scRNA-seq analysis

Shir ohayon & Ye'ela granot & Reut lev

2023-05-23

in this exercise we chose to perform analysis to lung cells isolated from Adult Mouse organism from 10X Genomics. the cells were sequenced using Illumina NovaSeq 6000. let's load the dataset and see its sturcture.

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

# Load the dataset
mouseLung.data <- Read10X(data.dir = "filtered_feature_bc_matrix")
# Initialize the Seurat object with the raw (non-normalized data).
mouseLung <- CreateSeuratObject(counts = mouseLung.data, project = "lungCell",
                                min.cells = 3, min.features = 200)
mouseLung
```

```
## An object of class Seurat
## 22418 features across 7779 samples within 1 assay
## Active assay: RNA (22418 features, 0 variable features)
```

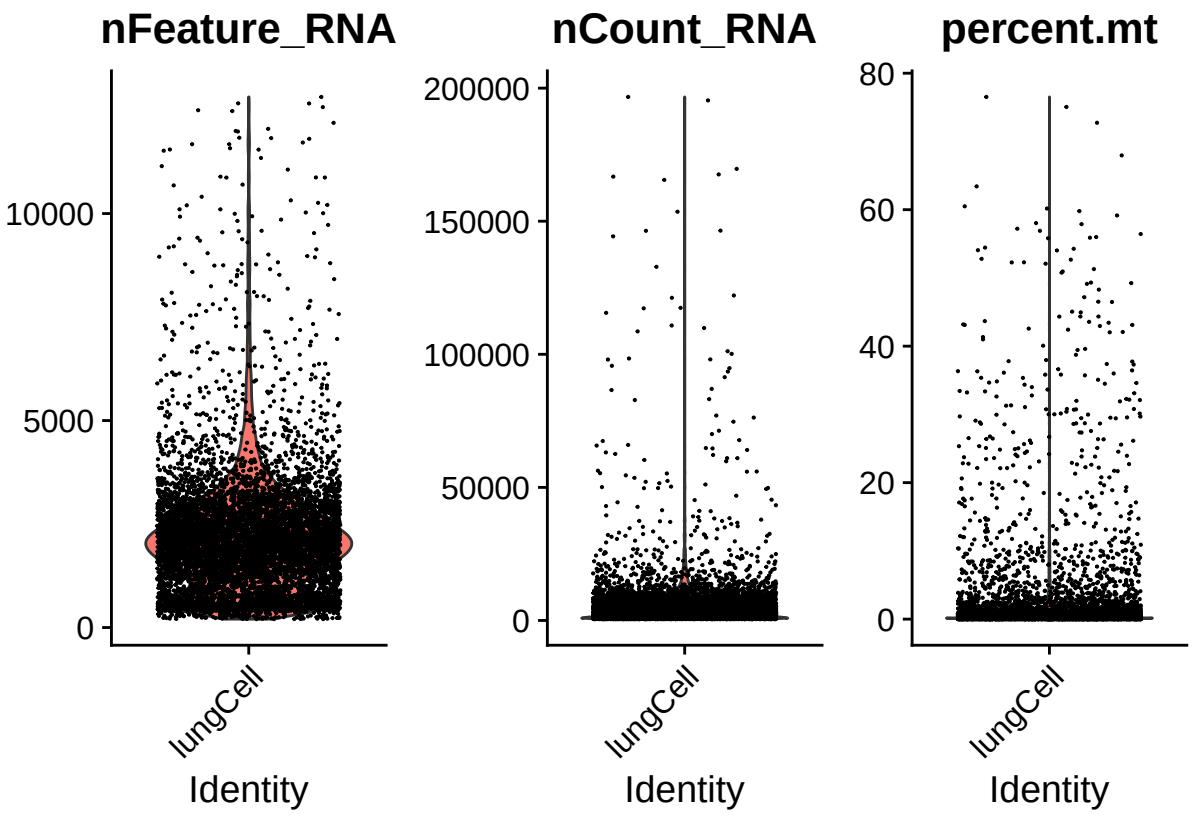
as it can be seen our dataset contains 22418 genes across 7779 lung cells. We'll start performing the pre-processing of the data. we would like to count for each cell how many mitochondrial genes it has so we could detect if the cell is in good health or dying.

```
mouseLung[["percent.mt"]] <- PercentageFeatureSet(mouseLung, pattern = "^mt-")
head(mouseLung@meta.data, 5)
```

```
##                     orig.ident nCount_RNA nFeature_RNA percent.mt
## AAACCCAAGAAAGCGA-1   lungCell       7031         2915 0.05689091
## AAACCCAAGACATCAA-1   lungCell       5787         2720 0.20736133
## AAACCCAAGGCCTGAA-1   lungCell       7299         2865 0.04110152
## AAACCCAAGTCAGGGT-1   lungCell       4340         2135 0.06912442
## AAACCCACATACAGAA-1   lungCell       6955         2827 0.18691589
```

After adding the mitochondrial genes count feature we'll visualize QC metrics as a violin plot.

```
VlnPlot(mouseLung, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```



from the violin graph we see that cells should be filtered in this following way: +have unique feature counts less than 7,000 +have <5% mitochondrial counts.

```
mouseLung <- subset(mouseLung, subset = nFeature_RNA < 7000 & percent.mt < 5)
mouseLung
```

```
## An object of class Seurat
## 22418 features across 7098 samples within 1 assay
## Active assay: RNA (22418 features, 0 variable features)
```

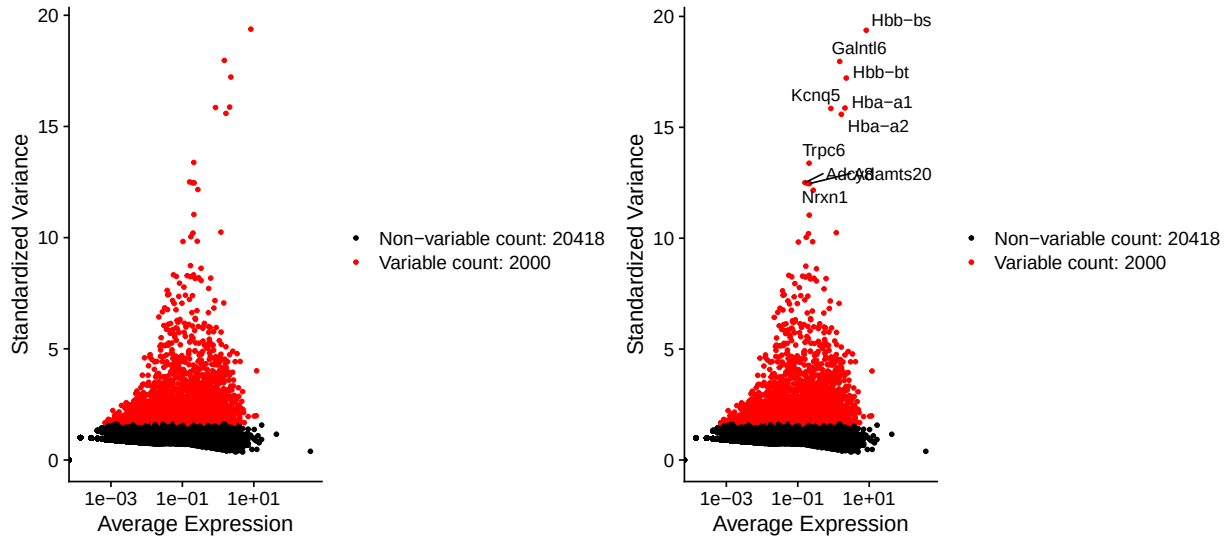
now we cleaned 681 cells in the pre-processing phase. Now we normalize the data values for each cell.

```
mouseLung <- NormalizeData(mouseLung)
```

Next, we apply identification of genes that exhibit high variation in the dataset.

```
mouseLung <- FindVariableFeatures(mouseLung, selection.method = 'vst', nfeatures = 2000)
# Identify the 10 most highly variable genes
top10 <- head(VariableFeatures(mouseLung), 10)
# plot variable features with and without labels
plot1 <- VariableFeaturePlot(mouseLung)
plot2 <- LabelPoints(plot = plot1, points = top10, repel = TRUE)
plot1 + plot2
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
## Transformation introduced infinite values in continuous x-axis
```



Now we will perform general data scaling so the different expressions level in each gene won't dominant the others genes. it performed by setting the average values of each gene to 0 and variance to 1.

```
all.genes <- rownames(mouseLung)
mouseLung <- ScaleData(mouseLung, features = all.genes)
```

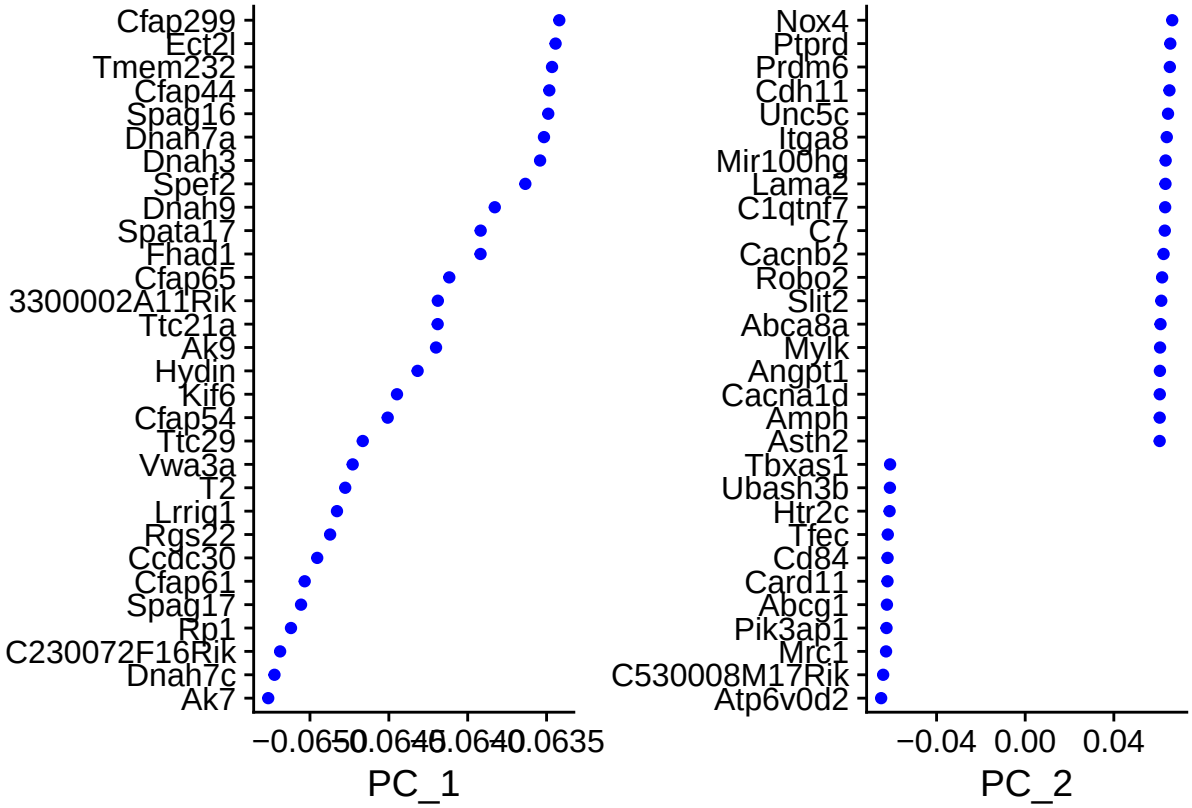
Until now we have done pre-processing, feature selection and then we normalized and scaled the data. In this pahse we would like to perform PCA. Since we have a lot of features this will allow us to have a better visualization of the data for the analysis. we'll start using linear PCA.

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

```
print(mouseLung[['pca']], dims = 1:5, nfeatures = 5)
```

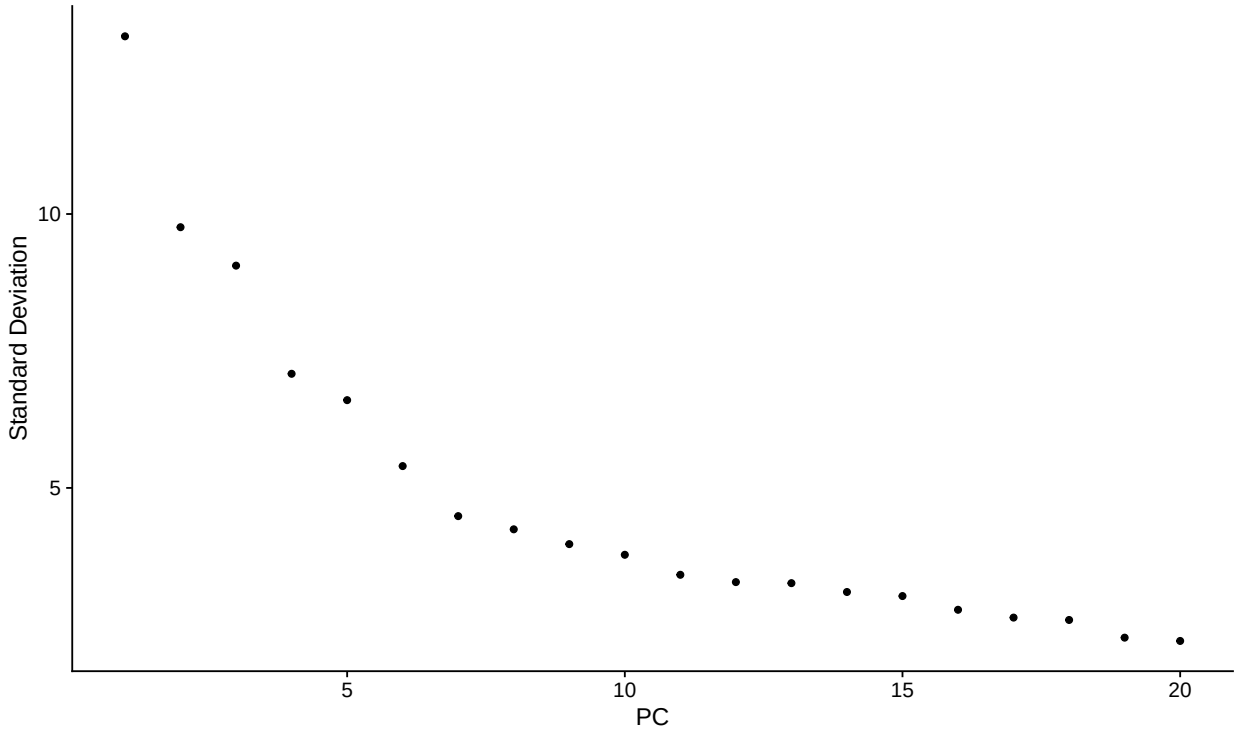
```
## PC_ 1
## Positive:  Dock4, Zeb2, Arhgap6, Prex2, Qk
## Negative:  Ak7, Dnah7c, C230072F16Rik, Rp1, Spag17
## PC_ 2
## Positive:  Nox4, Ptprd, Prdm6, Cdh11, Unc5c
## Negative:  Atp6v0d2, C530008M17Rik, Mrc1, Pik3ap1, Abcg1
## PC_ 3
## Positive:  Pid1, Adarb1, Zeb2, Lrp1, Slc8a1
## Negative:  Lmo7, Magi3, Magi1, Pde8b, Col4a3
## PC_ 4
## Positive:  Col4a3, Col4a4, Rtkn2, Lama3, Sema3a
## Negative:  Cyyr1, Calcrl, Tspan7, Fendrr, Tspan18
## PC_ 5
## Positive:  Pakap.1, Rasgrf2, Rtkn2, Cav1, Scel
## Negative:  Aox3, 5330417C22Rik, Cbr2, Col23a1, Cadm2
```

```
VizDimLoadings(mouseLung, dims = 1:2, reduction = 'pca')
```



for determining how many components of the dataset should we choose to include we will use the elbow method.

```
ElbowPlot(mouseLung)
```



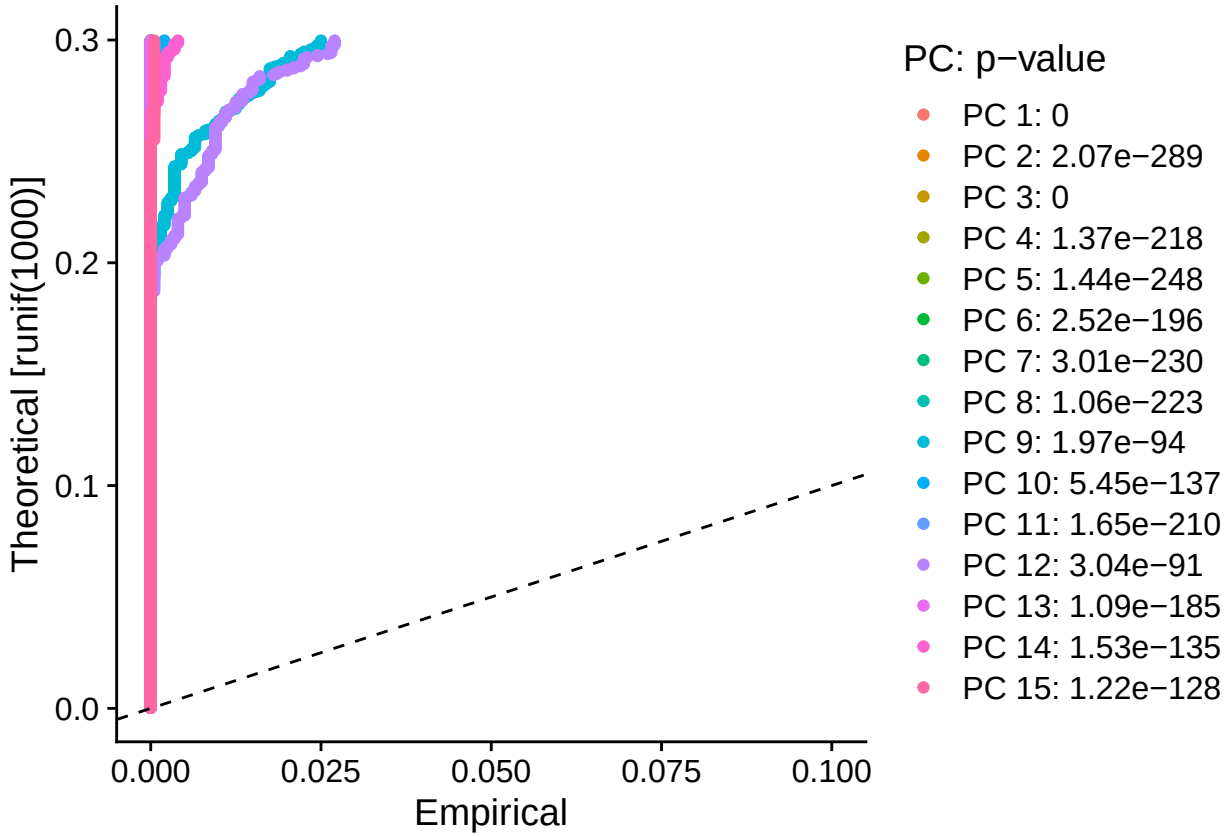
it can be observed that we get 'elbow' around PC6 which means that the majority of true signal is captured in the first 6 PCs, so this will be the cutoff we'll take.

To be sure, let's look at the reliability of the information in another way

```
mouseLung <- JackStraw(mouseLung, num.replicate = 100)
mouseLung <- ScoreJackStraw(mouseLung, dims = 1:20)
```

```
JackStrawPlot(mouseLung, dims = 1:15)
```

```
## Warning: Removed 21000 rows containing missing values (`geom_point()`).
```



It seems that the first 6 appear with a low P VALUE so that will be the cutoff we will take.

Let's cluster the data into different interconnected 'communities' using the KNN algorithm.

```
mouseLung <- FindNeighbors(mouseLung, dims = 1:6)
```

```
## Computing nearest neighbor graph

## Computing SNN
```

```
mouseLung <- FindClusters(mouseLung, resolution = 0.4)
```

```
head(Idents(mouseLung), 5)
```

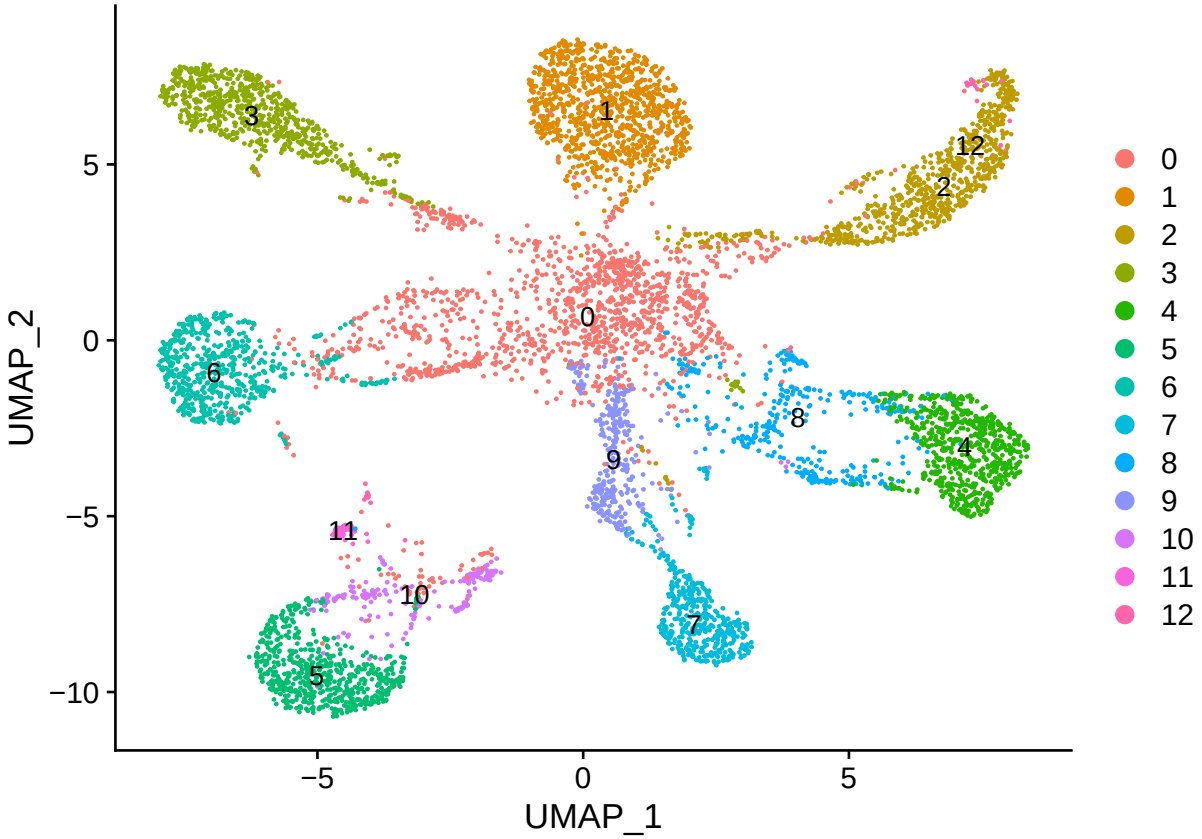
```
## AAACCCAAGAAAGCGA-1 AAACCCAAGACATCAA-1 AAACCCAAGGCCTGAA-1 AAACCCAAGTCAGGGT-1
##                 12                  6                  6                  2
## AAACCCACATACAGAA-1
##                  5
## Levels: 0 1 2 3 4 5 6 7 8 9 10 11 12
```

Next, we'll try to perform non-linear reduction method to visualize the clustering using UMAP.

```
mouseLung <- RunUMAP(mouseLung, dims = 1:6)
```

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

```
umap_dimplot <- DimPlot(mouseLung, reduction = 'umap', label=TRUE)
umap_dimplot
```



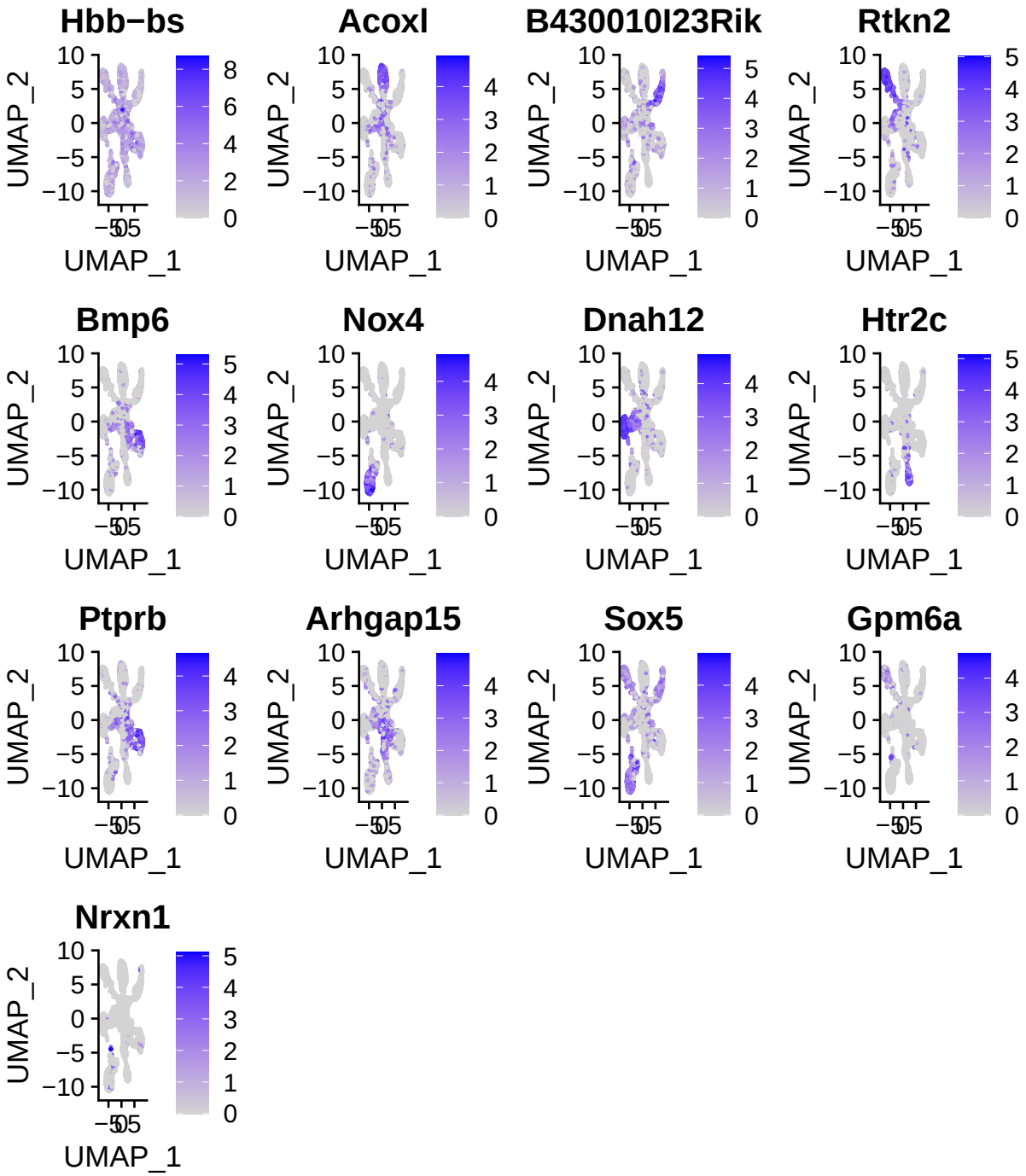
Now we will find markers that define clusters via positive differential expression. the markers for every cluster will be compared to all remaining cells.

```
mouseLung.markers <- FindAllMarkers(mouseLung, only.pos = TRUE, min.pct = 0.75)

mouseLung.markers %>% group_by(cluster) %>% slice_max(n = 2, order_by = avg_log2FC)
```

To visualize the features of the differential expression we get, we will use FeaturePlot.

```
FeaturePlot(mouseLung, features = c("Hbb-bs","Acoxl","B430010I23Rik","Rtkn2","Bmp6",
                                    "Nox4","Dnah12","Htr2c","Ptprb","Arhgap15","Sox5","Gpm6a","Nrxn1"))
```



As you can see that there are several genes that are really expressed only in specific clusters such as Nox4 or Dnah12, while there are clusters in which no unique expression of genes is found for them such as Hbb-bs whose most differentially expressed gene is actually expressed in other clusters as well. In addition, you can also see genes that are expressed in a very limited number of cells, for example Nrxn1 Gpm6a

We would like to identify the cell types according to the level of different expression of the genes in each cluster Since we are working with mouse data we will use MouseRNAseqData

```
#install.packages("devtools")   # unnecessary if you have it already
#devtools::install_github("Bioconductor/BiocManager", ref="ghost-binary-repo")
#BiocManager::install("celldex")
#BiocManager::install("SingleR")
#BiocManager::install("SingleCellExperiment")
```

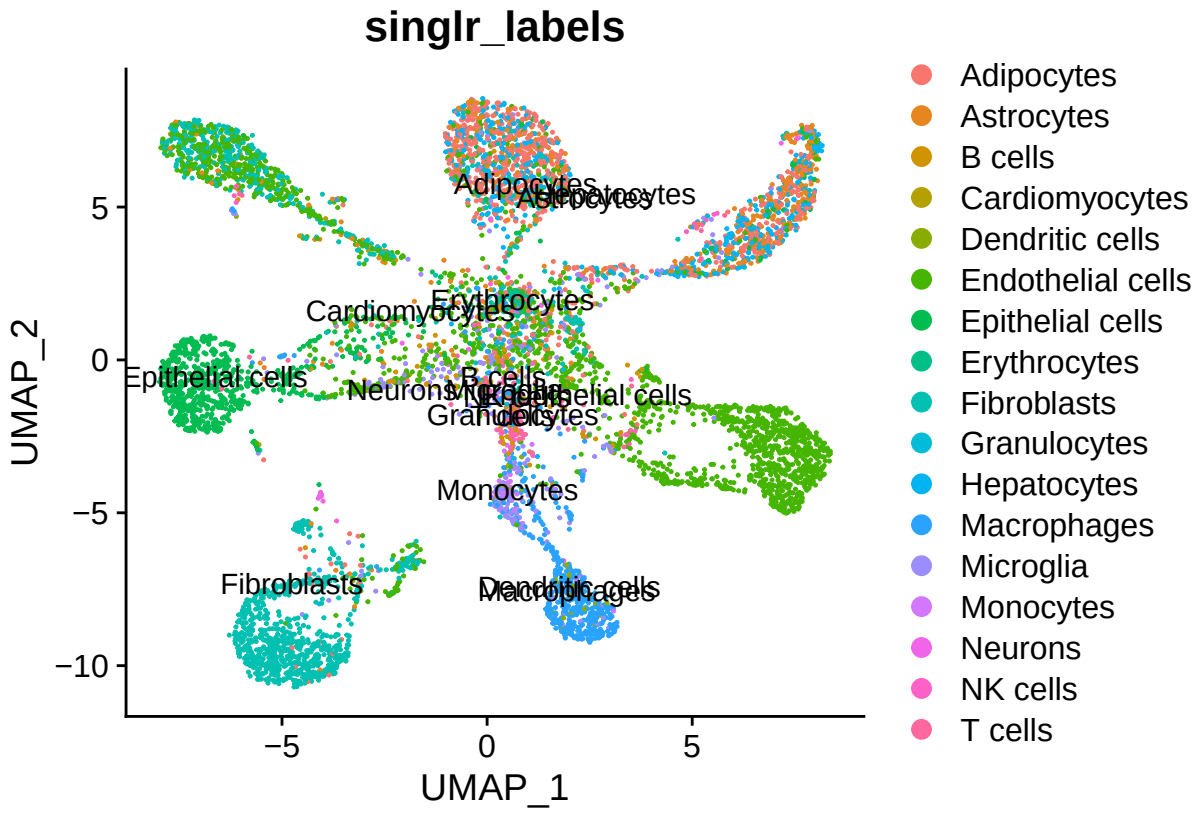
```
library(SingleR)
library(celldex)

ref <- celldex::MouseRNAseqData()
```

```
# dataframe of the mapped cell labels
results <- SingleR(test = as.SingleCellExperiment(mouseLung), ref = ref, labels = ref$label.main)
```

```
mouseLung$singlr_labels <- results$labels
```

```
DimPlot(mouseLung, reduction = 'umap', group.by = 'singlr_labels', label = TRUE)
```



Now you can see the different cell types It can be seen that in the lungs the different types of cells are often mixed and yet there are some types that cluster together for example - Fibroblast, Monocytes and Macrophages

In this exercise we experimented with Single cell gene analysis First we chose the database - 5k Adult Mouse Lung Nuclei Isolated with Chromium Nuclei Isolation Kit To work with the Data we had to clean the data, for example we filtered out mitochondrial genes and filtered out cells that had high variation in gene

expression. In addition, as a preliminary step to the analysis, we normalized the expression of the genes to values between 0 and 1. The PCA results - we chose the best PCs according to the p-value and the elbow method, We found 6 different clusters. We wanted to characterize each cluster according to the main gene expressed in it. We ran a function on each cluster that returns the main marker. We asked that the minimum percentage that the gene be expressed in the cluster be 0.75. Finally, we wanted to name the different cells according to gene expression levels. Using the SingleR celldex libraries