```r
#install.packages('Seurat')
#install.packages("parallelly")
library(SeuratObject)
```

```
## Loading required package: sp

##
## Attaching package: 'SeuratObject'

## The following objects are masked from 'package:base':
##
##     intersect, t
```

```r
library(Seurat)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

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

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

```r
# sample_names <- c("GSM4039241_f-ctrl-1", "GSM4039242_f-ctrl-2",
#                   "GSM4039243_f-tumor-1", "GSM4039244_f-tumor-2",
#                   "GSM4039245_m-ctrl-1", "GSM4039246_m-ctrl-2",
#                   "GSM4039247_m-tumor-1", "GSM4039248_m-tumor-2")

# seurat_list <- list()
# setwd("/mnt/pv_compute/yifan/practice/scRNA.practice/")
# for (sample in sample_names) {
#   data_dir <- paste0("processed/", sample)
#
#   # Read the data
#   sc_data <- Read10X(data.dir = data_dir)
#
#   # Create Seurat object
#   seurat_obj <- CreateSeuratObject(counts = sc_data, project = sample)
#
#   # Add sample metadata
#   seurat_obj$sample <- sample
#
#   # Assign condition based on sample name
#   if (grepl("f-ctrl", sample)) {
#     seurat_obj$condition <- "female_control"
#   } else if (grepl("f-tumor", sample)) {
#     seurat_obj$condition <- "female_tumor"
#   } else if (grepl("m-ctrl", sample)) {
#     seurat_obj$condition <- "male_control"
#   } else if (grepl("m-tumor", sample)) {
#     seurat_obj$condition <- "male_tumor"
#   }
#
```

```
#   # Add sex metadata
#   if (grepl("^GSM403924[1-4]", sample)) {
#     seurat_obj$sex <- "female"
#   } else {
#     seurat_obj$sex <- "male"
#   }
#
#   # Add treatment metadata
#   if (grepl("ctrl", sample)) {
#     seurat_obj$treatment <- "control"
#   } else if (grepl("tumor", sample)) {
#     seurat_obj$treatment <- "tumor"
#   }
#
#   # Store the Seurat object in the list
#   seurat_list[[sample]] <- seurat_obj
# }
#
#
# for (i in 1:length(seurat_list)) {
#   seurat_list[[i]][["percent.mt"]] <- PercentageFeatureSet(seurat_list[[i]], pattern = "^MT-")
#   seurat_list[[i]] <- subset(seurat_list[[i]], subset = nFeature_RNA > 200 & nFeature_RNA < 3000 & pe
# }
#
# for (i in 1:length(seurat_list)) {
#   seurat_list[[i]] <- NormalizeData(seurat_list[[i]],normalization.method = "LogNormalize", scale.fac
#   seurat_list[[i]] <- FindVariableFeatures(seurat_list[[i]], selection.method = "vst", nfeatures = 20
# }
```

```
# Read 10X data
test.data <- Read10X(data.dir = "/mnt/pv_compute/yifan/practice/scRNA.practice/processed/GSM4039241_f-c

# Create Seurat object
test <- CreateSeuratObject(counts = test.data, project = "female_control")

# Normalize the data
test <- NormalizeData(test, normalization.method = "LogNormalize", scale.factor = 10000)
```

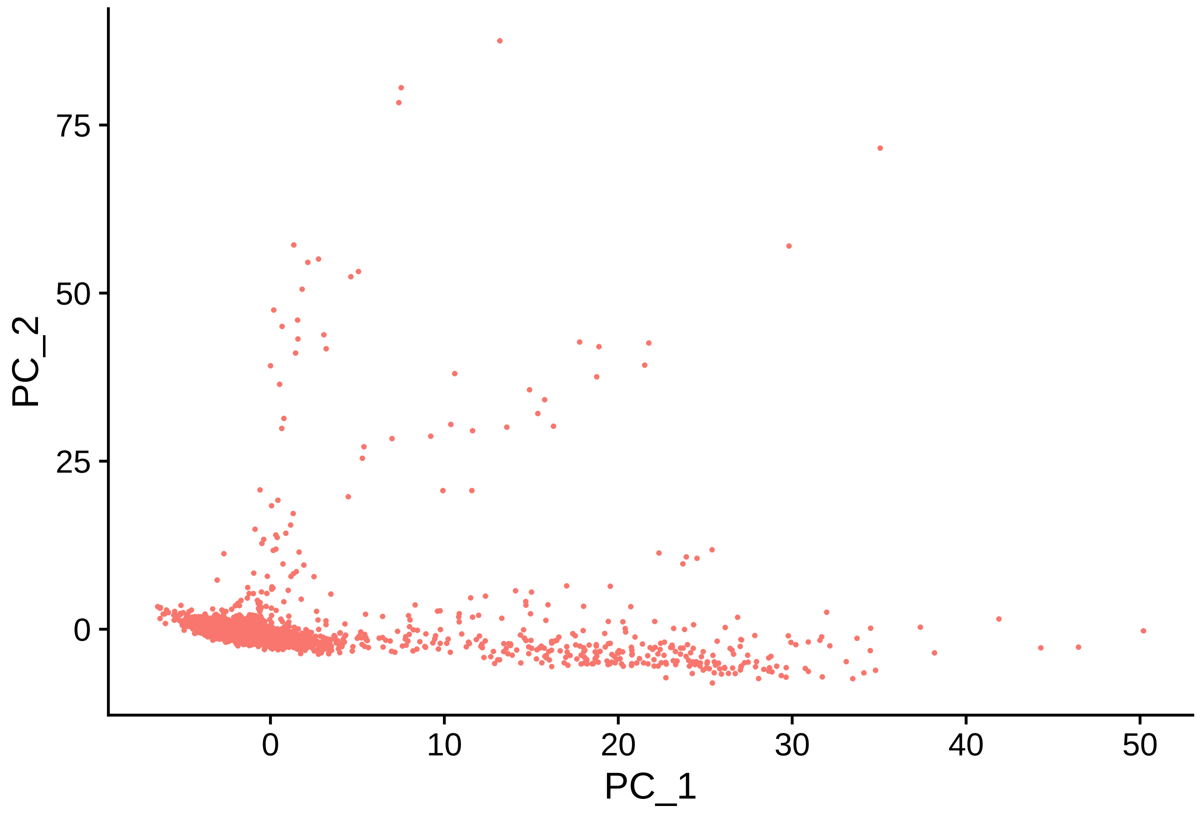
```
## Normalizing layer: counts
```

```
# Identify variable features (optional, for efficiency)
test <- FindVariableFeatures(test, selection.method = "vst", nfeatures = 2000)
```

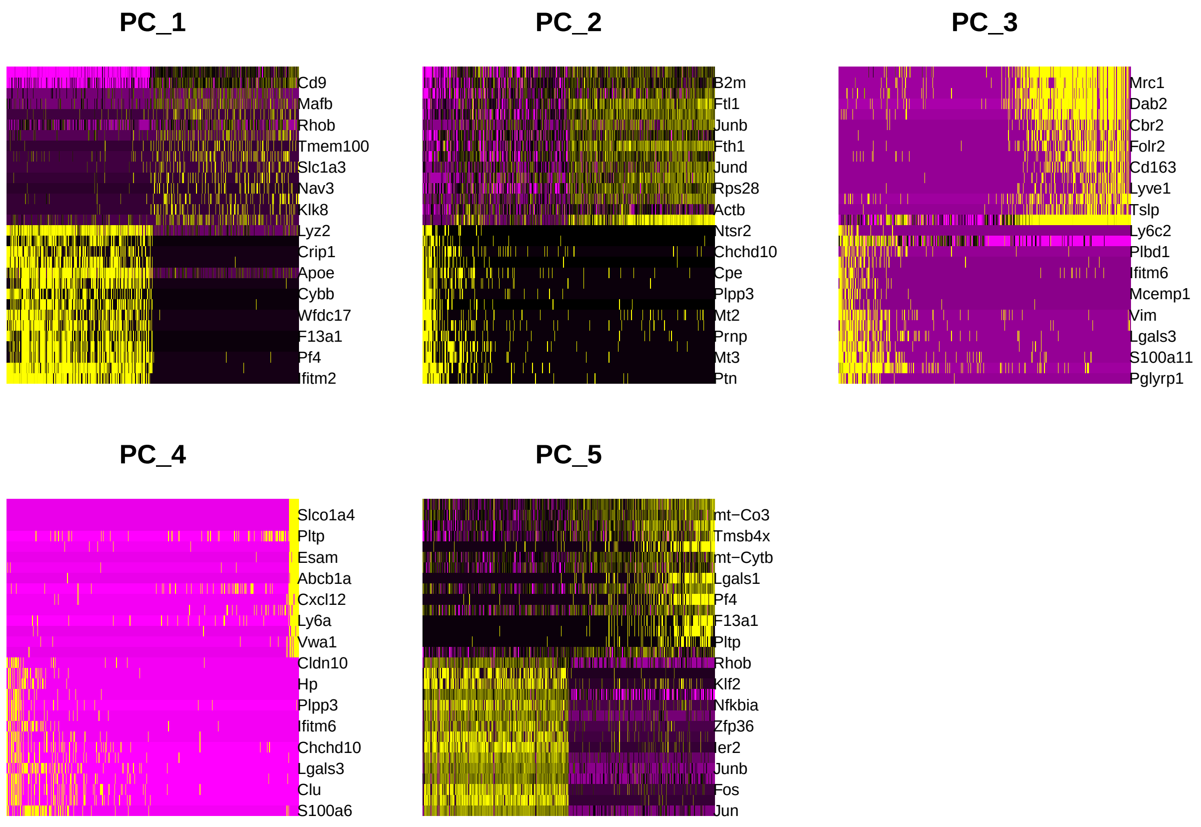
```
## Finding variable features for layer counts
```

```
# Scale the data
test <- ScaleData(test, verbose = FALSE)

# Run PCA
test <- RunPCA(test, features = VariableFeatures(object = test), verbose = FALSE)
```

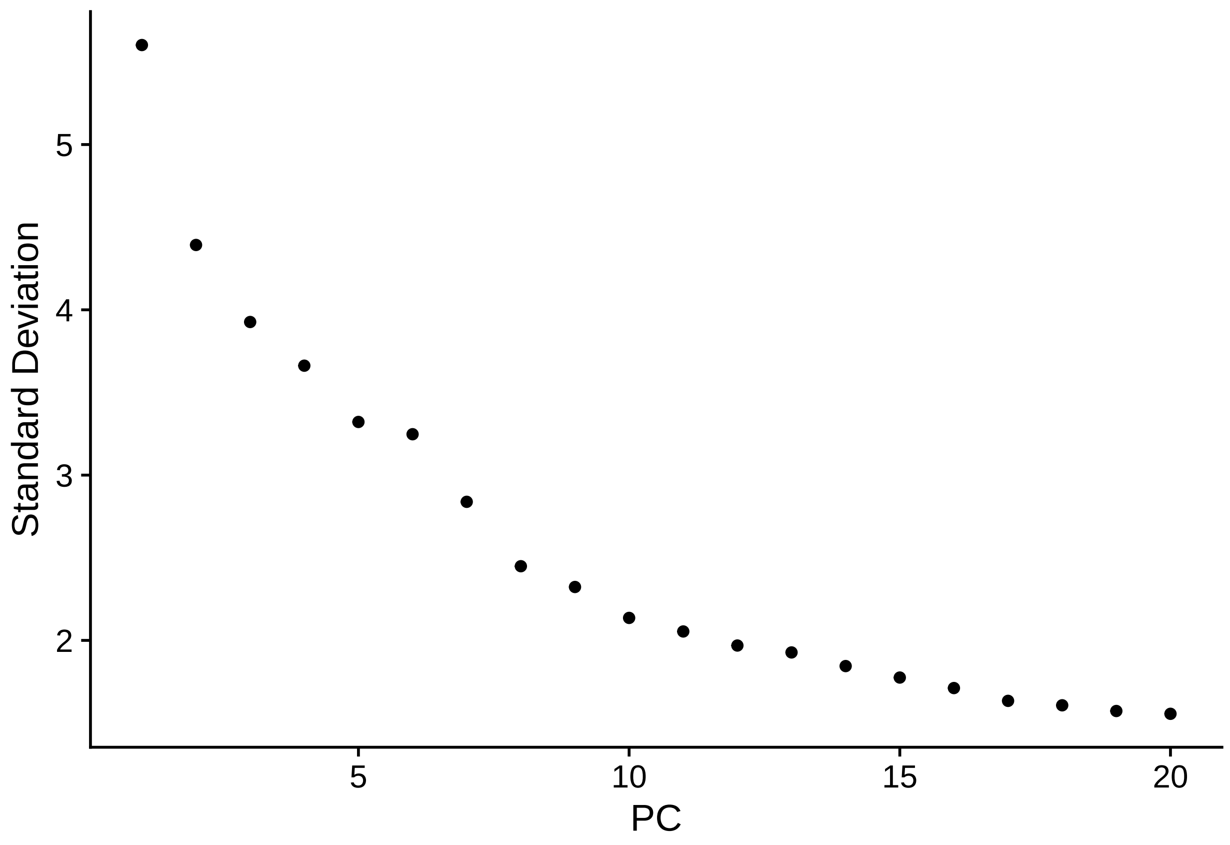
```
DimPlot(test, reduction = "pca") + NoLegend()
```



```r
DimHeatmap(test, dims = 1:5, cells = 500, balanced = TRUE)
```



```r
## check the ideal cluster
ElbowPlot(test)
```



```r
test = FindNeighbors(test, dims = 1:30)
```

```
## Computing nearest neighbor graph
```

```
## Computing SNN
```

```r
test = FindClusters(test,resolution= 0.3)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 5223
## Number of edges: 203588
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8339
## Number of communities: 8
## Elapsed time: 0 seconds
```

```r
test = RunUMAP(test, dims = 1:30)
```

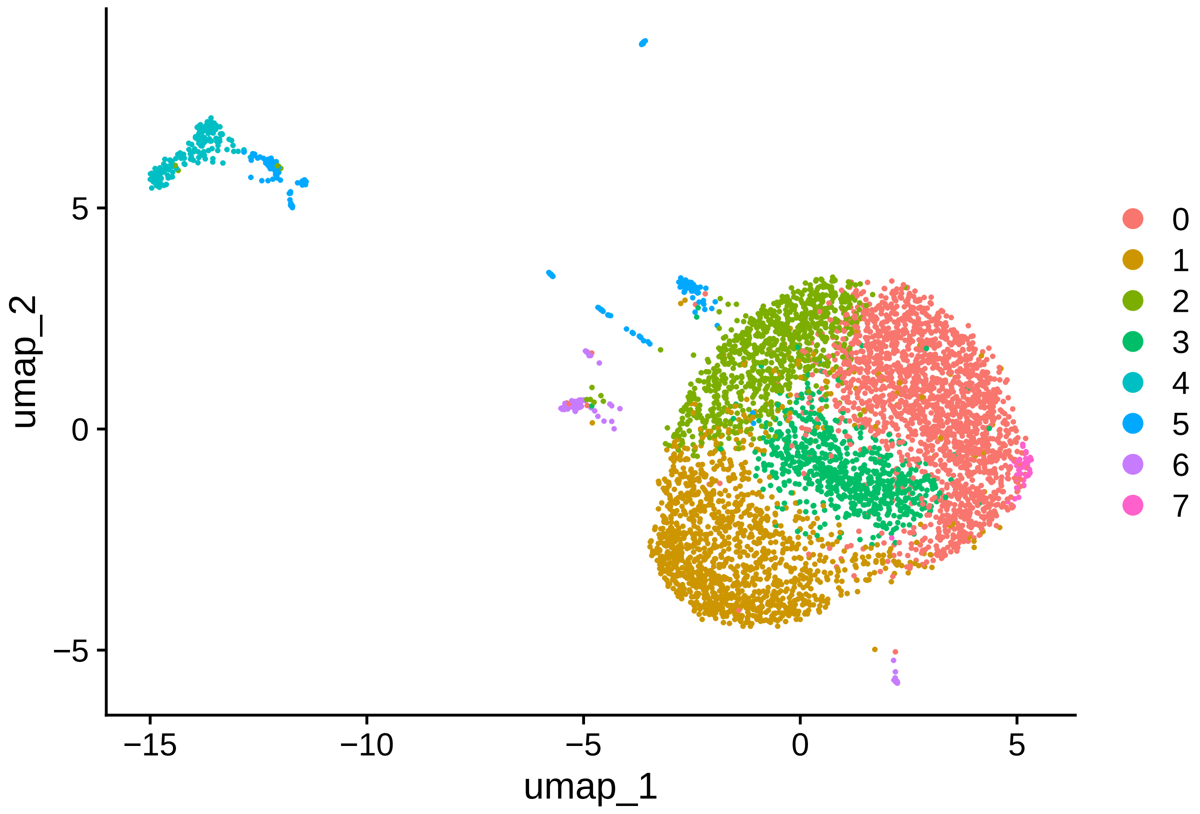
```
## Warning: The default method for RunUMAP has changed from calling Python UMAP via reticulate to the R
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric to 'correlation'
## This message will be shown once per session
```

```
## 22:15:59 UMAP embedding parameters a = 0.9922 b = 1.112
```

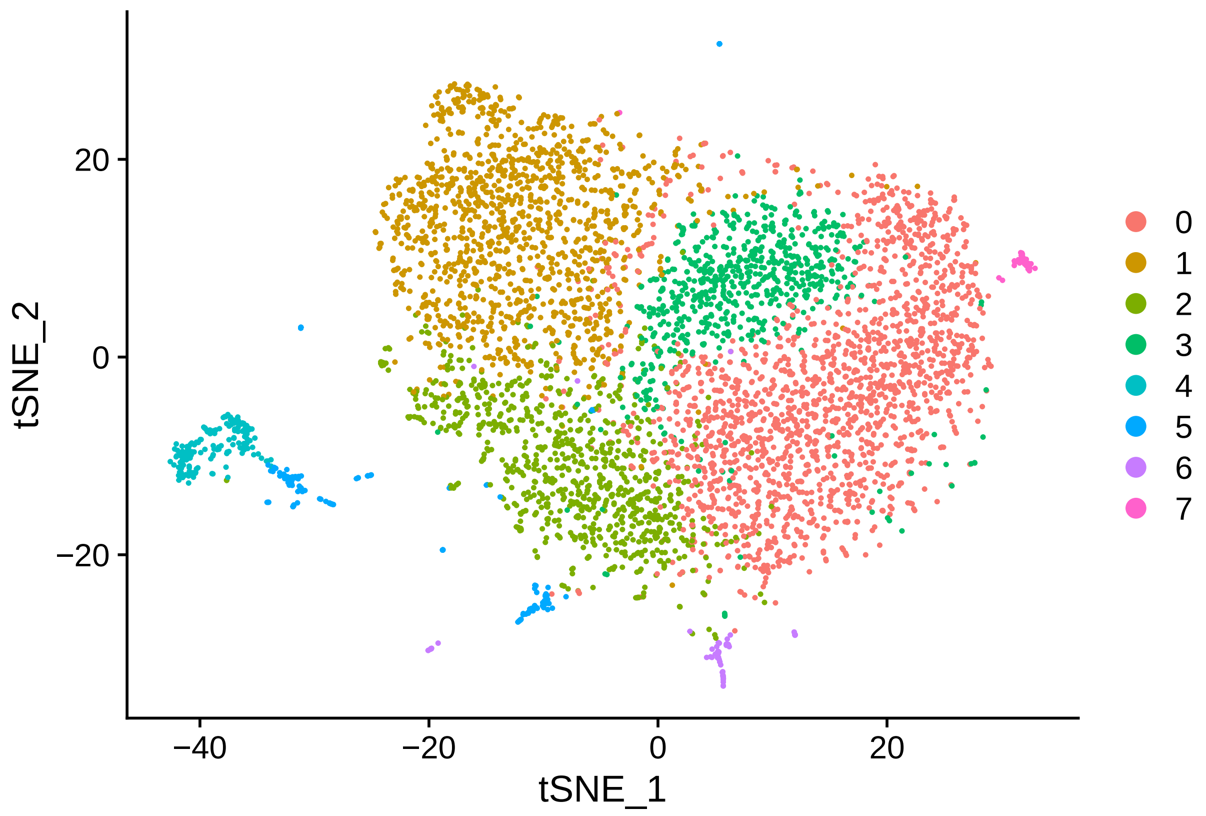
```
## 22:15:59 Read 5223 rows and found 30 numeric columns
```

```
## 22:15:59 Using Annoy for neighbor search, n_neighbors = 30
## 22:15:59 Building Annoy index with metric = cosine, n_trees = 50
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
## 22:15:59 Writing NN index file to temp file /tmp/RtmpIsyEiB/file2bf8e133e6bc1
## 22:15:59 Searching Annoy index using 1 thread, search_k = 3000
## 22:16:01 Annoy recall = 100%
## 22:16:01 Commencing smooth kNN distance calibration using 1 thread with target n_neighbors = 30
## 22:16:02 Initializing from normalized Laplacian + noise (using RSpectra)
## 22:16:02 Commencing optimization for 500 epochs, with 228770 positive edges
## 22:16:07 Optimization finished
```

```r
DimPlot(test, reduction = "umap")
```



```r
test <- RunTSNE(test, dims = 1:30)

DimPlot(test, reduction = "tsne")
```



```
#save by
#saveRDS(test, file = "../output/test.rds")
```

check markers

```
cluster0.markers = FindMarkers(test, ident.1 = 0)
head(cluster0.markers)
```

```
##                p_val  avg_log2FC pct.1 pct.2     p_val_adj
## Junb  1.483281e-200  -2.037836 0.530 0.806 4.606032e-196
## Jun   2.511226e-196  -2.358300 0.507 0.795 7.798110e-192
## Fos   7.671462e-183  -4.274760 0.111 0.495 2.382219e-178
## Jund  8.037950e-177  -1.918341 0.618 0.838 2.496025e-172
## Egr1  2.405958e-156  -3.808391 0.043 0.381 7.471221e-152
## Dusp1 1.035139e-142  -3.649144 0.086 0.415 3.214416e-138
```

```
test.markers <- FindAllMarkers(test, only.pos = TRUE)
```

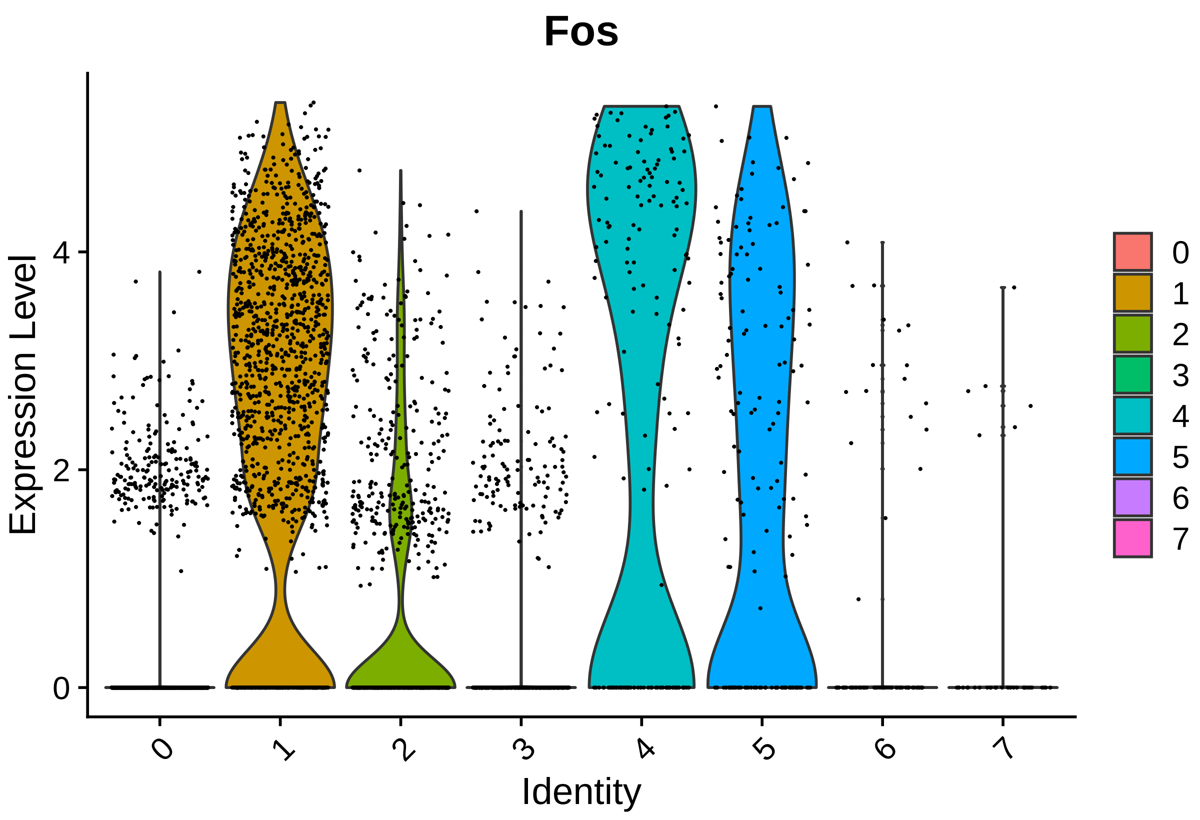
```
## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5
## Calculating cluster 6
```

```
## Calculating cluster 7
```

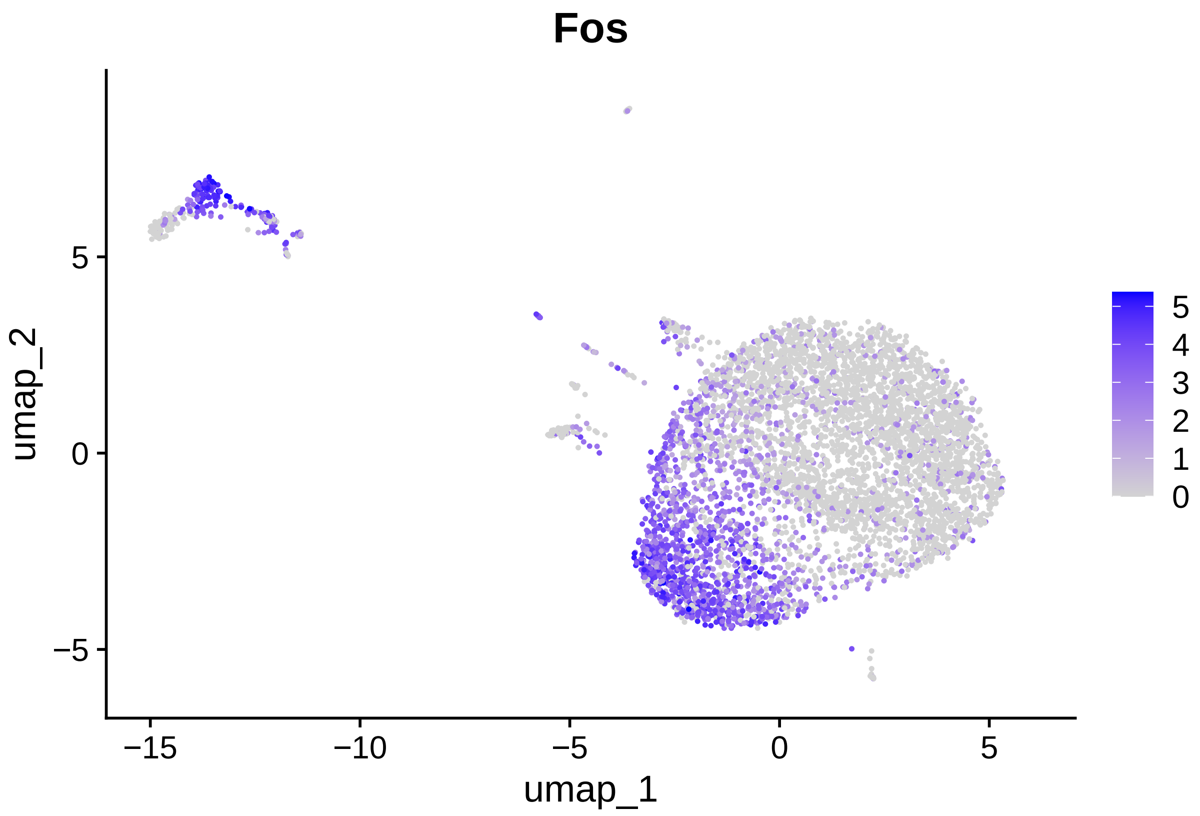
```r
test.markers = test.markers %>%
               group_by(cluster) %>%
               dplyr::filter(avg_log2FC > 1)
head(test.markers$gene)
```

```
## [1] "Snhg20" "Hyal1"  "Zfp266" "Als2"   "Gm3716" "Yeats2"
```

```r
VlnPlot(test, features = c("Fos"))
```



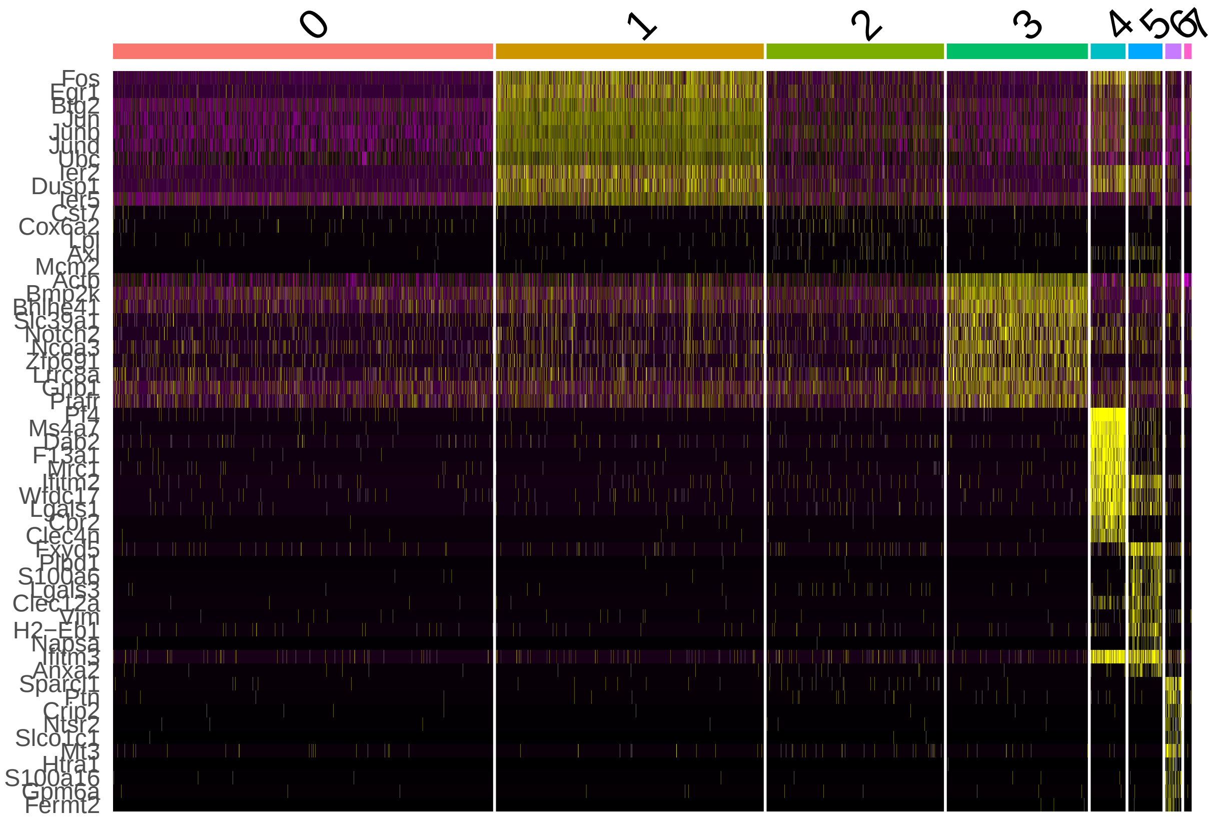
```r
FeaturePlot(test, features = c("Fos"))
```



## Annotation

```r
test.markers %>%
    group_by(cluster) %>%
    dplyr::filter(avg_log2FC > 1) %>%
    slice_head(n = 10) %>%
    ungroup() -> top10
DoHeatmap(test, features = top10$gene) + NoLegend()
```

```
## Warning in DoHeatmap(test, features = top10$gene): The following features were
## omitted as they were not found in the scale.data slot for the RNA assay:
## C230004F18Rik, Smarca1, Scn2a, Kcnj3, Grin1, Gm13199, Kcnt2, Unc80, Adam23,
## Fam155a, Kif16b, Gm11361, Gm10036, St8sia6, A930007I19Rik, Yeats2, Gm3716,
## Als2, Zfp266, Hyal1, Snhg20
```



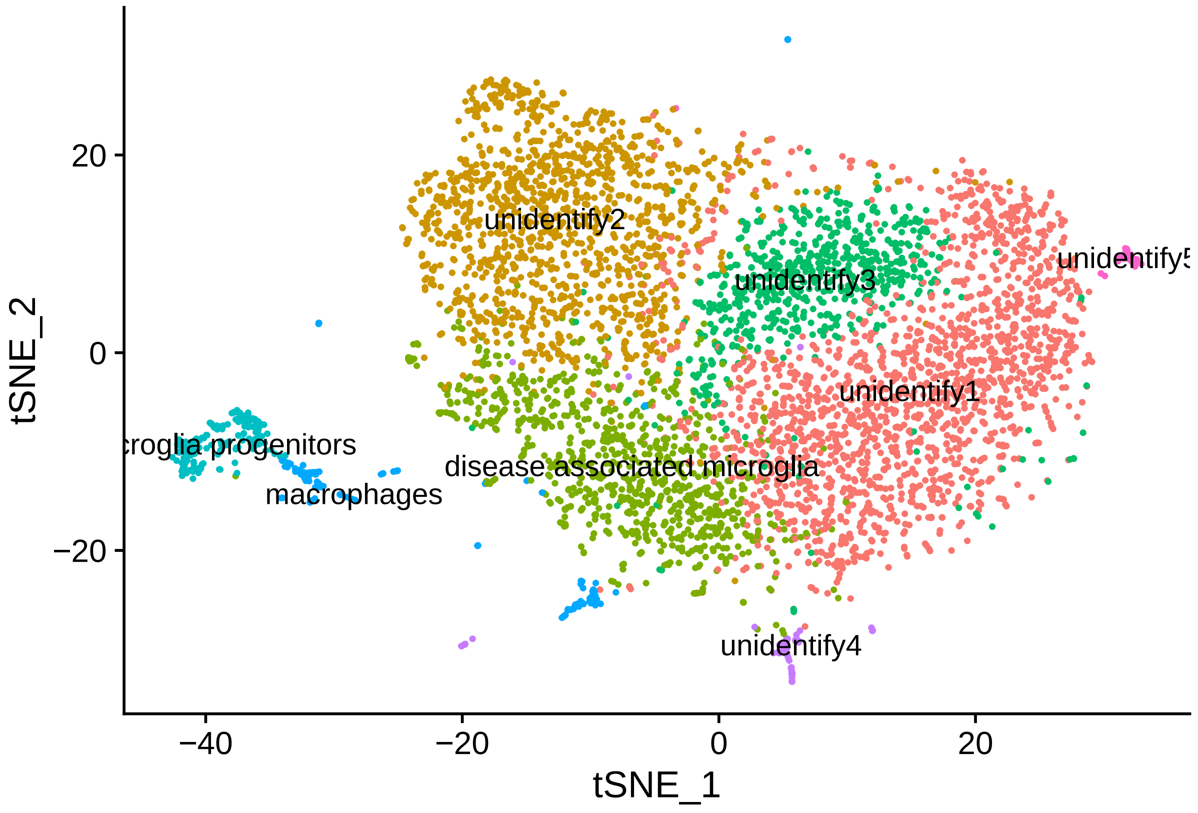
```r
annota.ref = read.csv("annotation.csv")
df <- data.frame()

matched_genes <- top10 %>%
  inner_join(annota.ref, by = c("gene" = "Gene"))

matched_genes
```

```
## # A tibble: 10 x 8
##        p_val avg_log2FC pct.1 pct.2 p_val_adj cluster gene   Target
##        <dbl>      <dbl> <dbl> <dbl>     <dbl> <fct>   <chr>  <chr>
##  1 6.25e- 11       1.39 0.061 0.021 1.94e-  6 2       Cst7   disease associated~
##  2 1.21e-  7       2.26 0.035 0.011 3.76e-  3 2       Lpl    disease associated~
##  3 0               8.36 0.983 0.021 0         4       Pf4    microglia progenit~
##  4 0               6.59 0.884 0.026 0         4       Dab2   early microglia
##  5 0               7.77 0.814 0.006 0         4       F13a1  microglia progenit~
##  6 0               7.77 0.814 0.006 0         4       F13a1  macrophages
##  7 0               7.23 0.82  0.015 0         4       Mrc1   Border Associated ~
##  8 0               5.03 0.767 0.031 0         4       Ifitm2 macrophages
##  9 1.68e-290       8.08 0.333 0.003 5.21e-286 5       S100a6 macrophages
## 10 2.41e-199       4.55 0.708 0.064 7.47e-195 5       Ifitm3 macrophages
```

```r
levels(test)
```

```
## [1] "0" "1" "2" "3" "4" "5" "6" "7"
```

```r
cluster.id= c("unidentify1","unidentify2","disease associated microglia",
           "unidentify3","microglia progenitors","macrophages",
            "unidentify4","unidentify5")
names(cluster.id) <- levels(test)
test <- RenameIdents(test, cluster.id)
DimPlot(test, reduction = "tsne", label = TRUE, pt.size = 0.5) + NoLegend()
```



# further analysis should focuse data integration # https://satijalab.org/seurat/articles/integration_intr oduction