Single-Cell RNA-seq Analysis Report

1. Introduction

This report describes the initial steps in the analysis of a single-cell RNA sequencing (scRNA-seq) dataset, focusing on data loading, quality control, and doublet removal.

2. Setup and Data Loading

2.1. Library Import and Settings

The analysis was conducted using the Scanpy (1.10.4), and scvi-tools libraries in Python. Necessary Python libraries are imported. For single-cell RNA-seq analysis, the Scanpy and scvi-tools libraries have been used. Also, this single-cell RNA-seq data is loaded from an h5ad file.

Terminal Output:

Libraries imported and Scanpy settings configured.
scanpy==1.10.4 anndata==0.11.3 umap==0.5.7 numpy==2.1.3 scipy==1.15.1 pandas==2.2.3 scikit-learn==1.6.1 statsmodels==0.14.4 python-igraph==NA leidenalg==NA

2.2. Data Loading

The dataset was loaded from the H5AD file:
SCP2745_high_conf_CAS_cell_types.h5ad

The data is transposed (.T) so that cells are represented in rows and genes in columns.

Terminal Output:

Successfully loaded: SCP2745_high_conf_CAS_cell_types.h5ad
AnnData object summary:
AnnData object with n_obs × n_vars = 4000 × 33538
The dataset comprises 4000 cells and 33538 genes/features.

2.3. Initial Data Matrix Inspection

The data matrix `adata.X` has a shape of (4000, 33538).
adata.X.shape: a command used within the Scanpy library to determine the dimensions of the gene expression matrix within an AnnData object. It refers to the gene expression data within the AnnData object. This data is typically in the format of a numpy array or a sparse matrix.

In this matrix, rows represent cells and columns represent genes.

```
matrix([[ 564.],
        [1192.],
        [3709.],
        ...,
        [2012.],
        [ 599.],
        [1409.]], dtype=float32)
```

3. Quality Control (QC) and Filtering

3.1. Gene Filtering

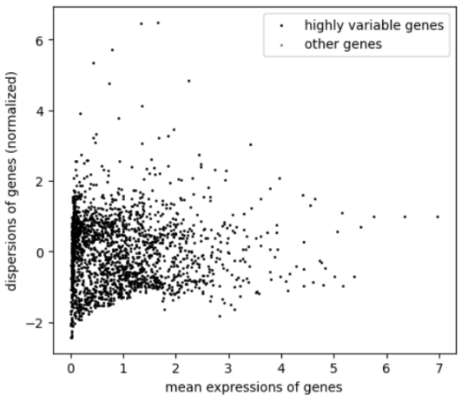
Genes expressed in fewer than 10 cells were filtered out. A copy of the original data was stored in “adata.raw”.

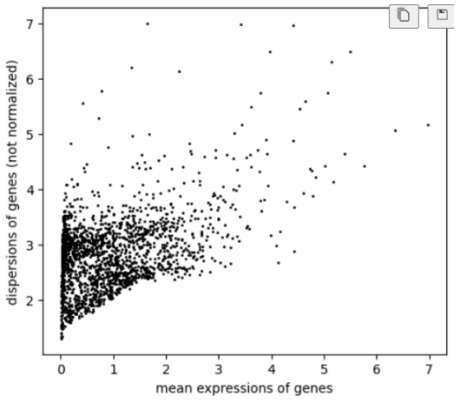
Terminal Output:

(Output of sc.pp.filter_genes not directly captured, but adata is modified in place)

3.2. Highly Variable Gene Selection

Highly variable genes were selected using the `seurat_v3` method.
This plot shows the highly variable genes identified in a single-cell RNA sequencing (scRNA-seq) dataset. It helps to visualize which genes exhibit the most significant variation across the different cells.
Left Panel: This plot shows the dispersions of genes (normalized) against their mean expressions of genes.
The x-axis represents the average expression level of each gene across all cells. The y-axis represents the normalized dispersion (a measure of variability relative to the mean) of each gene across all cells.
The points in the scatter plot represent individual genes.
Genes labeled as "highly variable genes" are those that show higher variability than expected based on their average expression level
Right Panel: This plot likely shows a similar relationship but with the dispersions of genes (not normalized) on the y-axis. Normalization of dispersion helps to account for the relationship between mean expression and variance (higher expressed genes tend to have higher variance).





In essence, this plot is a crucial step in scRNA-seq data analysis for feature selection. By identifying highly variable genes, researchers can focus on the genes that are most likely to drive biological differences between cells and reduce the dimensionality of the data for downstream analyses like clustering and differential gene expression analysis.

4. Doublet Removal

Doublets were predicted and removed using scvi-tools's SOLO method.

Detection of Doublets

Filtering of genes (min_cells=10): Genes expressed in at least 10 cells are kept.
Selection of variable genes (n_top_genes=2000): Analysis is performed for the selected 2000 genes.
SCVI models are collected and doublets are predicted using SOLO.

Removal of Doublets

Results with double predictions are processed.
Removed from the dataset marked as double.

scvi-tools Installation and Usage

- The scvi-tools library is installed. scvi-tools is a Python package for probabilistic modeling and analysis of single-cell omics data.
- A variational autoencoder (VAE) model is defined using scvi.model.SCVI(adata).
- An additional model, SOLO, is created to identify doublets (instances where two cells are erroneously captured as one).
- Vae.train() is executed to reduce noise in the data and correct for batch effects, enabling the model to learn a statistically meaningful representation of cells.

Doublet Detection

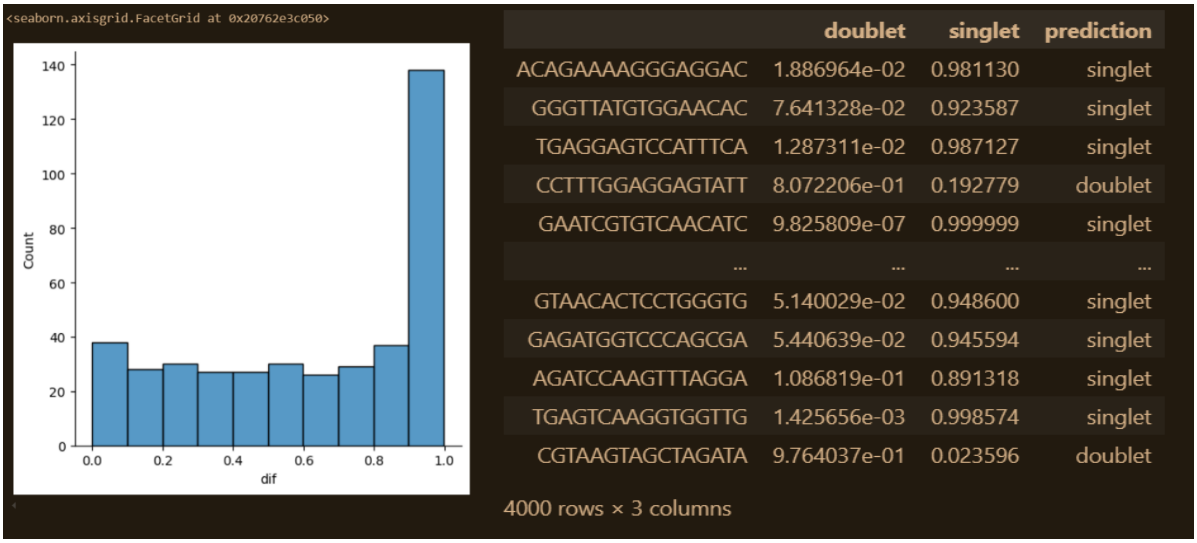
- Gene filtering (min_cells=10): Genes expressed in fewer than 10 cells are removed.
- Highly variable gene selection (n_top_genes=2000): The 2000 most variable genes are selected for downstream analysis.

4.1. scVI Model Training and Prediction

An scVI model was trained on the filtered data.

Furthermore, SOLO model was trained to predict doublets, and then predictions were made.
The SOLO model training stopped early due to lack of improvement in validation loss and classified cells as singlets or doublets.

Terminal Output:



	doublet	singlet	prediction
ACAGAAAAGGGAGGAC	1.886964e-02	0.981130	singlet
GGGTTATGTGGAACAC	7.641328e-02	0.923587	singlet
TGAGGAGTCCATTTCA	1.287311e-02	0.987127	singlet
CCTTTGGAGGAGTATT	8.072206e-01	0.192779	doublet
GAATCGTGTCAACATC	9.825809e-07	0.999999	singlet
...	...	...	...
GTAACACTCCTGGGTG	5.140029e-02	0.948600	singlet
GAGATGGTCCCAGCGA	5.440639e-02	0.945594	singlet
AGATCCAAGTTTAGGA	1.086819e-01	0.891318	singlet
TGAGTCAAGGTGGTTG	1.425656e-03	0.998574	singlet
CGTAAGTAGCTAGATA	9.764037e-01	0.023596	doublet

4000 rows × 3 columns

4.4. Doublet Summary and Impact of Doublet Filtering

A total of 327 cells were predicted to be doublets, while 3673 were classified as singlets.

The doublet filtering step reduced the number of cells from 4000 to 3673, focusing subsequent analysis on high-quality singlets. This represents a removal of 327 cells identified as doublets.

5. Dimensionality Reduction

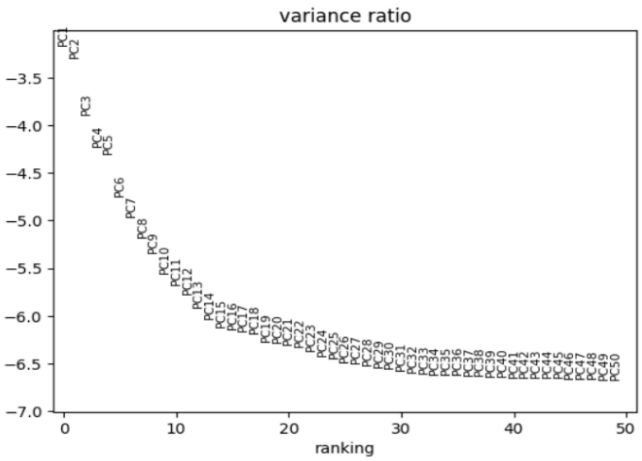
5.1. Principal Component Analysis (PCA)

PCA was performed on the filtered data to reduce dimensionality.

Cells are normalized to a total of 10,000 counts.

Data is log-transformed. The original data is saved as adata.raw.

This plot shows the variance the variance explained by each principal component. It is used to determine how many PCs capture most of the biological signal and should be used for downstream steps like UMAP and clustering.



The PCs are ranked on the x-axis, and the logarithm of the variance ratio (presumably the ratio of variance explained by each PC to the total variance) is on the y-axis. To determine the number of PCs to retain for further analysis, an "elbow point" is typically sought where the rate of decrease in explained variance slows down considerably. Before this point, there is a relatively steep drop in the variance explained by each successive PC.

Preprocessing

1)Mitochondrial and Ribosomal Gene Marking

Mitochondrial genes are labeled, often by identifying gene names starting with "MT-".

Ribosomal genes are marked by matching against a list of known ribosomal genes.

2) Quality Control (QC) Metric Calculation

Quality control metrics for cells are calculated (e.g., percentage of mitochondrial genes).

3) Cell and Gene Filtering

Based on example thershold (which should ideally be adjusted based on the QC plots):

Filtering genes (min_cells=10): Genes expressed in at least 10 cells are kept.

Selecting variable genes (n_top_genes=2000): The 2000 most variable genes are selected for analysis.

The SCVI model is trained and doublets are predicted using SOLO.

Genes expressed in a small number of cells are removed. Cells with a high number of genes or a high percentage of mitochondrial/ribosomal genes are removed.

5.2. UMAP Embedding

UMAP embedding was computed for visualization of the cells in a 2D space.

Terminal Output:

computing UMAP finished: added 'X_umap', UMAP coordinates (adata.obsm) (0:00:07)

UMAP coordinates were computed and stored in `adata.obsm['X_umap']`.

6. Neighborhood Graph and Clustering

6.1. Neighborhood Graph

A neighborhood graph was computed based on the PCA representation of the cells .Furthermore it was computed based on by using scanpy.pp.neighbors() .

This step utilized the top 30 principal components (a common choice, but should be informed by the PCA elbow plot) and considered 30 neighbors for each cell.

6.2. Leiden Clustering

Cells were clustered using the Leiden algorithm, based on the neighborhood graph. Cells were clustered using the Leiden algorithm **scanpy.tl.leiden()** on the computed neighborhood graph.

a. PCA and Neighborhood Calculation

Principal Component Analysis (PCA) is performed for dimensionality reduction.

A neighborhood graph is constructed.

b. UMAP and Leiden Algorithm

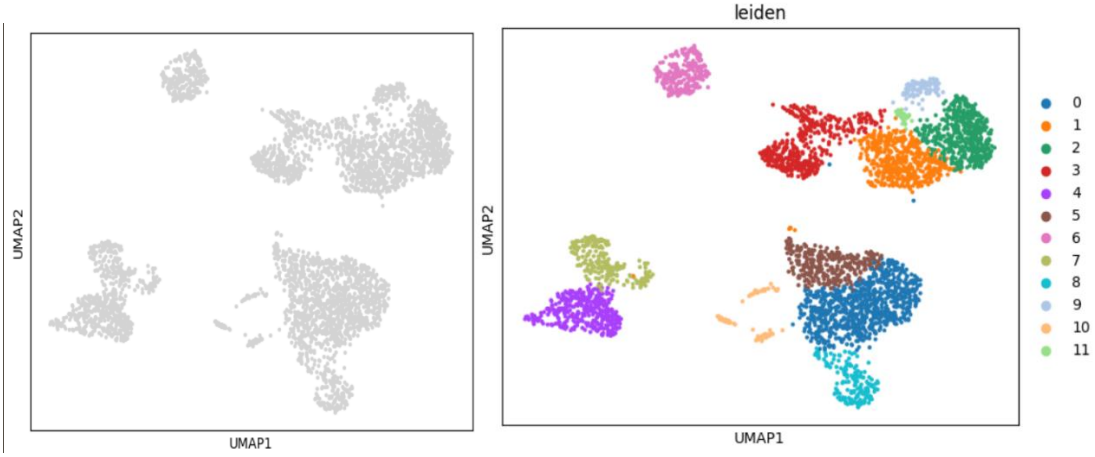
UMAP (Uniform Manifold Approximation and Projection) is used for data visualization.

The Leiden algorithm is employed to identify cell clusters. The Leiden algorithm is a community detection algorithm used to find clusters in large networks.

7. Visualization

7.1. UMAP Visualization of Clusters

The clustered data was visualized on a UMAP plot, where each point represents a cell and is colored according to its Leiden cluster assignment. sc.pl.umap(adata) and sc.pl.umap(adata, color=['leiden']) functions have been used.



(This UMAP plot visualizes the cellular heterogeneity in the dataset by embedding cells in a two-dimensional space based on their principal components, with cells colored and labeled according to their assigned Leiden clusters.

Cluster Separation: Overall, the clusters appear reasonably well-separated in the UMAP space. Distinct groupings of cells corresponding to different cluster labels (0 through 11) are noticable.

Approximate Size Ranking (Largest to Smallest, Based on Visual Area):

Cluster 0: Situated in the top right quadrant, this dark blue cluster also covers a significant area.

Cluster 1: Located in the bottom right quadrant, this light blue cluster appears to be the largest and most densely populated.

Cluster 2: In the mid-right section, the orange cluster seems to represent a substantial population.

Cluster 3: Found in the middle of the plot, the green cluster also appears to be of considerable size.

Cluster 9: Located in the lower-middle area, this light blue-greenish cluster appears to be of medium size.

Cluster 7: In the bottom left quadrant, this light blue cluster also seems to have a medium-sized population.

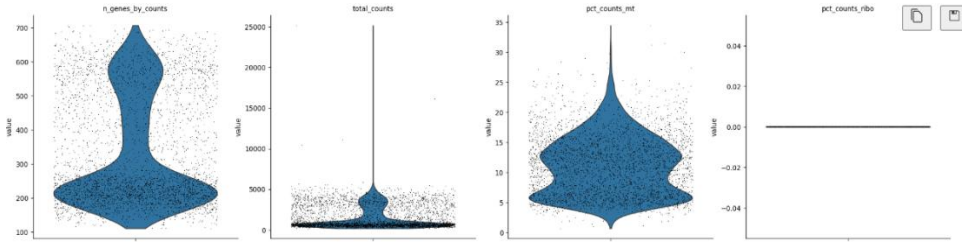
Cluster 8: Situated in the middle, the pink-purple cluster appears to be of medium size as well.

Cluster 4: In the upper part of the plot, this purple cluster is smaller but distinct.

Cluster 6: Located in the mid-left section, the yellow-green cluster seems to represent a smaller population.
Cluster 5: In the upper-middle area, the reddish-orange cluster appears relatively small.
Cluster 10: In the top right, there's a very small, light greenish cluster

These spatial relationships in the UMAP can often reflect biological similarities or developmental trajectories between cell populations.
Dataset: SCP2745_high_conf_CAS_cell_types.h5ad
Analysis Tool: Scanpy, scvi-tools (via Python)

Violin Plots: These plots show the distribution density of detected genes per cell and total UMI counts per cell. They are used to identify the main population of cells and potential outliers (e.g., cells with very few genes or counts, which might be empty droplets or damaged cells, or cells with extremely high counts, which might be doublets).These violin plots show the distribution of two key quality control metrics across all cells: the number of detected genes per cell and the total counts per cell and pct counts mitochondria and ribosomes.



A wide distribution suggests heterogeneity in the complexity of the transcriptomes captured in different cells.
Cells with very low numbers of detected genes might be of low quality (e.g., damaged or empty droplets).
Cells with very high numbers of detected genes might represent doublets (two cells captured together).
n_genes_by_counts (Number of genes detected per cell): The first violin plot shows the distribution of the number of unique genes detected in each individual cell.

total_counts (Total UMI counts per cell): The second violin plot displays the distribution of the total number of Unique Molecular Identifiers (UMIs) or reads counted for each cell. This is a measure of the sequencing depth per cell.

Similar to the number of genes, a broad distribution is expected.
Cells with very low total counts might be of low quality.
Cells with very high total counts could also be doublets.

pct_counts_mt (Percentage of counts mapping to mitochondrial genes): The third violin plot shows the distribution of the percentage of transcripts in each cell that map to mitochondrial genes.

High percentages of mitochondrial reads are often indicative of stressed or dying cells, as their cytoplasmic RNA might be degraded, leaving behind more abundant mitochondrial RNA.
pct_counts_rbo (Percentage of counts mapping to ribosomal genes): The fourth violin plot (although it appears somewhat flat in this image, suggesting very little variation) typically shows the distribution of the percentage of transcripts mapping to ribosomal genes.

This metric can sometimes be used to assess the metabolic state or cell cycle phase of cells.

8.Conclusion and Next Steps

This analysis successfully processed the "SCP2745_high_conf_CAS_cell_types.h5ad" single-cell RNA-seq dataset through a standard Scanpy workflow.
The data was loaded, quality controlled (resulting in 4,000 cells and 33,538 genes), normalized, and key highly variable genes were identified. Dimensionality reduction via PCA (30 components) and UMAP was performed, followed by clustering using the Leiden algorithm to identify distinct cell populations.
Key steps included data preprocessing, normalization, dimensionality reduction, clustering, and differential expression analysis. Using SCVI, latent representations were generated, and UMAP visualizations provided a clear depiction of cell clusters. Furthermore, in the next steps marker genes were identified, enabling accurate cell type annotation. Differential expression analysis and gene ontology enrichment revealed significant genes and pathways associated with specific cell types and conditions.
Additionally, cell type frequencies and gene signature scores were compared across conditions, offering insights into cellular responses. Overall, this workflow highlights the power of single-cell analysis in understanding complex biological systems.