singlecell_Seurat_pipeline

2023-02-12

Single-cell RNA-Seq analysis pipeline (GEO dataset)

This is a pipeline to analyze single-cell RNA Seq data from GEO. In this script the single cell RNA-Seq results from this paper is regenerated: https://pubmed.ncbi.nlm.nih.gov/34956864/ Title: Bulk and Single-Cell Profiling of Breast Tumors Identifies TREM-1 as a Dominant Immune Suppressive Marker Associated With Poor Outcomes

Initial library loading

```
library(Seurat)
library(tidyverse)
library(GEOquery)
```

0. Fetch data from GEO

```
file <- getGEOSuppFiles("GSE188600")
untar("GSE188600/GSE188600_RAW.tar", exdir = 'data/')
```

Note: No need to fetch the data if it is already created

1. Create the counts matrix

```
wd = getwd()
files = list.files(path = paste0(wd, '/data'), full.names = FALSE, recursive = FALSE)

mtx.cnts <- ReadMtx(mtx = paste0('data/', files[3]),
                features = paste0('data/', files[2]),
                cells = paste0('data/', files[1]))
```

Find the files in data folder (i.e., barcodes, features, matrix)

2. Create a seurat object

```
Seurat.obj <- CreateSeuratObject(counts = mtx.cnts)

Seurat.obj
```

```
## An object of class Seurat
## 33694 features across 770 samples within 1 assay
## Active assay: RNA (33694 features, 0 variable features)
```

3. QC and filtering

View Seurat object meta data

```
##                       orig.ident nCount_RNA nFeature_RNA
## AAACCTGTCGTGACAT-1 SeuratProject       3057         1301
## AAAGATGTCGCCTGTT-1 SeuratProject       7981         1903
## AAAGCAAAGAGCTATA-1 SeuratProject       1814          876
## AAAGCAACAAGTAGTA-1 SeuratProject       2749         1325
## AAAGTAGCATGCTAGT-1 SeuratProject       5421         1406
## AAAGTAGGTCCAGTAT-1 SeuratProject       1957         1017
## AAAGTAGGTTGTACAC-1 SeuratProject      13236         2703
## AAAGTAGTCACAGTAC-1 SeuratProject       5116         1516
## AAATGCCAGCCGATTT-1 SeuratProject       6864         1774
## AACACGTCAAGCCCAC-1 SeuratProject      18283         3069
```

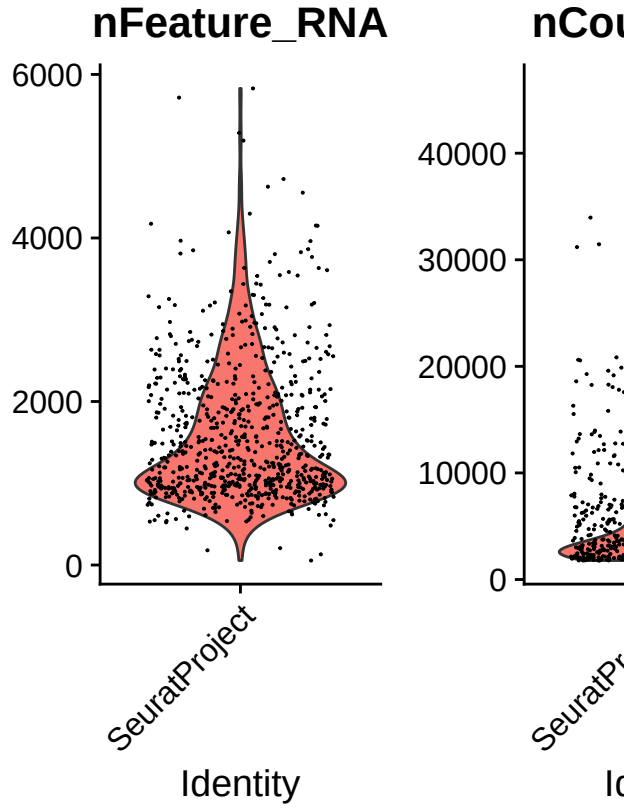
3.1 Calculate mitochondrial percentage

```
Seurat.obj$mito.prct <- PercentageFeatureSet(Seurat.obj, pattern = '^MT-')
```

High percentage shows bad quality

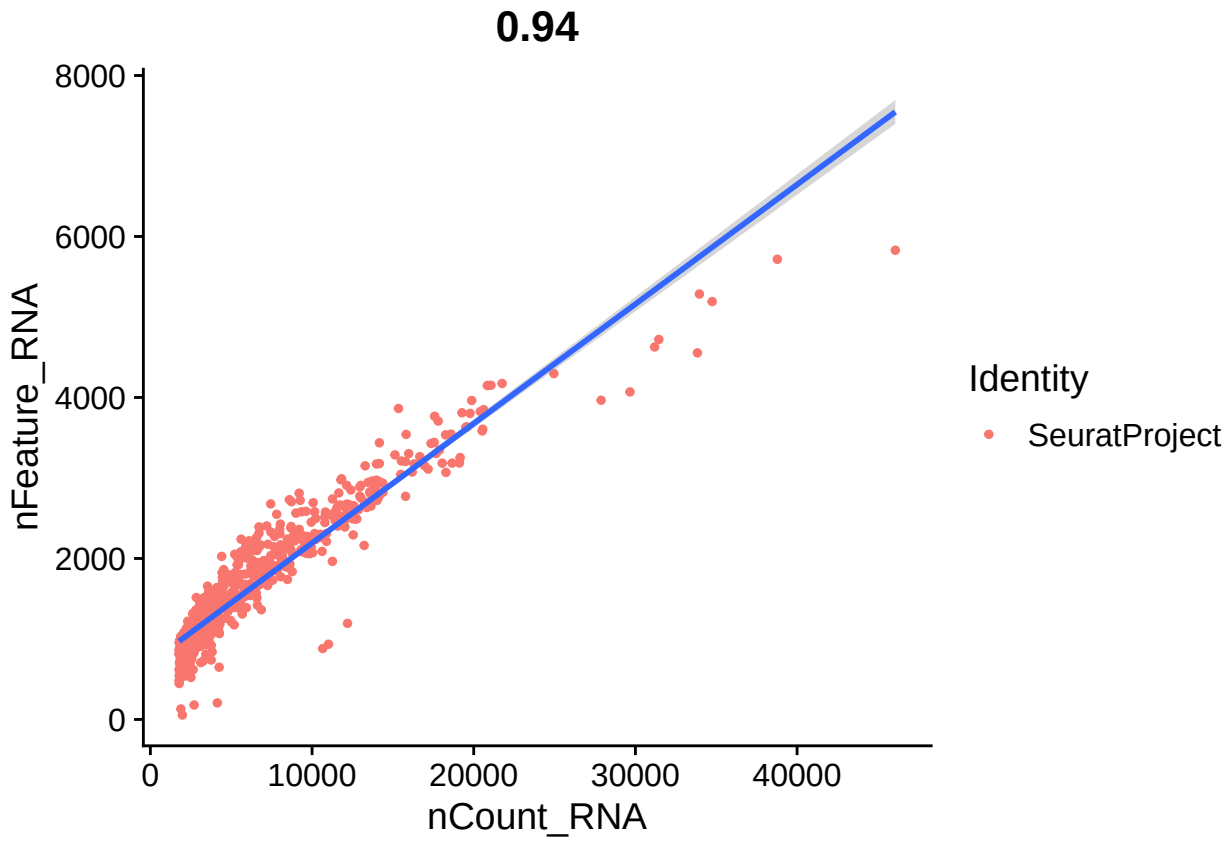
3.2 Explore QC

```
VlnPlot(Seurat.obj, features = c("nFeature_RNA", "nCount_RNA", "mito.prct"), ncol = 3)
```



Plot feature and RNA counts and mitochondial percentage

```
FeatureScatter(Seurat.obj, feature1 = "nCount_RNA", feature2 = "nFeature_RNA") +
  geom_smooth(method = 'lm')
```



3.3 Filter cells

```
Seurat.obj.filt <- subset(Seurat.obj, subset = nCount_RNA >800 &
                            nFeature_RNA > 500 &
                              mito.prct < 10)
```

more than 800 RNA count and more than 500 genes and less than 10 mitochondrial%

4. Finding Variable genes

note: This data set contains one sample, if more than 1 sample was used correction for potential batch effects should be considered

4.1 Normalize data

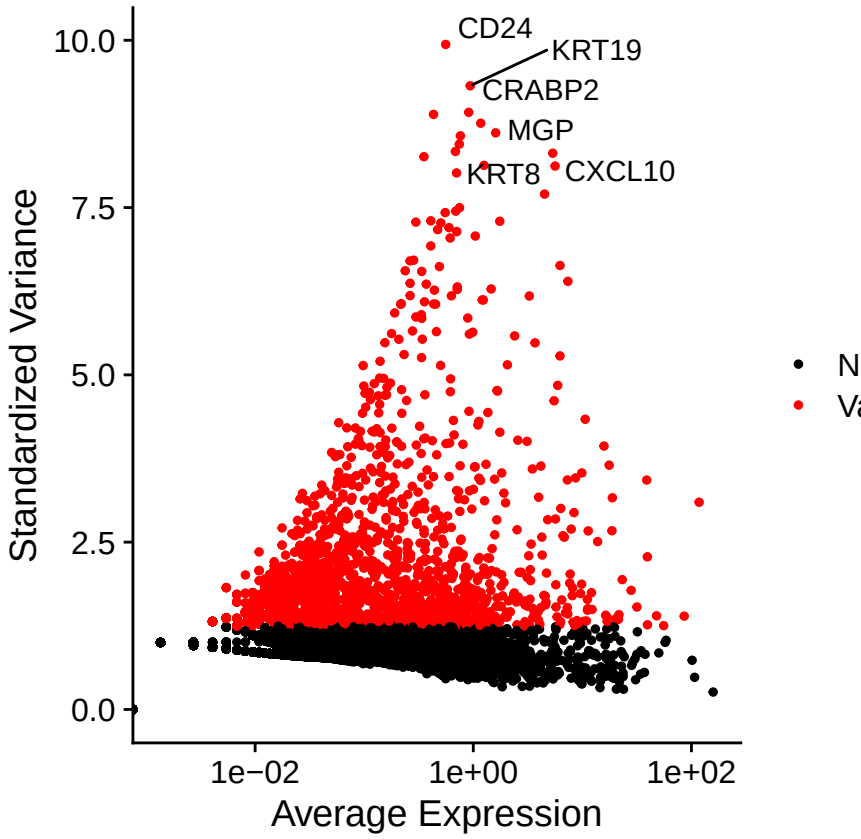
```
Seurat.obj.filt <- NormalizeData(object = Seurat.obj.filt)
```

4.2 Find variable genes

```
Seurat.obj.filt <- FindVariableFeatures(object = Seurat.obj.filt, selection.method = 'vst', nfeatures =

top10 <- head(VariableFeatures(Seurat.obj.filt), 10)

plot1 <- VariableFeaturePlot(Seurat.obj.filt)
LabelPoints(plot = plot1, points = top10, repel = TRUE)
```



with visualization of top 10 variable genes

5. Clustering the cells

5.1 Scale the data

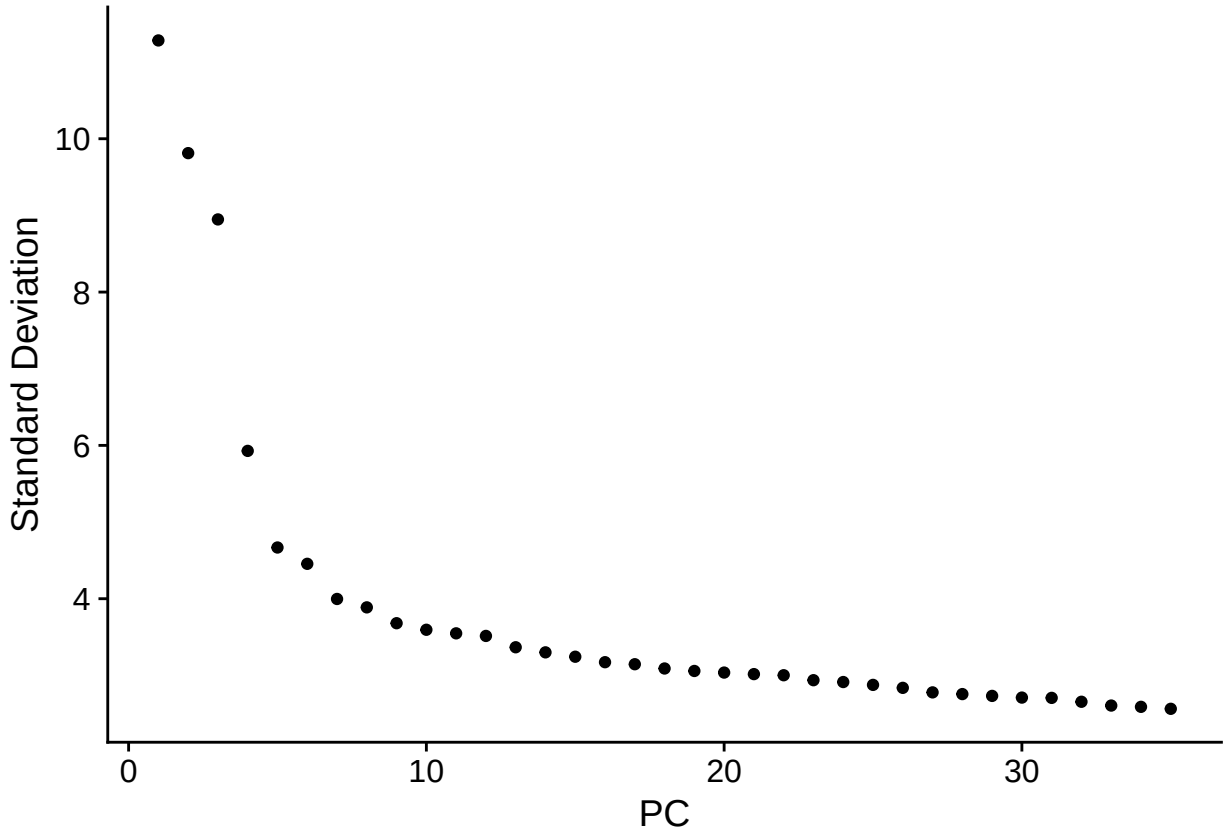
```
Seurat.obj.filt <- ScaleData(object = Seurat.obj.filt)
```

5.2 Perform linear dimensionality reduction

```
Seurat.obj.filt <- RunPCA(object = Seurat.obj.filt, features = VariableFeatures((Seurat.obj.filt)))
```

5.3 Select the PCA plots with elbow plot

```
ElbowPlot(Seurat.obj.filt, ndims = 35)
```



5.4 Find Neighbors

```
Seurat.obj.filt <- FindNeighbors(object = Seurat.obj.filt, dim = 1:20)
```

5.5 Understand the resolution

```
Seurat.obj.filt <- RunUMAP(object = Seurat.obj.filt, dim = 1:20)
Seurat.obj.filt <- FindClusters(object = Seurat.obj.filt, resolution = c(0.01, 0.1, 0.3, 0.5, 0.8, 1, 1
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 736
## Number of edges: 21217
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9932
```

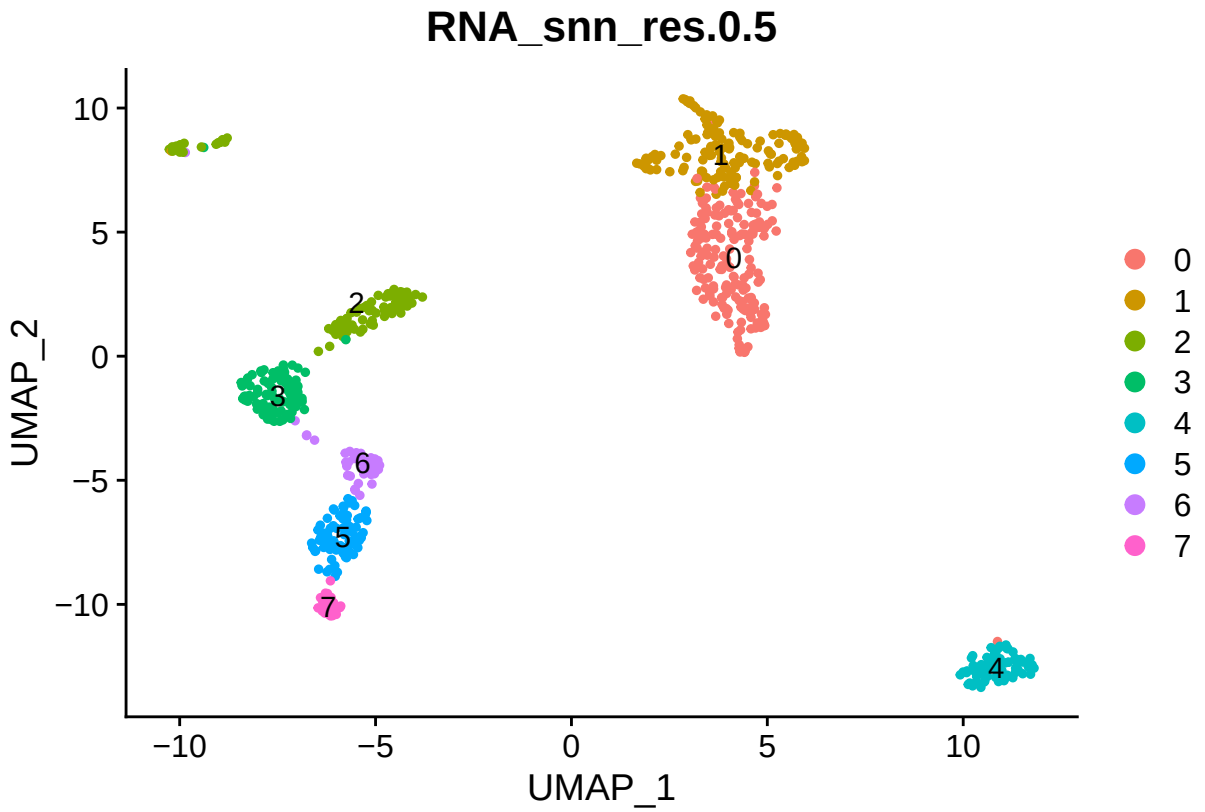
```
## Number of communities: 3
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 736
## Number of edges: 21217
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9618
## Number of communities: 4
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 736
## Number of edges: 21217
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9111
## Number of communities: 7
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 736
## Number of edges: 21217
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8790
## Number of communities: 8
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 736
## Number of edges: 21217
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8422
## Number of communities: 11
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 736
## Number of edges: 21217
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8190
## Number of communities: 11
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 736
## Number of edges: 21217
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7959
```

```
## Number of communities: 11
## Elapsed time: 0 seconds
```

5.6 Optimize the resolution

```
DimPlot(Seurat.obj.filt, group.by = 'RNA_snn_res.0.5', label = TRUE)
```

This resolution should be changed until the correct number of clusters are achieved

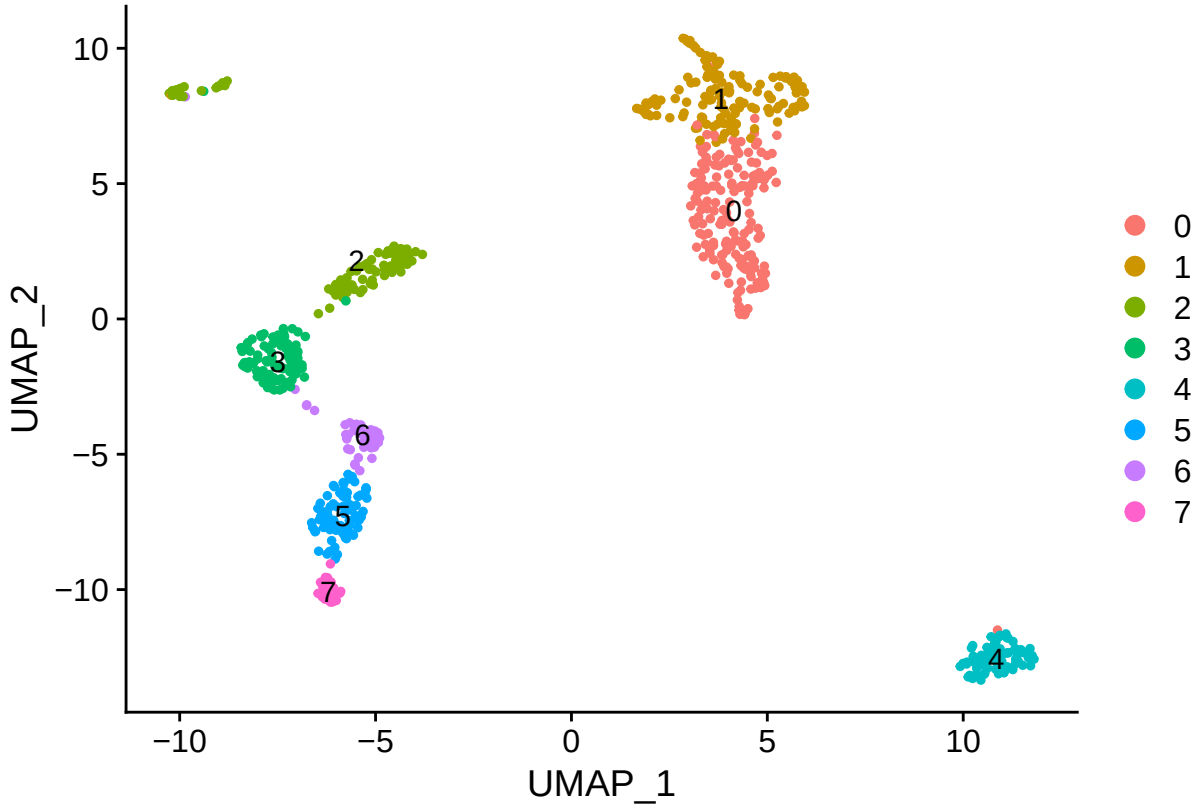


5.7 Set identity of clusters

```
Seurat.obj.filt <- RunUMAP(object = Seurat.obj.filt, dim = 1:20)
Seurat.obj.filt <- RunTSNE(object = Seurat.obj.filt, dims = 1:20)
Idents(Seurat.obj.filt) <- 'RNA_snn_res.0.5'
```

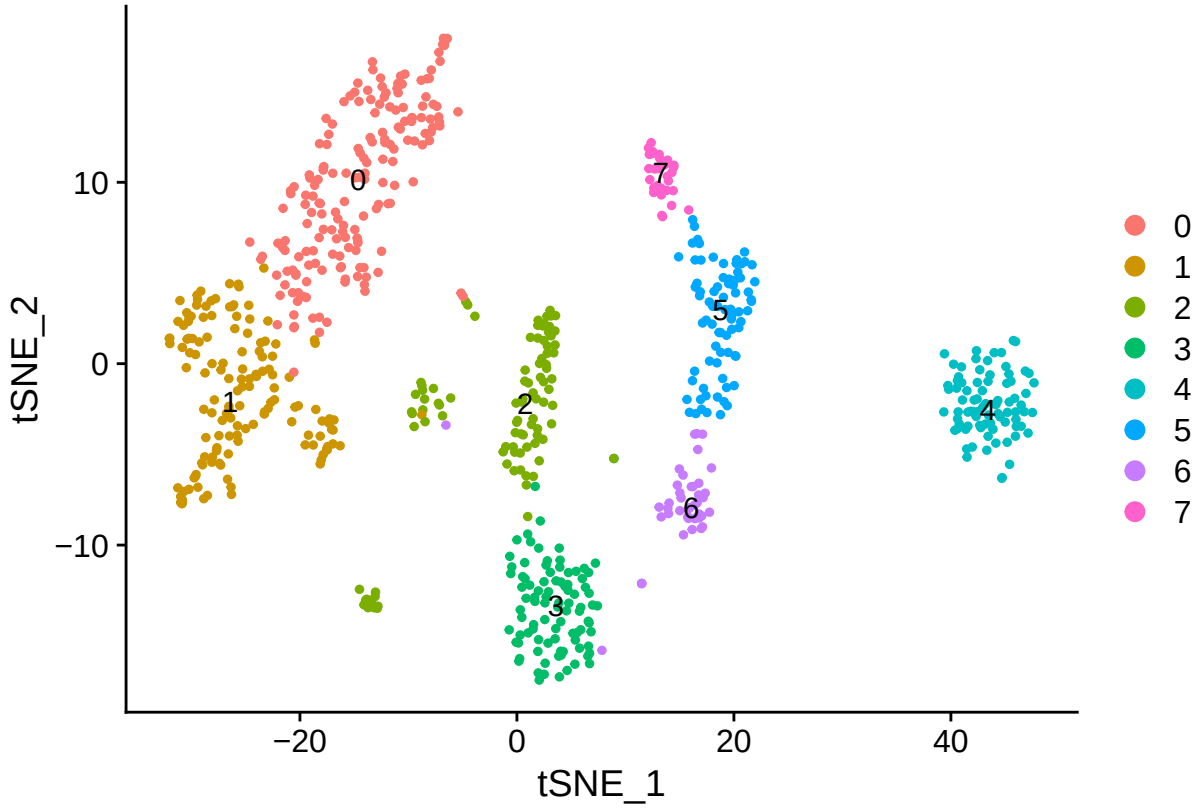
5.8 umap observation

```
DimPlot(Seurat.obj.filt, reduction = 'umap', label = TRUE)
```



5.9 tsne observation

```
DimPlot(Seurat.obj.filt, reduction = 'tsne' , label = TRUE)
```



6. Annotate the clusters

6.1 Find the assay type

```
DefaultAssay(Seurat.obj.filt) # make sure it is RNA
```

```
## [1] "RNA"
```

6.2 Find the differentially expressed markers

```
Seurat.obj.filt.markers <- FindAllMarkers(Seurat.obj.filt, only.pos = TRUE, min.pct = 0.25, logfc.thresh
```

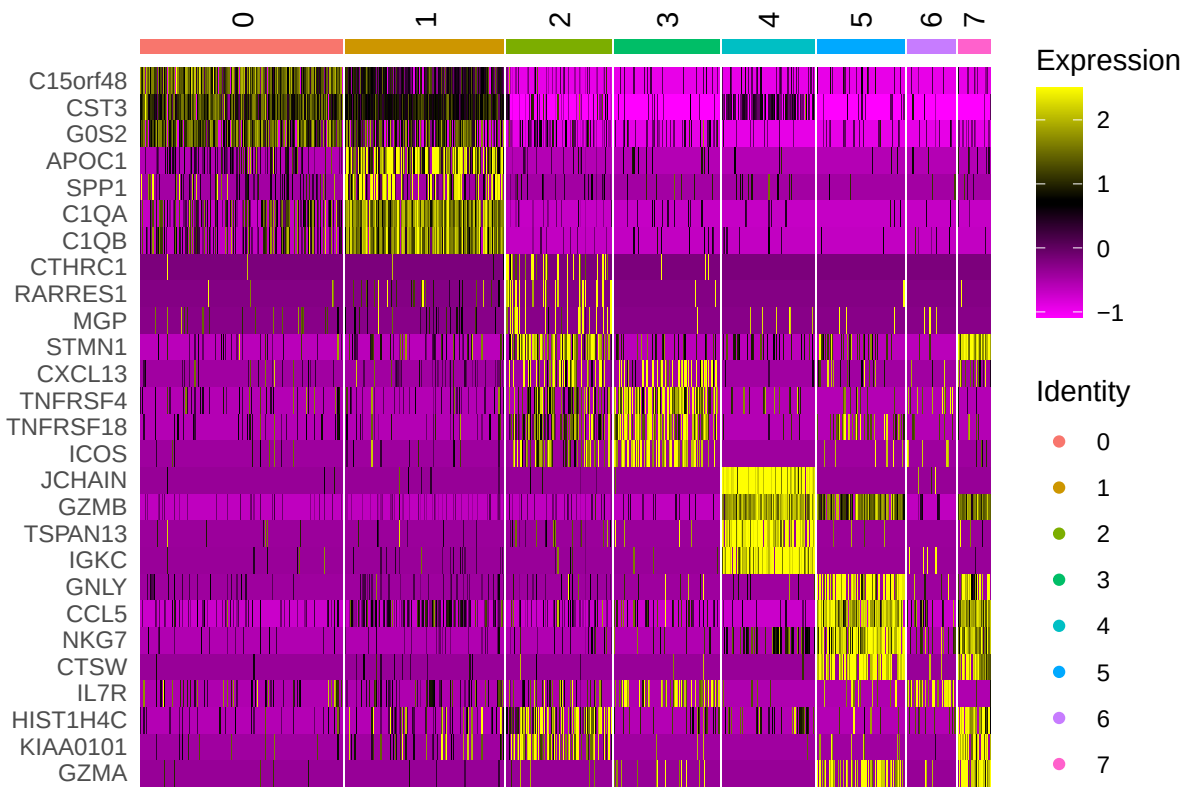
Find markers for every cluster compared to all remaining cells, report only the positive ones

6.2 Select top (4 -> can be changed) upregulated genes in each cluster

```
clust.markers <- Seurat.obj.filt.markers %>%
  group_by(cluster) %>%
  slice_max(n = 4, order_by = avg_log2FC)
```

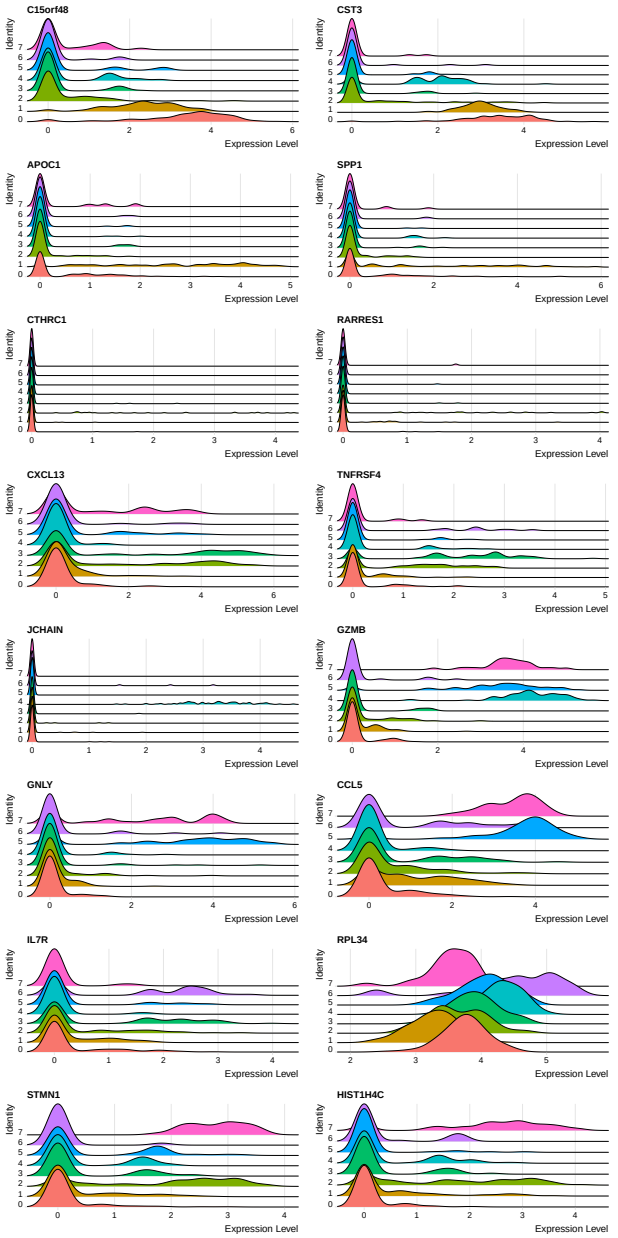
6.3 Visualize top upregulated genes in each cluster

```
DoHeatmap(Seurat.obj.filt, features = clust.markers$gene, size = 4,
          angle = 90)
```



6.4 Ridge plots - from ggridges.

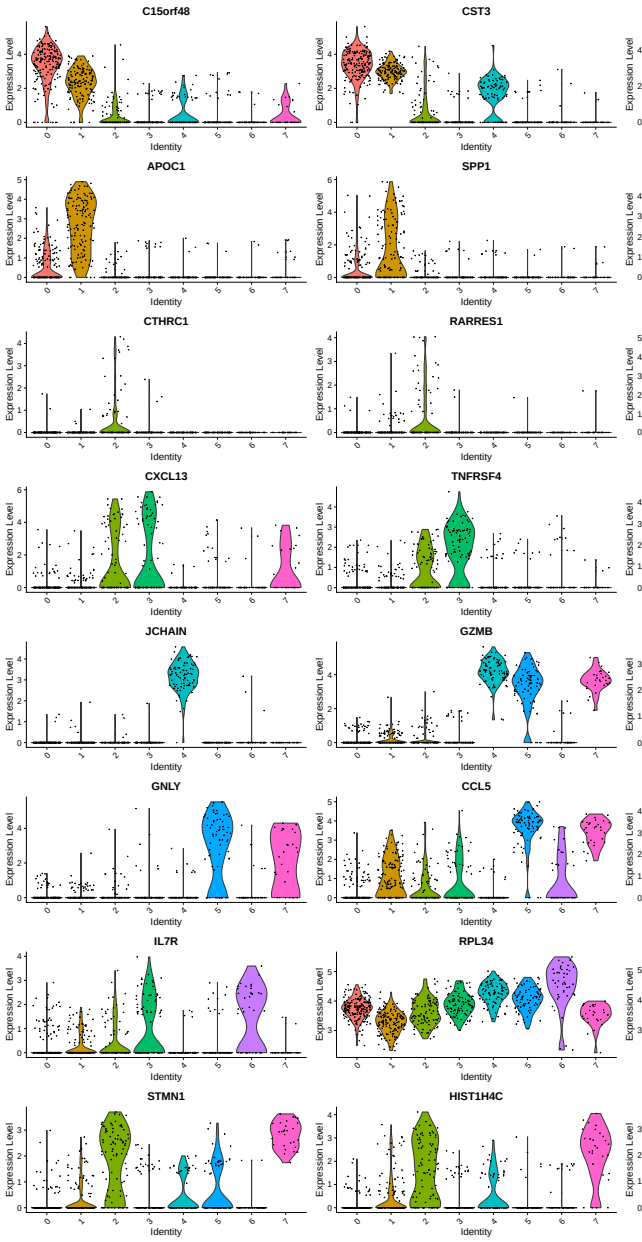
```
RidgePlot(Seurat.obj.filt, features = clust.markers$gene, ncol = 4)
```



Visualize single cell expression distributions in each cluster

6.5 Violin plot

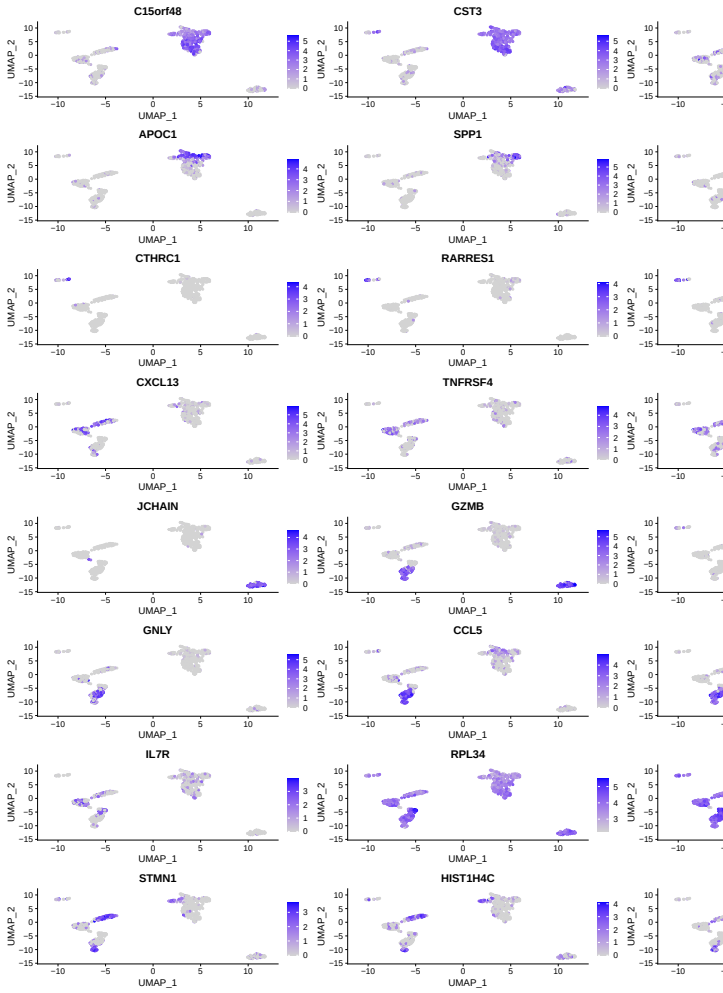
```
VlnPlot(Seurat.obj.filt, features = clust.markers$gene, ncol = 4)
```



Visualize single cell expression distributions in each cluster

6.6 Feature plot

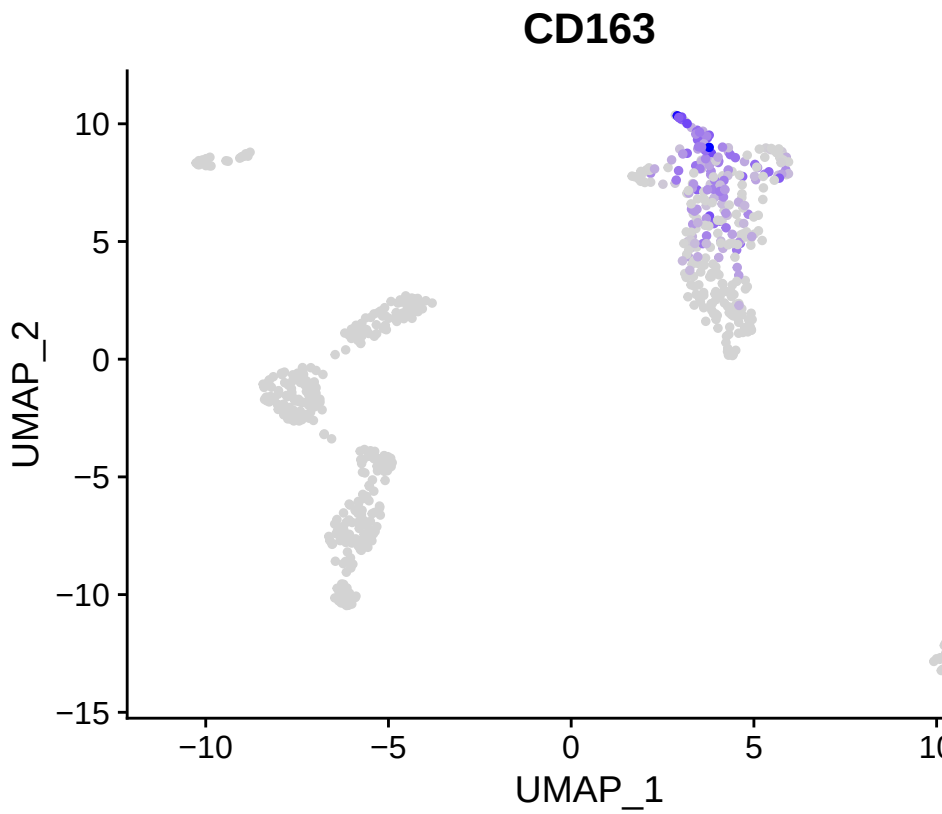
```
FeaturePlot(Seurat.obj.filt, features = clust.markers$gene, ncol = 4)
```



visualize feature expression in low-dimensional space

6.7 Check the individual feature

```
FeaturePlot(Seurat.obj.filt, features = c('CD163'), min.cutoff = 'q10')
```



Use for optimization of annotating

6.8 Select the top features

```
features <- c("CST3", "CD86", "SPP1", "C1QA", "CTHRC1", "MGP", "CXCL13", "TNFRSF18", "JCHAIN", "IGKC",
              "NKG7", "GNLY", "IL7R", "RPL34", "STMN1", "KIAA0101")
```

Feature identification based on up-reg genes and pangloadb

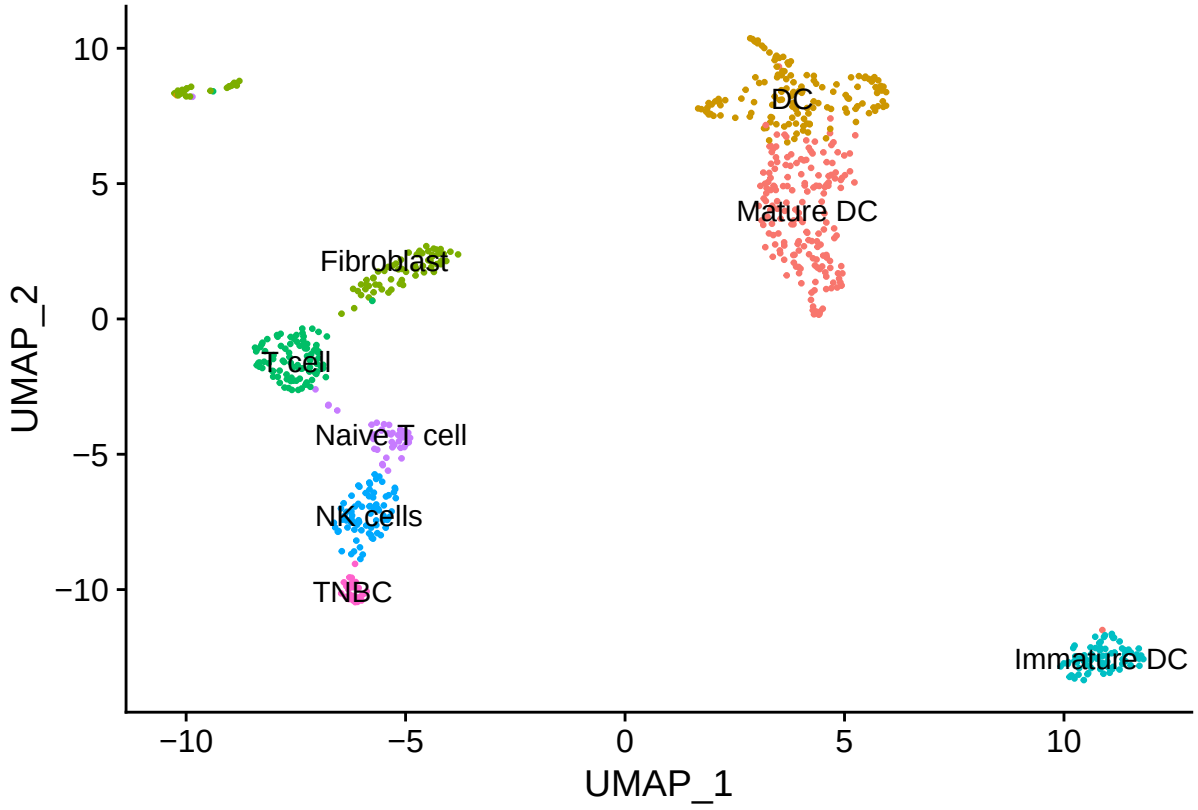
6.9 Assign the new clusters to data

```
new.cluster.ids <- c("Mature DC", "DC", "Fibroblast", "T cell", "Immature DC",
                     "NK cells", "Naive T cell", "TNBC")
```

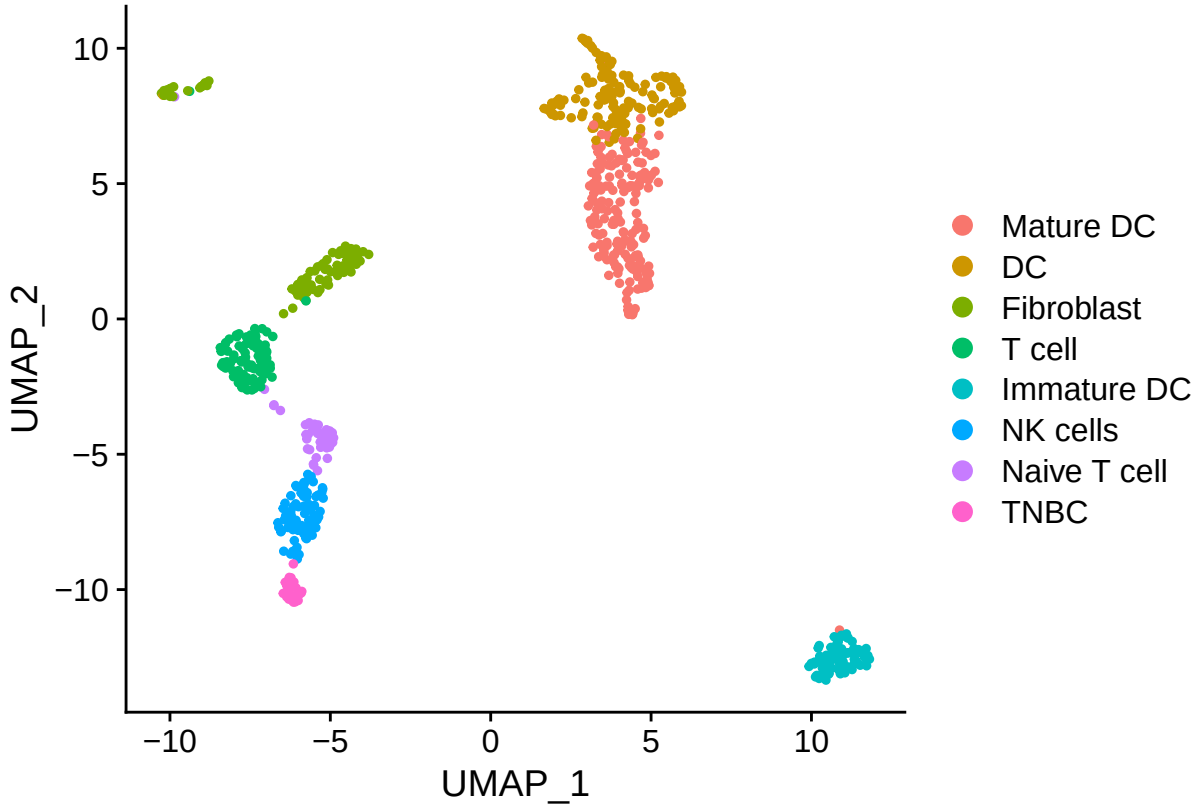
7. Visulization

7.1 View annotated clusters

```
names(new.cluster.ids) <- levels(Seurat.obj.filt)
Seurat.obj.filt <- RenameIdents(Seurat.obj.filt, new.cluster.ids)
DimPlot(Seurat.obj.filt, reduction = "umap", label = TRUE, pt.size = 0.5) + NoLegend()
```

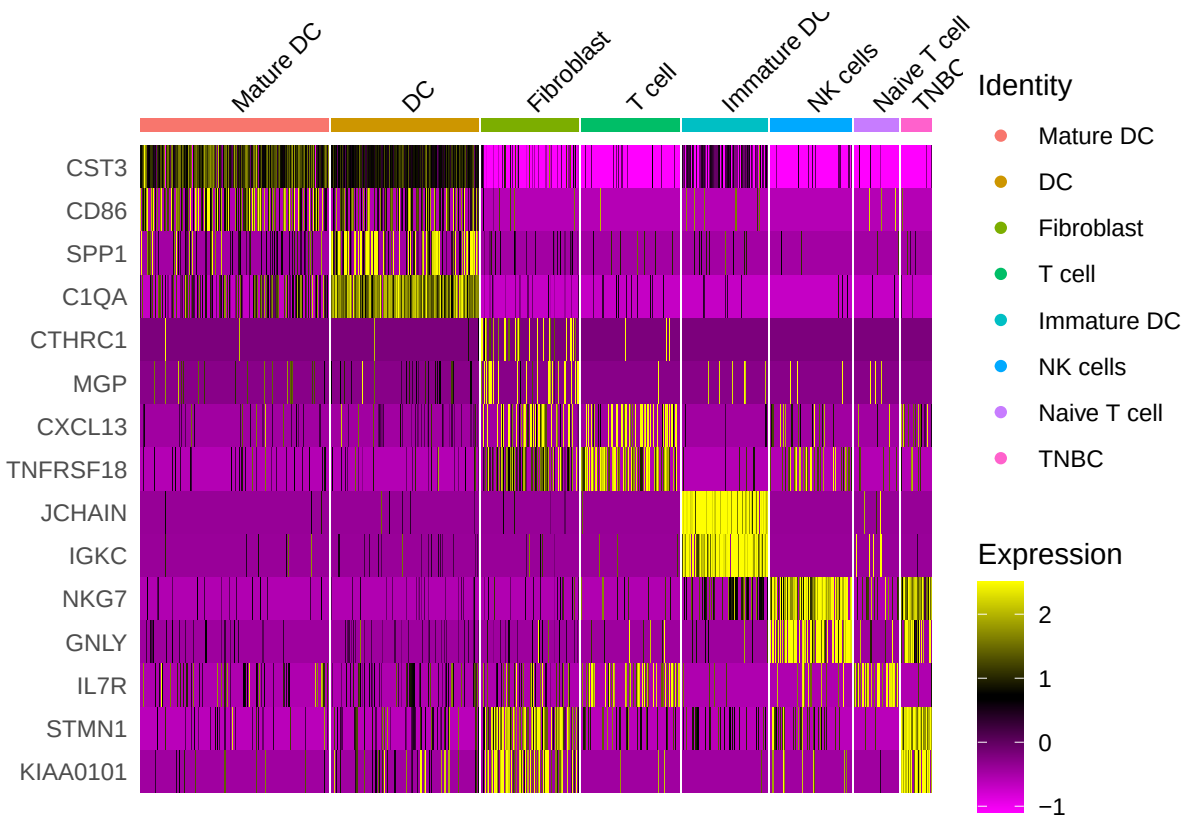



```
DimPlot(Seurat.obj.filt, reduction = "umap")
```



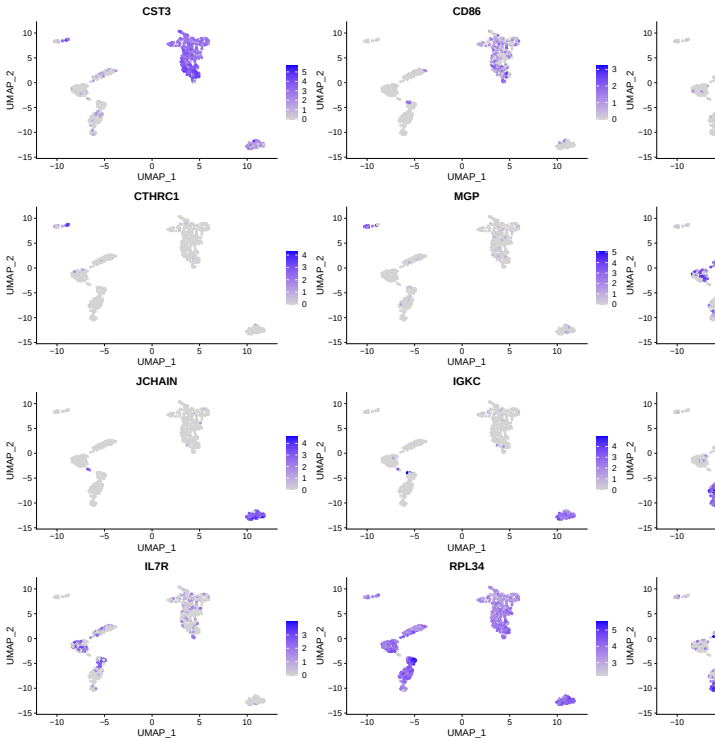
7.2 Single cell heatmap of features expression

```
DoHeatmap(subset(Seurat.obj.filt, downsample = 700), features = features, size = 3)
```



7.3 Feature plot

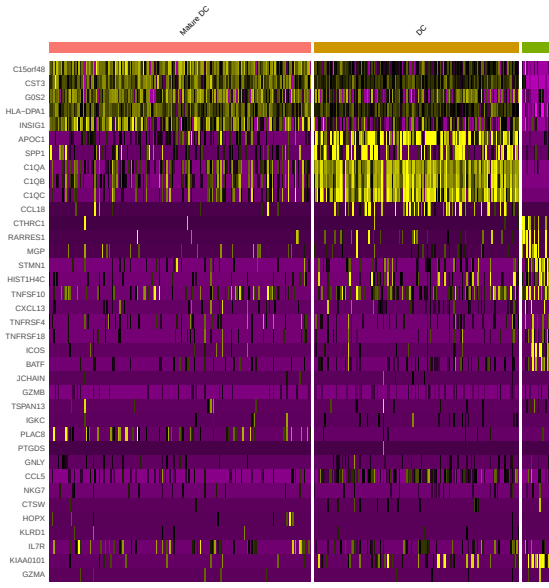
```
FeaturePlot(Seurat.obj.filt, features = features, ncol = 4)
```



Visualize feature expression in low-dimensional space

7.4 Heatmap plot

```
top.features <- Seurat.obj.filt.markers %>%
  group_by(cluster) %>%
  slice_max(n = 6, order_by = avg_log2FC)
DoHeatmap(subset(Seurat.obj.filt, downsample = 700), features = top.features$gene, size = 3)
```

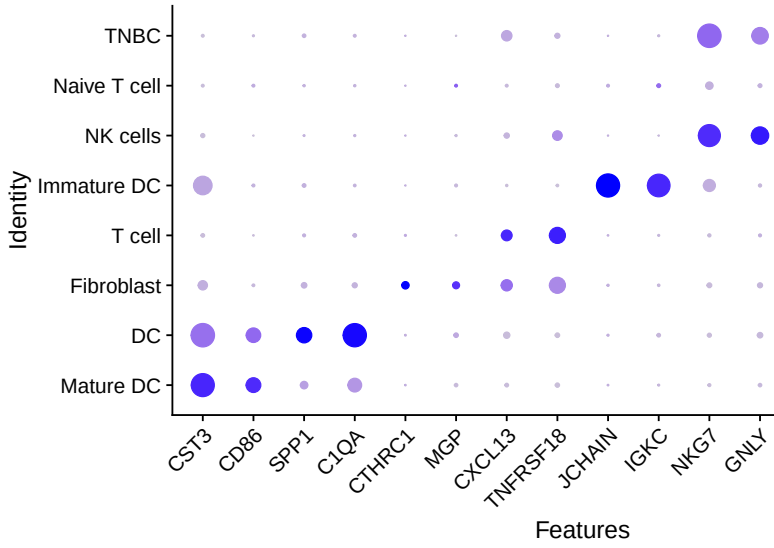


Select top (6 -> can be changed) upregulated genes in each cluster

7.5 Dot plots

```
DotPlot(Seurat.obj.filt, features = features) + RotatedAxis()
```

the size of the dot corresponds to the percentage of cells expressing the feature in each cluster.



The color represents the average expression level