sc rna

Sravani Vippaturi

2023-10-28

```
library(Seurat)
```

```
## Attaching SeuratObject
```

```
library(patchwork)
library(ggplot2)
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
```

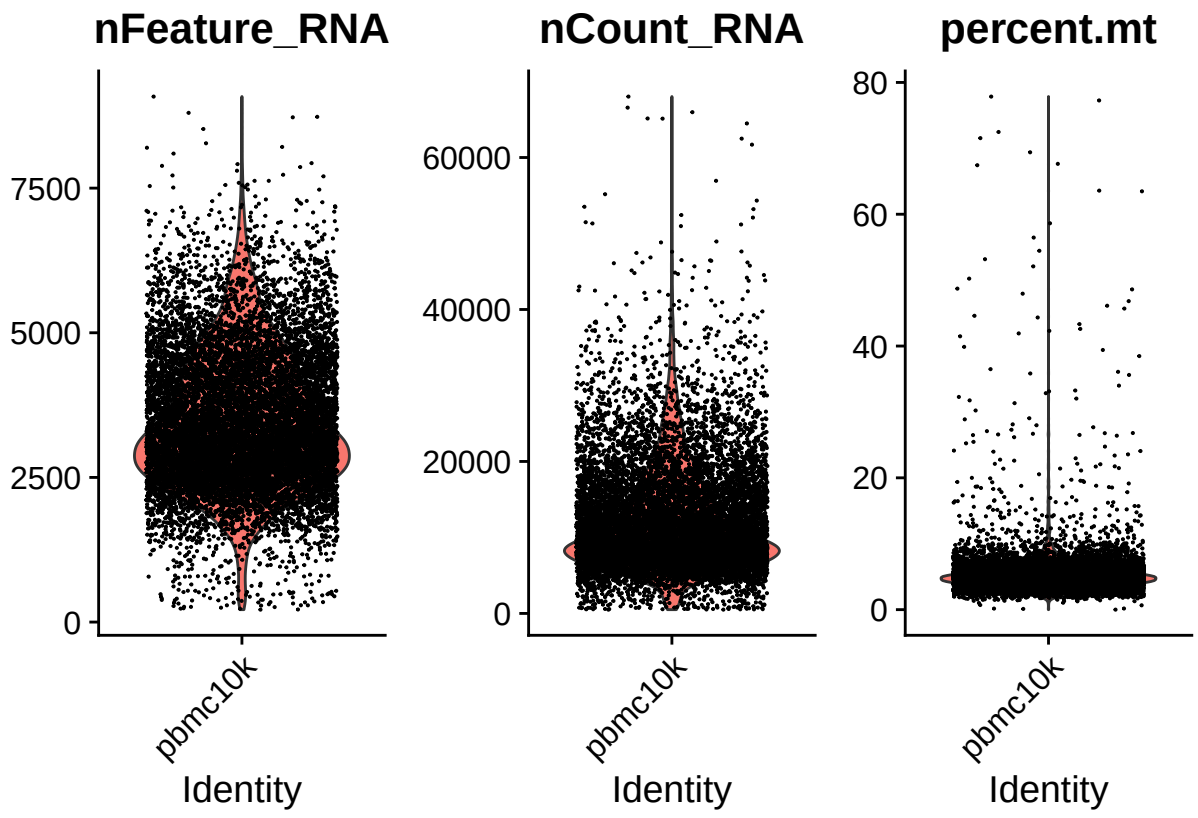
```
data <- Read10X(data.dir = "/Users/sravanisaadhu/Downloads/filtered_feature_bc_matrix/")
data <- CreateSeuratObject(counts = data, project = "pbmc10k", min.cells = 3, min.features = 200)
data
```

```
## An object of class Seurat
## 27363 features across 11952 samples within 1 assay
## Active assay: RNA (27363 features, 0 variable features)
```

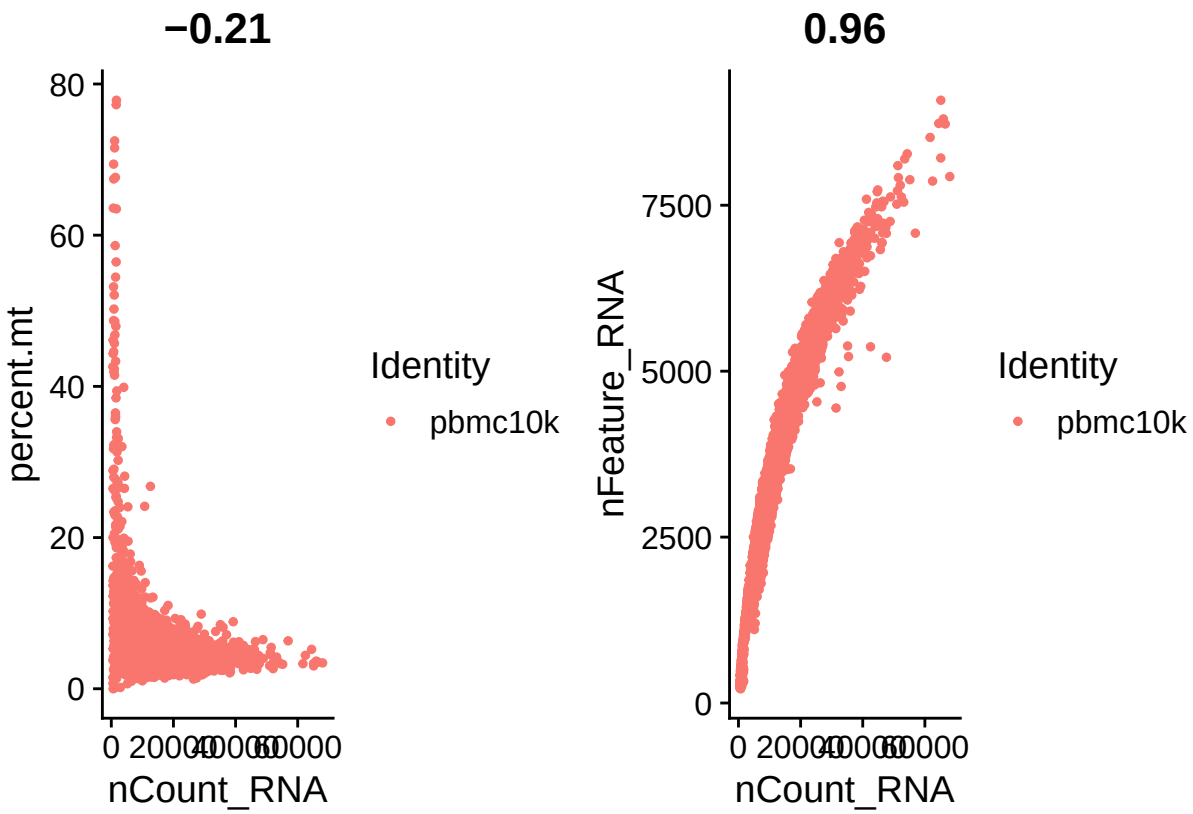
```
# The [[ operator can add columns to object metadata. This is a great place to stash QC stats
data[["percent.mt"]] <- PercentageFeatureSet(data, pattern = "^MT-")
# Show QC metrics for the first 5 cells
head(data@meta.data, 5)
```

```
##                      orig.ident nCount_RNA nFeature_RNA percent.mt
## AAACCCAAGGCCCAAA-1    pbmc10k      16164         4250   4.856471
## AAACCCAAGTAATACG-1    pbmc10k      11837         3784   4.181803
## AAACCCAAGTCACACT-1    pbmc10k      14493         4147   5.816601
## AAACCCACAAAGCGTG-1    pbmc10k       7684         2528   3.800104
## AAACCCACAATCGAAA-1    pbmc10k      11510         3493   5.699392
```

```
# Visualize QC metrics as a violin plot
VlnPlot(data, 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.

plot1 <- FeatureScatter(data, feature1 = "nCount_RNA", feature2 = "percent.mt")
plot2 <- FeatureScatter(data, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
plot1 + plot2
```




```
data <- subset(data, subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mt < 5)
```

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

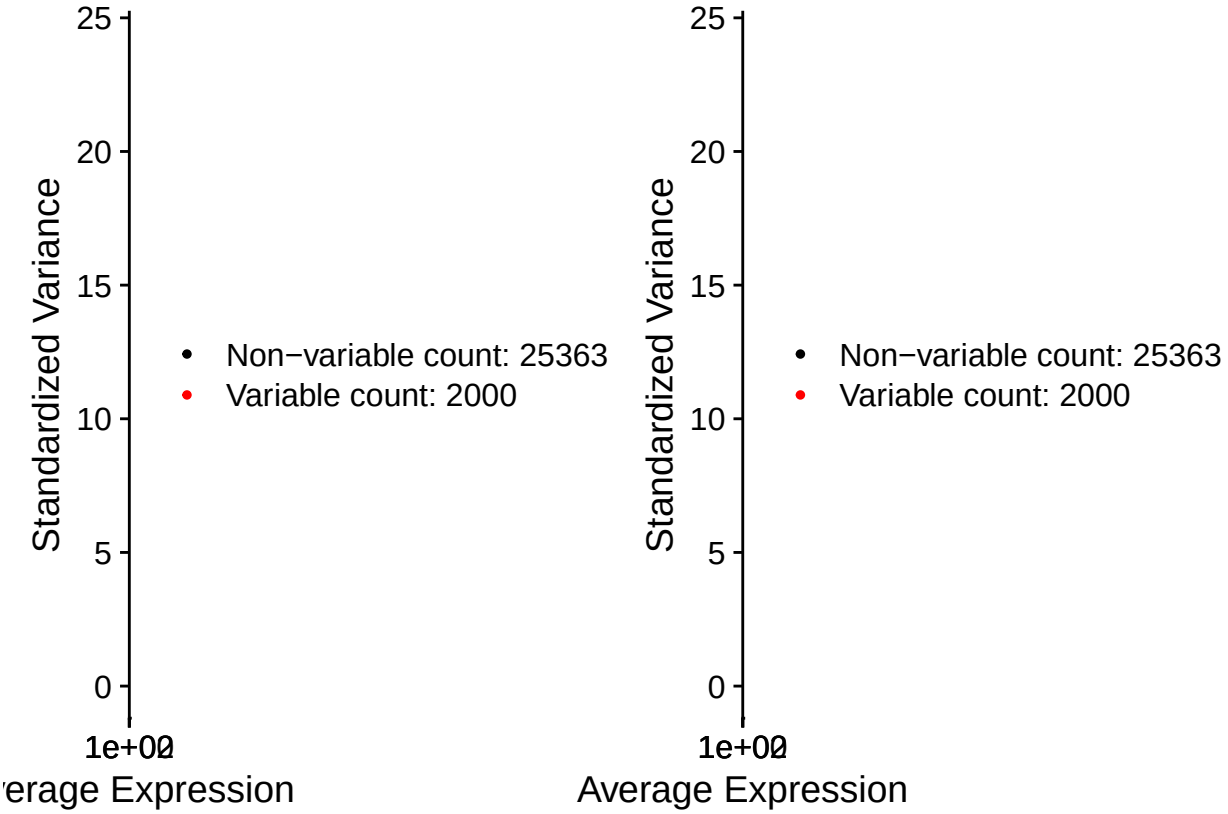
```
data <- FindVariableFeatures(data, selection.method = "vst", nfeatures = 2000)

# Identify the 50 most highly variable genes
top10 <- head(VariableFeatures(data), 10)

# plot variable features with and without labels
plot1 <- VariableFeaturePlot(data)
plot2 <- LabelPoints(plot1,geom_text_repel = TRUE,points=top10, xnudge = 0, ynudge = 0)
```

```
## Warning in geom.use(mapping = aes_string(x = xynames$x, y = xynames$y, label =
## "labels"), : Ignoring unknown parameters: 'geom_text_repel'
```

```
plot1 + plot2
```

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

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




```
all.genes <- rownames(data)
# data <- ScaleData(data, features = VariableFeatures(data)) skylar's note: this did not keep
# relevant genes in scale.data
data <- ScaleData(data, features = all.genes)
```

```
## Centering and scaling data matrix
```

```
data <- ScaleData(data, vars.to.regress = "percent.mt")
```

```
## Regressing out percent.mt

## Centering and scaling data matrix
```

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

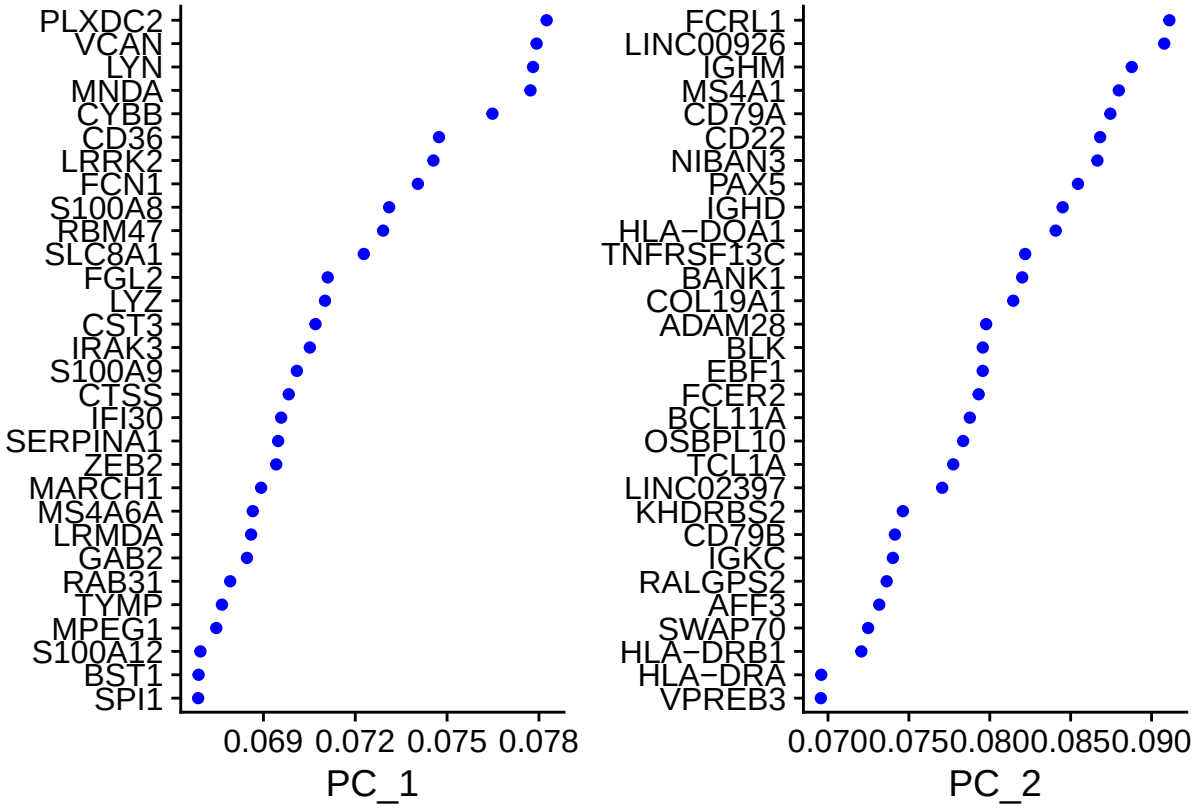
```
## PC_ 1
## Positive:  PLXDC2, VCAN, LYN, MNDA, CYBB, CD36, LRRK2, FCN1, S100A8, RBM47
##     SLC8A1, FGL2, LYZ, CST3, IRAK3, S100A9, CTSS, IFI30, SERPINA1, ZEB2
##     MARCH1, MS4A6A, LRMDA, GAB2, RAB31, TYMP, MPEG1, S100A12, BST1, SPI1
## Negative:  RPS12, LEF1, IL7R, CAMK4, EEF1A1, IL32, INPP4B, LTB, MALAT1, TXK
##     PRKCA, SERINC5, PCED1B, PDE3B, MLLT3, TRBC2, THEMIS, DOCK10, ANK3, FHIT
##     PITPNC1, RASGRF2, BACH2, TRBC1, BCL2, MAML2, AC139720.1, AK5, NR3C2, ZEB1
## PC_ 2
```

```
## Positive:  FCRL1, LINC00926, IGHM, MS4A1, CD79A, CD22, NIBAN3, PAX5, IGHD, HLA-DQA1
##     TNFRSF13C, BANK1, COL19A1, ADAM28, BLK, EBF1, FCER2, BCL11A, OSBPL10, TCL1A
##     LINC02397, KHDRBS2, CD79B, IGKC, RALGPS2, AFF3, SWAP70, HLA-DRB1, HLA-DRA, VPREB3
## Negative:  RGS18, NRGN, MPP1, RAB32, PRKAR2B, GMPR, ITGA2B, PPBP, GNG11, CAVIN2
##     PF4, TREML1, GP1BB, TRIM58, SPARC, CTDSPL, CMTM5, ACRBP, LMNA, TUBB1
##     MYL4, AC147651.1, ITGB3, DAB2, AP003068.2, CLEC1B, GP9, F13A1, ANKRD9, PTCRA
## PC_ 3
## Positive:  PDLIM1, TSPAN33, PPBP, PF4, TREML1, CAVIN2, GP1BB, GNG11, ITGA2B, AP003068.2
##     SPARC, CMTM5, MYL4, AC147651.1, ITGB3, PTCRA, TUBB1, TRIM58, CLEC1B, GP9
##     AP001636.3, TAL1, LGALSL, ANKRD9, CTDSPL, TMEM40, PRKAR2B, CLU, C2orf88, ACRBP
## Negative:  DPYD, S100A6, VCAN, FCN1, SLC8A1, FOS, RBM47, LRMDA, SERPINA1, FGL2
##     NAMPT, S100A8, NEAT1, MNDA, MALAT1, SLC11A1, TNFSF13B, ACSL1, ARHGAP26, DMXL2
##     AC020656.1, S100A4, NCF2, FNDC3B, CSF3R, DYSF, RGS2, GLT1D1, CEBPD, CSTA
## PC_ 4
## Positive:  NKG7, GNLY, CST7, FGFBP2, KLRD1, PRF1, GZMH, GZMA, GZMB, CCL4
##     ADGRG1, FCGR3A, KLRF1, HOPX, SPON2, CCL5, TGFBR3, FCRL6, CLIC3, TBX21
##     C1orf21, CTSW, CX3CR1, PTGDR, PRSS23, TTC38, PPP2R2B, TRDC, KLRB1, S1PR5
## Negative:  FOXP1, LEF1, MAML2, CAMK4, SERINC5, BACH2, PRKCA, IL7R, FHIT, PDE3B
##     IGF1R, LTB, RASGRF2, CERS6, INPP4B, ANK3, PLCL1, PCSK5, MLLT3, CSGALNACT1
##     FAM117B, AIF1, SSBP2, AC139720.1, RGS10, IMMP2L, NR3C2, GNAQ, NELL2, LEF1-AS1
## PC_ 5
## Positive:  MALAT1, FOXP1, DPYD, DOCK10, DOCK5, CMIP, PRKCA, AOAH, SUMF1, DENND1A
##     ATP8B4, FNDC3B, CERS6, PITPNC1, IGF1R, MAML2, COMMD10, ARHGAP26, DYSF, BACH2
##     PDE3B, SLC24A4, FAM117B, AP003086.1, ZSWIM6, DGKG, MIR646HG, TXK, ZCCHC7, AHCYL2
## Negative:  TMSB4X, EEF1A1, RPS12, FTH1, ACTB, FTL, SH3BGRL3, VIM, S100A6, OAZ1
##     S100A10, H3F3A, S100A11, S100A4, CRIP1, LTB, CST3, CLIC1, IGSF6, IL32
##     LST1, ANXA2, KCTD12, CFD, IFI30, AIF1, COTL1, CSTA, SERPINA1, CD68
```

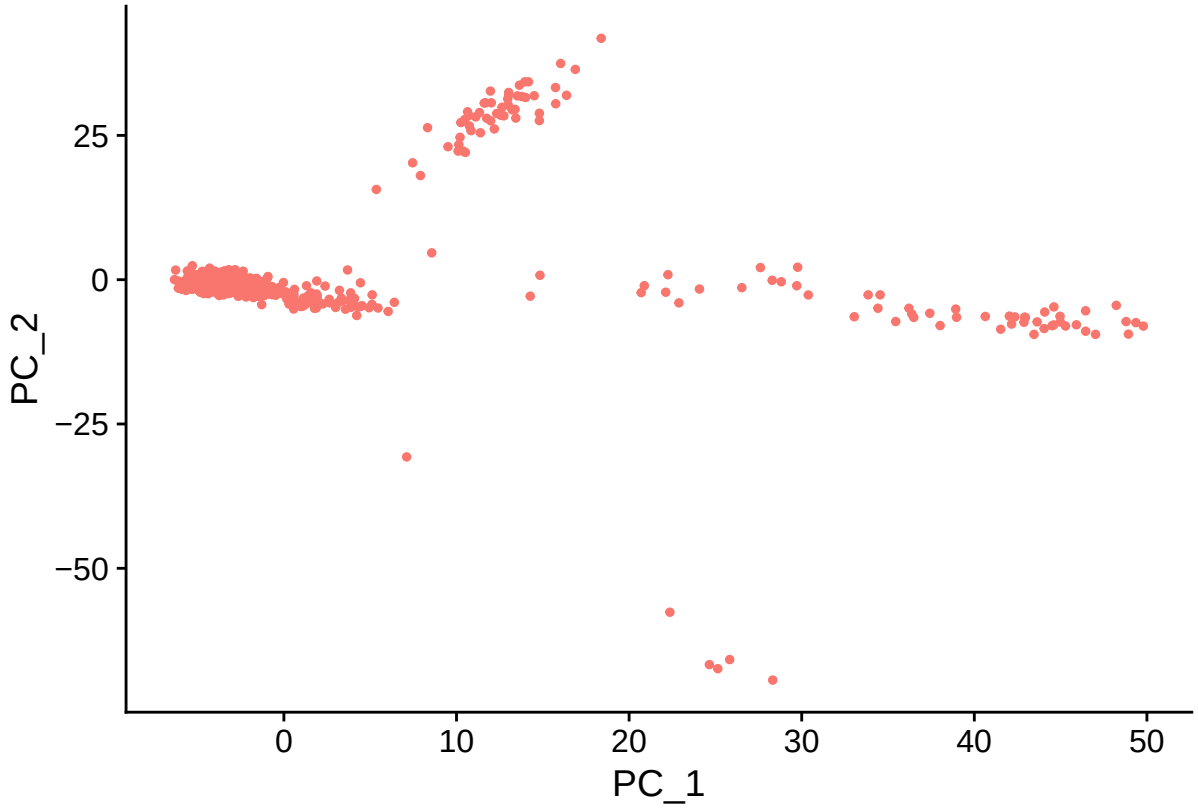
```
# Examine and visualize PCA results a few different ways
print(data[["pca"]], dims = 1:5, nfeatures = 5)
```

```
## PC_ 1
## Positive:  PLXDC2, VCAN, LYN, MNDA, CYBB
## Negative:  RPS12, LEF1, IL7R, CAMK4, EEF1A1
## PC_ 2
## Positive:  FCRL1, LINC00926, IGHM, MS4A1, CD79A
## Negative:  RGS18, NRGN, MPP1, RAB32, PRKAR2B
## PC_ 3
## Positive:  PDLIM1, TSPAN33, PPBP, PF4, TREML1
## Negative:  DPYD, S100A6, VCAN, FCN1, SLC8A1
## PC_ 4
## Positive:  NKG7, GNLY, CST7, FGFBP2, KLRD1
## Negative:  FOXP1, LEF1, MAML2, CAMK4, SERINC5
## PC_ 5
## Positive:  MALAT1, FOXP1, DPYD, DOCK10, DOCK5
## Negative:  TMSB4X, EEF1A1, RPS12, FTH1, ACTB
```

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

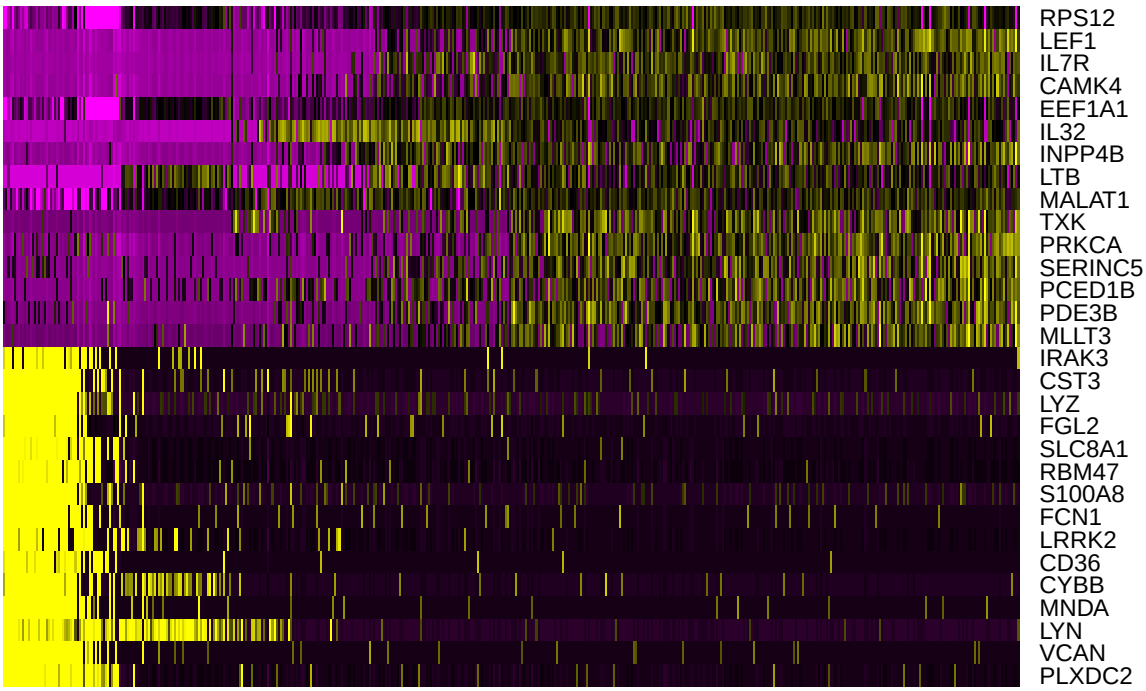



```
DimPlot(data, reduction = "pca") + NoLegend()
```

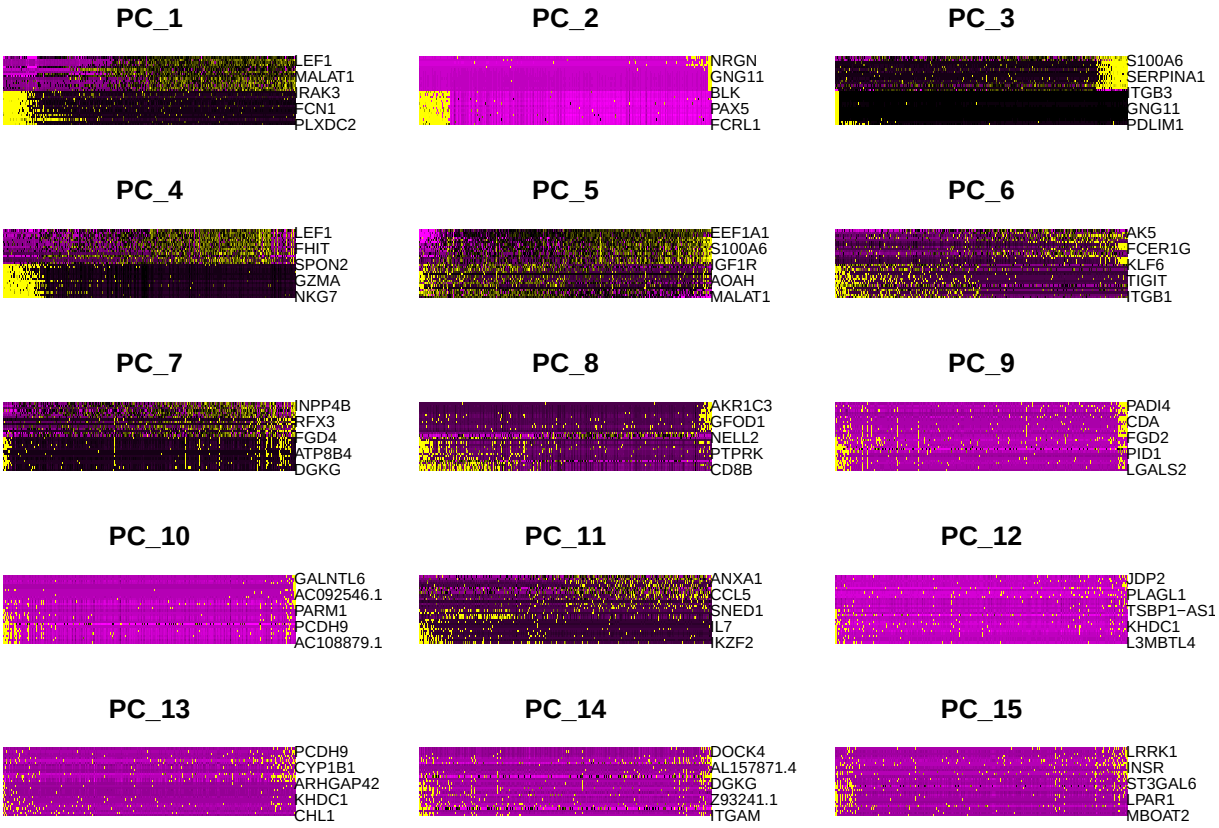



```
DimHeatmap(data, dims = 1, cells = 500, balanced = TRUE)
```

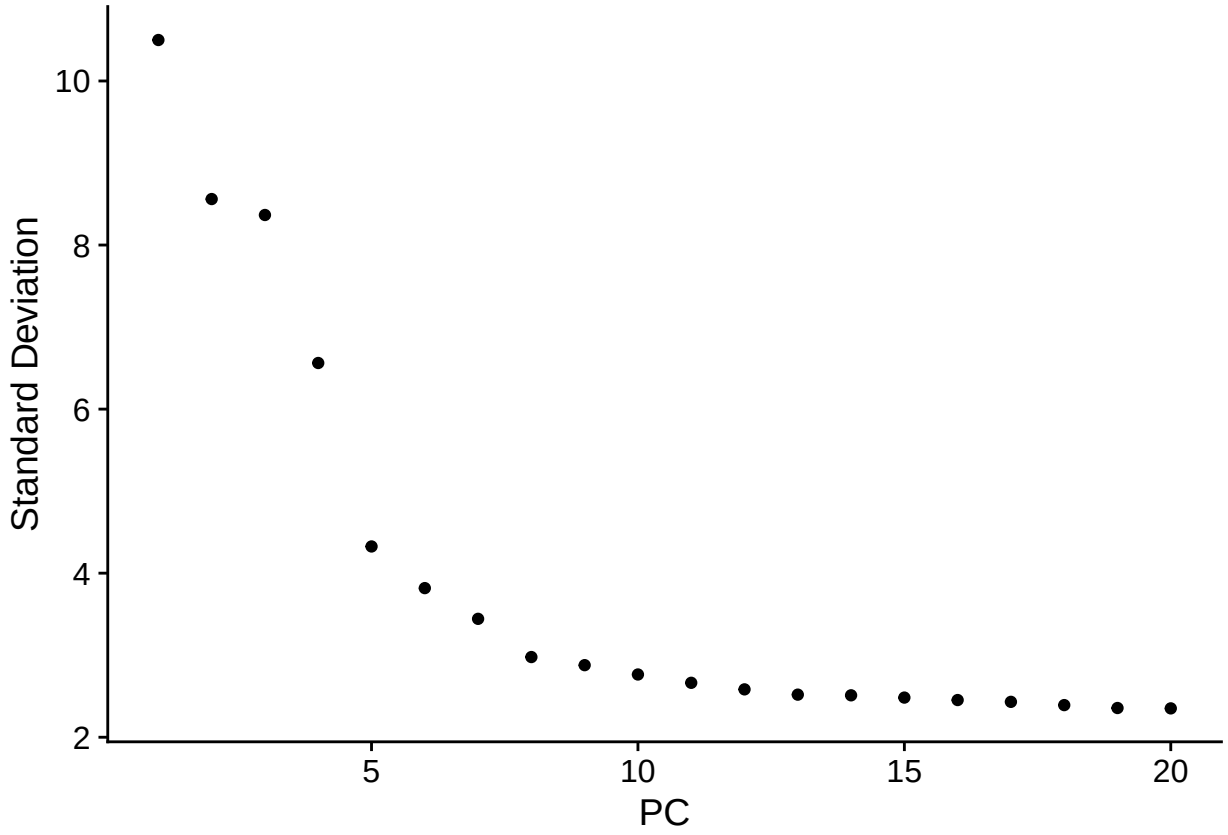
PC_1




```
DimHeatmap(data, dims = 1:15, cells = 500, balanced = TRUE)
```




```
ElbowPlot(data)
```




```
data <- FindNeighbors(data, dims = 1:10)
```

```
## Computing nearest neighbor graph

## Computing SNN
```

```
data <- FindClusters(data, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 964
## Number of edges: 30798
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8439
## Number of communities: 7
## Elapsed time: 0 seconds
```

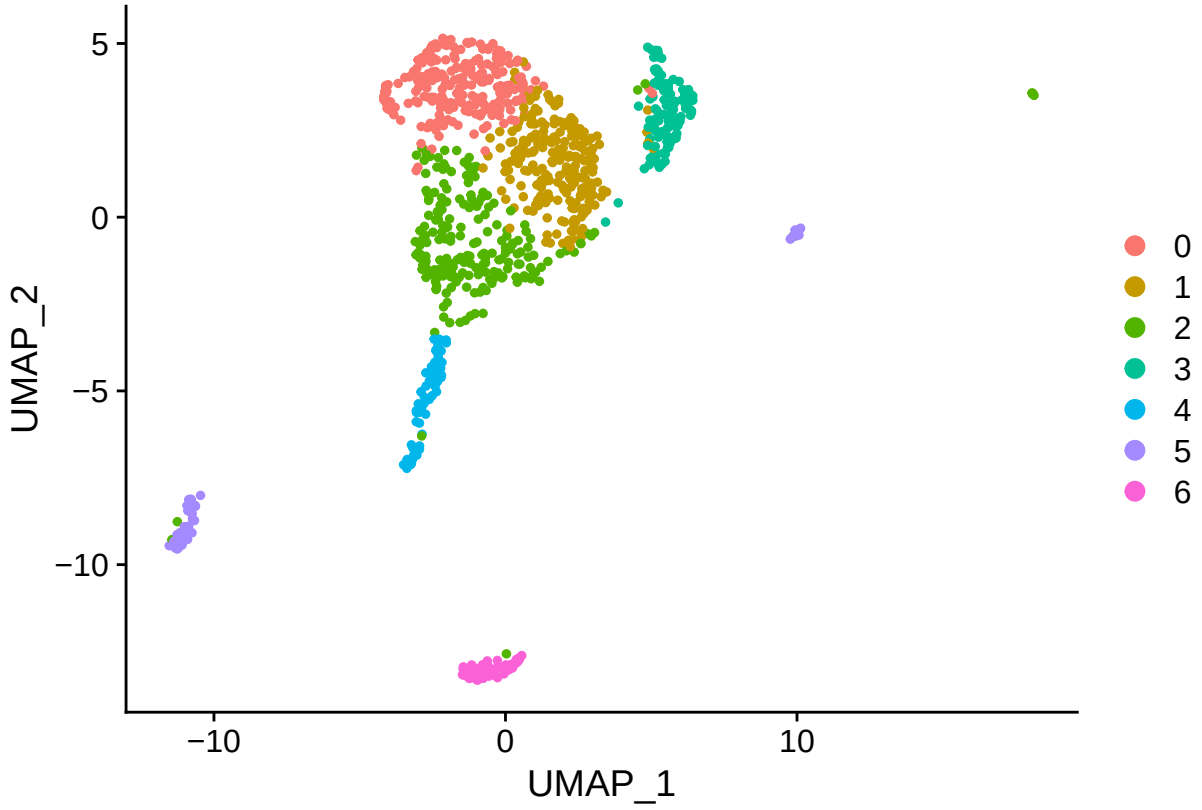
```
head(Idents(data), 5)
```

```
## AAAGGGCTCCCGGTAG-1 AAAGGTAAGTTGCCCG-1 AAAGGTACAATACAGA-1 AACAAAGAGATTAGAC-1
##                  1                  1                  1                  2
## AACAAAGAGCCATGCC-1
##                  4
## Levels: 0 1 2 3 4 5 6
```

```r
data <- RunUMAP(data, dims = 1:10)
```

```
## 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

## 16:56:39 UMAP embedding parameters a = 0.9922 b = 1.112

## 16:56:39 Read 964 rows and found 10 numeric columns

## 16:56:39 Using Annoy for neighbor search, n_neighbors = 30

## 16:56:39 Building Annoy index with metric = cosine, n_trees = 50

## 0%   10   20   30   40   50   60   70   80   90   100%

## [----|----|----|----|----|----|----|----|----|----|

## **************************************************|
## 16:56:39 Writing NN index file to temp file /var/folders/qp/z26w0hw109s4hfw__r7k7spm0000gn/T//RtmpDg
## 16:56:39 Searching Annoy index using 1 thread, search_k = 3000
## 16:56:39 Annoy recall = 100%
## 16:56:39 Commencing smooth kNN distance calibration using 1 thread with target n_neighbors = 30
## 16:56:40 Initializing from normalized Laplacian + noise (using irlba)
## 16:56:40 Commencing optimization for 500 epochs, with 35724 positive edges
## 16:56:41 Optimization finished
```

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




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

```
##              p_val avg_log2FC pct.1 pct.2    p_val_adj
## ITGB1 4.164971e-60  1.7051147 0.730 0.201 1.139661e-55
## BACH2 1.090860e-37 -1.6130838 0.495 0.819 2.984921e-33
## IL2RA 2.018973e-37  0.9793651 0.297 0.023 5.524515e-33
## B2M   7.193261e-36  0.5469399 0.995 0.995 1.968292e-31
## MAF   2.904367e-34  0.7132502 0.315 0.035 7.947218e-30
```

```
# find all markers distinguishing cluster 5 from clusters 0 and 3
cluster5.markers <- FindMarkers(data, ident.1 = 5, ident.2 = c(0, 3))
head(cluster5.markers, n = 5)
```

```
##               p_val avg_log2FC pct.1 pct.2    p_val_adj
## SLC8A1 2.083081e-68   4.184051 0.925 0.026 5.699934e-64
## RBM47  4.553611e-68   3.759641 0.887 0.018 1.246005e-63
## LYN    9.960032e-67   3.872754 0.943 0.038 2.725363e-62
## PLXDC2 4.935637e-66   4.590471 0.962 0.047 1.350538e-61
## ZEB2   8.193995e-66   4.732464 0.925 0.035 2.242123e-61
```

```
# find markers for every cluster compared to all remaining cells, report only the positive
# ones
data.markers <- FindAllMarkers(data, only.pos = TRUE)
```

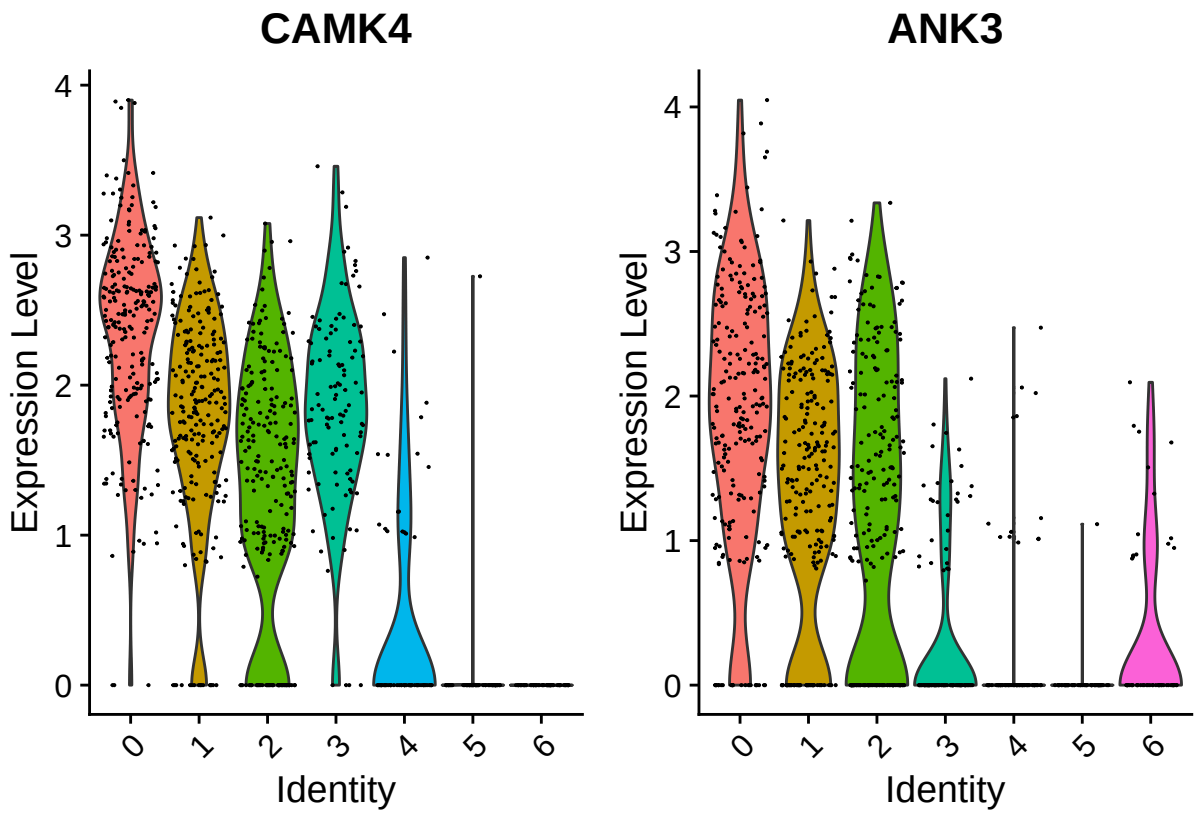
```
## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6
```

```
data.markers %>%
    group_by(cluster) %>%
    dplyr::filter(avg_log2FC > 1)
```

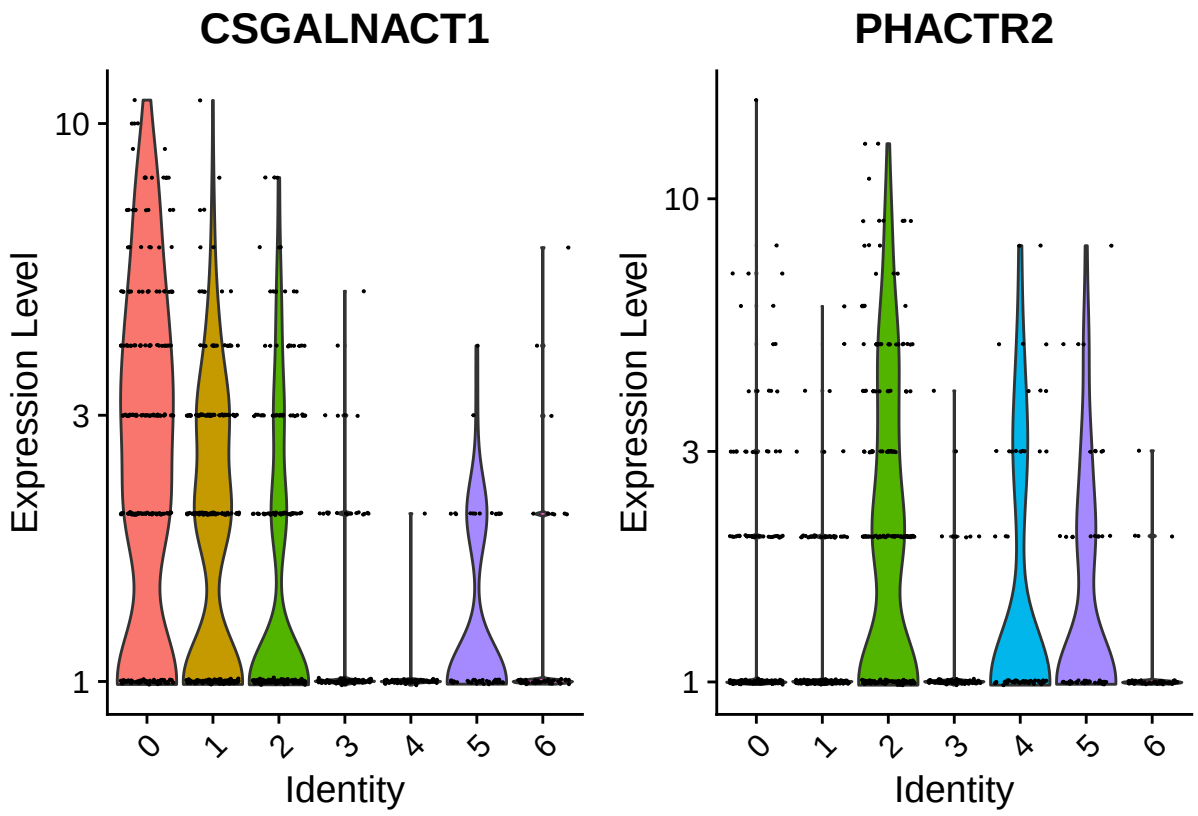
```
## # A tibble: 998 x 7
## # Groups:   cluster [6]
##       p_val avg_log2FC pct.1 pct.2 p_val_adj cluster gene
##       <dbl>      <dbl> <dbl> <dbl>     <dbl> <fct>   <chr>
##  1 5.12e-50       1.25 0.987 0.706  1.40e-45 0       CAMK4
##  2 2.52e-46       1.46 0.948 0.574  6.88e-42 0       FHIT
##  3 1.19e-41       1.19 0.987 0.695  3.27e-37 0       LEF1
##  4 3.11e-40       1.12 0.978 0.653  8.50e-36 0       PRKCA
##  5 7.26e-36       1.08 0.991 0.827  1.99e-31 0       MAML2
##  6 3.60e-35       1.28 0.897 0.531  9.84e-31 0       ANK3
##  7 8.95e-33       1.27 0.728 0.348  2.45e-28 0       PLCL1
##  8 2.40e-31       1.09 0.767 0.376  6.58e-27 0       IGF1R
##  9 2.43e-27       1.00 0.69  0.329  6.65e-23 0       ATP10A
## 10 1.45e-25       1.06 0.707 0.357  3.97e-21 0       CSGALNACT1
## # i 988 more rows
```

```
cluster0.markers <- FindMarkers(data, ident.1 = 0, logfc.threshold = 0.25, test.use = "roc", only.pos =
```

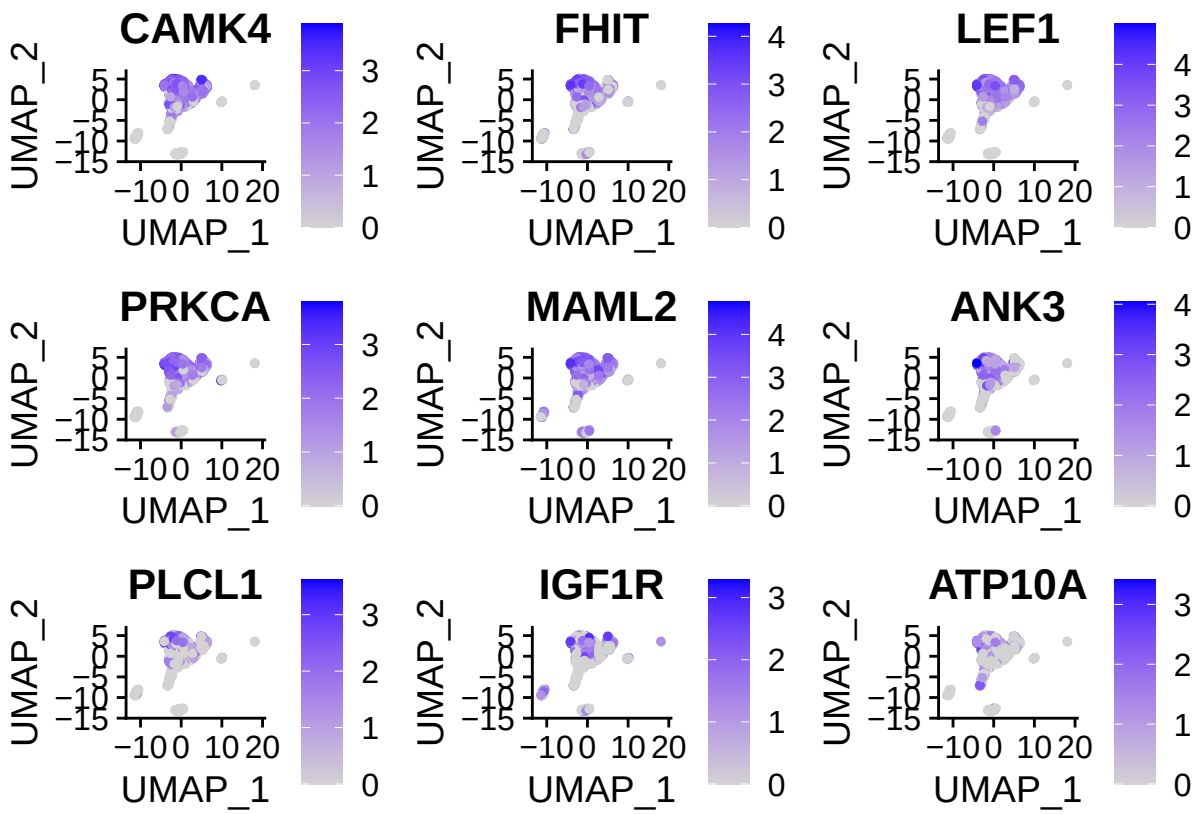
```
VlnPlot(data, features = c("CAMK4", "ANK3"))
```



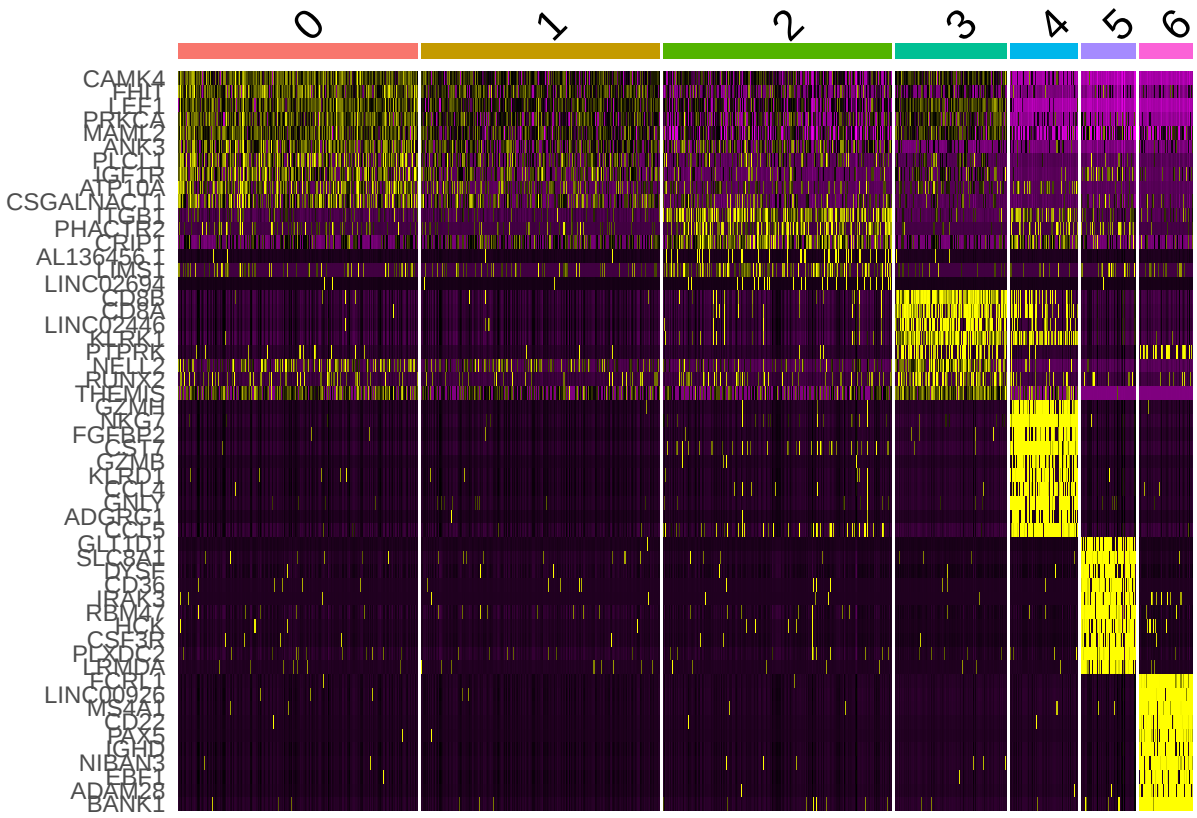

```
# you can plot raw counts as well
VlnPlot(data, features = c("CSGALNACT1", "PHACTR2"), slot = "counts", log = TRUE)
```




```
FeaturePlot(data, features = c("CAMK4", "FHIT", "LEF1", "PRKCA", "MAML2", "ANK3", "PLCL1", "IGF1R", "ATI
```



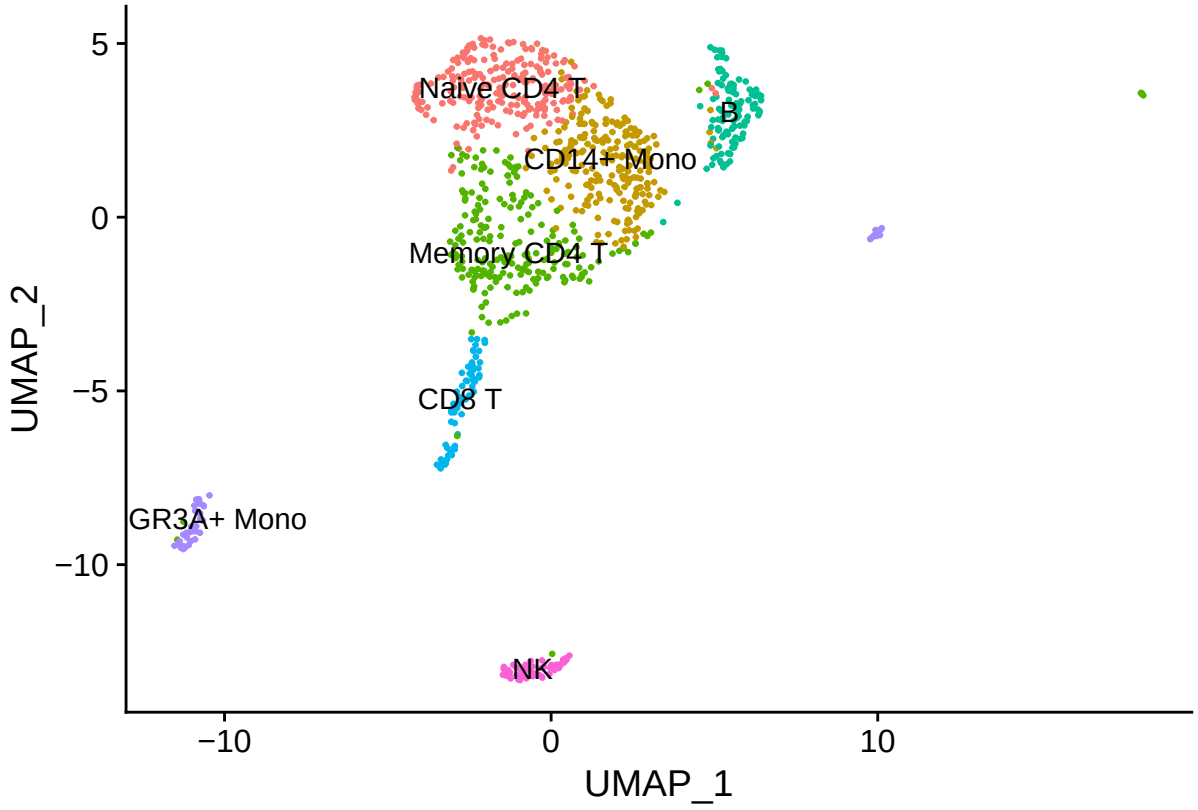

```
data.markers %>%
    group_by(cluster) %>%
    dplyr::filter(avg_log2FC > 1) %>%
    slice_head(n = 10) %>%
    ungroup() -> top10
DoHeatmap(data, features = top10$gene) + NoLegend()
```



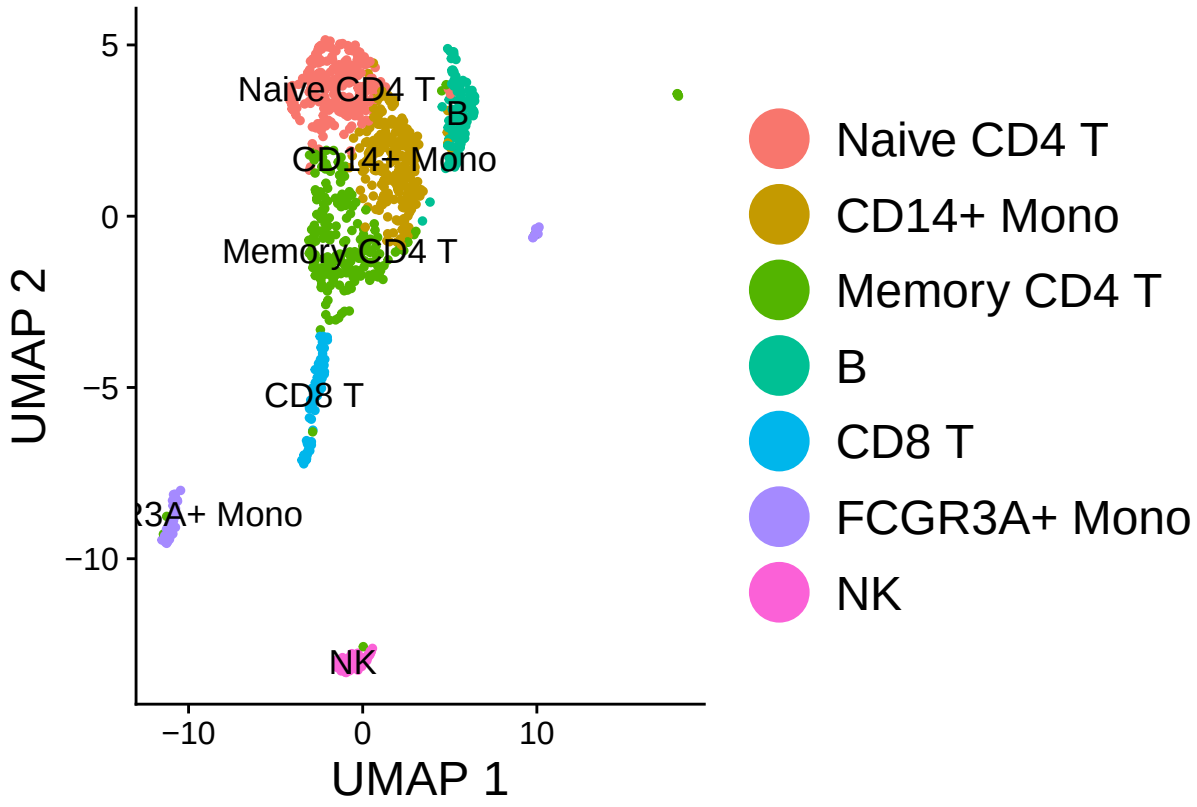

```
new.cluster.ids <- c("Naive CD4 T", "CD14+ Mono", "Memory CD4 T", "B", "CD8 T", "FCGR3A+ Mono", "NK", "
names(new.cluster.ids) <- levels(data)
data <- RenameIdents(data, new.cluster.ids)
```

```
## Warning: Cannot find identity NA

## Warning: Cannot find identity NA
```

```
DimPlot(data, reduction = "umap", label = TRUE, pt.size = 0.5) + NoLegend()
```




```
library(ggplot2)
DimPlot(data, reduction = "umap", label = TRUE, label.size = 4.5) + xlab("UMAP 1") + ylab("UMAP 2") + tl
```


Naive CD4 T
B
CD14+ Mono
Memory CD4 T
CD8 T
R3A+ Mono
NK
5
0
−5
−10
−10
0
10
UMAP 2
UMAP 1
Naive CD4 T
CD14+ Mono
Memory CD4 T
B
CD8 T
FCGR3A+ Mono
NK