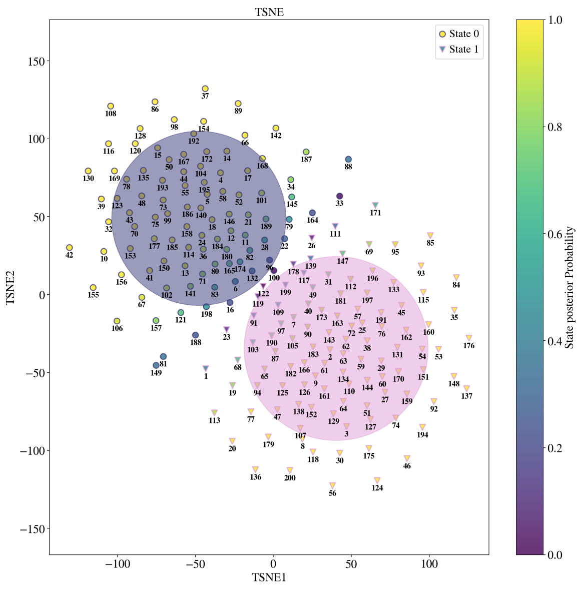
Figure 13: *Cell counts for either hard clustering or with a posterior probability threshold*



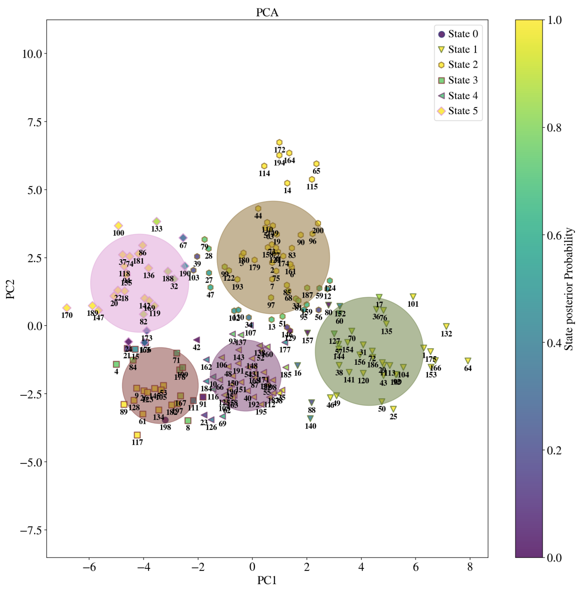
(a) Synthetic dataset 2 pipeline results



(b) Synthetic dataset 3 pipeline results



(c) Synthetic dataset 4 pipeline results



(d) Synthetic dataset 3 pipeline results

Figure 14: *Cell posterior graph results for each pipeline and dataset*