Complete scRNAseq Workflow

Asad

2023-05-27

Call libraries and load data

```
library(Seurat)
```

```
## Attaching SeuratObject
```

```
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
```

```
library(patchwork)
library(Matrix)
library(ggplot2)
library(reticulate)
library(viridis)
```

```
## Loading required package: viridisLite
```

```
library(ggmin)
library(DESeq2)
```

```
## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:Matrix':
##
##     expand, unname

## The following objects are masked from 'package:dplyr':
##
##     first, rename

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

## The following object is masked from 'package:grDevices':
##
##     windows

## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb
```

```
## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'matrixStats'

## The following object is masked from 'package:dplyr':
##
##     count

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians
```

```
##
## Attaching package: 'SummarizedExperiment'

## The following object is masked from 'package:SeuratObject':
##
##     Assays

## The following object is masked from 'package:Seurat':
##
##     Assays
```

```
library(colorspace)
library(RColorBrewer)
library(ggpubr)
library(SingleR)
library(celldex)
```

```
##
## Attaching package: 'celldex'

## The following objects are masked from 'package:SingleR':
##
##     BlueprintEncodeData, DatabaseImmuneCellExpressionData,
##     HumanPrimaryCellAtlasData, ImmGenData, MonacoImmuneData,
##     MouseRNAseqData, NovershternHematopoieticData
```

```
library(pheatmap)
library(DoubletFinder)

# Load the PBMC dataset
pbmc.data <- Read10X(data.dir = "E:/scRNA-seq/Clustering and Marker Identification")
#str(pbmc.data)
```

Initialize the Seurat object with the raw (non-normalized data).

```
pbmc <- CreateSeuratObject(counts = pbmc.data, project = "pbmc3k", min.cells = 3, min.features = 200)
```

```
## Warning: Feature names cannot have underscores ('_'), replacing with dashes
## ('-')
```

```
#pbmc
#str(pbmc)
```

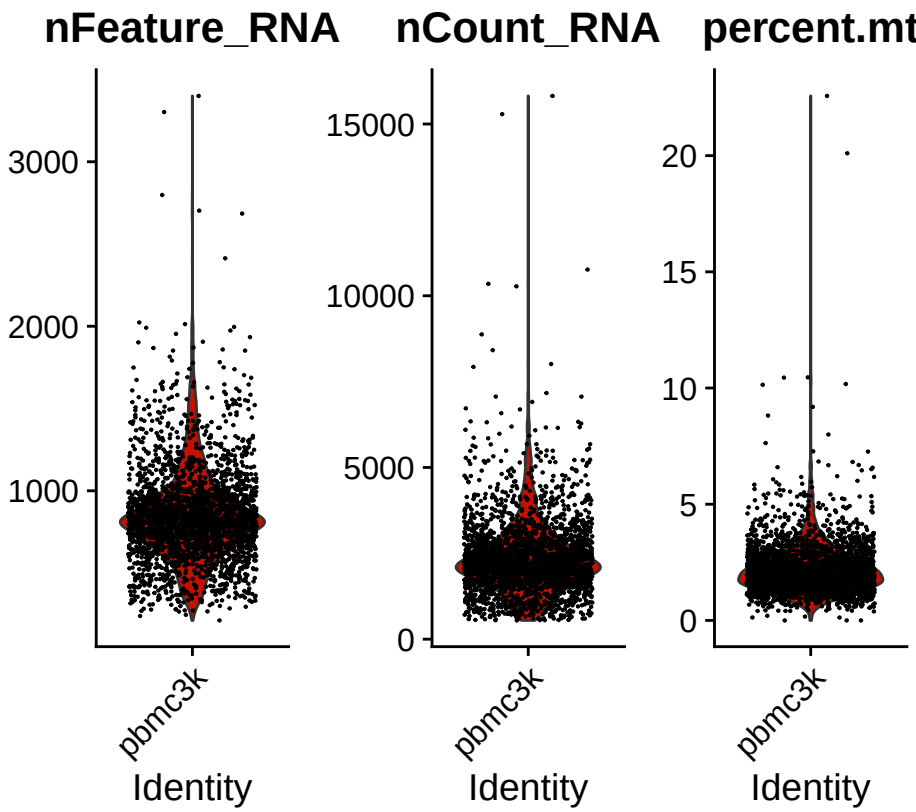
Quality assessment

Note MT: Bar-codes with a low count depth, few detected genes, and a high fraction of mitochondrial counts are indicative of cells whose cytoplasmic mRNA has leaked out through a broken membrane, and thus, only mRNA located in the mitochondria is still conserved.

Note RB: Here RB means ribosomal protein and RNA coding genes. High number of RB genes reflect metabolically active cells. Based on cell types 15-45% RB genes can be mapped. Based on requirement these genes can be removed (i.e., these genes don't contribute to heterogeneity).

```
#Check mitochondrial RNA percentage
pbmc[["percent.mt"]] <- PercentageFeatureSet(pbmc, pattern = "^MT-")

#Check ribosomal RNA percentage
pbmc[["percent.rb"]] <- PercentageFeatureSet(pbmc, pattern = "^RP[SL]")
View(pbmc@meta.data)
```

```
VlnPlot(pbmc, features = c("nFeature_RNA", "nCount_RNA", "percent.mt",
                           'percent.rb'), ncol = 4,
        cols = '#C71000')
```

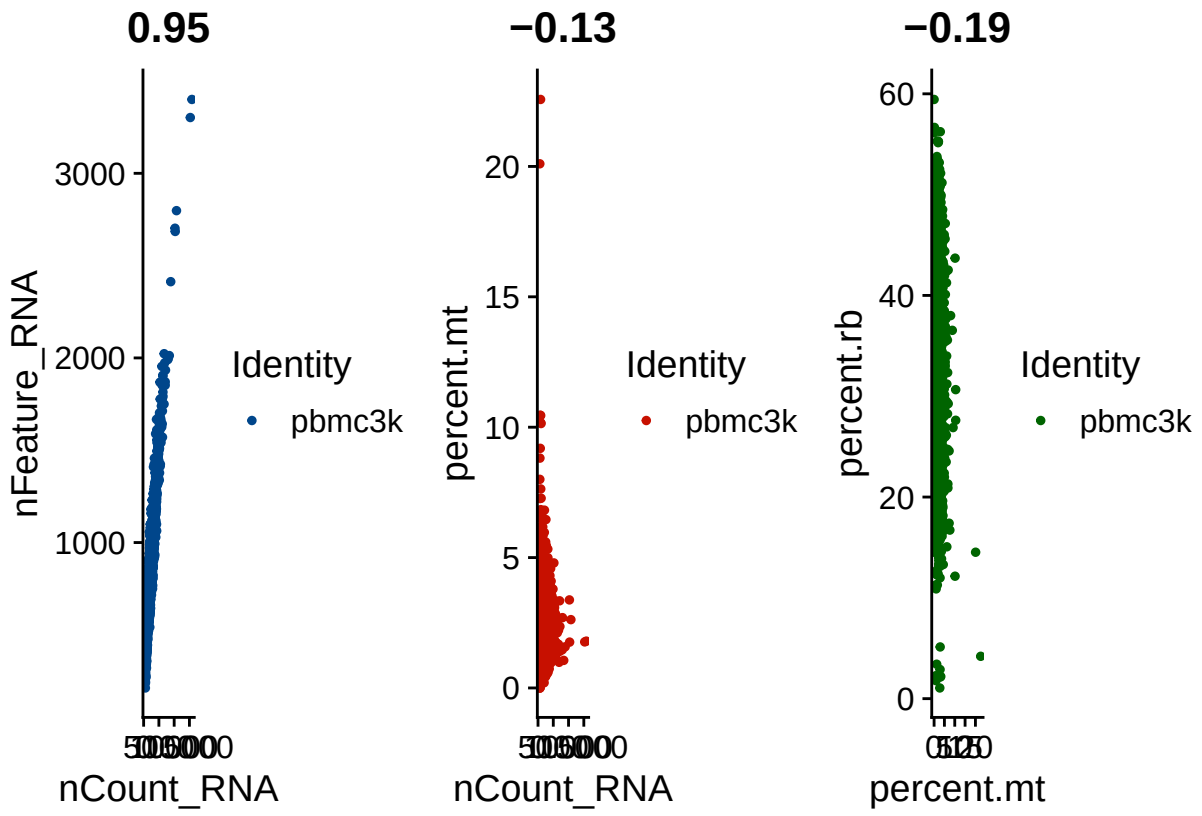


Visualize QC metrics as a violin plot

Assess correlation Result is in Pearson-rank

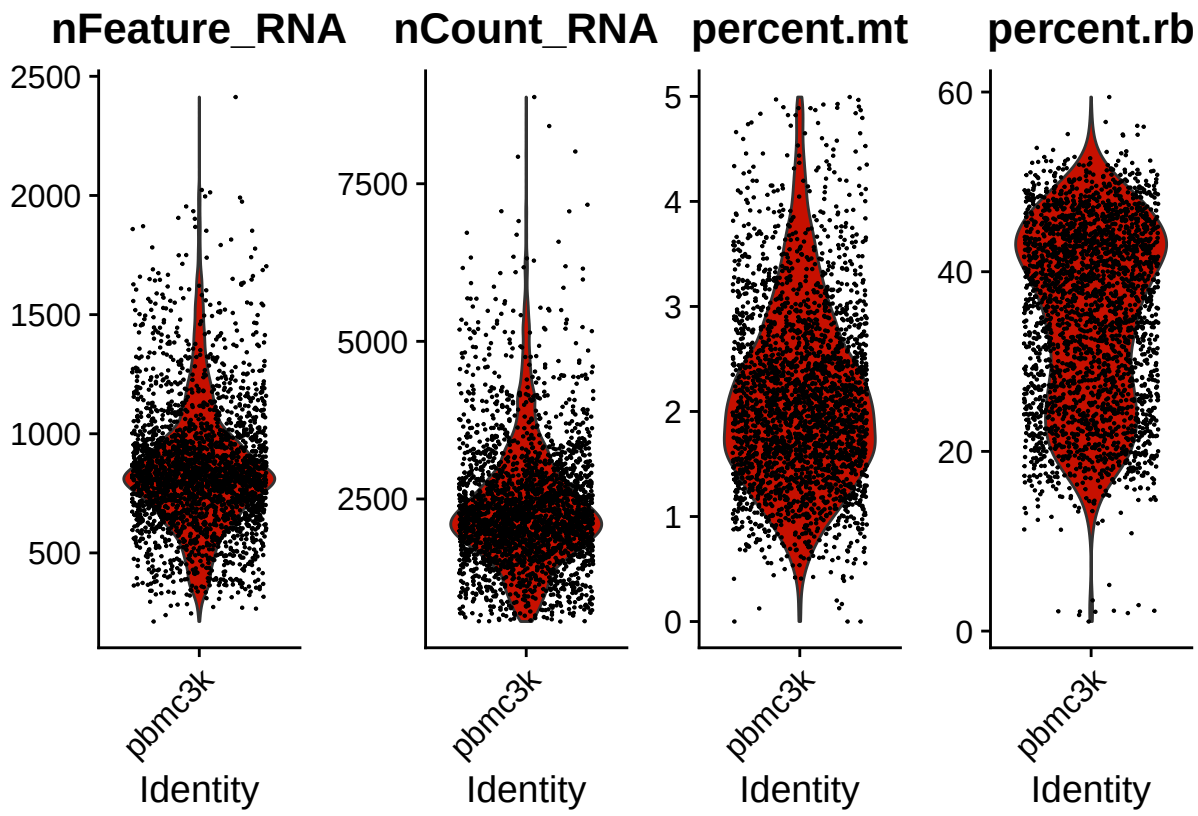
```
plot1 <- FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "nFeature_RNA",
                        cols = '#00468B', plot.cor = T, jitter = T, pt.size = 1)
plot2 <- FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "percent.mt",
                        cols = '#C71000', plot.cor = T, jitter = T, pt.size = 1)
plot3 <- FeatureScatter(pbmc, feature1 = "percent.mt", feature2 = "percent.rb",
                        cols = 'darkgreen', plot.cor = T, jitter = T, pt.size = 1)
```

```
plot1 + plot2 + plot3
```



Filtering Discard cells having unique feature counts over 2,500 and less than 200. Also filter out cells having >5% mitochondrial counts

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

# Let's visualize again after QC
VlnPlot(pbmc, features = c("nFeature_RNA", "nCount_RNA", "percent.mt",
                           "percent.rb"), ncol = 4, cols = '#C71000')
```



Normalizing data

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

```
pbmc <- NormalizeData(pbmc)
```

Alternative way that achives the same attribute as above method

Identify top variable features (2000 default for a single dataset)

```
pbmc <- FindVariableFeatures(pbmc, selection.method = "vst")
```

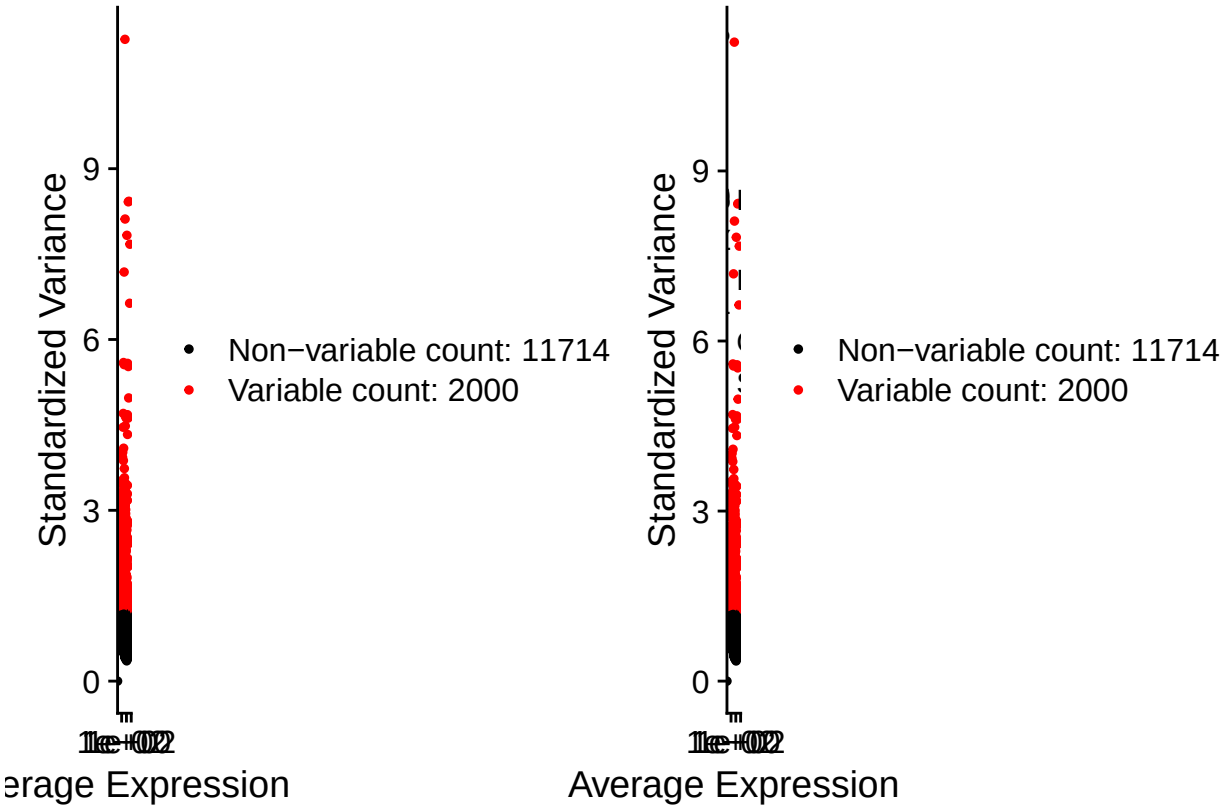
```
top10 <- head(VariableFeatures(pbmc), 10)

#plot variable features with and without labels
plot1 <- VariableFeaturePlot(pbmc)
plot2 <- LabelPoints(plot = plot1, points = top10, repel = TRUE)
```

Identify the 10 most highly variable genes

```
## When using repel, set xnudge and ynudge to 0 for optimal results
```

```
plot1 + plot2
```

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



Scaling the dataset

Scaling on large dataset may require extra time. An alternative way to avoid the time lag by applying scaling only to the variable Features identified previously (i.e., 2000 in this case) by the following way

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

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

```
#Alternative way
#pbmc <- ScaleData(pbmc)
```

Performing dimensionality reduction for PCA

```
pbmc <- RunPCA(object = pbmc)
```

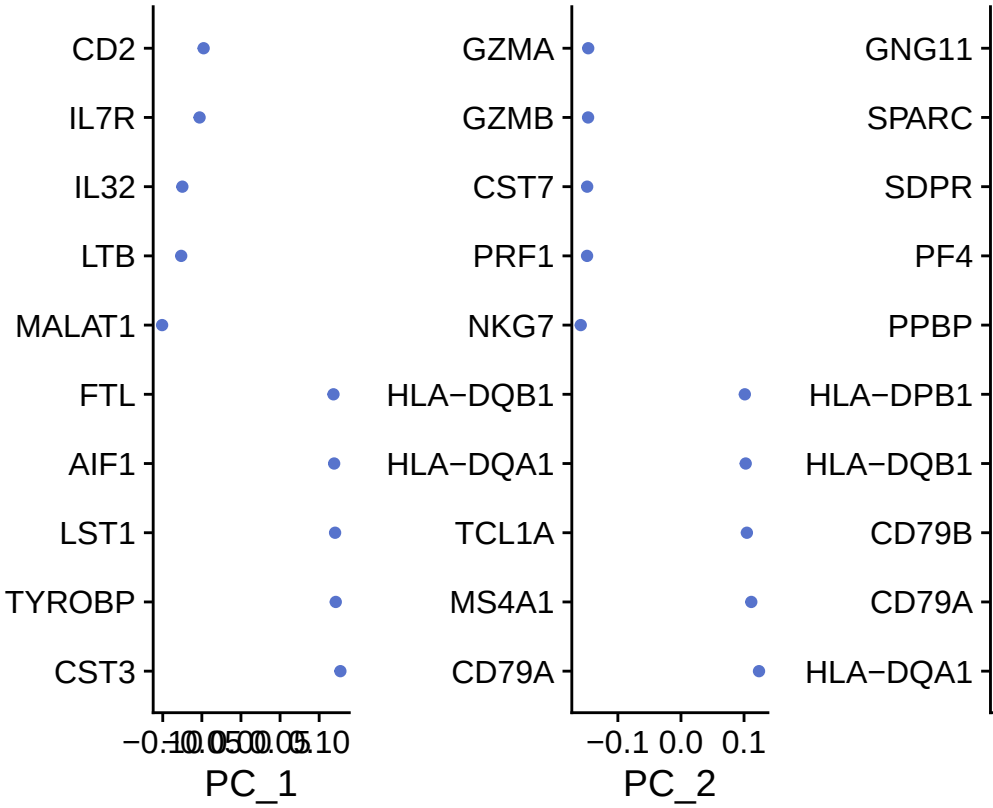
```
## PC_ 1
## Positive:  CST3, TYROBP, LST1, AIF1, FTL, FTH1, LYZ, FCN1, S100A9, TYMP
##     FCER1G, CFD, LGALS1, S100A8, CTSS, LGALS2, SERPINA1, IFITM3, SPI1, CFP
##     PSAP, IFI30, SAT1, COTL1, S100A11, NPC2, GRN, LGALS3, GSTP1, PYCARD
## Negative:  MALAT1, LTB, IL32, IL7R, CD2, B2M, ACAP1, CD27, STK17A, CTSW
##     CD247, GIMAP5, AQP3, CCL5, SELL, TRAF3IP3, GZMA, MAL, CST7, ITM2A
##     MYC, GIMAP7, HOPX, BEX2, LDLRAP1, GZMK, ETS1, ZAP70, TNFAIP8, RIC3
## PC_ 2
## Positive:  CD79A, MS4A1, TCL1A, HLA-DQA1, HLA-DQB1, HLA-DRA, LINC00926, CD79B, HLA-DRB1, CD74
##     HLA-DMA, HLA-DPB1, HLA-DQA2, CD37, HLA-DRB5, HLA-DMB, HLA-DPA1, FCRLA, HVCN1, LTB
##     BLNK, P2RX5, IGLL5, IRF8, SWAP70, ARHGAP24, FCGR2B, SMIM14, PPP1R14A, C16orf74
## Negative:  NKG7, PRF1, CST7, GZMB, GZMA, FGFBP2, CTSW, GNLY, B2M, SPON2
##     CCL4, GZMH, FCGR3A, CCL5, CD247, XCL2, CLIC3, AKR1C3, SRGN, HOPX
##     TTC38, APMAP, CTSC, S100A4, IGFBP7, ANXA1, ID2, IL32, XCL1, RHOC
## PC_ 3
## Positive:  HLA-DQA1, CD79A, CD79B, HLA-DQB1, HLA-DPB1, HLA-DPA1, CD74, MS4A1, HLA-DRB1, HLA-DRA
##     HLA-DRB5, HLA-DQA2, TCL1A, LINC00926, HLA-DMB, HLA-DMA, CD37, HVCN1, FCRLA, IRF8
##     PLAC8, BLNK, MALAT1, SMIM14, PLD4, LAT2, IGLL5, P2RX5, SWAP70, FCGR2B
## Negative:  PPBP, PF4, SDPR, SPARC, GNG11, NRGN, GP9, RGS18, TUBB1, CLU
##     HIST1H2AC, AP001189.4, ITGA2B, CD9, TMEM40, PTCRA, CA2, ACRBP, MMD, TREML1
##     NGFRAP1, F13A1, SEPT5, RUFY1, TSC22D1, MPP1, CMTM5, RP11-367G6.3, MYL9, GP1BA
## PC_ 4
## Positive:  HLA-DQA1, CD79B, CD79A, MS4A1, HLA-DQB1, CD74, HLA-DPB1, HIST1H2AC, PF4, TCL1A
##     SDPR, HLA-DPA1, HLA-DRB1, HLA-DQA2, HLA-DRA, PPBP, LINC00926, GNG11, HLA-DRB5, SPARC
##     GP9, AP001189.4, CA2, PTCRA, CD9, NRGN, RGS18, GZMB, CLU, TUBB1
## Negative:  VIM, IL7R, S100A6, IL32, S100A8, S100A4, GIMAP7, S100A10, S100A9, MAL
##     AQP3, CD2, CD14, FYB, LGALS2, GIMAP4, ANXA1, CD27, FCN1, RBP7
##     LYZ, S100A11, GIMAP5, MS4A6A, S100A12, FOLR3, TRABD2A, AIF1, IL8, IFI6
## PC_ 5
## Positive:  GZMB, NKG7, S100A8, FGFBP2, GNLY, CCL4, CST7, PRF1, GZMA, SPON2
##     GZMH, S100A9, LGALS2, CCL3, CTSW, XCL2, CD14, CLIC3, S100A12, CCL5
##     RBP7, MS4A6A, GSTP1, FOLR3, IGFBP7, TYROBP, TTC38, AKR1C3, XCL1, HOPX
## Negative:  LTB, IL7R, CKB, VIM, MS4A7, AQP3, CYTIP, RP11-290F20.3, SIGLEC10, HMOX1
##     PTGES3, LILRB2, MAL, CD27, HN1, CD2, GDI2, ANXA5, CORO1B, TUBA1B
##     FAM110A, ATP1A1, TRADD, PPA1, CCDC109B, ABRACL, CTD-2006K23.1, WARS, VMO1, FYB
```

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

```
## PC_ 1
## Positive:  CST3, TYROBP, LST1, AIF1, FTL
## Negative:  MALAT1, LTB, IL32, IL7R, CD2
## PC_ 2
## Positive:  CD79A, MS4A1, TCL1A, HLA-DQA1, HLA-DQB1
## Negative:  NKG7, PRF1, CST7, GZMB, GZMA
## PC_ 3
## Positive:  HLA-DQA1, CD79A, CD79B, HLA-DQB1, HLA-DPB1
## Negative:  PPBP, PF4, SDPR, SPARC, GNG11
```

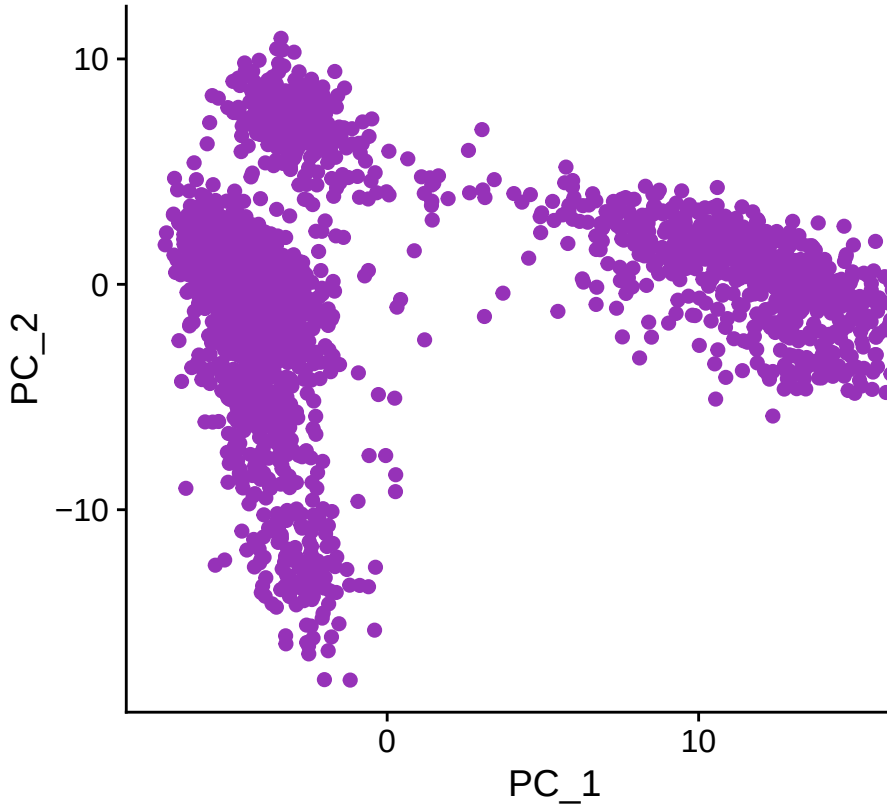
```
## PC_ 4
## Positive:  HLA-DQA1, CD79B, CD79A, MS4A1, HLA-DQB1
## Negative:  VIM, IL7R, S100A6, IL32, S100A8
## PC_ 5
## Positive:  GZMB, NKG7, S100A8, FGFBP2, GNLY
## Negative:  LTB, IL7R, CKB, VIM, MS4A7
```

```
VizDimLoadings(pbmc,
               dims = 1:3, #Number of dimensions to display
               reduction = "pca",
               col = '#5773CC',
               nfeatures = 10, #Number of features to display
               combine = T, #Combine all PCs
               ncol = 3, #Number of columns in figure
               balanced = T # Split total n (features) to + and -
               )
```



Visualize different dimensions

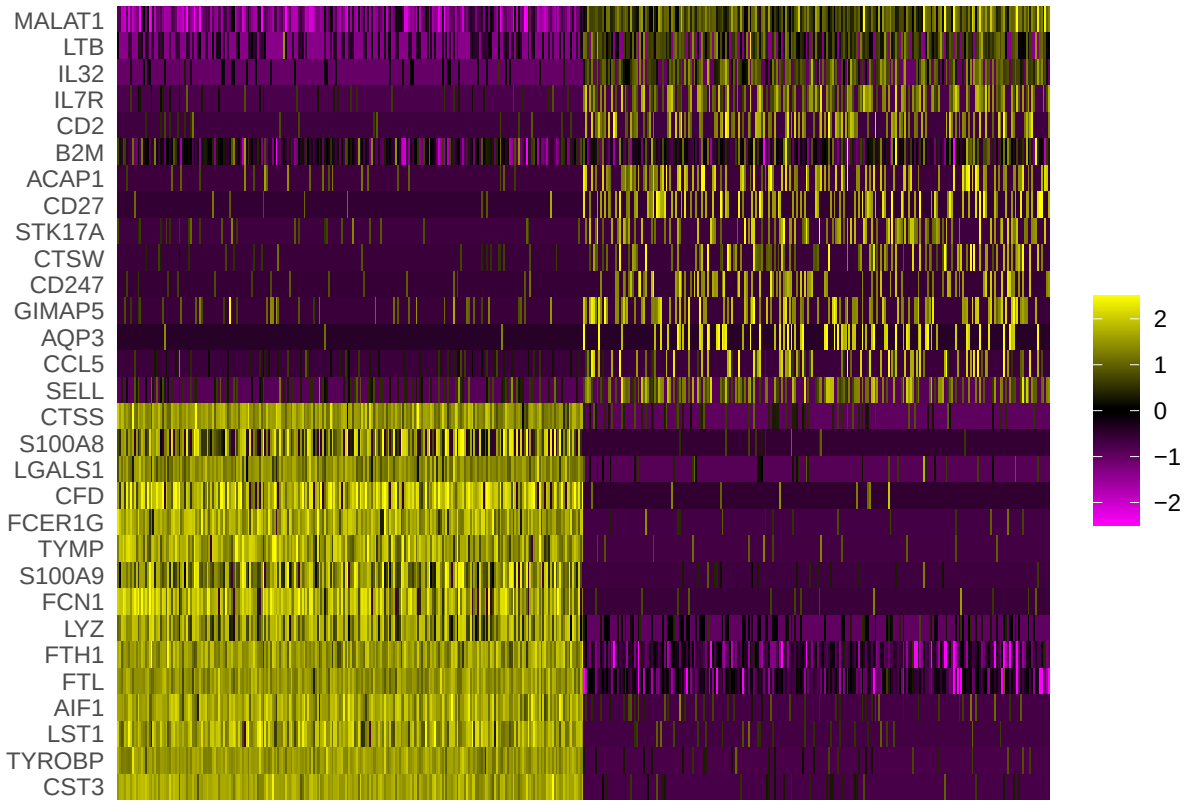
```
DimPlot(pbmc, reduction = "pca",
        pt.size = 2, #Size of the dots
        #group.by = 'orig.ident' # Group by class
        cols = '#9632B8')
```



Visualize in the form of dimension plot

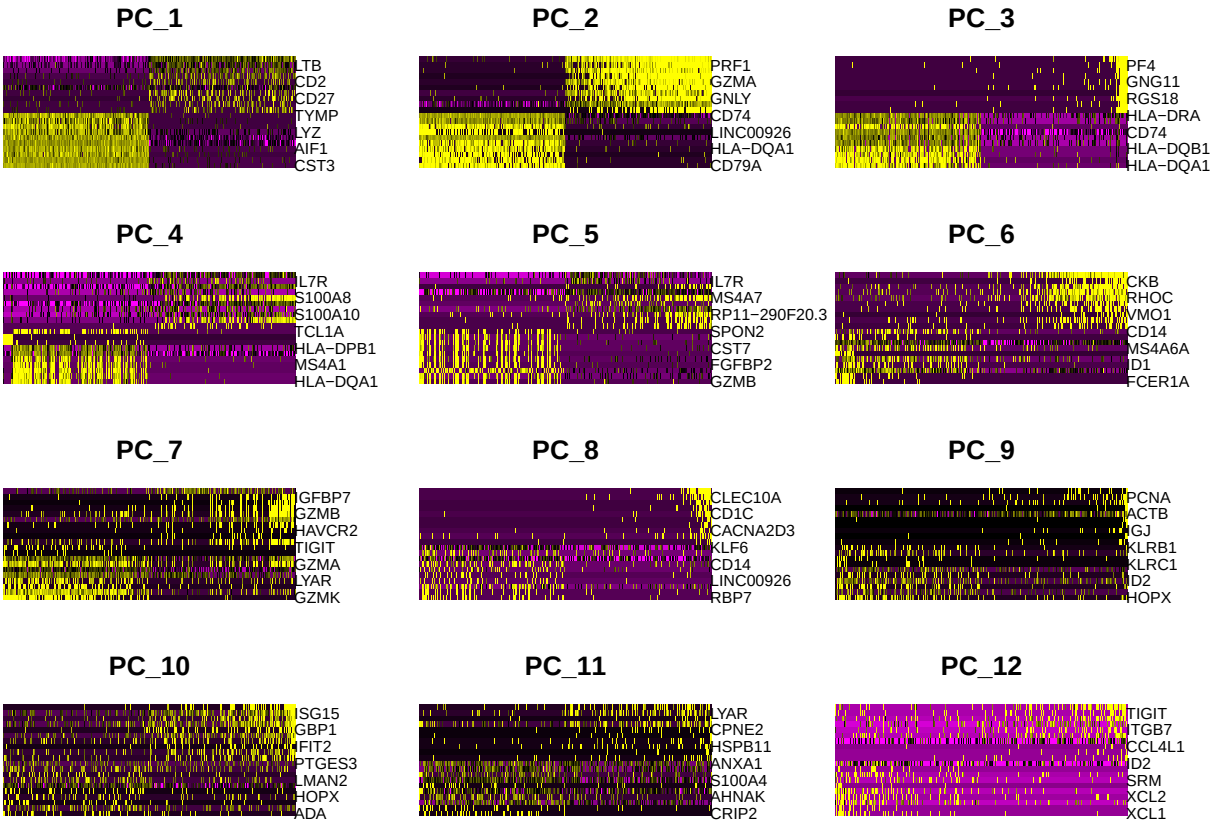
Investigate the heterogeneity among cells

```
DimHeatmap(pbmc, dims = 1,
           nfeatures = 30, #Number of top genes based on PCA scores
           cells = 500, #Number of top cells based on PCA scores
           balanced = TRUE, #Equal number of top genes based on both +&- PCA scores
           #disp.min = -10.5, #Negative cutoff
           #disp.max = 10.5 #Positive cutoff
           fast = F #Add legend (PC score scale)
           )
```

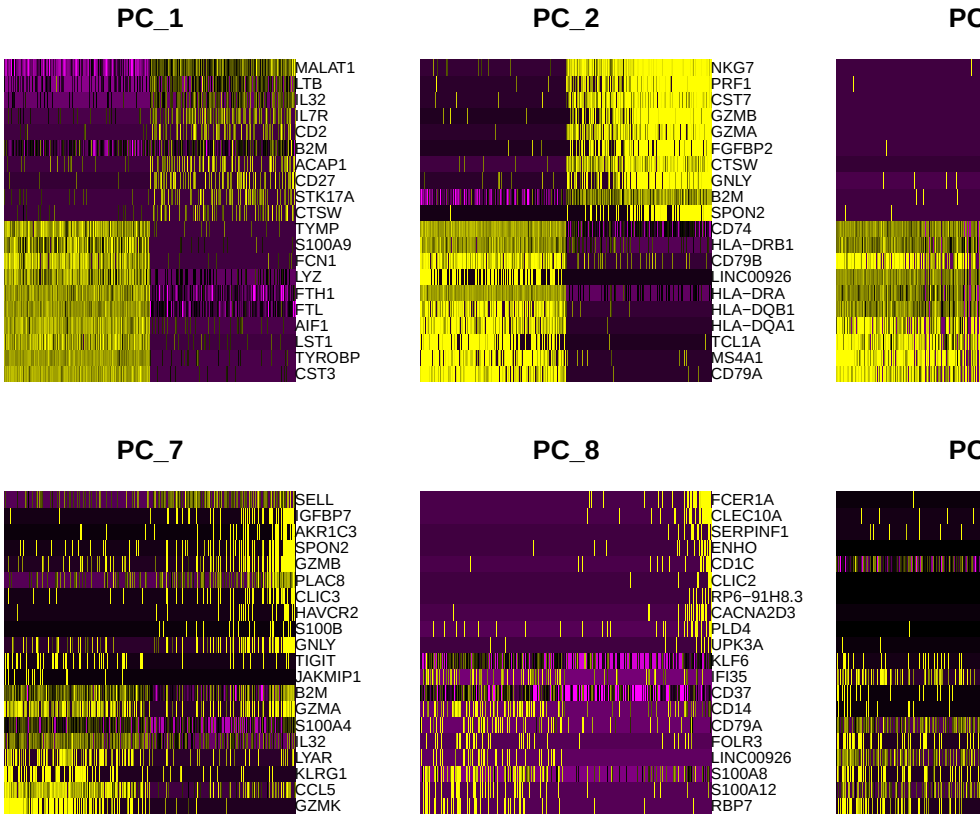


Investigate the heterogeneity among different number of dimensions In particular DimHeatmap() allows for easy exploration of the primary sources of heterogeneity in a dataset, and can be useful when trying to decide which PCs to include for further downstream analyses. The impression from the dim plot below is that PC1 and PC2 separetes the cell population into half whereas other PCs are not that comprehendible to understand heterogeneity like PC1 and PC2.

```
DimHeatmap(pbmc, dims = 1:12, nfeatures = 20, cells = 500, balanced = TRUE)
```




```
DimHeatmap(pbmc, dims = c(1,2,3,7:9), nfeatures = 20, cells = 500, balanced = TRUE)
```

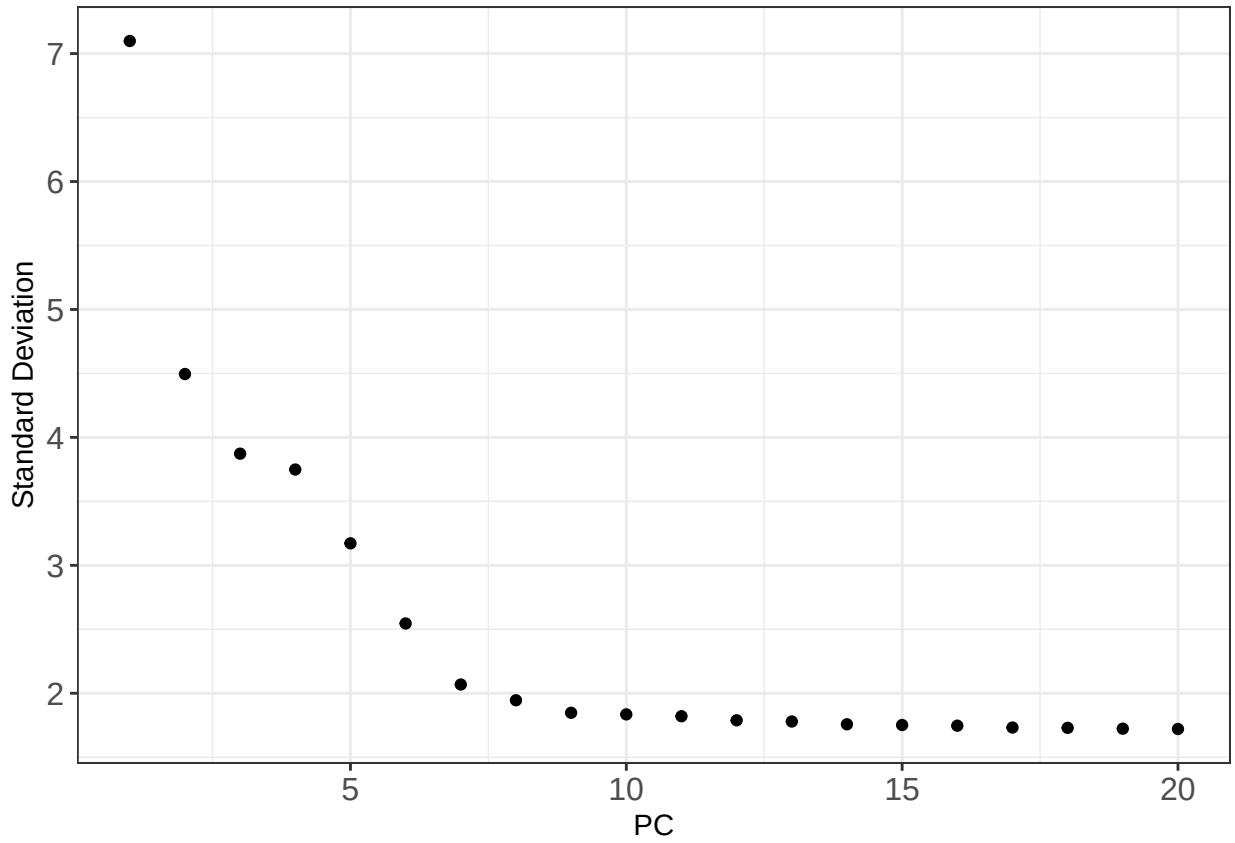


Heatmap with manual settings

Alternative:

Number of significant dimensions could be identified with an elbowplot. Elbowplot utilizes a heuristic method to calculate SD among the PCs. Commonly used

```
El_plot<-ElbowPlot(pbmc)
El_plot<- El_plot+theme_bw()+theme(axis.text = element_text(size = 12))
El_plot
```



Elbowplot It suggests that at around 15th PC, the hetereogeneity among the dimensions identified drops significantly

Another alternative way JackStraw method takes random identified dimendions and utilizes statistical model to assign p-value. This usually takes time for larger dataset and depends on computational power.Elbowplot is advised to use.

```
#pbmc <- JackStraw(pbmc, num.replicate = 100)
#pbmc <- ScoreJackStraw(pbmc, dims = 1:20)
#JackStrawPlot(pbmc, dims = 1:15)
```

Forming clusters

```
pbmc<- FindNeighbors(pbmc, dims = 1:15)
```

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

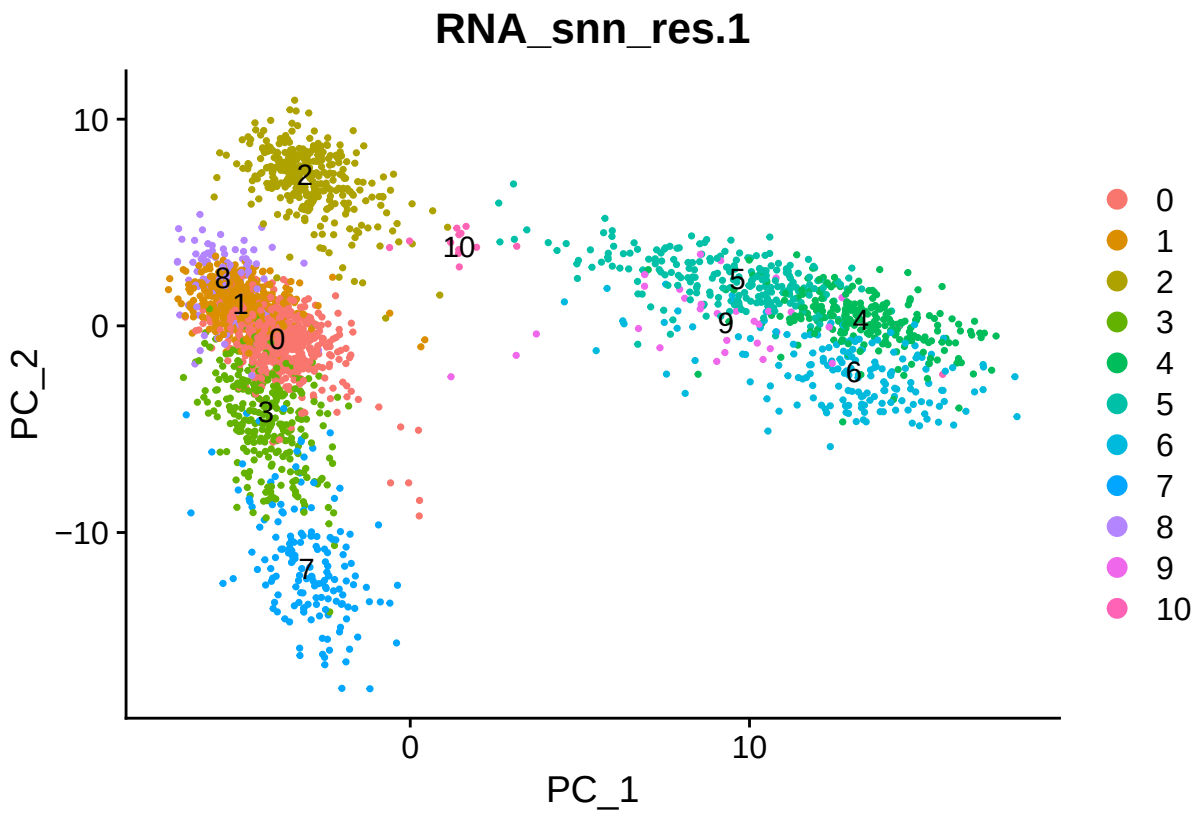
```
## Computing SNN
```

Setting resolution

```
pbmc <- FindClusters(pbmc, resolution = c(0.1,0.3, 0.5, 0.7, 1))
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2638
## Number of edges: 106055
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9615
## 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: 2638
## Number of edges: 106055
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9096
## 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: 2638
## Number of edges: 106055
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8680
## Number of communities: 9
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2638
## Number of edges: 106055
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8370
## Number of communities: 9
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2638
## Number of edges: 106055
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7932
## Number of communities: 11
## Elapsed time: 0 seconds
```

```
View(pbmc@meta.data)
DimPlot(pbmc, group.by = "RNA_snn_res.1", label = TRUE)
```



Run Umap

```
pbmc <- RunUMAP(pbmc, dims = 1:15)
```

```
## 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

## 01:22:46 UMAP embedding parameters a = 0.9922 b = 1.112

## 01:22:46 Read 2638 rows and found 15 numeric columns

## 01:22:46 Using Annoy for neighbor search, n_neighbors = 30

## 01:22:46 Building Annoy index with metric = cosine, n_trees = 50

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

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

```
## **************************************************|
## 01:22:46 Writing NN index file to temp file C:\Users\HP\AppData\Local\Temp\RtmpwjLP2E\file2770252314
## 01:22:46 Searching Annoy index using 1 thread, search_k = 3000
## 01:22:47 Annoy recall = 100%
## 01:22:47 Commencing smooth kNN distance calibration using 1 thread with target n_neighbors = 30
## 01:22:48 Initializing from normalized Laplacian + noise (using irlba)
## 01:22:49 Commencing optimization for 500 epochs, with 107718 positive edges
## 01:22:56 Optimization finished
```

Finding doublets

```
sweep.res.list_pbmc <- paramSweep_v3(pbmc, PCs = 1:15, sct = FALSE)
```

pK Identification (no ground-truth)

```
## Loading required package: fields

## Loading required package: spam

## Spam version 2.9-1 (2022-08-07) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

##
## Attaching package: 'spam'

## The following object is masked from 'package:stats4':
##
##     mle

## The following object is masked from 'package:Matrix':
##
##     det

## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve

##
## Try help(fields) to get started.

## [1] "Creating artificial doublets for pN = 5%"
## [1] "Creating Seurat object..."
## [1] "Normalizing Seurat object..."
## [1] "Finding variable genes..."
## [1] "Scaling data..."
```

```
## Centering and scaling data matrix

## [1] "Running PCA..."
## [1] "Calculating PC distance matrix..."
## [1] "Defining neighborhoods..."
## [1] "Computing pANN across all pK..."
## [1] "pK = 0.005..."
## [1] "pK = 0.01..."
## [1] "pK = 0.02..."
## [1] "pK = 0.03..."
## [1] "pK = 0.04..."
## [1] "pK = 0.05..."
## [1] "pK = 0.06..."
## [1] "pK = 0.07..."
## [1] "pK = 0.08..."
## [1] "pK = 0.09..."
## [1] "pK = 0.1..."
## [1] "pK = 0.11..."
## [1] "pK = 0.12..."
## [1] "pK = 0.13..."
## [1] "pK = 0.14..."
## [1] "pK = 0.15..."
## [1] "pK = 0.16..."
## [1] "pK = 0.17..."
## [1] "pK = 0.18..."
## [1] "pK = 0.19..."
## [1] "pK = 0.2..."
## [1] "pK = 0.21..."
## [1] "pK = 0.22..."
## [1] "pK = 0.23..."
## [1] "pK = 0.24..."
## [1] "pK = 0.25..."
## [1] "pK = 0.26..."
## [1] "pK = 0.27..."
## [1] "pK = 0.28..."
## [1] "pK = 0.29..."
## [1] "pK = 0.3..."
## [1] "Creating artificial doublets for pN = 10%"
## [1] "Creating Seurat object..."
## [1] "Normalizing Seurat object..."
## [1] "Finding variable genes..."
## [1] "Scaling data..."

## Centering and scaling data matrix

## [1] "Running PCA..."
## [1] "Calculating PC distance matrix..."
## [1] "Defining neighborhoods..."
## [1] "Computing pANN across all pK..."
## [1] "pK = 0.005..."
## [1] "pK = 0.01..."
## [1] "pK = 0.02..."
## [1] "pK = 0.03..."
```

```
## [1] "pK = 0.04..."
## [1] "pK = 0.05..."
## [1] "pK = 0.06..."
## [1] "pK = 0.07..."
## [1] "pK = 0.08..."
## [1] "pK = 0.09..."
## [1] "pK = 0.1..."
## [1] "pK = 0.11..."
## [1] "pK = 0.12..."
## [1] "pK = 0.13..."
## [1] "pK = 0.14..."
## [1] "pK = 0.15..."
## [1] "pK = 0.16..."
## [1] "pK = 0.17..."
## [1] "pK = 0.18..."
## [1] "pK = 0.19..."
## [1] "pK = 0.2..."
## [1] "pK = 0.21..."
## [1] "pK = 0.22..."
## [1] "pK = 0.23..."
## [1] "pK = 0.24..."
## [1] "pK = 0.25..."
## [1] "pK = 0.26..."
## [1] "pK = 0.27..."
## [1] "pK = 0.28..."
## [1] "pK = 0.29..."
## [1] "pK = 0.3..."
## [1] "Creating artificial doublets for pN = 15%"
## [1] "Creating Seurat object..."
## [1] "Normalizing Seurat object..."
## [1] "Finding variable genes..."
## [1] "Scaling data..."

## Centering and scaling data matrix

## [1] "Running PCA..."
## [1] "Calculating PC distance matrix..."
## [1] "Defining neighborhoods..."
## [1] "Computing pANN across all pK..."
## [1] "pK = 0.005..."
## [1] "pK = 0.01..."
## [1] "pK = 0.02..."
## [1] "pK = 0.03..."
## [1] "pK = 0.04..."
## [1] "pK = 0.05..."
## [1] "pK = 0.06..."
## [1] "pK = 0.07..."
## [1] "pK = 0.08..."
## [1] "pK = 0.09..."
## [1] "pK = 0.1..."
## [1] "pK = 0.11..."
## [1] "pK = 0.12..."
## [1] "pK = 0.13..."
## [1] "pK = 0.14..."
```

```
## [1] "pK = 0.15..."
## [1] "pK = 0.16..."
## [1] "pK = 0.17..."
## [1] "pK = 0.18..."
## [1] "pK = 0.19..."
## [1] "pK = 0.2..."
## [1] "pK = 0.21..."
## [1] "pK = 0.22..."
## [1] "pK = 0.23..."
## [1] "pK = 0.24..."
## [1] "pK = 0.25..."
## [1] "pK = 0.26..."
## [1] "pK = 0.27..."
## [1] "pK = 0.28..."
## [1] "pK = 0.29..."
## [1] "pK = 0.3..."
## [1] "Creating artificial doublets for pN = 20%"
## [1] "Creating Seurat object..."
## [1] "Normalizing Seurat object..."
## [1] "Finding variable genes..."
## [1] "Scaling data..."

## Centering and scaling data matrix

## [1] "Running PCA..."
## [1] "Calculating PC distance matrix..."
## [1] "Defining neighborhoods..."
## [1] "Computing pANN across all pK..."
## [1] "pK = 0.005..."
## [1] "pK = 0.01..."
## [1] "pK = 0.02..."
## [1] "pK = 0.03..."
## [1] "pK = 0.04..."
## [1] "pK = 0.05..."
## [1] "pK = 0.06..."
## [1] "pK = 0.07..."
## [1] "pK = 0.08..."
## [1] "pK = 0.09..."
## [1] "pK = 0.1..."
## [1] "pK = 0.11..."
## [1] "pK = 0.12..."
## [1] "pK = 0.13..."
## [1] "pK = 0.14..."
## [1] "pK = 0.15..."
## [1] "pK = 0.16..."
## [1] "pK = 0.17..."
## [1] "pK = 0.18..."
## [1] "pK = 0.19..."
## [1] "pK = 0.2..."
## [1] "pK = 0.21..."
## [1] "pK = 0.22..."
## [1] "pK = 0.23..."
## [1] "pK = 0.24..."
## [1] "pK = 0.25..."
```

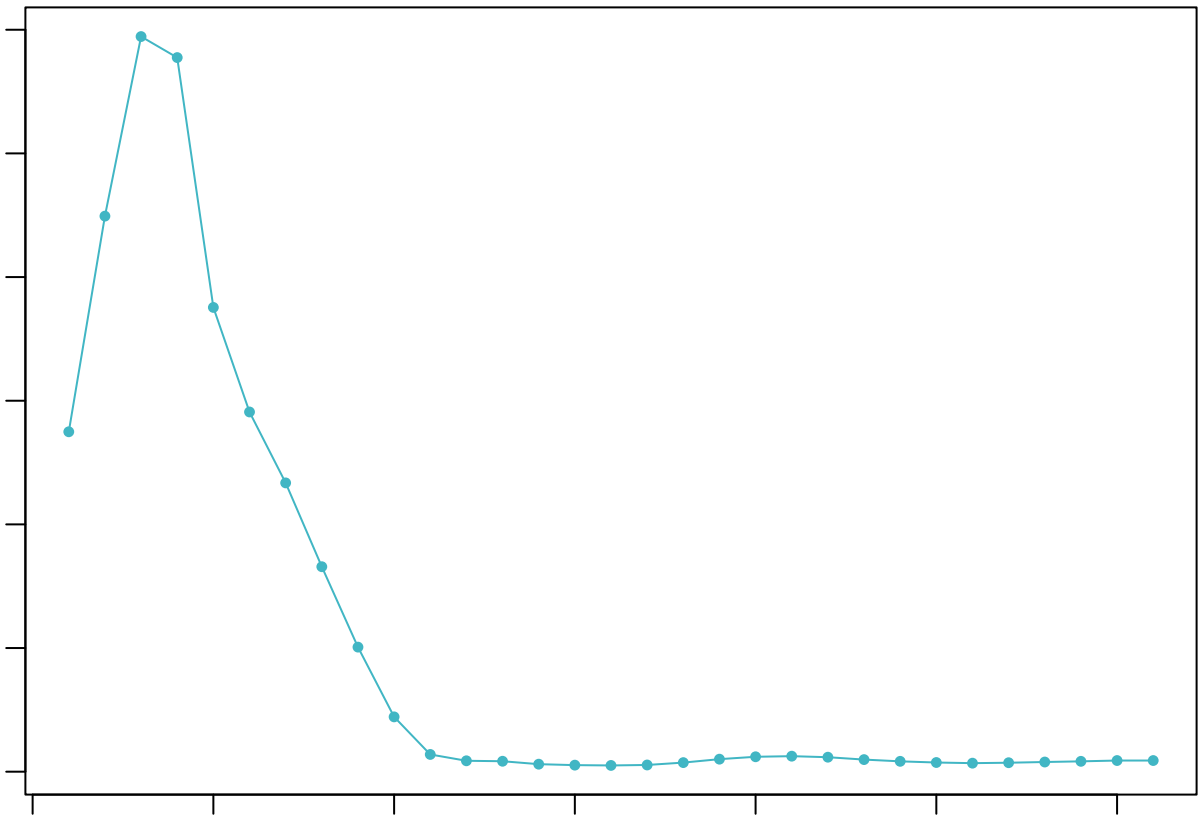
```
## [1] "pK = 0.26..."
## [1] "pK = 0.27..."
## [1] "pK = 0.28..."
## [1] "pK = 0.29..."
## [1] "pK = 0.3..."
## [1] "Creating artificial doublets for pN = 25%"
## [1] "Creating Seurat object..."
## [1] "Normalizing Seurat object..."
## [1] "Finding variable genes..."
## [1] "Scaling data..."

## Centering and scaling data matrix

## [1] "Running PCA..."
## [1] "Calculating PC distance matrix..."
## [1] "Defining neighborhoods..."
## [1] "Computing pANN across all pK..."
## [1] "pK = 0.005..."
## [1] "pK = 0.01..."
## [1] "pK = 0.02..."
## [1] "pK = 0.03..."
## [1] "pK = 0.04..."
## [1] "pK = 0.05..."
## [1] "pK = 0.06..."
## [1] "pK = 0.07..."
## [1] "pK = 0.08..."
## [1] "pK = 0.09..."
## [1] "pK = 0.1..."
## [1] "pK = 0.11..."
## [1] "pK = 0.12..."
## [1] "pK = 0.13..."
## [1] "pK = 0.14..."
## [1] "pK = 0.15..."
## [1] "pK = 0.16..."
## [1] "pK = 0.17..."
## [1] "pK = 0.18..."
## [1] "pK = 0.19..."
## [1] "pK = 0.2..."
## [1] "pK = 0.21..."
## [1] "pK = 0.22..."
## [1] "pK = 0.23..."
## [1] "pK = 0.24..."
## [1] "pK = 0.25..."
## [1] "pK = 0.26..."
## [1] "pK = 0.27..."
## [1] "pK = 0.28..."
## [1] "pK = 0.29..."
## [1] "pK = 0.3..."
## [1] "Creating artificial doublets for pN = 30%"
## [1] "Creating Seurat object..."
## [1] "Normalizing Seurat object..."
## [1] "Finding variable genes..."
## [1] "Scaling data..."
```

```
## Centering and scaling data matrix

## [1] "Running PCA..."
## [1] "Calculating PC distance matrix..."
## [1] "Defining neighborhoods..."
## [1] "Computing pANN across all pK..."
## [1] "pK = 0.005..."
## [1] "pK = 0.01..."
## [1] "pK = 0.02..."
## [1] "pK = 0.03..."
## [1] "pK = 0.04..."
## [1] "pK = 0.05..."
## [1] "pK = 0.06..."
## [1] "pK = 0.07..."
## [1] "pK = 0.08..."
## [1] "pK = 0.09..."
## [1] "pK = 0.1..."
## [1] "pK = 0.11..."
## [1] "pK = 0.12..."
## [1] "pK = 0.13..."
## [1] "pK = 0.14..."
## [1] "pK = 0.15..."
## [1] "pK = 0.16..."
## [1] "pK = 0.17..."
## [1] "pK = 0.18..."
## [1] "pK = 0.19..."
## [1] "pK = 0.2..."
## [1] "pK = 0.21..."
## [1] "pK = 0.22..."
## [1] "pK = 0.23..."
## [1] "pK = 0.24..."
## [1] "pK = 0.25..."
## [1] "pK = 0.26..."
## [1] "pK = 0.27..."
## [1] "pK = 0.28..."
## [1] "pK = 0.29..."
## [1] "pK = 0.3..."
```

```
sweep.stats_pbmc <- summarizeSweep(sweep.res.list_pbmc, GT = FALSE)
```

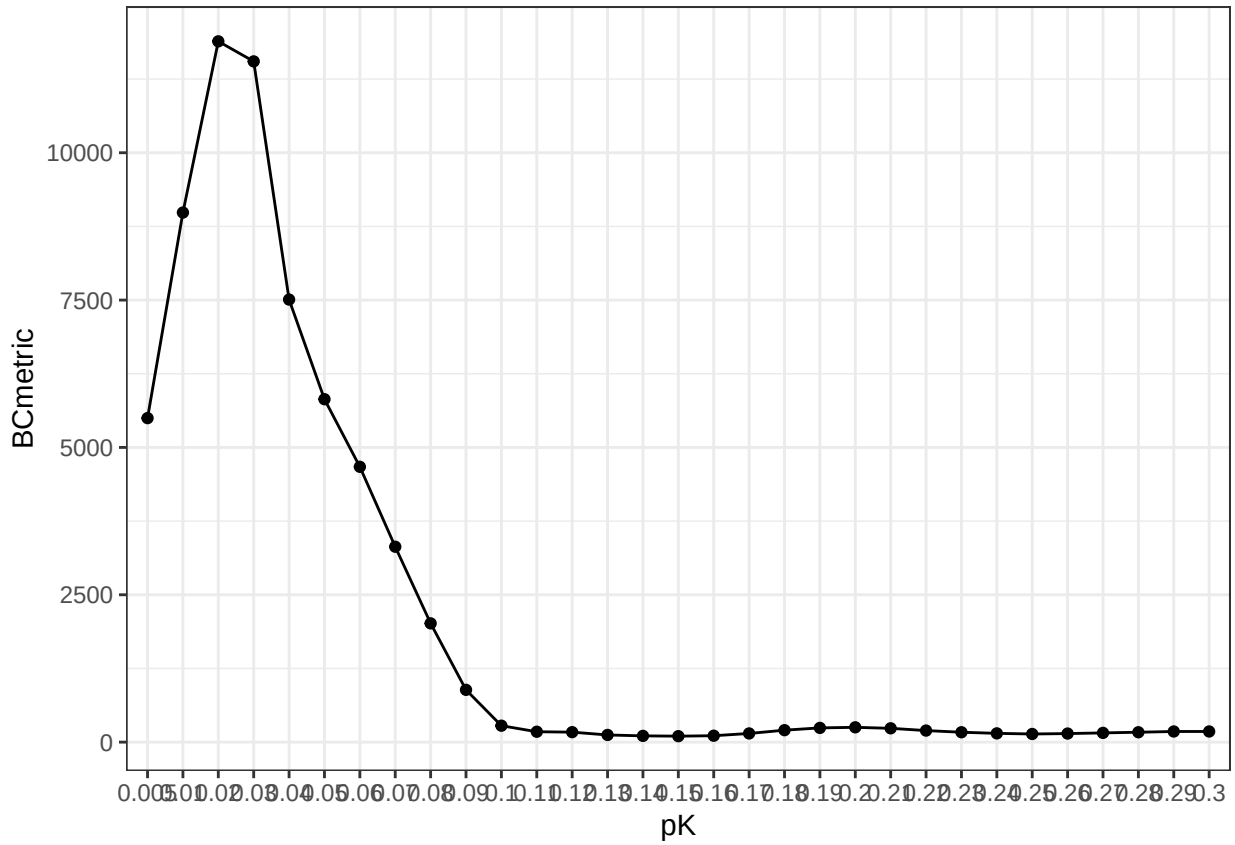
```
## Loading required package: KernSmooth

## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009

## Loading required package: ROCR
```

```
bcmvn_pbmc <- find.pK(sweep.stats_pbmc)
```



```
## NULL
```

```
ggplot(bcmvn_pbmc, aes(pK, BCmetric, group = 1)) +
  geom_point() + theme_bw()+
  geom_line()
```



pK Identification The highest BCmetric corresponding to pK value represents the optimum pk value (0.02 in this case)

```
pK <- bcmvn_pbmc %>%
  filter(BCmetric == max(BCmetric)) %>%
  select(pK)
pK <- as.numeric(as.character(pK[[1]]))
```

select the pK that corresponds to max bcmvn to perform doublet detection

Homotypic Doublet Proportion Estimate Additional step for homotypic doublets since DoubletFinder is less sensitive to homotypic doublets.

```
annotations <- pbmc@meta.data$seurat_clusters
homotypic.prop <- modelHomotypic(annotations)
nExp_doub <- round(0.015*nrow(pbmc@meta.data))
```

Homotypic Doublet Proportion Estimate Assuming 1.5% doublet in our dataset given 2700 cells (follow 10x genomics user guide or DoubletFinder instructions)

```
nExp_doub.adj <- round(nExp_doub*(1-homotypic.prop))
```

```
pbmc<- doubletFinder_v3(pbmc,
                        PCs = 1:20,
                        pN = 0.25, #default doesn't affect DblFnr performance
                        pK = pK,
                        nExp = nExp_doub.adj,
                        reuse.pANN = FALSE, sct = FALSE) #No sct trans.
```

Run doubletFinder

```
## [1] "Creating 879 artificial doublets..."
## [1] "Creating Seurat object..."
## [1] "Normalizing Seurat object..."
## [1] "Finding variable genes..."
## [1] "Scaling data..."

## Centering and scaling data matrix

## [1] "Running PCA..."
## [1] "Calculating PC distance matrix..."
## [1] "Computing pANN..."
## [1] "Classifying doublets.."
```

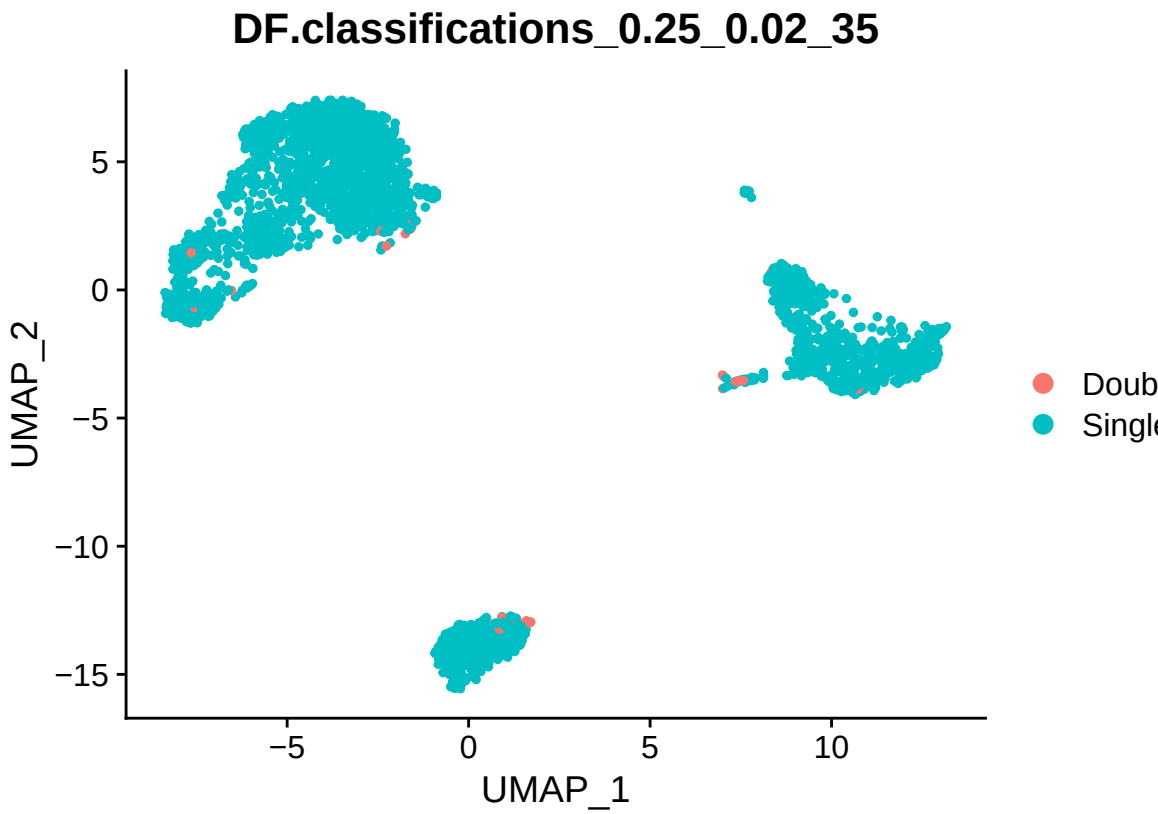
```
#View(pbmc@meta.data)
```

```
table(pbmc@meta.data$DF.classifications_0.25_0.02_35)
```

Numver of doublets and singlets

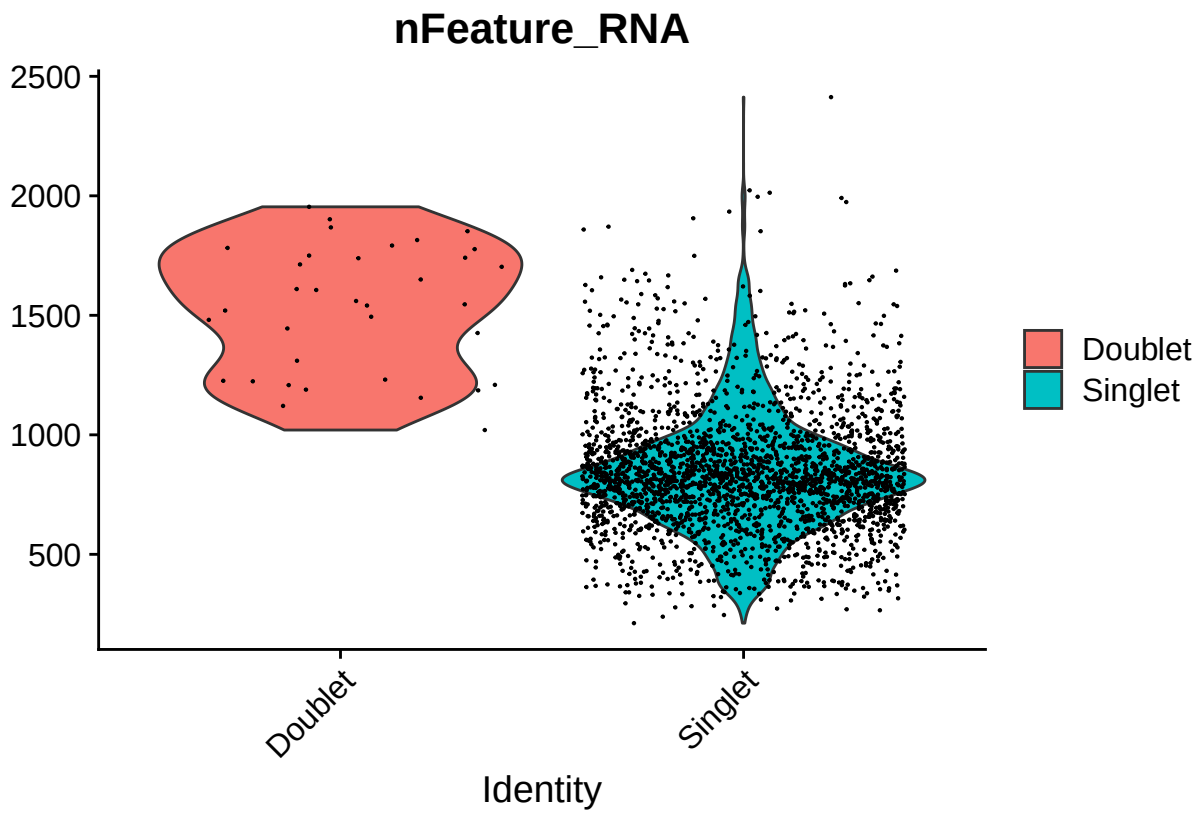
```
##
## Doublet Singlet
##      35    2603
```

```
DimPlot(pbmc, reduction = 'umap', group.by = "DF.classifications_0.25_0.02_35",
        pt.size = 1)
```



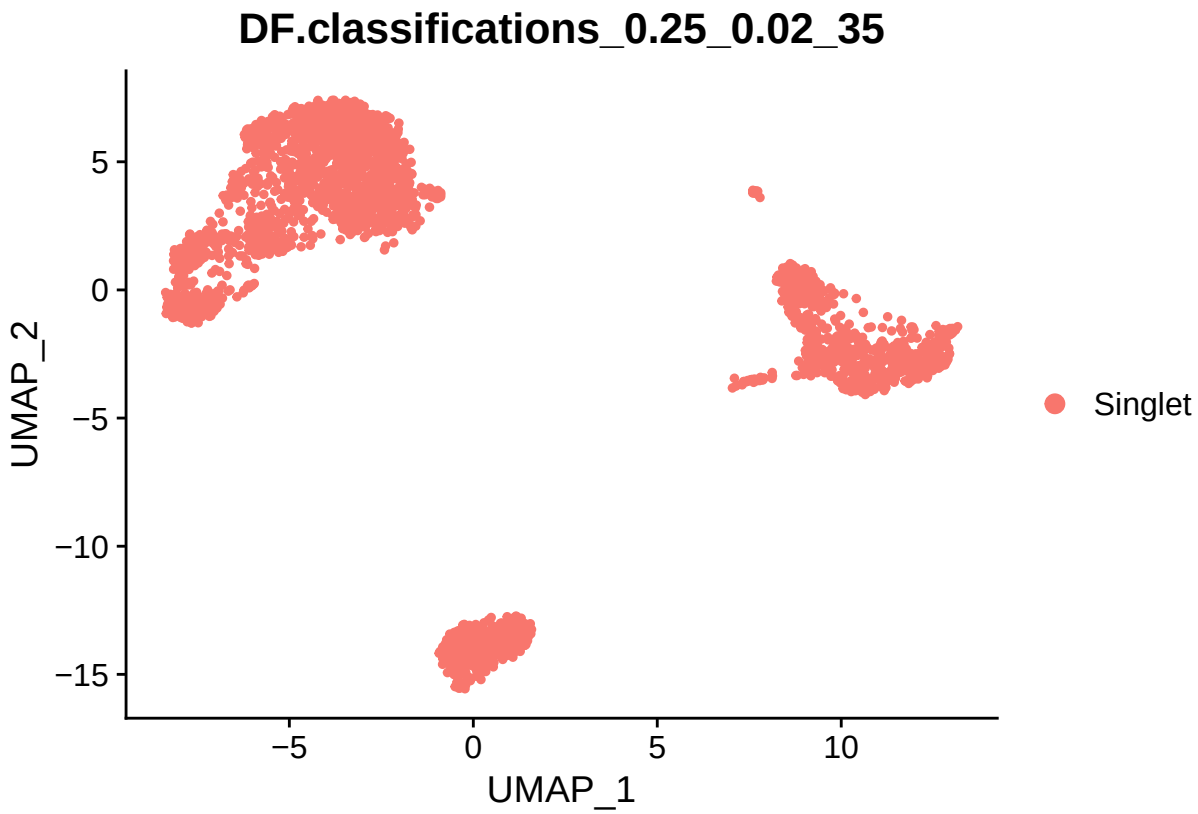
visualize doublets

```
VlnPlot(pbmc, features = "nFeature_RNA", group.by = "DF.classifications_0.25_0.02_35")
```



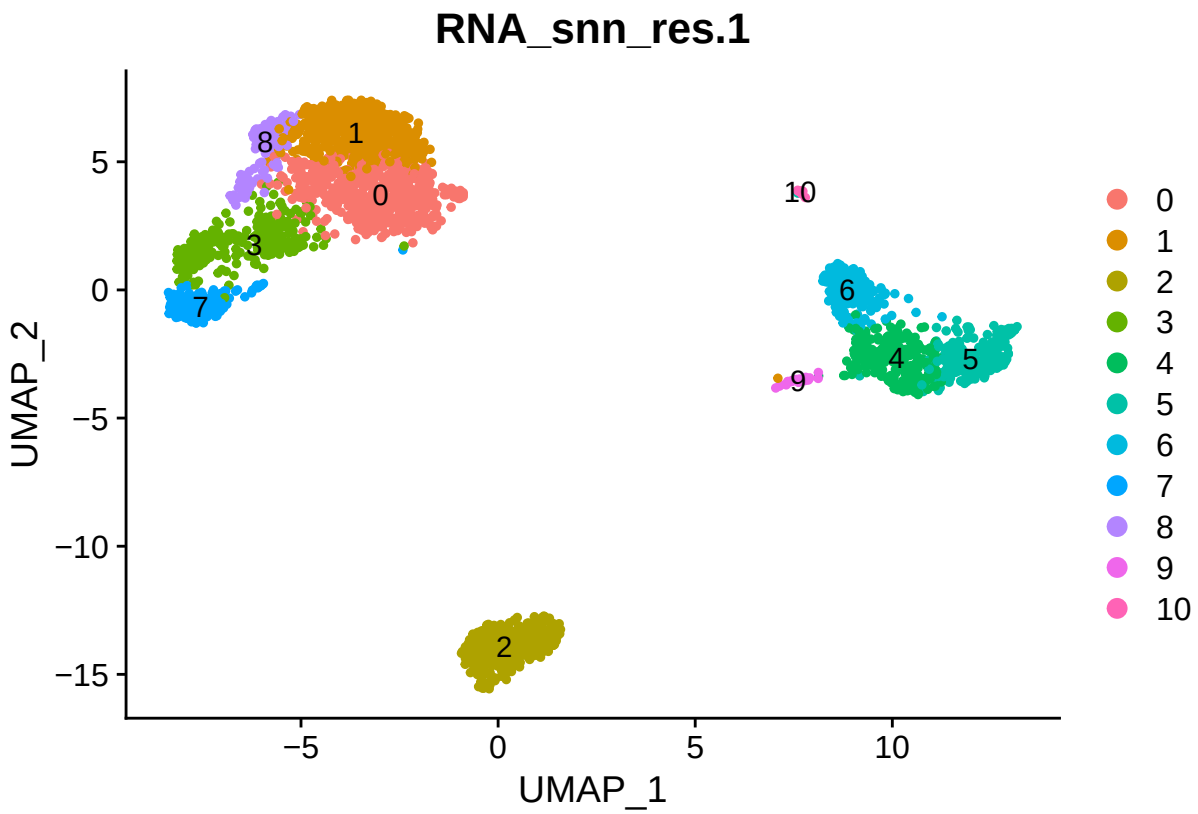
Remove doublets

```
pbmc = pbmc[, pbmc@meta.data[, "DF.classifications_0.25_0.02_35"] == "Singlet"]
#Now visualize again
DimPlot(pbmc, reduction = 'umap', group.by = "DF.classifications_0.25_0.02_35",
        pt.size = 1)
```




```
##############################################################################
### Plotting
```

```
DimPlot(pbmc, group.by = "RNA_snn_res.1", reduction = "umap", label = T , pt.size=1)
```



Customization Setting colors

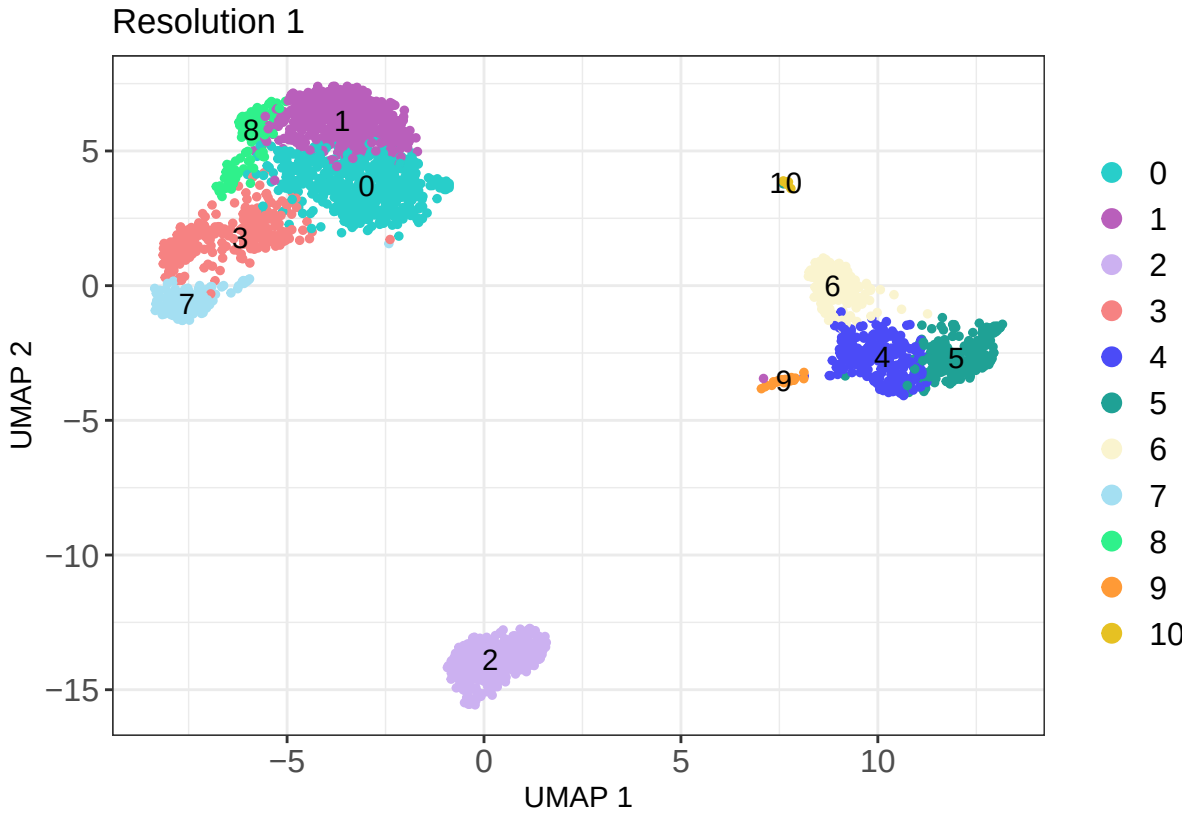
```
color.length <- subset(pbmc, idents = 0:10)
levels(Idents(color.length ))
```

```
##  [1] "0"  "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
```

```
cols <- c('3'='#F68282','5'='#1FA195','1'='#B95FBB',
          '9'='#ff9a36','8'='#2FF18B','6'='#faf4cf',
          '2'='#CCB1F1','7'='#A4DFF2', '0'='#28CECA',
          '4'='#4B4BF7','10'='#E6C122')
cols <- cols[order(as.integer(names(cols)))]
scales::show_col(cols)
```

#28CECA	#B95FBB	#CCB1F1	#F68282
#4B4BF7	#1FA195	#faf4cf	#A4DFF2
#2FF18B	#ff9a36	#E6C122	

```
umap<-DimPlot(pbmc, group.by="RNA_snn_res.1", reduction="umap", label=T ,
        pt.size=1, cols = cols)
umap<- umap+ theme_bw()+labs(x='UMAP 1', y='UMAP 2') + ggtitle('Resolution 1')+
  theme(axis.text = element_text(size = 12), legend.text = element_text(size = 12))
umap
```

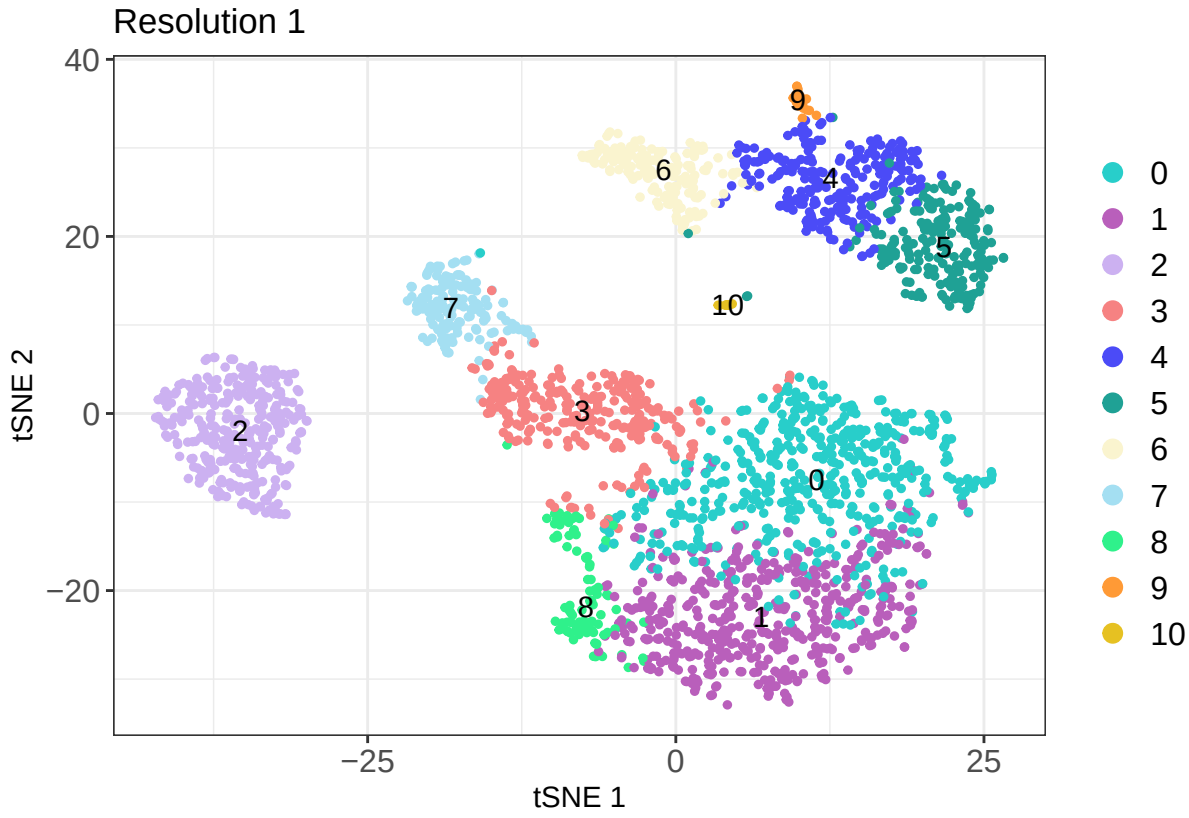


Plot and finalize

Calculate TSNE and Plot

```
pbmc<-RunTSNE(pbmc, dims = 1:10)
tsne<-DimPlot(pbmc, group.by = "RNA_snn_res.1", reduction = 'tsne', label = TRUE,
              cols=cols, pt.size=1)
```

```
tsne<- tsne+theme_bw()+labs(x='tSNE 1', y='tSNE 2') + ggtitle('Resolution 1')+
       theme(axis.text = element_text(size = 12), legend.text = element_text(size = 12))
tsne
```



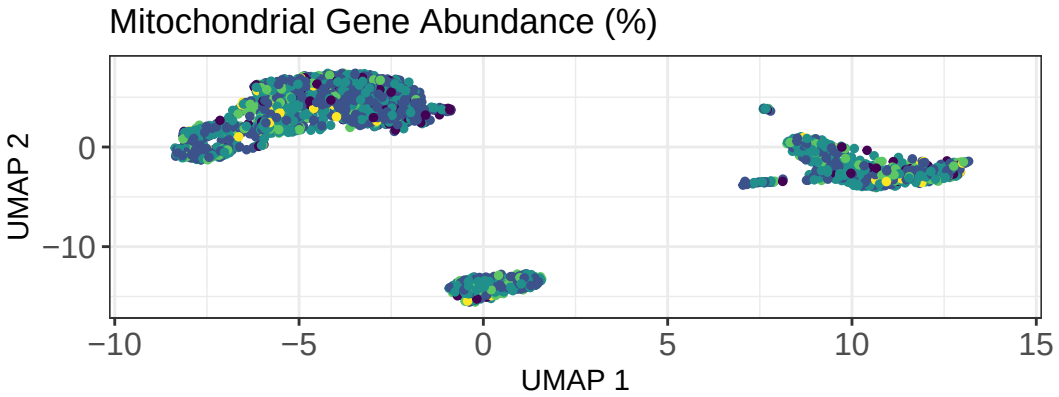
Customization

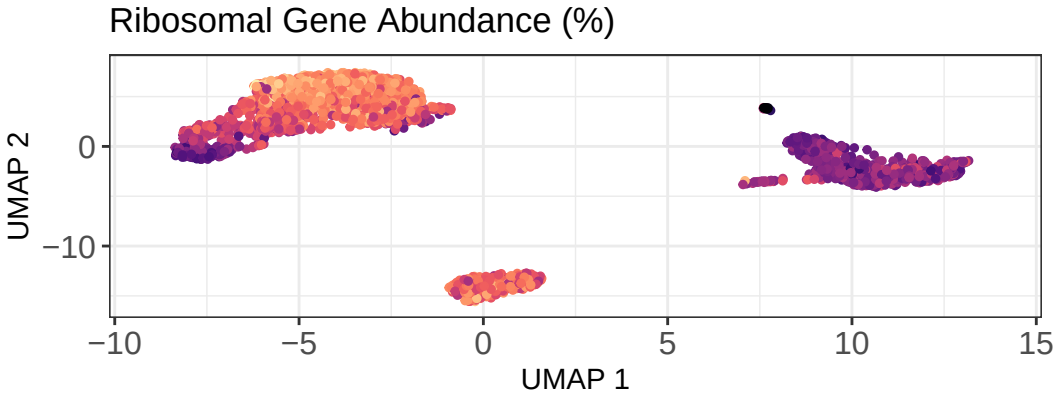
We can also visualize the percent of MT and RB genes in our datasets

Make some palletes

```
Mt.pal<- viridis(n = 5, option = "D")
Rb.pal<- viridis(n = 60, option = "A") #N depends on percent allowed in dataset
```

```
Mt<-FeaturePlot(pbmc, reduction='umap', features='percent.mt', pt.size=1, cols = Mt.pal)
Rb<-FeaturePlot(pbmc, reduction='umap', features='percent.rb', pt.size=1, cols = Rb.pal)

Mt<-Mt+theme_bw()+labs(x='UMAP 1', y='UMAP 2')+ ggtitle('Mitochondrial Gene Abundance (%)')+
    theme(axis.text = element_text(size = 12), legend.text = element_text(size = 12))
Rb<-Rb+theme_bw()+labs(x='UMAP 1', y='UMAP 2')+ ggtitle('Ribosomal Gene Abundance (%)')+
    theme(axis.text = element_text(size = 12), legend.text = element_text(size = 12))

Mt+Rb
```





Plot and customize

Save Result as RDS

```
saveRDS(pbmc, file = "PBMC.rds")
```

Find Markers

```
cluster2.markers <- FindMarkers(pbmc, ident.1 = 2, min.pct = 0.25)
```

Find all markers of cluster 2

```
## For a more efficient implementation of the Wilcoxon Rank Sum Test,
## (default method for FindMarkers) please install the limma package
## --------------------------------------------
## install.packages('BiocManager')
## BiocManager::install('limma')
## --------------------------------------------
## After installation of limma, Seurat will automatically use the more
## efficient implementation (no further action necessary).
## This message will be shown once per session
```

```
head(cluster2.markers, n = 5)
```

```
##                   p_val avg_log2FC pct.1 pct.2     p_val_adj
## MS4A1      0.000000e+00   3.384554 0.856 0.052  0.000000e+00
## CD79A      0.000000e+00   4.341007 0.934 0.041  0.000000e+00
## CD79B     2.404435e-273   3.510085 0.922 0.140 3.297442e-269
## TCL1A     1.479135e-269   3.616264 0.625 0.021 2.028486e-265
## LINC00926 3.057900e-268   2.859638 0.562 0.009 4.193604e-264
```

```
cluster5.markers <- FindMarkers(pbmc, ident.1 = 5, ident.2 = c(0, 3), min.pct = 0.25)
head(cluster5.markers, n = 5)
```

Find all markers distinguishing cluster 5 from clusters 0 and 3

```
##                p_val avg_log2FC pct.1 pct.2     p_val_adj
## S100A8 2.016383e-184   6.397932 0.986 0.074 2.765268e-180
## LGALS2 8.357796e-161   4.003743 0.833 0.031 1.146188e-156
## TYROBP 5.361162e-160   4.878543 0.986 0.152 7.352298e-156
## S100A9 9.910466e-157   6.541017 0.995 0.179 1.359121e-152
## FCN1   1.048857e-154   4.189723 0.914 0.094 1.438402e-150
```

```
All.markers <- FindAllMarkers(pbmc, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25)
```

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

```
## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6

## Calculating cluster 7

## Calculating cluster 8

## Calculating cluster 9

## Calculating cluster 10
```

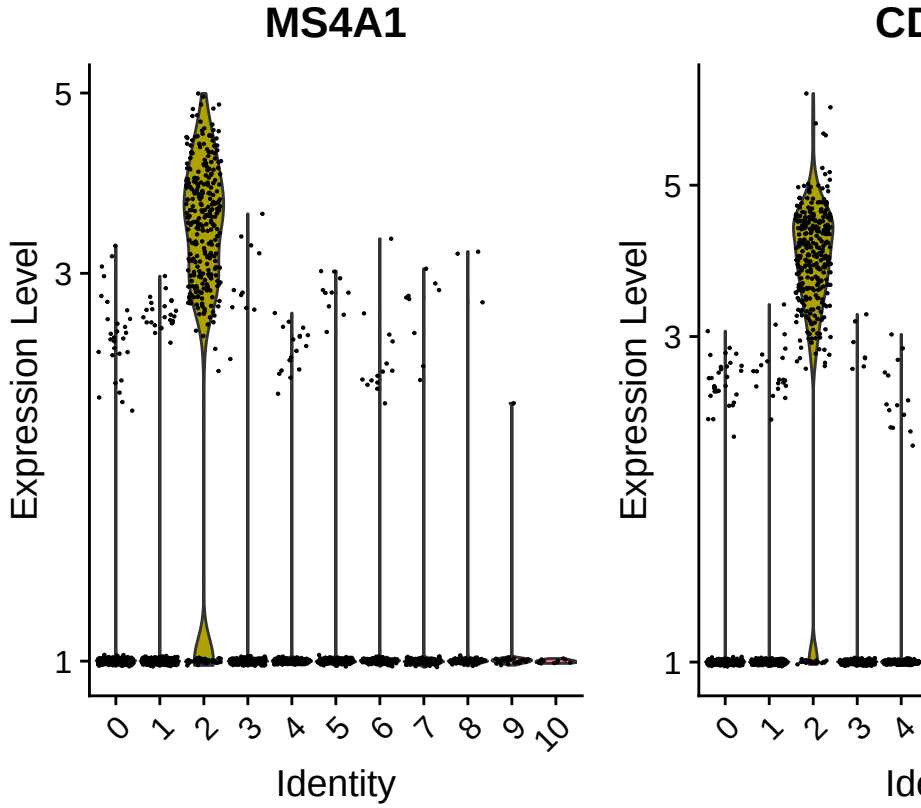
```
All.markers %>%
  group_by(cluster) %>%
  slice_max(n = 2, order_by = avg_log2FC)
```

```
## # A tibble: 22 x 7
## # Groups:   cluster [11]
##        p_val avg_log2FC pct.1 pct.2 p_val_adj cluster gene
##        <dbl>      <dbl> <dbl> <dbl>     <dbl> <fct>   <chr>
##  1 7.02e- 73       1.36 0.422 0.097 9.63e- 69 0       AQP3
##  2 1.66e-103       1.36 0.98  0.629 2.28e- 99 0       LTB
##  3 4.99e-100       1.42 0.523 0.114 6.84e- 96 1       CCR7
##  4 5.58e- 99       1.04 0.947 0.604 7.65e- 95 1       LDHB
##  5 0               4.34 0.934 0.041 0          2       CD79A
##  6 1.48e-269       3.62 0.625 0.021 2.03e-265 2       TCL1A
##  7 4.56e-182       3.07 0.606 0.051 6.26e-178 3       GZMK
##  8 1.47e-195       3.04 0.982 0.232 2.02e-191 3       CCL5
##  9 3.25e-132       3.12 1     0.561 4.46e-128 4       LYZ
## 10 7.26e-219       2.80 0.976 0.135 9.96e-215 4       LGALS2
## # i 12 more rows
```

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

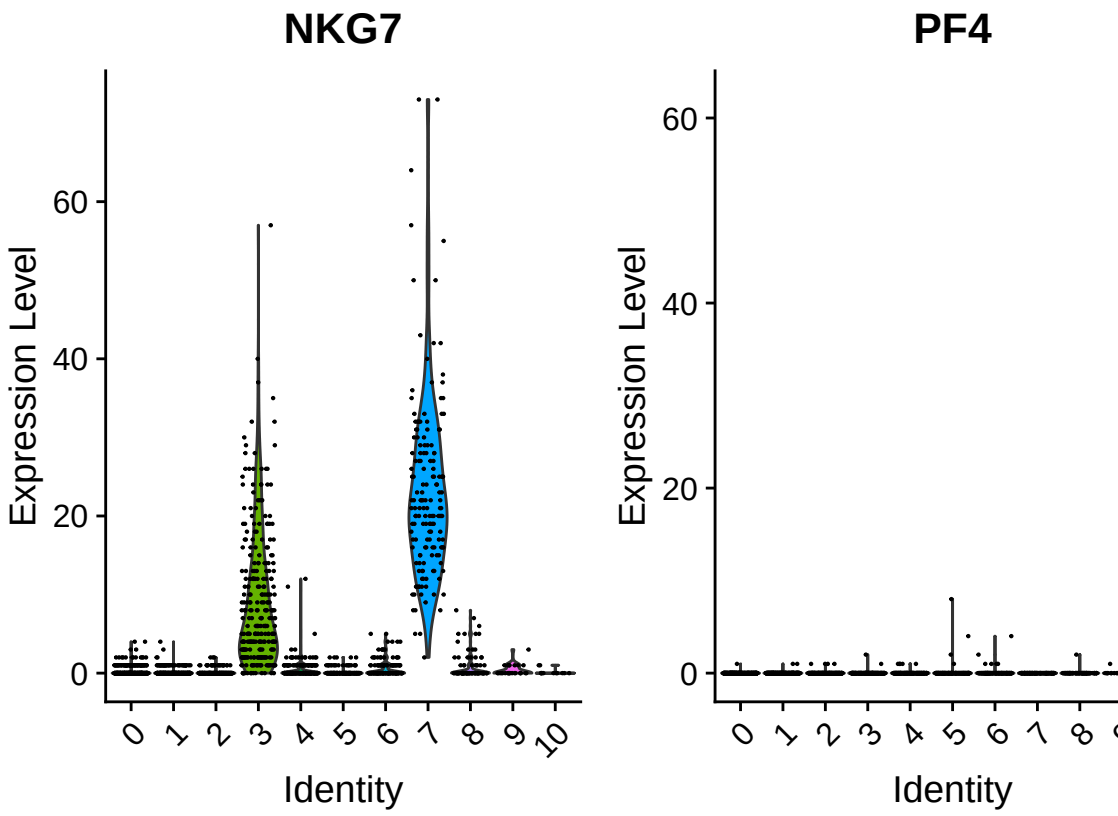
Find markers with specific tests

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



Checking features across all clusters

```
VlnPlot(pbmc, features = c("NKG7", "PF4"), slot = "counts", log = F)
```



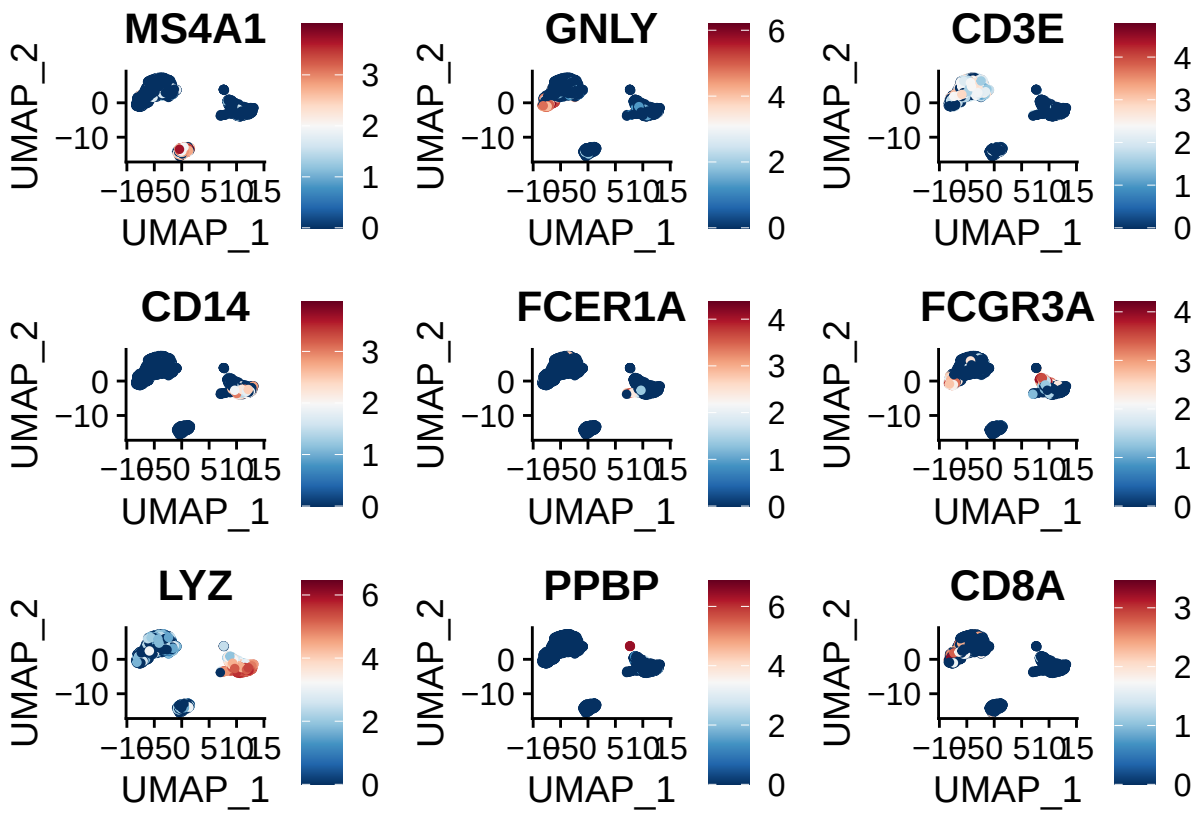
Checking raw counts

```
FeaturePlot(pbmc, features = c("MS4A1", "GNLY", "CD3E", "CD14", "FCER1A",
                               "FCGR3A", "LYZ", "PPBP", "CD8A"), pt.size = 1, ncol = 3) &
            scale_colour_gradientn(colours = rev(brewer.pal(n = 11, name = "RdBu")))
```

Check in the form of feature plot

```
## Scale for colour is already present.
## Adding another scale for colour, which will replace the existing scale.
## Scale for colour is already present.
## Adding another scale for colour, which will replace the existing scale.
## Scale for colour is already present.
## Adding another scale for colour, which will replace the existing scale.
## Scale for colour is already present.
## Adding another scale for colour, which will replace the existing scale.
## Scale for colour is already present.
## Adding another scale for colour, which will replace the existing scale.
## Scale for colour is already present.
## Adding another scale for colour, which will replace the existing scale.
## Scale for colour is already present.
## Adding another scale for colour, which will replace the existing scale.
## Scale for colour is already present.
## Adding another scale for colour, which will replace the existing scale.
## Scale for colour is already present.
```

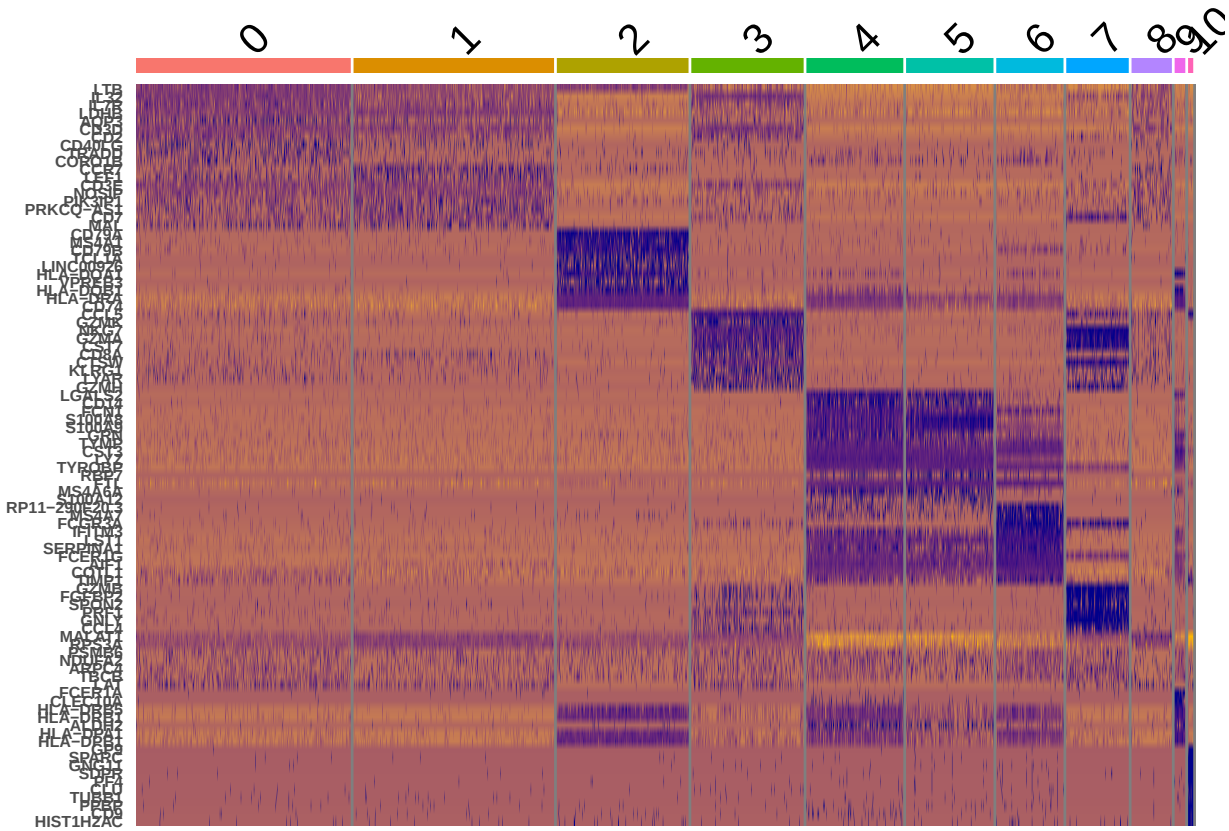
```
## Adding another scale for colour, which will replace the existing scale.
```




```
All.markers %>%
  group_by(cluster) %>%
  top_n(n = 10, wt = avg_log2FC) -> top10
DoHeatmap(pbmc, features = top10$gene)+ NoLegend()+
         scale_fill_gradient(high = 'darkblue', low = '#FFB900') +
         theme(axis.text = element_text(size=6, face = 'bold'))
```

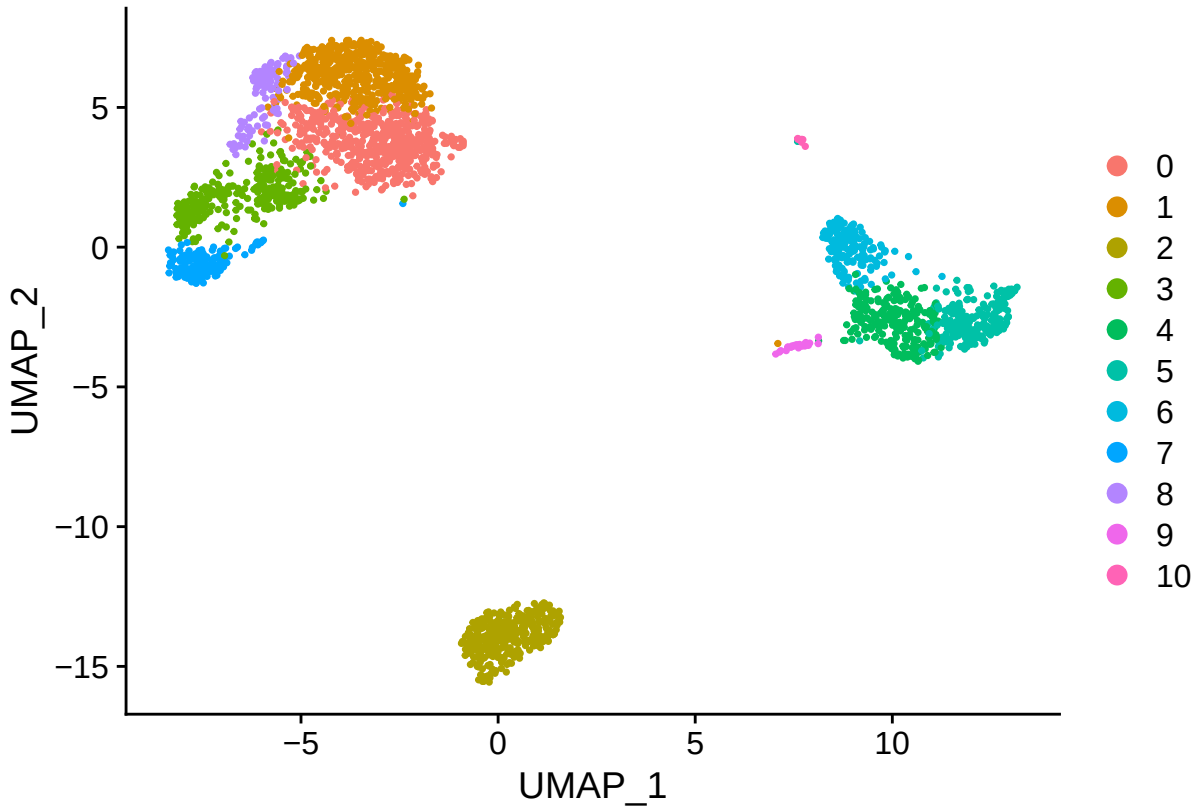
Higlight top 10 genes found in all clusters

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



##
Annotatimg Clusters

```
#pbmc<- readRDS('PBMC.rds')
#View(pbmc@meta.data)
DimPlot(pbmc, reduction = 'umap')
```




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

Set reference dataset

```
## snapshotDate(): 2021-10-19
## see ?celldex and browseVignettes('celldex') for documentation
## loading from cache
## see ?celldex and browseVignettes('celldex') for documentation
## loading from cache
```

```
View(as.data.frame(colData(ref)))
```

```
pbmc.counts<- GetAssayData(pbmc, slot = 'counts')
```

Get expression marix

```
pred <- SingleR(test = pbmc.counts,
                ref = ref,
                labels = ref$label.main)
```

Get prediction matrix from SingleR

```
pred$pruned.labels<- gsub('_'," ", as.character(pred$pruned.labels))
table(pred$pruned.labels)
```

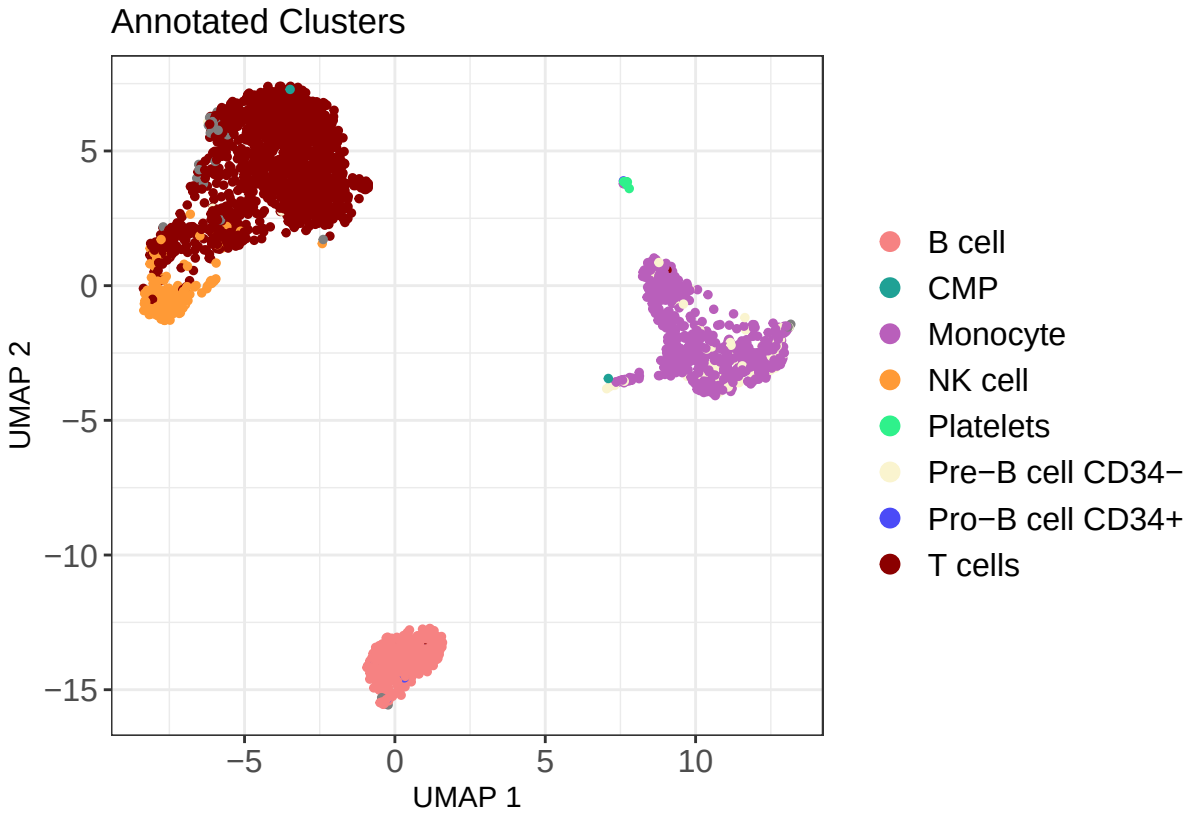
Tidy and inspect

```
##
##           B cell              CMP         Monocyte          NK cell
##              326                2              607              183
##        Platelets Pre-B cell CD34- Pro-B cell CD34+          T cells
##               12               58                5             1381
```

```
pbmc$annotation <- pred$pruned.labels[match(rownames(pbmc@meta.data), rownames(pred))]
```

Assign labels to seurat object

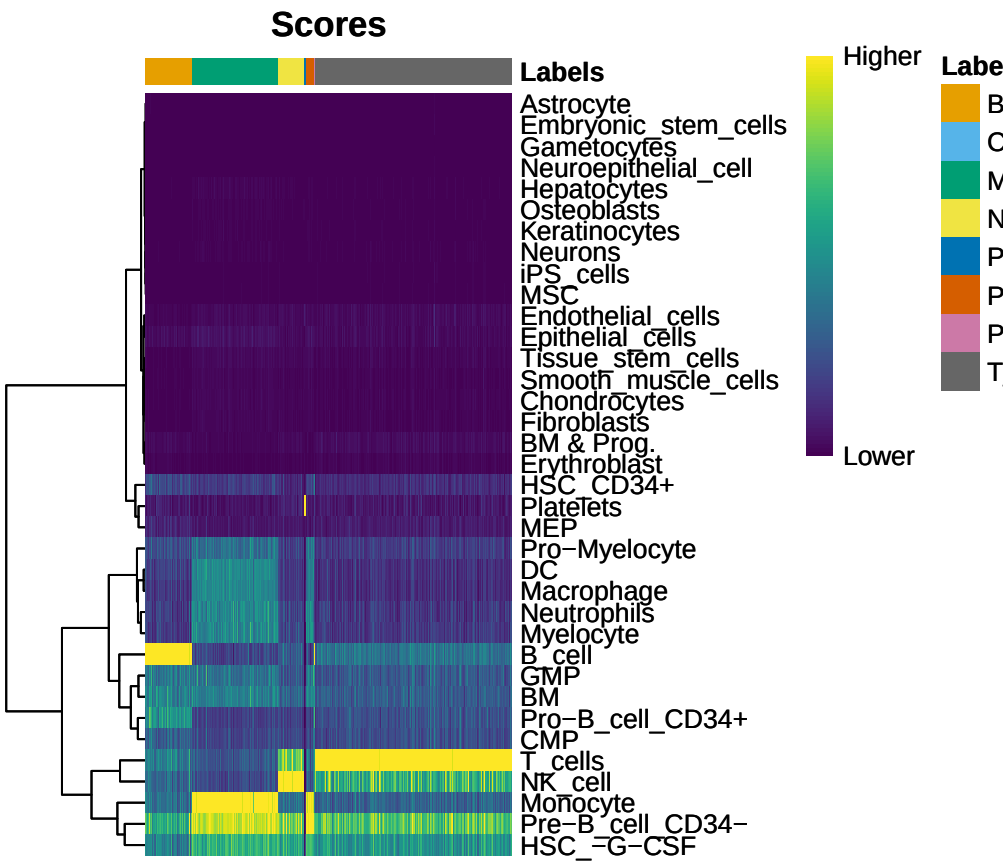
```
cols2 <- c('B cell'='#F68282','CMP'='#1FA195','Monocyte'='#B95FBB',
           'NK cell'='#ff9a36','Platelets'='#2FF18B','Pre-B cell CD34-'='#faf4cf',
           'Pro-B cell CD34+'='#4B4BF7','T cells'='darkred')

Final<-DimPlot(pbmc, reduction = 'umap', group.by = 'annotation', label = F, cols = cols2, pt.size = 1)
Final<- Final+ theme_bw()+labs(x='UMAP 1', y='UMAP 2') + ggtitle('Annotated Clusters')+
    theme(axis.text = element_text(size = 12), legend.text = element_text(size = 12))
Final
```



Final plotting

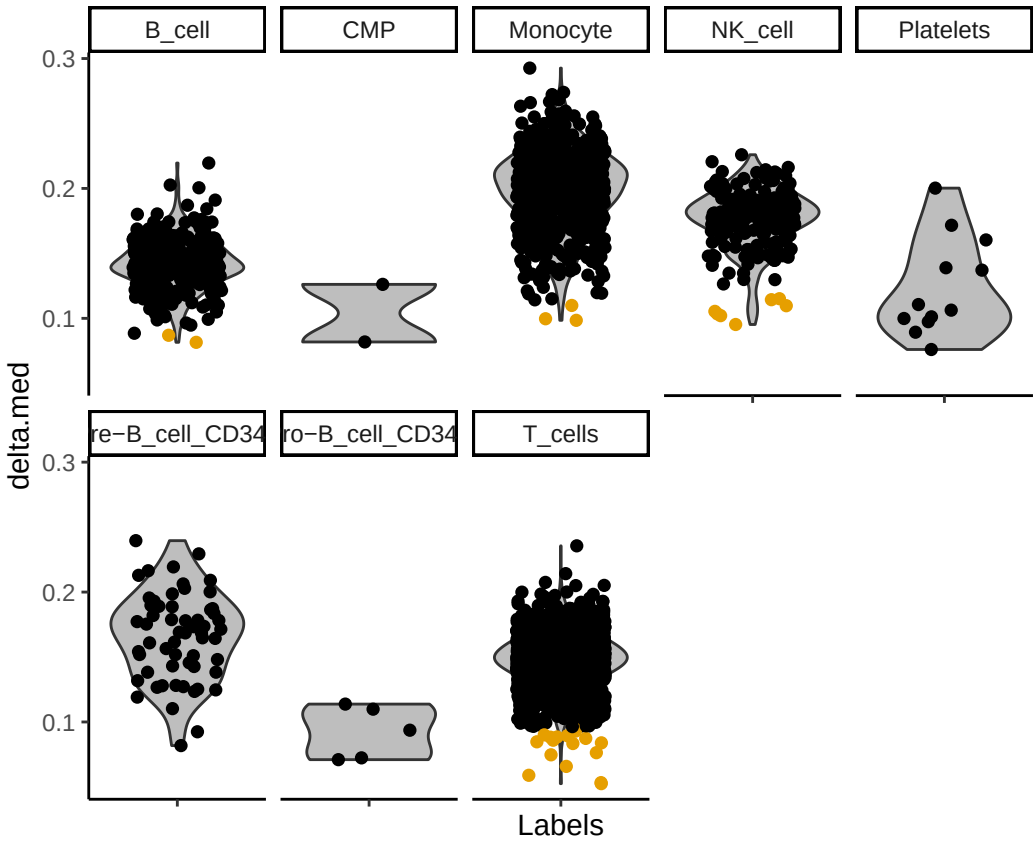
Annotation diagnostics

```
#pred
#pred$scores
plotScoreHeatmap(pred)
```



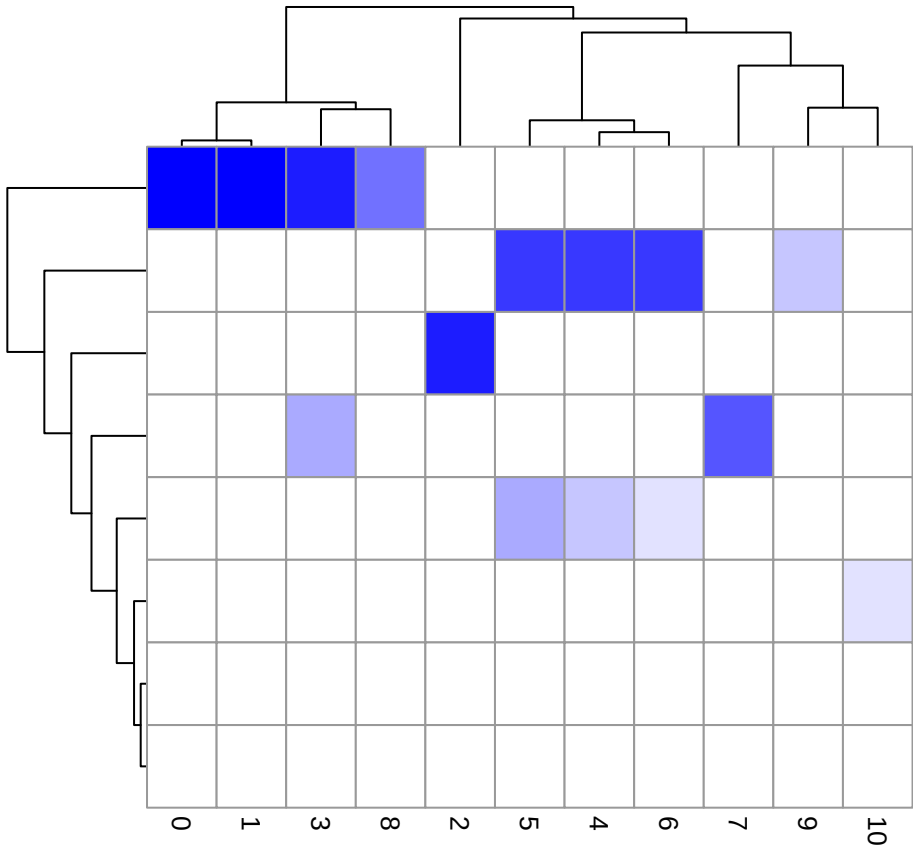
Based on the scores within cells

```
plotDeltaDistribution(pred)
```



Based on deltas across cells

```
pheatmap.data <- table(Assigned=pred$labels, Clusters=pbmc$seurat_clusters)
pheatmap(log10(pheatmap.data+10), color = colorRampPalette(c('white','blue'))(10))
```



Comparing to unsupervised clustering

Table for expected doublets

Rate (%) |Cells Loaded|Cells Recovered 0.40%——————|800——————-|500——————— 0.80%——————|1,600——————|1,000——————— 1.60%——————|3,200——————|2,000——————- 2.30%——————|4,800——————|3,000——————- 3.10%——————|6,400——————|4,000——————- 3.90%——————|8,000——————|5,000——————- 4.60%——————|9,600——————|6,000——————- 5.40%——————|11,200——————|7,000——————- 6.10%——————|12,800——————|8,000——————- 6.90%——————|14,400——————|9,000——————- 7.60%——————|16,000——————|10,000———————