HarvardX Data Science Professional Certificate Capstone Project: Single-Cell RNA and Protein Profiling

Delong Meng

1/1/2020

Introduction

Single cell RNA-sequencing (scRNA-seq) is a new technology that is rapidly growing recently in the biomedical science field. Prior to scRNA-seq, people use so-called bulk RNA-seq to perform transcriptome analysis, namely levels of gene expression of the whole genome in different groups of biomedical samples, such as tumor samples and normal control tissues, or cells of different treatments. By comparing gene expression in different samples, we usually get a list of differentially expressed genes, associated with disease or cell status. However, this bulk RNA-seq technique ignores the heterogeneity within a group and only capture the average gene expression of different cell in a group.

Through scRNA-seq, we can obtain gene expression levels of the whole genome in each individual cell. This is achived by labeling each cell with a unique barcode of DNA sequence. When we amplify the complementary DNA (cDNA) which is synthesized from RNAs, the unique barcode will also be amplified and eventually provide the cell identity information after the amplified DNAs are sequenced. The task of scRNA-seq data analysis is to perform an unsupervised machine learning process, and explore and identify the patterns of variation in cell states. A common strategy is to reduce the dimentionality and cluster the cells into discrete groups in a 2-D space, and identify the corresponding gene expression markers that distinguish these separated groups. We believe that cell populations with similar molecular features might also have similar functions and behaviors under pathophysiological conditions.

For this Capstone project, I will use a public dataset of scRNA-seq study of human cord blood mononuclear cells (CBMCs). Interestingly, the authors of this study creatively developed a new approach to simultaneously analyze RNA expression levels and cell surface protein levels (i.e., epitopes), which they called "cellular indexing of transcriptomes and epitopes by sequencing" (CITE-seq) [1]. Cell surface protein markers are critical indicators of immunophenotypes of blood cells, and are usually measured by flow cytometry, in which fluorescence labeled antibodies will be used to recognize these protein markers and quantified based on the fluorescence. The brief concept of this CITE-seq is that they conjugate the antibodies to oligonucleotides (oligos) that can be captured in the process of scRNA-seq library preparations. This oligo contains a unique barcode that can provide the information of the antibody in later analysis, and can be released from the antibody during cell lysis, generating so-called antibody-derived tags (ADTs) that can be labeled by the same cellular barcode with the RNAs and amplified together with the RNAs. This new approach makes it possible to analyze different levels of molecular features (RNAs and proteins here) at the same time, and achieve more detailed characterization of cellular phenotypes.

Here I will mainly use the Seurat R package to perform the single cell analysis. Seurat is a popular and powerful toolkit that provides solutions for each stage of single cell analysis. Developed by the *Satija lab*, now the most recent version is *Seurat Version 3* [2]. Very helpful tutorials can also be found in their webpage: https://satijalab.org/seurat/.

Methods

1. Installation of essential packages

I first set up the working envrionment by installing essential packages. To avoid re-installing the same packages, I used the "if(cond) expr" construct to check if the package has already been installed. The packages were then loaded.

```
if(!require(data.table)) install.packages("data.table")
if(!require(Seurat)) install.packages("Seurat")
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(devtools)) install.packages("devtools")
if(!require(BiocManager)) install.packages("BiocManager")
if(!require(dplyr)) install.packages("dplyr")
if(!require(ggplot2)) install.packages("ggplot2")
if(!require(kableExtra)) install.packages("kableExtra")
```

```
library(data.table)
library(Seurat)
library(tidyverse)
library(Matrix)
library(dplyr)
library(ggplot2)
library(kableExtra)
library(tinytex)
```

2. Data downloading and preparation

The data generated from the paper *Stoeckius et. al., 2017, Nature Methods* [1] have been deposited to the Gene Expression Omnibus (GEO) with the accession code GSE100866, and can be found here: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE100866.

From the webpage I first obtained the URLs, and then downloaded three *.csv* datasets that we need into local working direction (in the format of the *.gz* compressed files). The downloaded files were read and saved as *.Rdata* objects. The *.Rdata* objects can be easily loaded when necessary.

I used the *if... else...* control-flow construct to check if the *.Rdata* files already exist before re-downloading them.

```
if(file.exists("CBMC8K_RNA_umi.Rdata")) load("CBMC8K_RNA_umi.Rdata") else{
  url_CBMC8K_RNA_umi <- "https://www.ncbi.nlm.nih.gov/geo/download/?acc=GSE100866&format=
  file&file=GSE100866%5FCBMC%5F8K%5F13AB%5F10X%2DRNA%5Fumi%2Ecsv%2Egz"
  download.file(url_CBMC8K_RNA_umi, destfile = "./CBMC8K_RNA_umi.csv.gz")
  CBMC8K_RNA_umi <- read.csv("CBMC8K_RNA_umi.csv.gz", header=T, row.names=1)
  save(CBMC8K_RNA_umi, file = "CBMC8K_RNA_umi.Rdata")
  load("CBMC8K_RNA_umi.Rdata")
}

if(file.exists("CBMC8K_ADT_umi.Rdata")) load("CBMC8K_ADT_umi.Rdata") else{
  url_CBMC8K_ADT_umi <- "https://www.ncbi.nlm.nih.gov/geo/download/?acc=GSE100866&format=
  file&file=GSE100866%5FCBMC%5F8K%5F13AB%5F10X%2DADT%5Fumi%2Ecsv%2Egz"
  download.file(url_CBMC8K_ADT_umi, destfile = "./CBMC8K_ADT_umi.csv.gz")
  CBMC8K_ADT_umi <- read.csv("CBMC8K_ADT_umi.csv.gz", header=T, row.names=1)
```

```
  save(CBMC8K_ADT_umi, file = "CBMC8K_ADT_umi.Rdata")
  load("CBMC8K_ADT_umi.Rdata")
}

dim(CBMC8K_RNA_umi)
```

```
## [1] 36280  8617
```

```
dim(CBMC8K_ADT_umi)
```

```
## [1]   13 8617
```

We can see that the RNA dataset has 36280 lines (meaning 36280 genes) and the protein (ADT) dataset has 13 lines (meaning 13 proteins). They both have 8617 columns (8617 individual cells). UMI stands for "unique molecular identifier", and is the principal quantification method used for gene expression in scRNA-seq. Basically we can consider a UMI count as a copy of a given gene in a cell.

3. Separating human and mouse cells

According to the manuscript [1], to estimate the non-specific background binding of the protein antibodies, the authors spiked in 4% mouse cells into human cord blood mononuclear cells (CBMCs), and sequenced for both human genes and mouse genes. Human genes have the "HUMAN_" prefix and mouse genes have the "MOUSE_" prefix. As mentioned above, there are a total of 36280 genes (features) and 8617 cells. We first need to assign the species to each cell. A cell is considered as a human cell if more than 90% of the UMI counts are from human genes, a mouse cell if more than 90% of the UMI counts are from mouse genes, otherwise a "mixed" cell (probably due to droplets containing two or more cells from different species).

```
length(rownames(CBMC8K_RNA_umi))
```

```
## [1] 36280
```

```
n_distinct(rownames(CBMC8K_RNA_umi)) # make sure there's no redundant names
```

```
## [1] 36280
```

```
rownames(CBMC8K_RNA_umi)[1:10] # note that the first one starts with ERCC, which we'll remove later
```

```
##  [1] "ERCC_ERCC-00104" "HUMAN_A1BG"      "HUMAN_A1BG-AS1"  "HUMAN_A1CF"
##  [5] "HUMAN_A2M"       "HUMAN_A2M-AS1"   "HUMAN_A2ML1"     "HUMAN_A4GALT"
##  [9] "HUMAN_A4GNT"     "HUMAN_AAAS"
```

```
rownames(CBMC8K_RNA_umi)[36271:36280]
```

```
##  [1] "MOUSE_mt-Ti"    "MOUSE_mt-Tl1"   "MOUSE_mt-Tm"    "MOUSE_mt-Tp"
##  [5] "MOUSE_mt-Tq"    "MOUSE_mt-Tt"    "MOUSE_mt-Tw"    "MOUSE_n-R5s200"
##  [9] "MOUSE_n-R5s25"  "MOUSE_n-R5s31"
```

```
species <- tidyr::separate(as.data.frame(rownames(CBMC8K_RNA_umi)),
                           col="rownames(CBMC8K_RNA_umi)",
                           into=c("species","gene"),sep="_")
dim(species)
```

```
## [1] 36280     2
```

```
sum(species$species=="HUMAN") # number of human genes
```

```
## [1] 20400
```

```
sum(species$species=="MOUSE") # number of mouse genes
```

```
## [1] 15879
```

```
index_human <- species$species=="HUMAN"
index_mouse <- species$species=="MOUSE"
index_human_or_mouse <- species$species=="HUMAN"|species$species=="MOUSE"

CBMC8K_RNA_umi_human <- CBMC8K_RNA_umi[index_human,] # subset of human genes
# brief overview of some data
# note that the names of the cells are the unique barcodes corresponding to each cell
CBMC8K_RNA_umi_human[1:10,1:3]
```

```
##                CTGTTTACACCGCTAG CTCTACGGTGTGGCTC AGCAGCCAGGCTCATT
## HUMAN_A1BG                   0                0                0
## HUMAN_A1BG-AS1               0                0                0
## HUMAN_A1CF                   0                0                0
## HUMAN_A2M                    0                0                0
## HUMAN_A2M-AS1                0                0                0
## HUMAN_A2ML1                  0                0                0
## HUMAN_A4GALT                 0                0                0
## HUMAN_A4GNT                  0                0                0
## HUMAN_AAAS                   0                0                0
## HUMAN_AACS                   0                0                0
```

```
CBMC8K_RNA_umi_human[101:110,523:525]
```

```
##                  ACACTGATCCAAATGC GTACGTAGTTCCCTTG CCGGGATTCAATAAGG
## HUMAN_ABTB2                    0                0                0
## HUMAN_AC000003.2               0                0                0
## HUMAN_AC000036.4               0                0                0
## HUMAN_AC000068.5               0                0                0
## HUMAN_AC000111.6               0                0                0
## HUMAN_AC002044.4               0                0                0
## HUMAN_AC002116.7               0                0                0
## HUMAN_AC002117.1               0                0                0
## HUMAN_AC002306.1               0                0                0
## HUMAN_AC002310.12              0                0                0
```

```
CBMC8K_RNA_umi_mouse <- CBMC8K_RNA_umi[index_mouse,] # subset of mouse genes
CBMC8K_RNA_umi_mouse[1:10,1:3]
```

```
##                      CTGTTTACACCGCTAG CTCTACGGTGTGGCTC AGCAGCCAGGCTCATT
## MOUSE_0610007N19Rik               14                2                4
## MOUSE_0610007P14Rik                2                9                7
## MOUSE_0610009B22Rik                6                2                1
## MOUSE_0610009D07Rik               23               14               24
## MOUSE_0610009E02Rik                0                0                0
## MOUSE_0610009L18Rik                0                1                0
## MOUSE_0610009O20Rik                0                0                3
## MOUSE_0610010F05Rik                0                1                2
## MOUSE_0610010K14Rik                3                0                0
## MOUSE_0610011F06Rik                4                1                2
```

```
# for each cell, calculate the percentage of counts for human genes
percent_human <- 0
for(i in 1:8617){
  percent_human[i] <- sum(CBMC8K_RNA_umi_human[,i])/
    (sum(CBMC8K_RNA_umi_human[,i])+sum(CBMC8K_RNA_umi_mouse[,i]))
}
head(percent_human)
```

```
## [1] 0.02151877 0.04297119 0.02301669 0.02414594 0.02487352 0.02357249
```

```
cell_human_index <- percent_human>0.9
cell_mouse_index <- percent_human<0.1
cell_mixed_index <- percent_human>0.1&percent_human<0.9

sum(cell_human_index) # number of human cells
```

```
## [1] 8005
```

```
sum(cell_mouse_index) # number of mouse cells
```

```
## [1] 579
```

```
sum(cell_mixed_index) # number of mixed cells
```

```
## [1] 33
```

```
# take the subset of human genes and human cells
CBMC8K_RNA_umi_human_cell_gene <- CBMC8K_RNA_umi[index_human,cell_human_index]
CBMC8K_ADT_umi_human <- CBMC8K_ADT_umi[,cell_human_index]
```

We see that the 8617 cells we started with contains 8005 human cells, 579 mouse cells, and 33 mixed cells. There are a total of 20,400 human genes. For simplicity, we will keep only human cells and human genes for later analysis.

4. Establishing a scRNA-seq analyzing project using the Seurat package

We will establish a Seurat scRNA-seq object called *"CBMC8K_human"*, using the Seurat package. To do so, we define the original counts using our "gene X cell" matrix. Because most of the values are zeros, we use the *"as.sparse"* function to convert the matrix to a sparse format, where only the non-zero values will be stored. There can be several *"assays"* in a Seurat object to deal with different layers of data, for example RNA, protein, epigenetics, and so on. The default one is *"RNA"*, which can be reset, as shown later. There are also several *"slots"*. The original data will be stored in *"counts"*, normalized data will be in *"data"*, and scaled data will be in *"scale.data"*. The total numbers of counts and features (namely genes expressed) of each cell can be found in *"CBMC8K_human@meta.data"*.

```
# establish a Seurat object
CBMC8K_human <- CreateSeuratObject(counts  = as.sparse(CBMC8K_RNA_umi_human_cell_gene),
                                   project = "CBMC8K")

CBMC8K_human # brief overview of this Seurat object
```

```
## An object of class Seurat
## 20400 features across 8005 samples within 1 assay
## Active assay: RNA (20400 features)
```

```
# to visit the data matrix:
GetAssayData(object = CBMC8K_human, slot = "counts")[1:10,1:3]
```

```
## 10 x 3 sparse Matrix of class "dgCMatrix"
##                TACAGTGTCTCGGACG GTTTCTACATCATCCC ATCATGGAGTAGGCCA
## HUMAN-A1BG                   .                .                .
## HUMAN-A1BG-AS1               .                .                .
## HUMAN-A1CF                   .                .                .
## HUMAN-A2M                    .                .                .
## HUMAN-A2M-AS1                .                .                .
## HUMAN-A2ML1                  .                .                .
## HUMAN-A4GALT                 3                .                .
## HUMAN-A4GNT                  .                .                .
## HUMAN-AAAS                   2                1                .
## HUMAN-AACS                   .                .                .
```

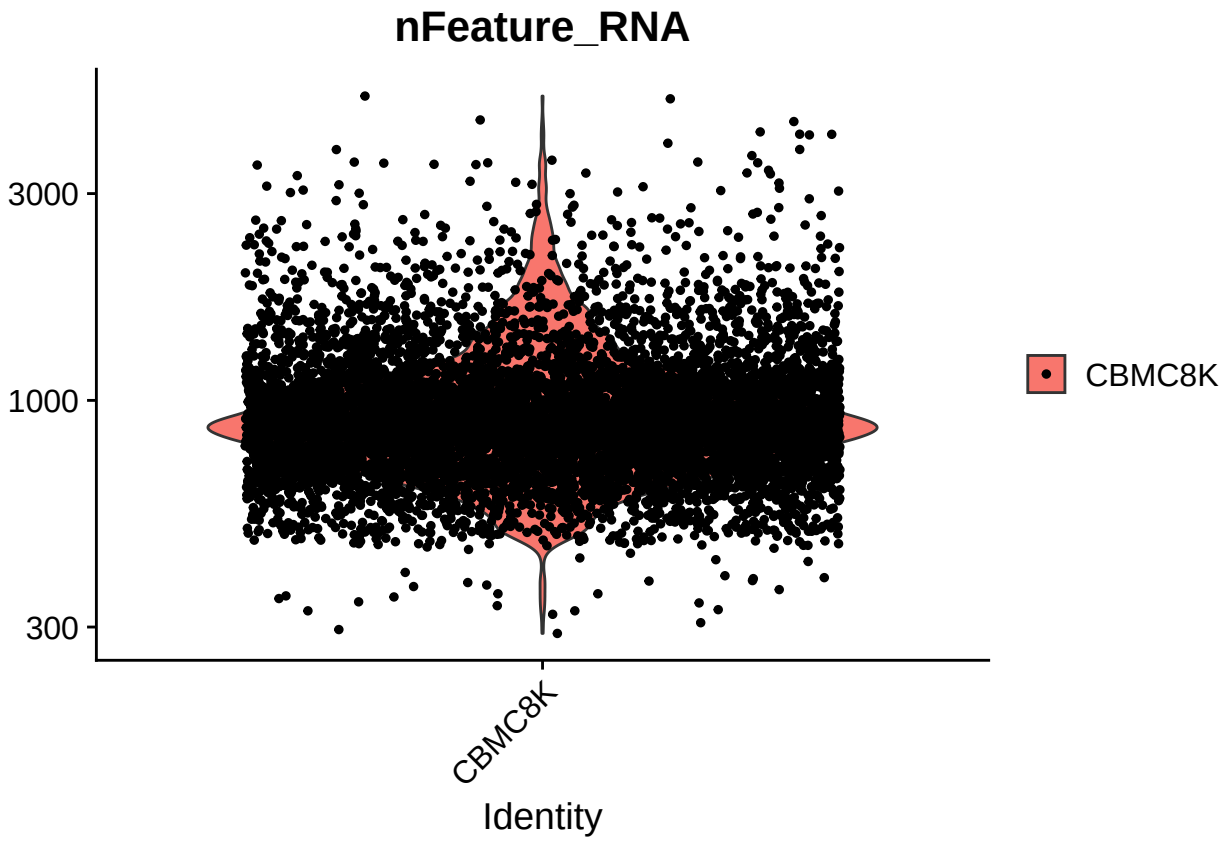
```
# or:
CBMC8K_human[["RNA"]]@counts[1:10,1:3]
```

```
## 10 x 3 sparse Matrix of class "dgCMatrix"
##                TACAGTGTCTCGGACG GTTTCTACATCATCCC ATCATGGAGTAGGCCA
## HUMAN-A1BG                   .                .                .
## HUMAN-A1BG-AS1               .                .                .
## HUMAN-A1CF                   .                .                .
## HUMAN-A2M                    .                .                .
## HUMAN-A2M-AS1                .                .                .
## HUMAN-A2ML1                  .                .                .
## HUMAN-A4GALT                 3                .                .
## HUMAN-A4GNT                  .                .                .
## HUMAN-AAAS                   2                1                .
## HUMAN-AACS                   .                .                .
```

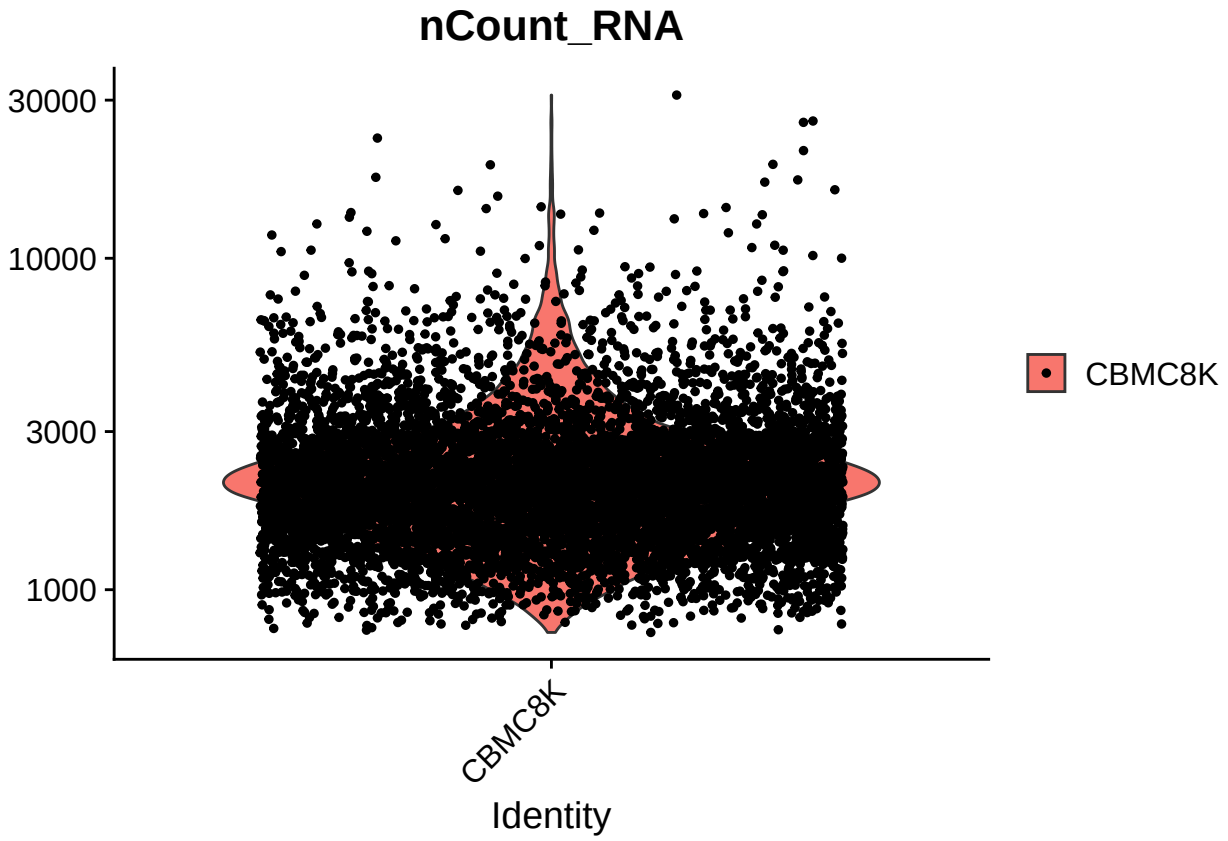
```
# use @meta.data to have a brief look at the numbers of counts and features of each cell
CBMC8K_human@meta.data[1:10,]
```

```
##                   orig.ident nCount_RNA nFeature_RNA
## TACAGTGTCTCGGACG     CBMC8K      31076         4962
## GTTTCTACATCATCCC     CBMC8K      23050         5035
## ATCATGGAGTAGGCCA     CBMC8K      16970         4162
## GTACGTATCCCATTTA     CBMC8K      25953         4096
## ATGTGTGGTCGCCATG     CBMC8K      25710         4111
## AACGTTGTCAGTTAGC     CBMC8K      19205         3394
## CAGAATCGTACATCCA     CBMC8K      21128         3791
## TCGGGACTCTGGCGTG     CBMC8K      19142         4435
## TGACTAGAGGATCGCA     CBMC8K      13150         3919
## GACCAATAGGGTATCG     CBMC8K      17238         4397
```

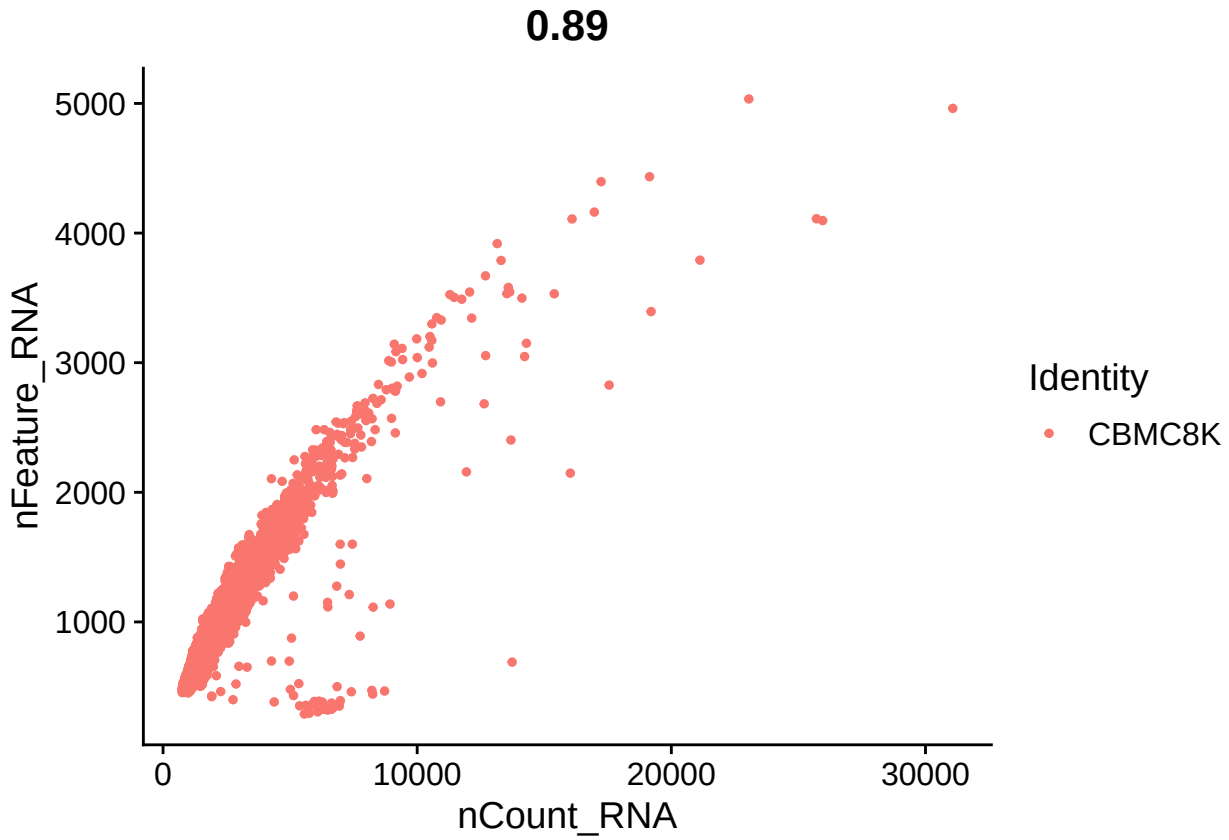
```
# Visualize the distribution of feature numbers and total counts as violin plots
VlnPlot(object = CBMC8K_human, features = "nFeature_RNA", log = TRUE) # number of genes
```




```
VlnPlot(object = CBMC8K_human, features = "nCount_RNA", log = TRUE) # number of counts
```




```
# Use FeatureScatter to visualize the relationship between nFeature and nCount
FeatureScatter(object = CBMC8K_human, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
```



Before we preform clustering based on gene expression, there are several preprocessing steps. The first one is quality control (QC), in which we need to filter out low-quality cells. People use different metrics in determining low-quality cells, such as number of genes the cell expresses, percentage of mitocondrial genes, the mumber of house keeping genes expressed, and so on. Here, I keep consistent with the filtering method used by the authors of this study: cells with less than 500 genes detected and cells with a total number of UMIs or genes that is more than 3 s.d. above or below the mean (at log10 level) will be removed.

```
# Quality control: filtering the single-cell RNA data #
# convert nFeature and nCount to log10 level and add to @meta.data
CBMC8K_human[["nFeature_log10"]] <- log10(CBMC8K_human@meta.data$nFeature_RNA)
CBMC8K_human[["nCount_log10"]] <- log10(CBMC8K_human@meta.data$nCount_RNA)
CBMC8K_human@meta.data[1:10,]
```

```
##                  orig.ident nCount_RNA nFeature_RNA nFeature_log10 nCount_log10
## TACAGTGTCTCGGACG     CBMC8K      31076         4962       3.695657     4.492425
## GTTTCTACATCATCCC     CBMC8K      23050         5035       3.701999     4.362671
## ATCATGGAGTAGGCCA     CBMC8K      16970         4162       3.619302     4.229682
## GTACGTATCCCATTTA     CBMC8K      25953         4096       3.612360     4.414188
## ATGTGTGGTCGCCATG     CBMC8K      25710         4111       3.613947     4.410102
## AACGTTGTCAGTTAGC     CBMC8K      19205         3394       3.530712     4.283414
## CAGAATCGTACATCCA     CBMC8K      21128         3791       3.578754     4.324858
## TCGGGACTCTGGCGTG     CBMC8K      19142         4435       3.646894     4.281987
## TGACTAGAGGATCGCA     CBMC8K      13150         3919       3.593175     4.118926
## GACCAATAGGGTATCG     CBMC8K      17238         4397       3.643156     4.236487
```

```r
summary(CBMC8K_human@meta.data$nFeature_log10)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.461   2.873   2.946   2.959   3.027   3.702
```

```r
summary(CBMC8K_human@meta.data$nCount_log10)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.870   3.198   3.312   3.320   3.415   4.492
```

```r
# calculate the upper and lower limit of nFeature
nFeature_upper <- mean(CBMC8K_human@meta.data$nFeature_log10)+
  3*sd(CBMC8K_human@meta.data$nFeature_log10)
nFeature_lower <- mean(CBMC8K_human@meta.data$nFeature_log10)-
  3*sd(CBMC8K_human@meta.data$nFeature_log10)

10^nFeature_upper # upper limit of nFeature calculated
```

```
## [1] 2368.07
```

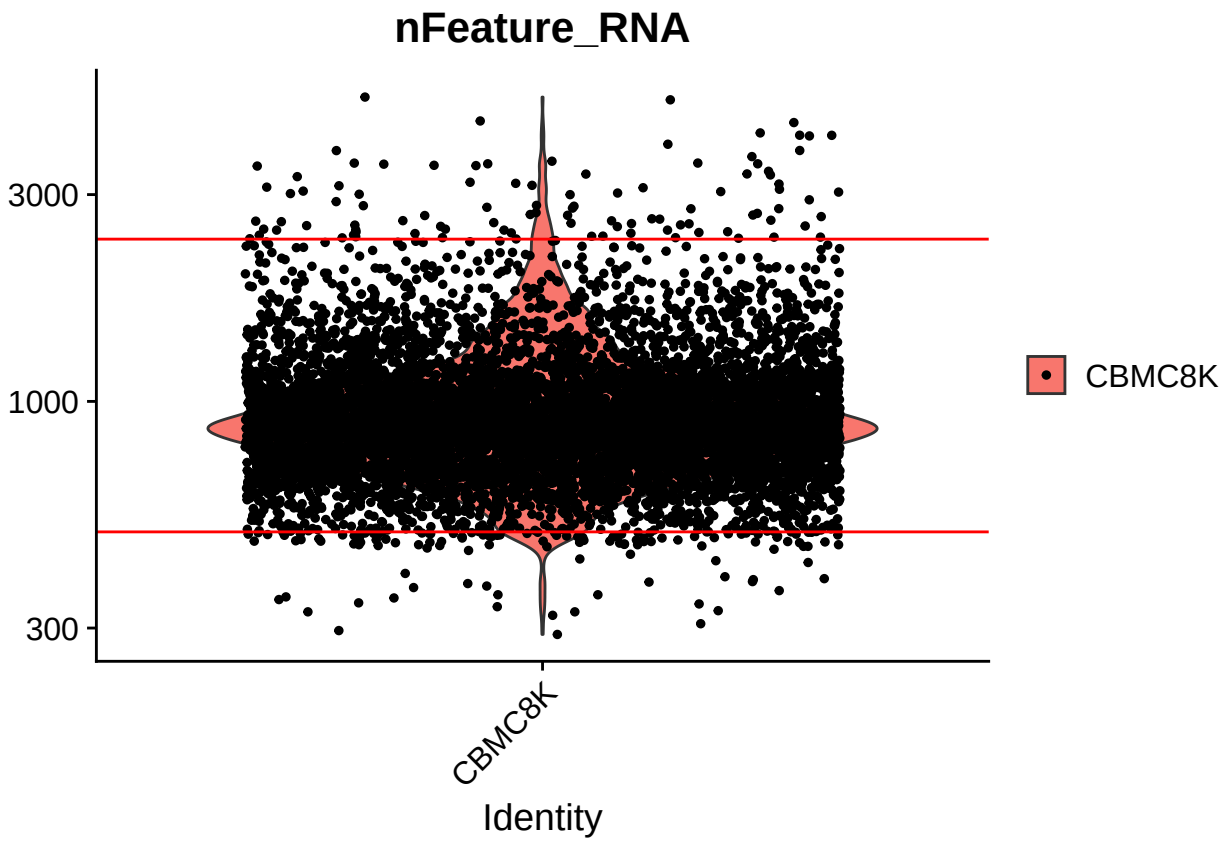
```r
10^nFeature_lower # lower limit of nFeature calculated
```

```
## [1] 349.5589
```

```r
min_nFeature <- 500 # min of nFeature as originally defined
max(min_nFeature,10^nFeature_lower) # final lower limit of nFeature
```

```
## [1] 500
```

```r
# Visualize the limits of QC metrics as a violin plot
p_feature <- VlnPlot(object = CBMC8K_human, features = "nFeature_RNA", log = TRUE, do.return=TRUE)
p_feature +
  geom_hline(yintercept = 10^nFeature_upper, color="red")+
  geom_hline(yintercept = max(min_nFeature,10^nFeature_lower), color="red")
```



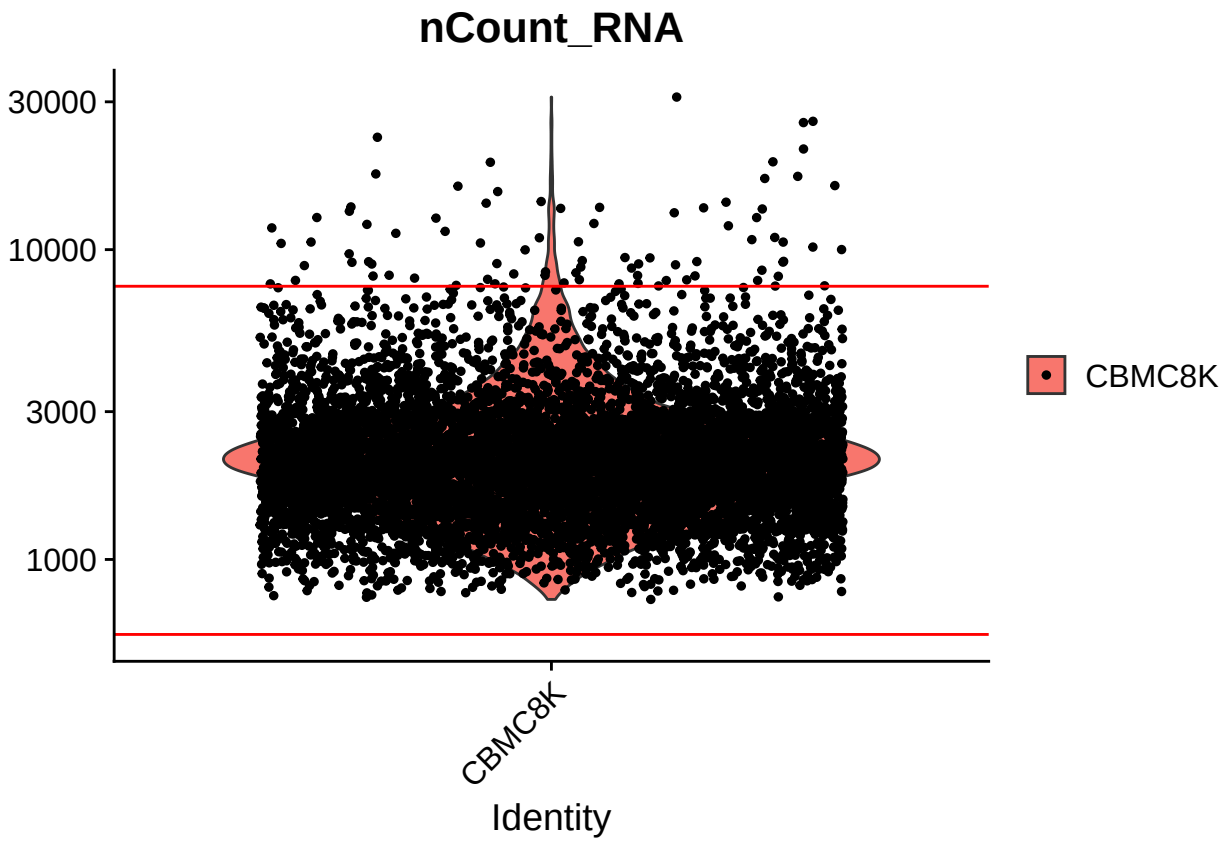

```
# do similar calculations for nCount
nCount_upper <- mean(CBMC8K_human@meta.data$nCount_log10) + 3*sd(CBMC8K_human@meta.data$nCount_log10)
nCount_lower <- mean(CBMC8K_human@meta.data$nCount_log10) - 3*sd(CBMC8K_human@meta.data$nCount_log10)
10^nCount_upper # upper limit of nCounts
```

```
## [1] 7617.333
```

```
10^nCount_lower # lower limit of nCounts
```

```
## [1] 572.6996
```

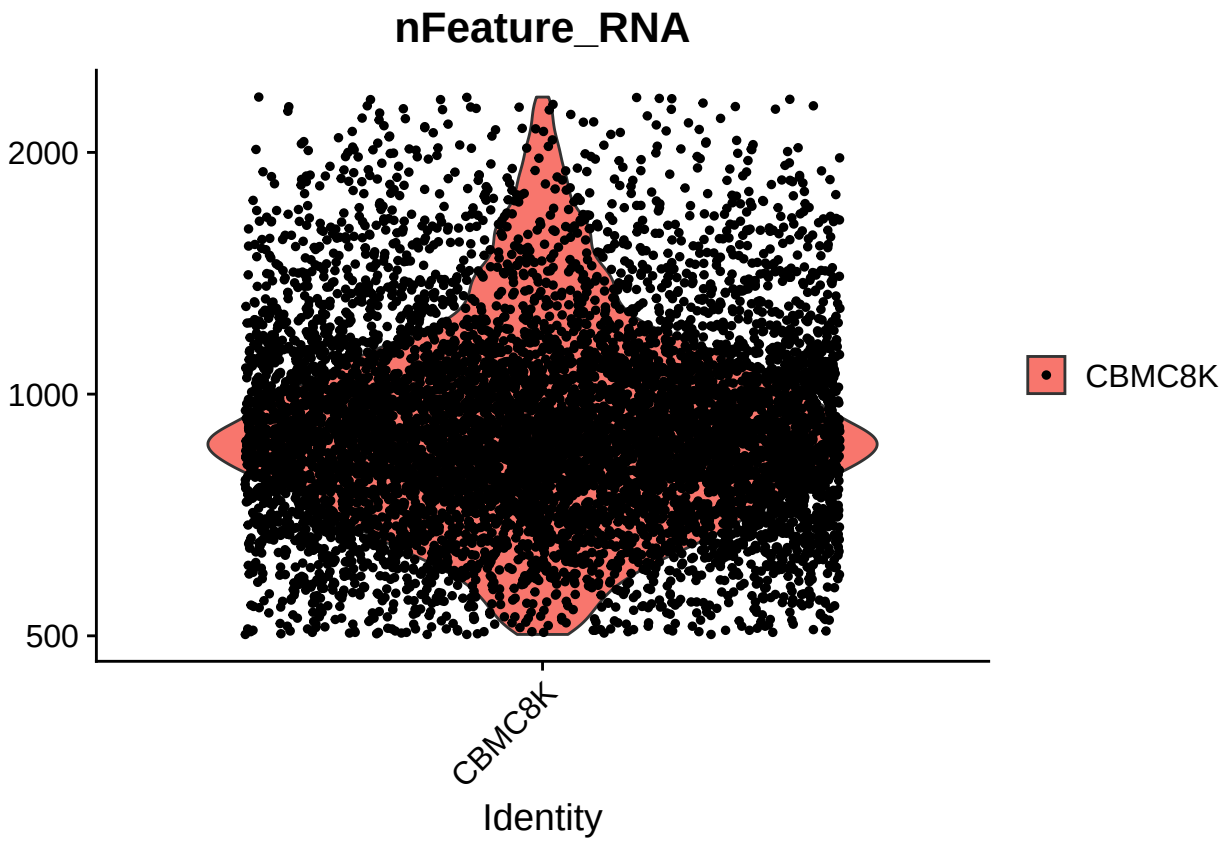
```
p_count <- VlnPlot(object = CBMC8K_human, features = "nCount_RNA", log = TRUE, do.return=TRUE)
p_count +
  geom_hline(yintercept = 10^nCount_upper, color="red")+
  geom_hline(yintercept = 10^nCount_lower, color="red")
```



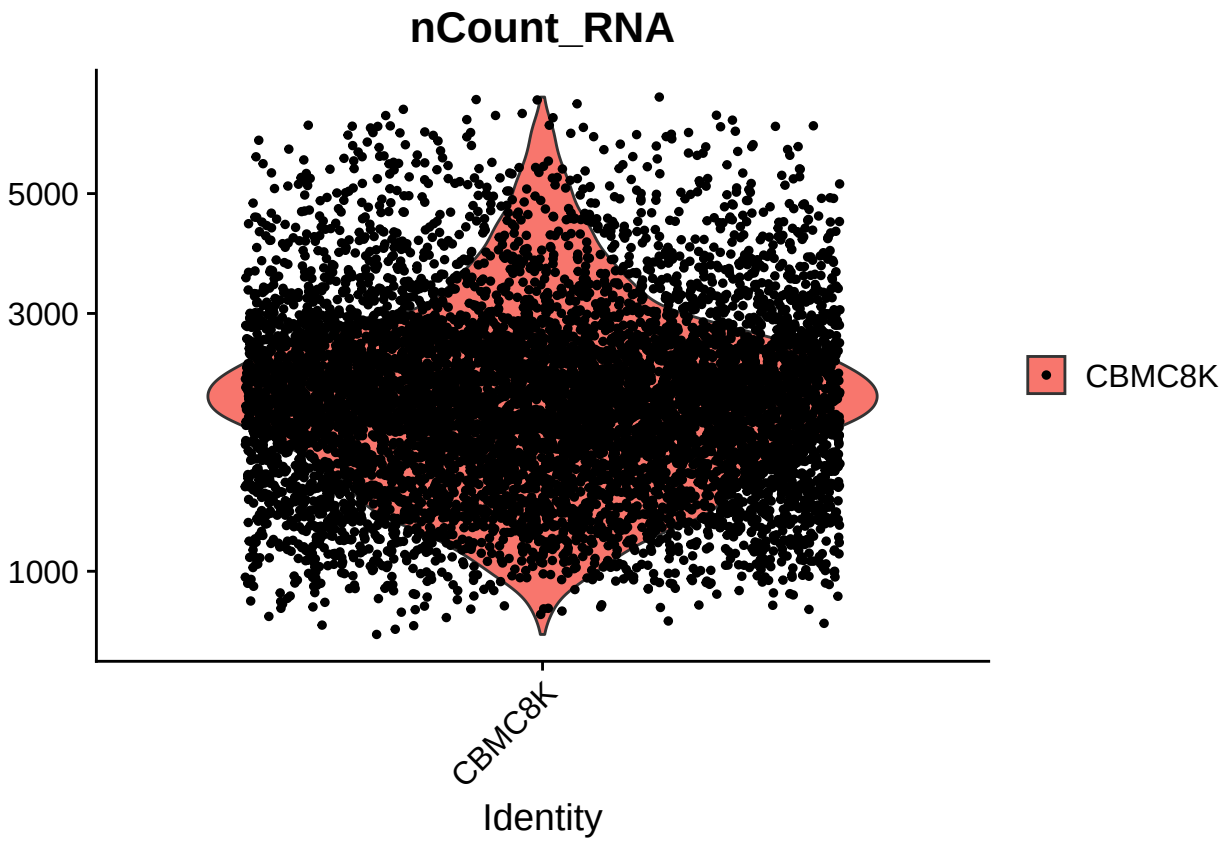

```
# According to this we will filter cells and only keep the ones with nFeature of 500-2368
# and nCount of 572-7617.
CBMC8K_human <- subset(CBMC8K_human, subset = nFeature_RNA > 500 &
                         nFeature_RNA < 10^nFeature_upper &
                         nCount_RNA > 10^nCount_lower &
                         nCount_RNA < 10^nCount_upper)
CBMC8K_human
```

```
## An object of class Seurat
## 20400 features across 7766 samples within 1 assay
## Active assay: RNA (20400 features)
```

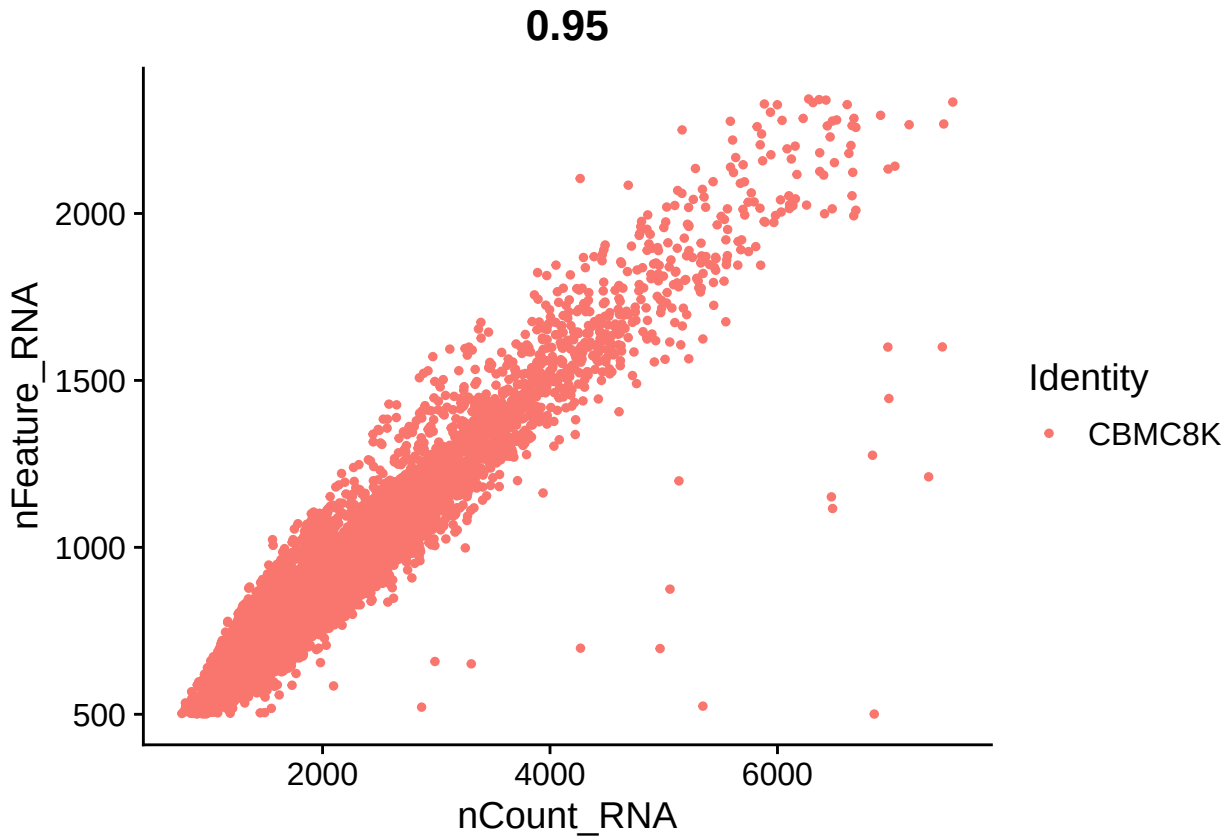
```
# visulize nFeature and nCount after filtering cells
VlnPlot(object = CBMC8K_human, features = "nFeature_RNA", log = TRUE)
```




```
VlnPlot(object = CBMC8K_human, features = "nCount_RNA", log = TRUE)
```




```
FeatureScatter(object = CBMC8K_human, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
```



After the quality control step, next is to normalize the data. By default, we employ a global-scaling normalization method called *"LogNormalize"* that normalizes the gene expression levels for each cell by the total expression. In another word, we suppose the total gene exression levels of every cell are the same, and the variation comes from sampling. Relative gene expression will be then multiplied by a scale factor (10,000 by default), and log-transformed. Normalized values are stored in *"[["RNA"]]@data"*.

```
# Now we will normalize the data.
# normalization.method = "LogNormalize" and scale.factor = 10000 are both default values
CBMC8K_human <- NormalizeData(CBMC8K_human)

# these 2 lines of codes both give preview of normalized data
CBMC8K_human[["RNA"]]@data[1:10, 1:3]
```

```
## 10 x 3 sparse Matrix of class "dgCMatrix"
##                CACTCCAGTTTCCACC TGGTTAGAGATGTTAG CCTAGCTGTGAGGGTT
## HUMAN-A1BG                   .                .                .
## HUMAN-A1BG-AS1               .                .                .
## HUMAN-A1CF                   .                .                .
## HUMAN-A2M                    .                .                .
## HUMAN-A2M-AS1                .                .                .
## HUMAN-A2ML1                  .                .                .
## HUMAN-A4GALT                 .                .                .
## HUMAN-A4GNT                  .                .                .
## HUMAN-AAAS                   .                .                .
## HUMAN-AACS                   .                .                .
```

```r
GetAssayData(object = CBMC8K_human)[1:10, 1:3]
```

```
## 10 x 3 sparse Matrix of class "dgCMatrix"
##                 CACTCCAGTTTCCACC TGGTTAGAGATGTTAG CCTAGCTGTGAGGGTT
## HUMAN-A1BG                     .                .                .
## HUMAN-A1BG-AS1                 .                .                .
## HUMAN-A1CF                     .                .                .
## HUMAN-A2M                      .                .                .
## HUMAN-A2M-AS1                  .                .                .
## HUMAN-A2ML1                    .                .                .
## HUMAN-A4GALT                   .                .                .
## HUMAN-A4GNT                    .                .                .
## HUMAN-AAAS                     .                .                .
## HUMAN-AACS                     .                .                .
```

5. Dimentional reduction and clustering methods

We will cluster the cells based on their molecular features. However, we have too many genes (more than 20,000 genes), making it a challenge to model. Because genes with higher variability usually provide more information in distinguishing cell types, we first need to identify those highly variable genes for downstream analysis. After this, we will scale the data so that each gene will have mean expression of 0 and variance of 1.

Dimentions are further reduced through linear or non-linear methods. Perheps the most popularly used dimention reduction method is the principal component analysis (PCA). PCA is a linear approach that identify the combinations of the basal features as principal component and maximize the variance of the data across these principal components. The distance in the reduced dimensional space still represents the similarity between single cells or cell groups. In scRNA-seq analysis, PCA is tipically used as a pre-processing step prior to non-linear dimentional reduction algorithms. Non-linear dimentional reduction method can better capture the local similarity instead of the global structure of the data, and thus better separate cell populations. The most commonly used one in scRNA-seq analysis is the t-distributed stochastic neighbour embedding (t-SNE).A common alternative to t-SNE is the uniform approximation and projection method (UMAP).

The standard approach to clustering scRNA-seq data is modularity optimization algorithm that finds community structure. Community detection rely on a graph representation of single-cell data, based on a k-nearest neighbour approach called KNN graph. Briefly, each cell is represented as a node in the graph, and connected to its K most similar cells, in terms of Euclidean distances on the PC-reduced expression space. THe communities of cells will be detected as groups of cells with more links between them than expected from the number of links the cells have in total.

Results

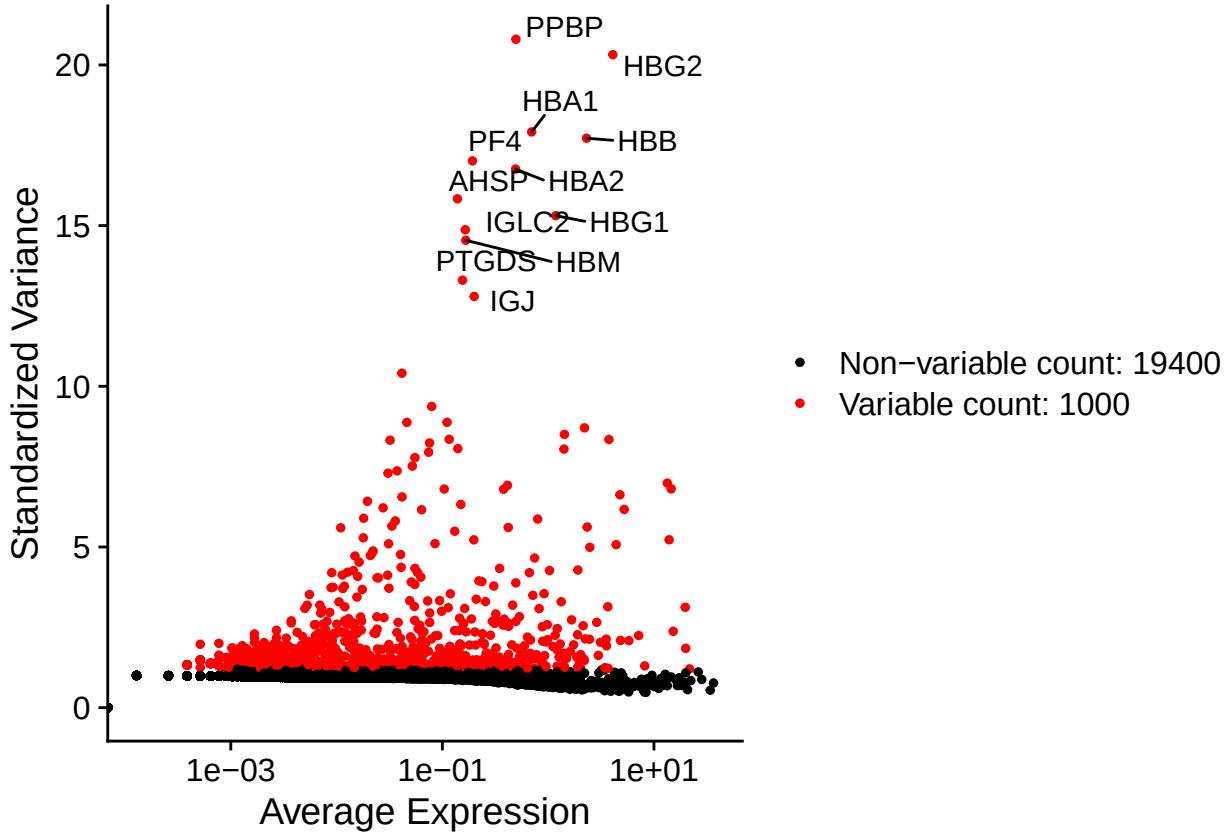
1. Identifying highly variable features

The first step of dimentional reduction is feature selection, in order to recognize and only keep those most informative features (genes here) for downstream analysis. *Highly variable genes (HVGs)* are commonly used, and can be easily found using the *"FindVariableFeatures"* function in the Seurat package. We set the number of features as 1000 here.

```
# Identification of highly variable features (feature selection)
# Use "nfeatures" to define the number of features. Here we use 1000.
CBMC8K_human <- FindVariableFeatures(CBMC8K_human,
                                     selection.method = "vst",
                                     nfeatures = 1000)

# We have access to the highly variable features via "VariableFeatures(CBMC8K_human)".
# For example, we can obtain the 12 most highly variable genes
top12 <- head(VariableFeatures(CBMC8K_human), 12)
top12_name <- separate(as.data.frame(top12),1,into = c("discard","name"), sep = "-")[,2]

# plot variable features with and without labels
plot_1 <- VariableFeaturePlot(CBMC8K_human)
LabelPoints(plot = plot_1, points = top12, labels = top12_name, repel = TRUE)
```



After we select HVGs, we will scale the data so that each gene has mean expression of 0 and variance of 1, to improve the comparisons between genes. By doing this, now all genes are weighted equally for downstream analysis. Scaled data are stored in the *"[[RNA]]@scale.data"* slot.

```
# scaling #
# Each gene is scaled to have mean expression of 0 and variance of 1.
CBMC8K_human <- ScaleData(CBMC8K_human, features = rownames(CBMC8K_human))

# preview of scaled data:
CBMC8K_human[["RNA"]]@scale.data[1:10, 1:3]
```

```
##                CACTCCAGTTTCCACC TGGTTAGAGATGTTAG CCTAGCTGTGAGGGTT
```

```
## HUMAN-A1BG          -0.02952120      -0.02952120      -0.02952120
## HUMAN-A1BG-AS1      -0.05006362      -0.05006362      -0.05006362
## HUMAN-A1CF          -0.01588888      -0.01588888      -0.01588888
## HUMAN-A2M           -0.04066210      -0.04066210      -0.04066210
## HUMAN-A2M-AS1       -0.14087171      -0.14087171      -0.14087171
## HUMAN-A2ML1         -0.01134753      -0.01134753      -0.01134753
## HUMAN-A4GALT        -0.03323043      -0.03323043      -0.03323043
## HUMAN-A4GNT         -0.02518250      -0.02518250      -0.02518250
## HUMAN-AAAS          -0.18975983      -0.18975983      -0.18975983
## HUMAN-AACS          -0.10596156      -0.10596156      -0.10596156
```

```
# or using this code:
GetAssayData(object = CBMC8K_human, slot = "scale.data")[1:10, 1:3]
```

```
##                CACTCCAGTTTCCACC TGGTTAGAGATGTTAG CCTAGCTGTGAGGGTT
## HUMAN-A1BG          -0.02952120      -0.02952120      -0.02952120
## HUMAN-A1BG-AS1      -0.05006362      -0.05006362      -0.05006362
## HUMAN-A1CF          -0.01588888      -0.01588888      -0.01588888
## HUMAN-A2M           -0.04066210      -0.04066210      -0.04066210
## HUMAN-A2M-AS1       -0.14087171      -0.14087171      -0.14087171
## HUMAN-A2ML1         -0.01134753      -0.01134753      -0.01134753
## HUMAN-A4GALT        -0.03323043      -0.03323043      -0.03323043
## HUMAN-A4GNT         -0.02518250      -0.02518250      -0.02518250
## HUMAN-AAAS          -0.18975983      -0.18975983      -0.18975983
## HUMAN-AACS          -0.10596156      -0.10596156      -0.10596156
```

2. Dimensional reduction

Now we perform linear dimensional reduction using PCA on the scaled data. Only the previously determined HVGs will be used as input. By default the first 50 PCs will be extracted.

```
# linear dimensional reduction using PCA #
CBMC8K_human <- RunPCA(CBMC8K_human, features = VariableFeatures(object = CBMC8K_human))

# by default RunPCA will extract the first 50 components (npcs = 50)
CBMC8K_human[["pca"]]
```

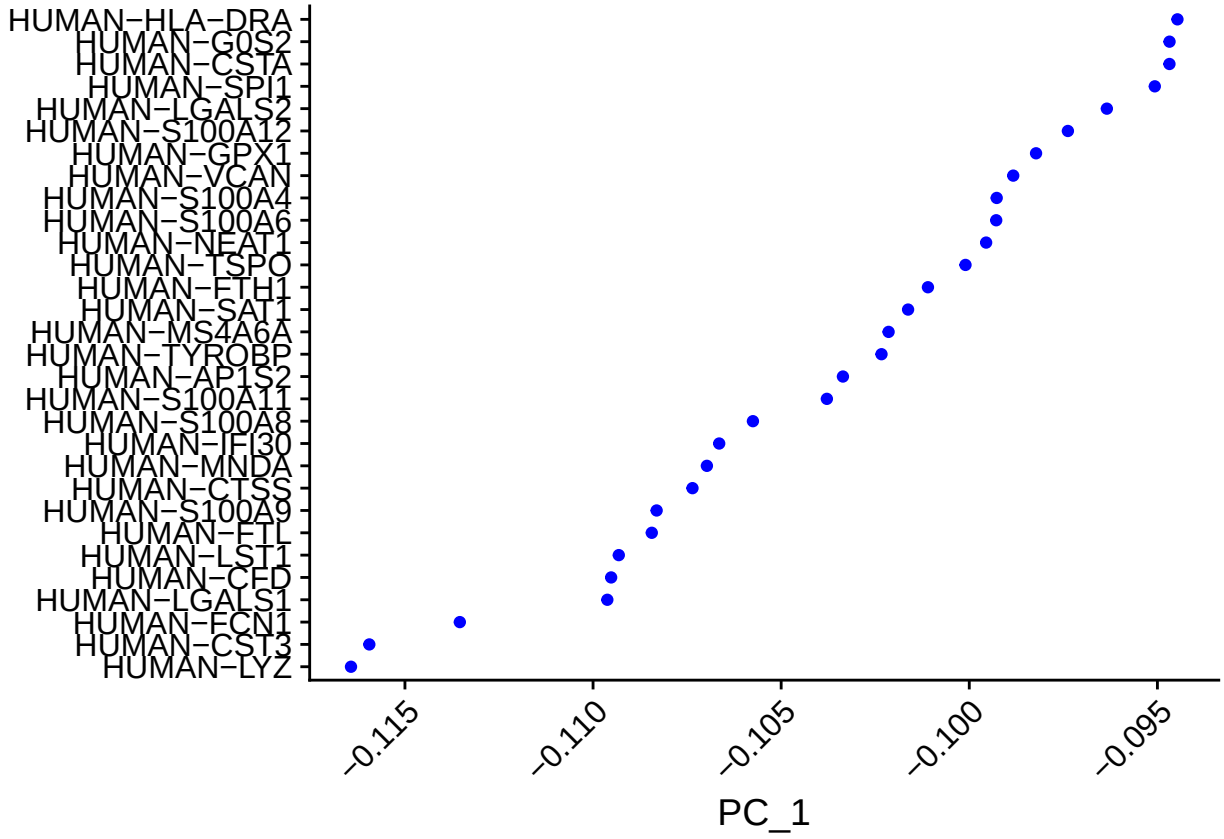
```
## A dimensional reduction object with key PC_
##  Number of dimensions: 50
##  Projected dimensional reduction calculated:  FALSE
##  Jackstraw run: FALSE
##  Computed using assay: RNA
```

```
# examine the first 5 components
print(CBMC8K_human[["pca"]], dims = 1:5, nfeatures = 5)
```

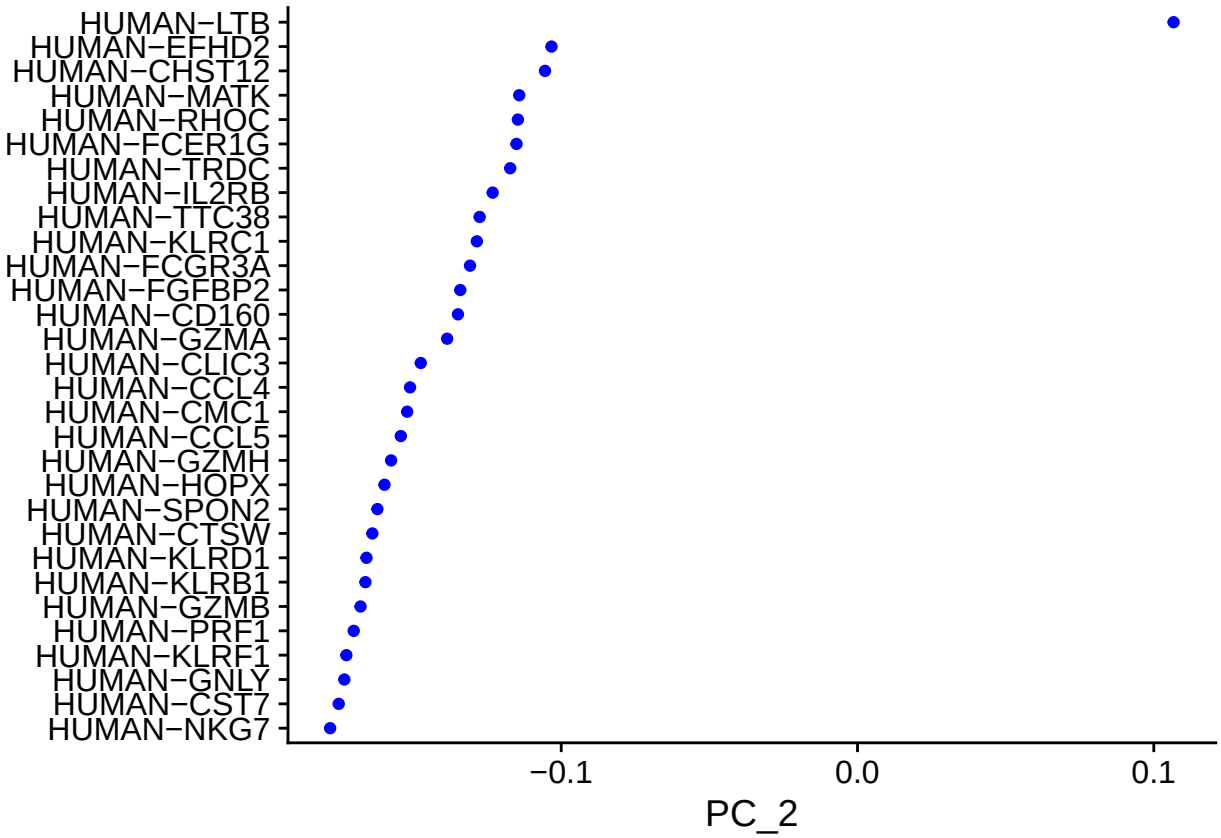
```
## PC_ 1
## Positive:  HUMAN-TRBC2, HUMAN-LTB, HUMAN-CD69, HUMAN-IL32, HUMAN-CD7
## Negative:  HUMAN-LYZ, HUMAN-CST3, HUMAN-FCN1, HUMAN-LGALS1, HUMAN-CFD
## PC_ 2
## Positive:  HUMAN-LTB, HUMAN-RPS4X, HUMAN-IL7R, HUMAN-TRAC, HUMAN-SOX4
```

```
## Negative:  HUMAN-NKG7, HUMAN-CST7, HUMAN-GNLY, HUMAN-KLRF1, HUMAN-PRF1
## PC_ 3
## Positive:  HUMAN-GNG11, HUMAN-IGHM, HUMAN-SDPR, HUMAN-TCL1A, HUMAN-PPP1R14A
## Negative:  HUMAN-JUNB, HUMAN-IL32, HUMAN-CD3D, HUMAN-ANXA1, HUMAN-IL7R
## PC_ 4
## Positive:  HUMAN-SDPR, HUMAN-PF4, HUMAN-TUBB1, HUMAN-PPBP, HUMAN-GP9
## Negative:  HUMAN-IGHM, HUMAN-TCL1A, HUMAN-IGHD, HUMAN-CD79A, HUMAN-MS4A1
## PC_ 5
## Positive:  HUMAN-C1QTNF4, HUMAN-SPINK2, HUMAN-AVP, HUMAN-EGFL7, HUMAN-CYTL1
## Negative:  HUMAN-PF4, HUMAN-PPBP, HUMAN-TUBB1, HUMAN-GP9, HUMAN-CMTM5
```

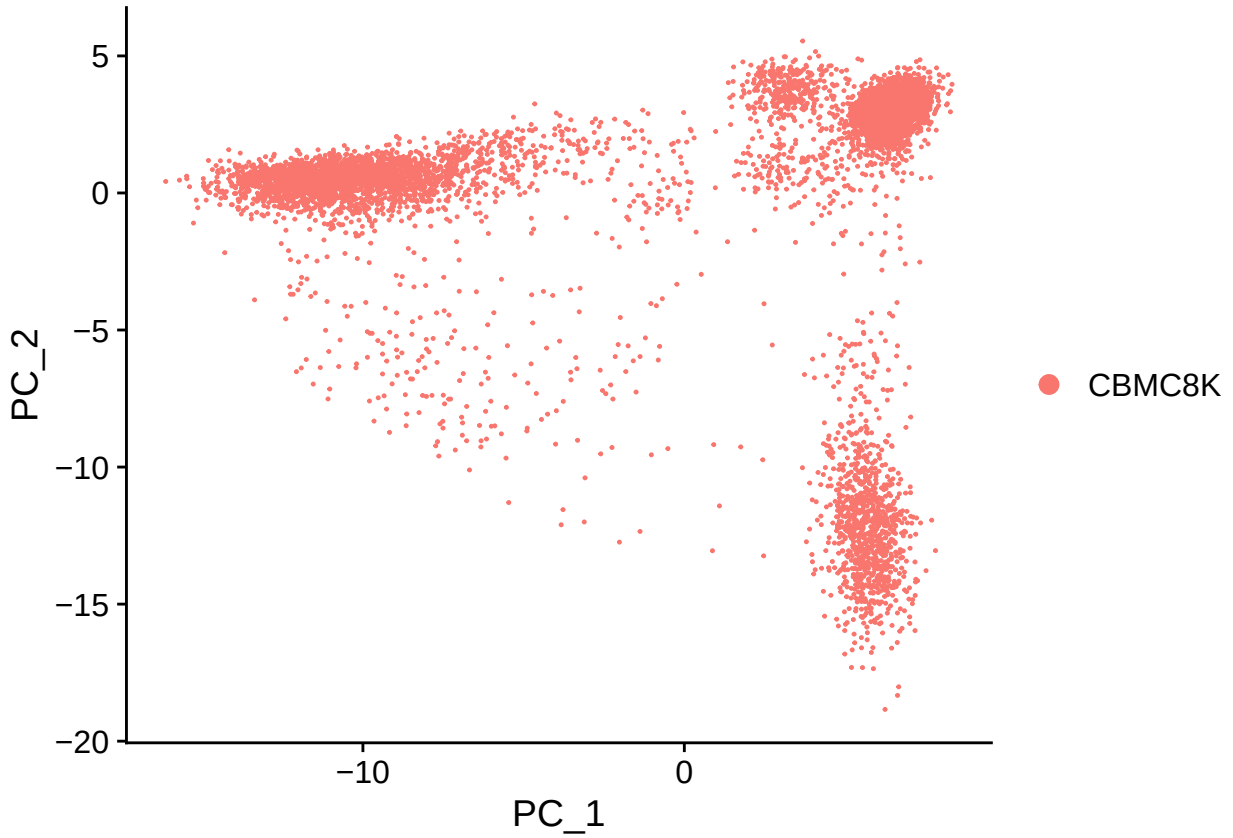
```r
# Visualize top genes associated with reduction components
plot_1 <- VizDimLoadings(CBMC8K_human, dims = 1, reduction = "pca",combine = TRUE)
plot_1 + theme(axis.text.x = element_text(angle = 45, hjust=1))
```




```r
VizDimLoadings(CBMC8K_human, dims = 2, reduction = "pca",combine = TRUE)
```



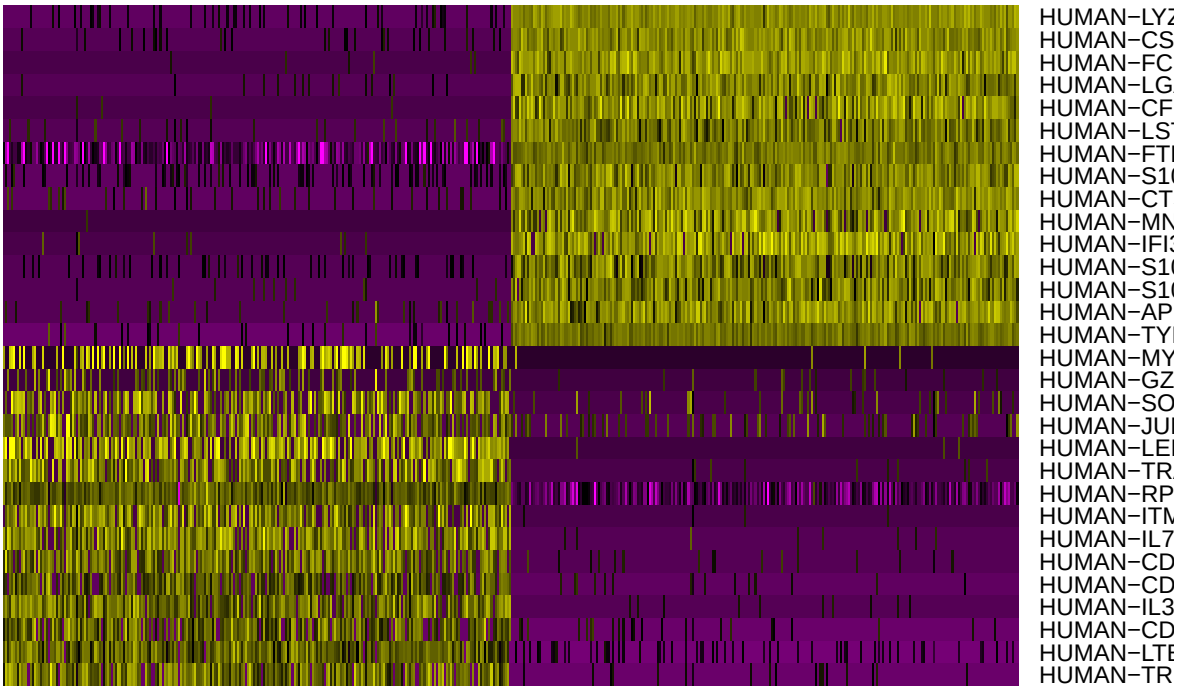

```
# Graphs the output of a dimensional reduction technique on a 2D scatter plot
DimPlot(CBMC8K_human, reduction = "pca")
```



There are multiple ways to decide which PCs to include for further downstream analyses. For example, *"DimHeatmap"* allows for easy exploration of the primary sources of heterogeneity in a dataset. Both cells and features are ordered according to their PCA scores. Secondly, the *"JackStraw plot"* can also be useful, which constructs a "null distribution" of feature scores by randomly permuting a subset of the data and re-runing PCA, and identifies "significant" PCs with strong enrichment of low p-value features. Alternatively, a more commonly used and intuitive method is to rank the PCs by the variance explained by each one in a so-called *"Elbow plot"*.

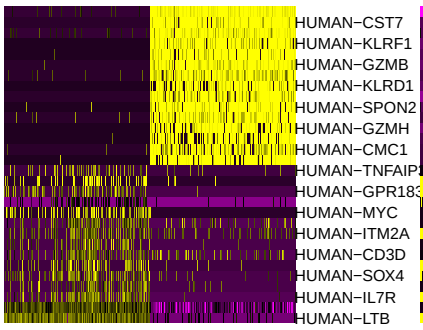
```
# Determine the 'dimensionality' of the dataset: DimHeatmap
# Setting cells to a number plots the 'extreme' cells on both ends of the spectrum,
# which dramatically speeds plotting for large datasets.
DimHeatmap(CBMC8K_human, dims = 1, cells = 500, balanced = TRUE)
```

PC_1

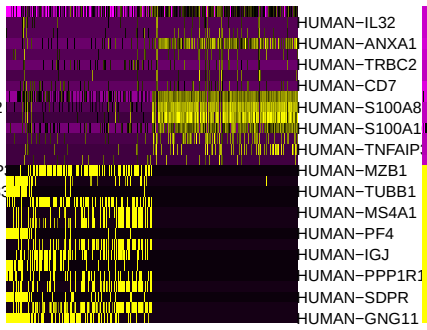



```
# can also visualize the 2nd to 5th PCs using the following code:
DimHeatmap(CBMC8K_human, dims = 2:5, cells = 500, balanced = TRUE)
```

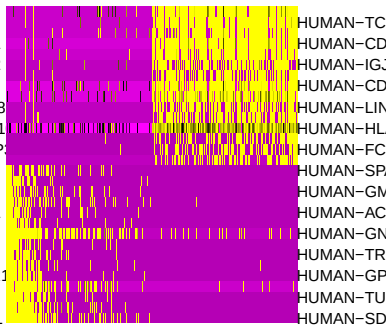
PC_2



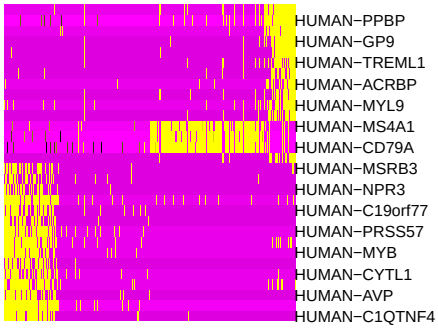
PC_3



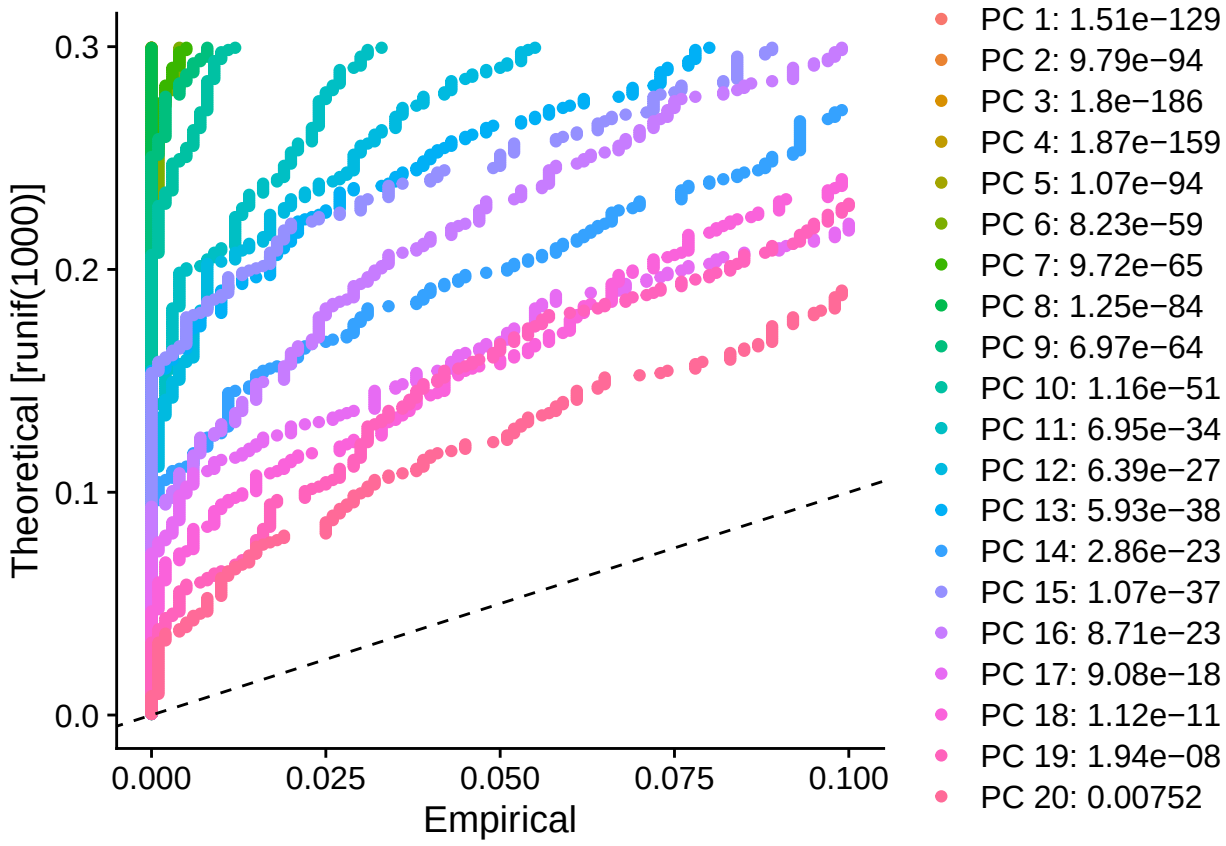
PC_4



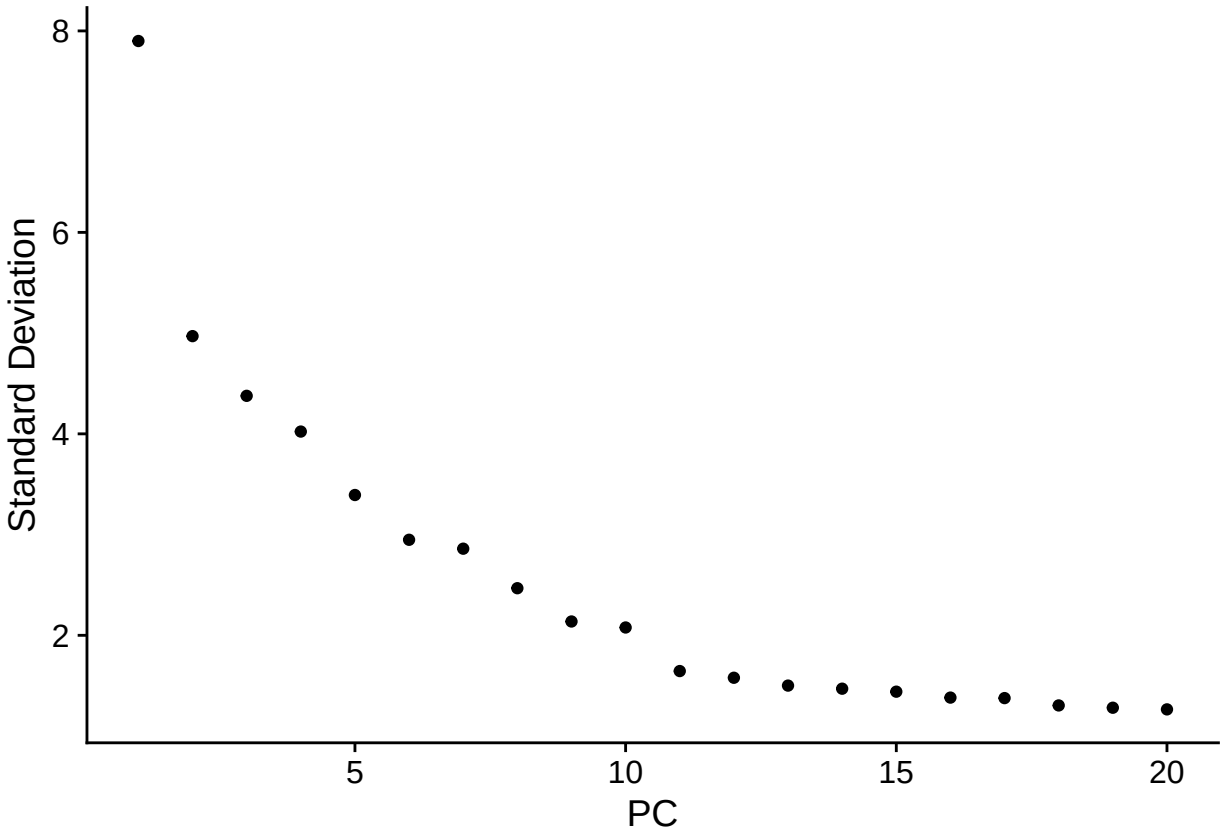
PC_5



```
# Determine the 'dimensionality' of the dataset: JackStraw plot
CBMC8K_human <- JackStraw(CBMC8K_human, num.replicate = 100)
CBMC8K_human <- ScoreJackStraw(CBMC8K_human, dims = 1:20)
JackStrawPlot(CBMC8K_human, dims = 1:20)
```




```
# Determine the 'dimensionality' of the dataset: Elbow plot
ElbowPlot(CBMC8K_human)
```



Here, from the "elbow plot", the turning point is around PC11-17-ish. To be safe, I chose the first 17 PCs for downstream processes, which also agrees with the *JackStraw plot*.

3. Clustering

To cluster the cells, the Seurat package provides a graph-based clustering approach as introduced in the *"Method"* section. Using the *"FindNeighbors"* function, the PCs we chose will be taken as input here, and the Euclidean distance in the PCA space of any two cells will be used to construct a KNN graph. The edge weights between cells will be refined based on the shared overlap in their local neighborhoods (Jaccard similarity). Next, the *"FindClusters"* function applies modularity optimization techniques to iteratively group cells together. We use the default "resolution" of 0.8 here, which could be further optimized to set the "granularity" of the clustering. The clusters of each cell can be found in *"Idents(CBMC8K_human)"*. From the "levels" of these identities, we get 15 different clusters (0-14).

```
# clustering #
# choose the first 17 PCs here
CBMC8K_human <- FindNeighbors(CBMC8K_human, dims = 1:17, k.param = 40)
CBMC8K_human <- FindClusters(CBMC8K_human, resolution = 0.8)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 7766
## Number of edges: 666599
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8318
```

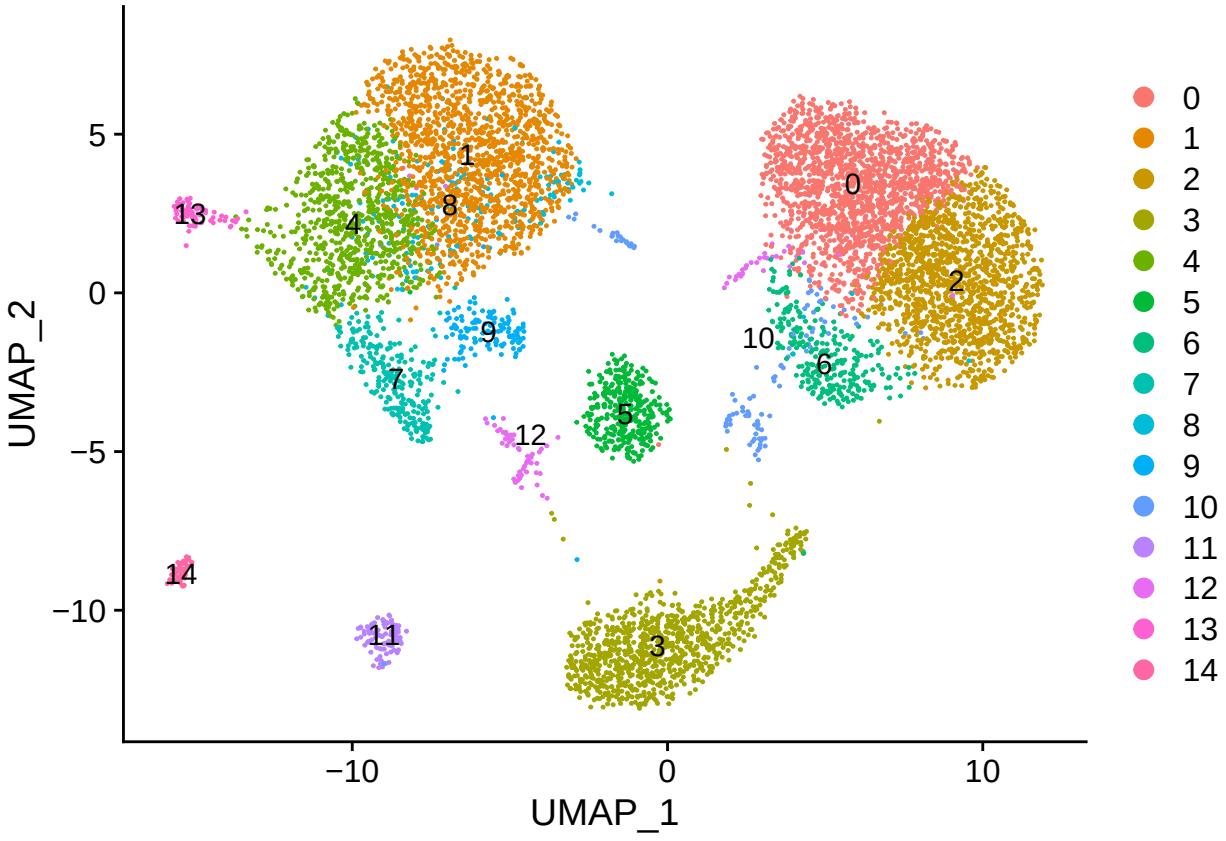
```
## Number of communities: 15
## Elapsed time: 1 seconds
```

```
head(Idents(CBMC8K_human), 4)
```

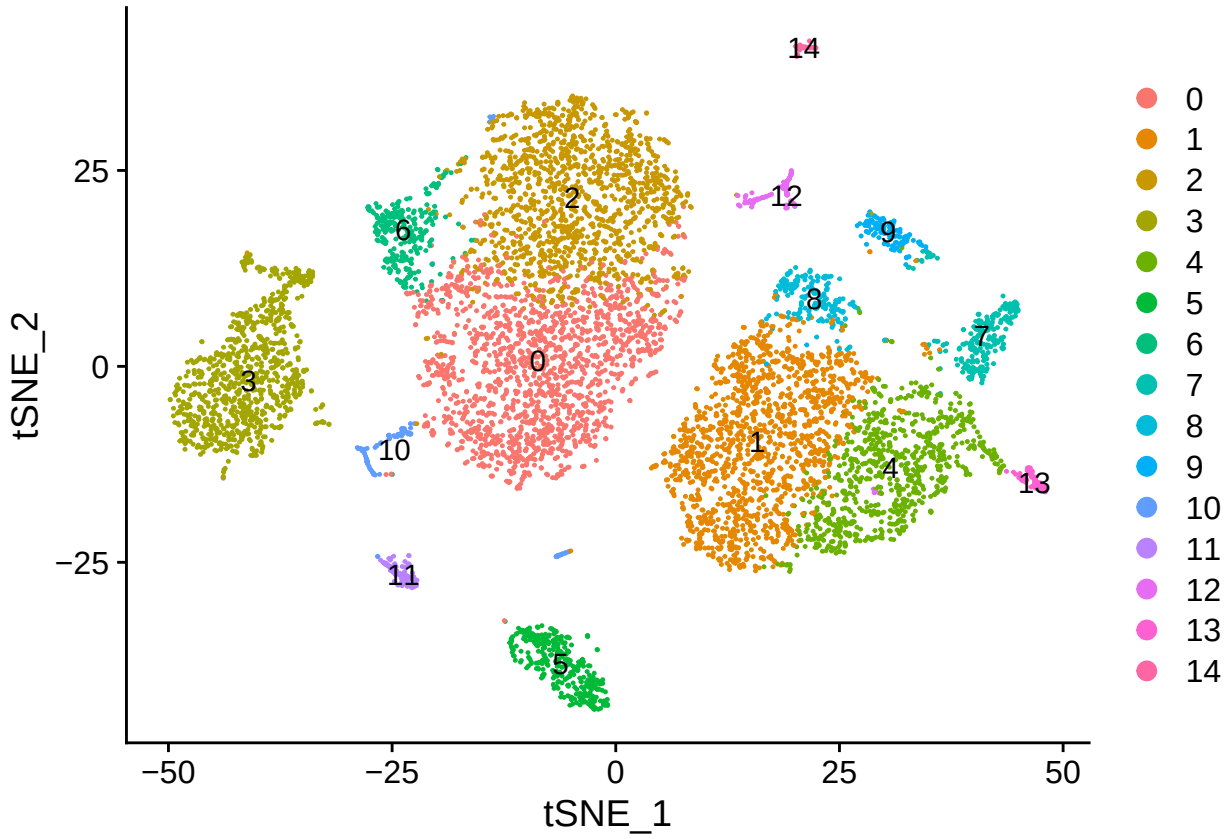
```
## CACTCCAGTTTCCACC TGGTTAGAGATGTTAG CCTAGCTGTGAGGGTT TGGCTGGTCAGCAACT
##                4                6                7                4
## Levels: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

Non-linear dimensional reduction techniques, such as UMAP and tSNE, can be applied to visualize and explore the data. The goal is to place similar cells together in low-dimensional space. The same PCs as input to the clustering analysis should be used here, and cells with the grapgh-based clusters determined above should also co-localize in these low-dimension plots.

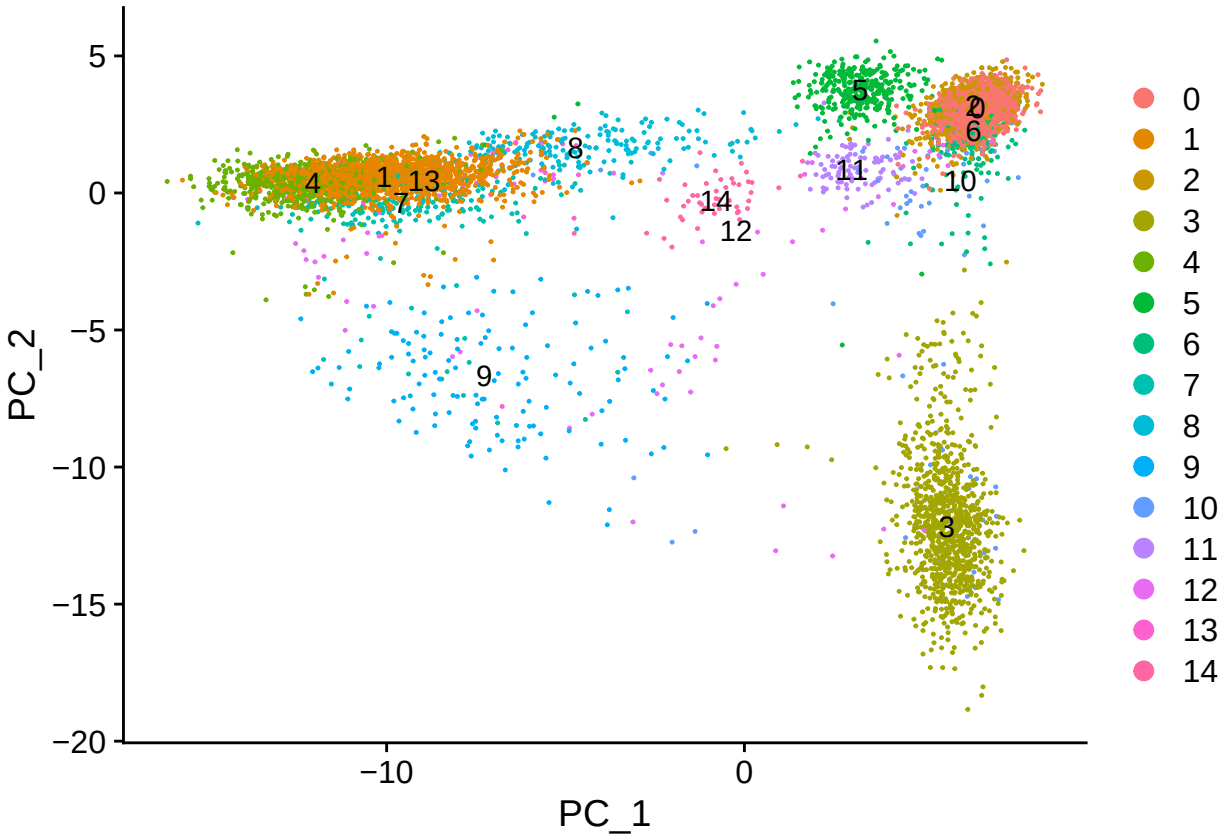
```
# non-linear dimensional reduction (UMAP/tSNE) #
# UMAP
CBMC8K_human <- RunUMAP(CBMC8K_human, dims = 1:17, min.dist = 0.75)
DimPlot(CBMC8K_human, reduction = "umap", label=T)
```




```
# TSNE
CBMC8K_human <- RunTSNE(seed.use = 1, object = CBMC8K_human,reduction = "pca",dims = 1:17)
DimPlot(CBMC8K_human, reduction = "tsne", label=T)
```




```
# can also visualize the clusters using PCA
DimPlot(CBMC8K_human, reduction = "pca", label=T)
```



As expected, PCA could not very well separate the groups of clusters 0, 2, 6 and clusters 1, 4, 7, 13. tSNE behaved much better in differentiating these clusters. UMAP is similar to tSNE with the only problem being not well separating clusters 1 and 8. Thus, we continue to use tSNE for visualization of the clustering results.

4. Identifying biomarkers

Now we have the clusters of the cells, the next step is to understand the biological significance of each cluster. First, we can identify the biomarkers of each cluster, which will hopefully give us some clues of what common features the cells within a certain group are sharing.

```
## Finding differentially expressed features (cluster biomarkers) ##

# find markers for every cluster compared to all remaining cells, report only the positive ones
CBMC8K_human.markers <- FindAllMarkers(CBMC8K_human, only.pos = TRUE, min.pct = 0.25,
                                       logfc.threshold = 0.25)

# It takes some time to find all the markers so we'd better store the markers in .Rdata
# object in case we need to re-load them later.
save(CBMC8K_human.markers, file = "CBMC8K_human.markers.Rdata")
```

```
# examine the top 5 markers of each cluster
CBMC8K_human.markers %>% dplyr::filter(cluster==0) %>% top_n(n = 5, wt = avg_logFC)
```

```
##           p_val avg_logFC pct.1 pct.2     p_val_adj cluster        gene
## 1  0.000000e+00 1.0303055 0.784 0.314  0.000000e+00       0 HUMAN-ITM2A
## 2  0.000000e+00 0.9704950 0.787 0.298  0.000000e+00       0  HUMAN-CD3E
```

```
## 3 4.844535e-307 0.9524008 0.819 0.310 9.882852e-303       0  HUMAN-IL7R
## 4 1.805043e-299 0.9554539 0.769 0.281 3.682288e-295       0  HUMAN-TRAC
## 5 6.559864e-243 0.9646646 0.699 0.302 1.338212e-238       0  HUMAN-SOX4
```

```
CBMC8K_human.markers %>% dplyr::filter(cluster==1) %>% top_n(n = 5, wt = avg_logFC)
```

```
##   p_val avg_logFC pct.1 pct.2 p_val_adj cluster          gene
## 1     0  2.704089 0.999 0.430         0       1  HUMAN-S100A8
## 2     0  2.511194 0.944 0.188         0       1 HUMAN-S100A12
## 3     0  2.493152 0.999 0.508         0       1  HUMAN-S100A9
## 4     0  1.916507 0.891 0.177         0       1    HUMAN-VCAN
## 5     0  1.913685 0.620 0.096         0       1     HUMAN-IL8
```

```
CBMC8K_human.markers %>% dplyr::filter(cluster==2) %>% top_n(n = 5, wt = avg_logFC)
```

```
##           p_val avg_logFC pct.1 pct.2     p_val_adj cluster        gene
## 1  0.000000e+00 1.0475167 0.896 0.313  0.000000e+00       2  HUMAN-IL7R
## 2 5.921691e-241 0.8487910 0.784 0.298 1.208025e-236       2  HUMAN-TRAC
## 3 9.627991e-218 0.9860724 0.372 0.070 1.964110e-213       2 HUMAN-USP10
## 4 6.244864e-178 0.8575570 0.665 0.301 1.273952e-173       2  HUMAN-TOB1
## 5 7.978149e-139 0.8378842 0.395 0.122 1.627542e-134       2  HUMAN-RGS1
```

```
CBMC8K_human.markers %>% dplyr::filter(cluster==3) %>% top_n(n = 5, wt = avg_logFC)
```

```
##   p_val avg_logFC pct.1 pct.2 p_val_adj cluster        gene
## 1     0  3.740045 0.969 0.213         0       3  HUMAN-GNLY
## 2     0  3.541506 0.999 0.195         0       3  HUMAN-NKG7
## 3     0  3.036732 0.760 0.069         0       3  HUMAN-CCL5
## 4     0  2.928701 0.971 0.109         0       3 HUMAN-KLRB1
## 5     0  2.792307 0.915 0.047         0       3  HUMAN-CST7
```

```
CBMC8K_human.markers %>% dplyr::filter(cluster==4) %>% top_n(n = 5, wt = avg_logFC)
```

```
##   p_val avg_logFC pct.1 pct.2 p_val_adj cluster           gene
## 1     0  1.572443 0.989 0.349         0       4 HUMAN-HLA-DRB1
## 2     0  1.424211 1.000 0.418         0       4     HUMAN-CST3
## 3     0  1.412079 0.933 0.226         0       4   HUMAN-LGALS2
## 4     0  1.372654 0.998 0.419         0       4  HUMAN-HLA-DRA
## 5     0  1.366984 0.892 0.222         0       4 HUMAN-HLA-DRB5
```

```
CBMC8K_human.markers %>% dplyr::filter(cluster==5) %>% top_n(n = 5, wt = avg_logFC)
```

```
##   p_val avg_logFC pct.1 pct.2 p_val_adj cluster        gene
## 1     0  3.504896 0.976 0.029         0       5  HUMAN-IGHM
## 2     0  2.933845 0.903 0.016         0       5 HUMAN-TCL1A
## 3     0  2.900368 0.962 0.036         0       5 HUMAN-CD79A
## 4     0  2.782416 0.285 0.004         0       5 HUMAN-IGLC2
## 5     0  2.613830 0.944 0.080         0       5 HUMAN-CD79B
```

```
CBMC8K_human.markers %>% dplyr::filter(cluster==6) %>% top_n(n = 5, wt = avg_logFC)
```

```
##           p_val avg_logFC pct.1 pct.2     p_val_adj cluster        gene
## 1  0.000000e+00 1.8469560 0.777 0.021  0.000000e+00       6  HUMAN-CD8B
## 2  0.000000e+00 1.2432640 0.543 0.023  0.000000e+00       6  HUMAN-CD8A
## 3 6.905406e-255 1.7555041 0.602 0.056 1.408703e-250       6 HUMAN-S100B
## 4  4.029897e-74 0.7816520 0.368 0.066  8.220990e-70       6 HUMAN-NELL2
## 5  5.939682e-71 0.8421533 0.944 0.450  1.211695e-66       6  HUMAN-CD3D
```

```
CBMC8K_human.markers %>% dplyr::filter(cluster==7) %>% top_n(n = 5, wt = avg_logFC)
```

```
##           p_val avg_logFC pct.1 pct.2     p_val_adj cluster               gene
## 1 2.265791e-262  1.854643 0.879 0.115 4.622214e-258       7        HUMAN-FCGR3A
## 2 1.778343e-261  1.652300 0.795 0.091 3.627821e-257       7 HUMAN-RP11-290F20.3
## 3 4.849643e-241  1.710224 0.837 0.115 9.893272e-237       7         HUMAN-MS4A7
## 4 3.562225e-164  1.618254 0.967 0.271 7.266939e-160       7      HUMAN-HLA-DPA1
## 5 2.192980e-157  1.734837 1.000 0.455 4.473679e-153       7          HUMAN-LST1
```

```
CBMC8K_human.markers %>% dplyr::filter(cluster==8) %>% top_n(n = 5, wt = avg_logFC)
```

```
##          p_val avg_logFC pct.1 pct.2    p_val_adj cluster         gene
## 1 2.151932e-33 0.4620845 0.760 0.287 4.389942e-29       8 HUMAN-LGALS2
## 2 4.114142e-31 0.4536181 0.865 0.359 8.392850e-27       8    HUMAN-CFD
## 3 5.053454e-30 0.4292994 1.000 0.976 1.030905e-25       8   HUMAN-FTH1
## 4 5.876249e-30 0.4476236 0.729 0.300 1.198755e-25       8   HUMAN-GOS2
## 5 5.255565e-20 0.4569683 0.818 0.453 1.072135e-15       8   HUMAN-MCL1
```

```
CBMC8K_human.markers %>% dplyr::filter(cluster==9) %>% top_n(n = 5, wt = avg_logFC)
```

```
##          p_val avg_logFC pct.1 pct.2    p_val_adj cluster        gene
## 1 8.895793e-96 1.2286263 0.866 0.161 1.814742e-91       9  HUMAN-CMC1
## 2 3.703728e-95 1.0600109 0.984 0.199 7.555604e-91       9 HUMAN-KLRB1
## 3 8.522566e-78 0.9490624 0.787 0.145 1.738604e-73       9  HUMAN-HOPX
## 4 3.611683e-71 1.0126527 0.976 0.292 7.367833e-67       9  HUMAN-GNLY
## 5 1.412665e-70 0.9663773 0.976 0.279 2.881837e-66       9  HUMAN-NKG7
```

```
CBMC8K_human.markers %>% dplyr::filter(cluster==10) %>% top_n(n = 5, wt = avg_logFC)
```

```
##          p_val avg_logFC pct.1 pct.2    p_val_adj cluster       gene
## 1 1.347982e-20  3.521625 0.377 0.110 2.749882e-16      10 HUMAN-HBA2
## 2 6.122462e-09  4.343693 0.298 0.131 1.248982e-04      10 HUMAN-HBG1
## 3 1.185564e-08  3.666221 0.342 0.162 2.418551e-04      10 HUMAN-HBA1
## 4 6.524799e-05  4.408921 0.544 0.372 1.000000e+00      10  HUMAN-HBB
## 5 8.132739e-05  5.113996 0.465 0.309 1.000000e+00      10 HUMAN-HBG2
```

```
CBMC8K_human.markers %>% dplyr::filter(cluster==11) %>% top_n(n = 5, wt = avg_logFC)
```

```
##   p_val avg_logFC pct.1 pct.2 p_val_adj cluster          gene
## 1     0  2.475669 0.958 0.020         0      11  HUMAN-SPINK2
```

```
## 2     0  2.065560 0.800 0.002         0      11  HUMAN-C1QTNF4
## 3     0  2.032769 0.853 0.012         0      11 HUMAN-KIAA0125
## 4     0  1.973337 0.642 0.001         0      11      HUMAN-AVP
## 5     0  1.849483 0.842 0.016         0      11   HUMAN-PRSS57
```

```
CBMC8K_human.markers %>% dplyr::filter(cluster==12) %>% top_n(n = 5, wt = avg_logFC)
```

```
##           p_val avg_logFC pct.1 pct.2     p_val_adj cluster           gene
## 1  0.000000e+00  4.124445 0.831 0.013  0.000000e+00      12       HUMAN-PF4
## 2  0.000000e+00  3.050684 0.775 0.013  0.000000e+00      12      HUMAN-SDPR
## 3 1.799999e-256  3.156301 0.775 0.037 3.671997e-252      12     HUMAN-GNG11
## 4 7.518353e-227  4.651385 0.865 0.057 1.533744e-222      12      HUMAN-PPBP
## 5 8.342667e-115  2.941246 0.663 0.065 1.701904e-110      12 HUMAN-HIST1H2AC
```

```
CBMC8K_human.markers %>% dplyr::filter(cluster==13) %>% top_n(n = 5, wt = avg_logFC)
```

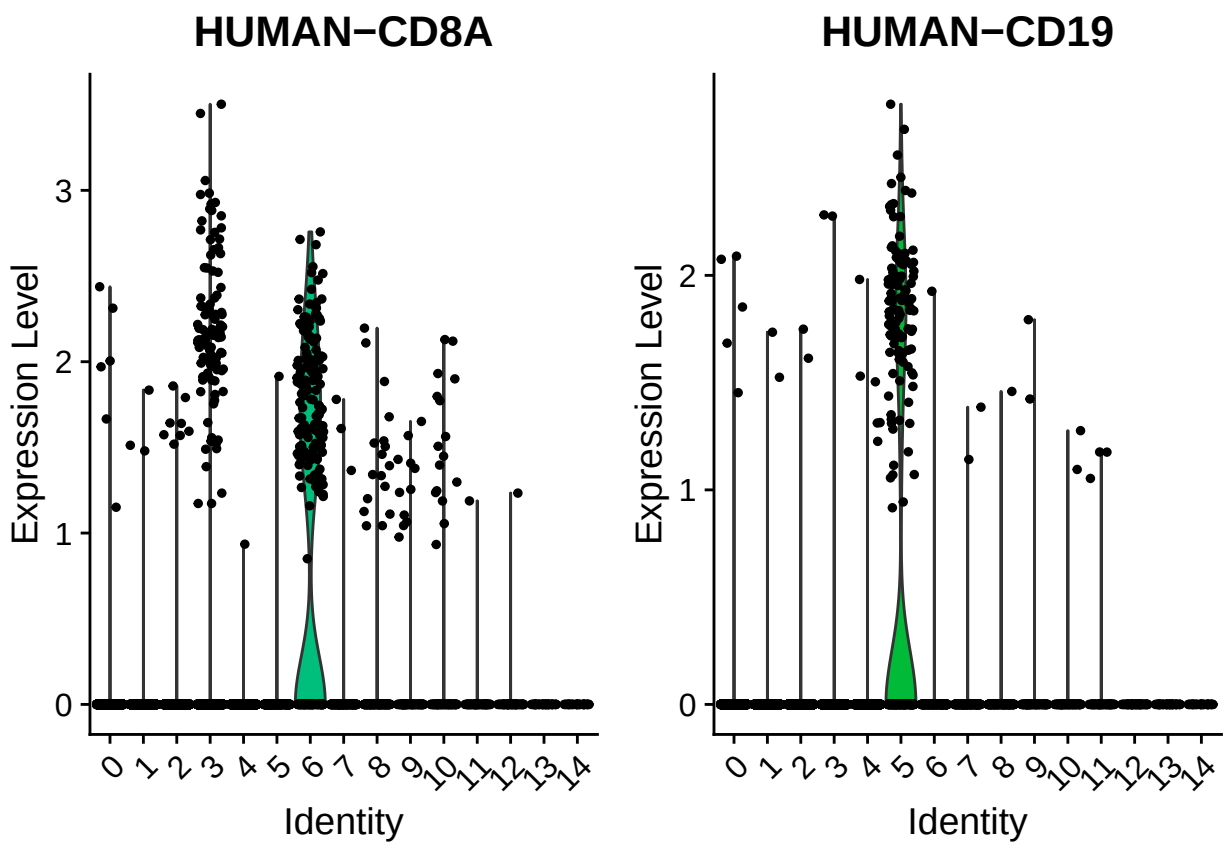
```
##           p_val avg_logFC pct.1 pct.2     p_val_adj cluster           gene
## 1  0.000000e+00  2.770860 0.882 0.009  0.000000e+00      13   HUMAN-FCER1A
## 2 3.533965e-226  2.591140 0.868 0.043 7.209289e-222      13  HUMAN-CLEC10A
## 3 3.066900e-122  2.793846 0.985 0.120 6.256477e-118      13 HUMAN-HLA-DQA1
## 4  1.253527e-71  2.754372 1.000 0.265  2.557194e-67      13 HUMAN-HLA-DPB1
## 5  9.545537e-68  2.777218 1.000 0.286  1.947289e-63      13 HUMAN-HLA-DPA1
```

```
CBMC8K_human.markers %>% dplyr::filter(cluster==14) %>% top_n(n = 5, wt = avg_logFC)
```

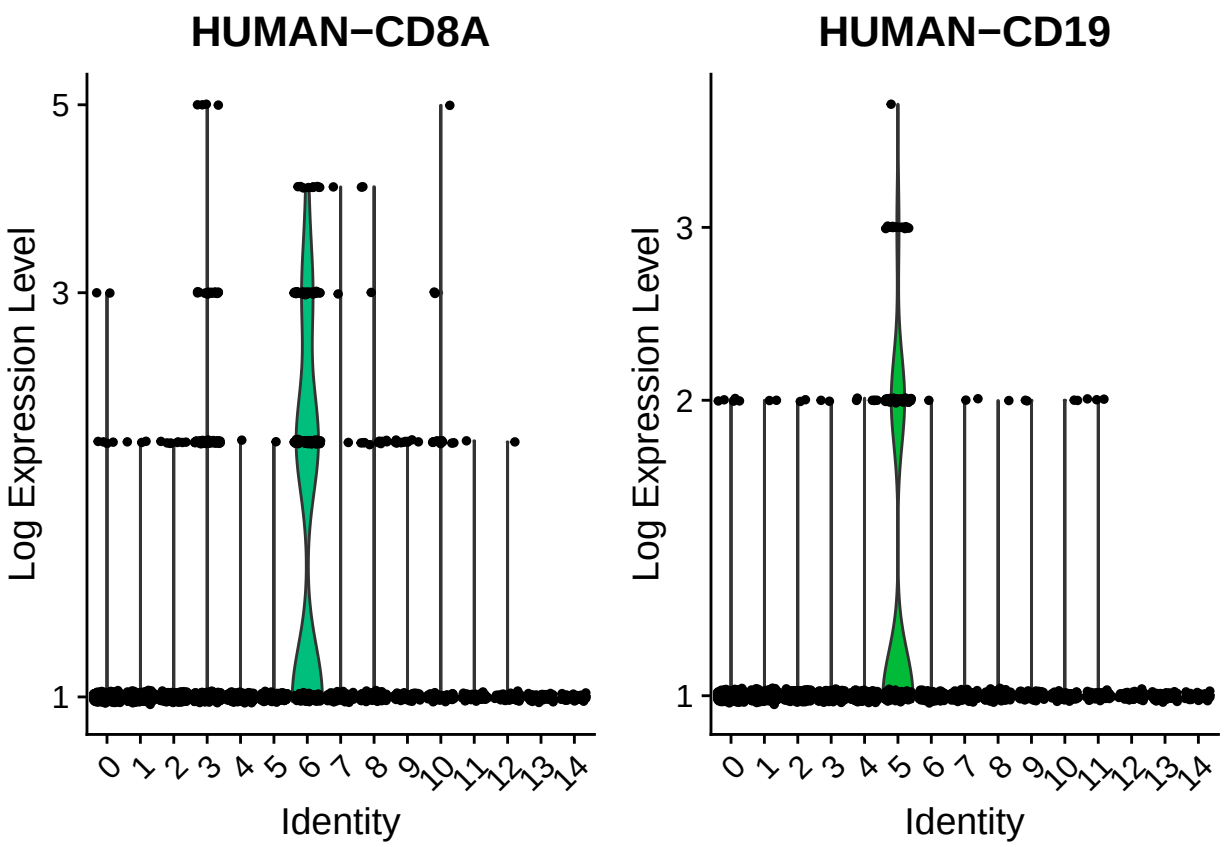
```
##           p_val avg_logFC pct.1 pct.2     p_val_adj cluster        gene
## 1 5.564889e-253  3.944727 1.000 0.038 1.135237e-248      14   HUMAN-IGJ
## 2 4.788731e-195  2.830406 0.980 0.049 9.769012e-191      14  HUMAN-PLD4
## 3 1.740018e-131  2.784151 0.980 0.079 3.549637e-127      14 HUMAN-ITM2C
## 4 4.419313e-102  2.716766 0.980 0.106  9.015398e-98      14  HUMAN-IRF7
## 5  1.205423e-41  3.152282 0.347 0.027  2.459062e-37      14 HUMAN-PTGDS
```

Blood cells can be classified into different subgroups, characterized by combinations of common biomarkers. For example, CD8 T-cells have the feature of CD3+ CD4- CD8+, and B-cells are CD3- CD19+. We can visualize expression of marker genes in different clusters.

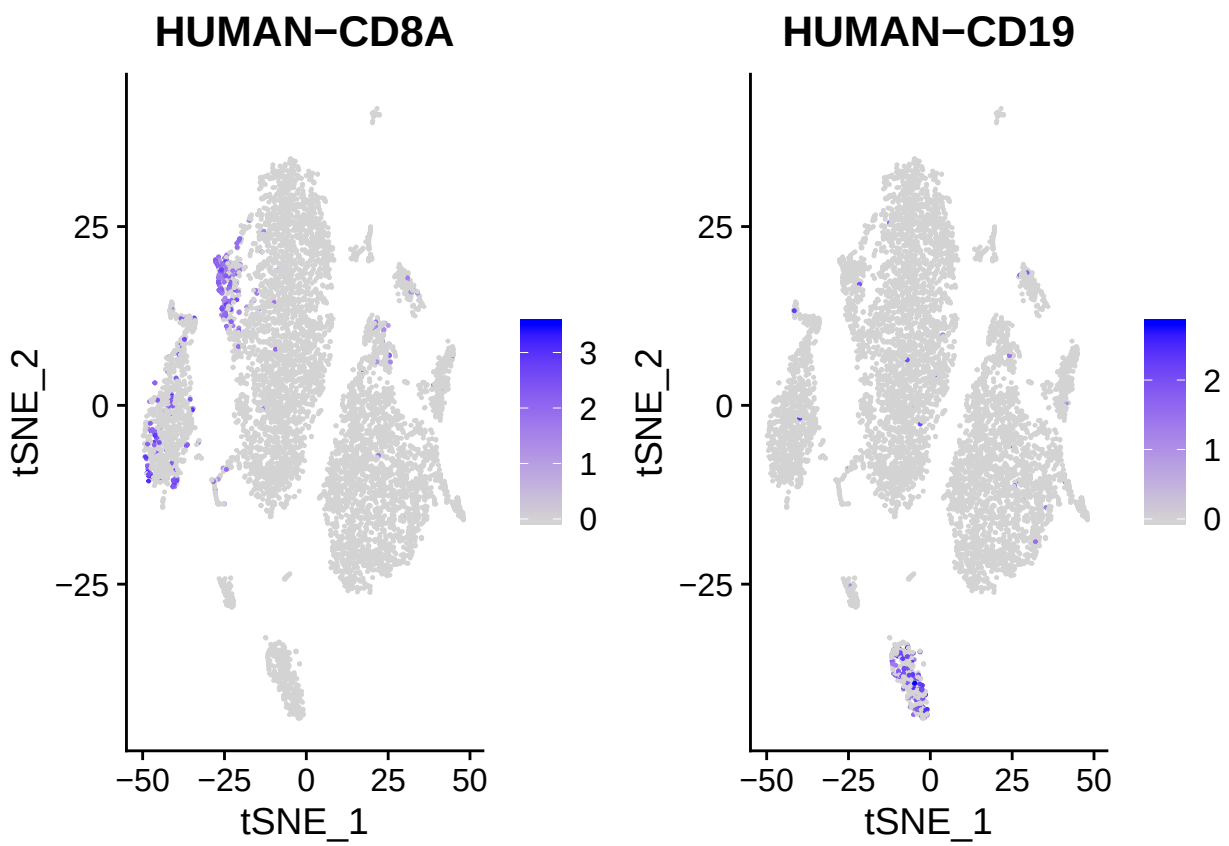
```
# VlnPlot shows expression probability distributions across clusters
VlnPlot(CBMC8K_human, features = c("HUMAN-CD8A", "HUMAN-CD19"))
```



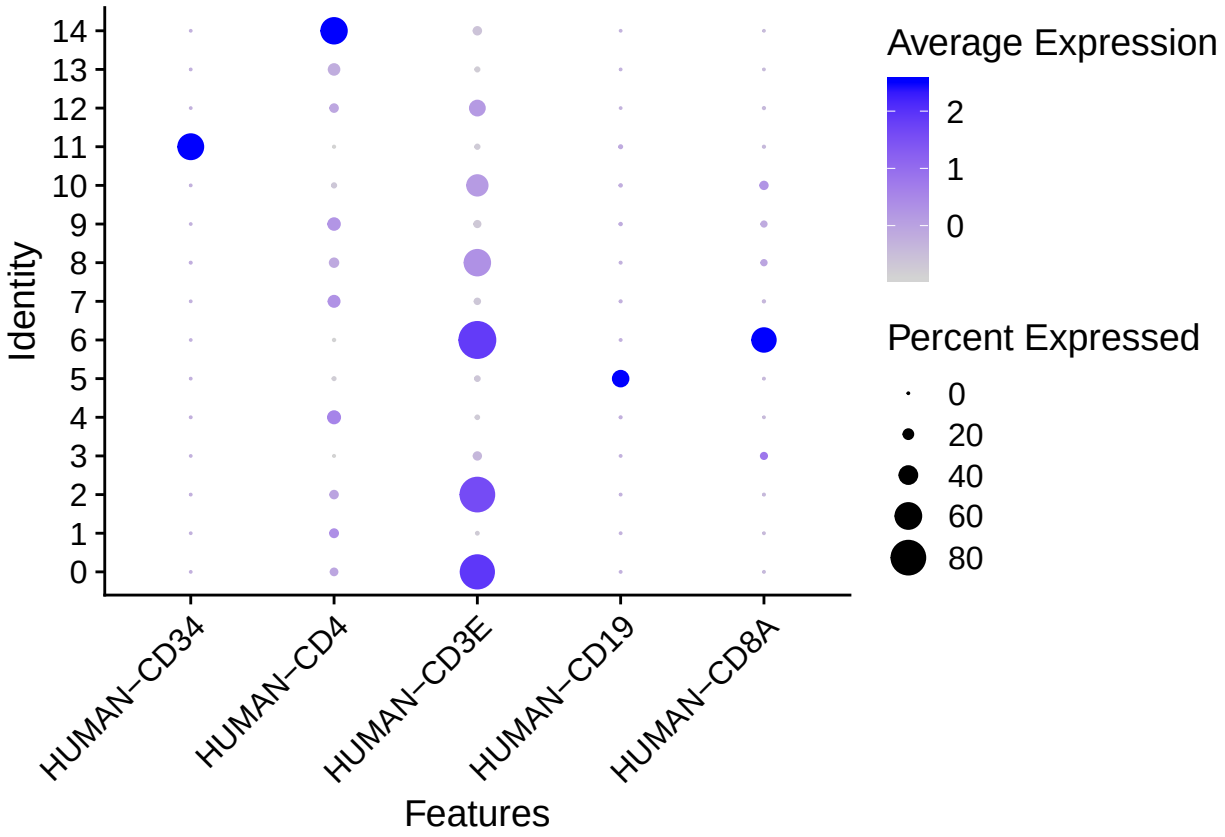

```
# you can plot raw counts as well
VlnPlot(CBMC8K_human, features = c("HUMAN-CD8A", "HUMAN-CD19"), slot = "counts", log = TRUE)
```




```
# FeaturePlot visualizes feature expression on a tSNE plot
FeaturePlot(CBMC8K_human, features = c("HUMAN-CD8A", "HUMAN-CD19"), reduction = "tsne")
```

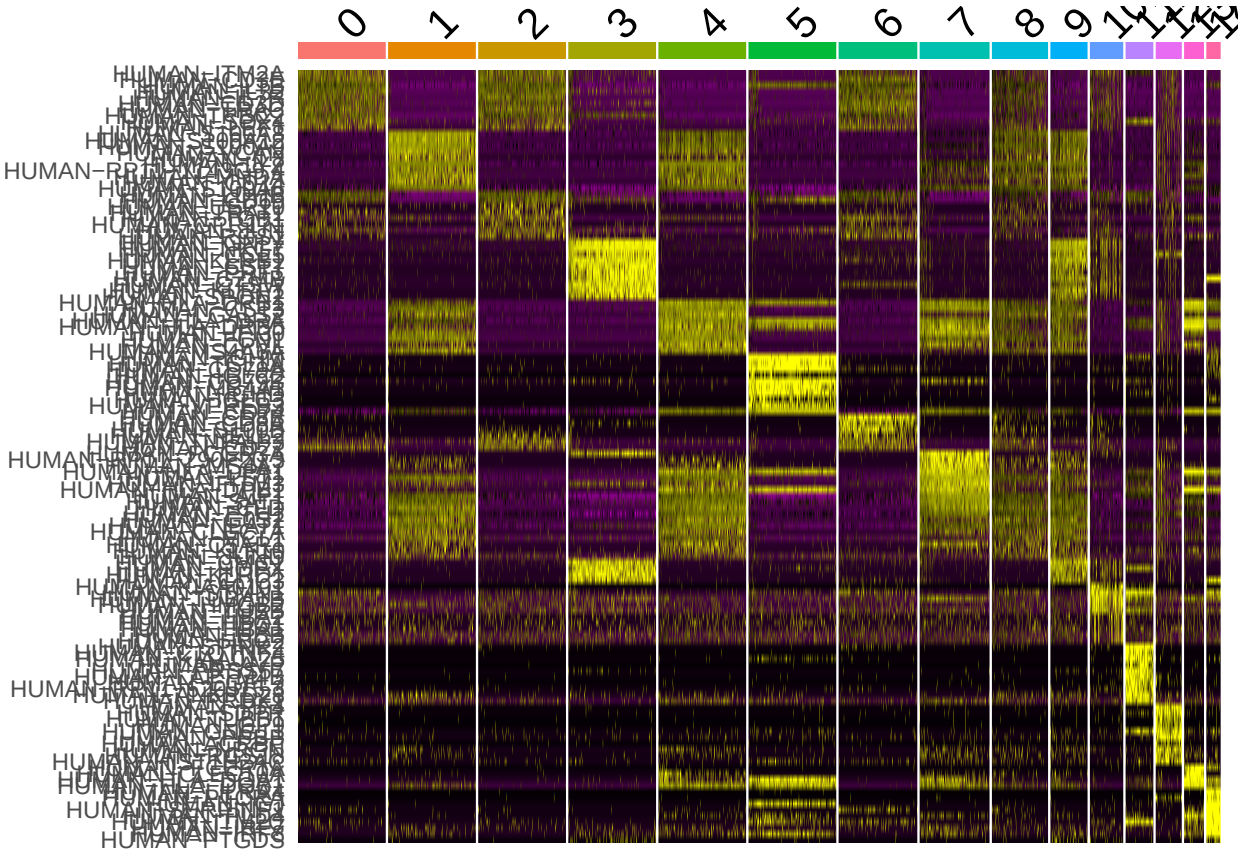



```
# There are additional methods such as DotPlot.
plot_1 <- DotPlot(CBMC8K_human, features =
                    c("HUMAN-CD8A", "HUMAN-CD19", "HUMAN-CD3E", "HUMAN-CD4", "HUMAN-CD34"))
plot_1 + theme(axis.text.x = element_text(angle = 45, hjust=1))
```




```
# DoHeatmap generates an expression heatmap for given cells and features.
# Downsample the clusters to a maximum of 300 cells each
# (makes the heatmap easier to see for small clusters)
CBMC8K_human_small <- subset(CBMC8K_human, downsample = 300)

# We will use the top 10 markers (or all markers if less than 10)
# for each cluster to generate the heatmap.
top10 <- CBMC8K_human.markers %>% group_by(cluster) %>% top_n(n = 10, wt = avg_logFC)
DoHeatmap(CBMC8K_human_small, features = top10$gene)  + NoLegend()
```



Now we have briefly explored the biomarkers of each cluster, what are their cell types? There are different types of blood cells, and we have accumulated significant amount of knowledge to recognize cell types based on biomarkers. Here's a table of canonical markers to easily match the unbiased clustering to known cell types:

```
# Assigning cell type identity to clusters #
celltype <- c("Naive CD4 T", "Memory CD4 T", "CD8 T", "B", "NK", "CD14+ Mono",
              "CD16+ Mono", "CD34+", "MK", "DC", "pDCs")
markergene <- c("CD3+ CD4+ CD8- CD2+/- CD57-", "CD3+ CD4+ CD8- CD2++ CD57-",
                "CD3+ CD4- CD8+", "CD3- CD19+", "CD16+ CD57+", "CD14++ CD16- CD15+",
                "CD14+ CD16+ CD15-", "CD34+", "CD41+", "CD141+ CD1c+",
                "CD11c+ CD45R+ CD123+ CD303+")
cell_type_table <- data.frame(CellType = celltype, Markers = markergene)
kable(cell_type_table) %>%
  kable_styling(bootstrap_options = "striped" , full_width = F , position = "center") %>%
  kable_styling(bootstrap_options = "bordered", full_width = F , position ="center") %>%
  column_spec(1,bold = F ) %>%
  column_spec(2,bold =T )
```

CellType	**Markers**
Naive CD4 T	**CD3+ CD4+ CD8- CD2+/- CD57-**
Memory CD4 T	**CD3+ CD4+ CD8- CD2++ CD57-**
CD8 T	**CD3+ CD4- CD8+**
B	**CD3- CD19+**
NK	**CD16+ CD57+**
CD14+ Mono	**CD14++ CD16- CD15+**
CD16+ Mono	**CD14+ CD16+ CD15-**
CD34+	**CD34+**
MK	**CD41+**
DC	**CD141+ CD1c+**
pDCs	**CD11c+ CD45R+ CD123+ CD303+**

Now we want to assign cell types to the clusters we got from the clustering process. We can visit the identities of each cell using *"Idents(CBMC8K_human)"*, or *"CBMC8K_human@meta.data\protect\T1\ textdollarseurat_clusters"*. *"levels(CBMC8K_human)"* returns the 15 clusters: "0" - "14". Based on the markers of each cluster we identified earlier, and the knowledge of features of each cell type, we can now assign the cell types to these clusters, and re-visualize them in the tSNE plot.

```
Idents(CBMC8K_human)[1:10]
```

```
## CACTCCAGTTTCCACC TGGTTAGAGATGTTAG CCTAGCTGTGAGGGTT TGGCTGGTCAGCAACT
##                4                6                7                4
## CAGAATCCATCAGTAC TCAGATGCAGTCAGCC GCGCAACGTGGTCTCG CAACCAAAGTACGTAA
##                4                5                7                7
## GTAACGTCAGCGTAAG TCACAAGGTACTCAAC
##                9                5
## Levels: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

```
CBMC8K_human@meta.data$seurat_clusters[1:10]
```

```
##  [1] 4 6 7 4 4 5 7 7 9 5
## Levels: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

```
levels(CBMC8K_human)
```

```
##  [1] "0"  "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
```

```
new.cluster.ids <- c("Naive CD4 T","CD14+ Mono", "Memory CD4 T", "NK", "CD14+ Mono",
                     "B", "CD8 T", "CD16+ Mono", "T/Mono doublets", "NK", "Multiplets",
                     "CD34+", "MK", "DC", "pDCs")
names(new.cluster.ids) <- levels(CBMC8K_human)
CBMC8K_human <- RenameIdents(CBMC8K_human, new.cluster.ids)

# Now we can see the IDs of each cell have been updated to their cell types.
Idents(CBMC8K_human)[1:10]
```

```
## CACTCCAGTTTCCACC TGGTTAGAGATGTTAG CCTAGCTGTGAGGGTT TGGCTGGTCAGCAACT
```

```
##       CD14+ Mono            CD8 T       CD16+ Mono       CD14+ Mono
## CAGAATCCATCAGTAC TCAGATGCAGTCAGCC GCGCAACGTGGTCTCG CAACCAAAGTACGTAA
##       CD14+ Mono                B       CD16+ Mono       CD16+ Mono
## GTAACGTCAGCGTAAG TCACAAGGTACTCAAC
##                NK                B
## 13 Levels: Naive CD4 T CD14+ Mono Memory CD4 T NK B CD8 T ... pDCs
```

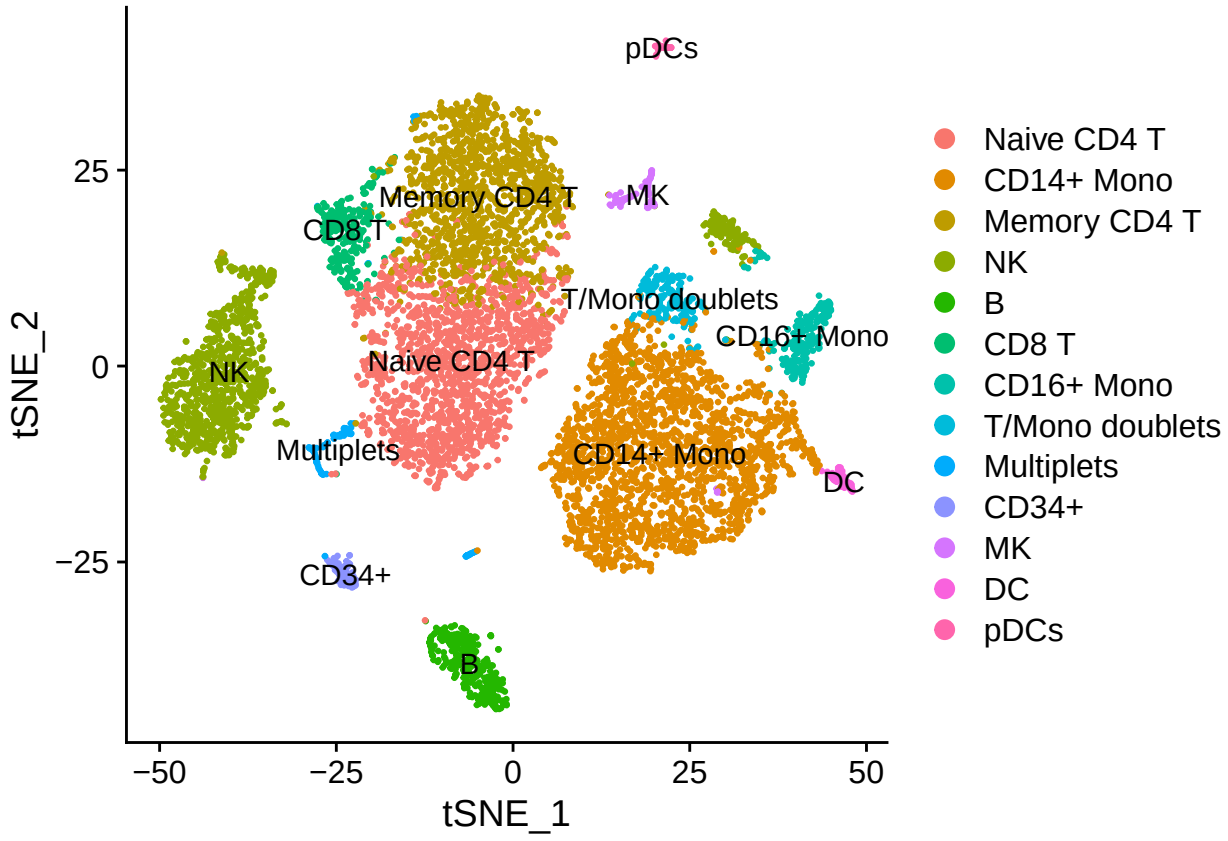
```
# "levels(CBMC8K_human)" returns the levels of these IDs (cell types).
levels(CBMC8K_human)
```

```
##  [1] "Naive CD4 T"     "CD14+ Mono"      "Memory CD4 T"    "NK"
##  [5] "B"               "CD8 T"           "CD16+ Mono"      "T/Mono doublets"
##  [9] "Multiplets"      "CD34+"           "MK"              "DC"
## [13] "pDCs"
```

```
# "CBMC8K_human@meta.data$seurat_clusters" still returns their original clusters.
CBMC8K_human@meta.data$seurat_clusters[1:10]
```

```
##  [1] 4 6 7 4 4 5 7 7 9 5
## Levels: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

```
# Now we can add the cell type information to the tsne plot instead of the clustering numbers:
DimPlot(CBMC8K_human, reduction = "tsne", label = TRUE, pt.size = 0.5)
```



5. Adding protein information

As mentioned in the *Introduction* section, the authors of this study developed a novel approach called "CITE-seq" to simultaneously obtain the expression levels of cell surface protein markers together with RNA expression of the same set of cells. The Seurat package has made it possible and convenient to perform multi-modal data analyses of single cell data. The protein expression levels, as measured for the antibody-devived tags (ADT), have been downloaded (see the "Method" section) and can be added to the Seurat object by adding an "ADT" assay to it. To be consistent, we only keep the ADT data for the cells that we have in the final "RNA" assay. We also perform normalization and scaling for the ADT data. Slightly differently, because there are only 10 features in the ADT data, we treat this data type as compositional data and apply the centered log ratio (CLR) transformation for the normalization purpose.

```
# Adding protein data to the Seurat object #

# Cell surface protein markers have also been accessed in this project.
# So we can intergrate this information to our Seurat project.
# By comparing to mouse cells as a non-specific background negative control,
# the authors ruled out 3 antibodies from their 13 antibodies used.
# So we first delete these 3 antibodies from the result.
load("CBMC8K_ADT_umi_human.Rdata")
CBMC8K_ADT_umi_human_10rows <- CBMC8K_ADT_umi_human[setdiff(rownames(
  x = CBMC8K_ADT_umi_human), c("CCR5", "CCR7", "CD10")), ]
dim(CBMC8K_ADT_umi_human_10rows)
```

```
## [1]   10 8005
```

```
# Because we filtered out some cells at the quality control stage,
# we now only keep the cells that exist in the final Seurat object.
CBMC8K_ADT_umi_human_10rows_7766 <- CBMC8K_ADT_umi_human_10rows[,colnames(CBMC8K_human)]
dim(CBMC8K_ADT_umi_human_10rows_7766)
```

```
## [1]   10 7766
```

```
# double check if the cells are the same ones:
identical(colnames(CBMC8K_human),colnames(CBMC8K_ADT_umi_human_10rows_7766))
```

```
## [1] TRUE
```

```
# Now we add the protein data to the "ADT" assay of the Seurat object.
CBMC8K_human[["ADT"]] <- CreateAssayObject(counts = CBMC8K_ADT_umi_human_10rows_7766)
CBMC8K_human
```

```
## An object of class Seurat
## 20410 features across 7766 samples within 2 assays
## Active assay: RNA (20400 features)
##  1 other assay present: ADT
##  3 dimensional reductions calculated: pca, umap, tsne
```

```
# Normalization: CLR
# For the protein data, we use the centered log ratio (CLR) transformation to normalize it.
CBMC8K_human <- NormalizeData(CBMC8K_human, assay = "ADT", normalization.method = "CLR")
```

```r
# Scaling
CBMC8K_human <- ScaleData(CBMC8K_human, assay = "ADT")
saveRDS(CBMC8K_human, file = "CBMC8K_human.rds")

# We can check the original counts, normalized data and scaled data using
# the following codes, respectively:
CBMC8K_human[["ADT"]]@counts[1:5,1:4]
```

```
## 5 x 4 sparse Matrix of class "dgCMatrix"
##        CACTCCAGTTTCCACC TGGTTAGAGATGTTAG CCTAGCTGTGAGGGTT TGGCTGGTCAGCAACT
## CD3                 820             3232              110              592
## CD4                1022             2720              550             1347
## CD8                  43             3581             1884               32
## CD45RA             2673            11822            24858             4808
## CD56                141              100              539               20
```

```r
CBMC8K_human[["ADT"]]@data[1:5,1:4]
```

```
##        CACTCCAGTTTCCACC TGGTTAGAGATGTTAG CCTAGCTGTGAGGGTT TGGCTGGTCAGCAACT
## CD3          1.7864994         3.024478        0.5107311        1.5232330
## CD4          1.7221311         2.582728        1.2451983        1.9541800
## CD8          0.5170136         4.049712        3.4230427        0.4080104
## CD45RA       0.8706611         1.965844        2.6327866        1.2520527
## CD56         1.3677943         1.123519        2.5004359        0.3472228
```

```r
CBMC8K_human[["ADT"]]@scale.data[1:5,1:4]
```

```
##        CACTCCAGTTTCCACC TGGTTAGAGATGTTAG CCTAGCTGTGAGGGTT TGGCTGGTCAGCAACT
## CD3          0.95218520        2.4240809       -0.5646411        0.6391742
## CD4          1.13939111        2.3687646        0.4580868        1.4708758
## CD8         -0.35032464        3.2917368        2.6456675       -0.4627023
## CD45RA      -0.01832639        1.7031657        2.7515174        0.5811741
## CD56         1.18370028        0.7049153        3.4037061       -0.8166455
```

```r
# Now we have two assays of this Seurat object: a default "RNA" assay and a new "ADT" assay.
CBMC8K_human$RNA
```

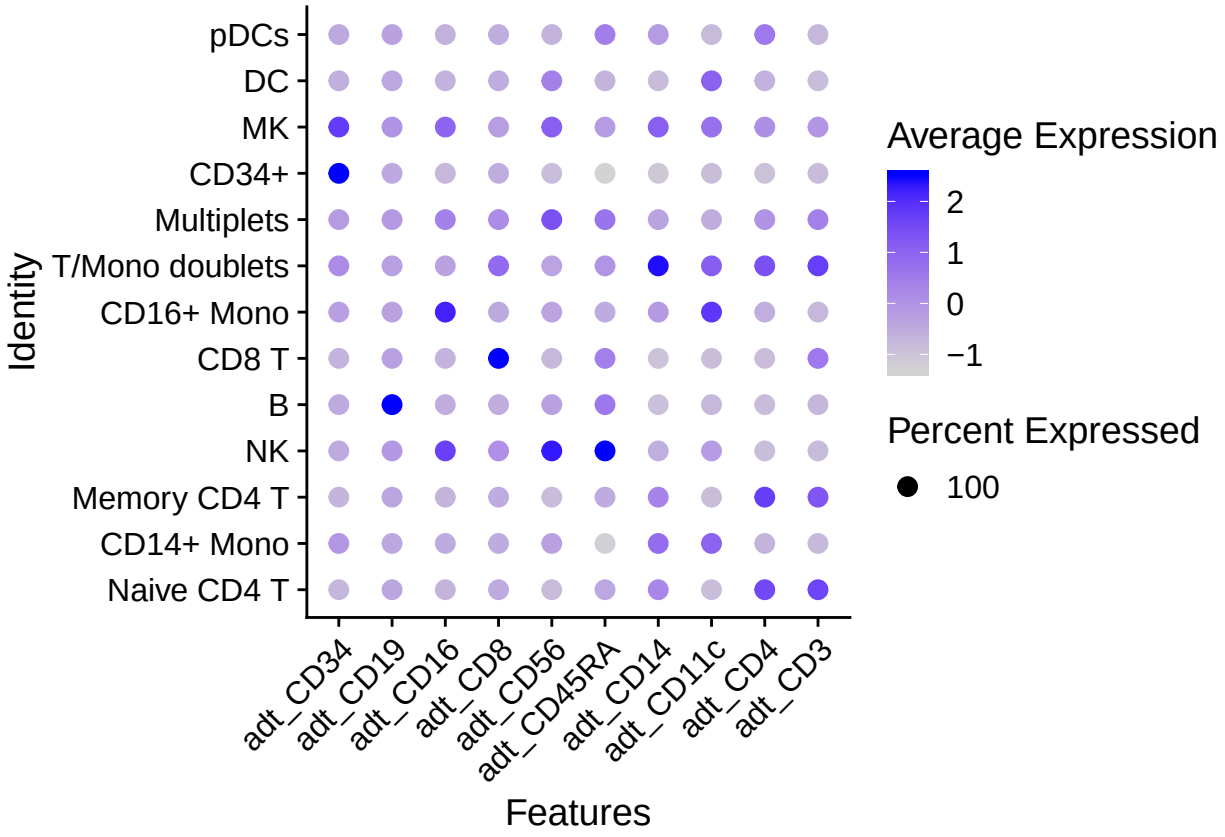
```
## Assay data with 20400 features for 7766 cells
## Top 10 variable features:
##  HUMAN-PPBP, HUMAN-HBG2, HUMAN-HBA1, HUMAN-HBB, HUMAN-PF4, HUMAN-HBA2,
## HUMAN-AHSP, HUMAN-HBG1, HUMAN-IGLC2, HUMAN-HBM
```

```r
CBMC8K_human$ADT
```

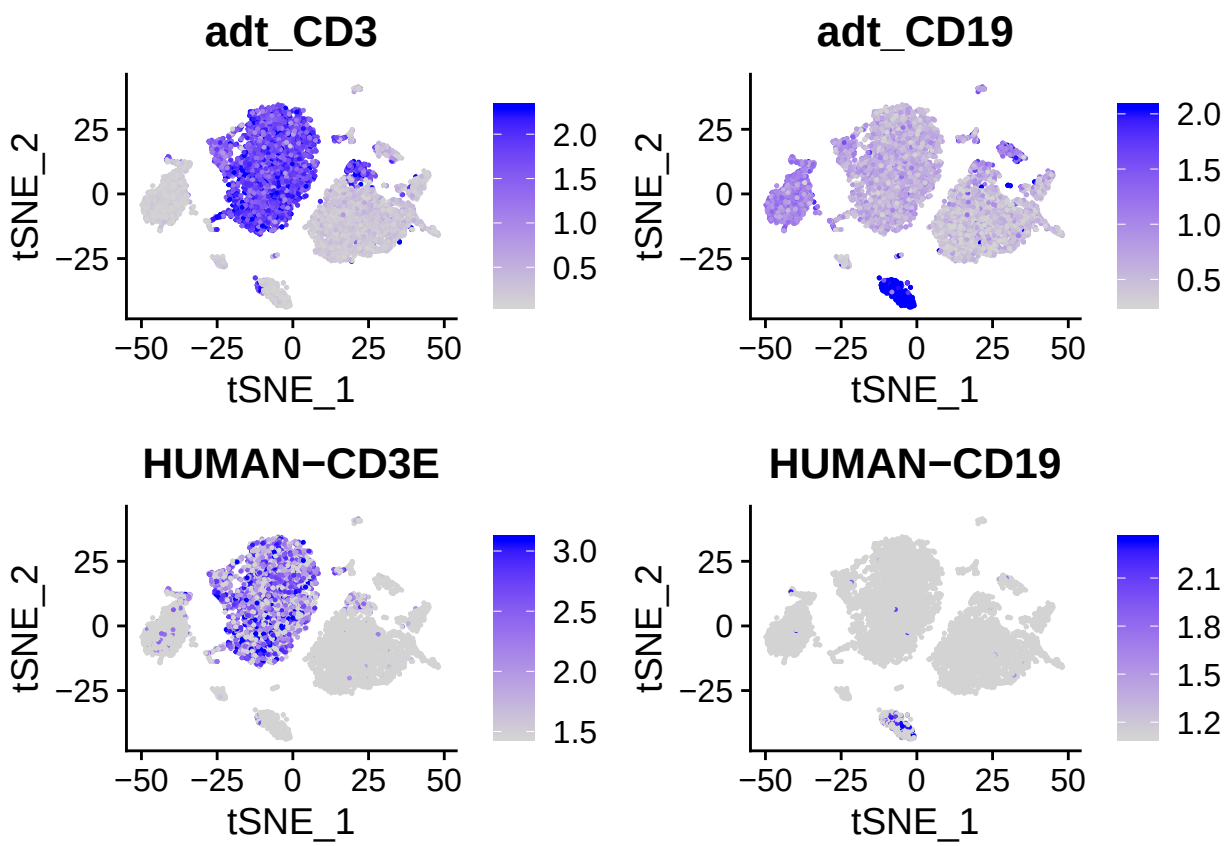
```
## Assay data with 10 features for 7766 cells
## First 10 features:
##  CD3, CD4, CD8, CD45RA, CD56, CD16, CD11c, CD14, CD19, CD34
```

We can visualize the cell surface protein expression levels across our different clusters, similarly to what we did for the RNA data. Moreover, we can also compare the protein and message-RNA (mRNA, which can be translated to a protein) levels of the same gene. Note that the name of genes could be different than the proteins they encode; for example, the gene that encodes CD11c is ITGAX. Also note that the cell surface protein levels do not necessarily agree with the mRNA levels because 1) it doesn't represent the total protein levels; 2) proteins can be regulated post-translationally.

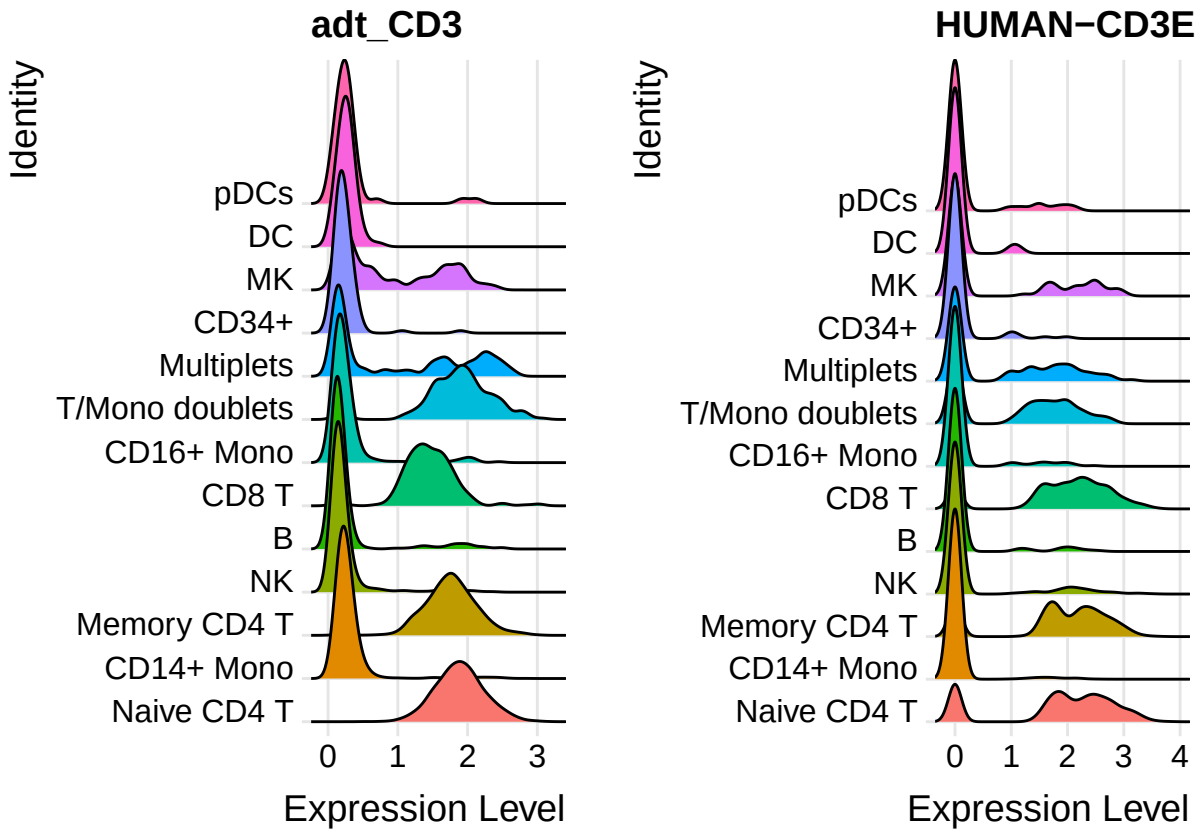
```
# We can visualize the overall protein expression pattern in different clusters
# using "DotPlot".
plot_1 <- DotPlot(CBMC8K_human, assay = "ADT",
        features = c("adt_CD3", "adt_CD4", "adt_CD11c", "adt_CD14", "adt_CD45RA",
                     "adt_CD56", "adt_CD8", "adt_CD16","adt_CD19","adt_CD34"))
plot_1 + theme(axis.text.x = element_text(angle = 45, hjust=1))
```



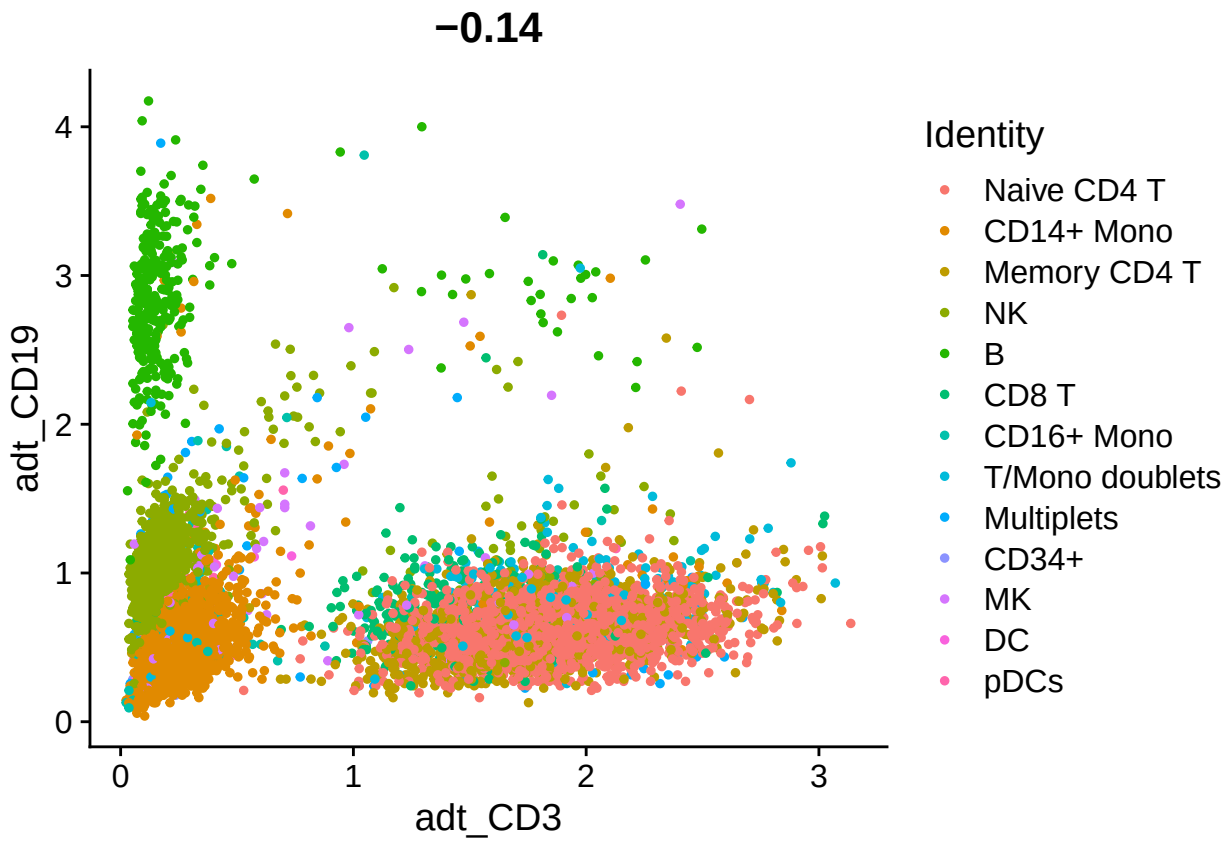

```
# We can compare the protein and mRNA levels of the same gene.
# Now we can check the expression patterns of the proteins in the clusters.
# In this plot, protein (ADT) levels are on top, and RNA levels are on the bottom.
FeaturePlot(CBMC8K_human, reduction = "tsne",
            features = c("adt_CD3", "adt_CD19", "HUMAN-CD3E","HUMAN-CD19"),
            min.cutoff = "q05", max.cutoff = "q95", ncol = 2)
```



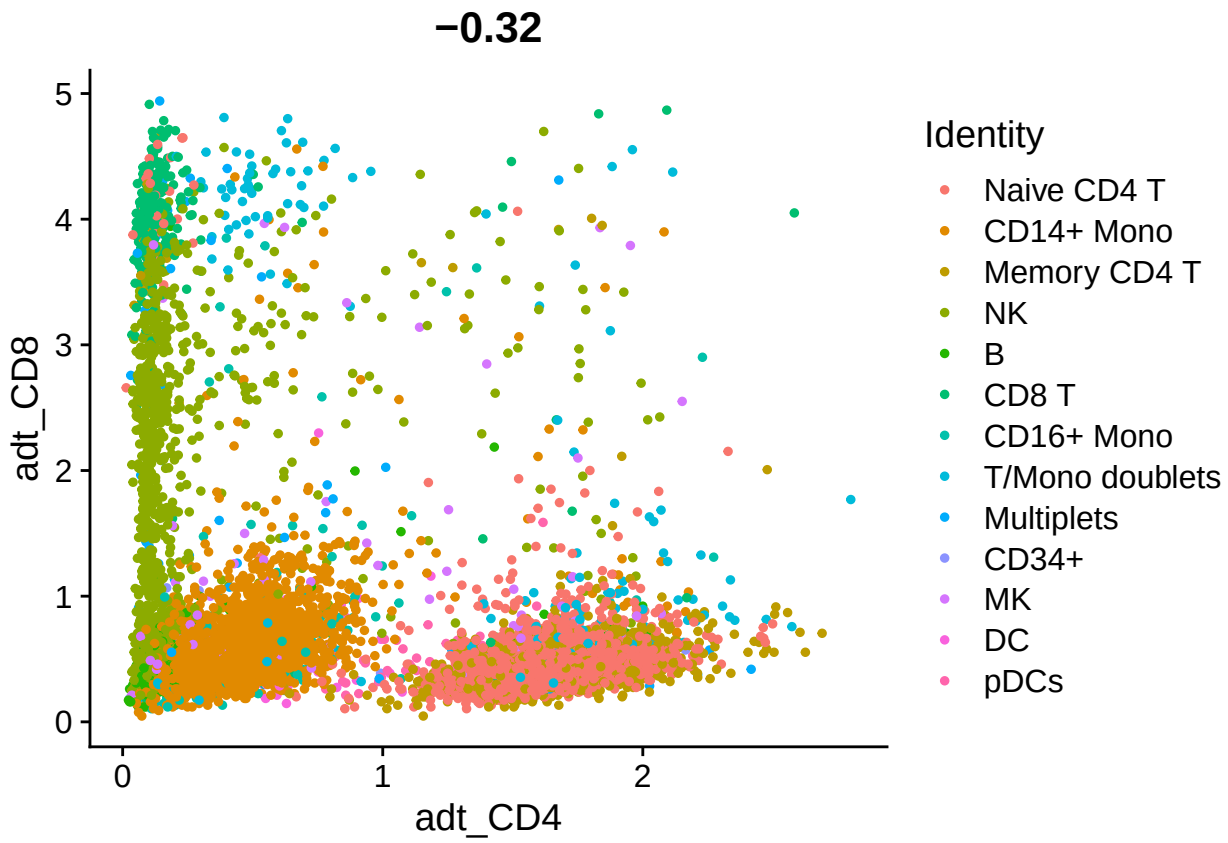

```
# We can also use "RidgePlot" to visualize the expression profile
# of these proteins in each cell type.
RidgePlot(CBMC8K_human, features = c("adt_CD3", "HUMAN-CD3E"), ncol = 2)
```



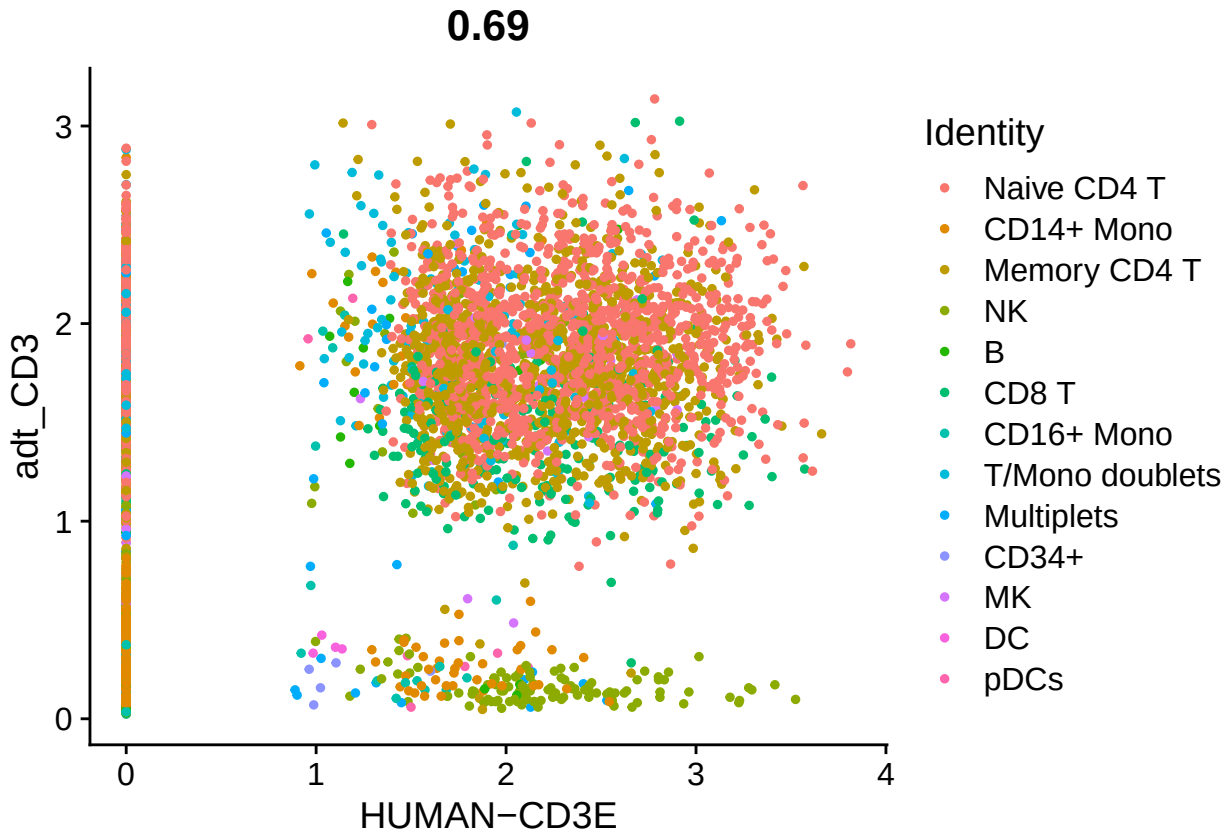

```
# We can even draw scatter plots based on protein levels
# (like the classical biaxial plots for FACS).
FeatureScatter(CBMC8K_human, feature1 = "adt_CD3", feature2 = "adt_CD19")
```




```
FeatureScatter(CBMC8K_human, feature1 = "adt_CD4", feature2 = "adt_CD8")
```



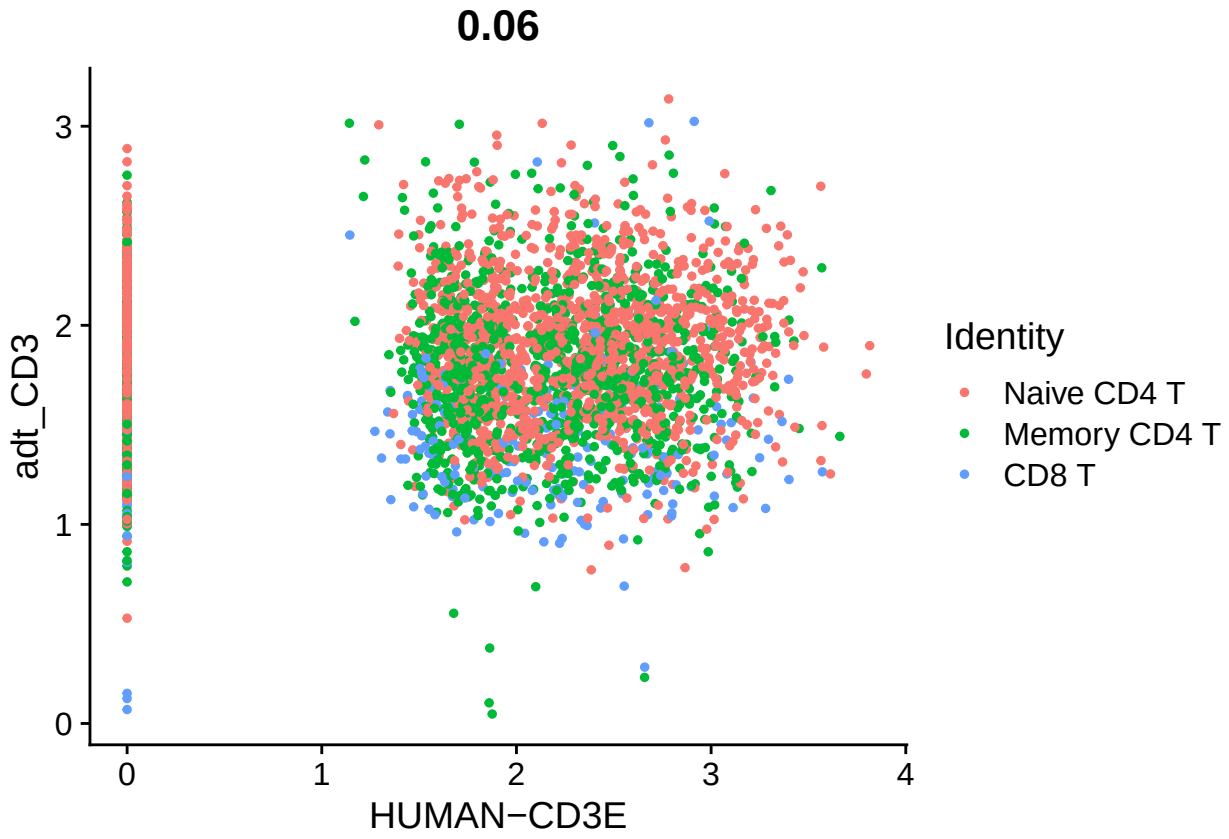

```
# view relationship between protein and RNA
FeatureScatter(CBMC8K_human, feature1 = "HUMAN-CD3E", feature2 = "adt_CD3")
```



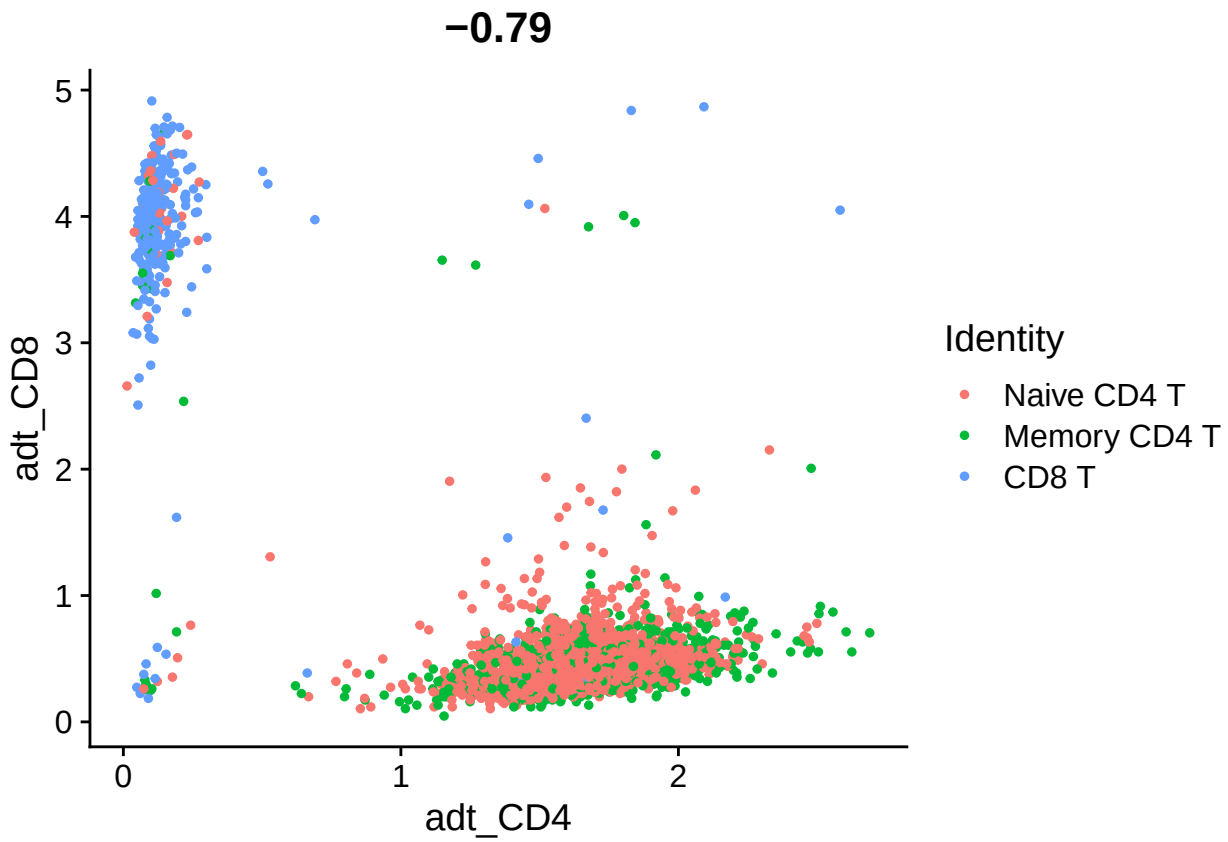

```
# calculate the correlation
cor(CBMC8K_human[["RNA"]]@data["HUMAN-CD3E",], CBMC8K_human[["ADT"]]@data["CD3",])
```

```
## [1] 0.6935498
```

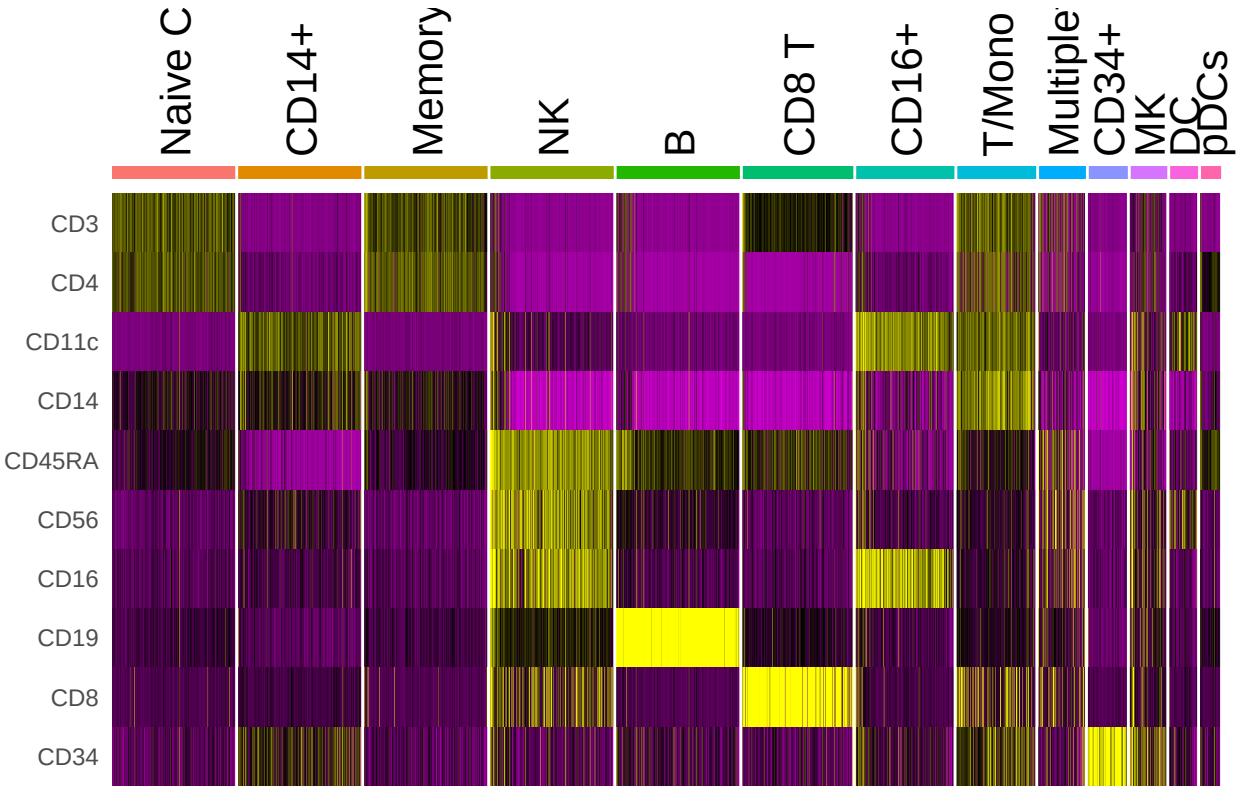
```
# We can also restrict the pattern to one or more specific cell types:
tcells <- subset(CBMC8K_human, idents = c("Naive CD4 T", "Memory CD4 T", "CD8 T"))
FeatureScatter(tcells, feature1 = "HUMAN-CD3E", feature2 = "adt_CD3")
```




```
FeatureScatter(tcells, feature1 = "adt_CD4", feature2 = "adt_CD8")
```




```
# We can also draw a heatmap based on protein levels, similarly to what we did for the mRNA data.
CBMC8K_human_small <- subset(CBMC8K_human, downsample = 300)
adt.markers <- FindAllMarkers(CBMC8K_human_small, assay = "ADT", only.pos = TRUE)
DoHeatmap(CBMC8K_human_small, features = unique(adt.markers$gene), assay = "ADT", angle = 90) + NoLegend
```



Instead of visualizing levels of these proteins on the clusters that we generated based on the RNA data, we can also directly cluster the cells using the protein information.

```
# Cluster directly on protein levels #

# To make it easier, we can switch the default assay to the 'ADT' assay,
# so that we don't need to specify it each time.
DefaultAssay(CBMC8K_human) <- "ADT"

# Although we could run a PCA, because we only have 10 features here,
# instead of doing PCA, we can just use a standard euclidean distance matrix here.
adt.data <- GetAssayData(CBMC8K_human, slot = "data")
adt.dist <- dist(t(adt.data))

# Before we re-cluster the data based on protein levels,
# we can store the currrent cluster IDs based on RNA expressions for later comparison.
CBMC8K_human[["rnaClusterID"]] <- Idents(CBMC8K_human)

# Using our distance matrix defined only on protein levels, we can now re-do tSNE
# using the following codes:
CBMC8K_human[["tsne_adt"]] <- RunTSNE(adt.dist, assay = "ADT", reduction.key = "adtTSNE_")
CBMC8K_human[["adt_snn"]] <- FindNeighbors(adt.dist)$snn
CBMC8K_human <- FindClusters(CBMC8K_human, resolution = 0.2, graph.name = "adt_snn")
```

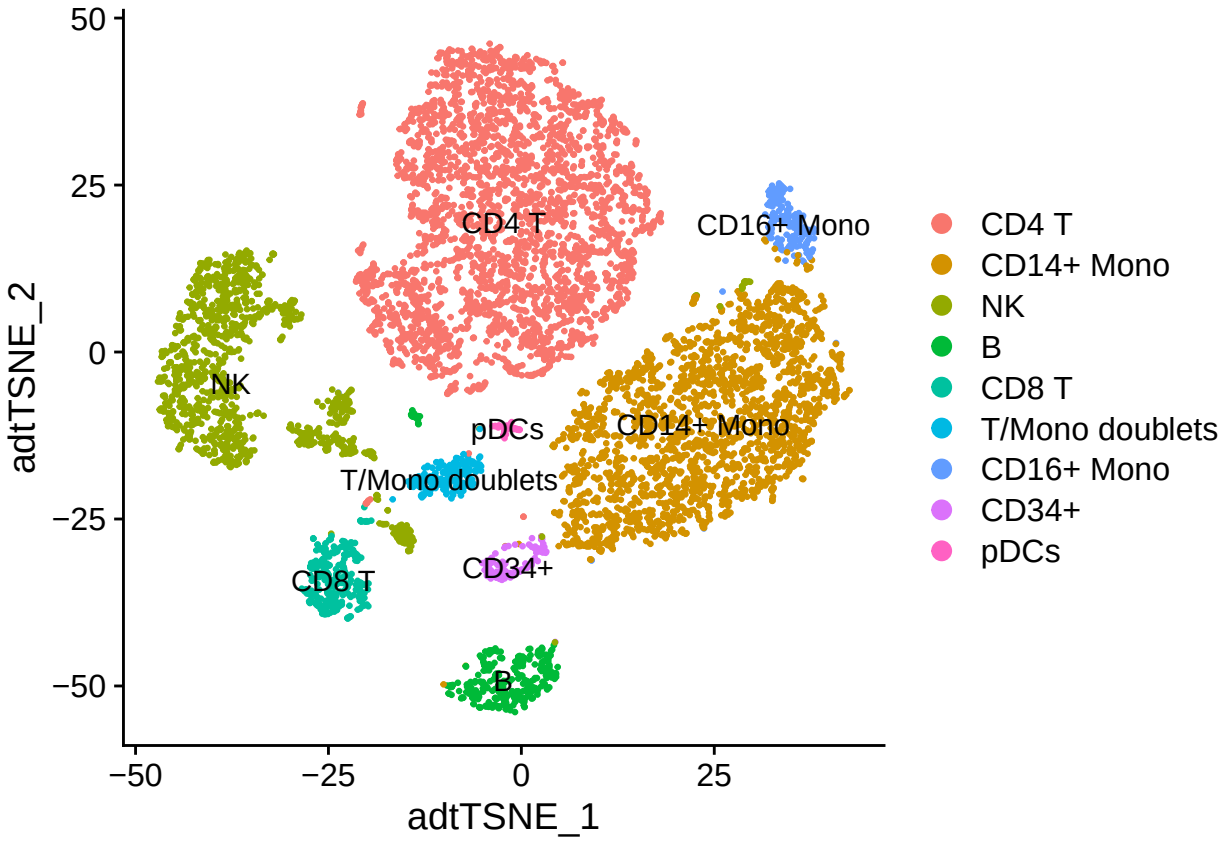
```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 7766
## Number of edges: 254306
##
```

```
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9473
## Number of communities: 11
## Elapsed time: 0 seconds
```

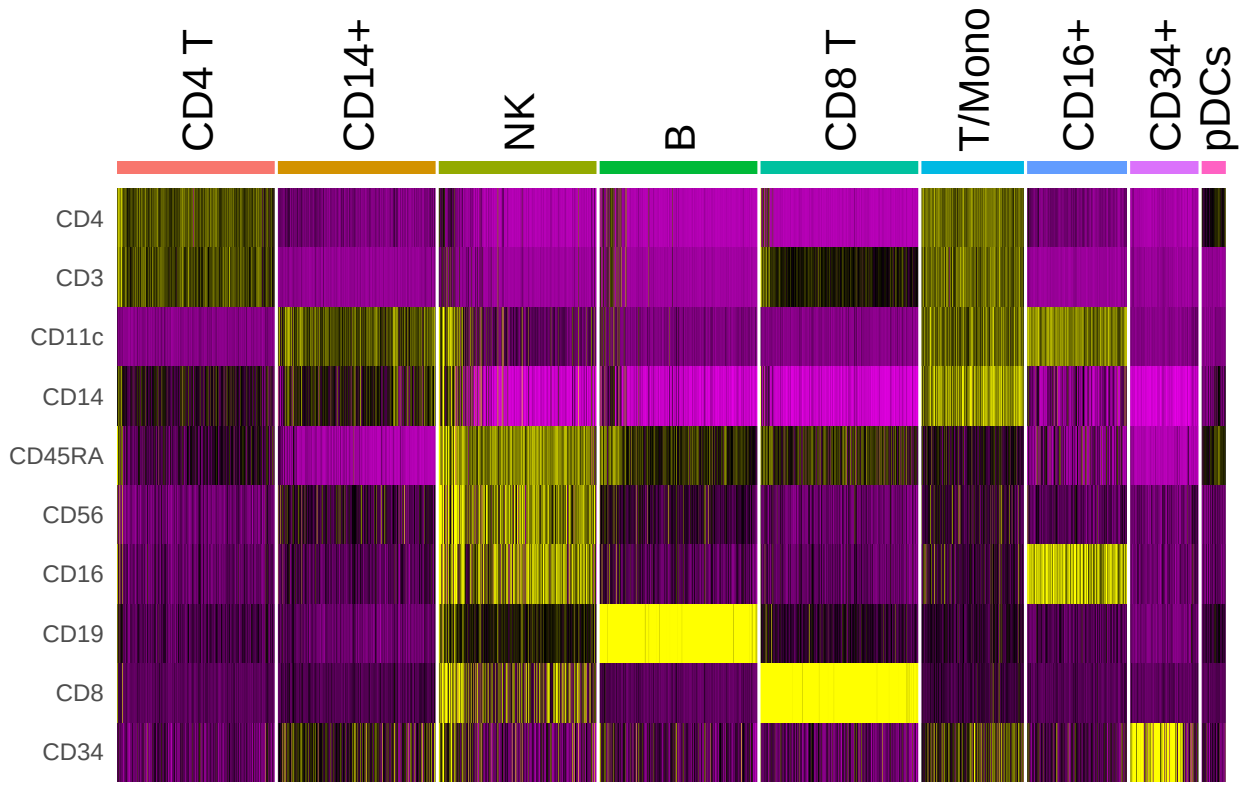
```
# To annotate the protein clustering, we can compare the RNA and protein clustering:
clustering.table <- table(Idents(CBMC8K_human), CBMC8K_human$rnaClusterID)
clustering.table
```

```
##
##      Naive CD4 T CD14+ Mono Memory CD4 T   NK    B CD8 T CD16+ Mono
##   0         1611          0          1365   28    0    19          0
##   1            0       2113             0    3    0     0         30
##   2            0          0             2  891    2     4          0
##   3            0          8             0    2  310     0          2
##   4           29          0            15    4    1   244          0
##   5            2         31             2  124    3     2         20
##   6            3         52             2    0    0     0         10
##   7            0          9             0    2    0     0        177
##   8            0          4             2    1    0     0          0
##   9            0          0             1    0    0     0          0
##   10           1          2             2    0   24     0          0
##
##      T/Mono doublets Multiplets CD34+   MK   DC pDCs
##   0                4         38     1   24    0    2
##   1                4          5     1   21   57    0
##   2                0         34     0    5    2    0
##   3                1          2     1    3    0    0
##   4                0          9     0    3    0    0
##   5               57         10     0   16    6    2
##   6              124          2     0    1    0    1
##   7                1          0     0    1    0    0
##   8                0         14    92   14    2    1
##   9                0          0     0    1    1   43
##   10               1          0     0    0    0    0
```

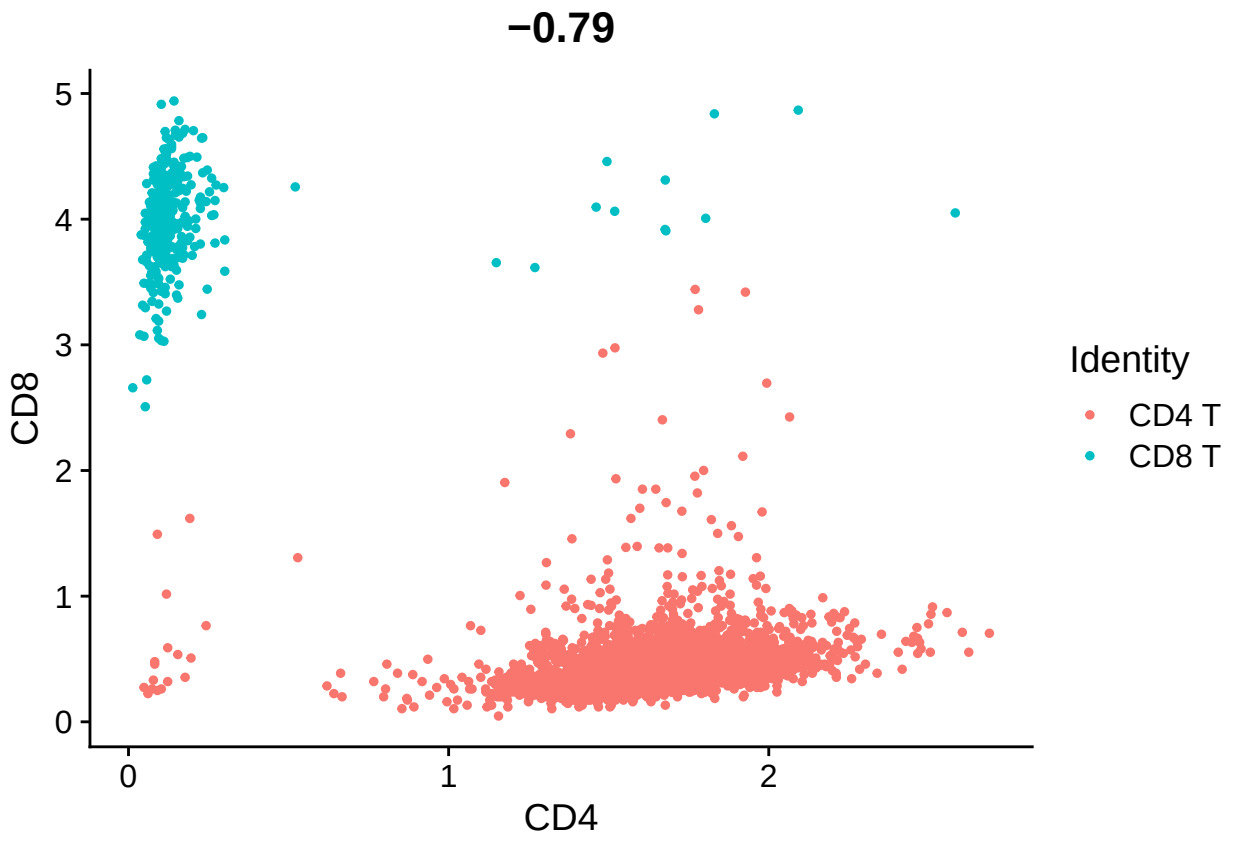
```
# Based on the clustering table, we can now label the clusters:
new.cluster.ids <- c("CD4 T", "CD14+ Mono", "NK", "B", "CD8 T", "NK", "T/Mono doublets",
                     "CD16+ Mono", "CD34+","pDCs", "B")
names(new.cluster.ids) <- levels(CBMC8K_human)
CBMC8K_human <- RenameIdents(CBMC8K_human, new.cluster.ids)
DimPlot(CBMC8K_human, reduction = "tsne_adt", label = TRUE, pt.size = 0.5)
```



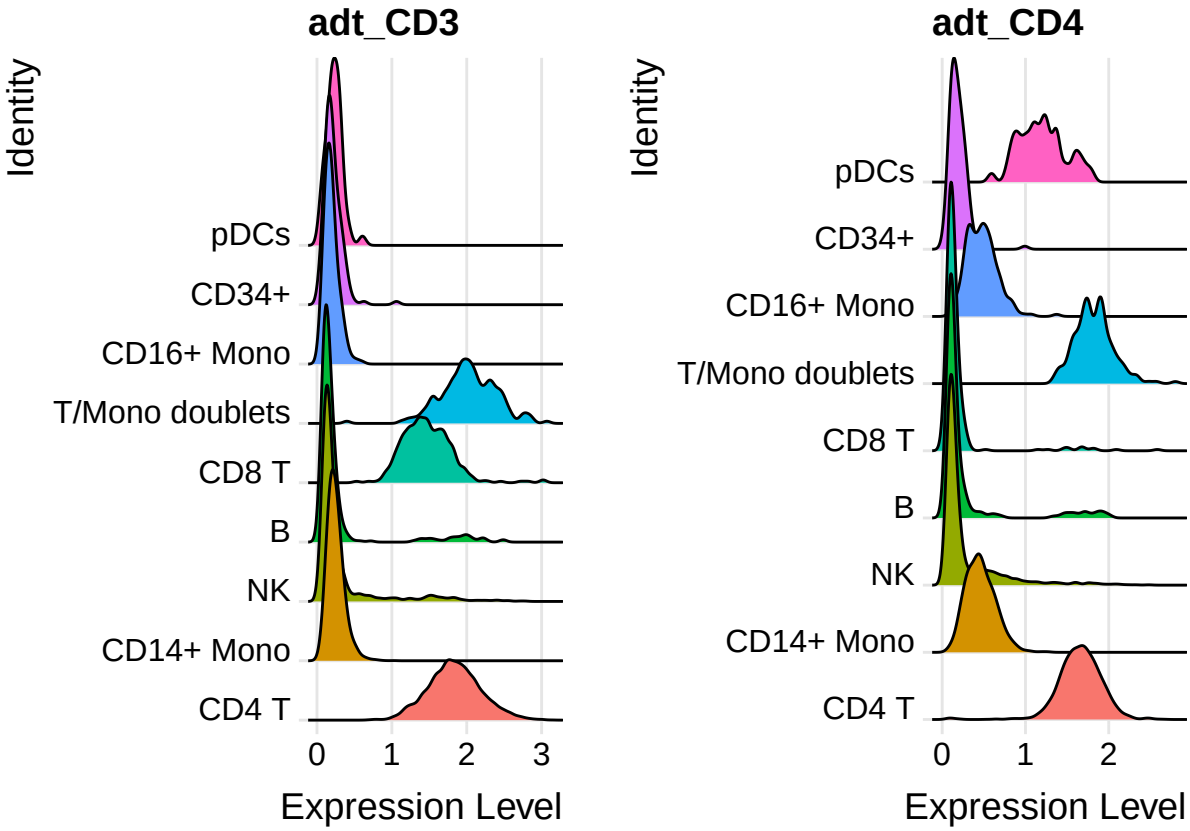

```
# Draw a heatmap using protein markers based on these clusters.
CBMC8K_human_small2 <- subset(CBMC8K_human, downsample = 300)
adt.markers2 <- FindAllMarkers(CBMC8K_human_small2, assay = "ADT", only.pos = TRUE)
DoHeatmap(CBMC8K_human_small2, features = unique(adt.markers2$gene),
          assay = "ADT", angle = 90) + NoLegend()
```




```
# Similar to what we did for the RNA clusters, we can visualize pretein levels
# based on the protein clusters:
tcells <- subset(CBMC8K_human, idents = c("CD4 T", "CD8 T"))
FeatureScatter(tcells, feature1 = "CD4", feature2 = "CD8")
```




```
RidgePlot(CBMC8K_human, features = c("adt_CD3", "adt_CD4"),ncol = 2)
```



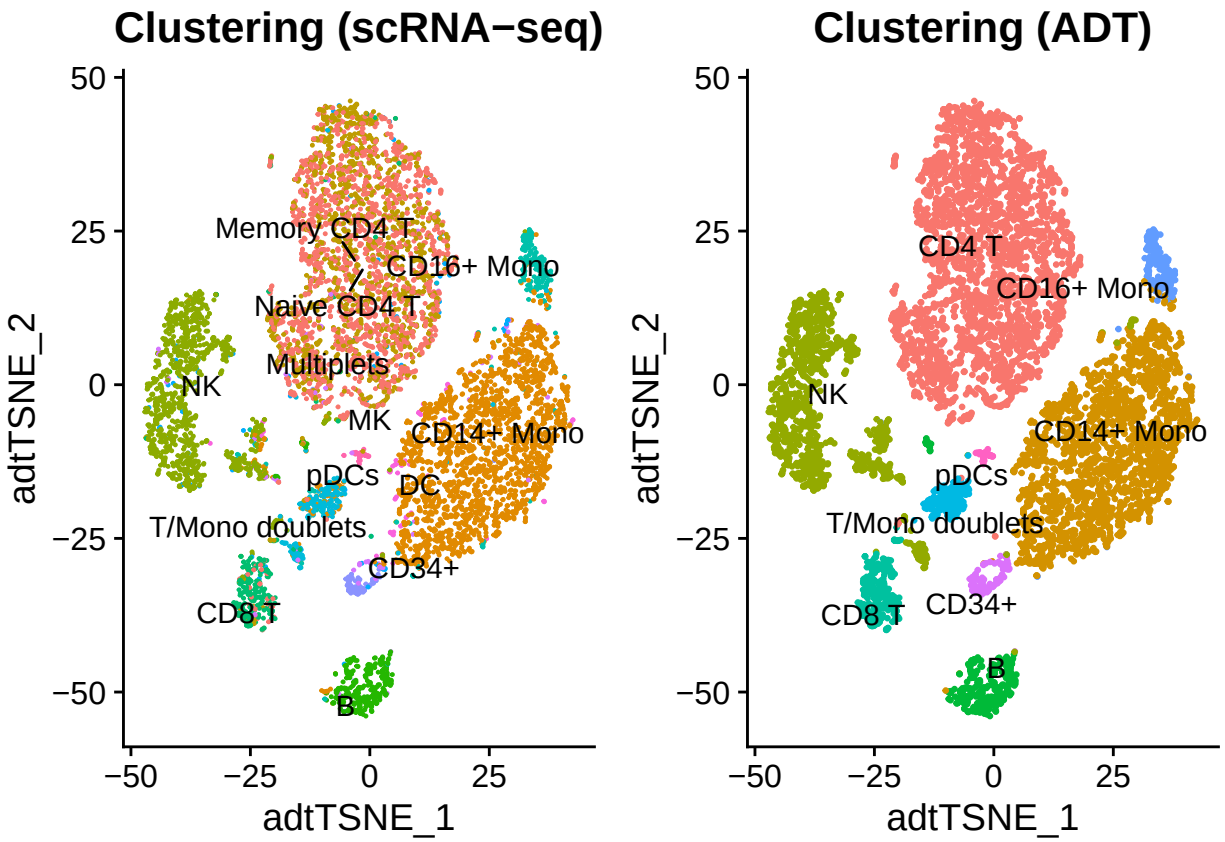
At the end, let's compare the clustering based on either RNA or protein levels of marker genes. Here we will visualize both the RNA and protein clustering based on a tSNE generated using the ADT distance matrix.

```
# comparing RNA and protein clustering #

# At the end, let's compare the clustering based on either RNA or protein levels
# of marker genes. Here we will visualize both the RNA and protein clustering based
# on a tSNE generated using the ADT distance matrix.
tsne_rnaClusters <- DimPlot(CBMC8K_human, reduction = "tsne_adt", group.by = "rnaClusterID") +
  NoLegend()
tsne_rnaClusters <- tsne_rnaClusters + ggtitle("Clustering (scRNA-seq)") +
  theme(plot.title = element_text(hjust = 0.5))
tsne_rnaClusters <- LabelClusters(plot = tsne_rnaClusters, id = "rnaClusterID", size = 4)

tsne_adtClusters <- DimPlot(CBMC8K_human, reduction = "tsne_adt", pt.size = 0.5) +
  NoLegend()
tsne_adtClusters <- tsne_adtClusters + ggtitle("Clustering (ADT)") +
  theme(plot.title = element_text(hjust = 0.5))
tsne_adtClusters <- LabelClusters(plot = tsne_adtClusters, id = "ident", size = 4)

CombinePlots(plots = list(tsne_rnaClusters, tsne_adtClusters), ncol = 2)
```




```
# Let reset the default assay back to "RNA" and save the Seurat object.
DefaultAssay(CBMC8K_human) <- "RNA"
saveRDS(CBMC8K_human, file = "CBMC8K_human.rds")
```

Although we only have 10 protein markers and don't have enough information to distinguish different subtypes of CD4 T cells, surprisingly, we could successfully cluster the cells to the main cell types as we did for RNA data. This is very promising because we can imagine that with "higher-throughput" of cell surface biomarkers, we could potentially further enhance the accuracy of this approach. Because cell surface protein markers are the units that cells accept extracellular signal and transduce to downstream biological processes, they are more directly linked to pathophysiological functions and catalogues of the cells. By integating transcriptomics data and other layers of biological data, we could better understand how these cells are functioning.

Conclusion

The single-cell RNA-sequencing technology is changing the whole field of life sciences, facilitating high-resolution analysis of massive data from biological samples. In this project, I analyzed a public dataset of RNA-seq data from cord blood mononuclear cells (CBMCs). Blood cells have been well characterized and can be classified into different subpopulations using the traditional fluorescence-activated cell sorting (FACS) technology based on the existence and/or combinations of cell surface protein markers. The transcriptomics (the overall RNA expression levels, especially mRNA expression of genes) data of more than 20,000 human genes on more than 8,000 human blood cells is a big challenge for data analysis. Using linear dimentionality reduction (PCA) and non-linear dimensional reduction (UMAP and tSNE), we can cluster the cells into subgroups in a 2-D space and assign identities to them based on their biomarkers. In addtion, multiple layers of biological information, i.e. multi-omics data, can be integrated together for a more comprehensive

analysis, which further add the complexity to the task. Here, the expression levels of a serious of cell surface protein markers are integrated into our object, not only validating the clustering based on the RNA data, but also demonstrating the feasibility of multi-omics single-cell analysis.

References

[1] Stoeckius, M., Hafemeister, C., Stephenson, W. et al. Simultaneous epitope and transcriptome measurement in single cells. Nat Methods 14, 865–868 (2017) doi:10.1038/nmeth.4380

[2] Stuart, T., Butler, A., Hoffman, P., Hafemeister, C., Papalexi, E., Mauck III, W. M. et al. Comprehensive Integration of Single-Cell Data. Cell 177, 1888-1902 (2019) doi:10.1016/j.cell.2019.05.031.