snRNA-seq Clustering

Julissa Zelaya-Portillo

2024-09-24

The GSE accession, GSE138852, maps to a specific dataset focused on single-cell RNA-sequencing (snRNA-seq) of Alzheimer's disease and control samples. In the analysis of this data, Seurat will be used to categorize single nuclei into clusters based on their gene expression heterogeneity. Seurat is an R package designed for exploration, analysis, and quality-control of snRNA-seq data.

Create a Seurat Object

The GSE138852 snRNA-seq data is pulled from the GEO database. This may be done manually from the GEO database, using the GEO REST API, or using command-line tools like `wget` to download the count matrix file. The analysis will begin with the assumption that base calling, mapping, and read counting on the GSE138852 dataset has been done.

The count matrix data is read and a Seurat object is created to allow Seurat to store the steps and results of the forming analysis.

```
# Read the CSV file directly
count_matrix <- read.csv("GSE138852_counts.csv", header = TRUE, row.names = 1)

# Convert the count matrix to a Seurat object
seurat_obj <- CreateSeuratObject(counts = count_matrix)
```

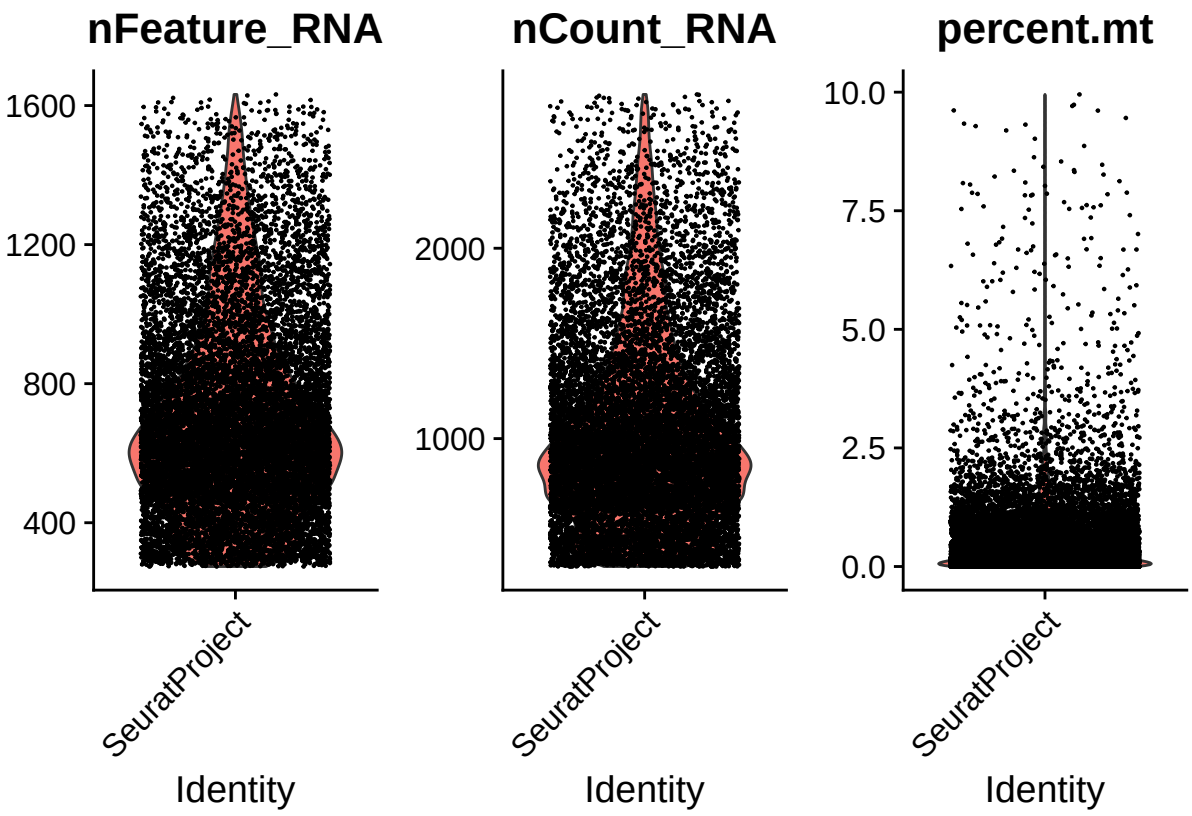
Quality Control

Poor-quality cells such as those with low gene counts or high mitochondrial RNA content are generally filtered out to reduce noise in the data.

Gene counts are summarized by Seurat automatically when the Seurat object is created but the mitochondrial transcript percentage needs to be calculated manually via the `PercentageFeatureSet` function.

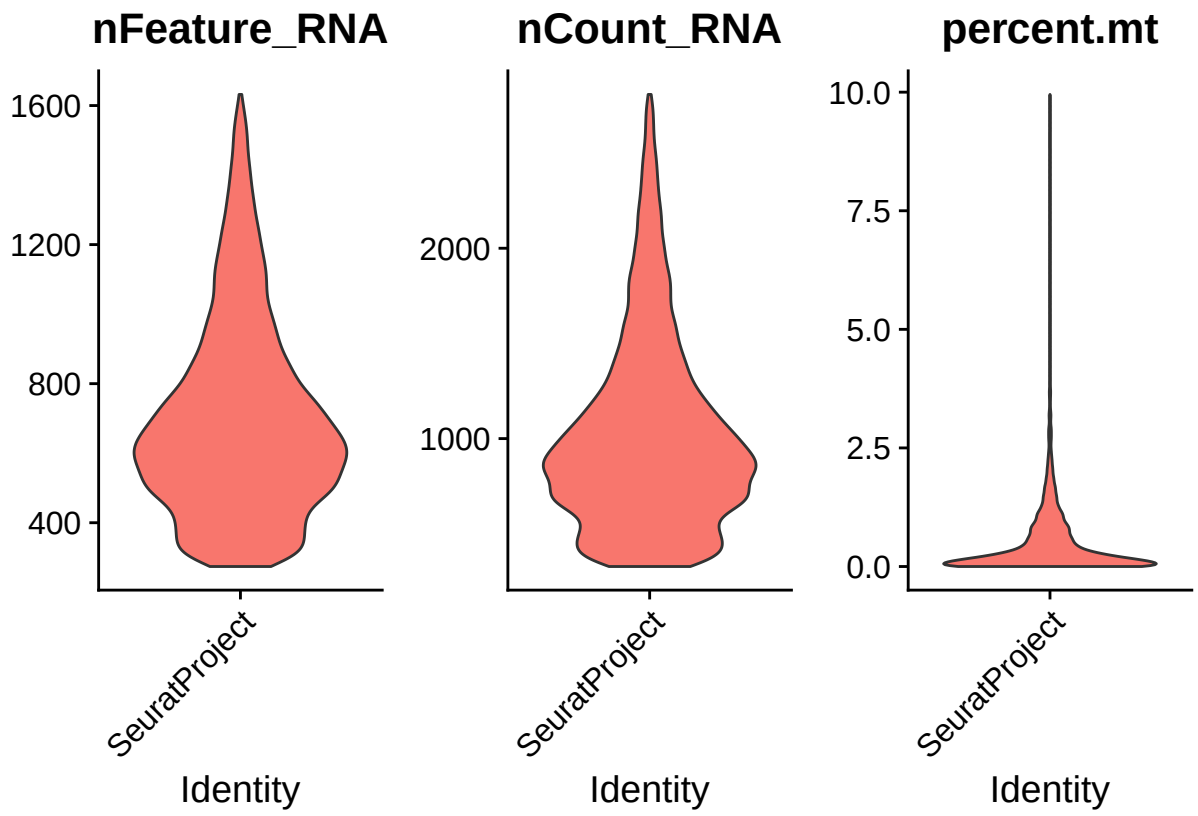
To filter out outlier cells, the distribution is observed by creating a violin plot for the following three metrics: the number of features or distinct genes in a particular cell, the total count of all RNA molecules in that cell, and the percentage of mitochondrial transcript that was previously calculated.

```
# Calculate QC metrics
seurat_obj[["percent.mt"]] <- PercentageFeatureSet(seurat_obj, pattern = "^MT-")

# Plot QC metrics
VlnPlot(seurat_obj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```



Another version of the above plot is shown below, removing the use of dots (show of individual cells).

```
VlnPlot(seurat_obj,
        features = c("nFeature_RNA", "nCount_RNA", "percent.mt"),
        ncol = 3, pt.size=0)
```



Based on common practices, detected gene numbers outside of 200 and 2500 features are removed as they may indicate low RNA content. It is also common to remove cells with greater than 5% mitochondrial transcript. Cells over this threshold may have low-quality cells. A subset of the Seurat data, removing these outliers, then proceeds forward in the working analysis.

```
# Filter based on QC metrics
qc_subset <- subset(seurat_obj, subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mt < 5)
```

Normalization

Normalization aims to make gene expression between cells comparable.

```
## Normalizing layer: counts
```

Feature Selection

Feature selection is performed to identify highly variable features/genes based on the most varied expression level across cells. Seurat calculates the standard variance of each gene across cells and picks the top 2000 as being the highly variable features. This should reduce noise and improve model performance.

```
# Find Variable Features
seurat_feat <- FindVariableFeatures(seurat_norm)
```

```
## Finding variable features for layer counts
```

Data Scaling

Different genes have different base expression levels and distributions. To avoid the analysis depending on genes that are highly expressed, data scaling is performed.

```
# Data scaling
seurat_scale <- ScaleData(seurat_feat)
```

```
## Centering and scaling data matrix
```

Principal Component Analysis (PCA)

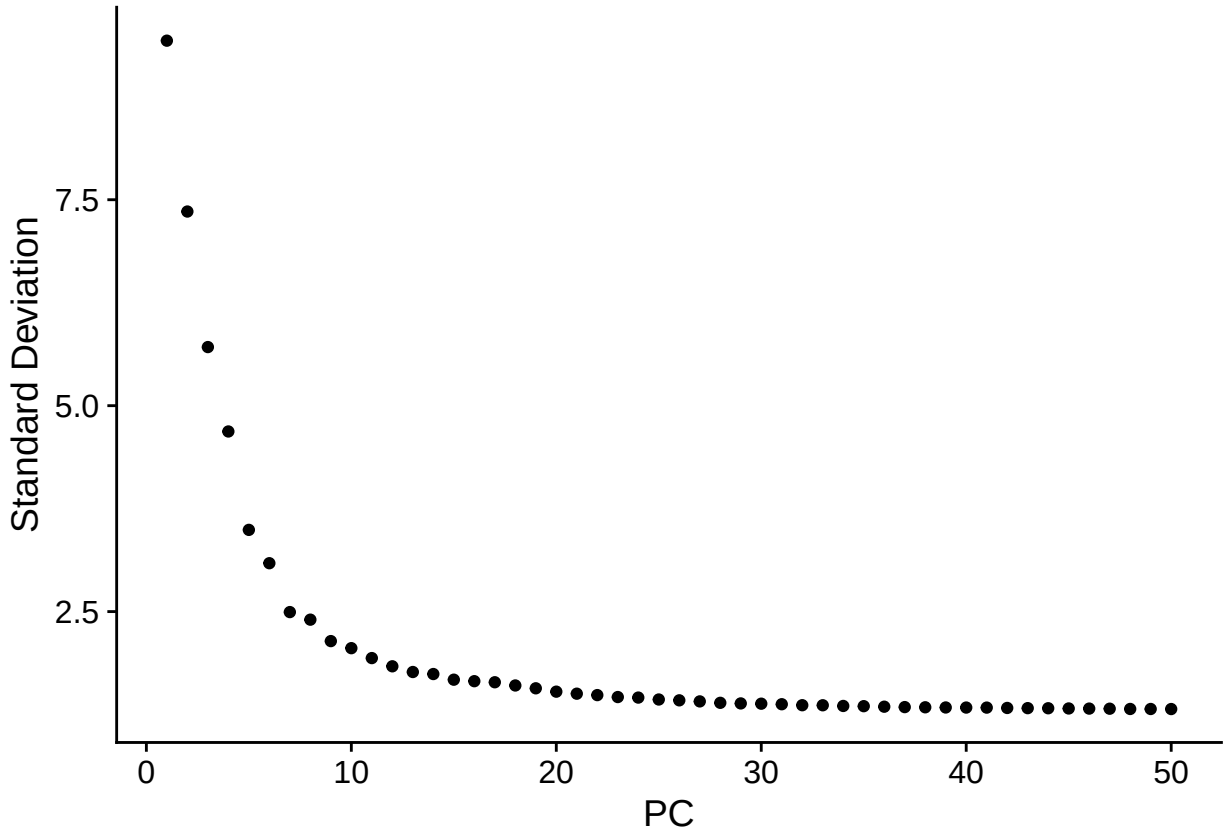
Applying a linear dimension reduction is beneficial to make the data more compact and is here done using Principal Component Analysis (PCA).

```
# PCA
seurat_PCA <- RunPCA(seurat_scale)
```

```
## PC_ 1
## Positive:  ST18, SPP1, LINC01608, CNDP1, KCNMB4, KIRREL3, PALM2, CDH19, SYNJ2, LINGO1
##     SLC5A11, CNTNAP4, PDE1A, ADAMTS18, RP11-81H3.2, BOK, TMTC4, LRP2, RP11-267C16.1, GLDN
##     APOD, VRK2, KLHL4, CHN2, LINC01170, RP11-50D16.4, CD55, HS3ST5, LAMA2, ITGA2
## Negative:  GPM6A, NRG3, NRXN1, RORA, NKAIN3, SLC1A2, DPP10, GPC5, CTNNA2, ADGRV1
##     RP11-384F7.2, DTNA, FAM155A, RNF219-AS1, NEBL, PITPNC1, LSAMP, NRCAM, GABRB1, NTRK2
##     SOX5, FMN2, SPARCL1, PTPRZ1, TRPM3, RYR3, TENM2, CSGALNACT1, CACNB2, ADCY2
## PC_ 2
## Positive:  RNF219-AS1, ADGRV1, RYR3, TPD52L1, GLIS3, AQP4, BMPR1B, EMX2, LINC00499, PRKG1
##     LRRC16A, NEAT1, ETNPPL, SLC1A3, PTGDS, RANBP3L, TRPM3, PAMR1, STON2, GPC5
##     SLC4A4, SFXN5, SLC14A1, ZNRF3, GLI3, AC002429.5, COL5A3, SLCO1C1, SLC1A2, SLC7A11
## Negative:  OPCML, SNTG1, FGF14, CSMD1, DSCAM, PCDH15, KCNIP4, LHFPL3, TNR, NXPH1
##     GRIK2, FGF12, LUZP2, ATRNL1, MMP16, LRRTM4, GRIK1, CA10, GRM7, CSMD3
##     MDGA2, SGCZ, RP4-668E10.4, KCND2, RBFOX1, GRID2, XKR4, RIMS2, SCN1A, GRM5
## PC_ 3
## Positive:  ERBB4, PTGDS, NPAS3, ST18, LRP1B, NTM, LSAMP, APOD, CTNND2, LINC01608
##     NRXN3, CNDP1, NOVA1, KIRREL3, DLC1, SYNJ2, PALM2, KCNMB4, CDH19, HS3ST5
##     CNTNAP4, GALNT13, LINC01170, BOK, PDE1A, FUT9, AC012593.1, LINGO1, ROBO1, ADARB2
## Negative:  DOCK8, RP11-624C23.1, APBB1IP, ADAM28, LPAR6, HS3ST4, ST6GAL1, FYB, ATP8B4, CD74
##     SYK, PTPRC, SRGAP2B, TBXAS1, SRGAP2, CSF2RA, MEF2C, SRGN, C10orf11, A2M
##     INPP5D, ARHGAP24, P2RY12, SAMSN1, ARHGAP15, CSF3R, SLCO2B1, BLNK, CD86, HLA-DRA
## PC_ 4
## Positive:  SYT1, KCNC2, ROBO2, MTUS2, LINGO2, CACNA1B, KCNQ5, SYNPR, FSTL5, FRMPD4
##     CELF4, GABRB2, GRIP1, RP11-123O10.4, WBSCR17, GRIN2A, DLX6-AS1, HCN1, GRIN1, KCNH7
##     CDH9, KCNJ3, ADGRL2, GALNTL6, GRIA1, CHRM3, GABBR2, CNTN5, DCLK1, CCSER1
## Negative:  RP4-668E10.4, LHFPL3, TNR, XYLT1, PCDH15, VCAN, PTPRZ1, MMP16, LUZP2, CA10
##     PDZRN4, MEGF11, SOX6, BRINP3, SEMA5A, DSCAM, MIR3681HG, NOVA1-AS1, CHST11, KCNMB2-AS1
##     SMOC1, LRRC4C, COL9A1, COL11A1, DCC, STK32A, CALCRL, NOVA1, GPC6, LINC00511
## PC_ 5
## Positive:  MALAT1, PDE1A, ST18, LINC01608, KIRREL3, SLC5A11, ZNF565, PALM2, CNTNAP4, NRXN3
##     LINC01170, RP11-81H3.2, SYNJ2, DLC1, XIST, GLDN, ADAMTS18, RP11-267C16.1, HS3ST5, CNDP1
##     TMTC4, AC012593.1, LMCD1-AS1, NPAS3, CNTNAP2, SLC4A8, KCNQ1OT1, ST6GALNAC3, ROBO1, SLC6A1-AS1
## Negative:  HSPA1A, LINGO1, GFAP, MT-ND4, HSPB1, MT-CO3, DNAJB1, HSPA1B, BOK, MT-ATP6
##     AEBP1, MT-CYB, MT-ND2, MT-ND3, MT-ND1, RHPN1, FOS, DHCR24, MT-CO1, HIF3A
##     HSPH1, UBC, ITGB4, CLU, IFITM3, MEG3, VCAN, CD44, PHYHIP, PDZD4
```

To make decisions based on the PCA analysis, the `Embeddings` function constructs an elbow plot to show the standardized variation of all the PCs that are calculated. Higher-ranked PCs have higher standard deviations and explain more variation in the data than lower-ranked PCs.

```
ElbowPlot(seurat_PCA, ndims = ncol(Embeddings(seurat_PCA, "pca")))
```



The "elbow" point or turning point of the curve occurs in the 5th/6th PCs and then becomes flat. The first phase of the curve could represent the signal related to biological differences between cell populations while the second phase could represent technical variations.

Clustering Analysis

First, a k-nearest neighbor of cells is created to create a connection between cells with the shortest distance based on their PC values. With the k-NN network created, the clustering algorithm used by Seurat is applied.

```
# A k-Nearest Neighbor is done through Seurat
seurat_kNN <- FindNeighbors(seurat_PCA)
```

```
## Computing nearest neighbor graph
```

```
## Computing SNN
```

```r
# Clustering analysis
seurat_clustered <- FindClusters(seurat_kNN, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 13096
## Number of edges: 431844
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9256
## Number of communities: 13
## Elapsed time: 2 seconds
```

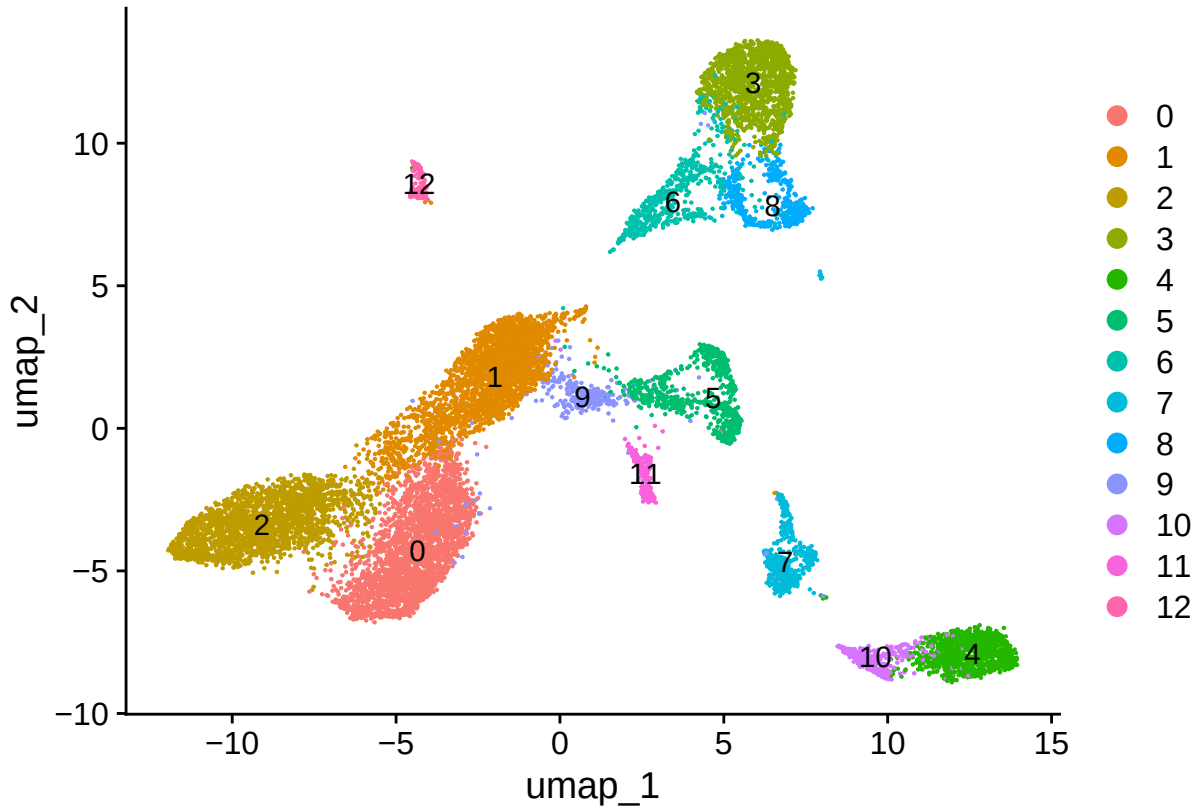
Non-Linear Dimension

The linear dimension that was previously run lacks the three dimensional analysis that is here presented with UMAP (Uniform Manifold Approximation and Projection). Using the clustered analysis, we now have a visual of our cluster networks.

```r
# Run UMPA and then plot UMAP
seurat_umap <- RunUMAP(seurat_clustered, dims = 1:10)
```

```
## Warning: The default method for RunUMAP has changed from calling Python UMAP via reticulate to the R
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric to 'correlation'
## This message will be shown once per session

## 21:14:32 UMAP embedding parameters a = 0.9922 b = 1.112

## 21:14:32 Read 13096 rows and found 10 numeric columns

## 21:14:32 Using Annoy for neighbor search, n_neighbors = 30

## 21:14:32 Building Annoy index with metric = cosine, n_trees = 50

## 0%   10   20   30   40   50   60   70   80   90   100%

## [----|----|----|----|----|----|----|----|----|----|

## **************************************************|
## 21:14:34 Writing NN index file to temp file /var/folders/zy/2dvzl4x52_v5fvrlsgrzn7880000gp/T//RtmpyM
## 21:14:34 Searching Annoy index using 1 thread, search_k = 3000
## 21:14:40 Annoy recall = 100%
## 21:14:40 Commencing smooth kNN distance calibration using 1 thread with target n_neighbors = 30
## 21:14:41 Initializing from normalized Laplacian + noise (using RSpectra)
## 21:14:42 Commencing optimization for 200 epochs, with 554622 positive edges
## 21:14:51 Optimization finished
```

```r
# Visualize UMAP
cluster_plot <- DimPlot(seurat_umap, reduction = "umap", label = TRUE)
cluster_plot
```

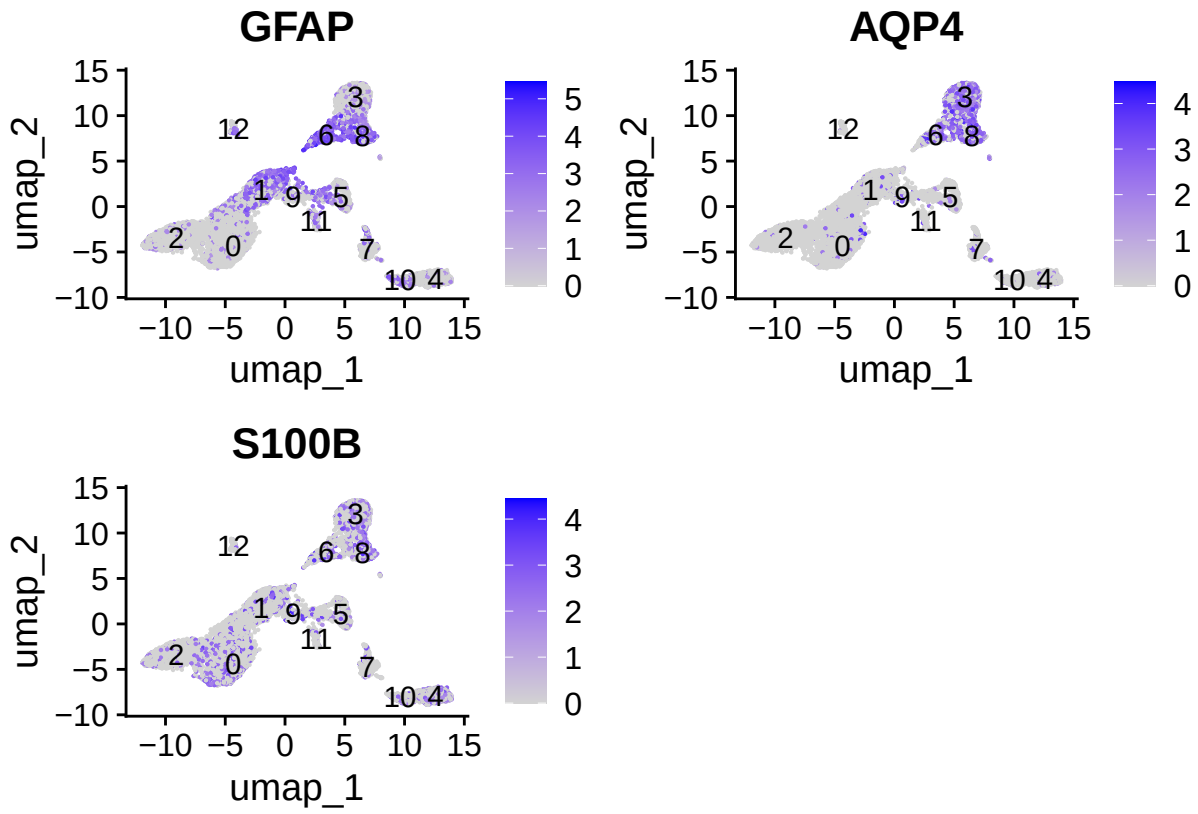


The results of the non-linear dimension analysis can then be visualized. From here it is evident that there are 12 distinct clusters. Areas of minimal overlap such as clusters 0, 1, and 2 suggest strong heterogeneity between cell populations. Due to their large size, they may also represent major cell populations. On the other hand, cluster 12 appears as a small outlier which would require further analysis to determine if there is additional technical noise that could be resolved during our data quality process.

Cluster Annotation

To later isolate the astrocytes from the dataset and make more specific analyses of the clustering and later sub-clutering results, annotation of cell clusters is performed. Typical markers of astrocytes are the following labels: GFAP / AQP4 / S100B. First the expression of known astrocyte markers are plotted to determine which clusters are likely to be astrocytes.

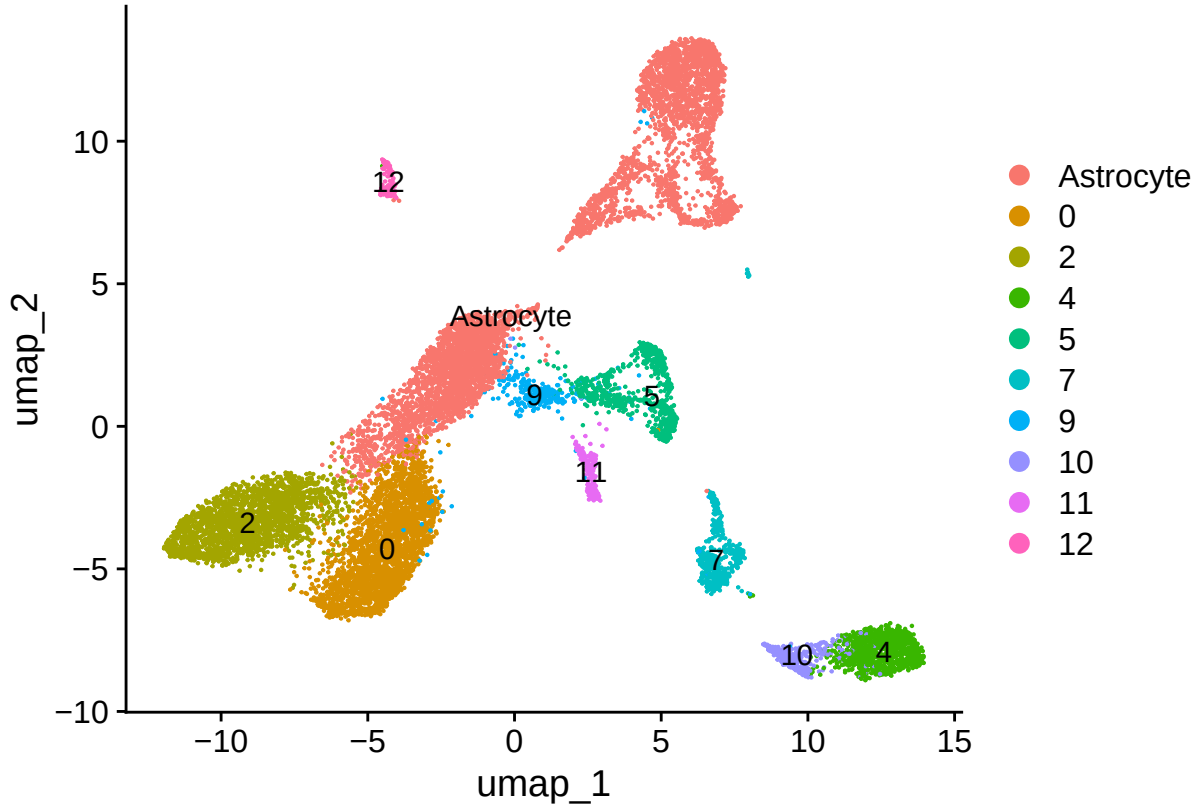
```
# Visualize the expression of astrocyte markers
FeaturePlot(seurat_umap, features = c("GFAP", "AQP4", "S100B"), label = TRUE)
```



Based on the subjective analysis of the above plot, it is determined that clusters 1, 3, 6, and 8 may be astrocytes. It is important to note that are less subjective tools to labeling astrocytes that would require additional tools and analyses.

Below, the identities of astrocytes are established based on marker gene expression. The annotations are then verified through another UMAP plot.

```
# Rename identities for astrocytes based on marker gene expression
seurat_astroAnnot <- RenameIdents(seurat_umap, `1` = "Astrocyte", `3` = "Astrocyte", `6` = "Astrocyte",

# Verify the annotation by visualizing the UMAP plot with labels
DimPlot(seurat_astroAnnot, reduction = "umap", label = TRUE)
```



Upon verifying the astrocyte annotations, these astrocytes are subseted for further analysis.

```
# Isolate the astrocytes from the original Seurate dataset
astrocytes <- subset(seurat_astroAnnot, idents = "Astrocyte")
```

Sub-Clustering Analysis

Analysis such as sub-clustering can further help to interpret the biological significance of these clusters.

The following manipulations aim to merge the metadata of the astrocytes subset with the AD classification within the covariates file. This mapping will allow this analysis to determine whether there are any Alzheimer's Disease (AD)- specific subclusters. The `AddMetaData` of the Seurat package facilitates this merge.

```
# Load the covariates data
covariates <- read.csv("GSE138852_covariates.csv")

# Set row names for the covariates
row.names(covariates) <- covariates$X

# Ensure the covariates are properly aligned with the Seurat object
astrocytes <- AddMetaData(object = astrocytes, metadata = covariates[, c("oupSample.subclustID", "oupSa

# Check if the metadata has been added successfully
head(astrocytes@meta.data)
```

```
##                              orig.ident nCount_RNA nFeature_RNA percent.mt
## AAACCTGGTAGCGATG_AD5_AD6 SeuratProject        720          527  2.9166667
## AAACCTGTCAGTCAGT_AD5_AD6 SeuratProject       1209          773  1.1579818
## AAACCTGTCCAGTATG_AD5_AD6 SeuratProject        562          434  0.5338078
## AAAGCAAGTCGAATCT_AD5_AD6 SeuratProject        586          431  0.3412969
## AAAGCAAGTTTGTTGG_AD5_AD6 SeuratProject        473          356  1.0570825
## AAAGTAGGTTCCACGG_AD5_AD6 SeuratProject        433          330  1.6166282
##                          RNA_snn_res.0.5 seurat_clusters oupSample.subclustID
## AAACCTGGTAGCGATG_AD5_AD6               1               1                   o3
## AAACCTGTCAGTCAGT_AD5_AD6               1               1                   o3
## AAACCTGTCCAGTATG_AD5_AD6               1               1                   o3
## AAAGCAAGTCGAATCT_AD5_AD6               1               1                   o3
## AAAGCAAGTTTGTTGG_AD5_AD6               1               1                   o3
## AAAGTAGGTTCCACGG_AD5_AD6               1               1                   o3
##                          oupSample.subclustCond
## AAACCTGGTAGCGATG_AD5_AD6                     AD
## AAACCTGTCAGTCAGT_AD5_AD6                     AD
## AAACCTGTCCAGTATG_AD5_AD6                     AD
## AAAGCAAGTCGAATCT_AD5_AD6                     AD
## AAAGCAAGTTTGTTGG_AD5_AD6                     AD
## AAAGTAGGTTCCACGG_AD5_AD6                     AD
```

The process of identifying sub-clusters within astrocytes then begins below.

```
# Identify sub-clusters within astrocytes
astrocytes <- FindNeighbors(astrocytes, dims = 1:10)
```

```
## Computing nearest neighbor graph
```

```
## Computing SNN
```

```
astrocytes <- FindClusters(astrocytes, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 5000
## Number of edges: 167318
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8570
## Number of communities: 8
## Elapsed time: 0 seconds
```

```
astrocytes <- RunUMAP(astrocytes, dims = 1:10)
```

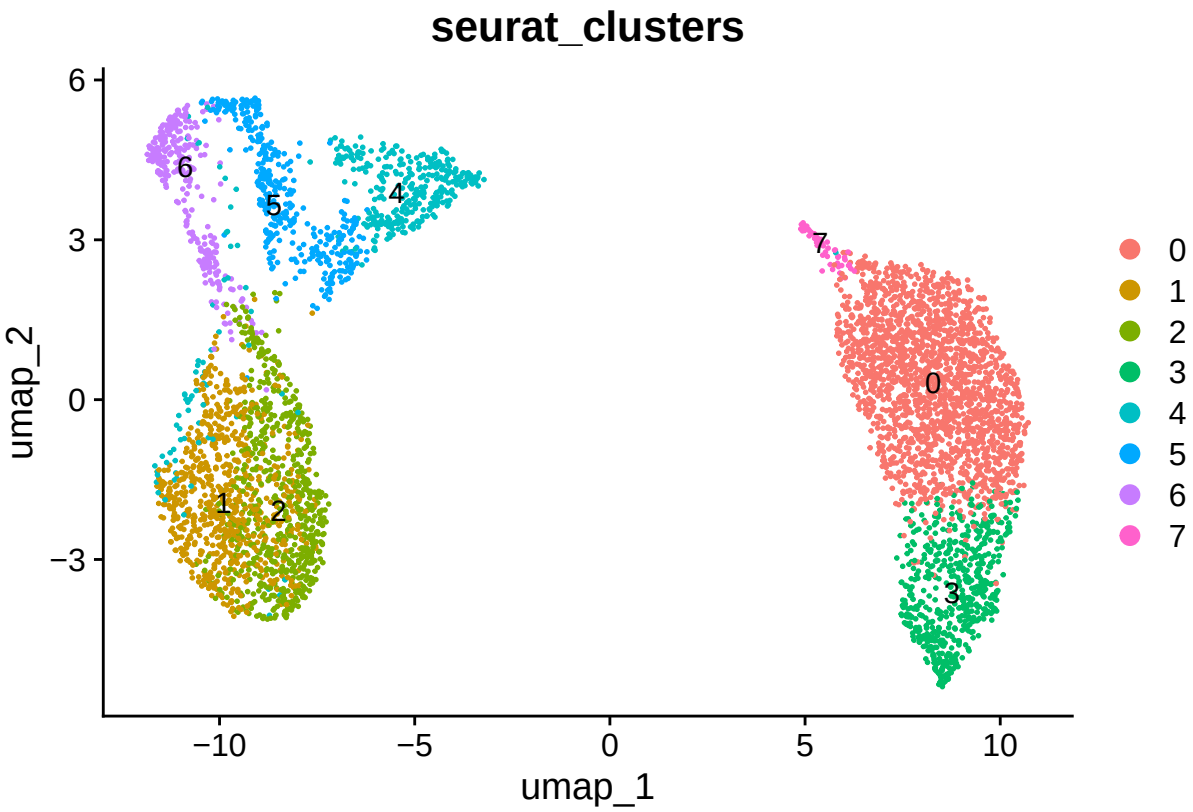
```
## 21:15:01 UMAP embedding parameters a = 0.9922 b = 1.112
```

```
## 21:15:01 Read 5000 rows and found 10 numeric columns
```

```
## 21:15:01 Using Annoy for neighbor search, n_neighbors = 30
```

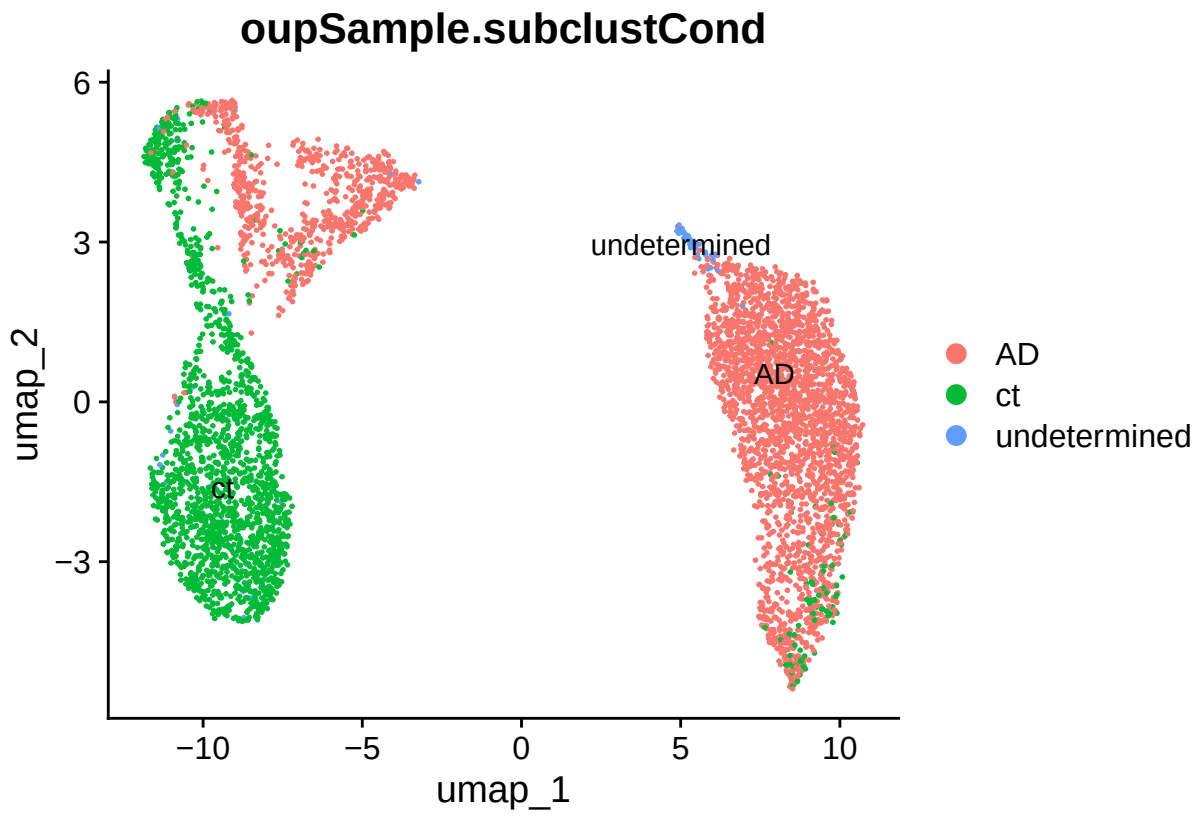
```
## 21:15:01 Building Annoy index with metric = cosine, n_trees = 50

## 0%   10   20   30   40   50   60   70   80   90   100%

## [----|----|----|----|----|----|----|----|----|----|

## **************************************************|
## 21:15:01 Writing NN index file to temp file /var/folders/zy/2dvzl4x52_v5fvrlsgrzn7880000gp/T//RtmpyM
## 21:15:01 Searching Annoy index using 1 thread, search_k = 3000
## 21:15:03 Annoy recall = 100%
## 21:15:04 Commencing smooth kNN distance calibration using 1 thread with target n_neighbors = 30
## 21:15:04 Initializing from normalized Laplacian + noise (using RSpectra)
## 21:15:04 Commencing optimization for 500 epochs, with 212644 positive edges
## 21:15:14 Optimization finished
```

```r
# Visualize the astrocyte sub-clusters
DimPlot(astrocytes, reduction = "umap", label = TRUE, group.by = "seurat_clusters")
```



There are seven identified subclusters in the isolated astrocytes. There appears to be stronger heterogeneity in clusters 0 and 3 followed by 4, 5, and 6. The overlap in clusters 1 and 2 may reflect weak evidence of heterogeneity. In addition, the small size of cluster 7 may be an outlier due to its very small representation of the astrocyte population.

```
# Visualize the astrocyte sub-clusters with disease status
DimPlot(astrocytes, reduction = "umap", group.by = "oupSample.subclustCond", label = TRUE)
```



Based on the visualization above, it appears that the previously labeled sub-clusters of 0,3,4 and 5 are marked with the AD disease status.

This is further supported by the table visualization of the proportion of cells in each sub-cluster for AD and controls.

```
# Check the proportion of cells in each sub-cluster for AD and Control
table(astrocytes$seurat_clusters, astrocytes$oupSample.subclustCond)
```

```
##
##       AD   ct undetermined
##    0 1893   15            4
##    1    1  717            4
##    2    3  655            0
##    3  516   70            0
##    4  332   56           18
##    5  347   22            2
##    6   11  274            1
##    7   16    0           43
```

Differential Analysis

Cluster zero is chosen among the astrocytes sub-clusters and a differential gene expression analysis between the AD and control group is performed.

```
# Using sub-cluster 0 for analysis
selected_cluster <- subset(astrocytes, idents = 0)

# Set the identities in selected_cluster based on the condition
Idents(selected_cluster) <- selected_cluster$oupSample.subclustCond

# Verify the identities
# table(Idents(selected_cluster))

# Run differential expression analysis
de_results <- FindMarkers(selected_cluster, ident.1 = "AD", ident.2 = "ct", test.use = "wilcox")
```

```
## For a (much!) faster implementation of the Wilcoxon Rank Sum Test,
## (default method for FindMarkers) please install the presto package
## --------------------------------------------
## install.packages('devtools')
## devtools::install_github('immunogenomics/presto')
## --------------------------------------------
## After installation of presto, Seurat will automatically use the more
## efficient implementation (no further action necessary).
## This message will be shown once per session
```

```
# View the top differentially expressed genes
head(de_results)
```

```
##                       p_val avg_log2FC pct.1 pct.2    p_val_adj
## XIST          3.025500e-38  -9.230385 0.001 0.133 3.282668e-34
## FAM105A       2.973201e-29 -11.943573 0.000 0.067 3.225923e-25
## RP11-478B9.1  6.219004e-29  -7.842563 0.001 0.133 6.747619e-25
## NUP43         6.427132e-17  -6.812117 0.003 0.133 6.973438e-13
## ANGPTL1       3.204518e-15  -7.041987 0.001 0.067 3.476902e-11
## KMO           3.204518e-15  -7.362594 0.001 0.067 3.476902e-11
```

The use of the Wilcoxon Rank-Sum test is to compare two independent groups, being the AD and control groups. This analytical method is also best for non-parametric data like snRNA seq data that doesn't meet the assumptions for parametric tests. Therefore the parameters of the differential expression analysis reflect the two comparison groups and the specification of the "wilcox" test.

The final result of this analysis should show the list of genes that show significant difference in expression between the AD and control groups. This can provide insight into genes that may be responsible for disease states.