Fall 2024 Final Report

Johnson & Johnson:

Comparative Evaluation of Tools for Single-Cell RNA Sequencing Data Analysis

Liwen Peng (lp3038), Julia Mengxuan Yu (jy3351), Yiran Dong (yd2730)

Executive Summary

This project compares advanced tools for analyzing single-cell RNA sequencing (scRNA-seq) human brain data, specifically focusing on Scanpy in Python and Seurat in R. Given the complexity of biological datasets, selecting the most suitable analysis tool is crucial for accurate and efficient data interpretation. Our comparison evaluates the algorithms, performance, and compatibility of Scanpy and Seurat to determine the optimal solution for scRNA-seq analysis.

We provide a detailed understanding of each tool's advantages and limitations to help identify those best suited for specific types of multi-omic data. Our findings show that Scanpy is more efficient in both time and memory than Seurat, making it a preferable choice for most scRNA-seq analysis steps, particularly due to its additional clustering capabilities. This project addresses the current gap in comprehensive tool comparisons, as many researchers rely on familiar tools like Seurat without considering potentially more efficient alternatives.

Using a common workflow—including data normalization, quality control, and clustering—we evaluated the effectiveness of three clustering algorithms: Leiden, Louvain, and Phenograph. Our goal was to identify robust clustering methods that accurately capture biologically relevant cell populations for downstream analyses. Both Phenograph and Leiden demonstrated high levels of consistency, indicated by high Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) scores, suggesting these clustering methods can provide reliable groupings for further analysis.

To assign biological identities to the clusters, we employed both automated annotation tools and hybrid approaches. Due to the scarcity of high-quality human brain reference datasets, we aim to randomly select a subset of 50,000 cells from the dataset and then utilize the cluster-based annotation method scCATCH in R to generate a reference file. The scCATCH method leveraged a tissue-specific reference database (CellMatch) to identify marker genes and annotate cell types relevant to human brain tissues. Although the scCATCH annotations provided an initial overview, they showed limitations such as mixed or ambiguous labels and variable confidence scores. This variability, along with the presence of unannotated clusters, highlighted the need for further refinement using high-quality reference datasets and additional validation tools and showed that the current data is not yet suitable as a reference file.

Based on our findings, Leiden and Phenograph agree on clustering results, which can be used to identify consensus clusters for annotating cell types, exploring biological pathways, and ultimately contributing to a deeper understanding of cellular diversity. Scanpy in Python is more computationally efficient. Even though we produced a reference file with scCATCH, the scCATCH annotations require further refinement in the future. In conclusion, the insights from this study provide valuable knowledge to the scientific community, helping to inform better decision-making and promoting the use of optimal tools for single-cell data analysis.

Table of Contents

1. Introduction

1.1 Background

The mammalian brain comprises billions of neurons and glia capable of executing highly complex behaviors. These cells are organized into several major functional regions, each with distinct developmental origins. While the cerebral cortex is the most studied region due to its role in cognition, other brain regions are equally essential.(Siletti et al., 2023)

scRNA-seq technologies enable us to unravel the complexity and heterogeneity of RNA transcripts at the level of individual cells, revealing the composition and functions of different cell types within highly organized tissues. This makes scRNA-seq particularly valuable for understanding the intricate cellular diversity and functional organization of the human brain. (Jovic, Dragomirka, et al., 2022) However, the high-dimensional nature of the data means computational complexity.

With the growth of scRNA-seq technologies, there has been a proliferation of tools designed to analyze complex data. Tools like Seurat, Scanpy, and newer options such as BPCells have enabled researchers to conduct powerful analyses across a wide range of single-cell datasets. However, the lack of comprehensive comparative studies complicates the selection of the most appropriate tool for specific research needs, especially when working with large datasets involving millions of cells.

This project, sponsored by Johnson & Johnson, aims to address this gap by providing a thorough comparison of these tools, focusing on their performance, scalability, and compatibility across various analytical steps, specifically for human brain data.

1.2 Problem Statement

This project aims to evaluate Seurat, Scanpy, and BPCells for scRNA-seq data analysis, focusing on handling large datasets of human brains efficiently. By comparing key functionalities, such as preprocessing, clustering, annotation, and integration, we aim to identify the strengths and limitations of each tool while also exploring the biological essence behind the data. This will provide valuable insights for researchers seeking the optimal solution for analyzing complex biological data, particularly at scale.

1.3 Overall approach

We began by studying biological concepts and scRNA-seq analysis using papers and tutorials on Scanpy and Seurat. The workflow includes data preprocessing and quality control, cell clustering, and cell type annotation. We used Scanpy to present insights for its scalability and unique built-in functions, while Seurat was used for the comparison of time and memory usage with Scanpy at each stage to compare tool performance.

2. Methods

2.1 Data

2.1.1 Data Preparation

To better understand scRNA-seq analysis, we began by exploring how the data is retrieved, processed, and prepared for downstream analysis. Although our primary focus is on scRNA-seq analysis, we also explored the data preparation steps for bulk RNA-seq to gain a comprehensive background.

FASTQ files in RNA-seq analysis are text files containing raw sequencing data from high-throughput machines. Our first task was to convert these FASTQ files into count matrices, filtering for high-quality, uniquely identifiable reads. We identified different pipelines for this conversion for both bulk RNA-seq and scRNA-seq. These pipelines involve multiple tools for quality control, alignment, and quantification, ultimately producing gene expression matrices for downstream analysis.

- **Bulk RNA-seq Pipeline**: FASTQ → FastQC → Trimmomatic → HISAT2 → featureCounts → Count Matrix
- **scRNA-seq Pipeline**: FASTQ → Index-Hopping Filter → Cell Ranger → Count Matrix

We selected some commonly used tools for each step and explored with FASTQ files "10X376-5_S19_L001-001.fastq.gz" and "10X376-4_S18_L001-001.fastq.gz" (NEMO Archive, 2023).

Pipeline 1

The tools needed here for bulk RNA-seq included FastQC v0.12.1, Trimmomatic v0.39, HISAT2 v2.2.1, featureCounts v2.0.6, and human GRCh38 for the reference genome. The first step was to download and add them to the PATH by updating the "./~bashrc" file. By doing this, we could access the executable from any terminal session without specifying the full path to the executable. Then, we ran FastQC to obtain a sequencing data quality report for insights into removing low-quality bases from the end reads:

```
fastqc data/10X376-5_S19_L001_R2_001.fastq -o data/
```

The FastQC would generate an ".html" file along with the ".fastq" file, where we could visualize sequencing data quality shown in Figure 2.1.1 (a). Since there were no sequences flagged as poor quality, we skipped the Trimmomatic step. However, if there were low quality bases, we could remove the end reads below certain quality scores.

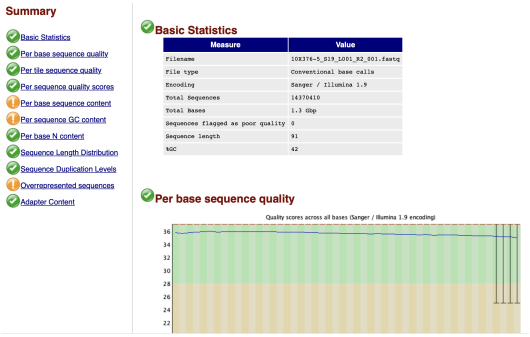
```
java -jar ~/Desktop/demo/tools/Trimmomatic-0.39/trimmomatic-0.39.jar SE
-threads 4 data/demo.fastq data/demo_trimmed.fastq TRAILING:4 -phred33
```

Then, because our data had been generated using a stranded protocol, we ran the below script to align reads to the reference genome.

```
hisat2 -q --rna-strandness R -x HISAT2/grch38/genome -U
data/10X376-5_S19_L001_R2_001.fastq | samtools sort -o
HISAT2/10X376-5_S19_L001_R2_001.bam
```

We got 87.48% reads primarily mapped for the FASTQ file as shown in Figure 2.1.1 (b). Finally, we ran featureCounts to store the count data in 10X376-5_S19_L001_R2_001.txt.

```
featureCounts -S 2 -a hg38/Homo_sapiens.GRCh38.112.gtf -o
quants/10X376-5_S19_L001_R2_001.txt HISAT2/10X376-5_S19_L001_R2_001.bam
```




```
15692720 + 0 in total (QC-passed reads + QC-failed reads)
14370410 + 0 primary
1322310 + 0 secondary
0 + 0 supplementary
0 + 0 duplicates
0 + 0 primary duplicates
13893505 + 0 mapped (88.53% : N/A)
12571195 + 0 primary mapped (87.48% : N/A)
0 + 0 paired in sequencing
0 + 0 read1
0 + 0 read2
0 + 0 properly paired (N/A : N/A)
0 + 0 with itself and mate mapped
0 + 0 singletons (N/A : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

Figure 2.1.1 (a) Sample fasqc.html for S19 (Left); (b) Sample Alignment Output for S19

Pipeline 2

We used cellranger-8.0.1 and index-hopping-filter v1.1.0 to obtain the count matrices, and human GRCh38 for the reference genome. Again, the tools were added to the PATH. We then extracted the ".fastq.gz" files to run the index-hopping-filter, intending to remove likely index-hopped reads from demultiplexed FASTQs. Here are the scripts to get the filtered outputs:

```
index-hopping-filter mkcsv ../original_data 1>config.csv
index-hopping-filter filter config.csv ./data
```

The emitted FASTQs were stored in outs/fastq_path and web_summary.html described the percentage of reads removed per sample:

Figure 2.1.1 (c) Sample web_summary.html for S18 & S19

The next step was to create a library CSV and run cellranger count. The CSV file contained columns fastqs, sample, and library_type, which corresponded to the absolute path to FASTQ file directory, sample name, and a valid library type ("Gene Expression" here). We ran

```
cellranger count --id=s18s19 \
                 --libraries=s18s19_lib.csv \
                 --transcriptome=refdata-gex-GRCh38-2024-A \
                 --create-bam=true
```

and got s18s19/outs/filtered_feature_bc_matrix, which included "barcodes.tsv.gz", "features.tsv.gz", and "matrix.mtx.gz" for further analysts in Scanpy and Seurat.

Summary

After exploring and understanding the process of transforming FASTQ files into count matrices, we decided not to process all the raw data ourselves due to the large volume: more than 2000 .fastq.tar files with the largest exceeding 200 GB in compressed size. Instead, we used the pre-processed dataset "human_adult-GRCh38-3.0.0.h5ad" (Linnarsson-lab-human, 2023) for downstream analysis. This dataset has already been processed and contains both the count matrix and associated metadata, allowing us to proceed directly with analysis.

2.1.2 Data Description

The data was generated from isolating postmortem tissue from three donors and enriching for neurons from approximately 100 locations across the forebrain (the cerebral cortex, hippocampus, cerebral nuclei, hypothalamus, and thalamus), midbrain, and hindbrain (the pons, medulla, and cerebellum). (Siletti et al., 2023) The final dataset we used is an expression matrix generated using the "standard" Cell Ranger and Velocyto pipelines with Cell Ranger GRCh38-3.0.0 annotations in an AnnData format, containing 3,369,219 rows (cells) and 33,538 columns (genes). The annotations follow the CellxGene format, ensuring compatibility for visualization and analysis.

After conducting data preprocessing, we used a subset of the data containing 500,000 cells, allowing us to evaluate performance while working with a more manageable data size before proceeding to the whole dataset, to gain an initial understanding of the dataset and compare the time and memory efficiency of different analysis tools, clustering algorithms and automated annotation methods.

2.2 Data Preprocessing

2.2.1 Quality Control

Quality Control

We controlled for mitochondrial, ribosomal, and hemoglobin genes to remove cells or genes that could introduce bias or noise. Among 33,538 genes, we identified 13 mitochondrial, 104 ribosomal, and 12 hemoglobin genes. Mitochondrial genes are encoded by mitochondrial DNA, and their mRNA abundance can indicate the metabolic state and health of a cell. High mitochondrial RNA content often indicates cell stress or death, leading to overrepresentation of mitochondrial transcripts. In Figure 2.2.1, we found most genes had low mitochondrial expression, indicating good health. However, we filtered out cells with more than 20% of mitochondrial gene expression to ensure better data quality.

Moreover, we checked the number of genes detected per cell ('n_genes_by_counts') and the total count of transcripts (UMI counts) per cell ('total_counts'). Cells with too few or too many detected genes or total counts may represent low-quality cells or doublets, where two cells are captured together. These cells are usually excluded to avoid introducing bias into the dataset.

From Figure 2.2.1 (b), the plot on the number of genes per cell shows a dense region around 500 to 10,000 genes. As a starting point, we decided to filter out cells with less than 100 genes and more than 12,000 genes to refrain from losing information. The plot on the total counts shows cells are mostly under 100,000, and we decided to filter out cells with more than 120,000 total counts. After applying these quality control metrics, there are 3,294,442 cells and 32,603 genes remaining.

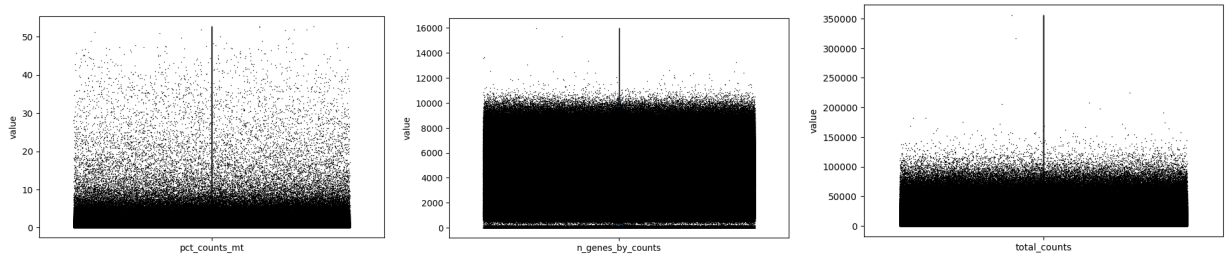


Figure 2.2.1 Violin Plots of with Jitter Points: Percentage of Mitochondrial Genes (Left), Number of Genes by Counts(Middle), Total Count of Transcripts/UMI counts (Right)

Quality Control Metrics

- Cells with less than 20% mitochondrial gene expression.
- Cells with more than 100 genes and less than 12,000 genes.
- Cells with less than 120,000 total counts.
- Gene detected in more than 3 cells (A commonly used threshold on gene)
- Applying conservative thresholds and revisiting when needed

2.2.2 Normalization

We adopted a common approach for normalization in scRNA-seq analysis: count depth scaling with subsequent log plus one transformation. Count depth scaling normalizes each cell's total counts to the dataset median, making expression values comparable across cells. The log transformation manages the highly skewed data distributions common in scRNA-seq datasets.

2.2.3 Feature Selection

To reduce dimensionality and retain the most informative genes, we performed feature selection to identify 2,000 highly variable genes (HVGs). The choice of 2,000 is often based on prior studies or empirical testing. We compared the top 10 HVGs identified by Seurat and Scanpy and found four genes (blue) in common, which exhibit high biological variance across cell types, aligning with our expectations. Despite slight differences in their implementation, both methods fundamentally rely on variance across cells to identify informative genes.

- Scanpy output:
  ENSG00000211899, ENSG00000211895, ENSG00000211592, ENSG00000275302, ENSG00000085563, ENSG00000103888, ENSG00000085276, ENSG00000102755, ENSG00000169245, ENSG00000185532
- Seurat output:
  ENSG00000211592, ENSG00000211679, ENSG00000211677, ENSG00000118271, ENSG00000211895, ENSG00000253755, ENSG00000211899, ENSG00000211896, ENSG00000132465, ENSG00000169245

2.2.4 Dimension Reduction

We performed PCA with 50 components to reduce dimensionality. Using the PCA representation, we computed the neighborhood graph with 15 nearest neighbors to assess biological diversity and technical influences. Batch effects are undesirable as they can mask true biological differences. From Figure 2.2.4, we could only observe a minor batch effect, indicating good data preprocessing, normalization, and potential batch correction.

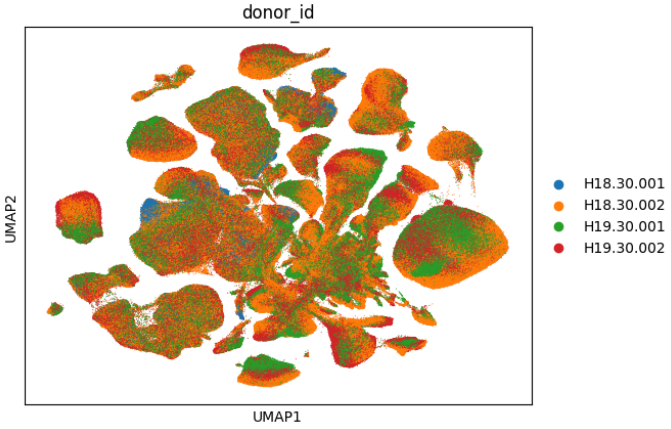


Figure 2.2.4 UMAP Plot on Donor ID

2.3 Clustering Methods

In our mid-term report, we found that Leiden Clustering and Louvain-based Phenograph Clustering produced biologically relevant and consistent clusters. This was indicated by high Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) values in Section 2.3.4. The time and memory efficiency of Leiden Clustering outperformed that of Louvain-based Phenograph Clustering. For this reason, we selected Leiden Clustering to analyze our full scRNA-seq dataset. Leiden Clustering identified about 80 distinct clusters.

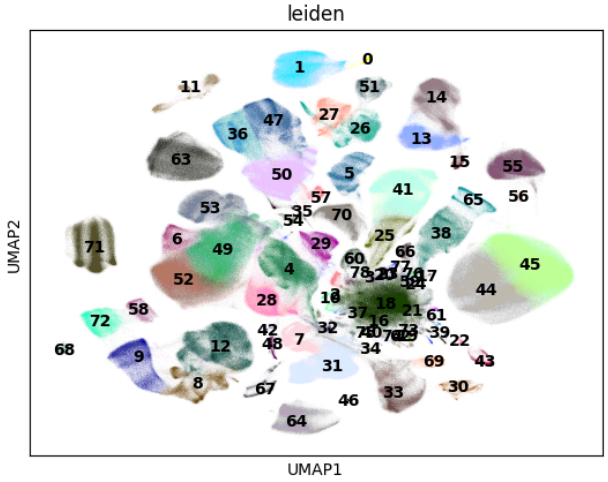


Figure 2.3 Leiden Clustering on Full Dataset

2.3.1 Leiden and Louvain Clustering (Subset)

The Louvain algorithm aims to partition cells into clusters by maximizing modularity, which is a measure of how well the cells are grouped into separate clusters compared to a random assignment, while the Leiden algorithm improves upon it by allowing nodes to move between communities and refining the results for better community detection. Both Louvain and Leiden run on the computed neighbor graph. In Figure 2.3.1, the results are similar, but there are differences; for instance, Leiden recognizes cluster 4 as a single cluster, whereas Louvain splits the corresponding region into two clusters, 4 and 7.

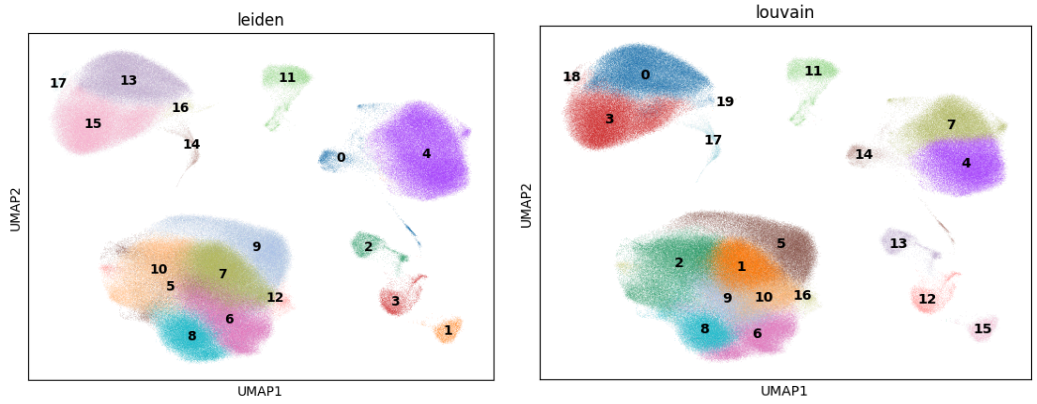


Figure 2.3.1 Leiden Clustering (Left) vs Louvain Clustering (Right)

2.3.2 Phenograph Clustering (Subset)

Phenograph is a clustering method for high-dimensional single-cell data. It creates a graph of phenotypic similarities and uses the Louvain or Leiden algorithm to identify communities. In Scanpy, the phenograph function is independent of sc.pp.neighbors(), so we explicitly set k=15 to control the number of nearest neighbors, consistent with previous clusterings. In Figure 2.3.2, we found that both clusterings identified broadly similar structures, but Phenograph using the Leiden algorithm seemed to provide more refined, smaller clusters.

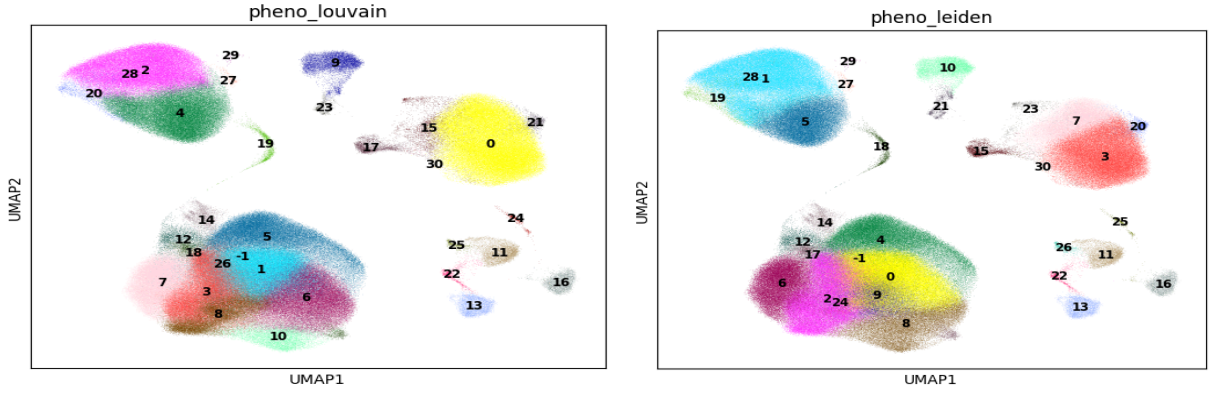


Figure 2.3.3 Phenograph_Louvain Clustering (Left) vs Phenograph_Leiden Clustering (Right)

2.3.3 Hierarchical Clustering (Subset)

The dendrogram illustrates the hierarchical relationships between clusters, with each leaf representing a cluster and branches indicating their similarity. Clusters merged at higher heights are less similar. In Figure 2.3.3 (a), for example, cluster 14 in Leiden's dendrogram is relatively distinct, merging only at a higher height compared to others. And Louvain's dendrogram also identifies this distinct cluster, 17. We notice cluster 14 in Leiden and cluster 17 in Louvain are the same groups of cells. Phenograph clusterings using Louvain and Leiden algorithms have more cluster numbers and also show similar hierarchical relationships.

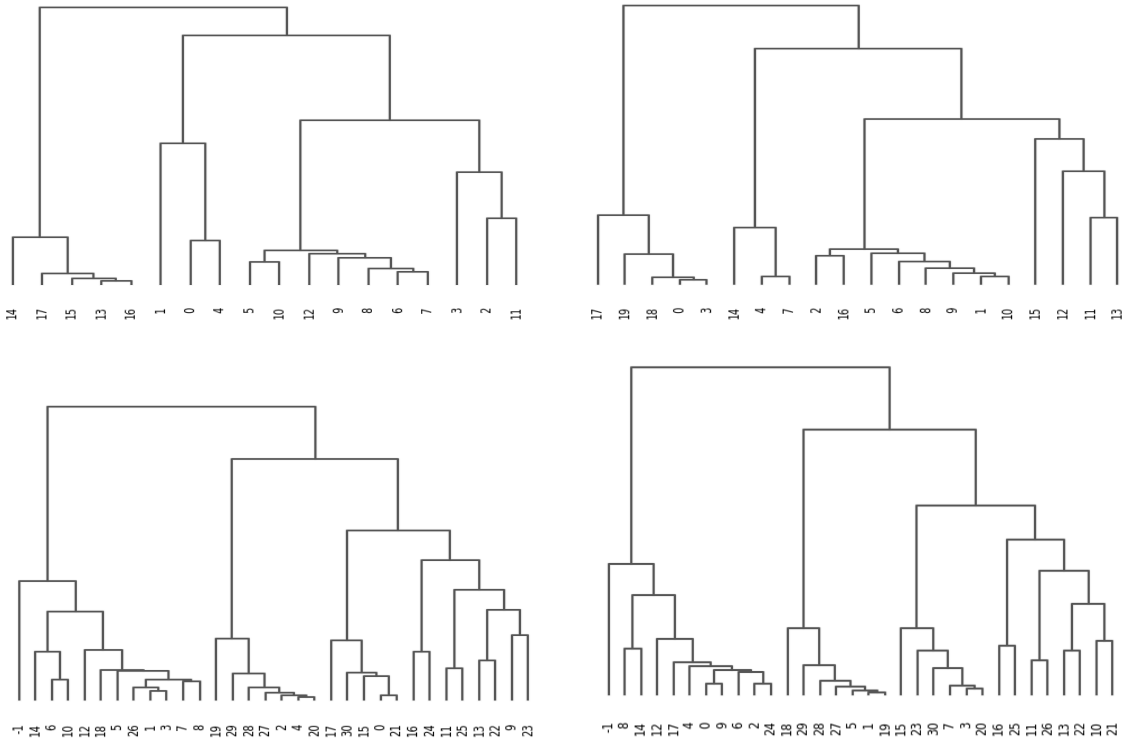


Figure 2.3.3 (a) Hierarchical Clustering: Leiden (top left), Louvain (top right), Phenograph_Louvain (bottom left), Phenograoh_Leiden (bottom right)

2.3.4 Comparing Clustering Methods (Subset)

We first compared the number of clusters identified by each method: Leiden identified 18 clusters; Louvain identified 20 clusters; Phenograph clusterings using the Leiden or Louvain algorithm for community detection both identified 32 clusters. Then we adopted two metrics for comparing the similarities between the clusterings:

- **Normalized Mutual Information (NMI)**: Measures the mutual dependence between two clusterings, with values between 0 (no agreement) and 1 (perfect agreement).
- **Adjusted Rand Index (ARI)**: Evaluates how similar two clusterings are, adjusted for chance, with values between -1 and 1 (with 1 indicating perfect agreement).

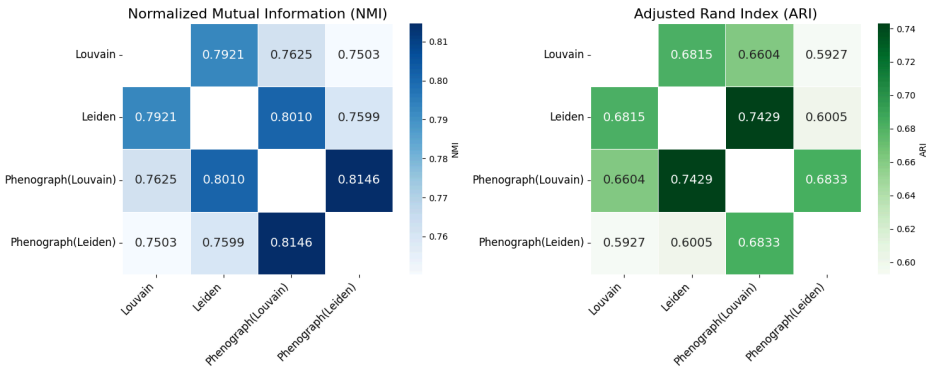


Figure 2.3.4 Heatmaps: Pairwise Clustering Comparison Results

In Figure 2.3.4, all clustering methods show a high degree of consistency, as indicated by the high NMI scores (NMI > 0.75). This suggests that the clustering assignments share a significant amount of mutual information and that the algorithms identify similar global structures within the dataset. Notably, Phenograph (Louvain) and Phenograph (Leiden) have the highest NMI score, indicating strong agreement. The pairwise ARI scores are mostly moderate, implying that while the clustering methods capture shared structures, there are some differences in the detailed point assignments. However, the ARI score between Phenograph (Louvain) and Leiden is high, suggesting robust and highly consistent clustering between these methods.

2.4 Automated Annotation

To assign biological identities to the clusters, we employed both automated annotation tools and hybrid approaches. Specifically, we utilized the cluster-based annotation method scCATCH in R, as well as the semi-supervised deep learning framework scANVI in Python. However, the application of scANVI was initially limited due to the scarcity of high-quality reference datasets for human brain data.

To address this, we tried to produce a reference file ourselves. We randomly selected a subset of 50,000 cells from the dataset and performed annotation using scCATCH in R. This method identifies potential marker genes for each cluster and assigns annotations based on an evidence-based scoring system by matching the identified marker genes with known cell markers curated in the tissue-specific cell taxonomy reference database, CellMatch.

The annotations derived from scCATCH will then be used as ground truth to train scANVI on the subset if it's solid. Once trained, the scANVI model was applied to the entire dataset, enabling robust and comprehensive cell type annotation across all clusters.

2.4.1 scCATCH in R

scCATCH, or the single-cell Cluster-based Automatic Annotation Toolkit for Cellular Heterogeneity, is an automated annotation method in R. After we preprocessed and scaled the scRNA-seq data using the Seurat workflow, we randomly selected a subset of 50,000 cells from the dataset to annotate and create a reference file.

Gene identifiers were mapped from Ensembl IDs to human-readable gene symbols using the Ensembl Biomart database. This step ensured compatibility with downstream analysis tools.

The scaled data and Louvain clustering results were then used in scCATCH. This marker-based annotation tool identified differentially expressed marker genes and assigned biologically relevant cell type annotations. Marker genes were filtered for human-specific and brain-related tissues to match the dataset's biological context. Examples of these tissues include "Brain," "Fetal brain," and "Dorsolateral prefrontal cortex." scCATCH matched cluster-specific marker genes to known cell types using a curated database. This process generated consistent and reproducible annotations. The annotated cell types and associated marker genes were added to the Seurat object's metadata for further analysis and visualization.

In Figure 2.4.1(a), the UMAP plot shows the diversity of cell types identified in the dataset. These include major populations such as Astrocytes, Microglial Cells, Oligodendrocytes, and Progenitor Cells. These findings are consistent with the cellular complexity expected in human brain datasets. Some clusters show mixed or overlapping annotations, like "Astrocyte, Interneuron, Lake Et Al.science…". This could reflect transcriptional similarity between cell types, limitations in clustering resolution, or gaps in the marker gene database used by scCATCH. An unannotated cluster ("NA") is also present. This suggests that some cells either lack distinguishing features or represent cell types not included in the reference database.

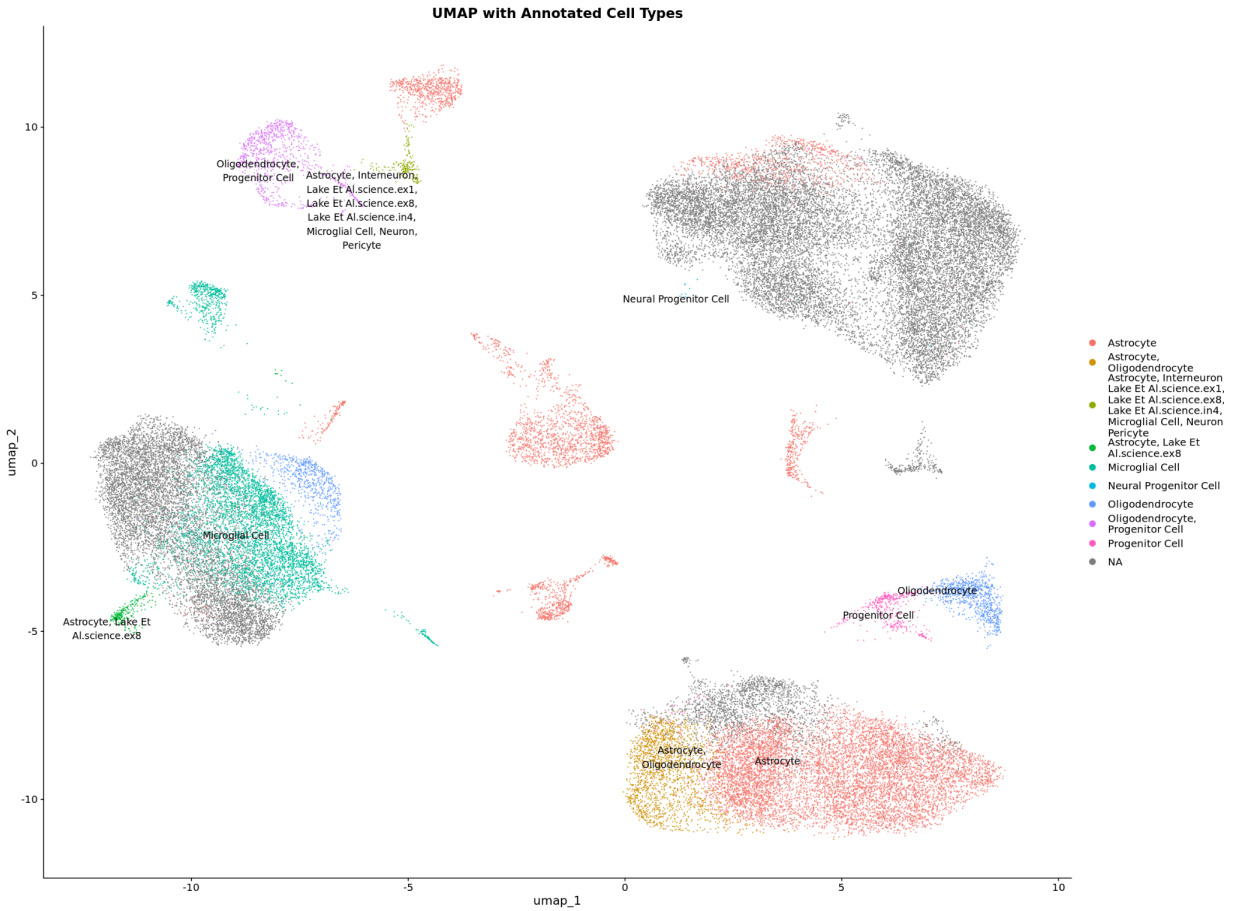


Figure 2.4.1 (a) UMAP with Annotated Cell Type

Then, we further evaluate this annotation by inspecting the cell type score, which quantifies the confidence in the cell type assignment for each cluster. The cell type scores in Figure 2.4.1(b) reflect the confidence in cluster annotations, ranging from 0.5 (minimum confidence) to 0.77 (high confidence), with some rows displaying single values and others containing multiple scores separated by commas. This inconsistency in formatting suggests the need for data cleaning, such as splitting multiple scores into separate rows or computing summary statistics like the mean or median for each cluster. Clusters with repeated low scores (e.g., "0.5, 0.5, 0.5") likely indicate ambiguous or weak annotations, possibly due to insufficient marker evidence or poorly resolved subclusters. Conversely, high-confidence clusters (e.g., scores above 0.7) represent well-defined populations with robust marker support.

	celltype_score
1	0.58
2	0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5
3	0.61
4	0.71
5	0.58
6	0.5, 0.5
7	0.5
8	0.5
9	0.73
10	0.65
11	0.5
12	0.65
13	0.58, 0.58
14	0.71
15	0.67
16	0.58
17	0.58
18	0.67
19	0.77
20	0.76
21	0.75, 0.75
22	0.5

Figure 2.4.1 (b) Cell Type Scores

Hence, the annotations generated by scCATCH, while providing a preliminary overview of cell type identities, are not suitable as ground truth labels for downstream analysis due to the limitations we mentioned, reflecting the need for further refinement and validation of the annotations with solid human-brain reference files.

3. Results: Efficiency Comparison

Peak memory usage represents the highest memory demand, while wall time measures the total elapsed time of execution, including all processing and waiting. We used these metrics to compare the time and memory efficiencies of Scanpy and Seurat under consistent settings.

In Table 3, Scanpy demonstrates better time and memory efficiency compared to Seurat for most steps. Seurat requires significantly more memory and longer execution times for tasks such as normalization, feature selection, PCA, and UMAP. Additionally, using Leiden clustering in Seurat requires Python access, as it relies on the Python-based library leidenalg. Scanpy also offers other unique powerful clustering methods, such as Phenograph and hierarchical clustering, making it a potentially better choice overall.

	Scanpy		Seurat	
	Time Efficiency	**Memory Efficiency**	**Time Efficiency**	**Memory Efficiency**
Quality Control	Wall time: 372s	Peak memory usage: 10.7 GB	Wall time: 69s	Peak memory usage: 78.5 GB
Normalization	Wall time: 14 s	Peak memory usage: 21.3 GB	Wall time: 48s	Peak memory usage: 30.8 GB
Feature Selection	Wall time: 50 s	Peak memory usage: 13.4 GB	Wall time: 179s	Peak memory usage: 54.3 GB
Dimension Reduction (PCA)	Wall time: 62s	Peak memory usage: 2.35 GB	Wall time: 344s	Peak memory usage: 145 GB
Nearest Neighbor	Wall time: 304s	Peak memory usage: 1.88 GB	Wall time: 178s	Peak memory usage: 3.13 GB
UMAP	Wall time: 452s	Peak memory usage: 5.52 GB	Wall time: 1052s	Peak memory usage: 14.7 GB
Leiden	Wall time: 103s	Peak memory usage: 1.40 GB	Require Python access	Require Python access
Louvain	Wall time: 403s	Peak memory usage: 1.76 GB	Wall time: 357s	Peak memory usage: 1.37 GB
Phenograph (Leiden)	Wall time: 365s	Peak memory usage: 1.05 GB	NA	NA
Phenograph (Louvain)	Wall time: 837s	Peak memory usage: 1.09 GB	NA	NA

Table 3. Time and Memory Efficiency Comparison: Scanpy vs Seurat

4. Discussion

4.1 Conclusion

Using a subset of the original data, we established the workflow and identified potentially robust clustering methods. After data preprocessing and minimizing batch effects, we adopted three clustering approaches: Leiden, Louvain, and Phenograph (using the Louvain algorithm for community detection). Phenograph (Louvain-based) and Leiden clustering showed relatively high NMI and ARI, suggesting these methods effectively capture biologically relevant and consistent clusters, providing a potentially reliable foundation for downstream analysis such as

cell type annotation. Hierarchical clustering dendrograms indicated similar relationships among the clusters across different methods, allowing us to define consensus clusters that are robust across multiple clustering techniques for future analyses.

We compared Scanpy and Seurat efficiency by running code on a VM instance with 96-core, 614 GB memory, and 500GB disk. Although Seurat performed well with Louvain clustering, showing competitive runtimes and memory usage in this specific task, Scanpy is generally more efficient in both time and memory in broader scRNA-seq analysis steps, making it a potentially preferable choice for most workflows, particularly with its additional clustering capabilities.

For assigning accurate biological identities to clusters within this scRNA-seq dataset, we aimed to produce a reliable reference file by annotating a randomly selected subset of 50,000 cells using scCATCH, a marker-based annotation tool in R, given the lack of high-quality reference datasets for human brain data.

scCATCH generated preliminary annotations for a subset of 50,000 cells using a curated tissue-specific reference database to identify marker genes and assign biologically relevant cell types. The results identified key populations such as Astrocytes, Microglial Cells, and Progenitor Cells, reflecting the cellular complexity of human brain datasets. However, the annotations had limitations such as ambiguous or overlapping labels, variable confidence scores, and unannotated clusters. These shortcomings indicate that the annotation results are not reliable enough to serve as definitive ground truth labels.

4.2 Future Work

The next step is to enhance the robustness and accuracy of cell type annotations and address the limitations observed in the current workflow. Several key areas for future work have been identified:

1. **Manual Annotation and Refinement of scCATCH Annotations**: Due to the limitation of the human-brain reference file, incorporating manual annotation can help refine ambiguous clusters and resolve overlapping labels identified in the scCATCH annotations. Then, we could use the result for further annotation on the whole dataset. Besides, the annotations produced by scCATCH can be further improved by revisiting marker gene selections, incorporating additional tissue- and species-specific references, and improving clustering resolution to better define subpopulations. Iterative refinement of these annotations will help mitigate inconsistencies and strengthen their reliability for downstream analysis.
2. **Development of a Comprehensive Human Brain Reference Dataset**: The absence of a robust reference for human brain cell types was a key bottleneck in this study. Future efforts should prioritize compiling or integrating a high-quality, context-specific reference dataset.

3. **Evaluation of Additional Annotation Tools:** Beyond scCATCH, exploring complementary annotation tools, such as scANVI, CellTypist, or SingleR, could provide alternative perspectives and improve annotation reliability through cross-validation and consensus-building among methods.
4. **Validation of Annotation Accuracy**: To evaluate the annotation accuracy, we will cross-reference clusters and marker genes in the object with known marker genes. By leveraging well-characterized marker gene sets, such as those from PBMC (Peripheral Blood Mononuclear Cells), we can validate the cell type annotations based on established markers in scRNA-seq studies.

By addressing these areas, the workflow can be significantly improved to ensure more accurate and reliable cell type annotations, paving the way for deeper insights into the cellular diversity and functional biology of human brain datasets. These refinements will also enhance the reproducibility and applicability of this approach to other scRNA-seq studies.

4.3 Ethical Consideration

Since we are using a public dataset and do not intend to share the output with others, there are minimal concerns regarding data privacy. Besides, the analysis is strictly for internal research purposes, mitigating any potential ethical impact of the results.

References

1.Jovic, Dragomirka, et al. “Single‐Cell RNA Sequencing Technologies and Applications: A Brief Overview.” *Clinical and Translational Medicine*, vol. 12, no. 3, 2022, https://doi.org/10.1002/ctm2.694.

2. Siletti, Kimberly, et al. “Transcriptomic Diversity of Cell Types across the Adult Human Brain.” *Science*, vol. 382, no. 6667, 2023, https://doi.org/DOI: 10.1126/science.add7046.

3. “10X376-5_S19_L001-001.Fastq.Gz [Raw Data].” *NEMO Archive*, 31 Jan. 2023, data.nemoarchive.org/biccn/grant/u01_lein/linnarsson/transcriptome/sncell/10x_v3/human/raw/.

4. “Human_adult_GRCh38-3.0.0.H5ad [Final Dataset].” *Linnarsson-lab-human*, 14 Jul. 2023, console.cloud.google.com/storage/browser/linnarsson-lab-human;tab=objects?authuser=0&prefix=&forceOnObjectsSortingFiltering=false.