Single-Cell RNA Sequencing Analysis of PBMC Data

2025-04-20

Introduction: Single-Cell RNA-Seq Analysis of PBMCs

This report presents an analysis of single-cell RNA sequencing (scRNA-seq) data from Peripheral Blood Mononuclear Cells (PBMCs). PBMCs are a diverse population of immune cells that circulate in the bloodstream, including T cells, B cells, NK cells, monocytes, dendritic cells, and platelets. Understanding the heterogeneity and gene expression patterns in these cells is crucial for immunology research and clinical applications.

Dataset Description Biological context:

Organism: Human (Homo sapiens) Tissue: Blood (specifically PBMCs) Technology: 10x Genomics Chromium Single-Cell Gene Expression Sample size: ~3,000 cells Sequencing platform: Illumina NextSeq 500

Analysis Workflow

1. Data loading and quality control
2. Normalization and feature selection
3. Dimensionality reduction (PCA, UMAP, t-SNE)
4. Clustering
5. Marker gene identification
6. Cell type annotation using SingleR

Data loading and quality control

Data Loading and Initial Processing

We begin by loading the necessary R libraries. dplyr is used for data manipulation, Seurat is the main package for single-cell RNA-seq analysis, and patchwork helps in combining multiple plots.

```
library(dplyr)
library(Seurat)
library(patchwork)
```

Next, We load the single-cell dataset of peripheral blood mononuclear cells (PBMCs) from 10X Genomics. The data includes raw gene expression counts. We create a Seurat object:

Genes must be expressed in at least 3 cells.

Cells must have at least 200 detected genes to be retained.

This ensures that both low-quality genes and low-quality cells are excluded from downstream analyses.

```r
# Load the PBMC dataset
data_dir <- "C:/Users/Admin/MLbio/"
pbmc.data <- Read10X(data.dir = data_dir)

# Initialize the Seurat object with the raw (non-normalized data).
pbmc <- CreateSeuratObject(counts = pbmc.data, project = "pbmc3k", min.cells = 3, min.features = 200)
pbmc
```

```
## An object of class Seurat
## 13714 features across 2700 samples within 1 assay
## Active assay: RNA (13714 features, 0 variable features)
##  1 layer present: counts
```

The initialized Seurat object contains expression data for 13,714 genes (features) across 2,700 single cells (samples), indicating a relatively rich dataset following initial filtering. The active assay is set to "RNA", meaning that downstream analyses will focus on gene expression profiles. At this stage, no highly variable genes have been selected yet (0 variable features), which typically occurs in later preprocessing steps. Only one data layer—"counts"—is present, representing raw, unnormalized gene expression values. This setup forms the foundational input for subsequent analyses.

During the creation of the Seurat object, initial filtering criteria are applied to improve data quality: **min.cells = 3**: Each gene must be detected in at least 3 cells to be included. This filters out genes that might be technical artifacts or extremely rare transcripts.

min.features = 200: Each cell must express at least 200 genes to be included. This removes likely empty droplets or severely damaged cells.

Additionally, we calculate the percentage of mitochondrial gene expression per cell using genes that begin with the prefix "MT-". A high proportion of mitochondrial transcripts often suggests stressed or dying cells and can be used later to flag or exclude low-quality cells from the dataset.

Let's examine what the raw count matrix looks like for a few genes across the first 30 cells:

```r
# Examine a few genes in the first thirty cells
pbmc.data[c("CD3D","TCL1A","MS4A1"), 1:30]
```

```
## 3 x 30 sparse Matrix of class "dgCMatrix"

##
## CD3D  4 . 10 . . 1 2 3 1 . . 2 7 1 . . 1 3 . 2  3 . . . . . 3 4 1 5
## TCL1A . .  . . . . . . 1 . . . . . . . . . . . . 1 . . . . . . . .
## MS4A1 . 6  . . . . . . 1 1 1 . . . . . . . . . . 36 1 2 . . 2 . . . .
```

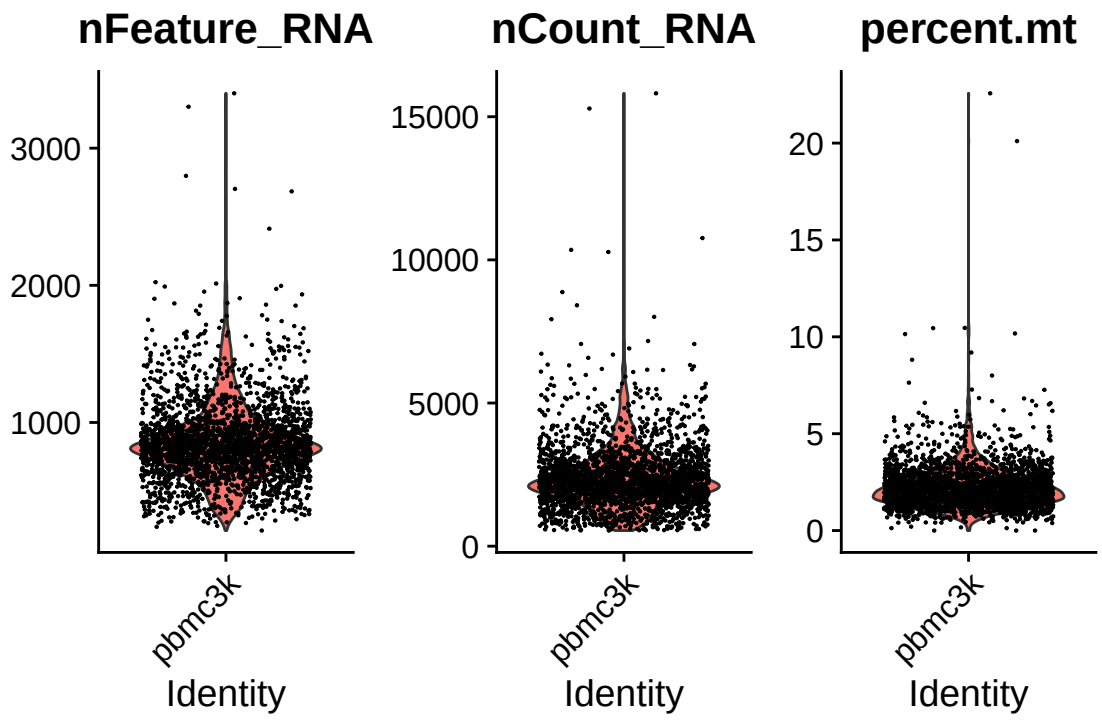
The extracted count matrix displays the raw expression levels (UMI counts) of three key marker genes—**CD3D**, **TCL1A**, and **MS4A1**—across the first 30 cells. These genes are commonly used to identify major immune cell types: CD3D marks T cells, TCL1A is associated with naïve B cells, and MS4A1 (CD20) marks mature B cells. The values indicate the number of unique RNA molecules detected for each gene in each cell. Cells with a dot (".") have zero expression for that gene, highlighting the sparsity typical of single-cell RNA-seq data. Notably, CD3D shows broad expression across multiple cells, suggesting the presence of T cells, while MS4A1 has high expression in specific cells (e.g., cell 21), consistent with B cell identity. This early inspection helps validate the data and provides an initial glimpse into the sample.

Quality Control

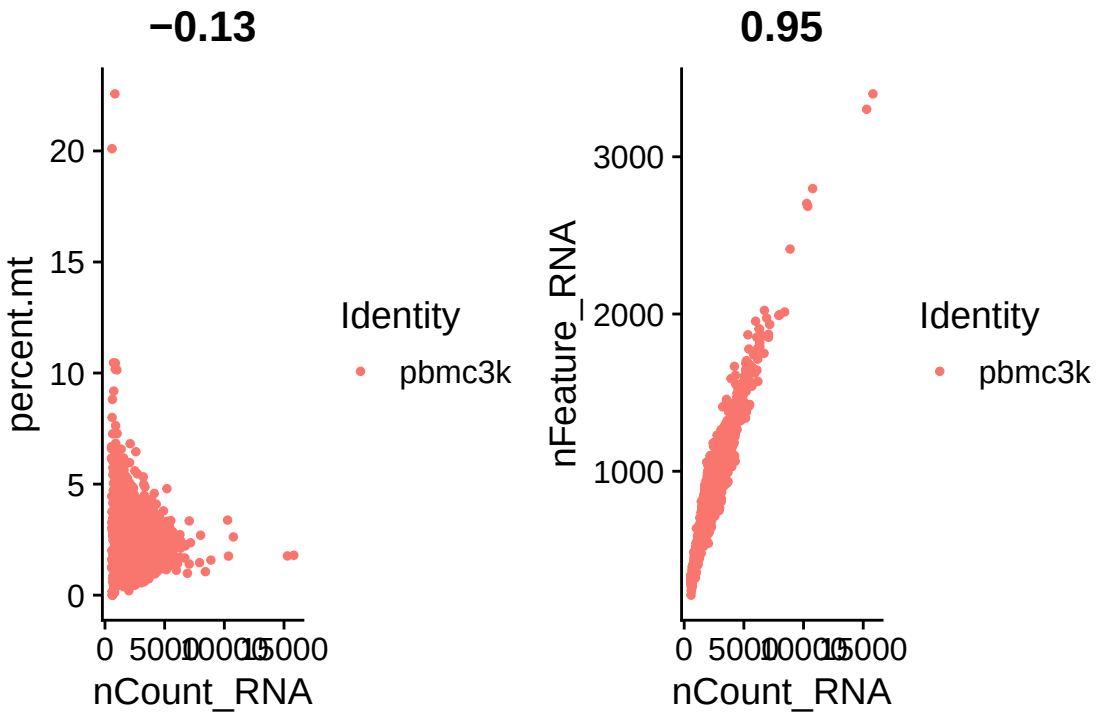
Before proceeding with analysis, we need to identify and filter out low-quality cells. Common quality control metrics include number of unique genes detected per cell (feature count), total number of molecules detected per cell (count depth), and percentage of reads mapping to mitochondrial genes

```
# Calculate percentage of mitochondrial genes
pbmc[["percent.mt"]] <- PercentageFeatureSet(pbmc, pattern = "^MT-")

# Visualize QC metrics as violin plots
VlnPlot(pbmc, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```




```
# Visualize feature-feature relationships
plot1 <- FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "percent.mt")
plot2 <- FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
plot1 + plot2
```



Looking at these QC plots helps us determine appropriate filtering thresholds:

1. **nFeature_RNA** - genes per cell: Cells with too few genes are likely empty droplets or poor-quality cells. Cells with too many genes could be doublets. Based on that, we'll filter cells to keep those with between 200 and 2,500 genes.
2. **percent.mt** - mitochondrial percentage: High mitochondrial content indicates dying or stressed cells where cytoplasmic RNA has leaked out but mitochondrial RNA remains. We'll use a 5% threshold.
3. The **scatter plot** shows a strong correlation between gene count and molecule count, which is expected. This suggests the data has good technical quality. A high Pearson correlation coefficient (R = 0.95) between the number of detected genes (nFeature_RNA) and total RNA molecules (nCount_RNA) indicates that as the sequencing depth increases, more unique genes are consistently captured rather than just increasing redundancy of already abundant transcripts. This linear relationship implies that the dataset is not heavily biased by technical noise or over-amplification, and supports the assumption that differences in gene detection across cells are biologically meaningful rather than technical artifacts.

Now we apply the filtering:

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

Normalization and Feature Selection

After quality control, we normalize the data to account for differences in sequencing depth between cells.

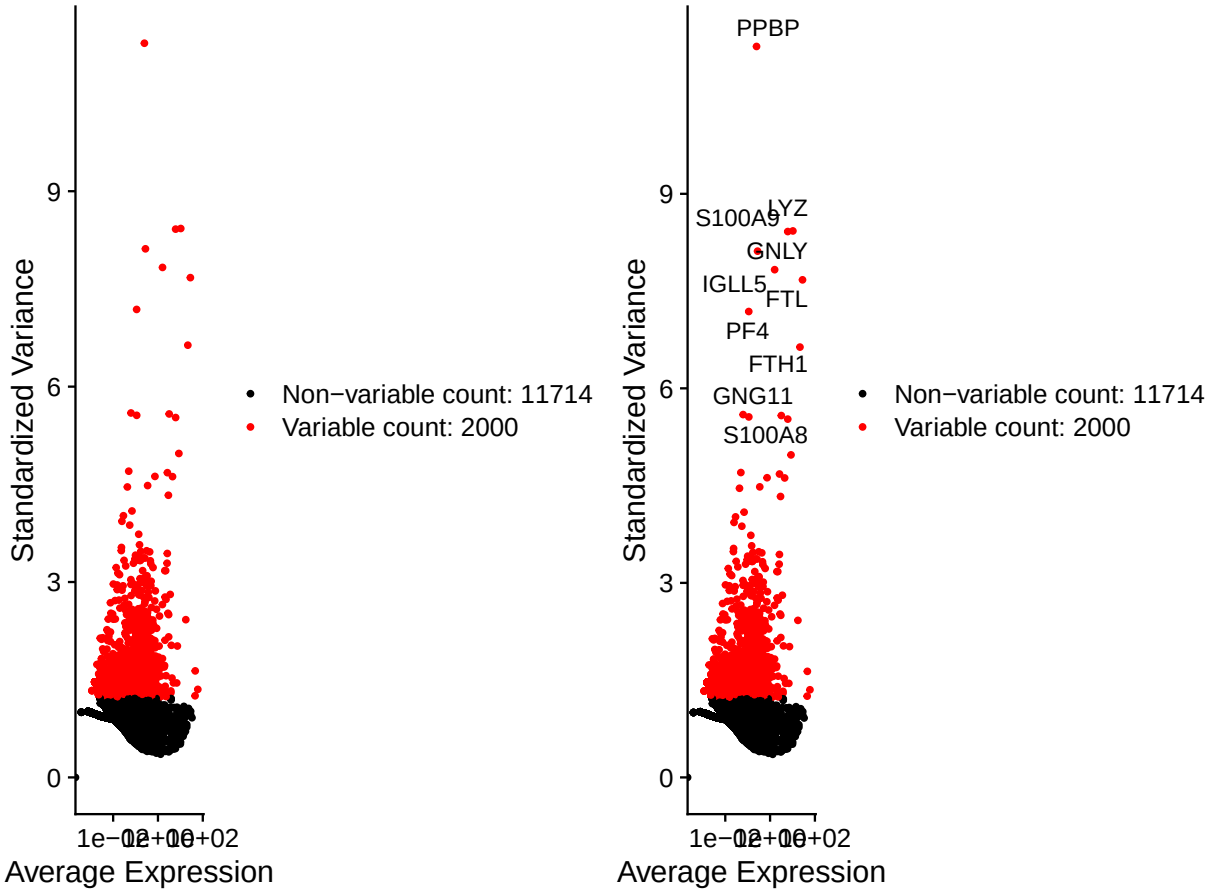
```
# Normalize data
pbmc <- NormalizeData(pbmc, normalization.method = "LogNormalize", scale.factor = 10000)
```

This approach works by scaling the expression of each gene in each cell by the total expression in that cell, multiplying by a scale factor (here, 10,000), and then applying a natural logarithm transformation. This helps stabilize variance across genes and cells and prepares the data for downstream analyses such as dimensionality reduction and clustering.

Next, we identify the most variable genes, which will be used for downstream analysis:

```
# Find variable features
pbmc <- FindVariableFeatures(pbmc, selection.method = 'vst', nfeatures = 2000)

# Plot variable features
top10 <- head(VariableFeatures(pbmc), 10)
plot1 <- VariableFeaturePlot(pbmc)
plot2 <- LabelPoints(plot = plot1, points = top10, repel = TRUE)
plot1 + plot2
```



The variable feature plots above illustrate the relationship between average gene expression (x-axis) and standardized variance (y-axis) across all cells. In the left panel, the 2,000 most variable genes are highlighted in red against the background of all genes (black). The right panel identifies some of the top variable genes, including markers like PPBP (platelet factor), S100A8 (found in neutrophils), GNLY (natural killer cells), and IGLL5 (B cells). These highly variable genes often correspond to cell-type-specific markers that help

distinguish between different cell populations in the PBMC sample. Genes with both high expression and high variance are particularly valuable for downstream analysis as they provide robust signals for cell type identification.

Before dimensional reduction, we scale the data:

```
# Scale data
all.genes <- rownames(pbmc)
pbmc <- ScaleData(pbmc, features = all.genes)
```

Scaling involves centering the expression of each gene so that it has a mean of zero and scaling it to have unit variance. Without this step, highly expressed genes could disproportionately influence the results. Scaling is a standard and essential preprocessing procedure that improves the performance and interpretability of techniques such as PCA and clustering.

Dimensionality Reduction

Principal Component Analysis (PCA)

First, we performed linear dimensionality reduction using PCA:

```
pbmc <- RunPCA(pbmc, features = VariableFeatures(object = pbmc))
```

PCA transforms our high-dimensional expression data into a lower-dimensional space where each dimension (principal component) captures a direction of maximum variation in the data. Looking at the top genes contributing to each PC helps us understand what biological signals are being captured:

PC1 includes T cell markers (CD3D, IL32) and B cell markers (CD79A), suggesting it distinguishes major lymphocyte populations **PC2** includes monocyte markers (S100A9, S100A8) and lymphocyte markers, likely separating myeloid from lymphoid cells

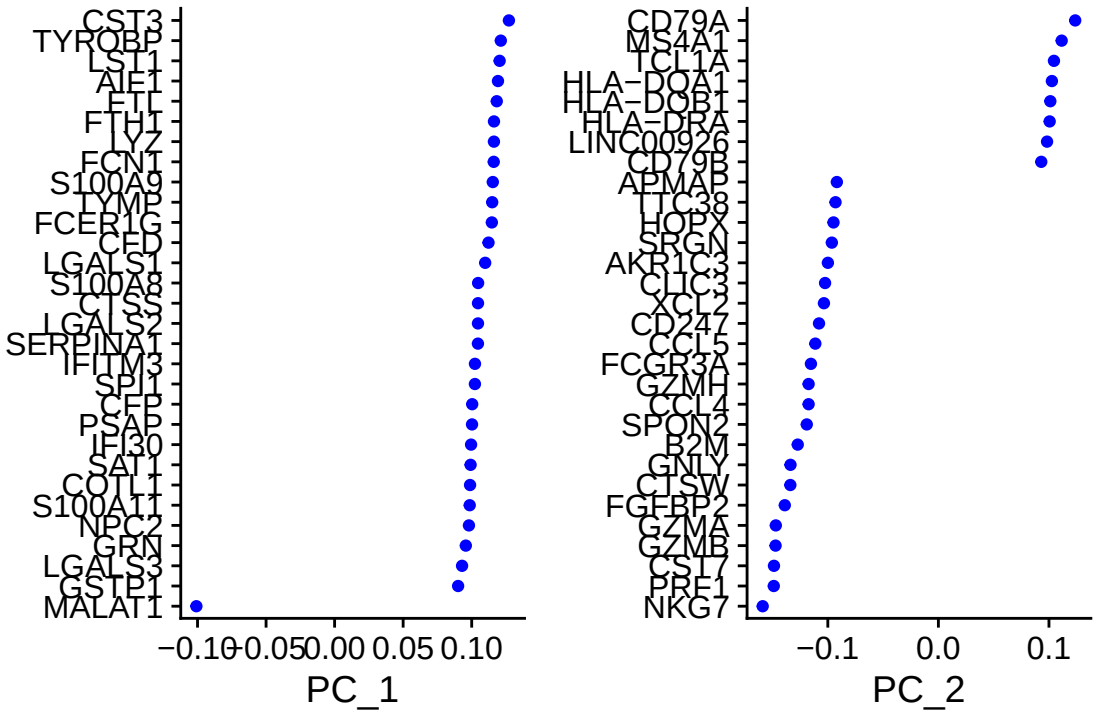
Seurat provides several powerful tools for visualizing PCA results:

PCA Loadings: To identify which genes contribute most to each principal component.

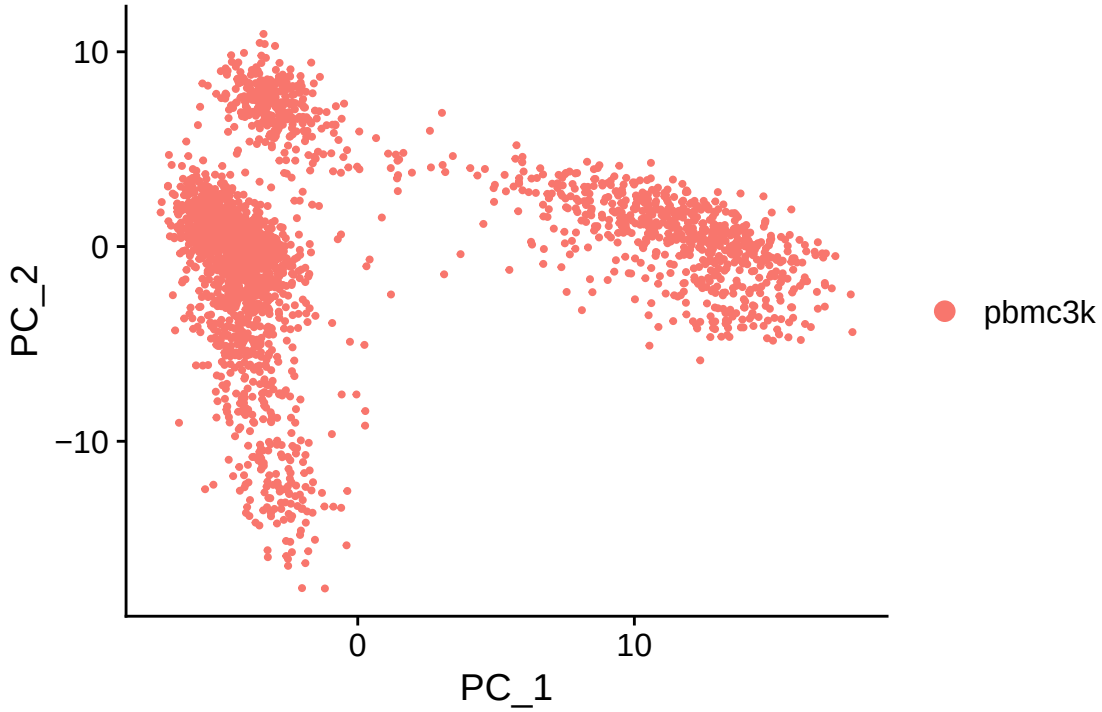
```
# Inspect PCA results
print(pbmc[['pca']], dims = 1:5, nfeatures = 5)
```

```
## PC_ 1
## Positive:  CST3, TYROBP, LST1, AIF1, FTL
## Negative:  MALAT1, LTB, IL32, IL7R, CD2
## PC_ 2
## Positive:  CD79A, MS4A1, TCL1A, HLA-DQA1, HLA-DQB1
## Negative:  NKG7, PRF1, CST7, GZMB, GZMA
## PC_ 3
## Positive:  HLA-DQA1, CD79A, CD79B, HLA-DQB1, HLA-DPB1
## Negative:  PPBP, PF4, SDPR, SPARC, GNG11
## PC_ 4
## Positive:  HLA-DQA1, CD79B, CD79A, MS4A1, HLA-DQB1
## Negative:  VIM, IL7R, S100A6, IL32, S100A8
## PC_ 5
## Positive:  GZMB, NKG7, S100A8, FGFBP2, GNLY
## Negative:  LTB, IL7R, CKB, VIM, MS4A7
```

```
# Visualize the gene loadings for PC1 and PC2
VizDimLoadings(pbmc, dims = 1:2, reduction = 'pca')
```

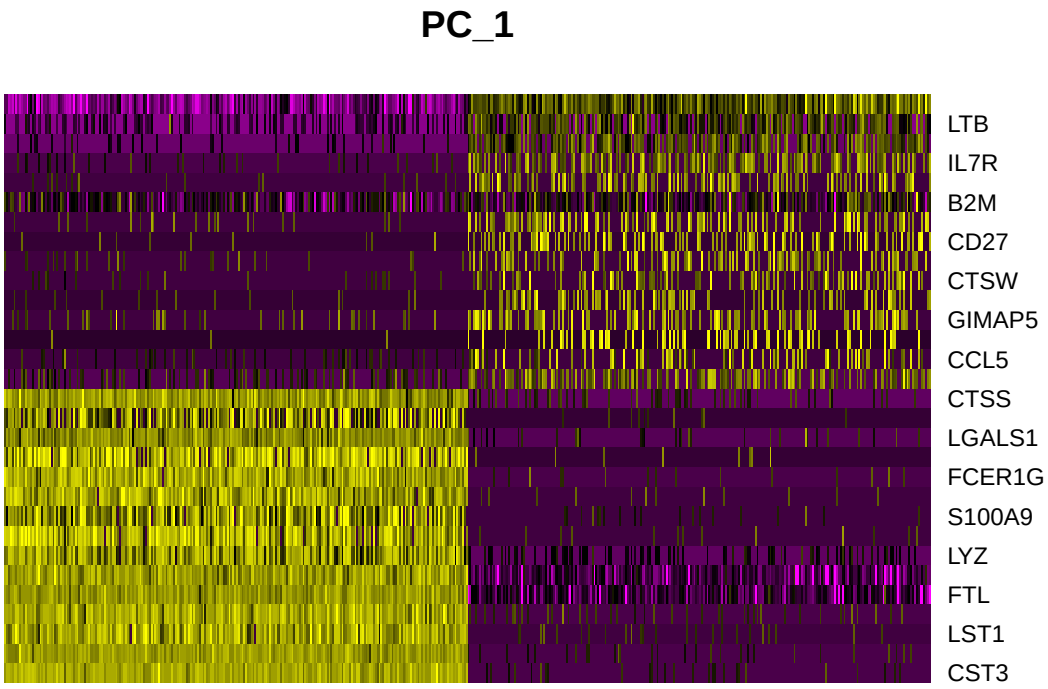



```
# Plot cells projected onto the first two PCs
DimPlot(pbmc, reduction = 'pca')
```

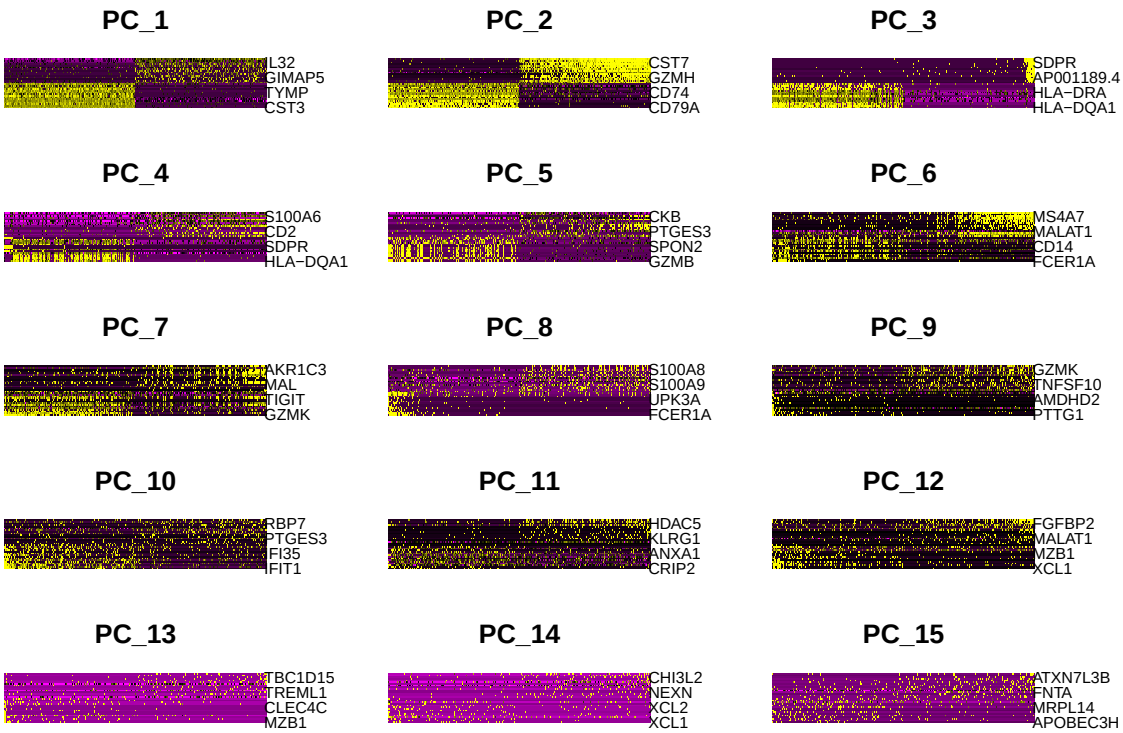


Heatmaps: To visualize gene expression patterns across PCs.

```
# Heatmap for PC1
DimHeatmap(pbmc, dims = 1, cells = 500, balanced = TRUE)
```



```
# Heatmaps for PCs 1 through 15
DimHeatmap(pbmc, dims = 1:15, cells = 500, balanced = TRUE)
```

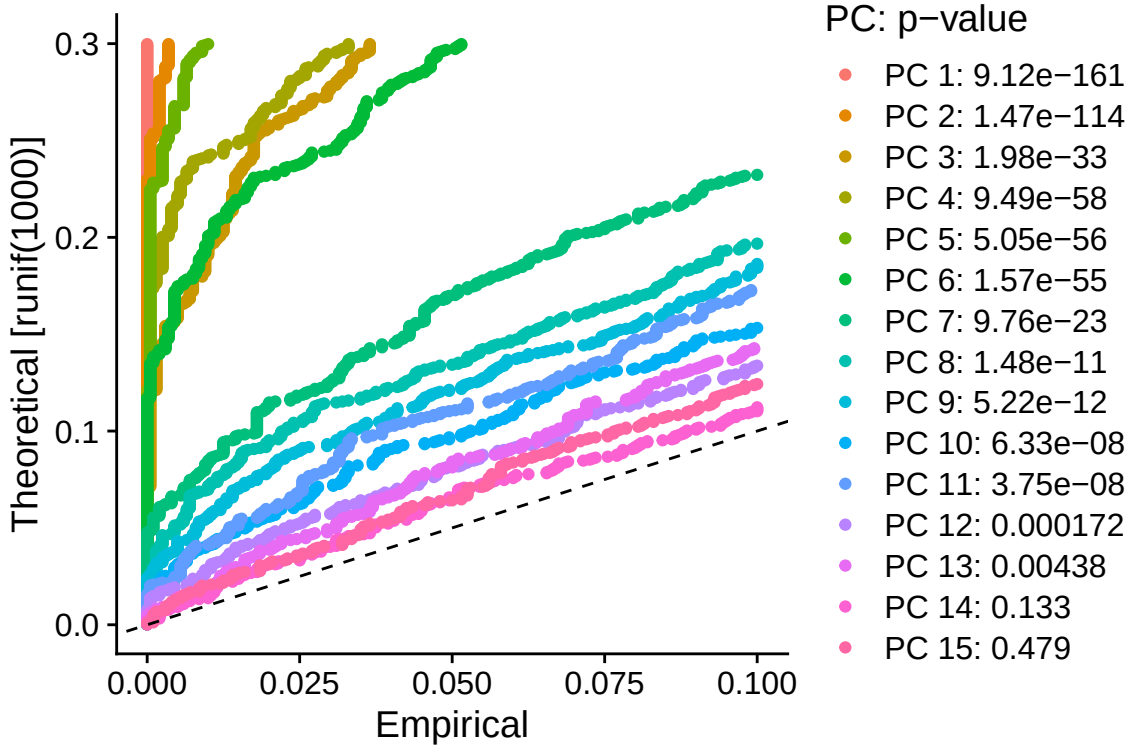


The heatmap shows the expression patterns of genes that define each PC across cells. Clear patterns in early PCs (1-10) indicate strong biological signal, while later PCs show more subtle patterns.

Determining the 'Dimensionality' of the Dataset

To determine how many PCs to include in downstream analysis, we can use both statistical and heuristic approaches: **JackStraw Plot:** A statistical approach to identify significant PCs

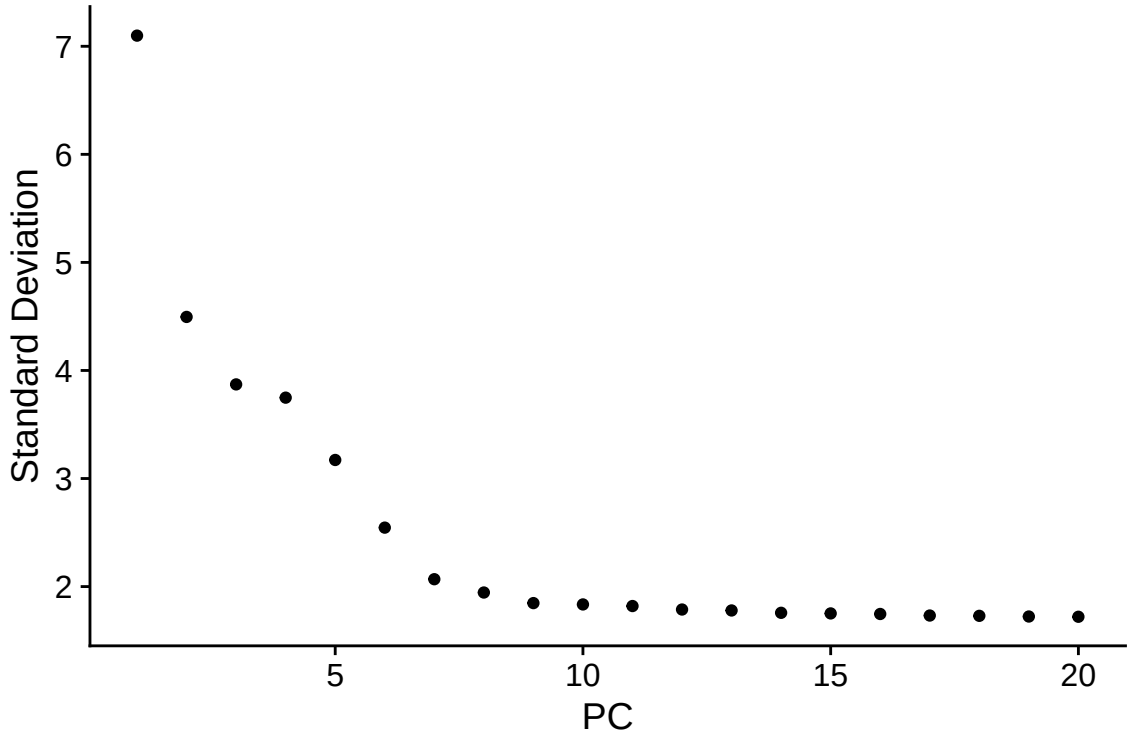
```
# Perform JackStraw analysis
pbmc <- JackStraw(pbmc, num.replicate = 100)
pbmc <- ScoreJackStraw(pbmc, dims = 1:20)
JackStrawPlot(pbmc, dims = 1:15)
```



The JackStraw analysis identifies principal components (PCs) that capture significant biological variation in the dataset. PCs with a strong enrichment of low p-values (solid curve above the dashed line) indicate structured, non-random gene expression patterns. These significant PCs likely represent key sources of heterogeneity, such as differences between major cell types, activation states, or other biologically relevant processes. Selecting these PCs for downstream analysis ensures that the dimensionality reduction retains meaningful biological signals rather than technical noise.

Elbow Plot: To determine how many PCs to retain for downstream analysis, we also examined the elbow plot, which shows the standard deviation of each principal component. This helps to identify an "elbow" point beyond which additional PCs contribute little added variance.

```
ElbowPlot(pbmc)
```



The Elbow Plot shows diminishing returns in explained variance after approximately PC 10-12. Based on this and the heatmap, we'll use the first 10 PCs for downstream analysis, as they capture most of the meaningful variation in the data.

Cell Clustering and Non-linear Dimensionality Reduction

Now we perform clustering to identify groups of cells with similar expression profiles:

```
# Find neighbors and clusters
pbmc <- FindNeighbors(pbmc, dims = 1:10)
pbmc <- FindClusters(pbmc, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2638
## Number of edges: 95927
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8728
## Number of communities: 9
## Elapsed time: 0 seconds
```

The clustering approach includes:

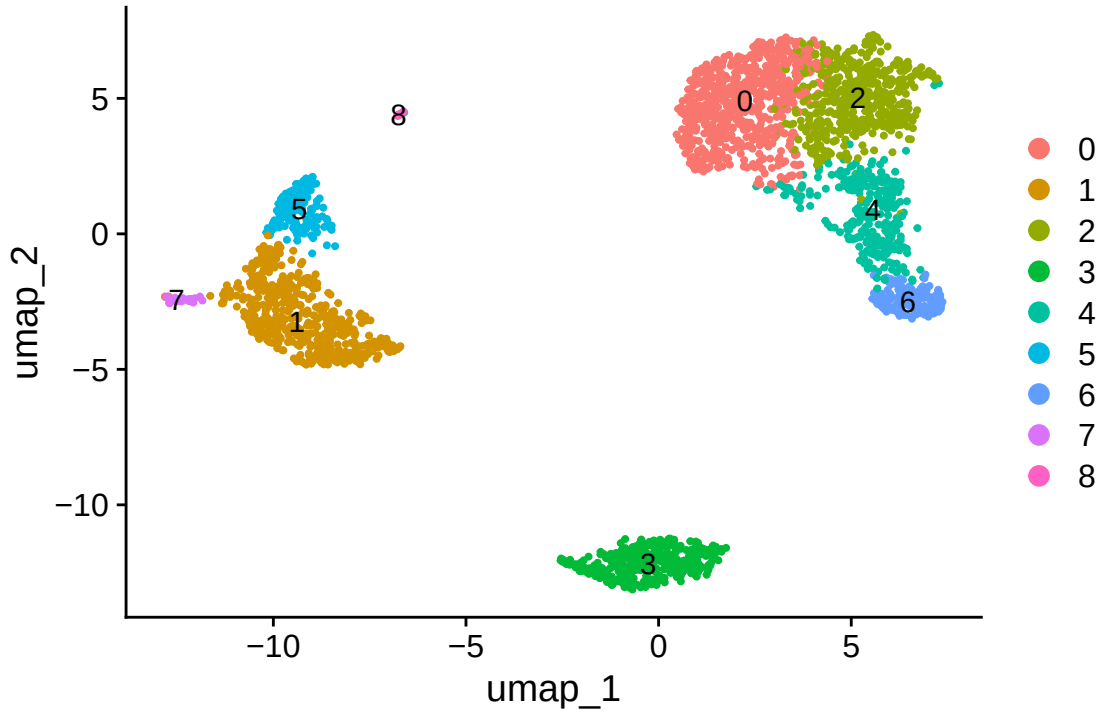
1. Constructs a K-nearest neighbor (KNN) graph in PCA space
2. Refines edge weights using Jaccard similarity
3. Applies the Louvain algorithm to identify communities (clusters) The resolution parameter (0.5) controls clustering granularity - higher values lead to more clusters

```
# Look at cluster IDs of the first 5 cells
head(Idents(pbmc), 5)
```

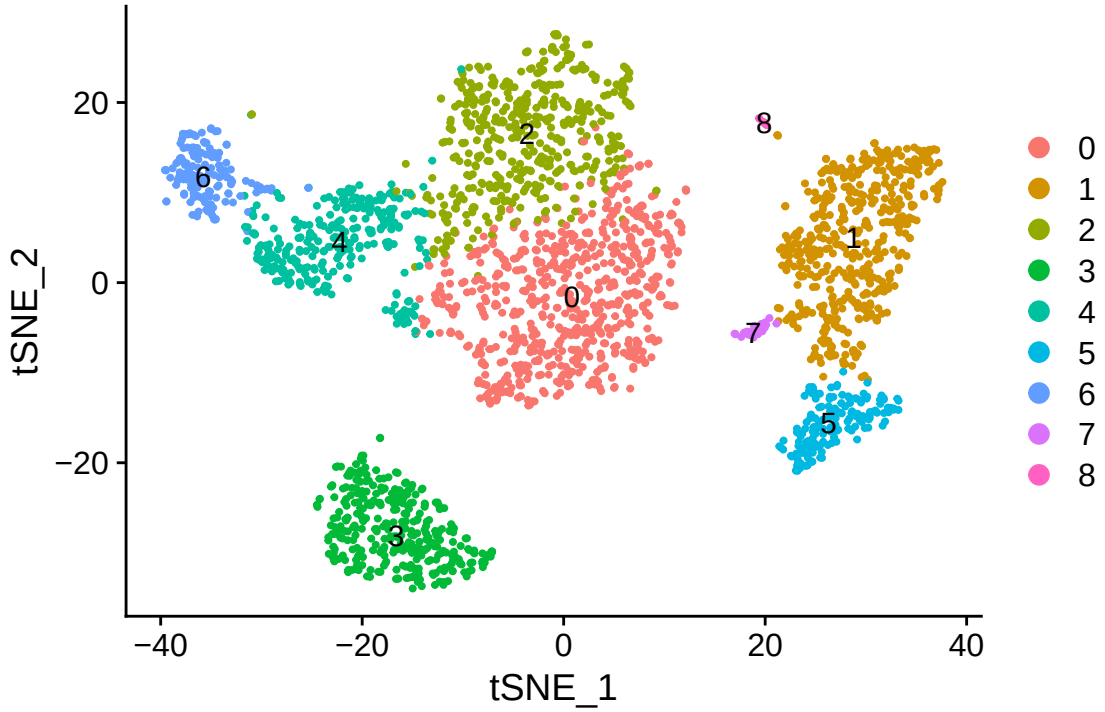
```
## AAACATACAACCAC-1 AAACATTGAGCTAC-1 AAACATTGATCAGC-1 AAACCGTGCTTCCG-1
##                2                3                2                1
## AAACCGTGTATGCG-1
##                6
## Levels: 0 1 2 3 4 5 6 7 8
```

To visualize the clusters in 2D space, we run both UMAP and t-SNE:

```
# Run UMAP
pbmc <- RunUMAP(pbmc, dims = 1:10)
umap_plot <- DimPlot(pbmc, reduction = 'umap', label = TRUE)
umap_plot
```




```
# Run t-SNE
pbmc <- RunTSNE(pbmc, dims = 1:10)
tsne_plot <- DimPlot(pbmc, reduction = 'tsne', label = TRUE)
tsne_plot
```



Both UMAP and t-SNE are non-linear dimensionality reduction techniques that preserve local neighborhood relationships better than PCA. They help visualize the high-dimensional clustering results in 2D space. Each point represents a single cell, and colors represent different clusters.

The UMAP and t-SNE plots show 9 distinct clusters, suggesting at least 9 different cell populations in our PBMC sample. UMAP tends to better preserve both local and global structure of the data compared to t-SNE, which is why it's becoming the preferred visualization method.

Finding Differentially Expressed Features (Cluster Biomarkers)

After clustering, the next crucial step is to identify markers that define each cluster. These marker genes help us assign biological identity to the clusters.

```
# Find markers for cluster 2
cluster2.markers <- FindMarkers(pbmc, ident.1 = 2, min.pct = 0.25)
head(cluster2.markers, n = 5)
```

```
##              p_val avg_log2FC pct.1 pct.2    p_val_adj
## IL32 2.892340e-90  1.3070772 0.947 0.465 3.966555e-86
## LTB  1.060121e-86  1.3312674 0.981 0.643 1.453850e-82
## CD3D 8.794641e-71  1.0597620 0.922 0.432 1.206097e-66
## IL7R 3.516098e-68  1.4377848 0.750 0.326 4.821977e-64
## LDHB 1.642480e-67  0.9911924 0.954 0.614 2.252497e-63
```

min.pct: Requires features to be detected in a minimum percentage of cells in either of the two populations

logfc.threshold: Requires features to have a minimum difference in expression between the two groups

Comparing Specific Clusters

We can also find markers that distinguish one cluster from specific other clusters:

```
# Find markers distinguishing cluster 5 from clusters 0 and 3
cluster5.markers <- FindMarkers(pbmc, ident.1 = 5, ident.2 = c(0, 3), min.pct = 0.25)
head(cluster5.markers, n = 5)
```

```
##                       p_val avg_log2FC pct.1 pct.2     p_val_adj
## FCGR3A        8.246578e-205   6.794969 0.975 0.040 1.130936e-200
## IFITM3        1.677613e-195   6.192558 0.975 0.049 2.300678e-191
## CFD           2.401156e-193   6.015172 0.938 0.038 3.292945e-189
## CD68          2.900384e-191   5.530330 0.926 0.035 3.977587e-187
## RP11-290F20.3 2.513244e-186   6.297999 0.840 0.017 3.446663e-182
```

Finding Markers for All Clusters

To identify markers for all clusters at once:

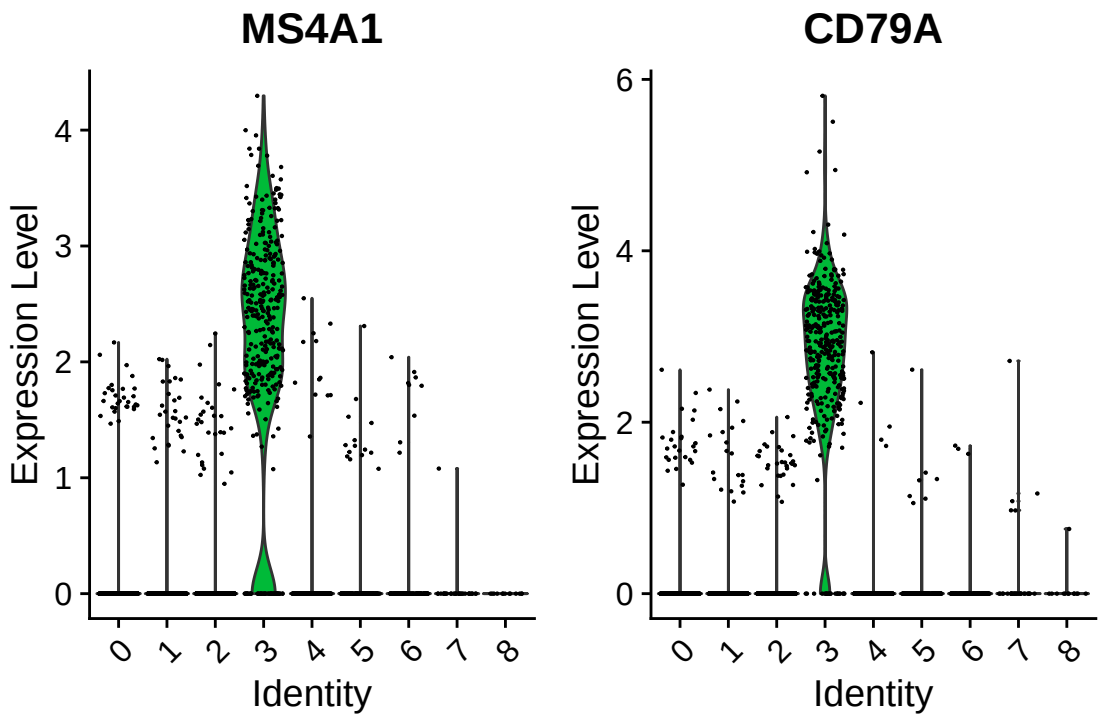
```
# Find all markers for all clusters
pbmc.markers <- FindAllMarkers(pbmc, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25)
# Display top 2 markers per cluster
pbmc.markers %>% group_by(cluster) %>% slice_max(n = 2, order_by = avg_log2FC)
```

```
## # A tibble: 18 x 7
## # Groups:   cluster [9]
##        p_val avg_log2FC pct.1 pct.2 p_val_adj cluster gene
##        <dbl>      <dbl> <dbl> <dbl>     <dbl> <fct>   <chr>
##  1 9.57e- 88       2.40 0.447 0.108 1.31e- 83 0       CCR7
##  2 1.35e- 51       2.14 0.342 0.103 1.86e- 47 0       LEF1
##  3 7.07e-139       7.28 0.299 0.004 9.70e-135 1       FOLR3
##  4 3.38e-121       6.74 0.277 0.006 4.64e-117 1       S100A12
##  5 2.97e- 58       2.09 0.42  0.111 4.07e- 54 2       AQP3
##  6 5.03e- 34       1.87 0.263 0.07  6.90e- 30 2       CD40LG
##  7 2.40e-272       7.38 0.564 0.009 3.29e-268 3       LINC00926
##  8 2.75e-237       7.14 0.488 0.007 3.76e-233 3       VPREB3
##  9 7.25e-165       4.41 0.577 0.055 9.95e-161 4       GZMK
## 10 3.27e- 88       3.74 0.419 0.061 4.48e- 84 4       GZMH
## 11 8.23e-168       5.88 0.37  0.005 1.13e-163 5       CKB
## 12 1.69e-212       5.43 0.506 0.01  2.32e-208 5       CDKN1C
## 13 8.10e-179       6.22 0.471 0.013 1.11e-174 6       AKR1C3
## 14 5.38e-112       6.07 0.29  0.007 7.38e-108 6       SH2D1B
## 15 1.46e-207       8.03 0.5   0.002 2.00e-203 7       SERPINF1
## 16 1.48e-220       7.63 0.812 0.011 2.03e-216 7       FCER1A
## 17 0              14.4  0.615 0     0         8       LY6G6F
## 18 7.32e-222      13.9  0.385 0     1.00e-217 8       RP11-879F14.2
```

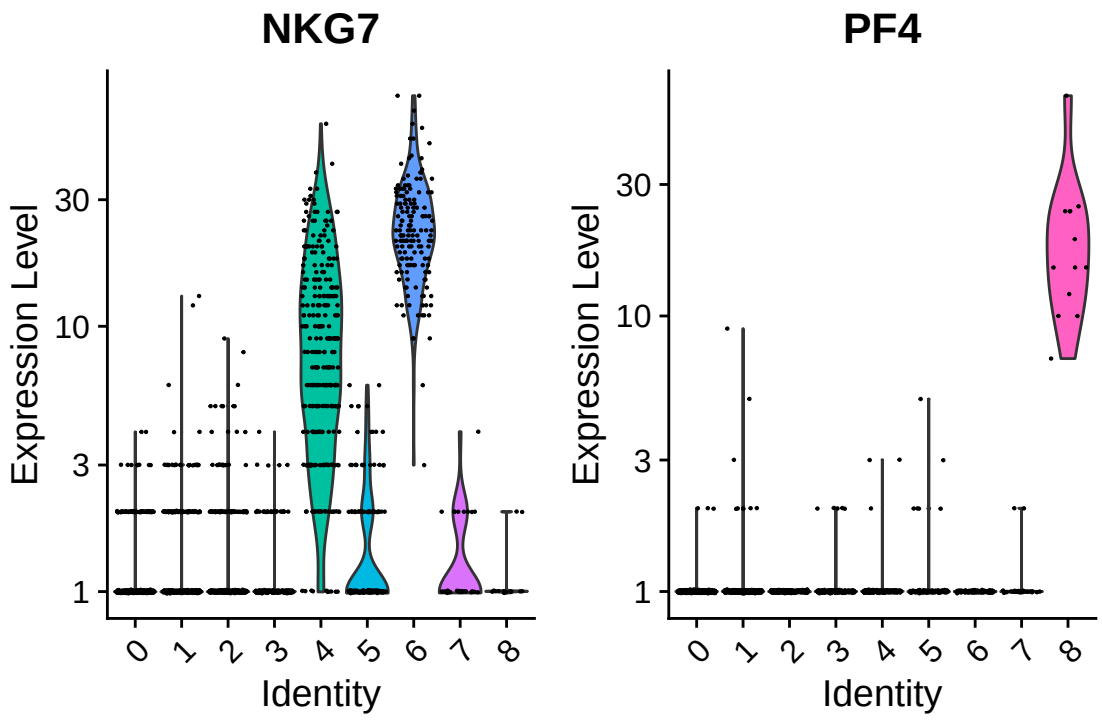
There are several ways to visualize the expression of marker genes across clusters:

Violin Plots: Show expression distributions across clusters

```
# Violin plots for B cell markers
VlnPlot(pbmc, features = c("MS4A1", "CD79A"))
```

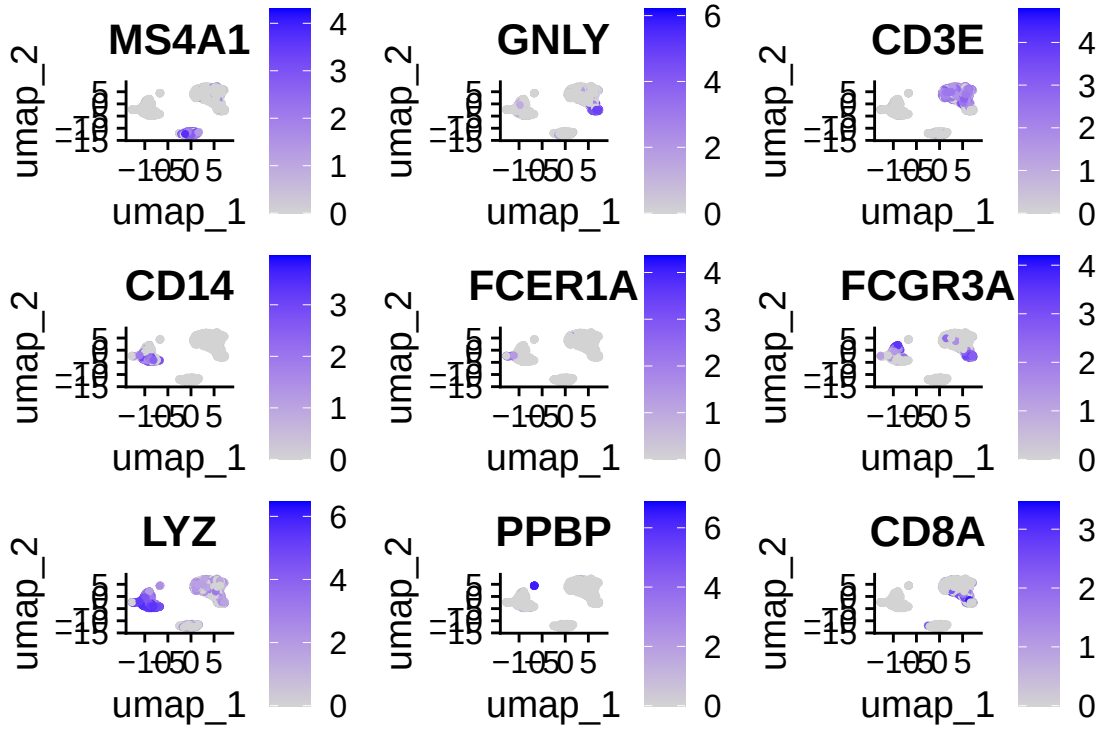



```
# Violin plots with log counts
VlnPlot(pbmc, features = c("NKG7", "PF4"), slot = 'counts', log = TRUE)
```



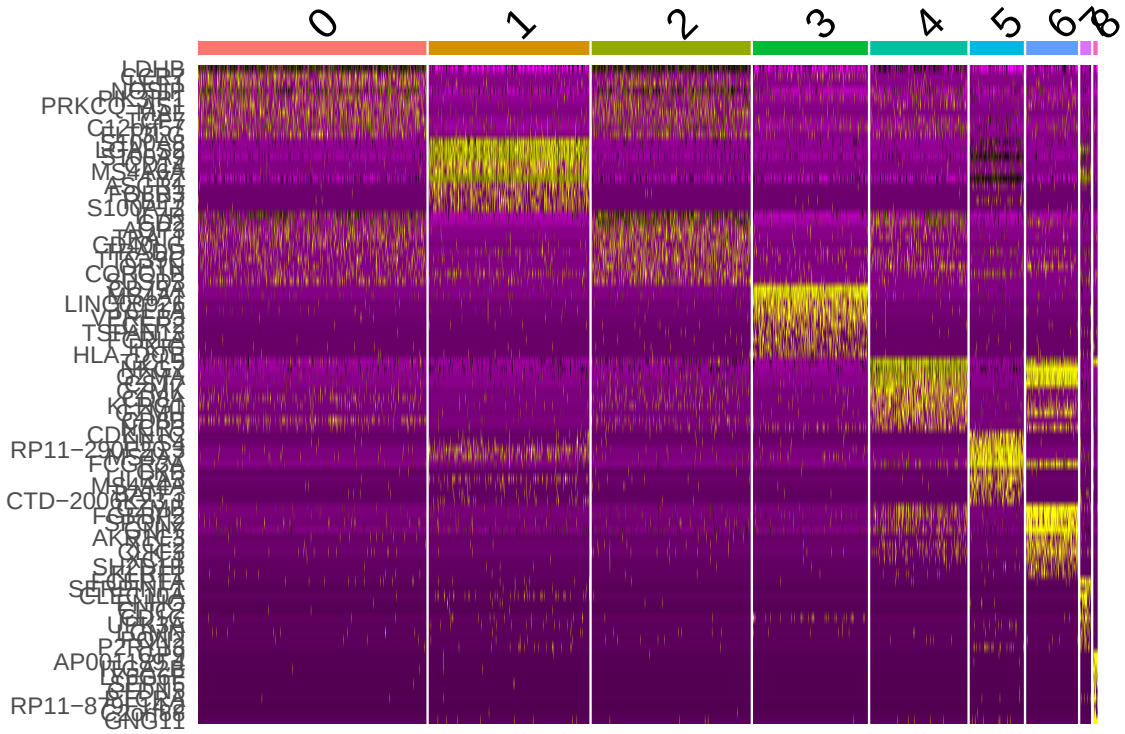
Feature Plots: Visualize expression on UMAP or t-SNE embeddings

```
# Feature plots for key cell type markers
FeaturePlot(pbmc, features = c("MS4A1", "GNLY", "CD3E", "CD14", "FCER1A", "FCGR3A", "LYZ", "PPBP", "CD8
```



Heatmaps: Visualize top markers across all clusters

```
# Heatmap of top 10 markers per cluster
pbmc.markers %>% group_by(cluster) %>% top_n(n = 10, wt = avg_log2FC) -> top10
DoHeatmap(pbmc, features = top10$gene) + NoLegend()
```



Based on these marker genes, we can assign cell type identities to each cluster:

Clusters expressing **MS4A1, CD79A: B cells** Clusters expressing **CD3E, CD8A: T cells** Clusters expressing **CD14, LYZ: Monocytes** Clusters expressing **FCER1A: Dendritic cells** Clusters expressing **GNLY, NKG7: NK cells** Clusters expressing **PPBP: Megakaryocytes**

The marker analysis is a critical step in single-cell RNA-seq analysis as it allows us to assign biological meaning to the computational clusters, translating them into recognizable cell types or states that can be interpreted in a biological context.