scRNA-seq Analysis Using Seurat

by Ning Shen, shenningzju@zju.edu.cn

2022-03-25

1. Introduction

1.1 about scRNAseq

In this practical, we will learn how to analyze single cell RNA-seq data. As you learnt from the lecture, single cell RNA-seq can be used to map the expression of individual genes in each cells. There are many important concepts in scRNA-seq data, such as UMI, index etc. Do you still remember what they mean? Also, there are different ways to do single cell RNA-seq, such as SMART-seq, DROP-seq, 10X genomics etc. Do you know the similarities and differences among them?

1.2 What is CD86?

In this practical, we will analyze the single cell RNA-seq from human lung samples. As you all know, macrophage is a kind of common immune cells in our body. When macrophage fight with bacteria or virus, they will switch to M1 state, a proinflammatory state. CD86 is the marker gene of M1 . Thus, mapping the expression pattern of CD86 in Peripheral Blood Mononuclear Cells(pbmc) is crucial in understanding disease progress.

1.3 scRNAseq data from Peripheral Blood Mononuclear Cells

To answer this question, we collect various published 10X genomics scRNA-seq data from human pbmc(*https://cf.10xgenomics.com/samples/cell/pbmc3k/pbmc3k_filtered_gene_bc_matrices.tar.gz*). In this practicals, we will try to answer the following questions: **1. What sub celltypes are there in lung tissues? 2. What's the expression pattern of CD86 in different subtypes of pbmc? 3. CD86 expressing cell subtypes are what type of cells in lung?**

2. Installation of Seurat

In this practical, we will use a R toolkit - Seurat to analyze scRNAseq data. For more info, please check: https://satijalab.org/seurat/

2.1 Installation instructions for Seurat

```
# Enter commands in R (or R studio, if installed)
install.packages('BiocManager')
BiocManager::install('multtest')
install.packages('Seurat')
```

if you have trouble installing Seurat package, here is what you can do:

1. check your R version (3.4 or higher) for Seurat v 3.1.4.
2. re-start your R and Rstudio
3. google/bing the error you get

3. Analyze the data

Before we start this practical, please download the scRNA-seq matrix data from blackboard.

3.1 Load library

```
library(Seurat)

## Warning: package 'Seurat' was built under R version 4.0.5

## Attaching SeuratObject

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.0.5

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(patchwork)

## Warning: package 'patchwork' was built under R version 4.0.5

library(Matrix)

## Warning: package 'Matrix' was built under R version 4.0.5
```

3.3 Load the data

To begin with, we will use pbmc_data, a 10X genomics scRNAseq data. We start by reading in the data. Since the raw data is in h5 format, we will use The *Read10X* function reads in the output of the cellranger pipeline from 10X, returning a unique molecular identified (UMI) count matrix. The values in this matrix represent the number of molecules for each feature (i.e. gene; row) that are detected in each cell (column). We next use the count matrix to create a Seurat object. The object serves as a container that contains both data (like the count matrix) and analysis (like PCA, or clustering results) for a single-cell dataset. For example, the count matrix is stored in pbmc[["RNA"]]@counts.

For a technical discussion of the Seurat object structure, check out our GitHub Wiki (https://github.com/satijalab/seurat/wiki).

```
#change this to your working directory

pbmc.data <- Read10X(data.dir =
"pbmc3k_filtered_gene_bc_matrices/filtered_gene_bc_matrices/hg19/")
```

3.3 data quality control

Seurat allows you to easily explore QC metrics and filter cells based on any user-defined criteria. A few QC metrics commonly used by the community include:

- The number of unique genes detected in each cell.
  - Low-quality cells or empty droplets will often have very few genes
  - Cell doublets or multiplets may exhibit an aberrantly high gene count
- Similarly, the total number of molecules detected within a cell (correlates strongly with unique genes)
- The percentage of reads that map to the mitochondrial genome
  - Low-quality / dying cells often exhibit extensive mitochondrial contamination
  - We calculate mitochondrial QC metrics with the PercentageFeatureSet function, which calculates the percentage of counts originating from a set of features
  - We use the set of all genes starting with MT- as a set of mitochondrial genes

Let’s first check the baisc properities of this dataset.

```
class(pbmc.data)

## [1] "dgCMatrix"
## attr(,"package")
## [1] "Matrix"

dim(pbmc.data)

## [1] 32738  2700

#let's check the readcount of first 6 genes in the first 6 cells
pbmc.data[1:6,1:6]

## 6 x 6 sparse Matrix of class "dgCMatrix"
##               AAACATACAACCAC-1 AAACATTGAGCTAC-1 AAACATTGATCAGC-1
## MIR1302-10                   .                .                .
## FAM138A                      .                .                .
## OR4F5                        .                .                .
## RP11-34P13.7                 .                .                .
## RP11-34P13.8                 .                .                .
## AL627309.1                   .                .                .
##                AAACCGTGCTTCCG-1 AAACCGTGTATGCG-1 AAACGCACTGGTAC-1
## MIR1302-10                    .                .                .
## FAM138A                       .                .                .
## OR4F5                         .                .                .
## RP11-34P13.7                  .                .                .
## RP11-34P13.8                  .                .                .
## AL627309.1                    .                .                .
```

Question:How many cells and how many genes are there in pbmc sample?

Because some of the cells have extremely low covered genes or some genes got very low UMIs, if we keep those for further analysis, it may introduce bias in our result. Thus, we will filter out genes and cells with low quality/coverage before we contiune the analysis. In this practical, we will only keep genes expressed in three or more UMIs per cells and keeping cells with at least 200 detected genes.

Question:How many genes and cells left after we filtered out those genes/cells with low coverage?

```
pbmc = CreateSeuratObject(counts = pbmc.data, project = "pbmc", min.cel
ls = 3, min.features = 200)

## Warning: Feature names cannot have underscores ('_'), replacing with
 dashes
## ('-')

#check the dimension
dim(pbmc)

## [1] 13714  2700

# see ?seurat for more information on the class
class(pbmc)

## [1] "Seurat"
## attr(,"package")
## [1] "SeuratObject"

#seurat object is a complex class with heterogeneous data type mixed to
gether (data.frame, vector,list etc.)
pbmc

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

slotNames(pbmc)

##  [1] "assays"       "meta.data"    "active.assay" "active.ident" "gr
aphs"
##  [6] "neighbors"    "reductions"   "images"       "project.name" "mi
```

```
sc"
## [11] "version"      "commands"     "tools"
```

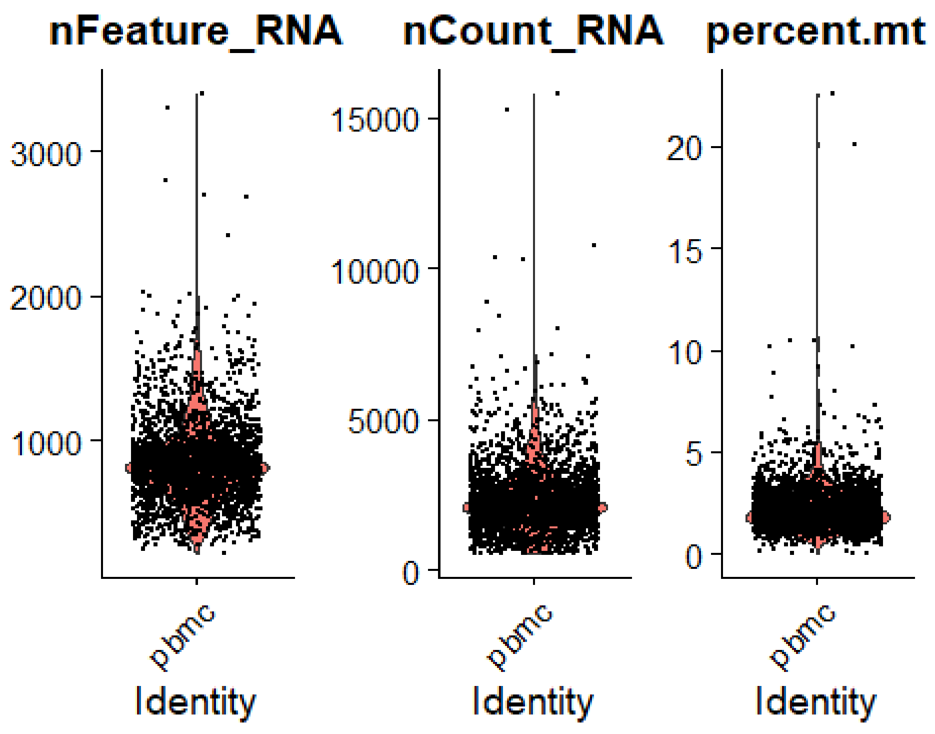
A common quality control metric is the percentage of transcripts from the mitochondrial genome. According to the paper *Classification of low quality cells from single-cell RNA-seq data (Ilicic T et al., 2016, Genome Biol.)* the reason this is a quality control metric is because if a single cell is lysed, cytoplasmic RNA will be lost apart from the RNA that is enclosed in the mitochondria, which will be retained and sequenced.

```
# mitochondria genes conveniently start with MT
# The [[ operator can add columns to object metadata. This is a great p
lace to stash QC stats
pbmc[["percent.mt"]] = PercentageFeatureSet(pbmc, pattern = "^MT-")
```

In the example below, we visualize QC metrics, and use these to filter cells.

- We filter cells that have unique feature counts (genes) over 6000 or less than 200
- We filter cells that have > 10% mitochondrial counts

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



```
summary(pbmc[["nFeature_RNA"]])
```

```
##   nFeature_RNA
##  Min.   : 212.0
##  1st Qu.: 690.0
##  Median : 816.0
##  Mean   : 845.5
##  3rd Qu.: 952.0
##  Max.   :3400.0
```

```
summary(pbmc[["nCount_RNA"]])
```

```
##    nCount_RNA
##  Min.   :  546
##  1st Qu.: 1756
##  Median : 2196
##  Mean   : 2365
##  3rd Qu.: 2762
##  Max.   :15818
```

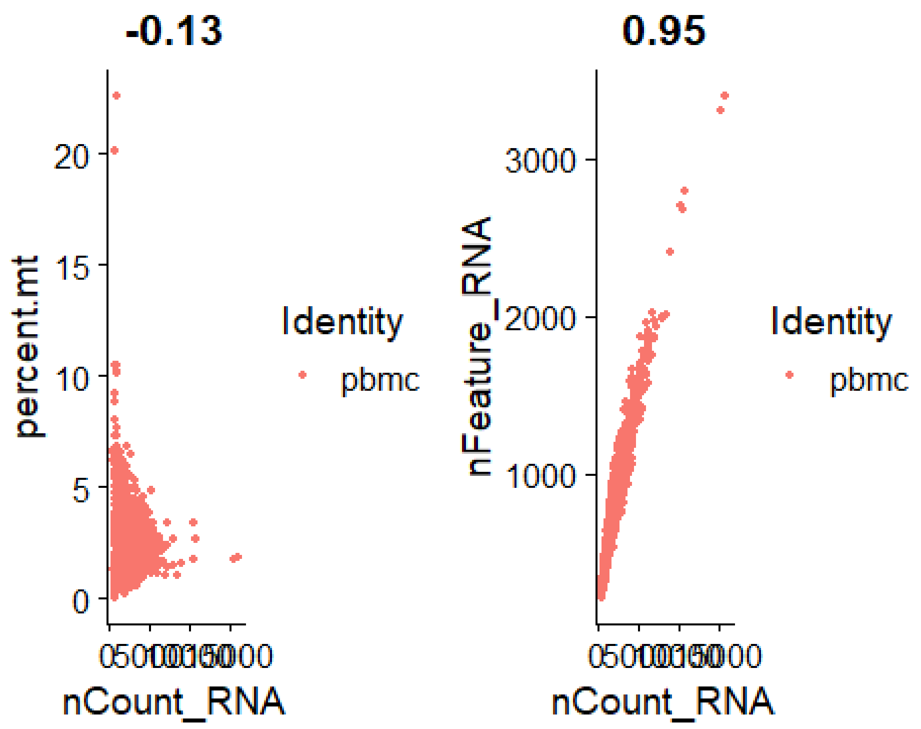
```
summary(pbmc[["percent.mt"]])
```

```
##    percent.mt
##  Min.   : 0.000
##  1st Qu.: 1.537
##  Median : 2.031
##  Mean   : 2.217
##  3rd Qu.: 2.643
##  Max.   :22.569
```

- nFeature_RNA : the number of genes detected in each cells.
- nCount_RNA : the total UMIs detected in each cell.
- percent.mt : mitochondrial genes percentage we calcualted before.

Questions: what conclusion you can get from this figure?

```
# FeatureScatter is typically used to visualize feature-feature relatio
nships, but can be used
# for anything calculated by the object, i.e. columns in object metadat
a, PC scores etc.
plot1 = FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "perce
nt.mt")
plot2 = FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "nFeat
ure_RNA")
plot1 + plot2
```



From the above quality control for this scRNAseq data, we can see that

1. majority of cells captured at least 200 genes no more than 2500 genes.
2. majority of cells have less than 10% mito genes.

Thus we will do the filtering according to this:

```
pbmc = subset(pbmc, subset = nFeature_RNA > 200 & nFeature_RNA <6000 &
percent.mt < 10)
dim(pbmc)

## [1] 13714  2694
```

Question:How many genes and cells left after we filtered out those genes/cells with extreme value or more mitochondrial genes?

3.4 Normalizing the data

After removing unwanted cells from the dataset, the next step is to normalize the data. By default, Seurat employ a global-scaling normalization method **LogNormalize** that normalizes the feature expression measurements for each cell by the total expression, multiplies this by a scale factor (10,000 by default), and log-transforms the result. Normalized values are stored in *pbmc[[“RNA”]]@data*.

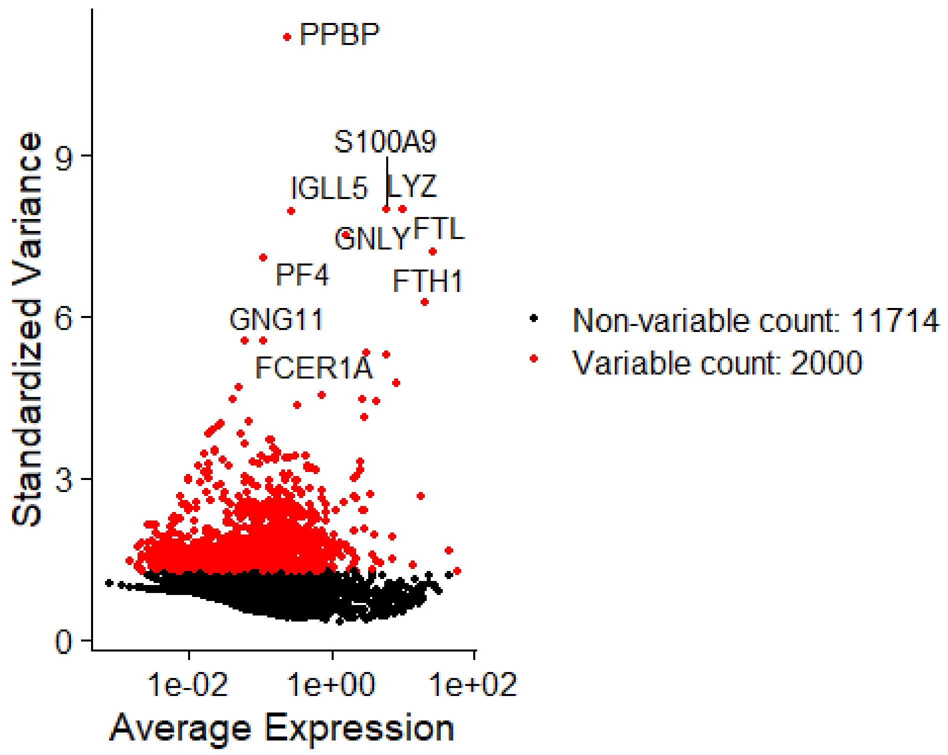
```
pbmc = NormalizeData(pbmc, normalization.method = "LogNormalize", scale.
factor = 10000)
```

3.5 Identification of highly variable features (feature selection)

We next calculate a subset of features that exhibit high cell-to-cell variation in the dataset (i.e, they are highly expressed in some cells, and lowly expressed in others). Scientists have found that focusing on these genes in downstream analysis helps to highlight biological signal in single-cell datasets.

By default, Seurat return 2,000 features per dataset. These will be used in downstream analysis, like PCA.

```
pbmc = FindVariableFeatures(pbmc, selection.method = "vst", nfeatures =
 2000)

# Identify the 10 most highly variable genes
top10 = head(VariableFeatures(pbmc), 10)

# plot variable features with and without labels
plot1 = VariableFeaturePlot(pbmc)
plot2 = LabelPoints(plot = plot1, points = top10, repel = TRUE)

## When using repel, set xnudge and ynudge to 0 for optimal results

plot2
```



3.6

Scaling the data Next, we apply a linear transformation (*scaling*) that is a standard pre-processing step prior to dimensional reduction techniques like PCA. The ScaleData function:

- Shifts the expression of each gene, so that the mean expression across cells is 0
- Scales the expression of each gene, so that the variance across cells is 1
  - This step gives equal weight in downstream analyses, so that highly-expressed genes do not dominate
- The results of this are stored in *pbmc[["RNA"]]@scale.data*

```
all.genes = rownames(pbmc)
pbmc = ScaleData(pbmc, features = all.genes)

## Centering and scaling data matrix
```

3.7 Perform linear dimensional reduction

Next we perform PCA on the scaled data. By default, only the previously determined variable features are used as input, but can be defined using features argument if you wish to choose a different subset. Seurat provides several useful ways of visualizing both cells and features that define the PCA, including *VizDimReduction*, *DimPlot*, and *DimHeatmap*

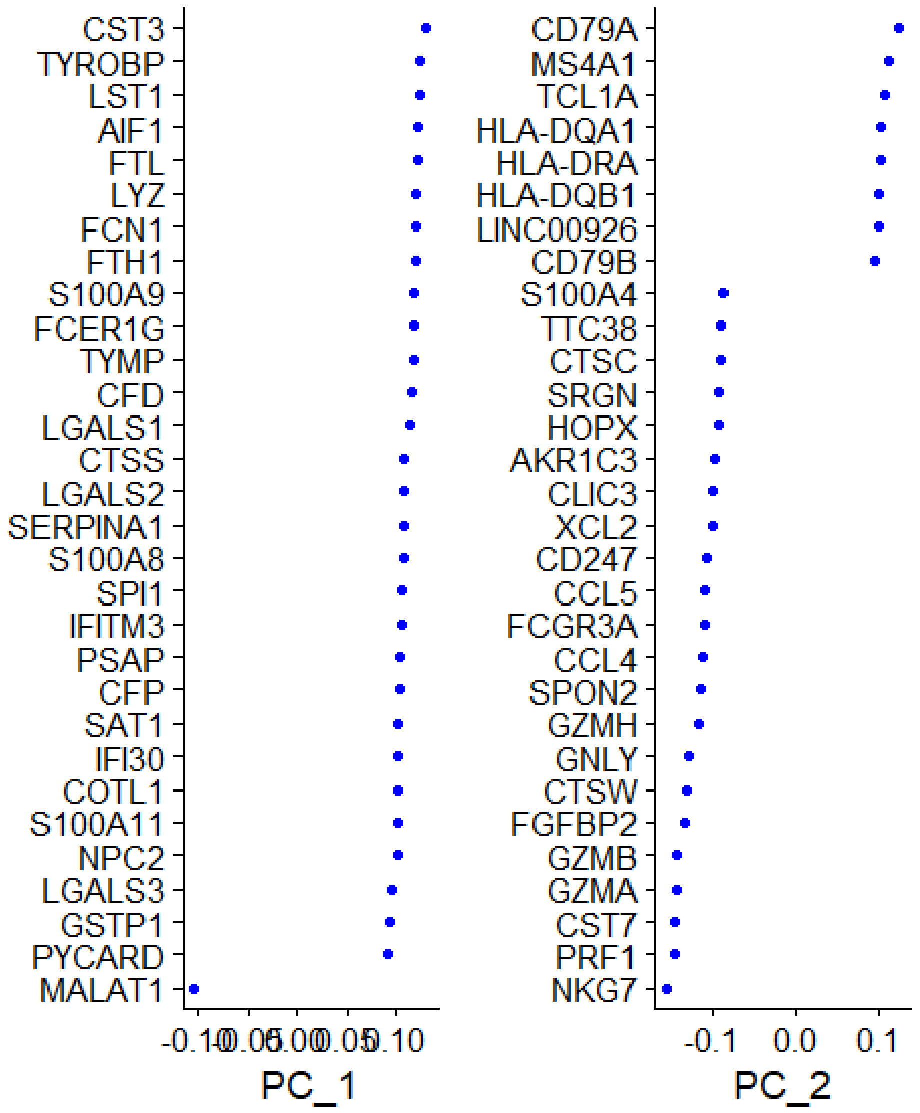
In particular *DimHeatmap* allows for easy exploration of the primary sources of heterogeneity in a dataset, and can be useful when trying to decide which PCs to include for further downstream analyses. Both cells and features are ordered according to their PCA scores. Setting cells to a number plots the ???extreme??? cells on both ends of the spectrum, which dramatically speeds plotting for large datasets. Though clearly a supervised analysis, we find this to be a valuable tool for exploring correlated feature sets.

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

## PC_ 1
## Positive:  CST3, TYROBP, LST1, AIF1, FTL, LYZ, FCN1, FTH1, S100A9, F
CER1G
##     TYMP, CFD, LGALS1, CTSS, LGALS2, SERPINA1, S100A8, SPI1, IFITM3,
 PSAP
##     CFP, SAT1, IFI30, COTL1, S100A11, NPC2, LGALS3, GSTP1, PYCARD, N
CF2
## Negative:  MALAT1, LTB, IL32, CD2, ACAP1, STK17A, CTSW, CD247, CCL5,
 GIMAP5
##     AQP3, GZMA, TRAF3IP3, CST7, MAL, HOPX, ITM2A, GZMK, MYC, BEX2
##     GIMAP7, ETS1, LDLRAP1, ZAP70, LYAR, RIC3, TNFAIP8, KLRG1, SAMD3,
 NKG7
## PC_ 2
## Positive:  CD79A, MS4A1, TCL1A, HLA-DQA1, HLA-DRA, HLA-DQB1, LINC009
26, CD79B, HLA-DRB1, CD74
##     HLA-DPB1, HLA-DMA, HLA-DQA2, HLA-DRB5, HLA-DPA1, HLA-DMB, HVCN1,
 FCRLA, LTB, BLNK
##     KIAA0125, P2RX5, IRF8, IGLL5, SWAP70, ARHGAP24, SMIM14, PPP1R14A,
 C16orf74, FCRL2
## Negative:  NKG7, PRF1, CST7, GZMA, GZMB, FGFBP2, CTSW, GNLY, GZMH, S
PON2
```

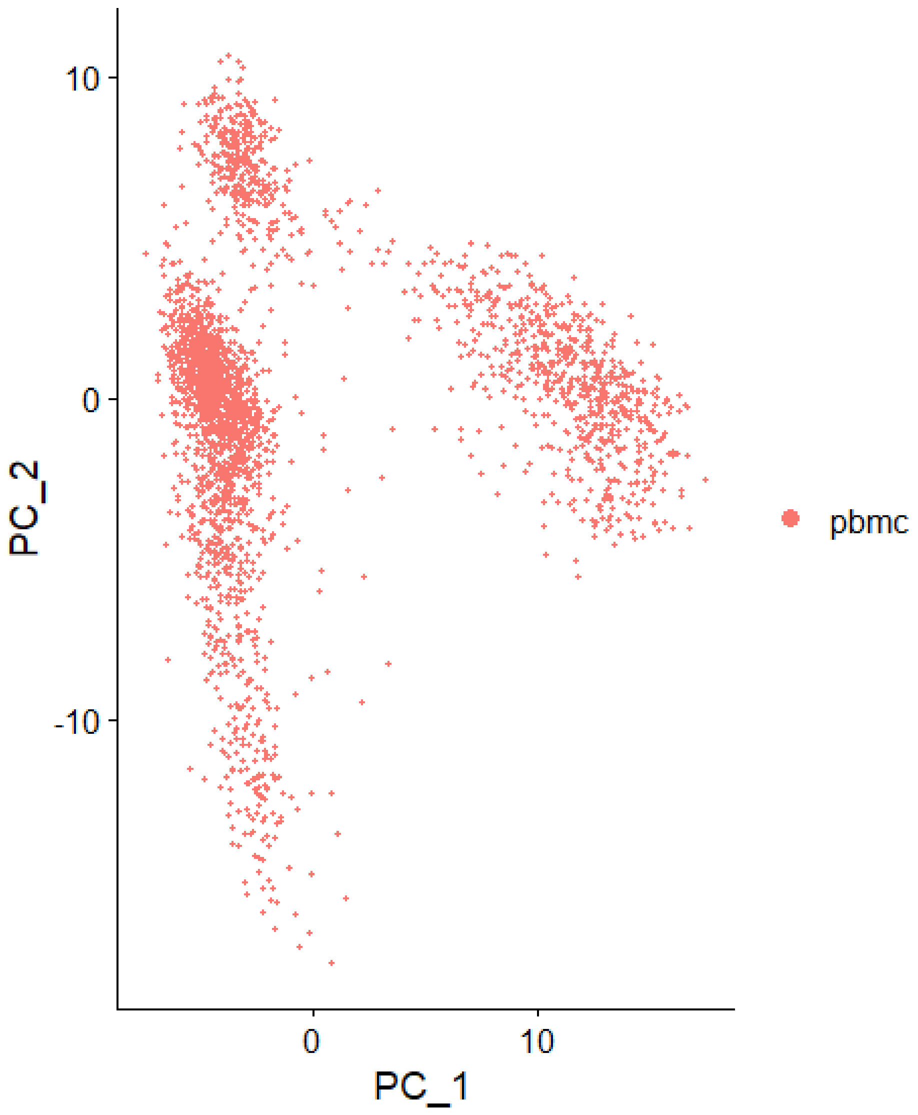
```
##     CCL4, FCGR3A, CCL5, CD247, XCL2, CLIC3, AKR1C3, HOPX, SRGN, CTSC

##     TTC38, S100A4, IL32, ANXA1, IGFBP7, ID2, ACTB, XCL1, APOBEC3G, S
AMD3
## PC_ 3
## Positive:  HLA-DQA1, CD79A, CD79B, HLA-DQB1, HLA-DPB1, CD74, HLA-DPA
1, MS4A1, HLA-DRB1, HLA-DRB5
##     HLA-DRA, HLA-DQA2, TCL1A, LINC00926, HLA-DMB, HLA-DMA, HVCN1, FC
RLA, IRF8, BLNK
##     KIAA0125, SMIM14, PLD4, IGLL5, P2RX5, SWAP70, LAT2, TMSB10, IGJ,
 MZB1
## Negative:  PPBP, PF4, SDPR, SPARC, GNG11, NRGN, GP9, RGS18, TUBB1, C
LU
##     HIST1H2AC, AP001189.4, ITGA2B, CD9, TMEM40, CA2, PTCRA, ACRBP, M
MD, NGFRAP1
##     TREML1, F13A1, RUFY1, SEPT5, MPP1, TSC22D1, CMTM5, RP11-367G6.3,
 MYL9, GP1BA
## PC_ 4
## Positive:  HLA-DQA1, HIST1H2AC, CD79A, PF4, CD79B, SDPR, PPBP, GNG11,
 HLA-DQB1, MS4A1
##     SPARC, CD74, GP9, HLA-DPB1, NRGN, RGS18, HLA-DQA2, PTCRA, CD9, H
LA-DRB1
##     AP001189.4, CLU, CA2, TUBB1, HLA-DPA1, TCL1A, HLA-DRA, ITGA2B, T
MEM40, LINC00926
## Negative:  VIM, S100A8, S100A6, S100A4, TMSB10, S100A9, IL32, GIMAP7,
 S100A10, LGALS2
##     RBP7, FCN1, MAL, LYZ, S100A12, MS4A6A, CD2, S100A11, FYB, GIMAP4

##     AQP3, FOLR3, ANXA1, MALAT1, AIF1, GIMAP5, IL8, IFI6, TRABD2A, AS
GR1
## PC_ 5
## Positive:  GZMB, FGFBP2, NKG7, GNLY, PRF1, CST7, CCL4, SPON2, GZMA,
GZMH
##     XCL2, CLIC3, CTSW, TTC38, CCL5, AKR1C3, IGFBP7, XCL1, S100A8, CC
L3
##     TYROBP, HOPX, CD160, S100A9, HAVCR2, PTGDR, FCER1G, LGALS2, RBP7,
 S100A12
## Negative:  LTB, VIM, AQP3, PPA1, MAL, KIAA0101, CD2, CORO1B, CYTIP,
FYB
##     IL32, TYMS, ANXA5, TRADD, TUBA1B, HN1, COTL1, ITM2A, PTGES3, GPR
183
##     TNFAIP8, ACTG1, ZWINT, ATP5C1, TRAF3IP3, GIMAP4, PRDX1, NGFRAP1,
 LDLRAP1, ABRACL
```

```r
print(pbmc[["pca"]], dims = 1:5, nfeatures = 5)
```

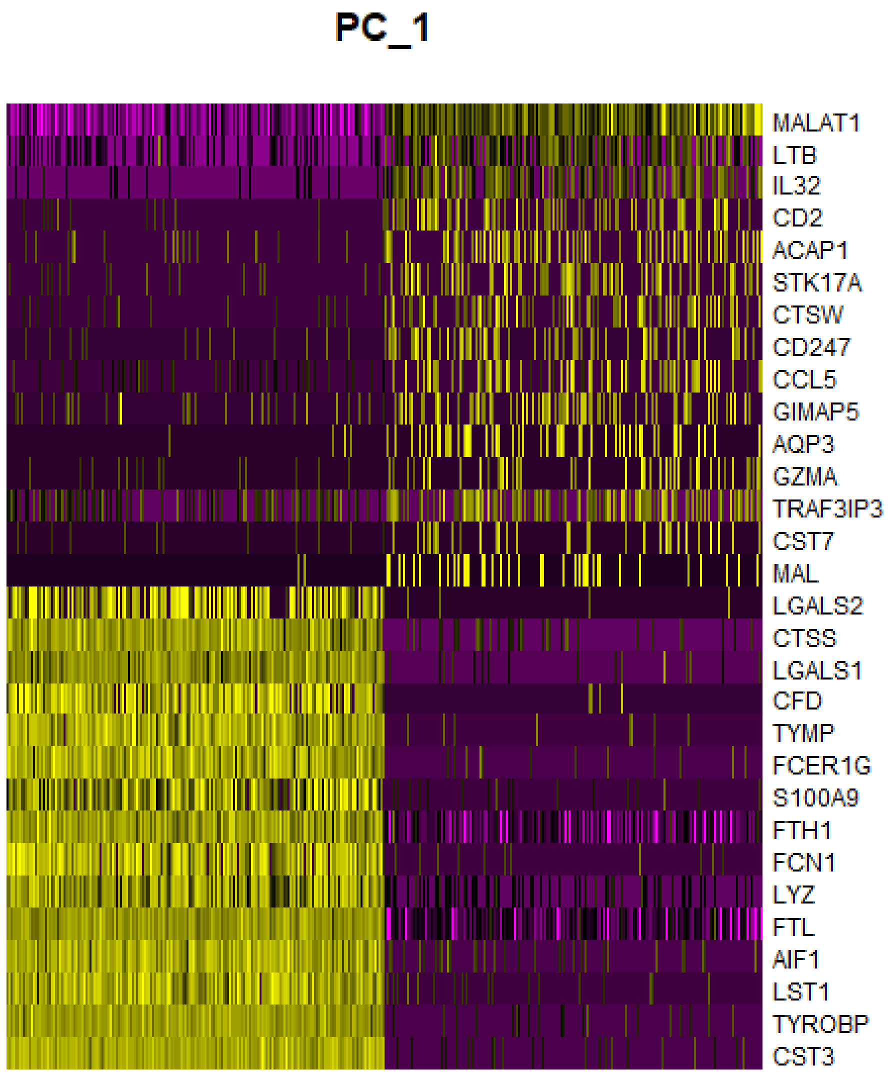
```
## PC_ 1
## Positive:  CST3, TYROBP, LST1, AIF1, FTL
## Negative:  MALAT1, LTB, IL32, CD2, ACAP1
## PC_ 2
## Positive:  CD79A, MS4A1, TCL1A, HLA-DQA1, HLA-DRA
## Negative:  NKG7, PRF1, CST7, GZMA, GZMB
## PC_ 3
## Positive:  HLA-DQA1, CD79A, CD79B, HLA-DQB1, HLA-DPB1
## Negative:  PPBP, PF4, SDPR, SPARC, GNG11
## PC_ 4
## Positive:  HLA-DQA1, HIST1H2AC, CD79A, PF4, CD79B
## Negative:  VIM, S100A8, S100A6, S100A4, TMSB10
## PC_ 5
## Positive:  GZMB, FGFBP2, NKG7, GNLY, PRF1
## Negative:  LTB, VIM, AQP3, PPA1, MAL

VizDimLoadings(pbmc, dims = 1:2, reduction = "pca")
```



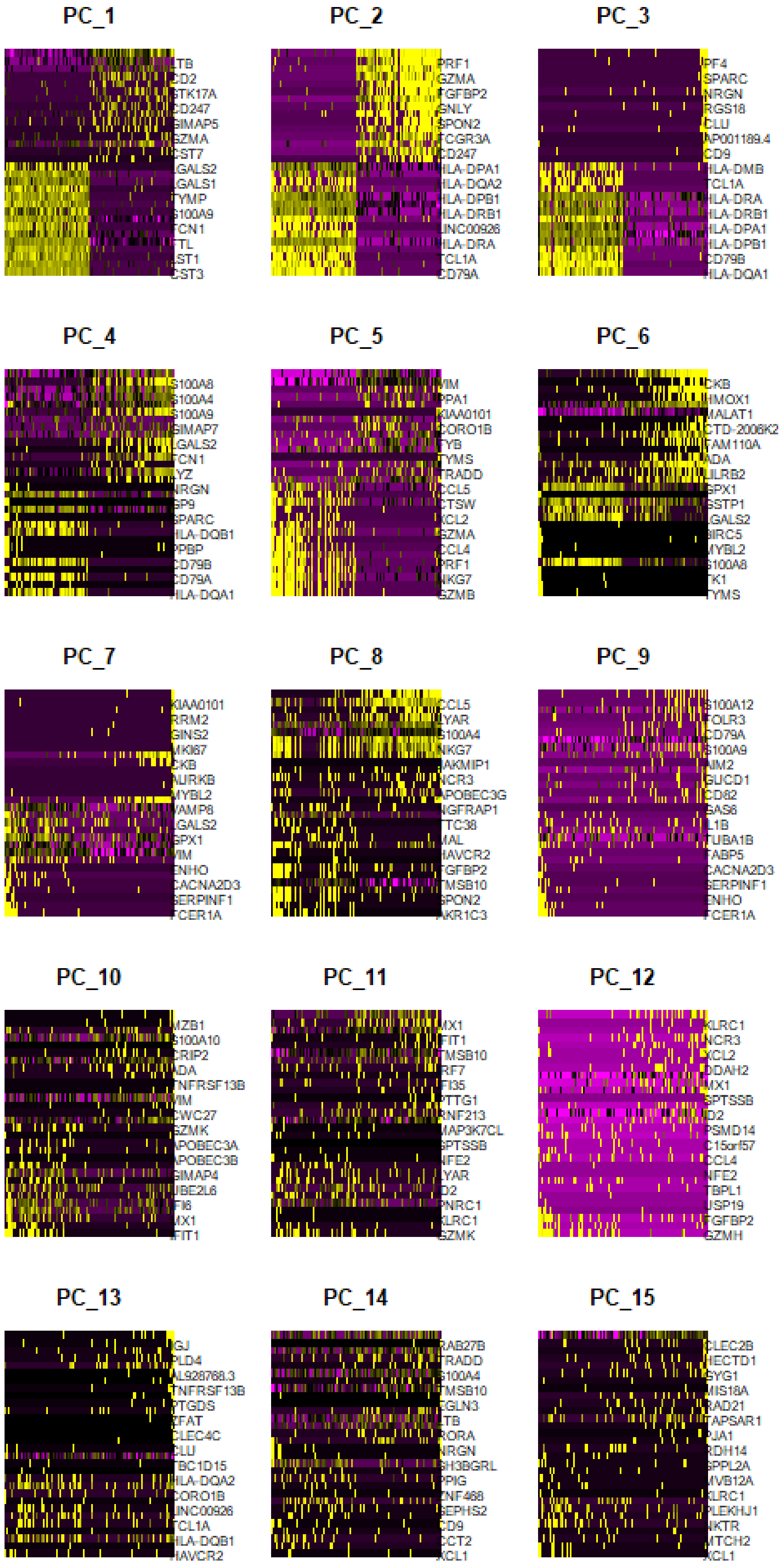
```
DimPlot(pbmc, reduction = "pca")
```



```
DimHeatmap(pbmc, dims = 1, cells = 500, balanced = TRUE)
```



```
DimHeatmap(pbmc, dims = 1:15, cells = 500, balanced = TRUE)
```

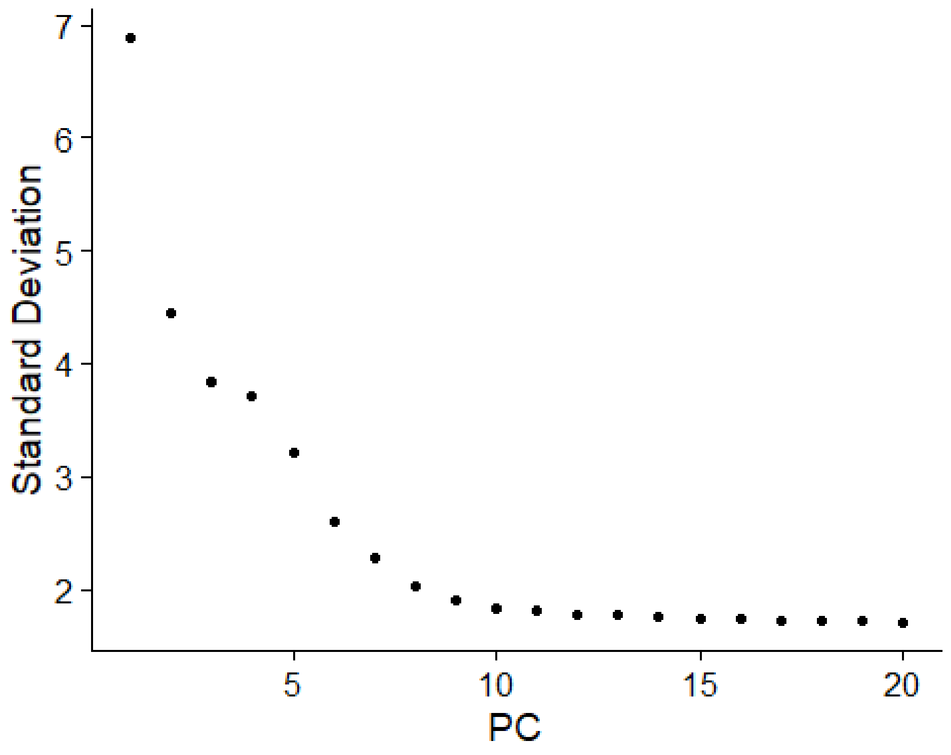

PC_1
LTB
CD2
STK17A
CD247
GIMAP5
GZMA
CST7
LGALS2
LGALS1
TYMP
S100A9
FCN1
FTL
LST1
CST3
PC_2
PRF1
GZMA
FGFBP2
GNLY
SPON2
FCGR3A
CD247
HLA-DPA1
HLA-DQA2
HLA-DPB1
HLA-DRB1
LINC00926
HLA-DRA
TCL1A
CD79A
PC_3
PF4
SPARC
NRGN
RGS18
CLU
AP001189.4
CD9
HLA-DMB
TCL1A
HLA-DRA
HLA-DRB1
HLA-DPA1
HLA-DPB1
CD79B
HLA-DQA1
PC_4
S100A8
S100A4
S100A9
GIMAP7
LGALS2
FCN1
LYZ
NRGN
GP9
SPARC
HLA-DQB1
PPBP
CD79B
CD79A
HLA-DQA1
PC_5
VIM
PPA1
KIAA0101
CORO1B
TYB
TYMS
TRADD
CCL5
CTSW
XCL2
GZMA
CCL4
PRF1
NKG7
GZMB
PC_6
CKB
HMOX1
MALAT1
CTD-2006K2
FAM110A
ADA
LILRB2
GPX1
GSTP1
LGALS2
BIRC5
MYBL2
S100A8
TK1
TYMS
PC_7
KIAA0101
RRM2
GINS2
MKI67
CKB
AURKB
MYBL2
VAMP8
LGALS2
GPX1
VIM
ENHO
CACNA2D3
SERPINF1
FCER1A
PC_8
CCL5
LYAR
S100A4
NKG7
AKMIP1
NCR3
APOBEC3G
NGFRAP1
TTC38
MAL
HAVCR2
FGFBP2
TMSB10
SPON2
AKR1C3
PC_9
S100A12
FOLR3
CD79A
S100A9
NM2
GUCD1
CD82
SAS6
IL1B
TUBA1B
FABP5
CACNA2D3
SERPINF1
ENHO
FCER1A
PC_10
MZB1
S100A10
CRIP2
ADA
TNFRSF13B
VIM
CWC27
GZMK
APOBEC3A
APOBEC3B
GIMAP4
UBE2L6
IFI6
MX1
IFIT1
PC_11
MX1
IFIT1
TMSB10
IRF7
IFI35
PTTG1
RNF213
MAP3K7CL
SPTSSB
NFE2
LYAR
ID2
PNRC1
KLRC1
GZMK
PC_12
KLRC1
NCR3
XCL2
DDAH2
MX1
SPTSSB
ID2
PSMD14
C15orf57
CCL4
NFE2
TBPL1
USP19
FGFBP2
GZMH
PC_13
IGJ
PLD4
AL928768.3
TNFRSF13B
PTGDS
ZFAT
CLEC4C
CLU
TBC1D15
HLA-DQA2
CORO1B
LINC00926
TCL1A
HLA-DQB1
HAVCR2
PC_14
RAB27B
TRADD
S100A4
TMSB10
EGLN3
LTB
RORA
NRGN
SH3BGRL
PPIG
ZNF468
SEPHS2
CD9
CCT2
XCL1
PC_15
CLEC2B
HECTD1
SYG1
MIS18A
RAD21
TAPSAR1
PJA1
RDH14
SPPL2A
MVB12A
KLRC1
PLEKHJ1
NKTR
MTCH2
XCL1

3.8 Determine the ???dimensionality??? of the dataset

To overcome the extensive technical noise in any single feature for scRNA-seq data, Seurat clusters cells based on their PCA scores, with each PC essentially representing a *metafeature* that combines information across a correlated feature set. The top principal components therefore represent a robust compression of the dataset. However, how many componenets should we choose to include? 10? 20? 100?

One of the methods generates an ???Elbow plot???: a ranking of principle components based on the percentage of variance explained by each one (ElbowPlot function). In this example, we can observe an ???elbow??? around PC9-10, suggesting that the majority of true signal is captured in the first 17 PCs.

```
ElbowPlot(pbmc)
```



3.9 Cluster the cells

Since there might be heterogenous in the lung tissue, i.e. there might be different cell subtypes. We want to cluster the cells from the scRNAseq data.

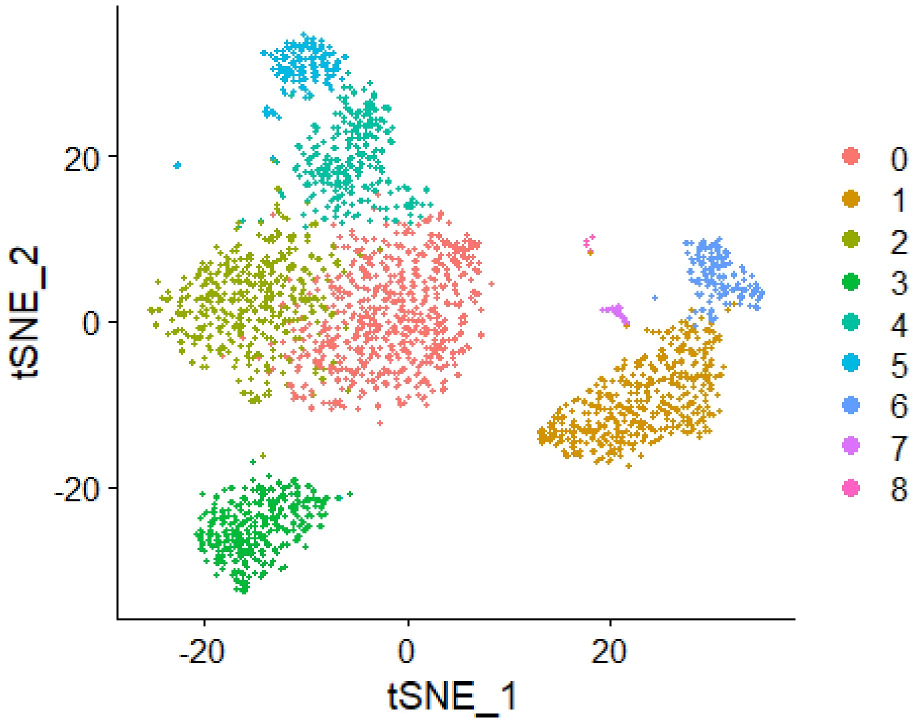
To cluster the cells, Seurat next apply modularity optimization techniques such as the Louvain algorithm (default) or SLM [SLM, Blondel et al., Journal of Statistical Mechanics], to iteratively group cells together, with the goal of optimizing the standard modularity function. The FindClusters function implements this procedure, and contains a resolution parameter that sets the ???granularity??? of the downstream clustering, with increased values leading to a greater number of clusters. Seurat find that setting this parameter between 0.4-1.2 typically returns good results for single-cell datasets of around 3K cells. Optimal resolution often increases for larger datasets. The clusters can be found using the Idents function.

```
pbmc = FindNeighbors(pbmc, dims = 1:17)

## Computing nearest neighbor graph

## Computing SNN

pbmc = FindClusters(pbmc, resolution = 0.5)
head(Idents(pbmc), 5)
```

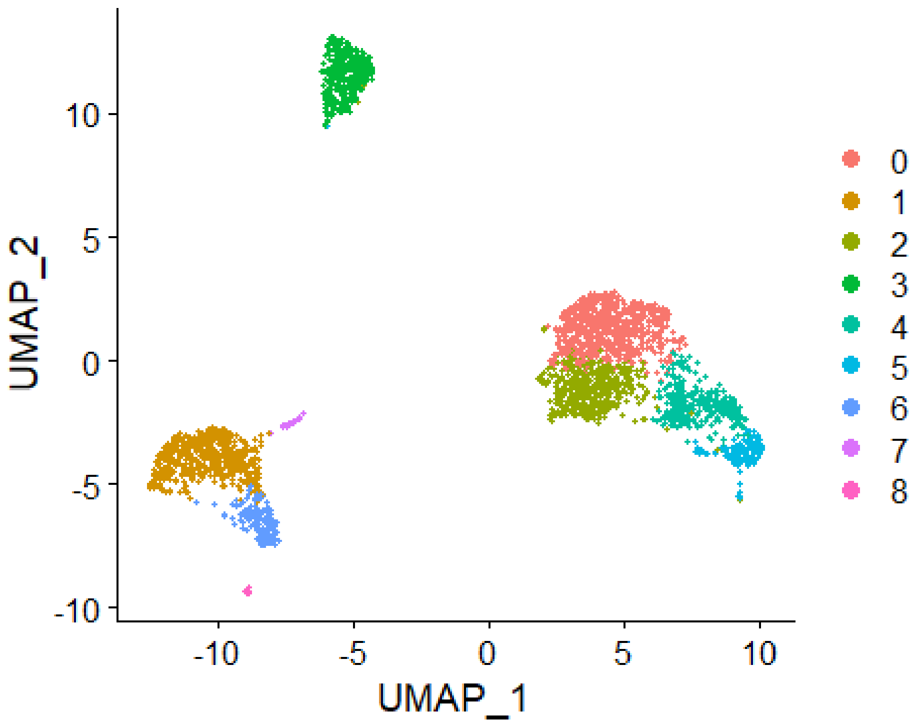
3.10 Run non-linear dimensional reduction (UMAP/tSNE)

Seurat offers several non-linear dimensional reduction techniques, such as tSNE and UMAP, to visualize and explore these datasets. The goal of these algorithms is to learn the underlying manifold of the data in order to place similar cells together in low-dimensional space. Cells within the graph-based clusters determined above should co-localize on these dimension reduction plots. As input to the UMAP and tSNE, seurat suggest using the same PCs as input to the clustering analysis.

```
#here you can also use T-SNE  to do the dimension reduction
pbmc = RunTSNE(pbmc, dims = 1:17)
DimPlot(pbmc, reduction = "tsne")
```



```
# If you haven't installed UMAP, you can do so via reticulate::py_insta
ll(packages = 'umap-learn')
pbmc = RunUMAP(pbmc, dims = 1:17)

## Warning: The default method for RunUMAP has changed from calling Pyt
hon UMAP via reticulate to the R-native UWOT using the cosine metric
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' a
nd metric to 'correlation'
## This message will be shown once per session

## 11:59:32 UMAP embedding parameters a = 0.9922 b = 1.112

## 11:59:32 Read 2694 rows and found 17 numeric columns

## 11:59:32 Using Annoy for neighbor search, n_neighbors = 30

## 11:59:32 Building Annoy index with metric = cosine, n_trees = 50

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

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

## **************************************************|
## 11:59:33 Writing NN index file to temp file D:\系统缓存\RtmpSwjm2v\fi
le334446fa4e3c
## 11:59:33 Searching Annoy index using 1 thread, search_k = 3000
## 11:59:33 Annoy recall = 100%
## 11:59:34 Commencing smooth kNN distance calibration using 1 thread
## 11:59:34 Initializing from normalized Laplacian + noise
## 11:59:34 Commencing optimization for 500 epochs, with 110290 positiv
e edges
## 11:59:41 Optimization finished

# note that you can set `label = TRUE` or use the LabelClusters functio
n to help label individual clusters
DimPlot(pbmc, reduction = "umap")
```

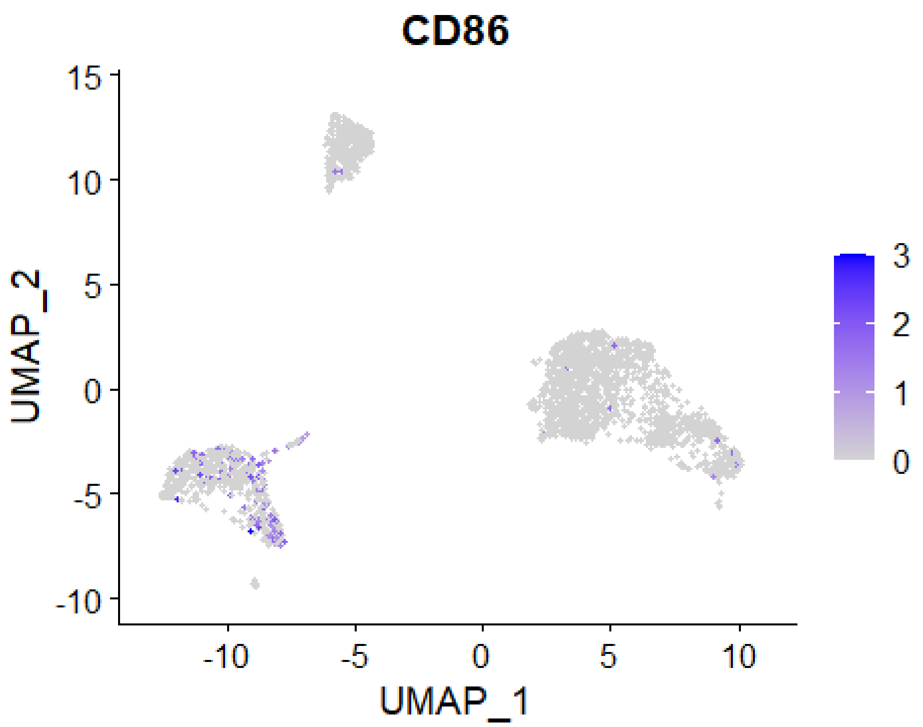


Question:What's your feeling about UMAP vs tSNE? Does they give similar results?

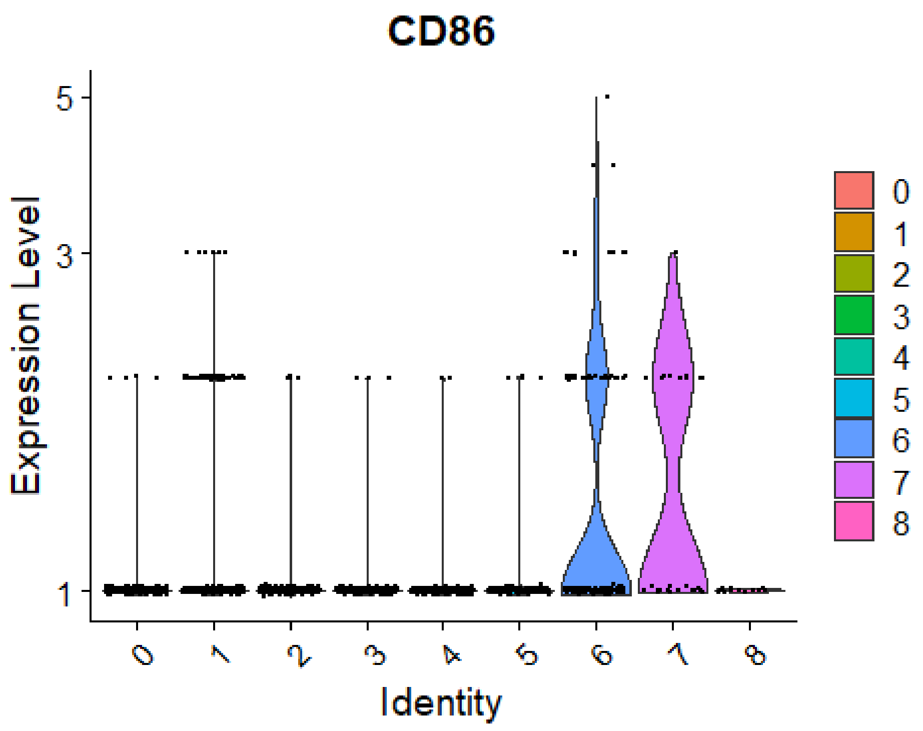
3.11 CD86 expression in different clusters

FeaturePlot can be used to visualize the expression of centain genes in different cluster.

```
FeaturePlot(pbmc, features = c("CD86"))
```



```
VlnPlot(pbmc, features = c("CD86"), slot = "counts", log = TRUE)
```



What's your conclusion from above figures?

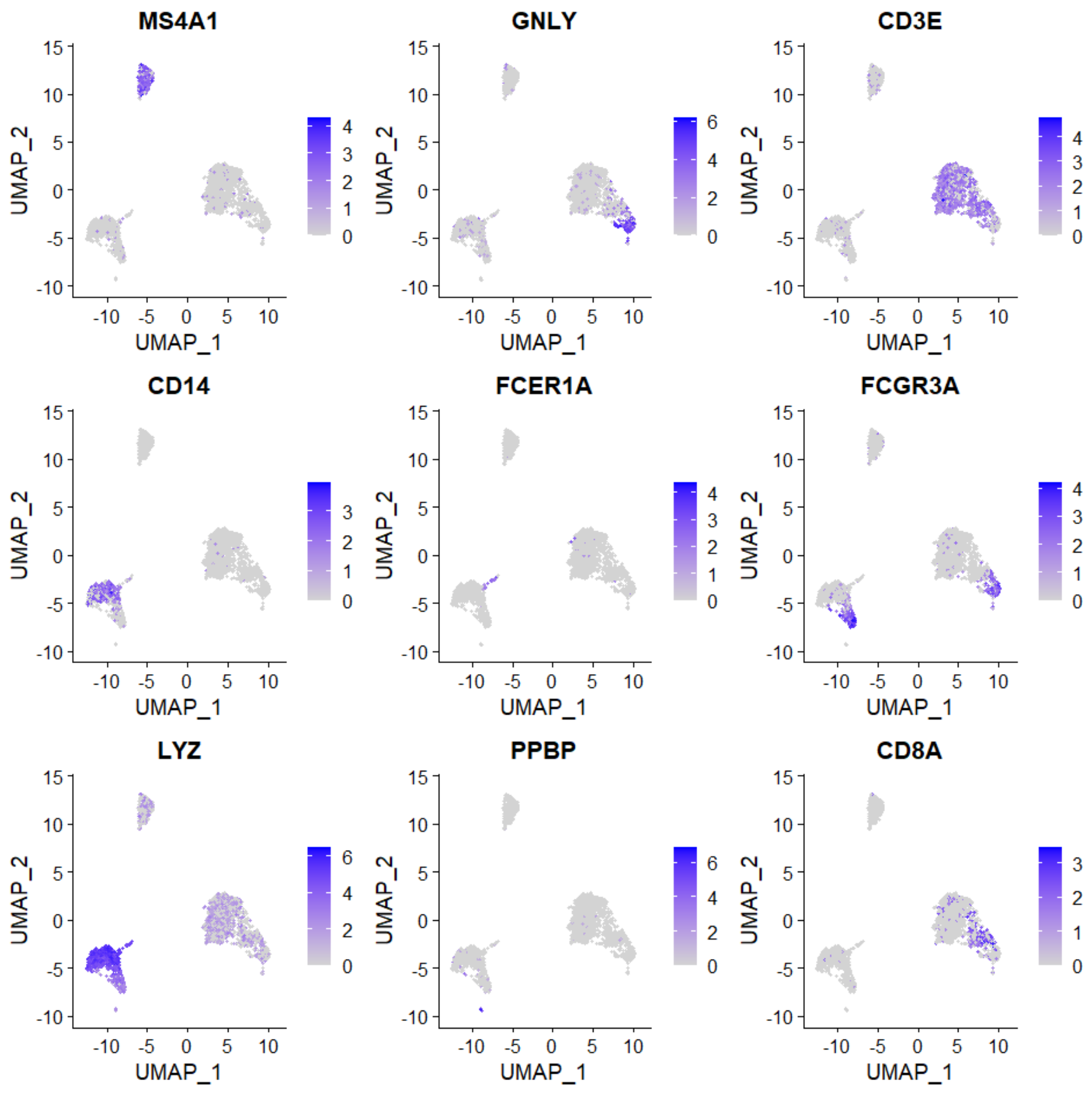
3.12 Finding differentially expressed features (cluster biomarkers)

From previous results, seems like cluster 0 is the only cell subtypes that have high expression of CD86. Then what type is this cell subtypes?

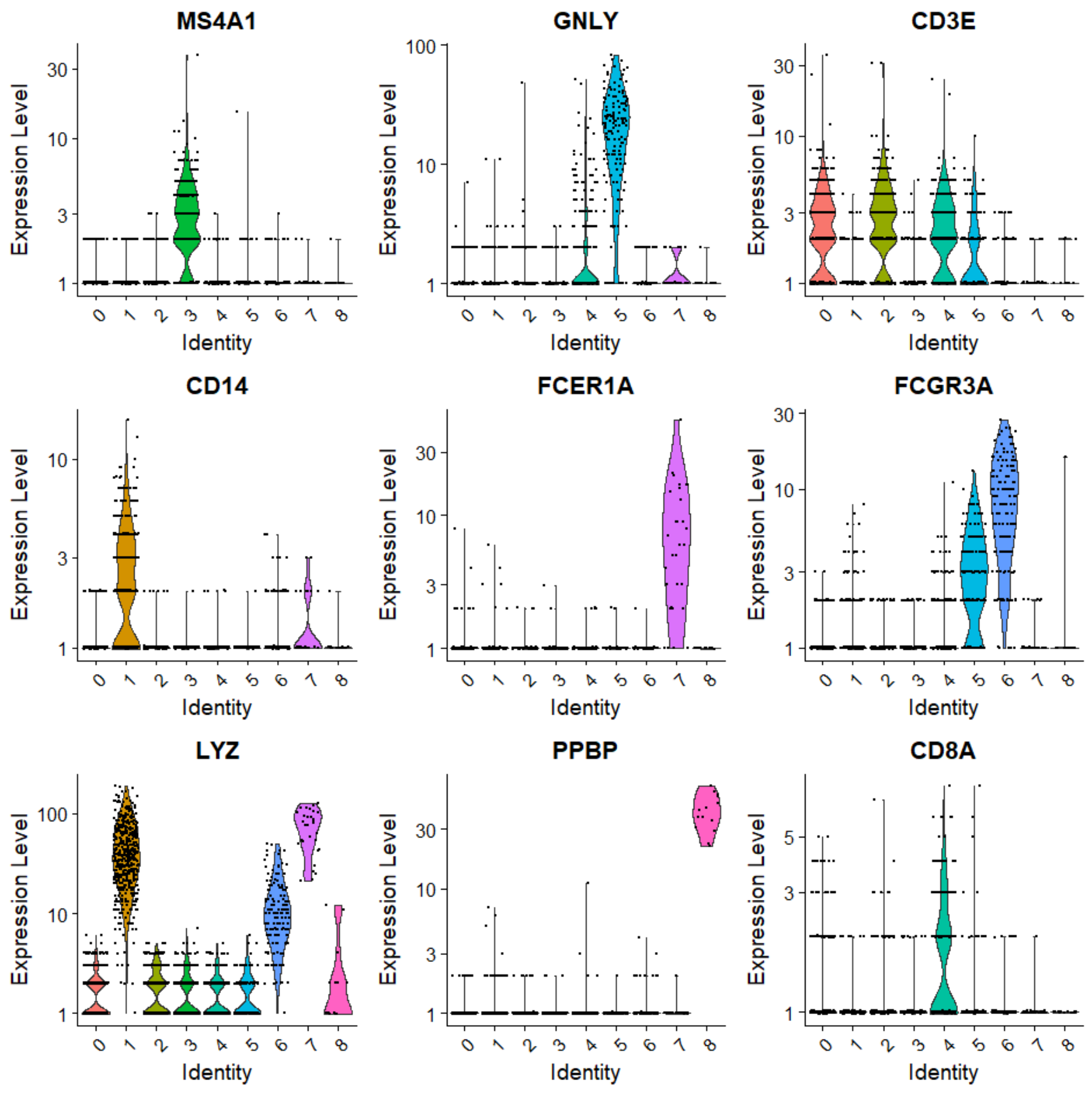
Seurat can help you find markers that define clusters via differential expression. By default, it identifes positive and negative markers of a single cluster (specified in *ident.6,7*), compared to all other cells. *FindAllMarkers* automates this process for all clusters, but you can also test groups of clusters vs. each other, or against all cells.

The *min.pct* argument requires a feature to be detected at a minimum percentage in either of the two groups of cells, and the *thresh.test* argument requires a feature to be differentially expressed (on average) by some amount between the two groups. You can set both of these to 0, but with a dramatic increase in time - since this will test a large number of features that are unlikely to be highly discriminatory. As another option to speed up these computations, *max.cells.per.ident* can be set. This will downsample each identity class to have no more cells than whatever this is set to. While there is generally going to be a loss in power, the speed increases can be significiant and the most highly differentially expressed features will likely still rise to the top.

```
# find all markers of cluster 1. This step might take couple of mins.
cluster0.markers = FindMarkers(pbmc, ident.1 = 0, min.pct = 0.25)
head(cluster0.markers, n = 6)

##               p_val avg_log2FC pct.1 pct.2     p_val_adj
## RPL32 5.109693e-143  0.6290521 0.999 0.995 7.007433e-139
## RPS12 1.388308e-140  0.7172400 1.000 0.990 1.903925e-136
## RPS6  4.155142e-138  0.6560843 0.997 0.995 5.698361e-134
## RPS27 8.504616e-138  0.7124785 0.999 0.992 1.166323e-133
## RPS25 1.403810e-127  0.7606019 0.997 0.974 1.925186e-123
## RPL31 4.103179e-127  0.7700598 0.997 0.963 5.627100e-123

FeaturePlot(pbmc, features = c("MS4A1", "GNLY", "CD3E", "CD14", "FCER1A
", "FCGR3A", "LYZ", "PPBP",
      "CD8A"))
```



```
VlnPlot(pbmc, features = c("MS4A1", "GNLY", "CD3E", "CD14", "FCER1A", "
FCGR3A", "LYZ", "PPBP","CD8A"), slot = "counts", log = TRUE)
```



Is there any other conclusion you can draw from this analysis?

3.13 Some other stuff you can explore (Optional)

Seems like we identified 8 cell subtypes from this scRNAseq dataset from lung. What's the biomarkers for other cell subtypes? Can you assign those cell subtypes to different cell types? You can refer to this paper which analyze the same set of data.

(#^.^#)Enjoy Exploring!!!