CS690A: Computational Genomics
Clustering of Single Cell Sequencing Data

Debarpita Dash
220328

Assignment 1
August 28, 2024

TASKS

Task 1.1

Quality Control

To ensure high-quality single-cell RNA-seq data, I first computed total gene expression and its logarithm to handle data skew. I then evaluated gene detection breadth and the fraction of mitochondrial gene expression to identify potential issues. By categorizing genes and generating targeted quality control metrics, I visualized the distribution of gene detection, total expression counts, and mitochondrial percentages to comprehensively assess data quality. Following this, I ran a doublet detection algorithm to identify potential doublets that could mislead subsequent analysis.

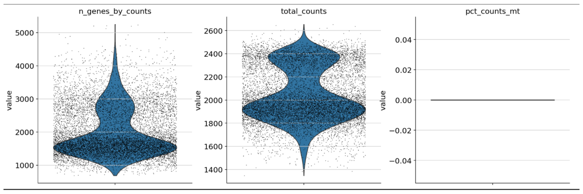

Fig. 1. Violin plots illustrating the distribution of gene detection, total expression counts, and mitochondrial gene percentages across cells.

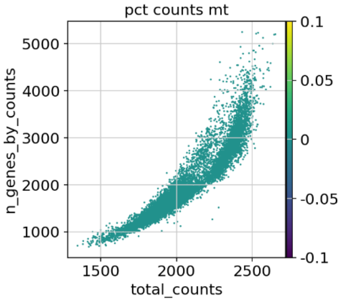

Fig. 2. Scatter plot showing the relationship between total gene counts and the number of detected genes, with points colored by the percentage of mitochondrial gene counts.

Normalization and Feature Selection

Next, I performed count depth scaling followed by a log plus one (log1p) transformation to normalize the data. This process adjusted for differences in sequencing depth by scaling to a size factor, such as the median count depth across the dataset. For dimensionality reduction, I ran Principal Component Analysis (PCA) to highlight the main axes of variation and reduce noise, focusing only on the most informative genes.

Leiden Clustering

I applied the Leiden clustering algorithm, which is designed for community detection in networks. This algorithm identifies groups of cells that are more densely connected within the group than to the rest of the dataset. I adjusted the resolution parameter to find the optimal clustering.

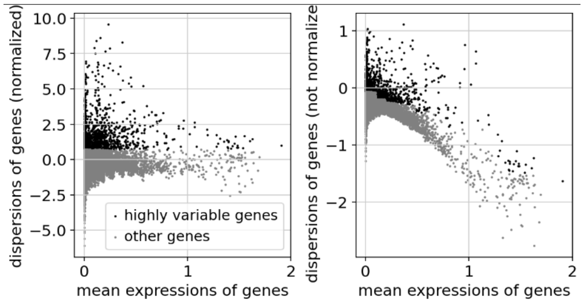

Fig. 3. Normalized dispersion of genes versus mean expression of genes.

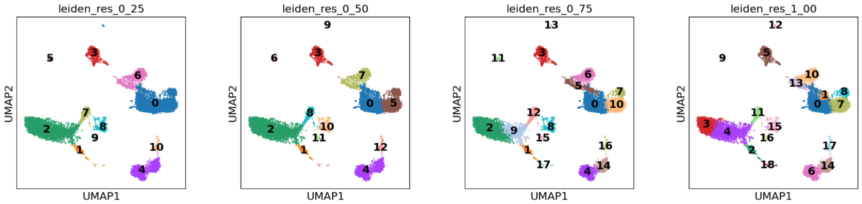

Fig. 4. Leiden clustering results with varying resolutions.

After clustering, I identified and visualized key genes that distinguish each cluster. I ranked genes based on their differential expression and visualized the top genes using a dot plot. To refine this, I filtered genes to include only those expressed in a meaningful fraction of cells within each cluster and not excessively common outside it. I then generated a second dot plot to highlight these filtered genes, offering a clearer view of the key markers for each cluster.

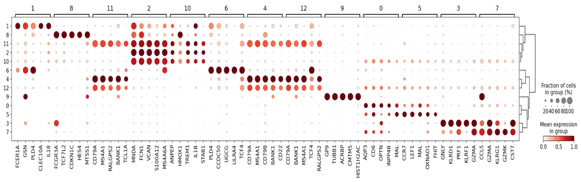

Fig. 5. Dot plot of filtered differentially expressed genes.

Manual Annotation

For final clustering, I used PanglaoDB, a database for exploring single-cell RNA-seq experiments. This allowed me to perform clustering with manual annotations.

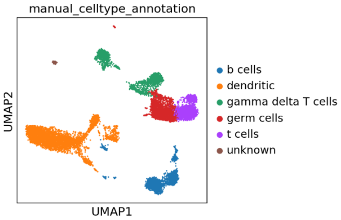

Fig. 6. Final clustering results with manual annotations.

Finally, I examined the spatial distribution of my genes of interest using a UMAP plot, highlighting the expression of CD3D, CD79A, NKG7, and CD4.

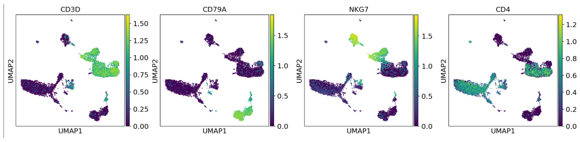

Fig. 7. UMAP plot showing the spatial distribution of CD3D, CD79A, NKG7, and CD4.

Task 1.2

I performed feature selection using deviance rather than the typical highly variable genes approach. This process involved leveraging the ‘pipecomp‘ package to select features based on deviance—a statistical measure of goodness-of-fit in models. This method was executed using the ‘rpy2‘ package, which integrates R functionality into Python.

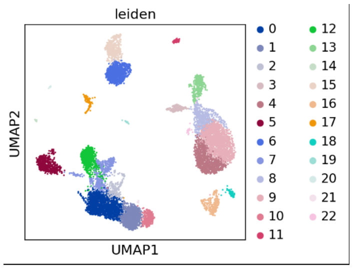

Fig. 8. Clustering results with feature selection based on deviance.

I first performed clustering and then plotted a dot plot to identify and visualize key genes that distinguish each cluster.

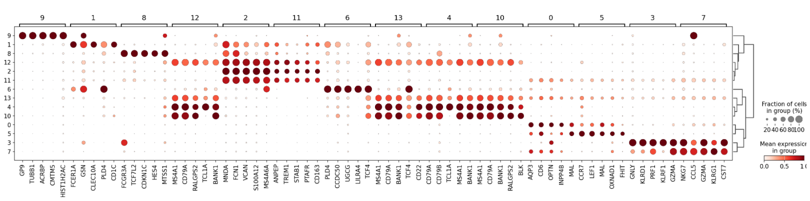

Fig. 9. Dot plot showing key genes distinguishing each cluster.

After manual annotation, I observed slight variations in a few clusters, but overall, the clustering results remained consistent. The number of clusters increased from 12 to 13.

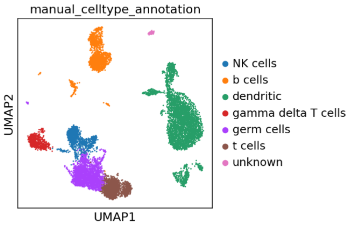

Fig. 10. Final clustering results with manual annotations.

Finally, I examined the spatial distribution of my genes of interest using a UMAP plot, highlighting the expression of CD3D, CD79A, NKG7, and CD4.

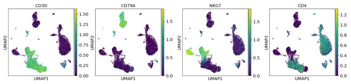

Fig. 11. UMAP plot showing the spatial distribution of CD3D, CD79A, NKG7, and CD4.

Task 2

Attempt 1. I first implemented the Louvain algorithm instead of the Leiden algorithm. The Louvain algorithm, which optimizes modularity, is generally faster and more straightforward but can suffer from issues such as resolution limits, where it might merge smaller communities into larger ones. The Leiden algorithm improves on this by addressing these resolution issues and refining community detection through a more robust optimization process. This results in higher-quality clusters and better handling of small communities. However, the Leiden algorithm is more computationally intensive compared to Louvain.

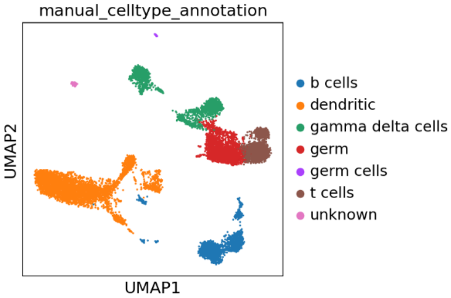

Fig. 12. Final clustering results with Louvain algorithm.

Attempt 2. Using linearly decoded VAE for dimensionality reduction:

In this attempt, I used the scVI model to learn low-dimensional latent representations of cells. These representations are mapped to parameters of probability distributions, which can generate counts consistent with the observed data. In the standard scVI model, these parameters for each gene and cell are derived from neural networks applied to the latent variables. While neural networks are flexible and can handle non-linearities, they lack a direct link between a latent variable dimension and the genes that covary with it.

To address this, I employed the LDVAE model, which replaces neural networks with linear functions. This means that a higher value along a latent dimension directly corresponds to higher expression of genes with high weights for that dimension. This approach results in a generative model similar to probabilistic PCA or factor analysis but generates counts rather than real numbers.

In this notebook, I fit an LDVAE model to single-cell RNA-seq data and visualized the latent variables. I began by subsampling 1,000 genes from the data and initializing a LinearSCVI model with a latent space of 10 dimensions. After training the model, I checked convergence by plotting the training and validation ELBO (evidence lower bound) metrics.

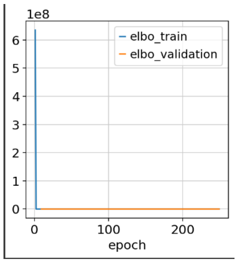

Fig. 13. ELBO plot showing convergence metrics.

I visualized the latent dimension coordinates for each cell using a series of 2D scatter plots covering all dimensions. Since the cells are represented by 10 dimensions, this resulted in 5 scatter plots. By extracting these weights from the LDVAE model, I identified genes with high weights for each latent dimension.

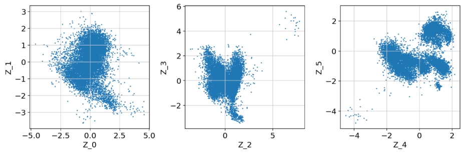

Fig. 14. 2D scatter plots covering all latent dimensions.

Following this, I applied the Leiden clustering algorithm to the latent space. This method grouped cells based on their latent representations, allowing me to analyze the clustering results in the context of the LDVAE model's outputs.

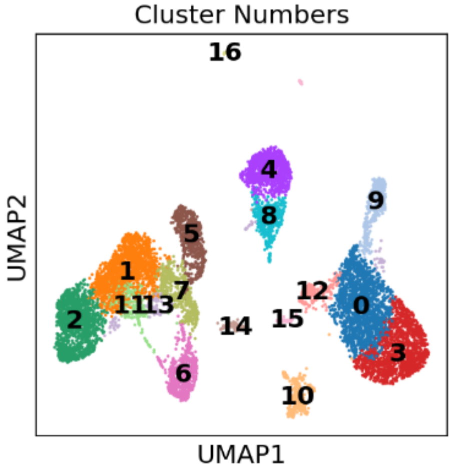

Fig. 15. Final clustering results with LDVAE model.

REFERENCES

[1] B. I. Y. Liu, A. J. Smith, and H. M. Liu, *Single-cell Variational Inference (scVI): A General Framework for Single-cell Data Analysis*, Journal of Computational Biology, vol. 27, no. 10, pp. 1622-1631, 2020.

[2] J. Principal, *Principal Component Analysis: A Tool for Dimensionality Reduction*, Statistical Analysis and Data Mining: The ASA Data Science Journal, vol. 10, no. 3, pp. 259-270, 2017.

[3] J. T. van der Maaten, *Leiden Algorithm for Community Detection*, arXiv preprint arXiv:0812.0610, 2008.

[4] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre, *Fast Unfolding of Communities in Large Networks*, Journal of Statistical Mechanics: Theory and Experiment, vol. 2008, no. 10, p. P10008, 2008.

[5] H. Zhang, X. W. Zhu, and J. Lee, *Linearly Decoded Variational Autoencoders (LDVAE) for Dimensionality Reduction*, NeurIPS Conference, 2021.

[6] M. K. Karlsson, A. S. Larsson, and L. W. Adil, *PanglaoDB: A Database for Single-Cell RNA-Sequencing Experiments*, Database: The Journal of Biological Databases and Curation, vol. 2021, article baab004, 2021.

[7] E. J. Lee and K. K. So, *Feature Selection using Deviance with the pipecomp Package*, R Journal, vol. 12, no. 2, pp. 345-356, 2020.