Preliminary single cell analysis

Jieun Jeong

2025-02-25

This example analyzes data from a single cell experiment in one of the forms accepted by Seurat package, namely a directory with barcode.tsv, the list of cell identifiers, genes.tsv, the list of genes with Ensembl and Symbol rdentifiers, and matrix.mxt, counts (in 3rd column) of gene UMIs or fragments detected for each gene (number in 1st column, row number in genes.tsv) and each cell (number in 2nd column, row number in barcodes.tsv).

We create the initial Seurat object:

```
pa = "" # change it if you have a path to a directory described above
library(dplyr)
library(Seurat)
library(patchwork)
library(ggplot2)

if (pa != "") {
  pbmc.data <- Read10X(data.dir = pa)
  # Initialize the Seurat object with the raw (non-normalized data).
  pbmc <- CreateSeuratObject(counts = pbmc.data, project = "pbmc3k",
          min.cells = 3, min.features = 200)
} else
{
  pb = "/Users/jieun/Work/Git_test/Seurat/data/filtered_gene_bc_matrices/hg19" # change it if you have
  pbmc <- readRDS(file.path(pb,"pbmc.RDS"))
}
pbmc
```

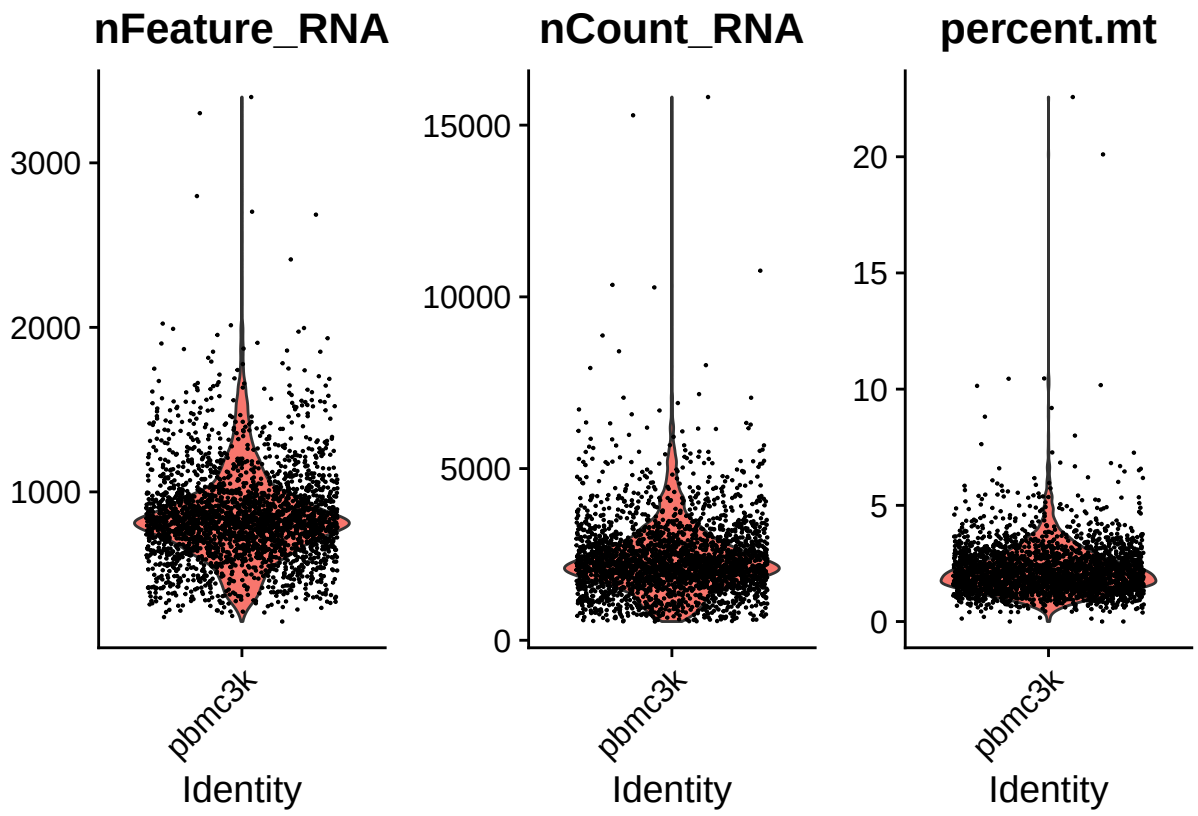
```
## An object of class Seurat
## 13714 features across 2700 samples within 1 assay
## Active assay: RNA (13714 features, 0 variable features)
##  1 layer present: counts
```

Quality contol metrics can be used to exclude some cells and gene from further analysis according to the distributions of the metrics in the data.

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

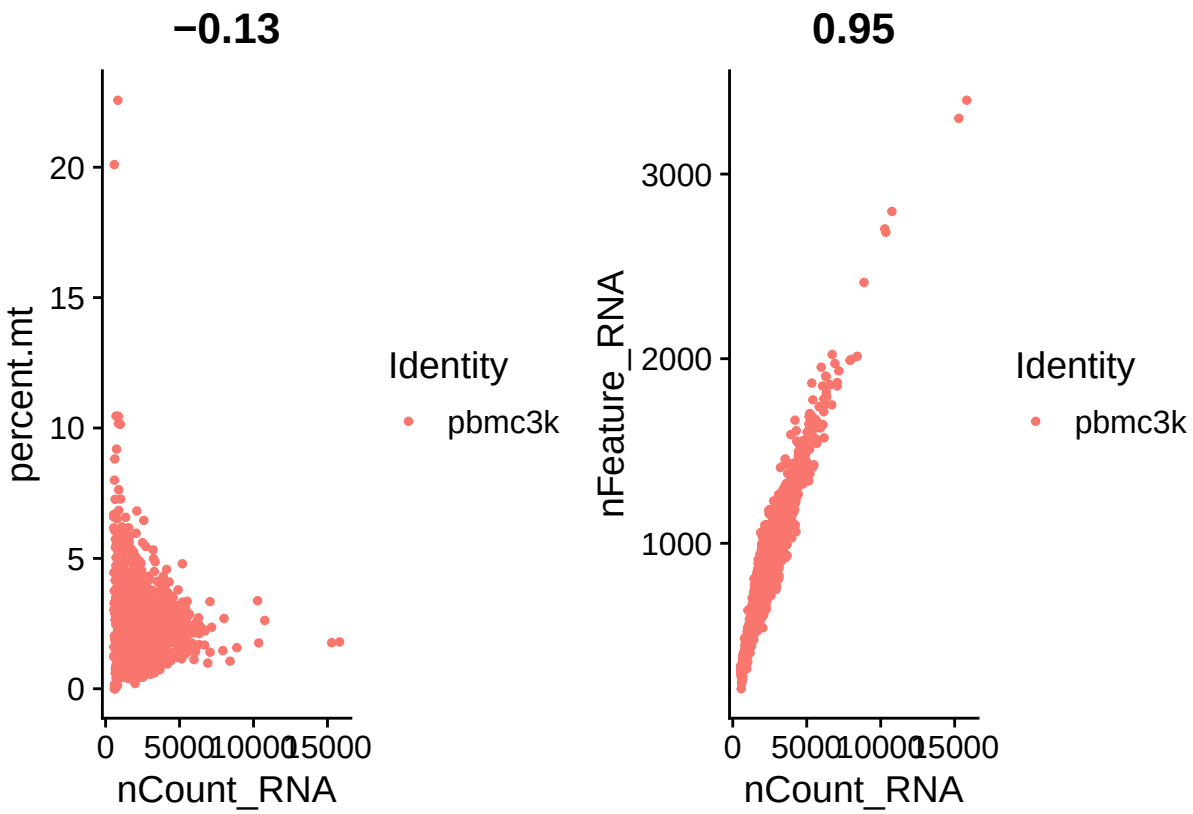
```
pbmc[["percent.mt"]] <- PercentageFeatureSet(pbmc, pattern = "^MT-")

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



FeatureScatter is typically used to visualize feature-feature relationships, but can be used for anything calculated by the object, i.e. columns in object metadata, PC scores etc.

```
# Visualize QC metrics as a scatter plot
plot1 <- FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "percent.mt")
plot2 <- FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
plot1 + plot2
```



One may use median statistics as a guide to filtering parameters.

```
MEDnF <- median(pbmc$nFeature_RNA)
MEDmt <- median(pbmc$percent.mt)
sprintf("MEDnF = %d, MEDmt = %6.4f",MEDnF,MEDmt)
```

```
## [1] "MEDnF = 816, MEDmt = 2.0308"
```

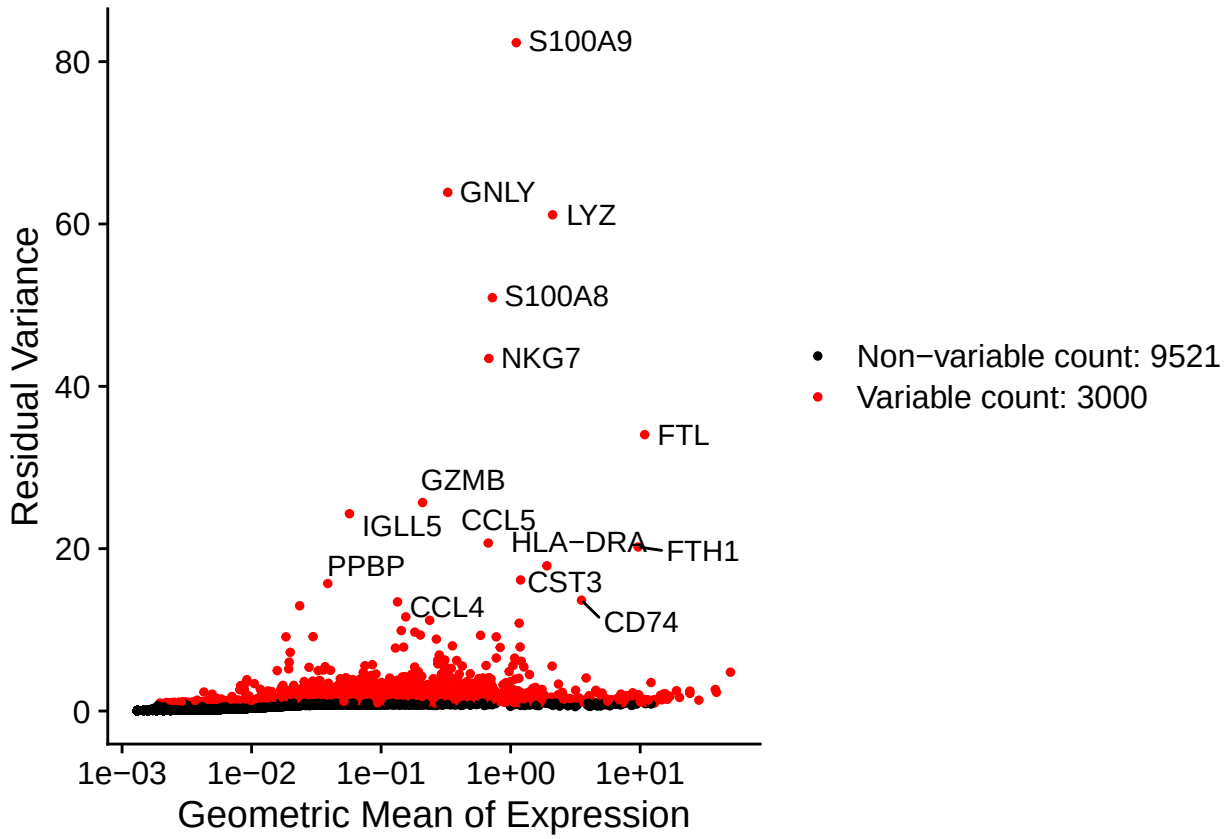
A reasonable choice of limits on `nFeature` is at least $0.25 \times MEDnF$ and at most $3 \times MEDnF$. Similarly, one can require that `percent.mt` is at most $2.5 \times MEDmt$. One can revise those thresholds if the first round of the analysis provides reasons.

Improvements in technology are increasing the counts per cell and `SCTransform()` takes advantage of those improvements using a more accurate null model. The distribution of UMIs of a gene among cells is overdispersed compared to Poisson distribution, and Poisson model deviates from the data stronger when the number of UMIs per cell increases.

```
pbmc <- subset(pbmc, subset = nFeature_RNA > 0.25*MEDnF &
                 nFeature_RNA < 3*MEDnF & percent.mt < 2.5*MEDmt)
```

```
library(sctransform)
pbmc <- SCTransform(pbmc) # log normalizes, scales, finds variable features
# Identify most highly variable genes (illustration)
topf <- head(VariableFeatures(pbmc), 15)
# plot variable features with top labels
p <- VariableFeaturePlot(pbmc)
```

```
p <- LabelPoints(plot = p, points = topf, repel = TRUE)
p
```



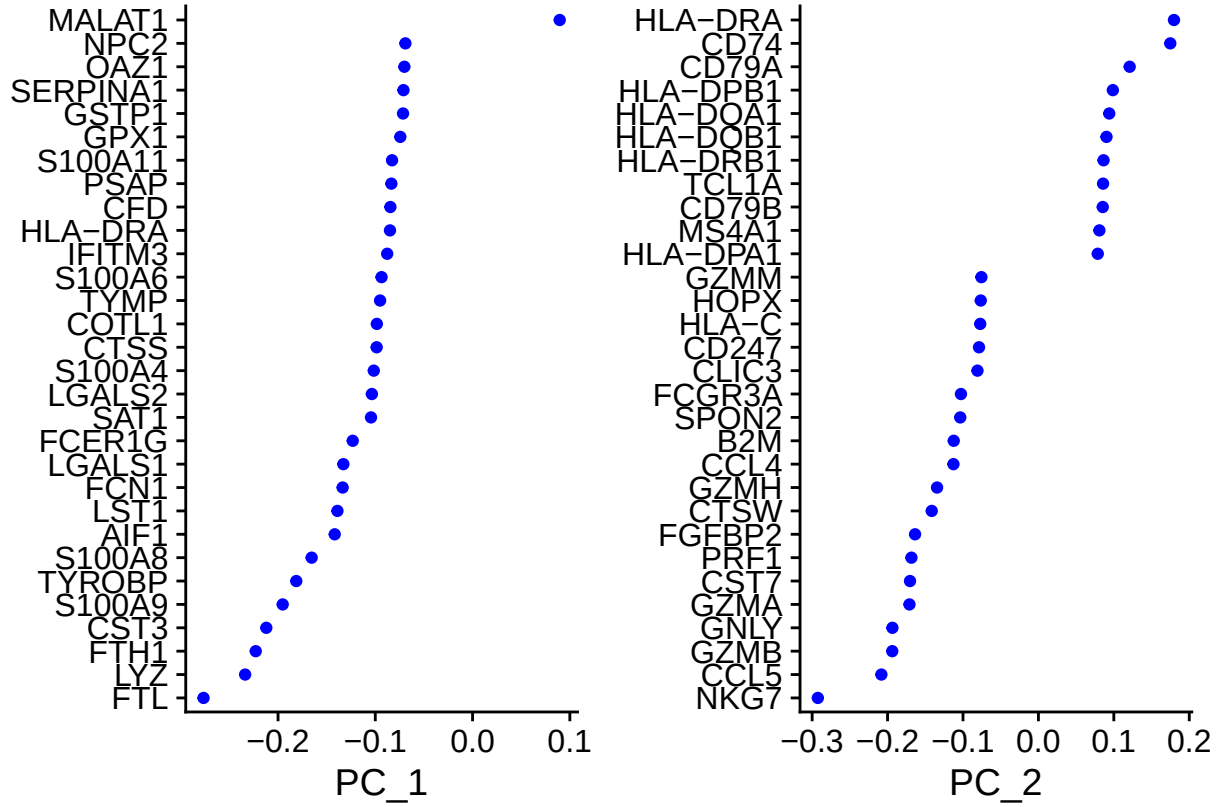
Now we run PCA and print a highlight of the result:

```
pbmc <- RunPCA(pbmc)
# Highlight: top genes in top PC dimensions
print(pbmc[["pca"]], dims = 1:5, nfeatures = 9)
```

```
## PC_ 1
## Positive:  MALAT1, RPS27A, LTB, RPS6, RPS27, RPL13A, RPS3A, RPL3, CCL5
## Negative:  FTL, LYZ, FTH1, CST3, S100A9, TYROBP, S100A8, AIF1, LST1
## PC_ 2
## Positive:  HLA-DRA, CD74, CD79A, HLA-DPB1, HLA-DQA1, HLA-DQB1, HLA-DRB1, TCL1A, CD79B
## Negative:  NKG7, CCL5, GZMB, GNLY, GZMA, CST7, PRF1, FGFBP2, CTSW
## PC_ 3
## Positive:  CD74, HLA-DRA, CD79A, HLA-DPB1, HLA-DQA1, HLA-DRB1, CD79B, HLA-DPA1, HLA-DQB1
## Negative:  S100A8, S100A9, LYZ, FTL, LDHB, RPS12, JUNB, IL7R, CD3D
## PC_ 4
## Positive:  S100A8, S100A9, LYZ, LGALS2, GPX1, CD14, GSTP1, MS4A6A, NKG7
## Negative:  FCGR3A, LST1, FCER1G, AIF1, IFITM3, IFITM2, MS4A7, COTL1, FTH1
## PC_ 5
## Positive:  GNLY, GZMB, FGFBP2, FCGR3A, NKG7, PRF1, TYROBP, FCER1G, LST1
## Negative:  CCL5, GPX1, PPBP, PF4, SDPR, SPARC, GNG11, HIST1H2AC, CD9
```

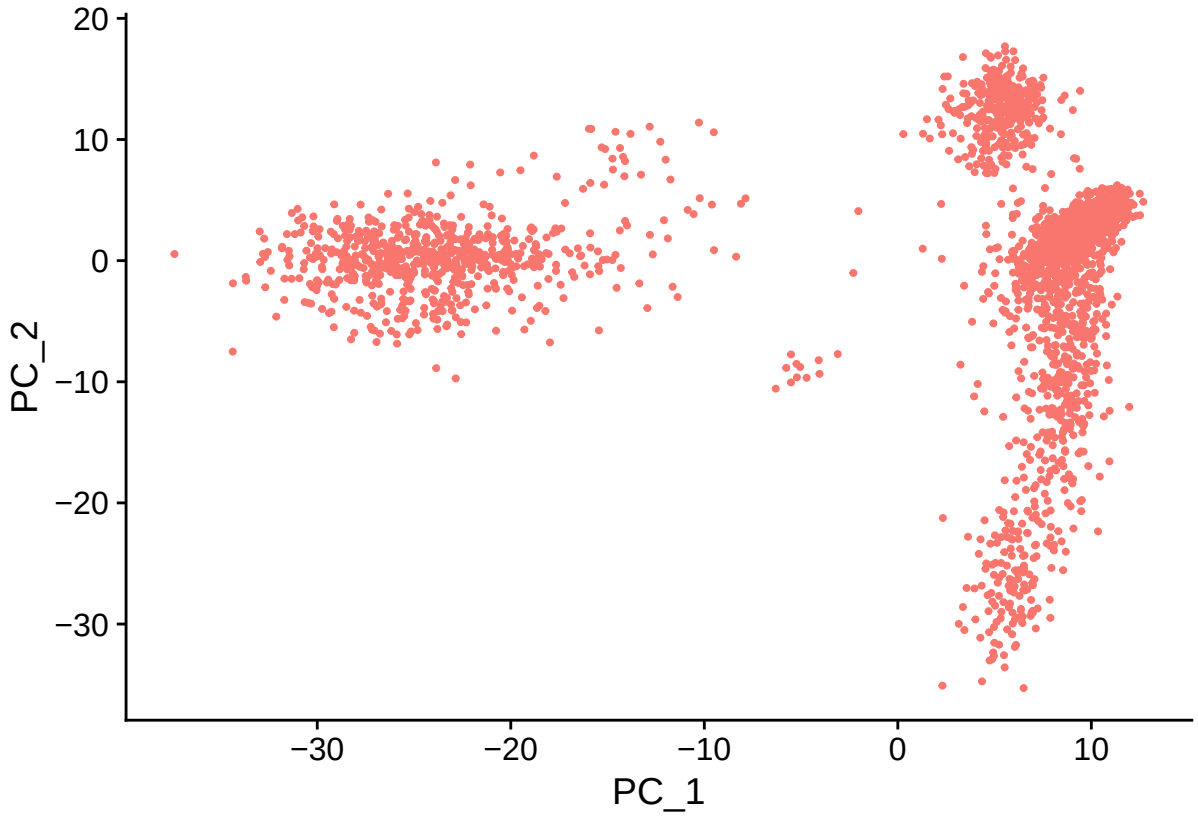
The coefficients of PC's can be visualized too:

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



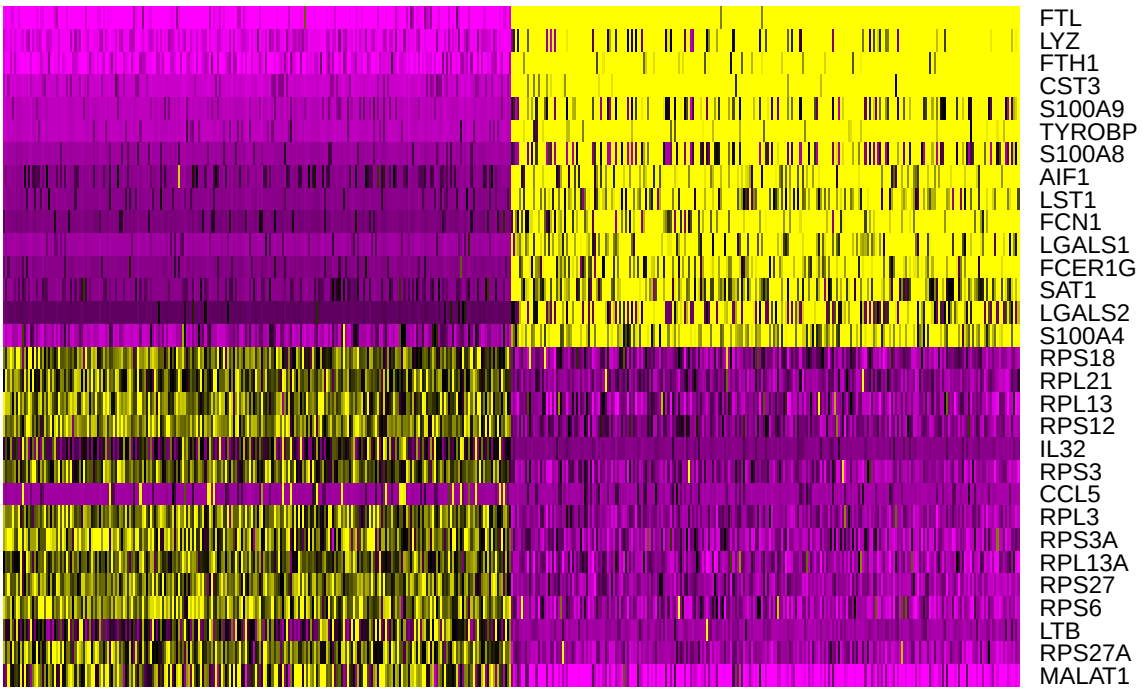
We can also see a standard placement of cells in 2D diagram before clustering

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



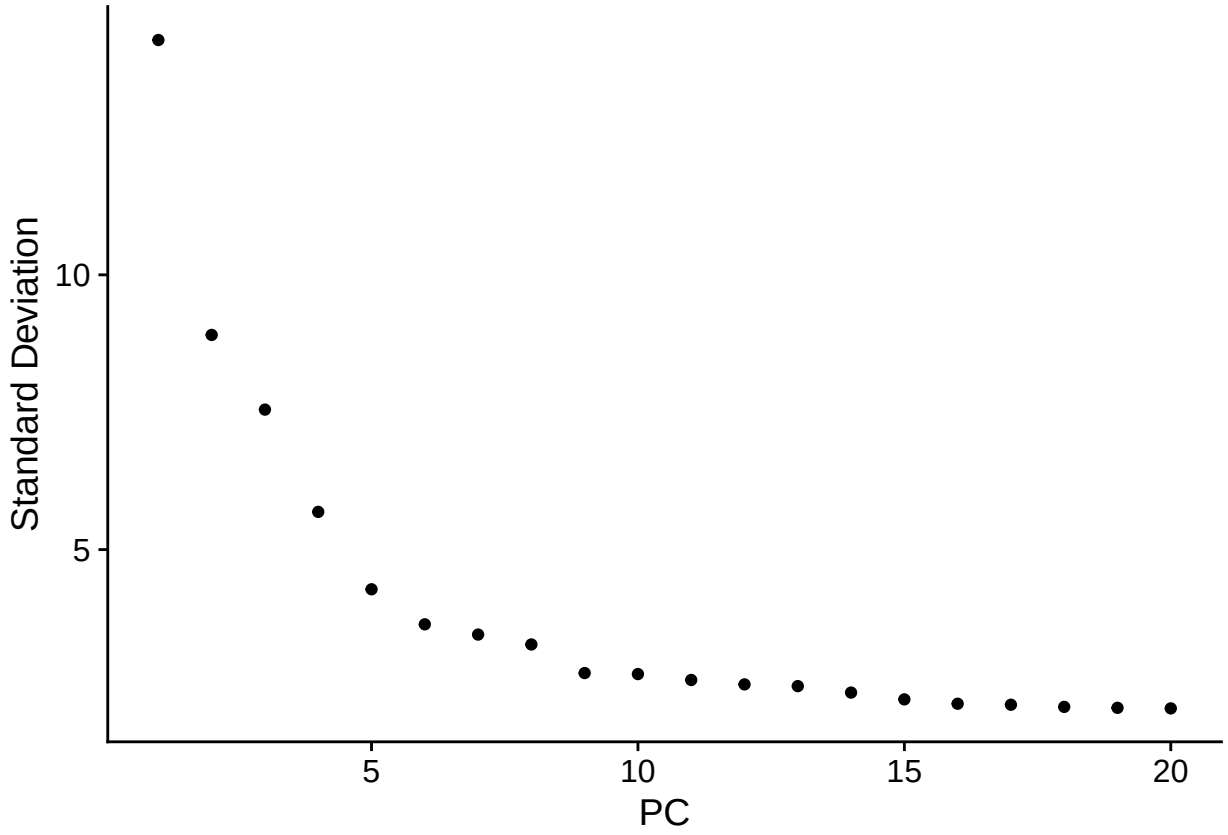
One can also inspect how genes with high coefficient in a PC behave across cells

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



In this data sets, starting from PC_10 it is hard to see distinct patterns, so perhaps these dimensions are not important. While it is even more clear in Elbow Plot below, 9 or 10 does nor need to be the best number, and Seurat designers encourage to compare results when we use more dimensions.

```
ElbowPlot(pbmc)
```



Now it is the time to prepare the main dish: the clusters!

```
pbmc <- FindNeighbors(pbmc, dims = 1:10)
pbmc <- FindClusters(pbmc, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2640
## Number of edges: 88719
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8933
## Number of communities: 10
## Elapsed time: 0 seconds
```

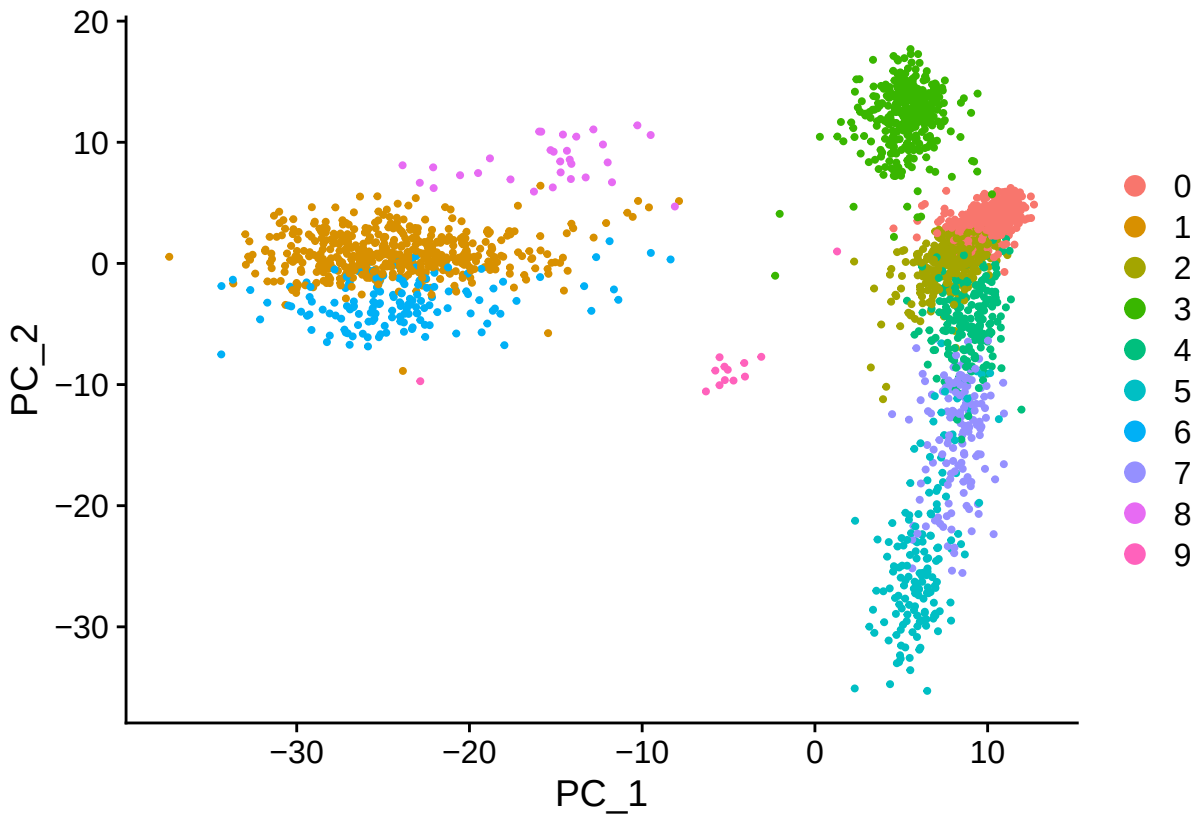
We got clusters numbered from 0, and sizes can be seen simply:

```
table(pbmc$seurat_clusters)
```

```
##
##   0   1   2   3   4   5   6   7   8   9
## 660 492 439 347 205 159 151 144  30  13
```

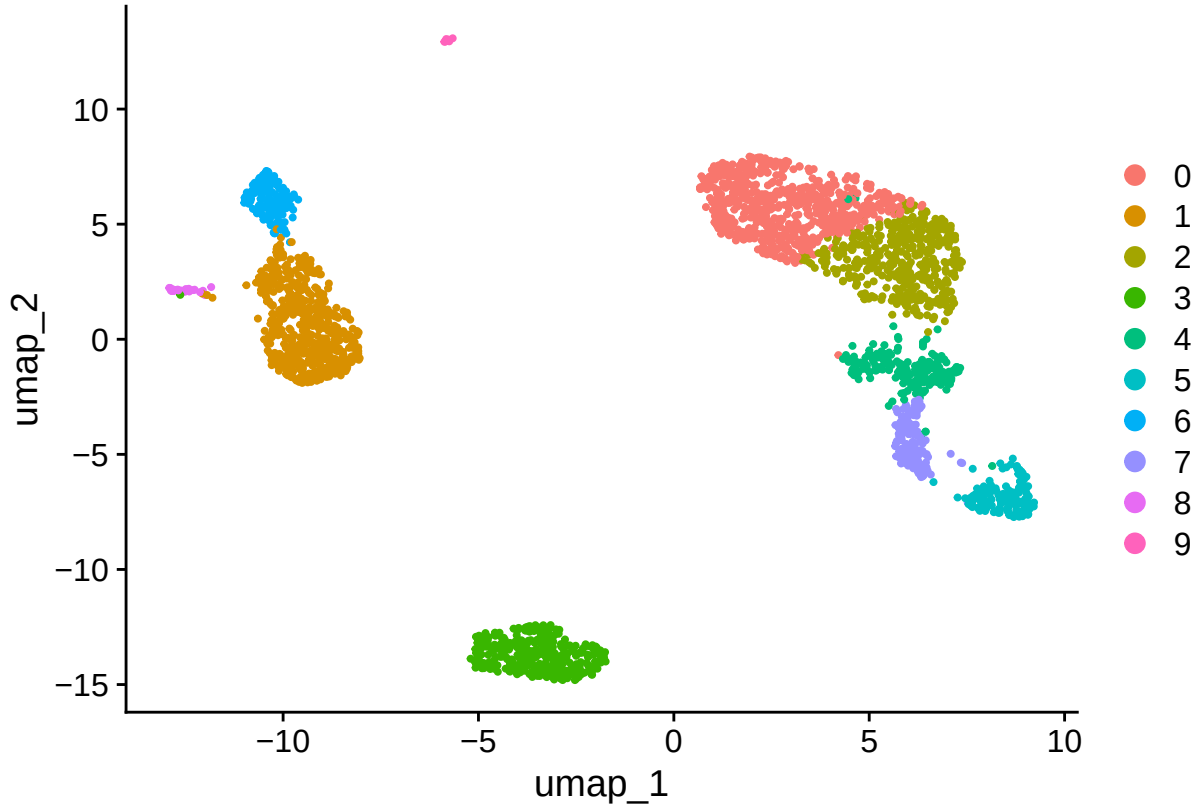
We can also see positions cells according to PC_1 and PC_2

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



but usually a better view is obtained with non-linear coordinates of UMAP. Moreover, after matching clusters with cell types, we can make meanigful legened with the LabelClusters function.

```
pbmc <- RunUMAP(pbmc, dims = 1:10)
DimPlot(pbmc, reduction = "umap")
```



Markers of cluster allow to identify the respective cell type base on prior knowledge. The function FindwMarkers() allows to find differential genes of every individual cluster against the rest of the cells, but it also allows to find differential genes for any pair of groups of cluster. That may be helpful if we want to find a classifier of all types of cells. Of course, it is not productive to compare every pair of cluster groups, but a figures like two previus one hint what is worth checking.

```
# find all markers of cluster 2
cluster2.markers <- FindMarkers(pbmc, ident.1 = 2)
head(cluster2.markers, n = 5)
```

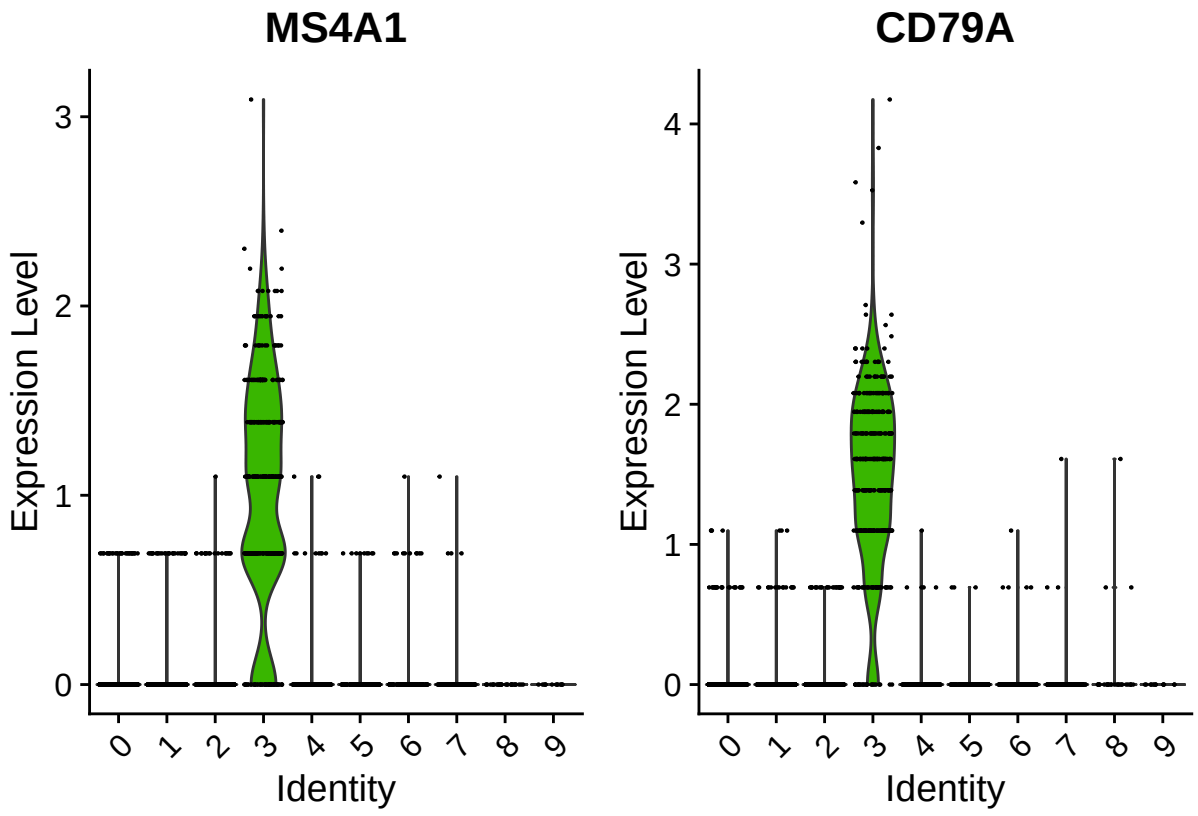
```
##             p_val avg_log2FC pct.1 pct.2    p_val_adj
## IL32 3.582872e-93   1.435411 0.948 0.464 4.486114e-89
## LTB  3.747914e-84   1.452143 0.970 0.647 4.692764e-80
## CD3D 6.135673e-73   1.162257 0.902 0.435 7.682476e-69
## IL7R 1.567187e-55   1.274438 0.711 0.329 1.962275e-51
## AQP3 3.894587e-52   2.253397 0.405 0.118 4.876413e-48
```

```
cluster5.markers <- FindMarkers(pbmc,ident.1 = 5, ident.2 = c(1,7))
head(cluster5.markers, n = 5)
```

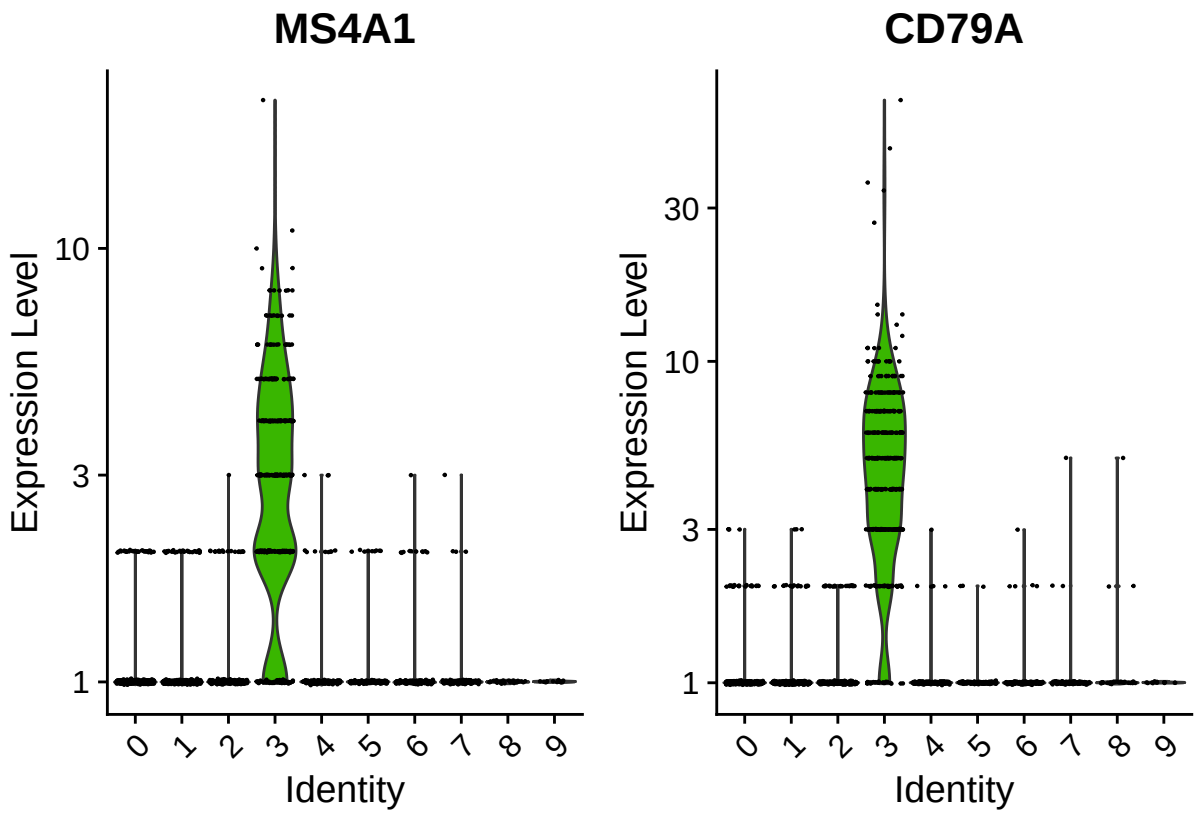
```
##              p_val avg_log2FC pct.1 pct.2     p_val_adj
## GNLY  6.988482e-108   5.131040 0.969 0.165 8.750279e-104
## GZMB  9.093541e-102   4.449584 0.950 0.157  1.138602e-97
## PRF1   1.333280e-96   4.177157 0.937 0.162  1.669400e-92
## CD7    7.863283e-92   4.216719 0.862 0.118  9.845616e-88
## CD247  2.504088e-84   3.981543 0.843 0.124  3.135369e-80
```

Some markers are common to more than one cluster, and some are "negative", so it is interesting how they distribute among cluster. We can find it using three approaches: (a) violin plot of expression levels in every cluster (b) plot of presence in UMAP diagram (c) table of summary statistic of presence in clusters. The last approach is best when we have many clusters, e.g. 25 or more but the first two may allow for quick insights. Method (c) may require a short script, but (a) and (b) have Seurat functions.

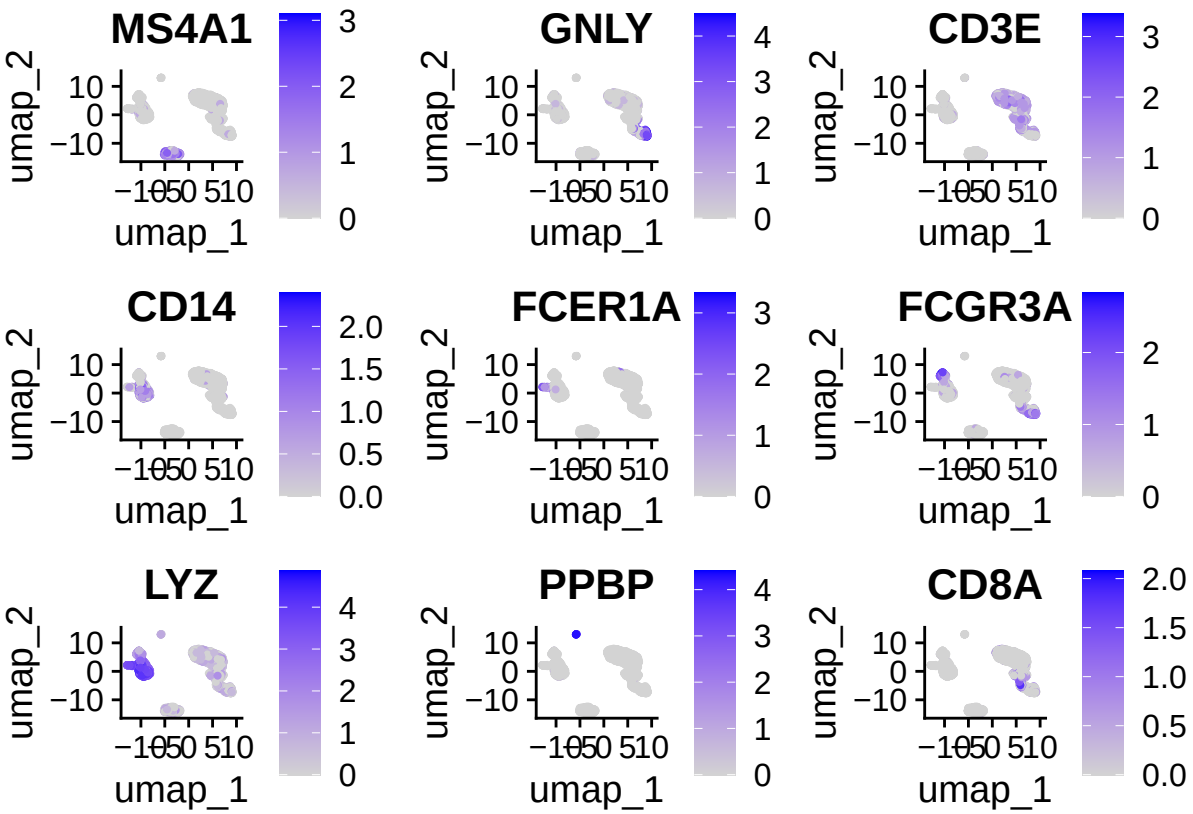
```
VlnPlot(pbmc, features = c("MS4A1", "CD79A"))
```




```
VlnPlot(pbmc, features = c("MS4A1", "CD79A"), layer = "counts", log = TRUE)
```

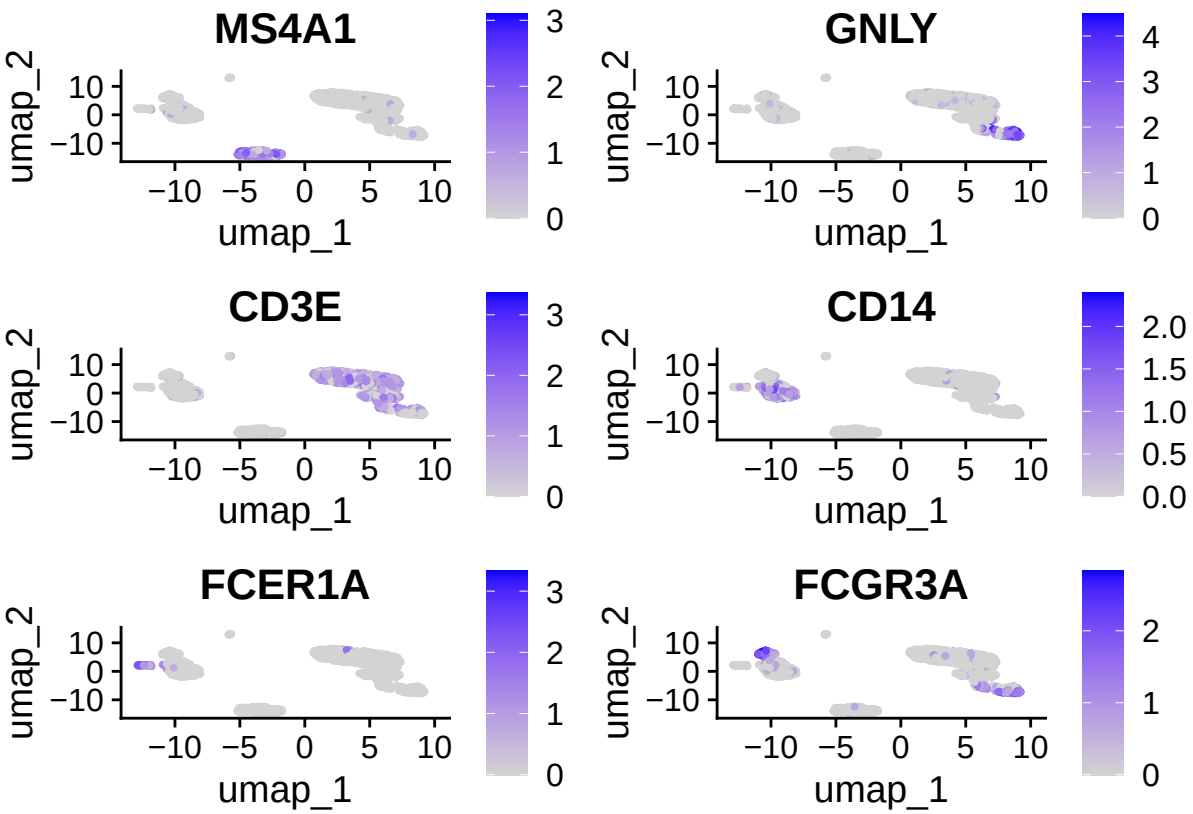



```
FeaturePlot(pbmc, features = c("MS4A1", "GNLY", "CD3E", "CD14", "FCER1A",
                               "FCGR3A", "LYZ", "PPBP","CD8A"))
```



The last plot has too many facets, we may inspect markers in smaller batches:

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

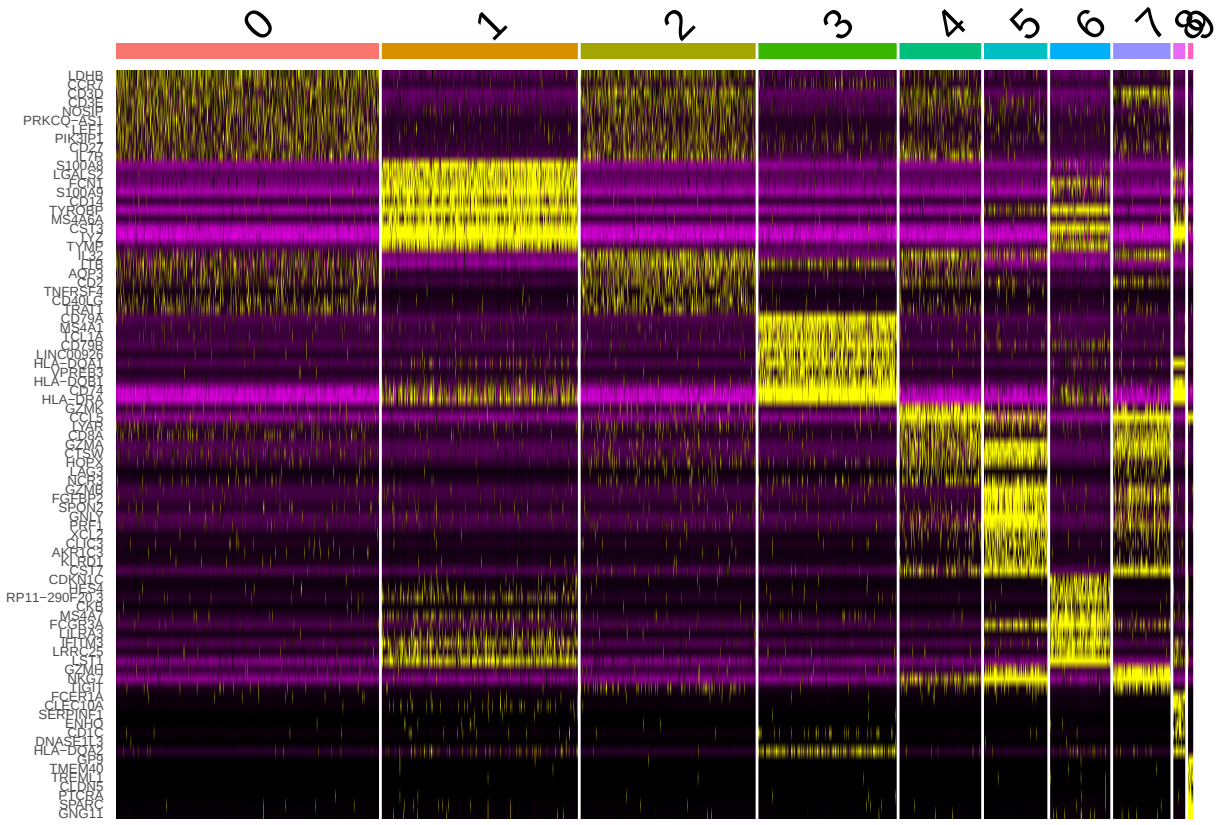


Finally, we can have a heatmap of top markers. To read gene names, see it in a separate window and stretch vertically.

```
# find markers for every cluster compared to all remaining cells,
# report only the positive ones
pbmc.markers <- FindAllMarkers(pbmc, only.pos = TRUE)
pbmc.markers %>%
    group_by(cluster) %>%
    dplyr::filter(avg_log2FC > 1 & -log10(p_val_adj) > 5)
```

```
## # A tibble: 2,056 x 7
## # Groups:   cluster [10]
##        p_val avg_log2FC pct.1 pct.2 p_val_adj cluster gene
##        <dbl>      <dbl> <dbl> <dbl>     <dbl> <fct>   <chr>
##  1 8.92e-134       1.32 0.953 0.564 1.12e-129 0       LDHB
##  2 2.83e- 99       2.53 0.468 0.102 3.55e- 95 0       CCR7
##  3 6.63e- 71       1.00 0.856 0.398 8.30e- 67 0       CD3D
##  4 5.93e- 69       1.11 0.771 0.373 7.42e- 65 0       CD3E
##  5 1.39e- 64       1.29 0.679 0.33  1.74e- 60 0       NOSIP
##  6 8.07e- 58       2.08 0.365 0.102 1.01e- 53 0       PRKCQ-AS1
##  7 8.91e- 56       1.99 0.359 0.099 1.11e- 51 0       LEF1
##  8 1.38e- 49       1.51 0.461 0.178 1.73e- 45 0       PIK3IP1
##  9 2.23e- 48       1.21 0.452 0.165 2.79e- 44 0       CD27
## 10 4.67e- 47       1.12 0.638 0.311 5.85e- 43 0       IL7R
## # i 2,046 more rows
```

```
pbmc.markers %>%
    group_by(cluster) %>%
    dplyr::filter(avg_log2FC > 1) %>%
    slice_head(n = 10) %>%
    ungroup() -> top10
DoHeatmap(pbmc, features = top10$gene) + NoLegend() +
  theme(axis.text.y = element_text(size = 5))
```



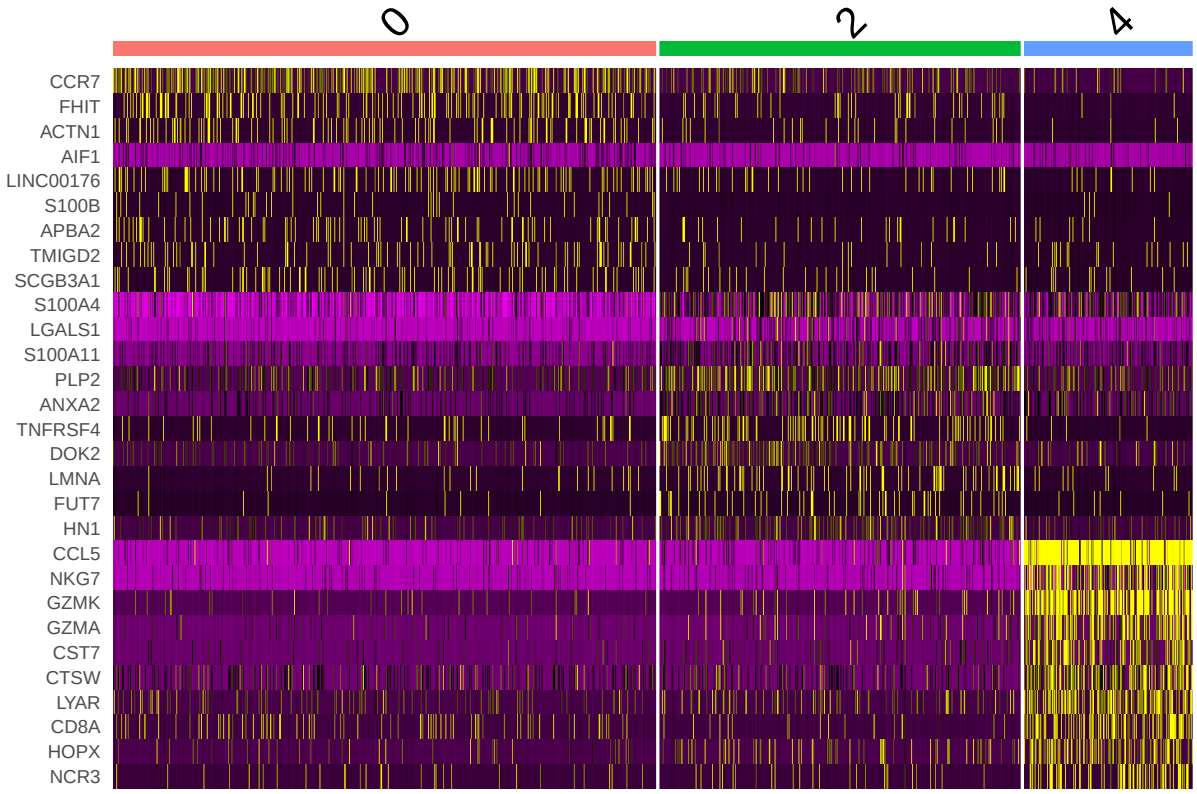
We see that top 10 markers poorly differenciate between clusters 0, 2 and 4, so we can search for alternatives using powerful subset() function of Seurat.

```
pbmc_024 <- subset(pbmc, idents = c(0, 2, 4))
pbmc_024.markers <- FindAllMarkers(pbmc_024, only.pos = TRUE)
pbmc_024.markers %>%
    group_by(cluster) %>%
    dplyr::filter(avg_log2FC > 1 & -log10(p_val_adj) > 5)
```

```
## # A tibble: 54 x 7
## # Groups:   cluster [3]
##       p_val avg_log2FC pct.1 pct.2 p_val_adj cluster gene
##       <dbl>      <dbl> <dbl> <dbl>     <dbl> <fct>   <chr>
##  1 2.61e-28       1.56 0.468 0.194  3.26e-24 0       CCR7
##  2 8.40e-14       1.82 0.211 0.068  1.05e- 9 0       FHIT
##  3 2.25e-11       1.83 0.159 0.047  2.81e- 7 0       ACTN1
##  4 2.73e-10       1.27 0.279 0.135  3.41e- 6 0       AIF1
##  5 1.99e-48       1.10 0.945 0.76   2.49e-44 2       S100A4
```

```
##  6 1.43e-26       1.88 0.453 0.187  1.79e-22 2       LGALS1
##  7 7.68e-24       1.25 0.592 0.331  9.61e-20 2       S100A11
##  8 4.98e-21       1.16 0.544 0.299  6.24e-17 2       PLP2
##  9 8.97e-21       1.34 0.46  0.22   1.12e-16 2       ANXA2
## 10 7.25e-16       2.02 0.203 0.058  9.07e-12 2       TNFRSF4
## # i 44 more rows
```

```
pbmc_024.markers %>%
    group_by(cluster) %>%
    dplyr::filter(avg_log2FC > 1) %>%
    slice_head(n = 10) %>%
    ungroup() -> top10
DoHeatmap(pbmc_024, features = top10$gene) + NoLegend() +
  theme(axis.text.y = element_text(size = 7))
```



Now cluster 4 is clearly differenciated, and for 4 vs 6 we can repeat the aproach.