HELLENIC REPUBLIC
National and Kapodistrian
University of Athens

Postgraduate Program: «Data Science and Information Technologies»

Machine Learning in Computational Biology

Assignment #2

Postgraduate Student: Gavrielatos Marios
Registration Number: 7115152100023

Introduction

In this assignment we want to perform unsupervised learning for five different single-cell synthetic datasets. Each dataset provides the gene expression profiles (200 Genes) of 200 cells. In this report we will analyze three steps of the data analysis pipeline that accepts a dataset file as input. The first part of the pipeline is the preprocessing, following by a dimensionality reduction step, and finally the clustering of the dimensionally reduced data. First, we will perform a manual inspection of the analysis, testing different parameter combination and their effect on the final clustering results. For this analysis we used `dataset1`. Finally, we will present the automated pipeline for clustering using 3 different dimensionality reduction techniques.

Pre-processing

Before applying any preprocessing analysis, we have to inspect the dataset. We observe that the gene expression profiles have decimal numbers. This means that a preprocessing analysis may have been performed on these datasets. However, because the datasets are synthetic, we produced similar barplots of Figure 1 in order to check if the datasets are normalized. As we can see the data are not normalized and the gene distributions are skewed. For this reason, we will first perform a data pre-processing step before moving on to the dimensionality reduction stage.

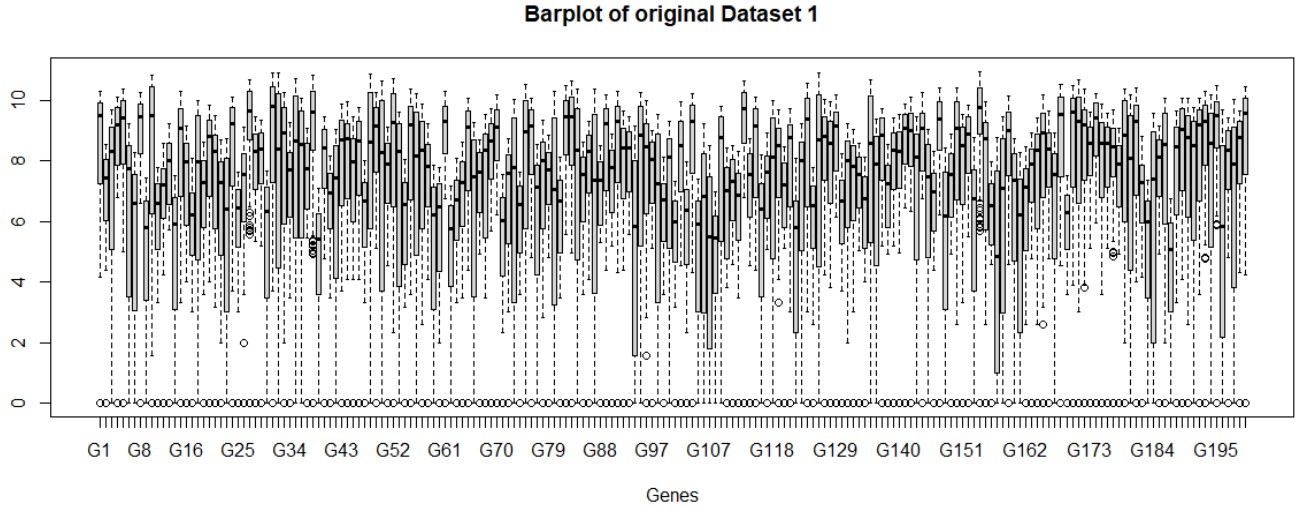


Figure 1: *Barplot of `dataset1` before any preprocessing analysis.*

Decorrelation

We removed the highly correlated variables, meaning the variables with correlation bigger than 0.7. Highly correlated variables do not provide extra information about our data and for this reason it is suggested to remove them. We use the R findCorrelation function in order to return a vector of integers corresponding to columns to remove to reduce pair-wise correlations. For `dataset1` the high correlated genes are the following: "G10" "G32" "G48" "G80" "G87" "G95" "G126" "G135" "G180" "G196" "G50" "G73". The rest of the analysis will be performed using the dataset without these genes.

Standardization

The next step of the data preprocessing stage is the standardization and the normalization of the dataset. Many algorithms, like PCA, expect the features to be scaled. For our analysis we performed different combinations of standardization and normalization. In particular, we examined the combinations of log transformation, min-max normalization, centering and scaling. For each combination we created a boxplot (Figure 2). From the following boxplots we choose the Centered and Scaled version of `dataset1` for the downstream analysis.

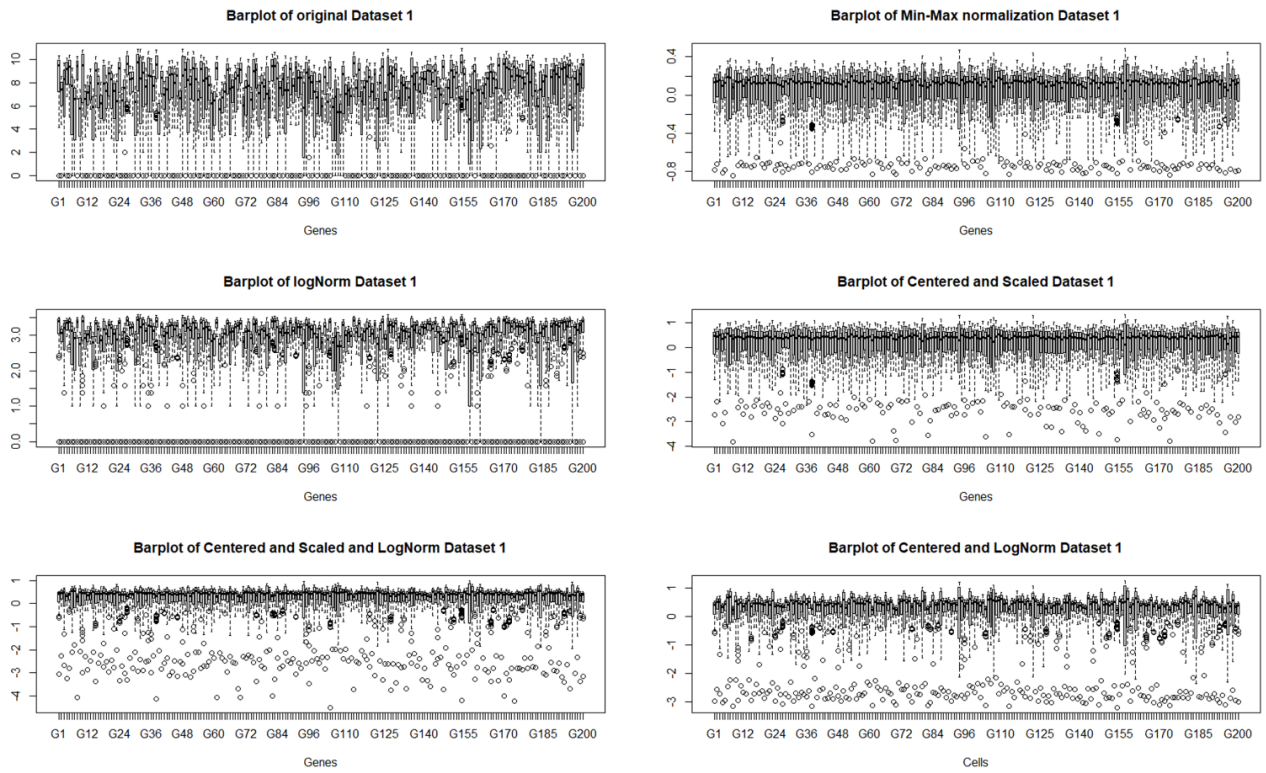


Figure 2: *Barplot of* `dataset1` *after different kinds of Normalization and Standardization.*

Dimensionality reduction

PCA

The first dimensionality reduction method we perform is Principal Component Analysis (PCA) (Pearson 1901). For this purpose, we use the R prcomp. Using the fviz_pca_ind function we produce the following plot:

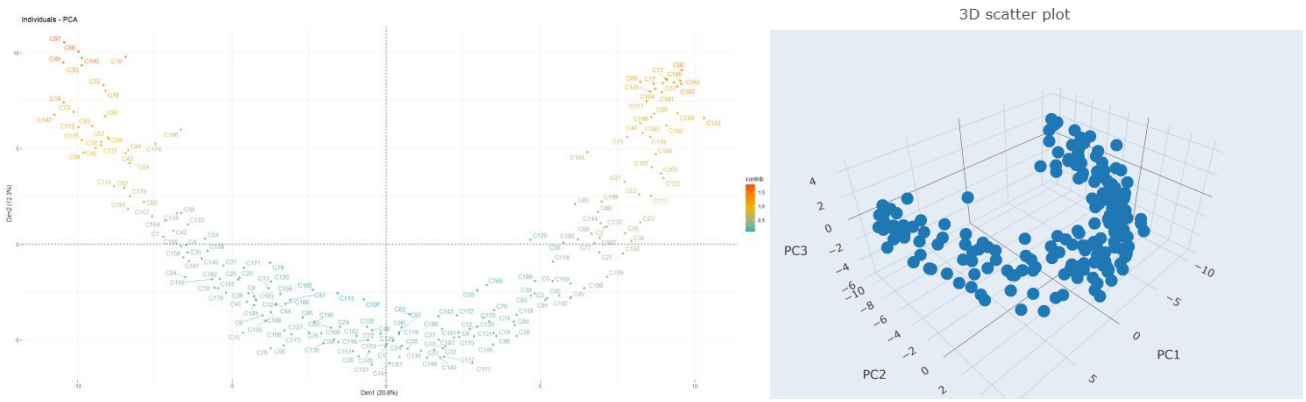


Figure 3: *2D and 3D scatter plots of the dimensionally reduced data produced with PCA.*

We would prefer to keep as many principal components in order to retain 70-80% of cumulative variance. In order to check how many principal components we need to reach 80% of the cumulative variance we create a scree plot. Furthermore, we shuffle the dataset and perform PCA and check what would the above plot look like for the permuted matrix (red line) (Figure 4). This way we can see that only the first two components are not explained by chance. Therefore, the optimal number of components (components with p-value > 0.05) using PCA is 2. In order to reach the 80% level of variance we would need 59 components.

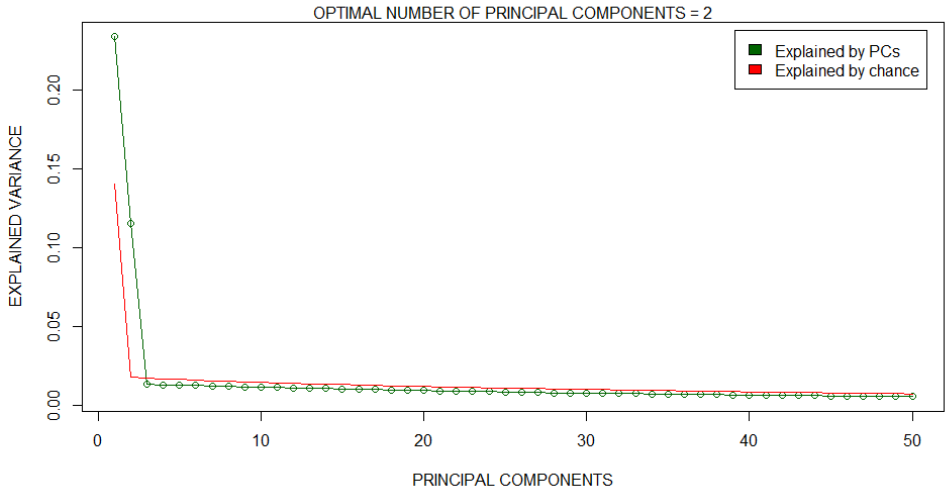


Figure 4: *Scree plot of the original dataset (green line) and scree plot of the randomized dataset (red line). This plot allows as to understand which Principal Components can be explained by chance and therefore, we are able to find the optimal number of principal components.*

tSNE

tSNE (t-distributed stochastic neighbor embedding) (Van der Maaten and Hinton 2008) is a non-linear dimensionality reduction technique which is widely used for Single-Cell analysis (Zhou and Jin 2020). tSNE is a dimensionality reduction technique that does not retain distances between the points but probabilities. This happens because the optimization tSNE uses the Kullback-Leibler divergence (cost function) which does not preserve global distances (Van der Maaten and Hinton 2008). In order to perform tSNE analysis, we will use the R Rtsne function. Before implementing tSNE analysis we must tune certain hyperparameters and in particular **perplexity**, **initial dimensions,** and **the number of iterations** to perform.

Starting with the number of initial principal components (default 50), in order to perform the tSNE analysis, it is suggested to use a reduced number of initial components and in particular in the range of 30 to 50 components. For this reason, we must include only the informative PCs of our dataset. From the PCA analysis we already performed we are able to conclude that there are only two informative PCs in `dataset1`. By reducing the number of initial components, we reduce the random noise in the tSNE analysis. However, if we select too few initial dimensions, we will lose important information. We choose to use `initial_dims=30` if the optimum components calculated from PCA are less than 10.

The second hyperparameters we must tune in the perplexity, which is a measure for information that is defined as 2 to the power of the Shannon entropy (Van der Maaten and Hinton 2008). Typical values for the perplexity range between 5 and 50 as stated by the tSNE author Laurens van der Maaten. However, larger datasets require larger perplexity values. A rule of thumb is to use Perplexity = N^(1/2), where N is the number of variables, however this will not be the optimal result. For this purpose, we test the following perplexity values [3, 6, 8, 11, 14, 17, 20, 22, 62]. The following figure (Figure 5) shows the tSNE result for the aforementioned perplexity values with PC=30 for `dataset1`.

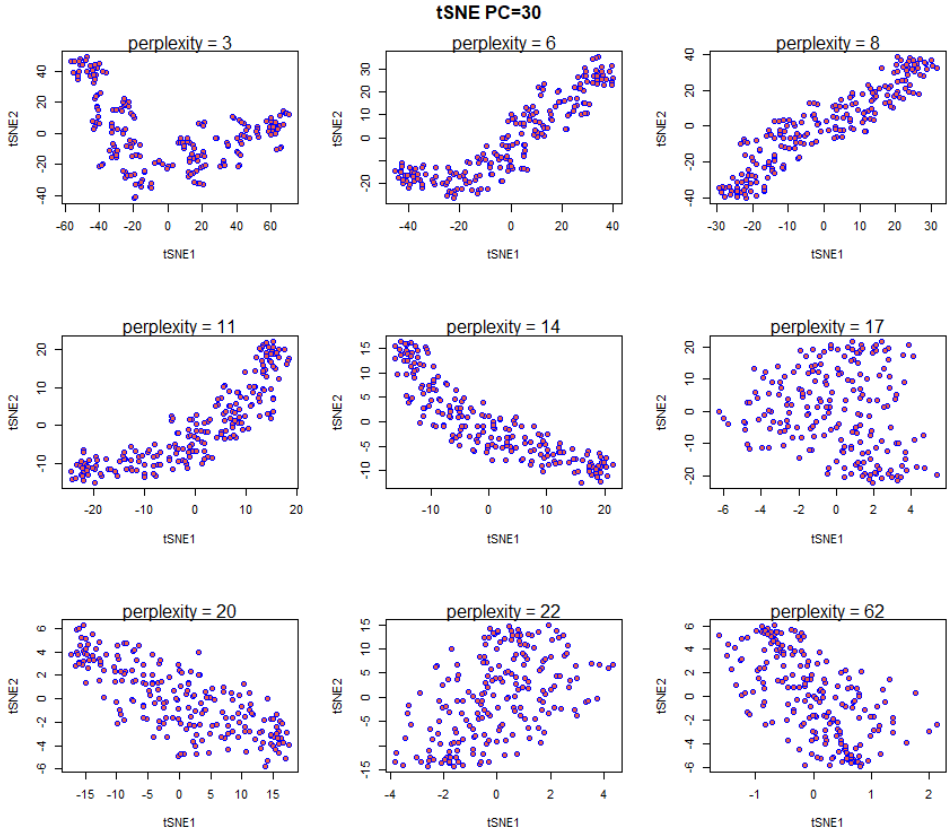


Figure 5: *tSNE analysis using 30 initial components testing for 9 different values of perplexity.*

We also perform the same analysis with PC=2 (optimal number of PCs) and PC=59 (required number of principal components in order to reach 80% of cumulative variance) (Figure 6).

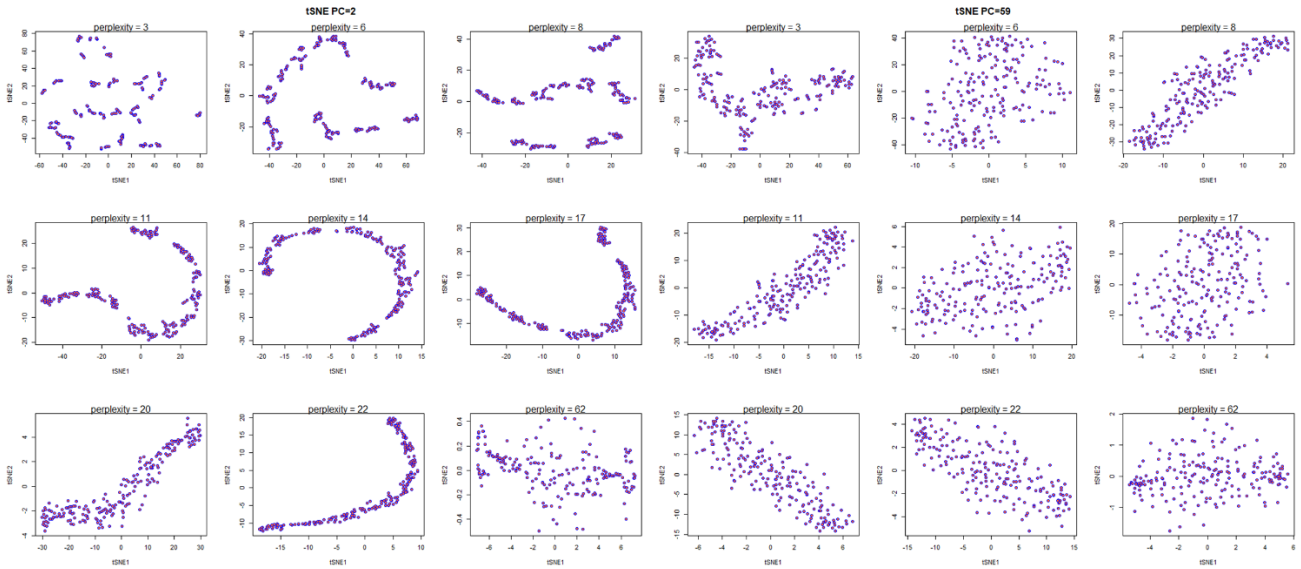


Figure 6: *tSNE analysis using 2 and 59 initial components testing for 9 different values of perplexity.*

By observing the above plots, we are able to see that by using 2 initial dimensions, dense clusters are formed and this may lead to losing the global structure of the data and over estimate the number of true clusters in the dataset. On the other hand, when using 59 initial dimensions we introduce a lot of random noise in the tSNE analysis. For these reasons, we choose to perform the rest of the tSNE analysis using 30 initial dimensions.

Moreover, we must tune the number of iterations. The aforementioned experiments were performed using max_iter = 5000. In Figure 7 we are able to see the differences between the different number of maximum iterations. We will choose max_iter=5000 because the global structure of the data seems to stabilize at that point.

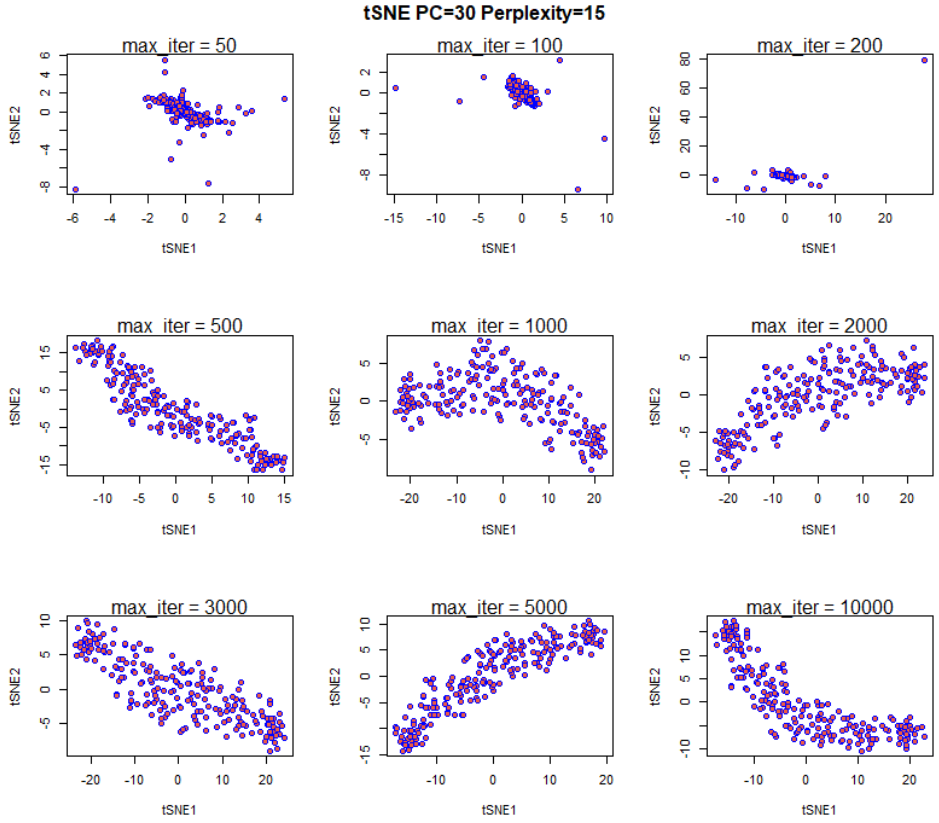


Figure 7: *tSNE analysis using 25 initial components testing for 9 different values of maximum iterations.*

The tSNE parameters (Figure 8) we will use for clustering of `dataset1` are the following:

- `initial_dims = 30`
- `perplexity=15`
- `max_iter = 5000`

For the automated pipeline we will need to tune `perplexity` for every specific dataset. For larger datasets we will have to tune the number of `max_iter` as well. The number of `initial_dims` depend on the PCA results. For larger datasets we need a larger number of `initial_dims`.

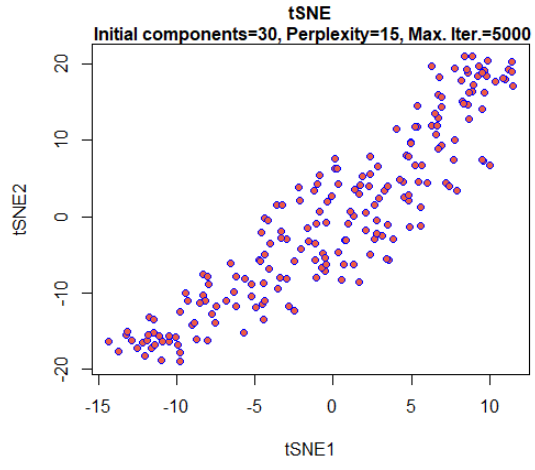


Figure 8: *The above plot will be used for the clustering analysis.*

UMAP

UMAP (Uniform Manifold Approximation and Projection) (McInnes, Healy, and Melville 2018), like tSNE, is a non-linear dimensionality reduction technique which is widely used in the analysis of scRNA-seq data (Becht et al. 2018). In contrast to tSNE, UMAP preserves the global structure of the data when performing dimensionality reduction. Unlike KL-divergence, the cost function of UMAP (Cross Entropy) retains the global structure of the data (McInnes, Healy, and Melville 2018). We implement UMAP using the R umap function. Like tSNE, before implementing UMAP analysis we must tune certain hyperparameters and in particular **the number of neighbors** and the number of **effective minimum distance between embedded points**.

The number of minimum distance may affect the visualization of the dataset. As seen from the UMAP docs, large values of minimum distance create sparse data, losing the more detailed topological structures. For this reason, we choose the following `min_dist` values for our tests: (0.05, 0.1, 0.2, 0.3, 0.4). The second parameter we will investigate is the number of neighbors (default=15) which controls the balance between the local and the global structure of the data. Low values favor the local structures whereas large values favor the global structures. We will test the output of UMAP using the following `n_neighbors` values: (7, 15, 25, 35). In Figure 9 we can observe how the UMAP algorithm works for the different parameter combinations. It is suggested by the UMAP docs to reduce the number of components, and thus reduce the random noise by using PCA and then implement UMAP in order to further reduce the number of dimensions. In this analysis we used the first 50 principal components as input for the UMAP function.

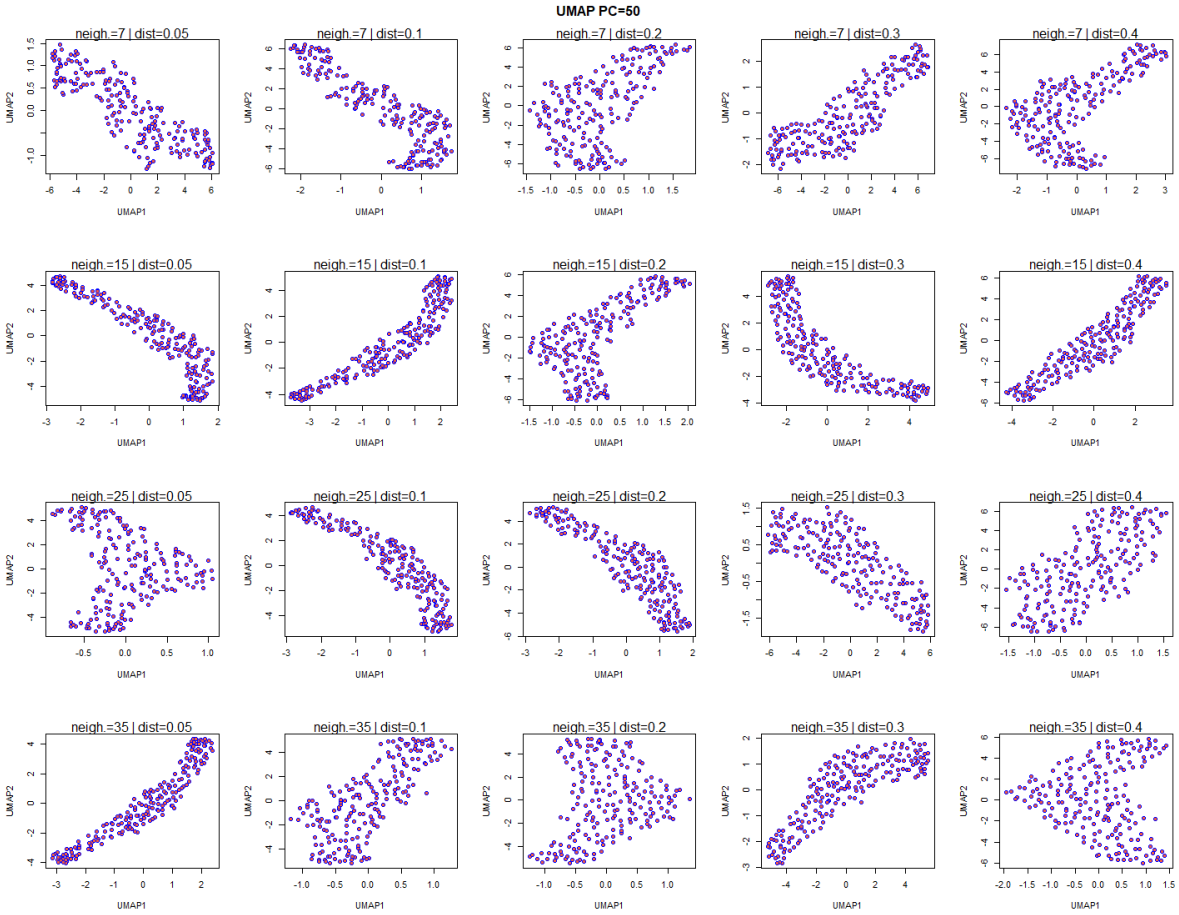


Figure 9: *UMAP analysis using 50 initial components testing for different values of number of neighbors and minimum distance values.*

By observing the plots, we are able to see that the output of UMAP is steadier and more consistent that the output of tSNE which is more random. Different combinations of number of neighbors and distance values will preserve the global structure of the data, creating more sparse representations or denser clusters. In general, as we increase the number of neighbors the local structures are lost and even in this small range of `min_dist` values we can observe that as we increase the value of the parameter the data points are sparser.

The UMAP parameters we will use for clustering of `dataset1` are the following:

- `min_dist = 0.1` (default)
- `n_neighbors = 25`
- `n_epochs= 2000`
- 50 initial components

For the automated pipeline we will need to tune `min_dist` and `n_neighbors` for every specific dataset. For larger datasets we will have to increase the number of `n_epochs` as well.

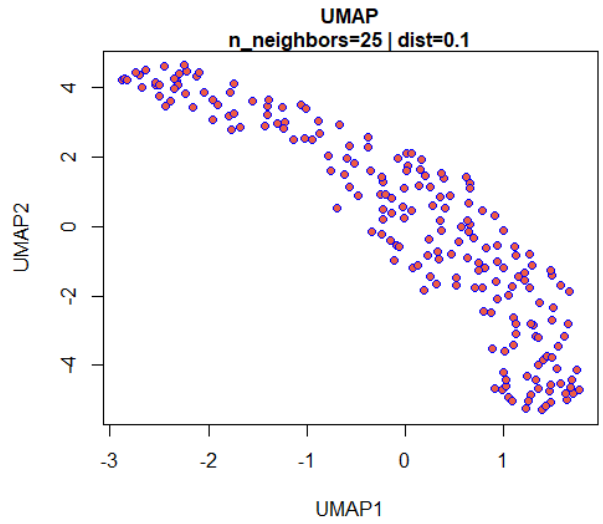


Figure 10: *The above plot will be used for the clustering analysis.*

Finally, we tested different initialization methods for both tSNE and UMAP. The initialization method did not seem to have an effect on the results of both algorithms and for this reason we do not report them in this report.

Clustering

We use the R Mlclust package (Scrucca et al. 2016) for model-based clustering. We use the BIC (Bayesian information criterion) criterion in order to find the optimal number of components and the covariance matrix structure that will be used by the Gaussian Mixture Model (GMM). We tested the clustering output using the PCA, the tSNE and the UMAP output with the hyperparameters tuned as we have already described. Furthermore, we created state-to-state trajectory plots using the cell posterior probabilities. The results are the following:

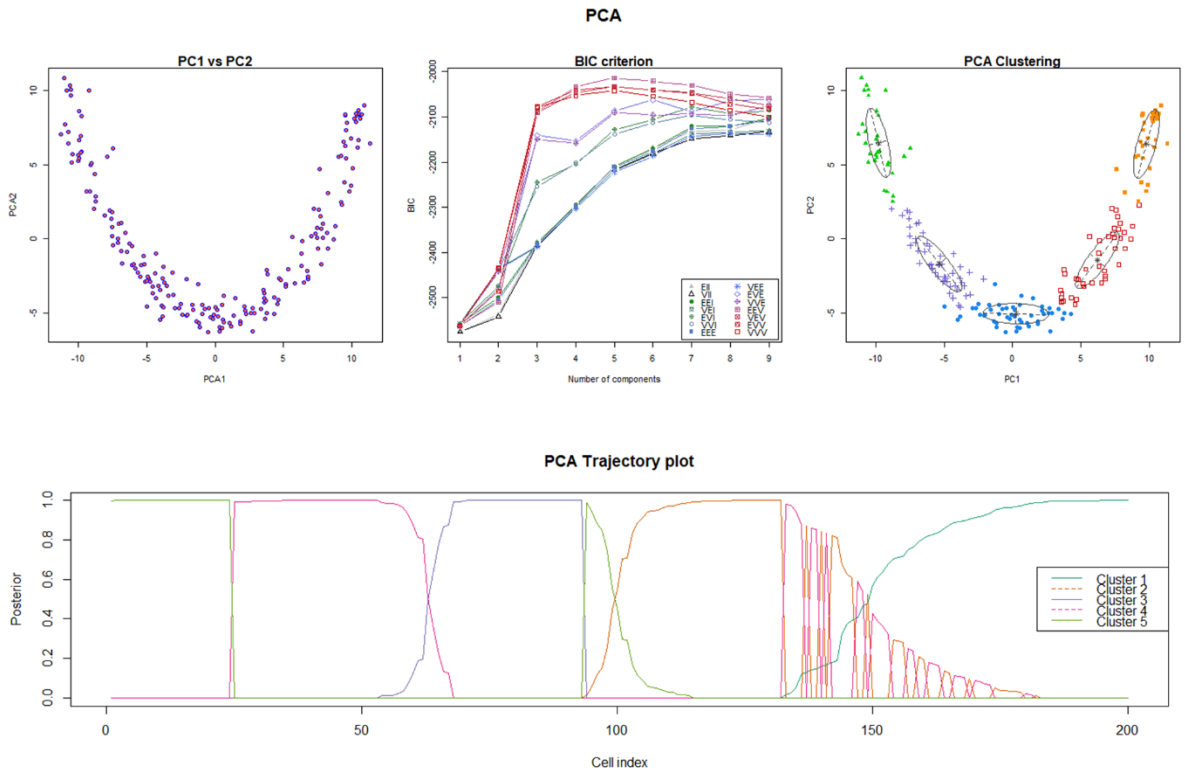


Figure 11: *PCA clustering results and trajectory plots.*

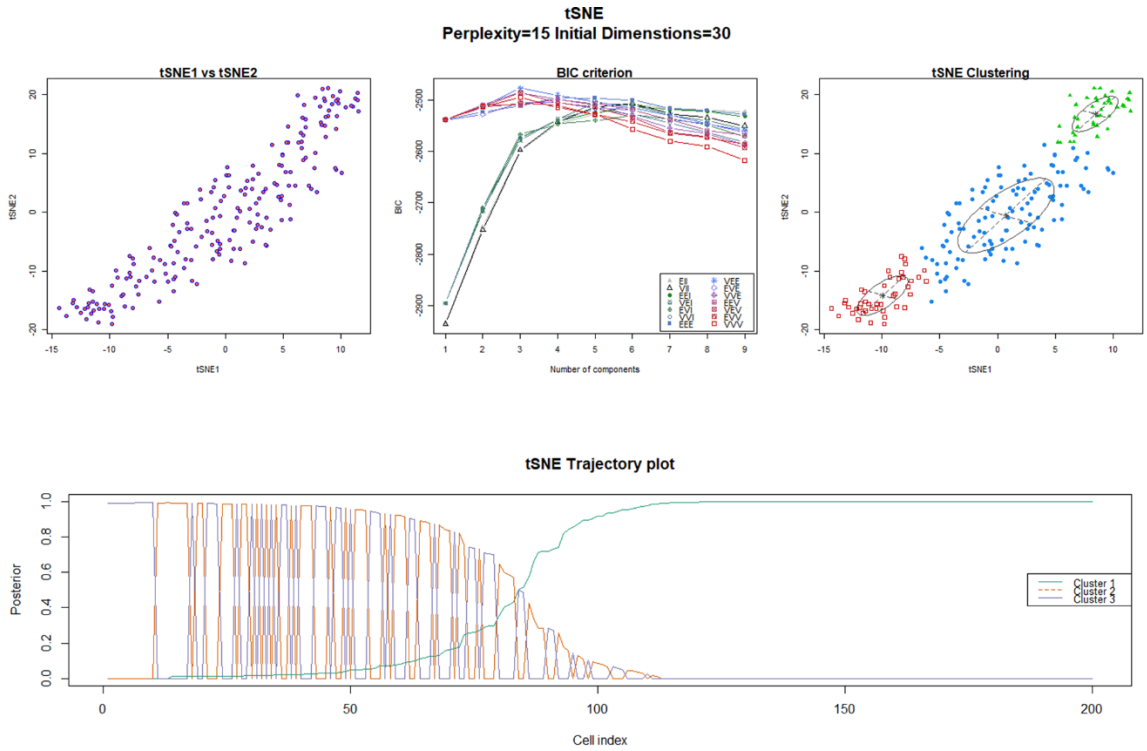


Figure 12: *tSNE clustering results and trajectory plots.*

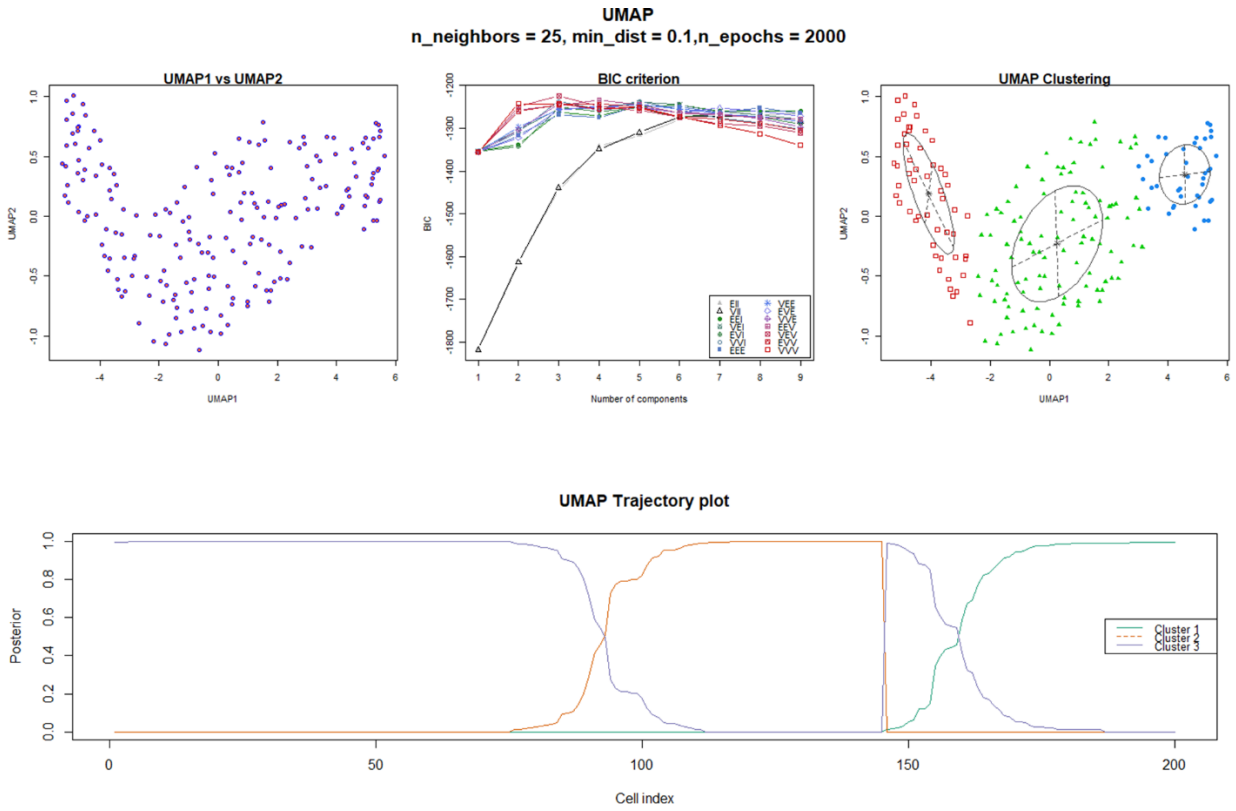


Figure 13: *UMAP clustering results and trajectory plots.*

From the above plots we are able to see that UMAP and tSNE create 3 clusters in contrast to PCA which forms 5 different clusters. We must not derive information from the tSNE plot about the global structure of the data because tSNE plots do not preserve the global structure of the data as we have already discussed. In addition, the stochastic nature of the algorithm does not give us constant results. In contrast, UMAP provides good information about the local and the global structure of the data. Therefore, it gives a more accurate presentation of the global structure, something very useful when we need to do trajectory analysis.

Among PCA, tSNE and UMAP, only PCA gives us information about the clustering and the variation of the data. However, PCA is a linear dimensionality reduction technique, thus, it is not optimal for the analysis of the sparse, non-linear scRNA-seq data. Furthermore, the state-to-state trajectory plot in Plot 2 created by UMAP indicates a clear transition between the 3 cell-states. Finally, we must note that no dimensionality reduction technique is perfect and it must not be treated as such. When we use non-linear dimensionality reduction techniques like tSNE and UMAP, we must tune the hyperparameters very carefully because they have a great impact on the clustering results. A criterion that can be used for this purpose is the BIC criterion.

Final pipeline

Taking everything under consideration, we created an automated pipelines using the aforementioned tools. The steps of the pipeline are the following:

1. Remove highly correlated variables.
2. Centering and scaling of the dataset.
3. PCA will be used in order to reduce the number of dimensions of the dataset to less than 50 dimensions.

4. Dimensionality reduction using tSNE and UMAP. We test the clustering results using both dimensionality reduction techniques. For each technique we test different combinations of hyperparameters, which depend on the length of the dataset.
    a. UMAP – 50 different combinations:
        i. n_neighbors: range of N*0.025 - N*0.25 with step equal to N*0.025. This will provide 10 different values of n_neighbors.
        ii. min_dist: test the values 0.05, 0.1, 0.2, 0.3 and 0.4.
        iii. The dataset will be reduced to 50 components using PCA.
        iv. epochs: 2000.
    b. tSNE:
        i. max_iter: 5000.
        ii. perplexity: range 5 to 50 with step equal to 5. The 50 value will have to change for large datasets. Note: tSNE is slow for large datasets and requires many computational resources and for this reason it is not advised to use tSNE for large datasets.
        iii. initial_dims: same as the *optimum initial components* produced by PCA, or the *number of initial components which retain 80% of the variance* if that number is less than 50, or 30.
5. We choose the optimum BIC value and its corresponding set of parameters for both tSNE and UMAP and perform clustering.

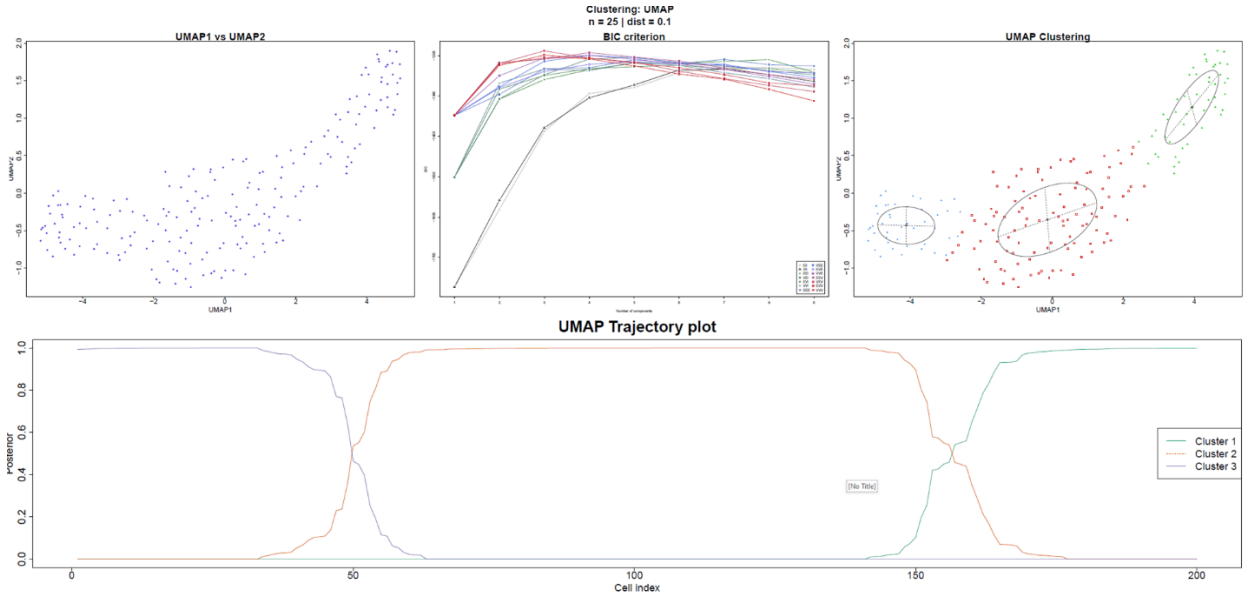
Results

As our final result we choose the UMAP results, as the tSNE and PCA results are produced for the purpose of this assignment. The pipeline will output a PDF file (datasetx_pdf) per dataset which includes the following figures:

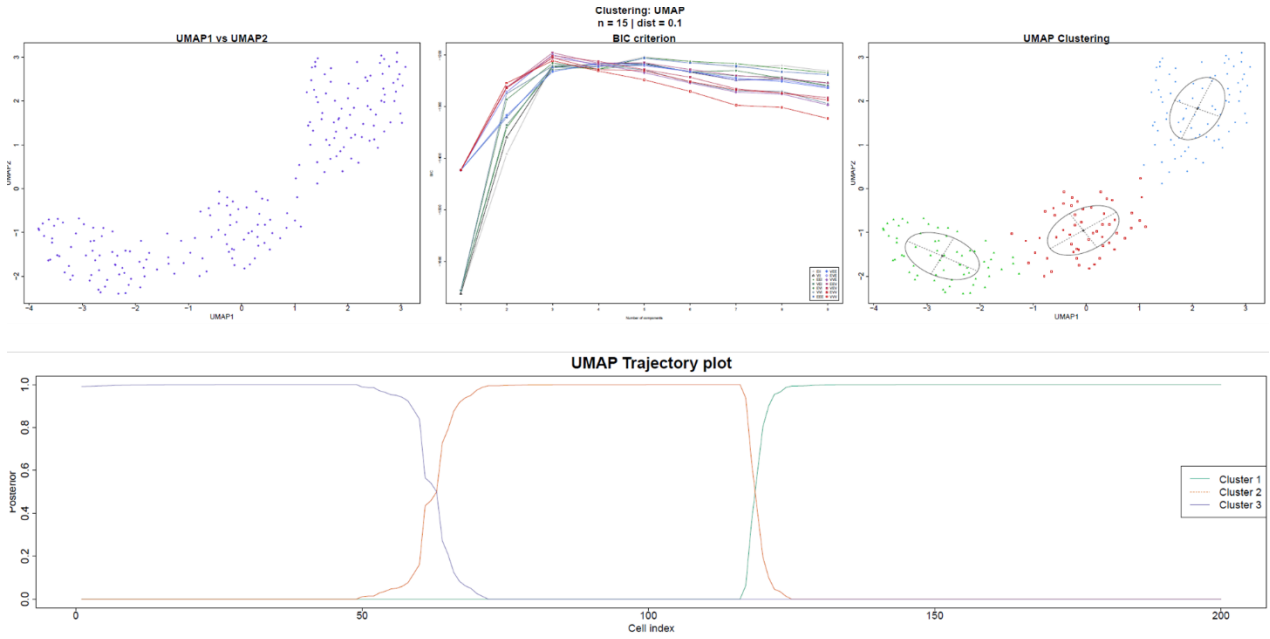
1. Boxplot of the centered and scaled data.
2. Scree plot to find the optimal number of components.
3. PCA, UMAP and tSNE results (clustering and trajectory results) for every parameter combination. The last plot from the UMAP and tSNE analysis is the plot with the optimal BIC.

We must not consider that the case with the optimum BIC is the optimum clustering result. The results from each dataset should be inspected carefully and taking into account the trajectory results and the information we may have about the dataset (not applied in our analysis) we should pick the best parameters. Taking everything into account the optimum results for each dataset are the following:

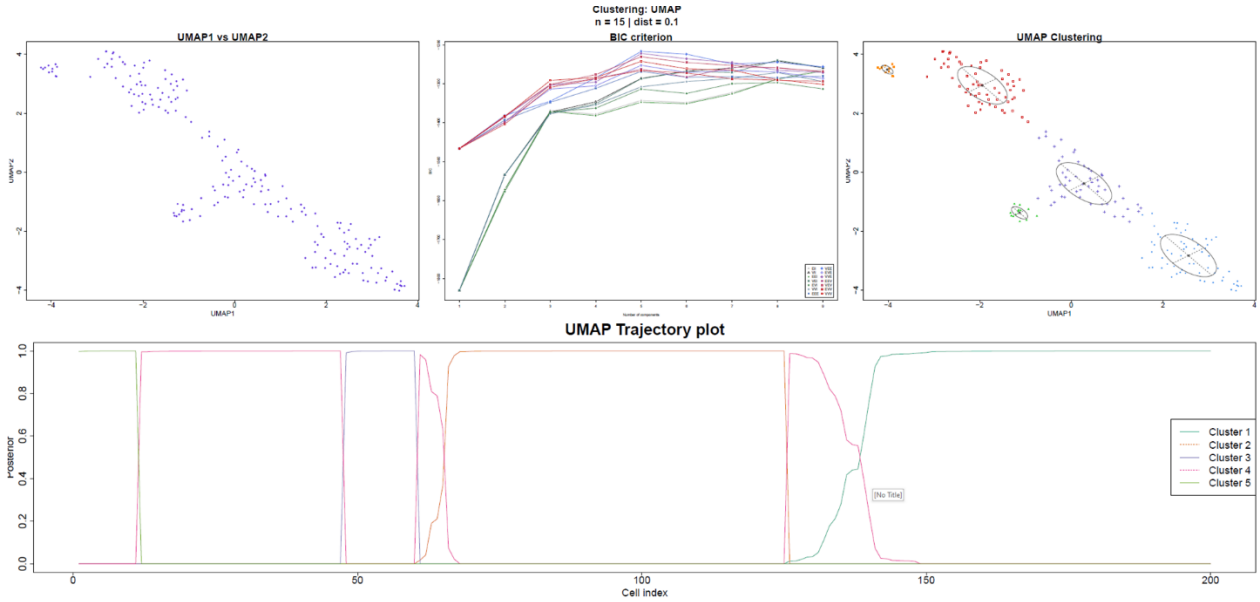
- Dataset 1: UMAP, 3 clusters, n_neighbors = 25, min_dist = 1



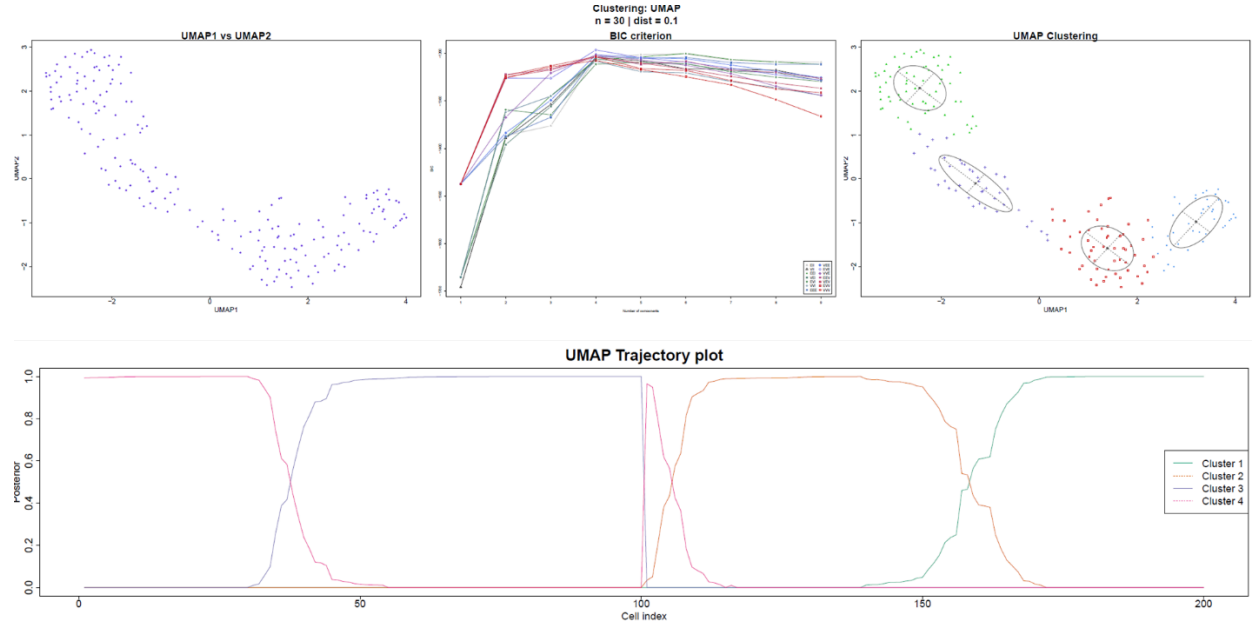
- Dataset 2: UMAP, 3 clusters, n_neighbors = 15, min_dist = 0.1



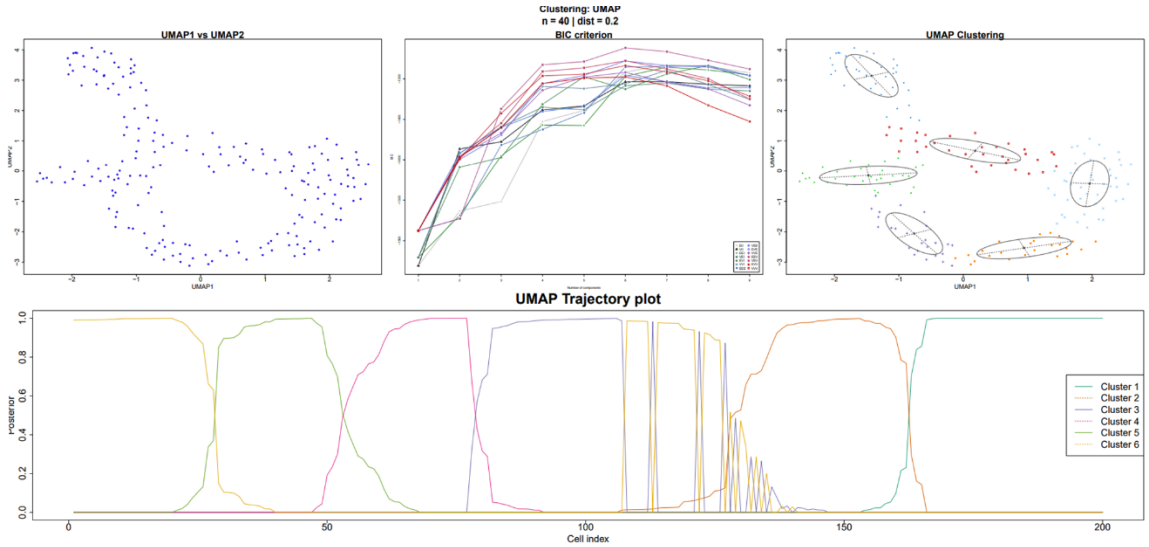
- Dataset 3: UMAP, 5 clusters, n_neighbors = 15, min_dist = 0.1



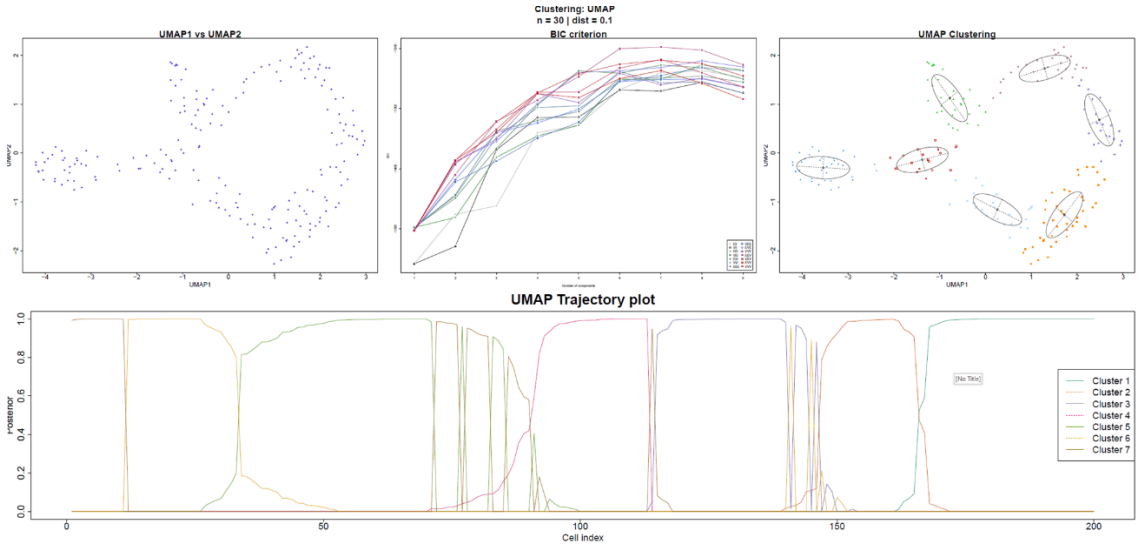
- Dataset 4: UMAP, 4 clusters, n_neighbors = 30, min_dist = 0.1



- Dataset 5: UMAP, 6 clusters, n_neighbors = 40, min_dist = 0.2



or UMAP, 7 clusters, n_neighbors = 30, min_dist = 0.1



Included files

1. Report in pdf format
2. Automated pipeline in .R format: scRNAseq_pipeline.R
3. Automated pipeline in .Rmd format: scRNAseq_pipeline.Rmd
4. Initial analysis for dataset1 in .R format: initial_analysis.R
5. 5 PDF files with the results

In the three R first files you will find the command used to install the necessary packages.

References

Becht, Etienne, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel W. H. Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W. Newell. 2018. “Dimensionality Reduction for Visualizing Single-Cell Data Using UMAP.” *Nature Biotechnology*, December. https://doi.org/10.1038/nbt.4314.

McInnes, Leland, John Healy, and James Melville. 2018. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.” *ArXiv [Stat.ML]*. arXiv. http://arxiv.org/abs/1802.03426.

Pearson, Karl. 1901. “LIII. On Lines and Planes of Closest Fit to Systems of Points in Space.” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2 (11): 559–72.

Scrucca, Luca, Michael Fop, T. Brendan Murphy, and Adrian E. Raftery. 2016. “Mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models.” *The R Journal* 8 (1): 289–317.

Van der Maaten, L., and G. Hinton. 2008. “Visualizing Data Using T-SNE.” *Journal of Machine Learning Research: JMLR*. https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf?fbclid=IwA.

Zhou, Bo, and Wenfei Jin. 2020. “Visualization of Single Cell RNA-Seq Data Using t-SNE in R.” *Methods in Molecular Biology* 2117: 159–67.