CD45POS_0816

Changhua Yu

8/16/2019

```
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(Seurat)
```

```
## Registered S3 method overwritten by 'R.oo':
##   method        from
##   throw.default R.methodsS3
```

```
library(purrr)
library(cowplot)
```

```
##
## ************************************************************
```

```
## Note: As of version 1.0.0, cowplot does not change the
```

```
##   default ggplot2 theme anymore. To recover the previous
```

```
##   behavior, execute:
##   theme_set(theme_cowplot())
```

```
## ************************************************************
```

```
library(parallel)
library(roxygen2)
library(reshape2)
library(tibble)
library(ggplot2)
source('../R/celltype_assign.R')
source('../R/cell_cytometry.R')
source("../R/DE_analysis.R")
```

Read in the Count matrix

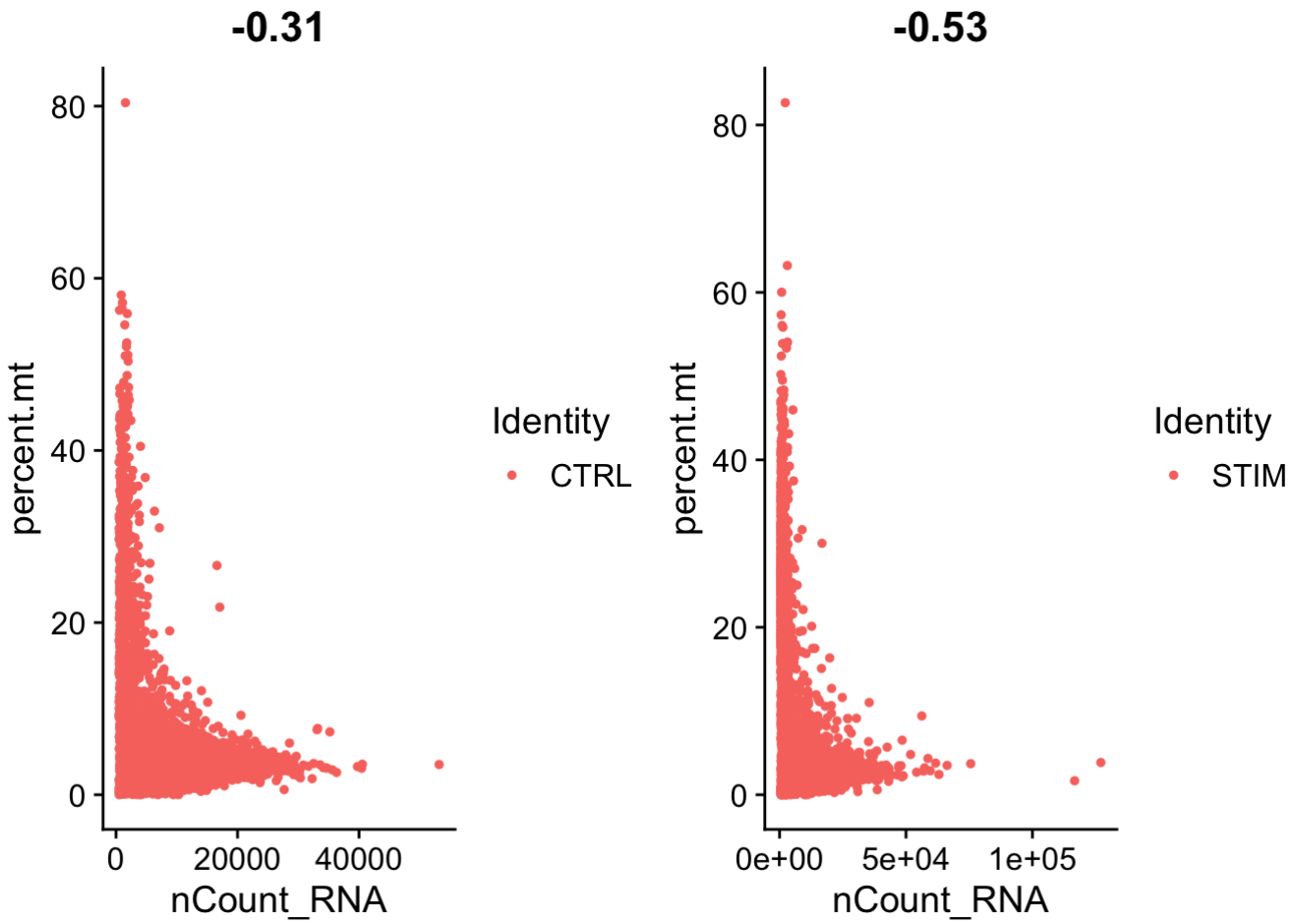
```
path1 = "../data/KO0531/"
path2 = "../data/WT0531/"
ko.mat = Read10X(data.dir = path1)
wt.mat = Read10X(data.dir = path2)

ctrl = CreateSeuratObject(counts = wt.mat, project = 'CTRL', min.cells = 3, min.features
= 200)
stim = CreateSeuratObject(counts = ko.mat, project = 'STIM', min.cells = 3, min.features
= 200)
```

```
ctrl$stim = "CTRL"
ctrl[["percent.mt"]] <- PercentageFeatureSet(ctrl, pattern = "^MT-|^mt")
stim$stim = "STIM"
stim[["percent.mt"]] <- PercentageFeatureSet(stim, pattern = "^MT-|^mt")
```

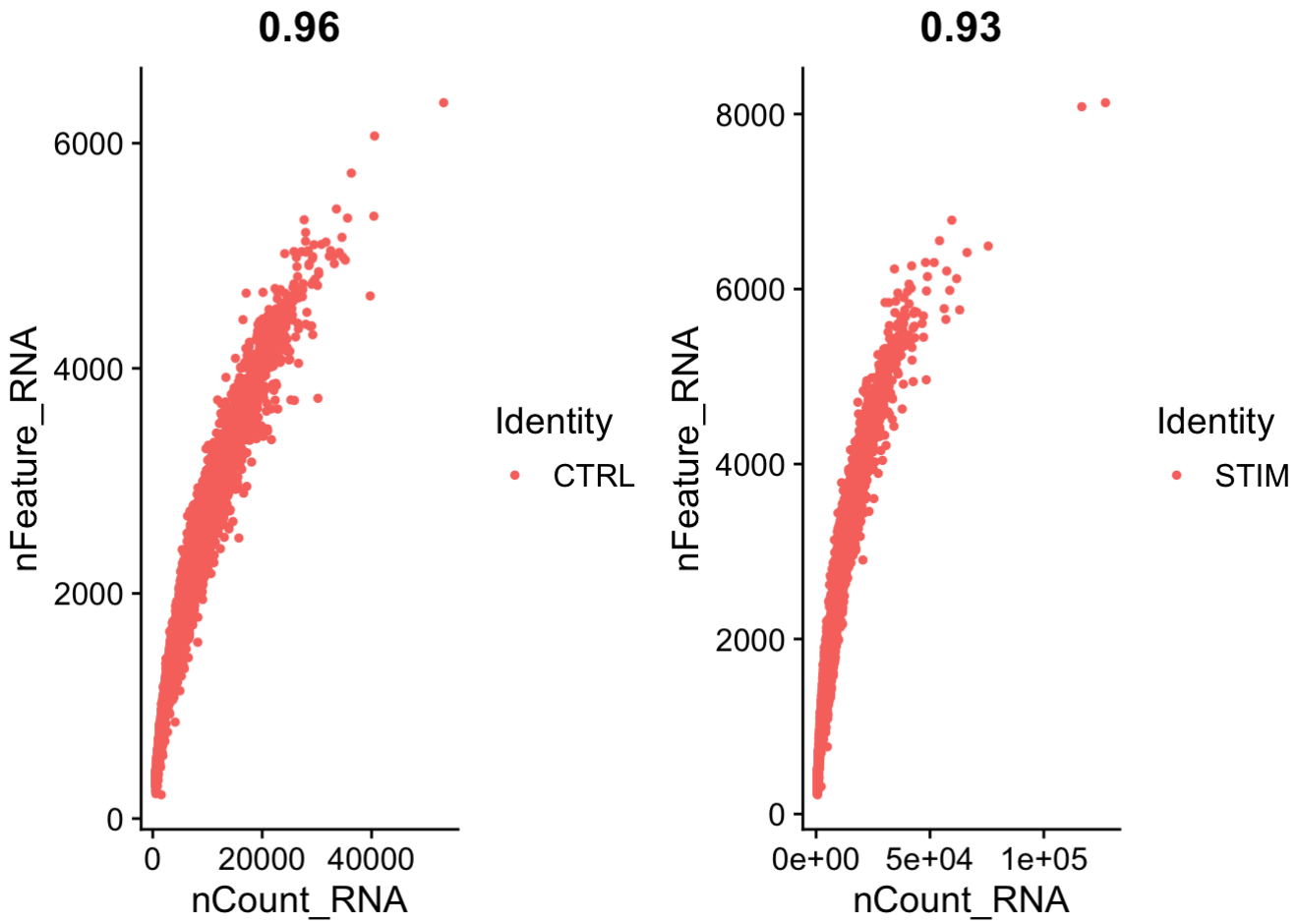
Plot out mitochondria percent distribution

```
plot1 <- FeatureScatter(ctrl, feature1 = "nCount_RNA", feature2 = "percent.mt")
plot2 <- FeatureScatter(stim, feature1 = "nCount_RNA", feature2 = "percent.mt")
CombinePlots(plots = list(plot1, plot2))
```



Plot out nFeature distribution

```
plot1 <- FeatureScatter(ctrl, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
plot2 <- FeatureScatter(stim, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
CombinePlots(plots = list(plot1, plot2))
```



Filter out abnormal cells

> Based on the mt percent distribution and nFeature distribution

```
ctrl <- subset(ctrl, subset = nFeature_RNA > 500 & nFeature_RNA < 6000 & percent.mt < 25
)
print(ctrl)
```

```
## An object of class Seurat
## 15018 features across 7055 samples within 1 assay
## Active assay: RNA (15018 features)
```

```
stim <- subset(stim, subset = nFeature_RNA > 500 & nFeature_RNA < 6000 & percent.mt < 25
)
print(stim)
```

```
## An object of class Seurat
## 15171 features across 4604 samples within 1 assay
## Active assay: RNA (15171 features)
```

```
ctrl <- NormalizeData(ctrl, normalization.method = "LogNormalize", scale.factor = 10000)
ctrl <- FindVariableFeatures(ctrl, selection.method = "vst", nfeatures = 2500)

stim <- NormalizeData(stim, normalization.method = "LogNormalize", scale.factor = 10000)
stim <- FindVariableFeatures(stim, selection.method = "vst", nfeatures = 2500)
```

Integrate two sample together

```
immune.anchors <- FindIntegrationAnchors(object.list = list(ctrl, stim))
```

```
## Warning in CheckDuplicateCellNames(object.list = object.list): Some cell
## names are duplicated across objects provided. Renaming to enforce unique
## cell names.
```

```
## Computing 2000 integration features
```

```
## Scaling features for provided objects
```

```
## Finding all pairwise anchors
```

```
## Running CCA
```

```
## Merging objects
```

```
## Finding neighborhoods
```

```
## Finding anchors
```

```
##  Found 10740 anchors
```

```
## Filtering anchors
```

```
##  Retained 7435 anchors
```

```
## Extracting within-dataset neighbors
```

```
immune.combined <- IntegrateData(anchorset = immune.anchors)
```

```
## Merging dataset 2 into 1
```

```
## Extracting anchors for merged samples
```

```
## Finding integration vectors
```

```
## Finding integration vector weights
```

```
## Integrating data
```

Proceed the anchored clustering

```
DefaultAssay(immune.combined) <- "integrated"
# Run the standard workflow for visualization and clustering
immune.combined <- ScaleData(immune.combined, verbose = FALSE)
immune.combined <- RunPCA(immune.combined, npcs = 30, verbose = FALSE)
# t-SNE and Clustering
immune.combined <- RunUMAP(immune.combined, reduction = "pca",dims = 1:15)
```

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

```
## 14:37:15 Read 11659 rows and found 15 numeric columns
```

```
## 14:37:15 Using Annoy for neighbor search, n_neighbors = 30
```

```
## 14:37:15 Building Annoy index with metric = cosine, n_trees = 50
```

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

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

```
## **************************************************|
## 14:37:16 Writing NN index file to temp file /var/folders/pw/6v34dsx97wjdql3g4qhfjpj80
000gn/T//RtmpdrjIGQ/file518f59327586
## 14:37:16 Searching Annoy index using 1 thread, search_k = 3000
## 14:37:19 Annoy recall = 100%
## 14:37:19 Commencing smooth kNN distance calibration using 1 thread
## 14:37:20 Initializing from normalized Laplacian + noise
## 14:37:20 Commencing optimization for 200 epochs, with 477176 positive edges
## 14:37:26 Optimization finished
```

```
immune.combined <- FindNeighbors(immune.combined, reduction = "pca",dims = 1:15)
```

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

```
immune.combined <- FindClusters(immune.combined, resolution = 0.6)
```

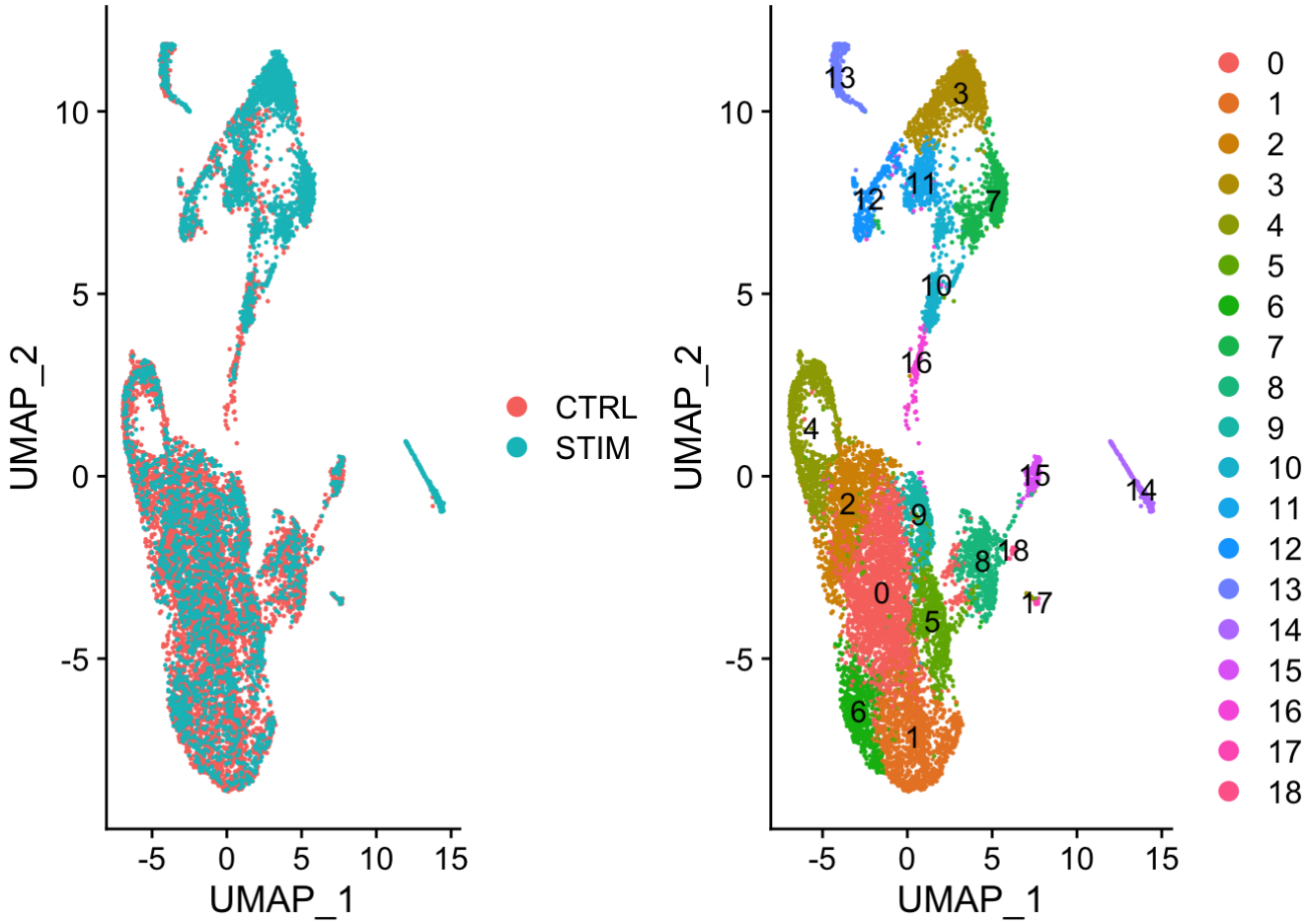
```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 11659
## Number of edges: 392061
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8841
## Number of communities: 19
## Elapsed time: 1 seconds
```

Visualizing UMAP clustering

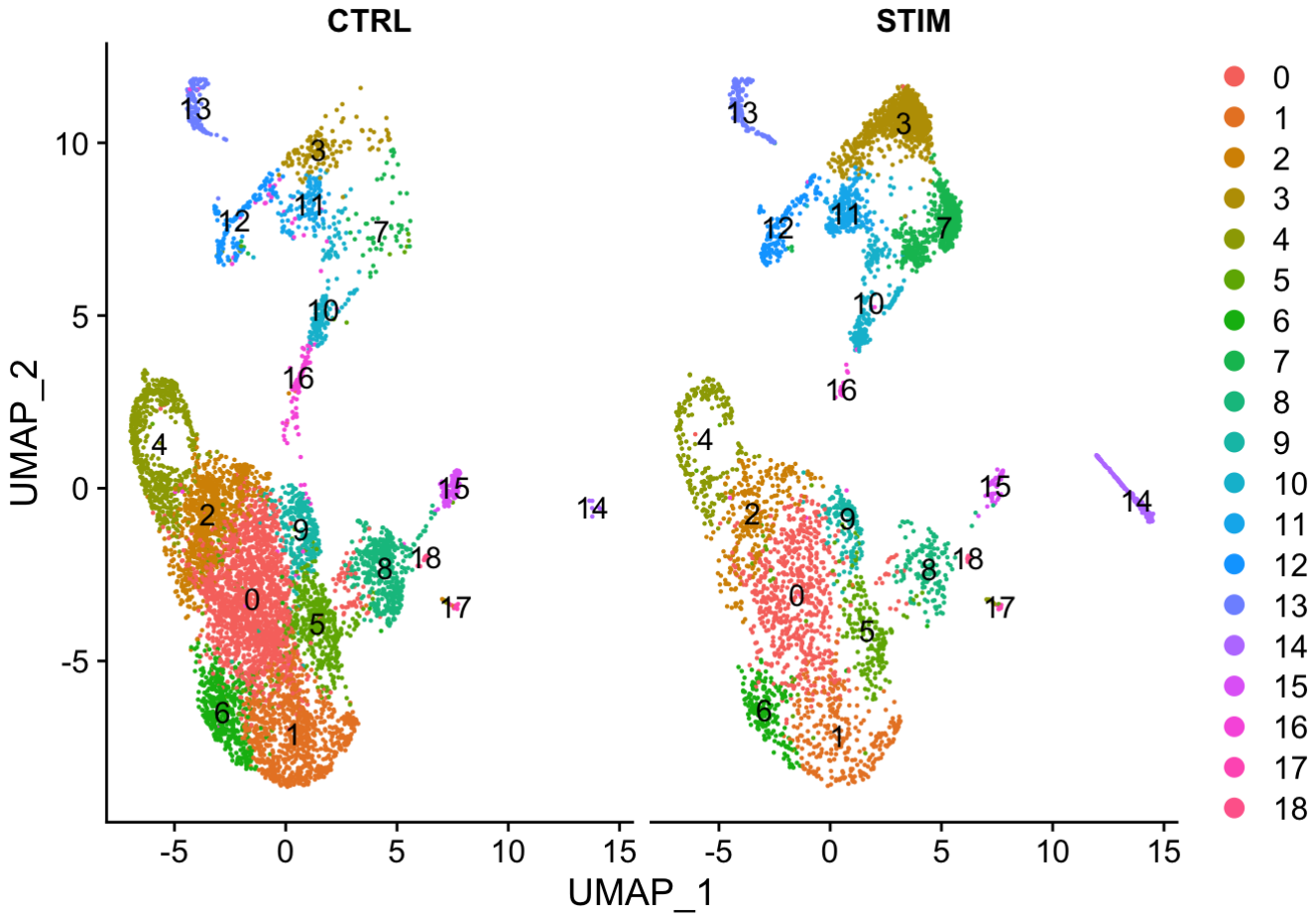
```
p1 <- DimPlot(immune.combined, reduction = "umap", group.by = "stim")
p2 <- DimPlot(immune.combined, reduction = "umap", label = TRUE)
```

```
## Warning: Using `as.character()` on a quosure is deprecated as of rlang 0.3.0.
## Please use `as_label()` or `as_name()` instead.
## This warning is displayed once per session.
```

```
plot_grid(p1, p2)
```



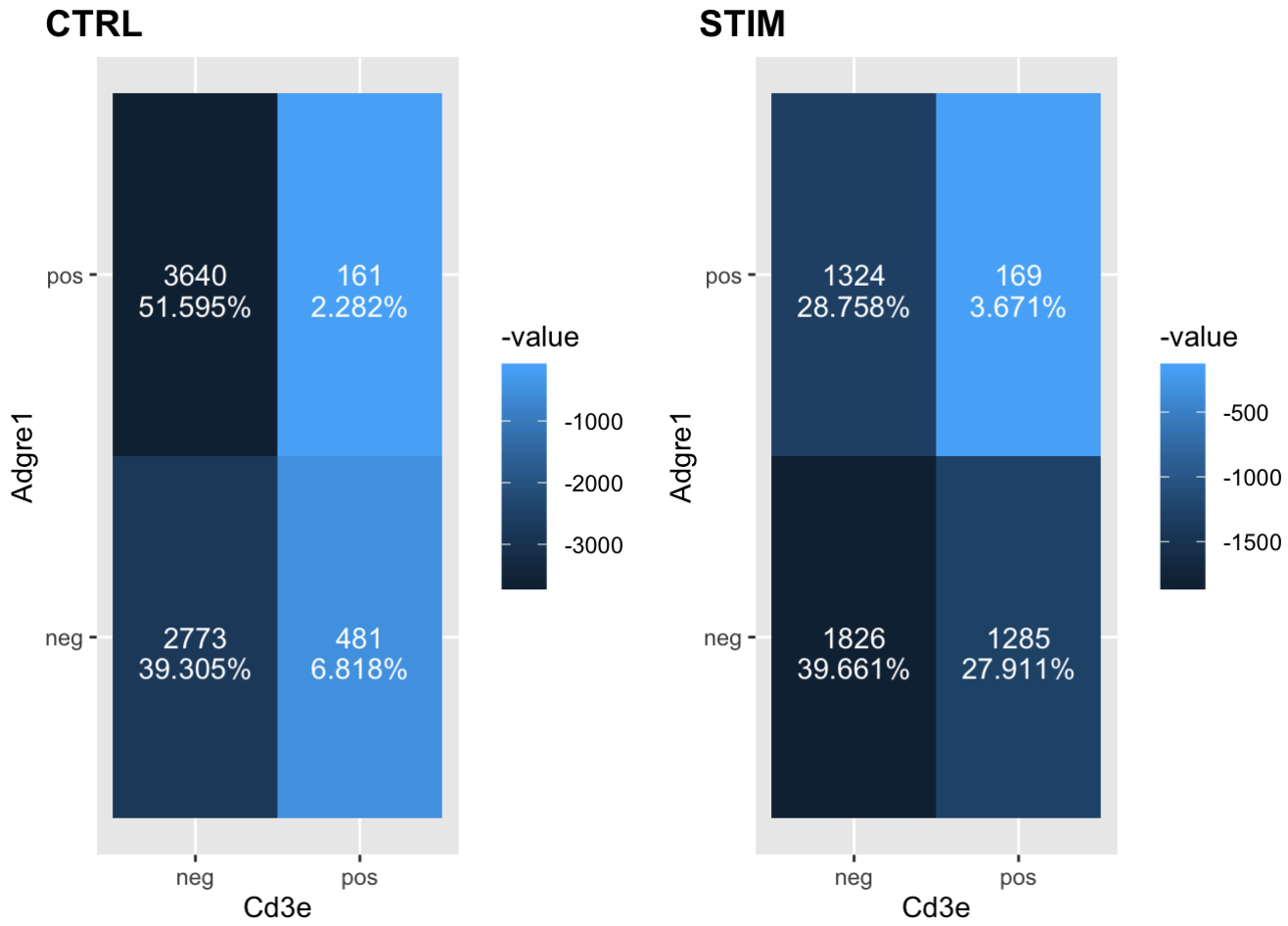

```
DimPlot(immune.combined, reduction = "umap", split.by = "stim",label = T)
```



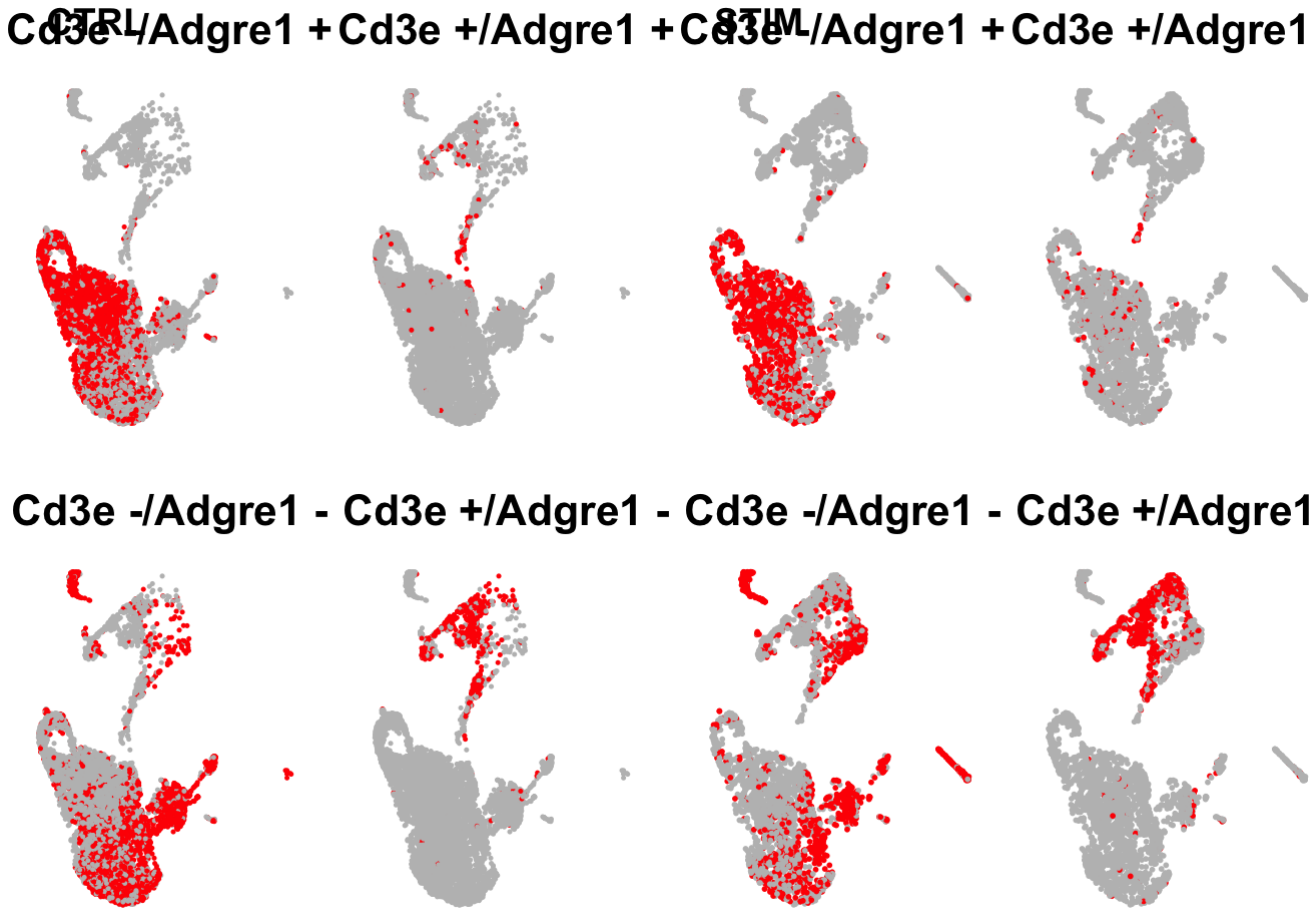
In Silico Cytometry with Cd3e and Adgre1

```
immune.combined.cyto.Adgre1 = Build.Cyto(immune.combined,"Cd3e","Adgre1",x.thresh = 0.1,
y.thresh = 0.1)
Plot.Cyto.Count(seurat.ob = immune.combined.cyto.Adgre1,
                x= "Cd3e",
                y = "Adgre1",split = T)
```

```
## Using x.label, y.label as id variables
## Using x.label, y.label as id variables
```



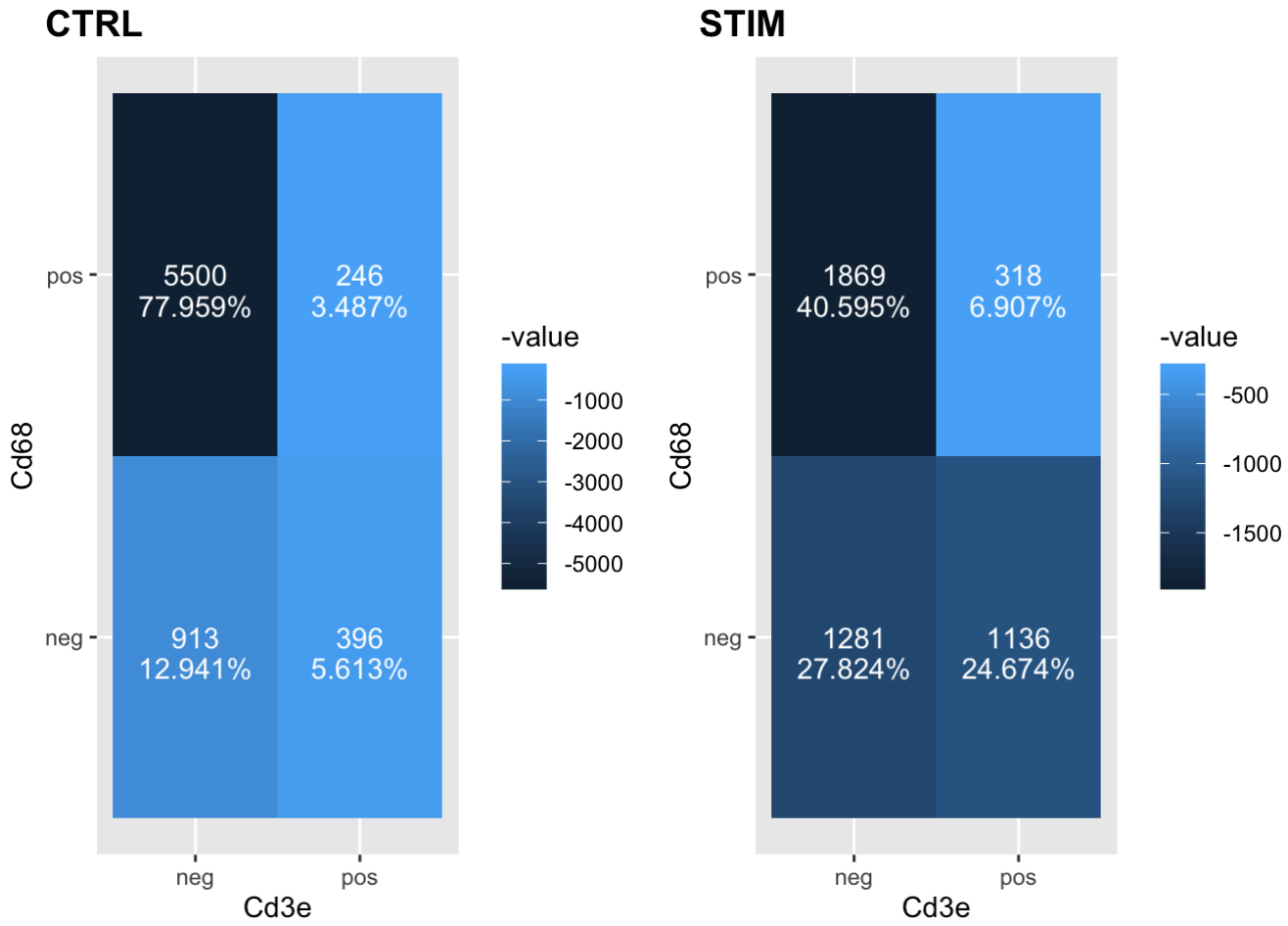

```
p = Plot.Cyto.Cluster(seurat.ob = immune.combined.cyto.Adgre1,
                      x= "Cd3e",
                      y = "Adgre1",split = T)
plot_grid(p$ctrl,p$stim,labels = c('CTRL','STIM'))
```



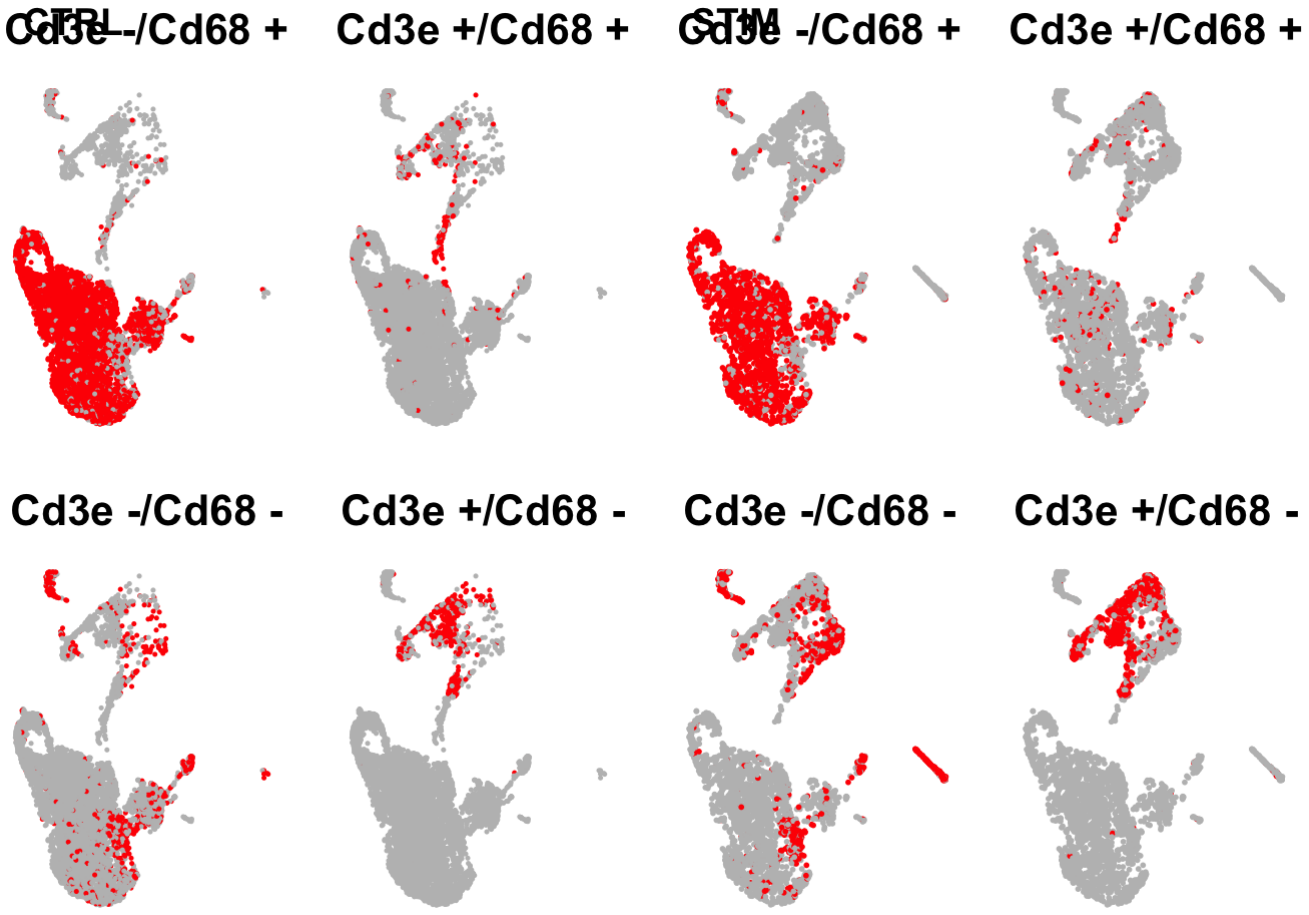
In Silico Cytometry with Cd3e and Cd68

```
immune.combined.cyto.cd68 = Build.Cyto(immune.combined,"Cd3e","Cd68",x.thresh = 0.1,y.th
resh = 0.1)
Plot.Cyto.Count(seurat.ob = immune.combined.cyto.cd68,
                x= "Cd3e",
                y = "Cd68",split = T)
```

```
## Using x.label, y.label as id variables
## Using x.label, y.label as id variables
```




```
p = Plot.Cyto.Cluster(seurat.ob = immune.combined.cyto.cd68,
                      x= "Cd3e",
                      y = "Cd68",split = T)
plot_grid(p$ctrl,p$stim,labels = c('CTRL','STIM'))
```



Extract out the macrophage population for reclustering

```
macrophage = Recluster.Quadrant(immune.combined.cyto.cd68,n.quadrant = 4)
DefaultAssay(macrophage) <- "integrated"
macrophage <- RunPCA(macrophage, npcs = 30, verbose = FALSE)
# t-SNE and Clustering
macrophage <- RunUMAP(macrophage, reduction = "pca",dims = 1:15)
```

```
## 14:37:43 Read 7369 rows and found 15 numeric columns
```

```
## 14:37:43 Using Annoy for neighbor search, n_neighbors = 30
```

```
## 14:37:43 Building Annoy index with metric = cosine, n_trees = 50
```

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

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

```
## ************************************************|
## 14:37:44 Writing NN index file to temp file /var/folders/pw/6v34dsx97wjdql3g4qhfjpj80
000gn/T//RtmpdrjIGQ/file518f3393ba15
## 14:37:44 Searching Annoy index using 1 thread, search_k = 3000
## 14:37:46 Annoy recall = 100%
## 14:37:46 Commencing smooth kNN distance calibration using 1 thread
## 14:37:46 Initializing from normalized Laplacian + noise
## 14:37:46 Commencing optimization for 500 epochs, with 300462 positive edges
## 14:37:55 Optimization finished
```

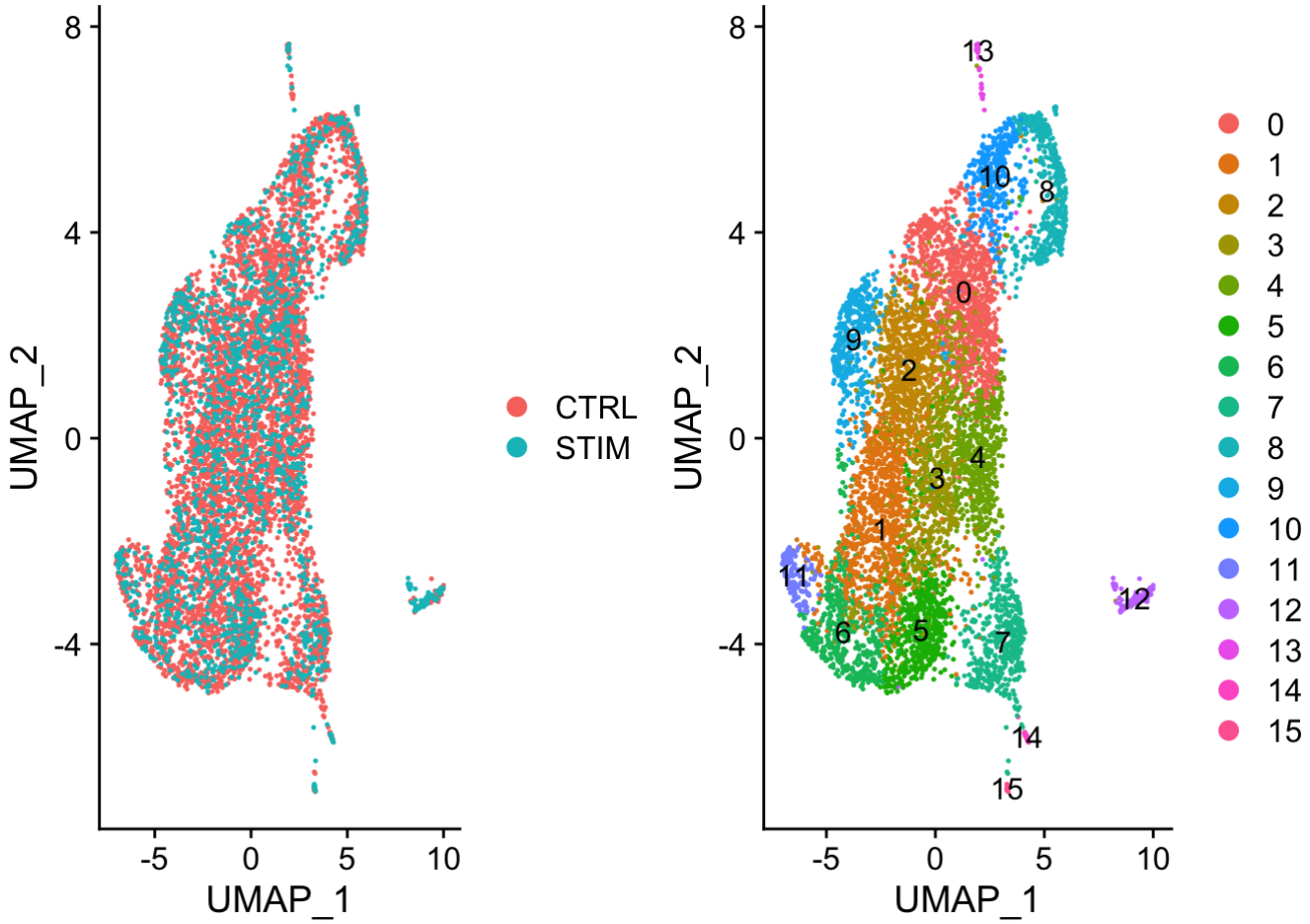
```
macrophage <- FindNeighbors(macrophage, reduction = "pca",dims = 1:15)
```

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

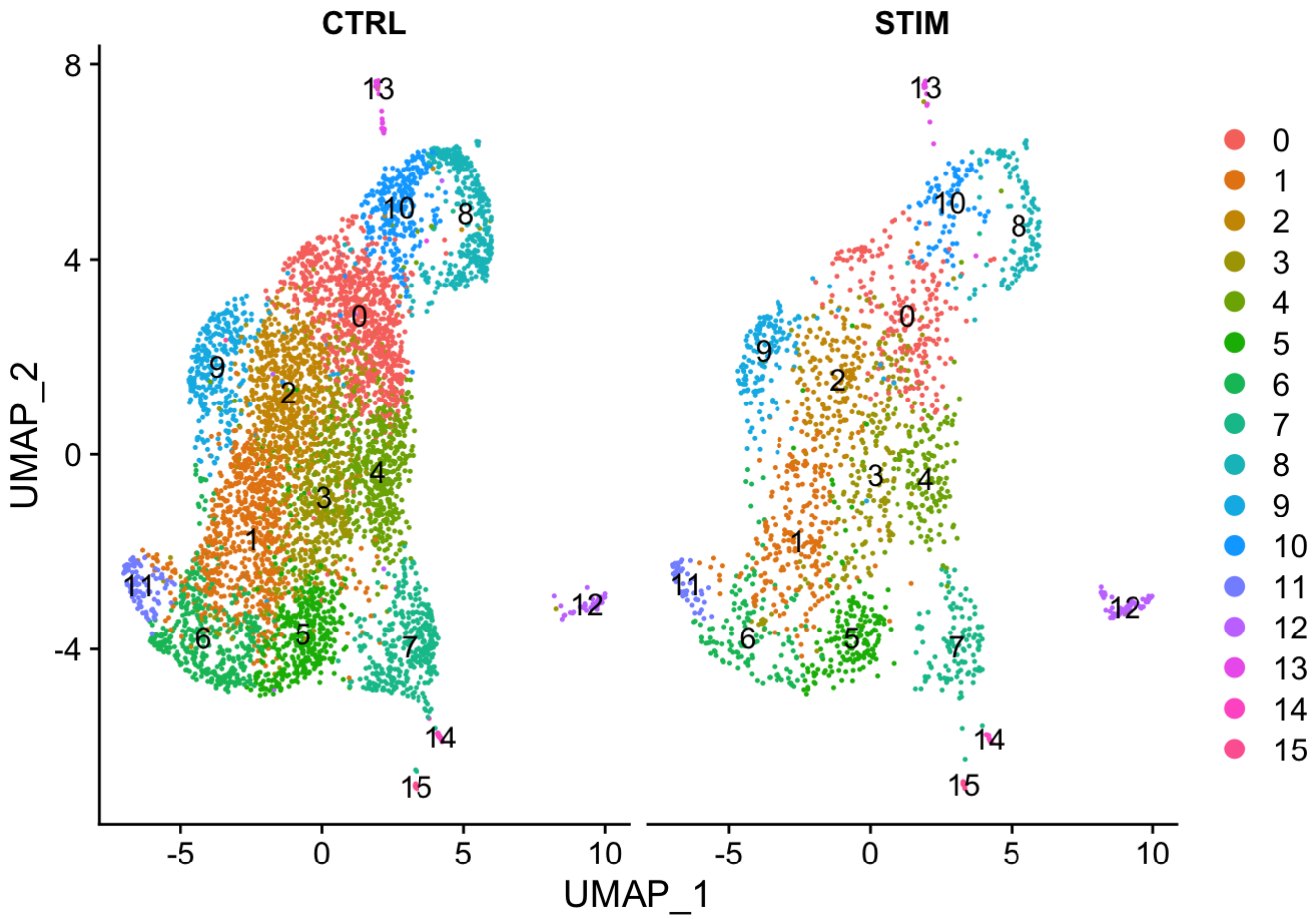
```
macrophage <- FindClusters(macrophage, resolution = 0.6)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 7369
## Number of edges: 239190
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8262
## Number of communities: 16
## Elapsed time: 0 seconds
```

```
p1 <- DimPlot(macrophage, reduction = "umap", group.by = "stim")
p2 <- DimPlot(macrophage, reduction = "umap", label = TRUE)
plot_grid(p1, p2)
```




```
DimPlot(macrophage, reduction = "umap", split.by = "stim",label = T)
```



Build the markers for ShinyApp output

```
macrophage.markers <- Build.ConserveMarkers.All(macrophage)
macrophage.markers.each = Find.Markers.Each(macrophage)
macrophage.diff = DE.Each.Cluster(macrophage)
macrophage.out = Shine.Out(ob = macrophage, diff = macrophage.diff, markers.each = macro
phage.markers.each, markers.conserved = macrophage.markers.flat)
```

```
saveRDS(object = macrophage.out,file = './Shiny_diff/input/macrophage_out_0820_0531.rds'
)
```

Extract out the population that is cd3e-/cd68-

```
other = Recluster.Quadrant(immune.combined.cyto.cd68,n.quadrant = 3)
DefaultAssay(other) <- "integrated"
other <- RunPCA(other, npcs = 30, verbose = FALSE)
# t-SNE and Clustering
other <- RunUMAP(other, reduction = "pca",dims = 1:15)
```

```
## 14:37:59 Read 2194 rows and found 15 numeric columns
```

```
## 14:37:59 Using Annoy for neighbor search, n_neighbors = 30
```

```
## 14:37:59 Building Annoy index with metric = cosine, n_trees = 50
```

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

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

```
## **************************************************|
## 14:37:59 Writing NN index file to temp file /var/folders/pw/6v34dsx97wjdql3g4qhfjpj80
000gn/T//RtmpdrjIGQ/file518f4ba85631
## 14:37:59 Searching Annoy index using 1 thread, search_k = 3000
## 14:38:00 Annoy recall = 100%
## 14:38:00 Commencing smooth kNN distance calibration using 1 thread
## 14:38:00 Initializing from normalized Laplacian + noise
## 14:38:00 Commencing optimization for 500 epochs, with 86470 positive edges
## 14:38:03 Optimization finished
```

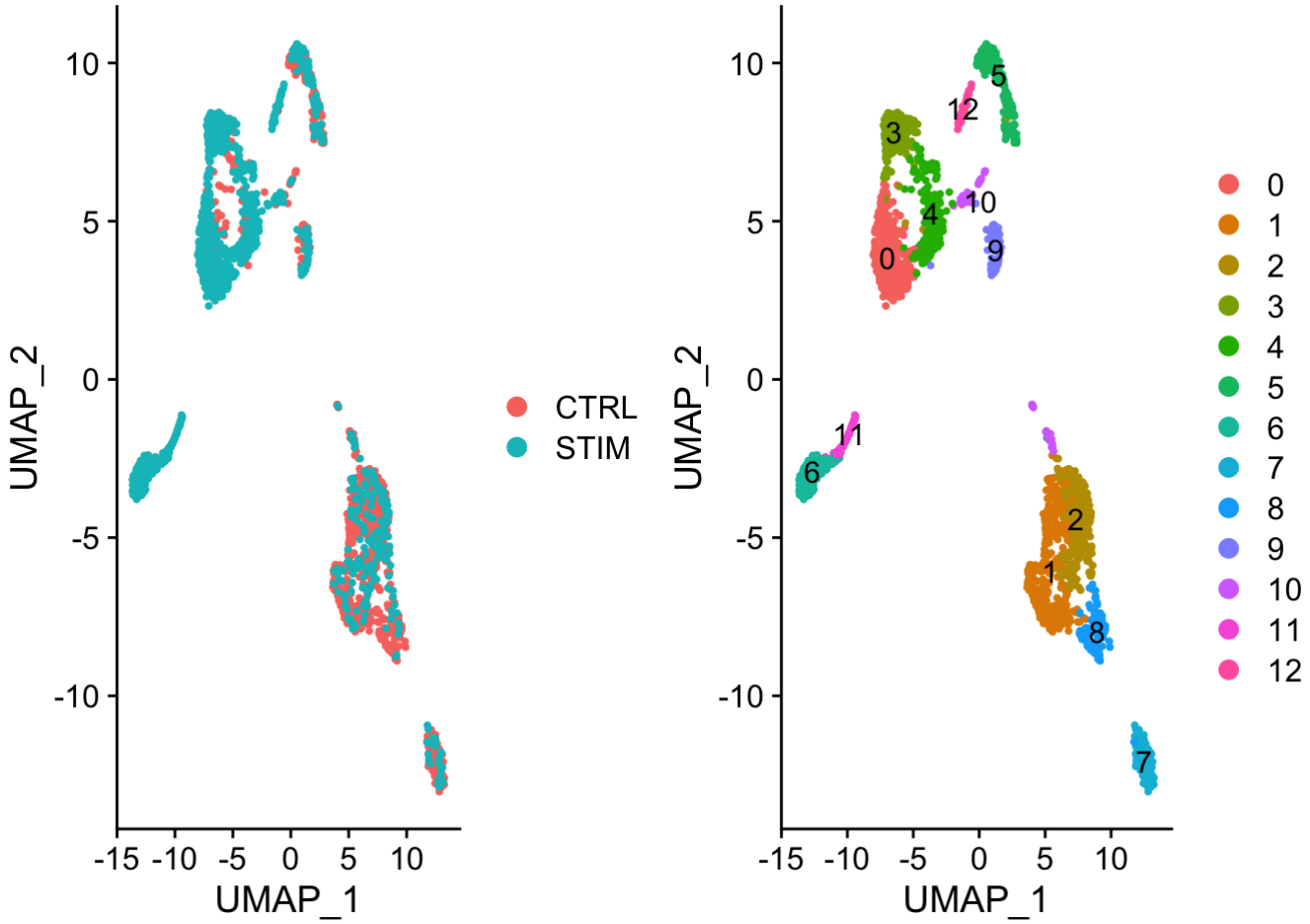
```
other <- FindNeighbors(other, reduction = "pca",dims = 1:15)
```

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

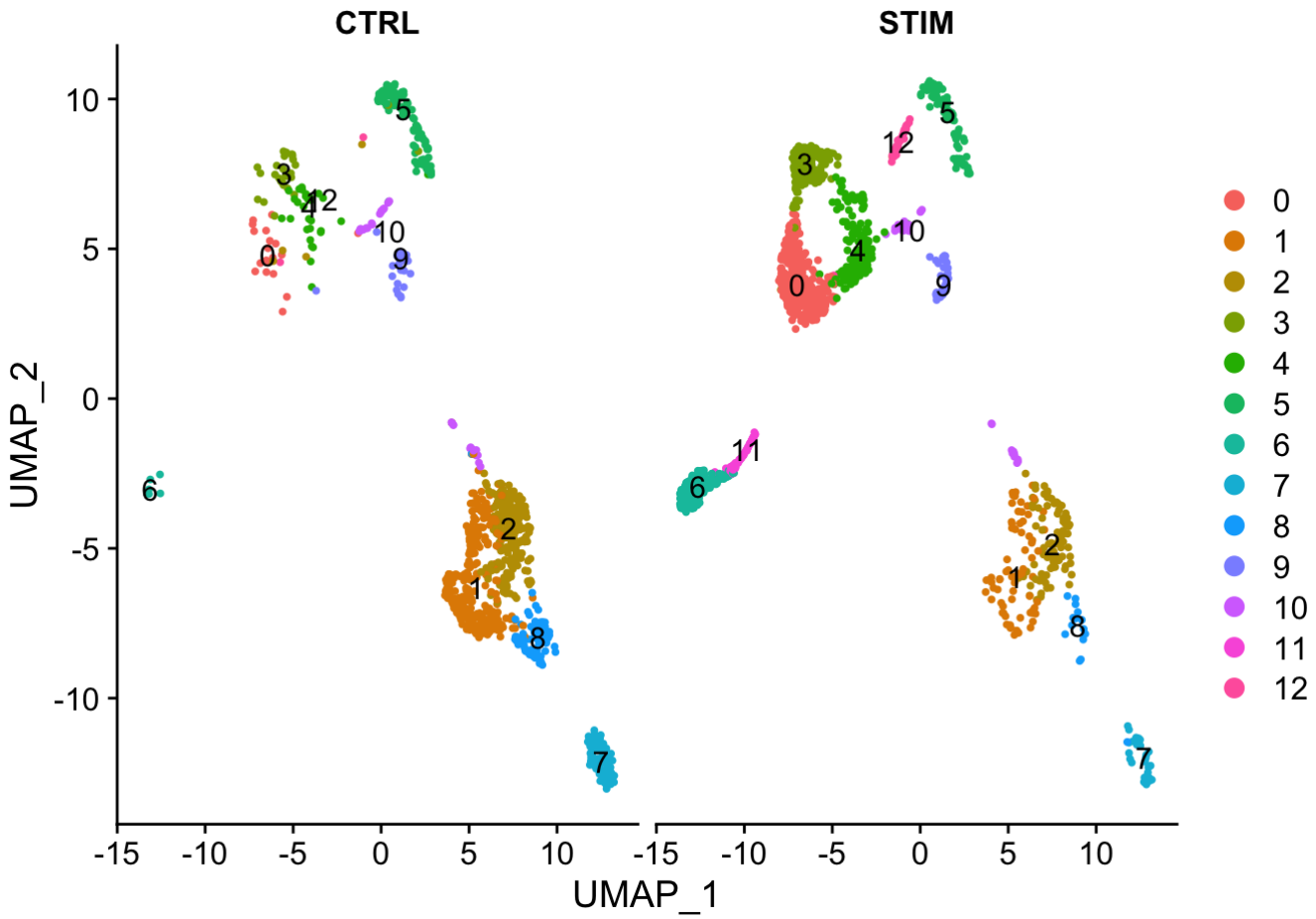
```
other <- FindClusters(other, resolution = 0.6)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2194
## Number of edges: 73133
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8951
## Number of communities: 13
## Elapsed time: 0 seconds
```

```
p1 <- DimPlot(other, reduction = "umap", group.by = "stim")
p2 <- DimPlot(other, reduction = "umap", label = TRUE)
plot_grid(p1, p2)
```




```
DimPlot(other, reduction = "umap", split.by = "stim",label = T)
```



Build the markers for ShinyApp output

```
other.markers = Build.ConserveMarkers.All(other)
other.markers.each <- Find.Markers.Each(other)
other.diff = DE.Each.Cluster(other)
other.out = Shine.Out(ob = other, diff = other.diff, markers.each = other.markers.each,m
arkers.conserved = other.markers.flat)
```

```
saveRDS(object = other.out,file = './Shiny_diff/input/other_out_0820_0531.rds')
```

Extract out the population that is T Cell

```
Tcell = Recluster.Quadrant(immune.combined.cyto.cd68,n.quadrant = 2)
DefaultAssay(Tcell) <- "integrated"
Tcell <- RunPCA(Tcell, npcs = 30, verbose = FALSE)
# t-SNE and Clustering
Tcell <- RunUMAP(Tcell, reduction = "pca",dims = 1:15)
```

```
## 14:38:05 Read 1532 rows and found 15 numeric columns
```

```
## 14:38:05 Using Annoy for neighbor search, n_neighbors = 30
```

```
## 14:38:05 Building Annoy index with metric = cosine, n_trees = 50
```

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

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

```
## **************************************************|
## 14:38:05 Writing NN index file to temp file /var/folders/pw/6v34dsx97wjdql3g4qhfjpj80
000gn/T//RtmpdrjIGQ/file518f669ff057
## 14:38:05 Searching Annoy index using 1 thread, search_k = 3000
## 14:38:06 Annoy recall = 100%
## 14:38:06 Commencing smooth kNN distance calibration using 1 thread
## 14:38:06 Initializing from normalized Laplacian + noise
## 14:38:06 Commencing optimization for 500 epochs, with 64154 positive edges
## 14:38:08 Optimization finished
```

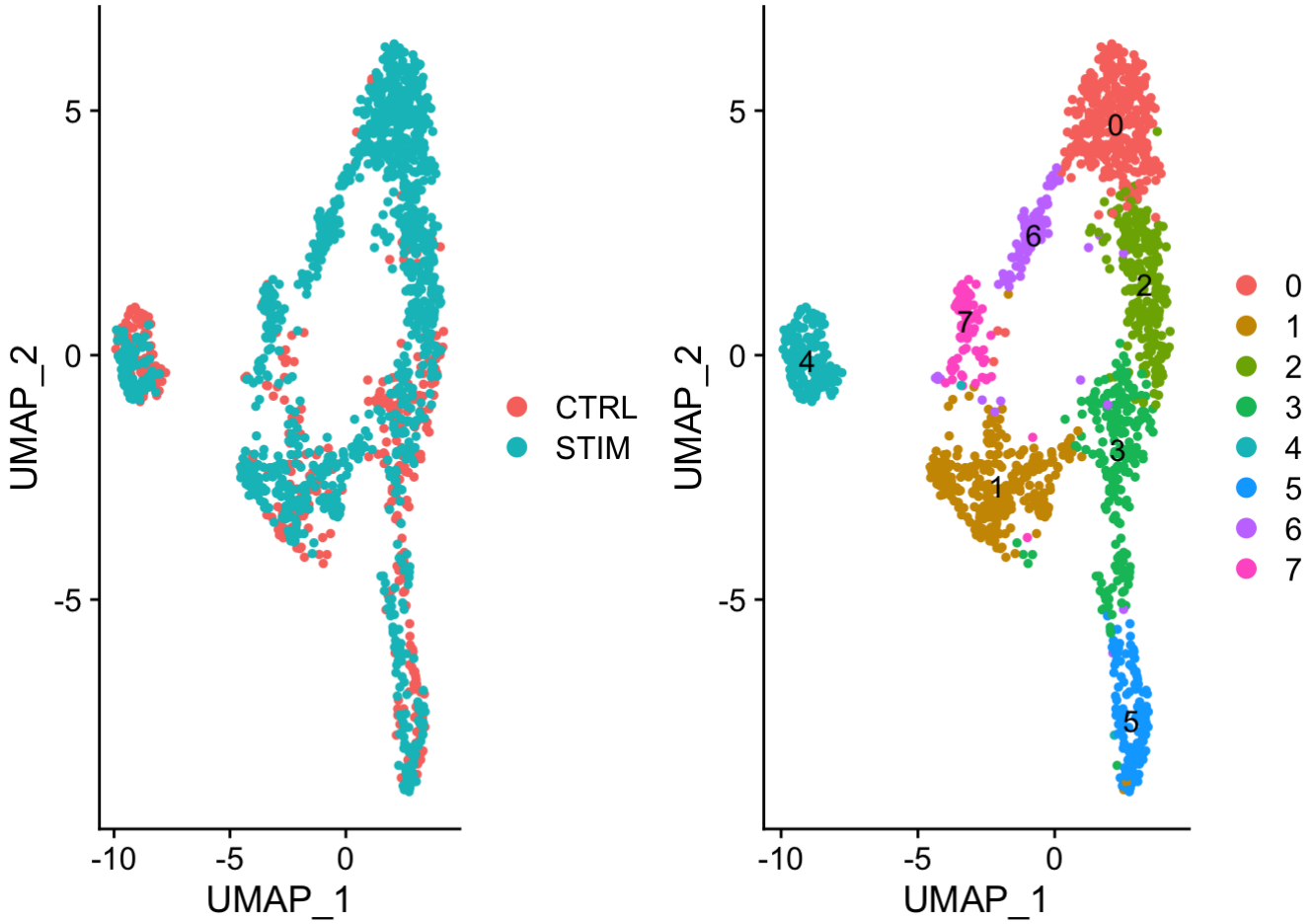
```
Tcell <- FindNeighbors(Tcell, reduction = "pca",dims = 1:15)
```

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

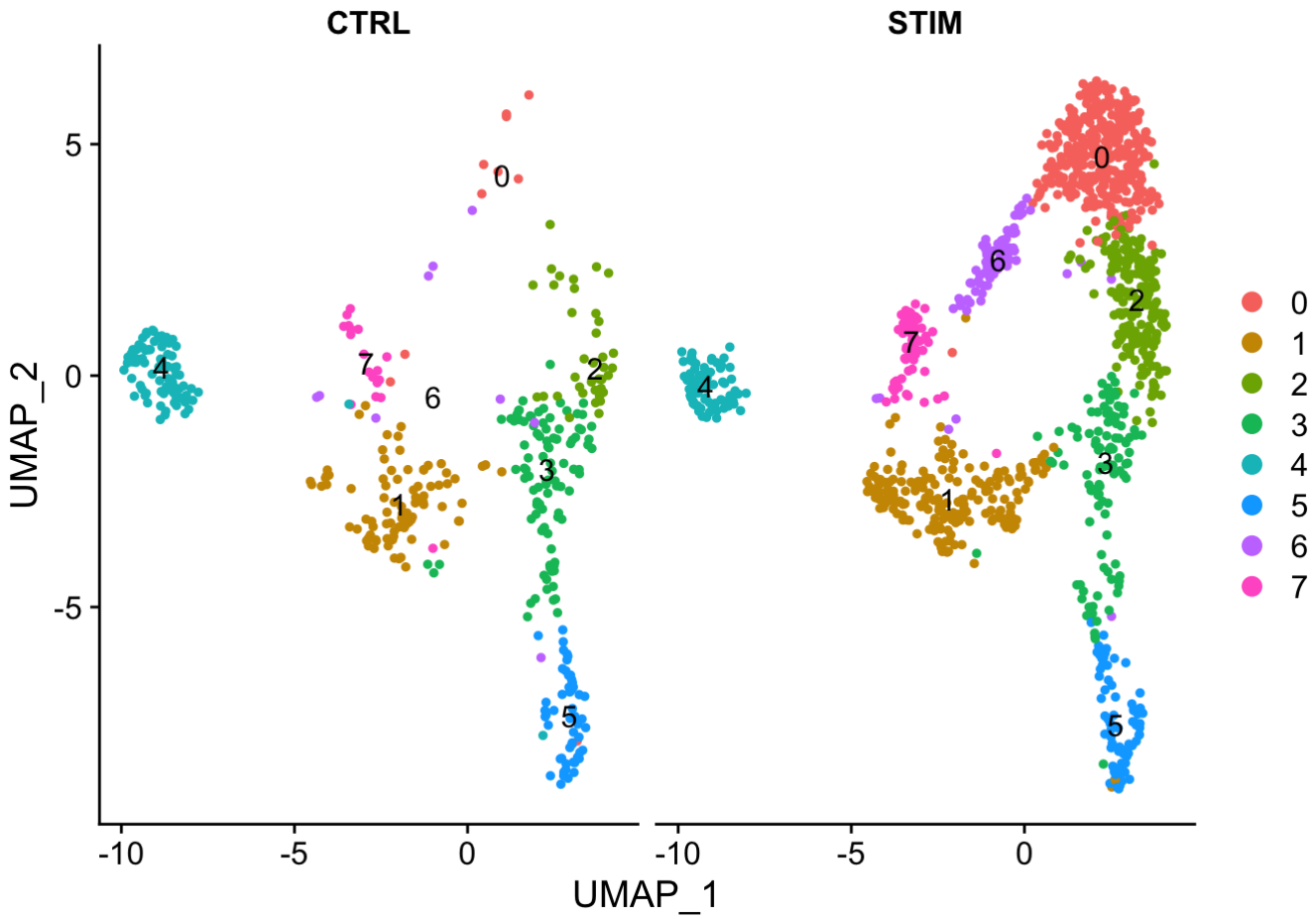
```
Tcell <- FindClusters(Tcell, resolution = 0.6)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 1532
## Number of edges: 59384
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8535
## Number of communities: 8
## Elapsed time: 0 seconds
```

```
p1 <- DimPlot(Tcell, reduction = "umap", group.by = "stim")
p2 <- DimPlot(Tcell, reduction = "umap", label = TRUE)
plot_grid(p1, p2)
```




```
DimPlot(Tcell, reduction = "umap", split.by = "stim",label = T)
```



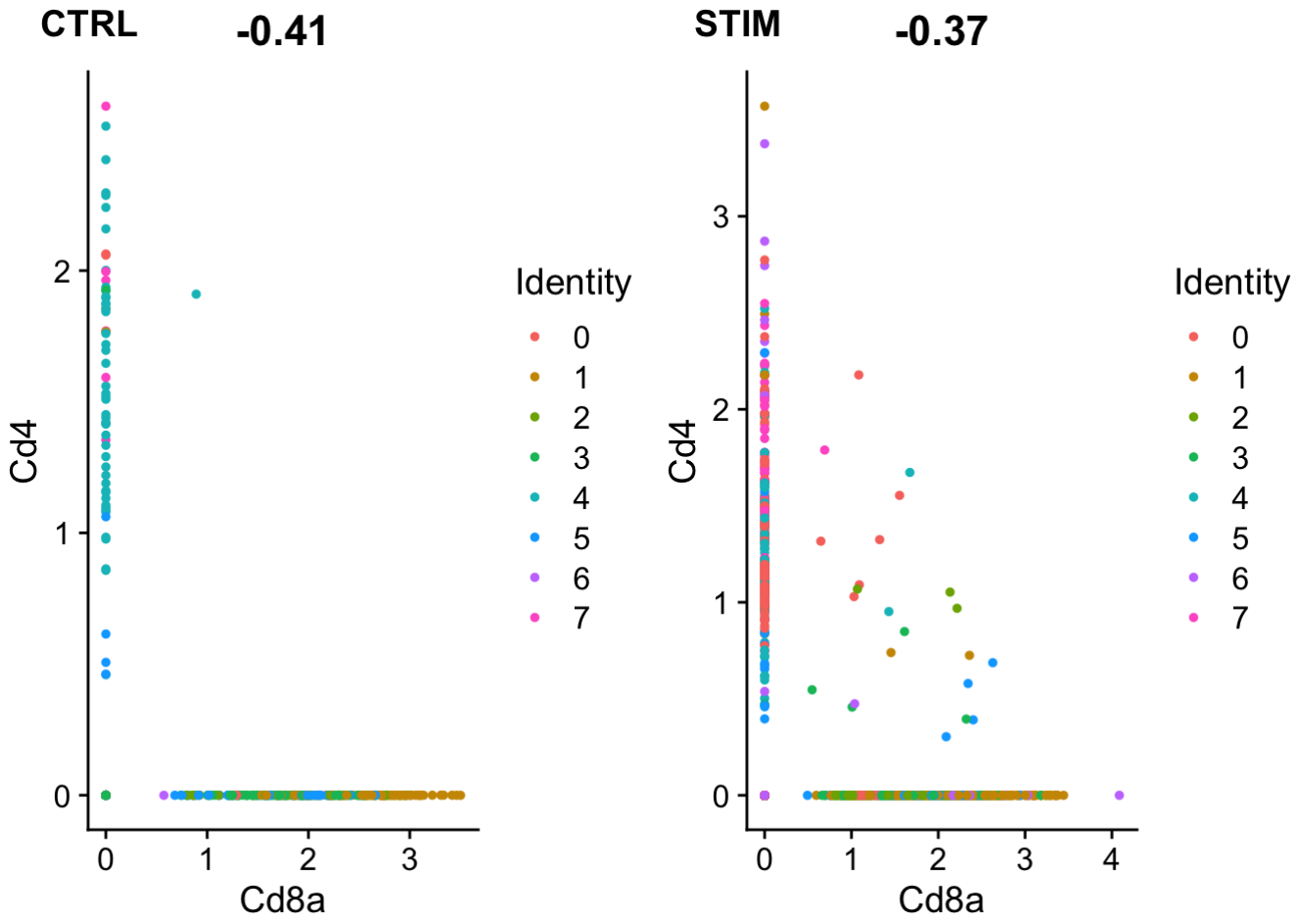
Expore the ShinyApp output for T Cell

```
T.markers = Build.ConserveMarkers.All(Tcell)
T.markers.each <- Find.Markers.Each(Tcell)
T.diff = DE.Each.Cluster(Tcell)
T.out = Shine.Out(ob = Tcell, diff = T.diff, markers.each = T.markers.each,markers.conse
rved = T.markers.flat)
```

```
saveRDS(object = T.out,file = './Shiny_diff/input/T_out_0820.rds')
```

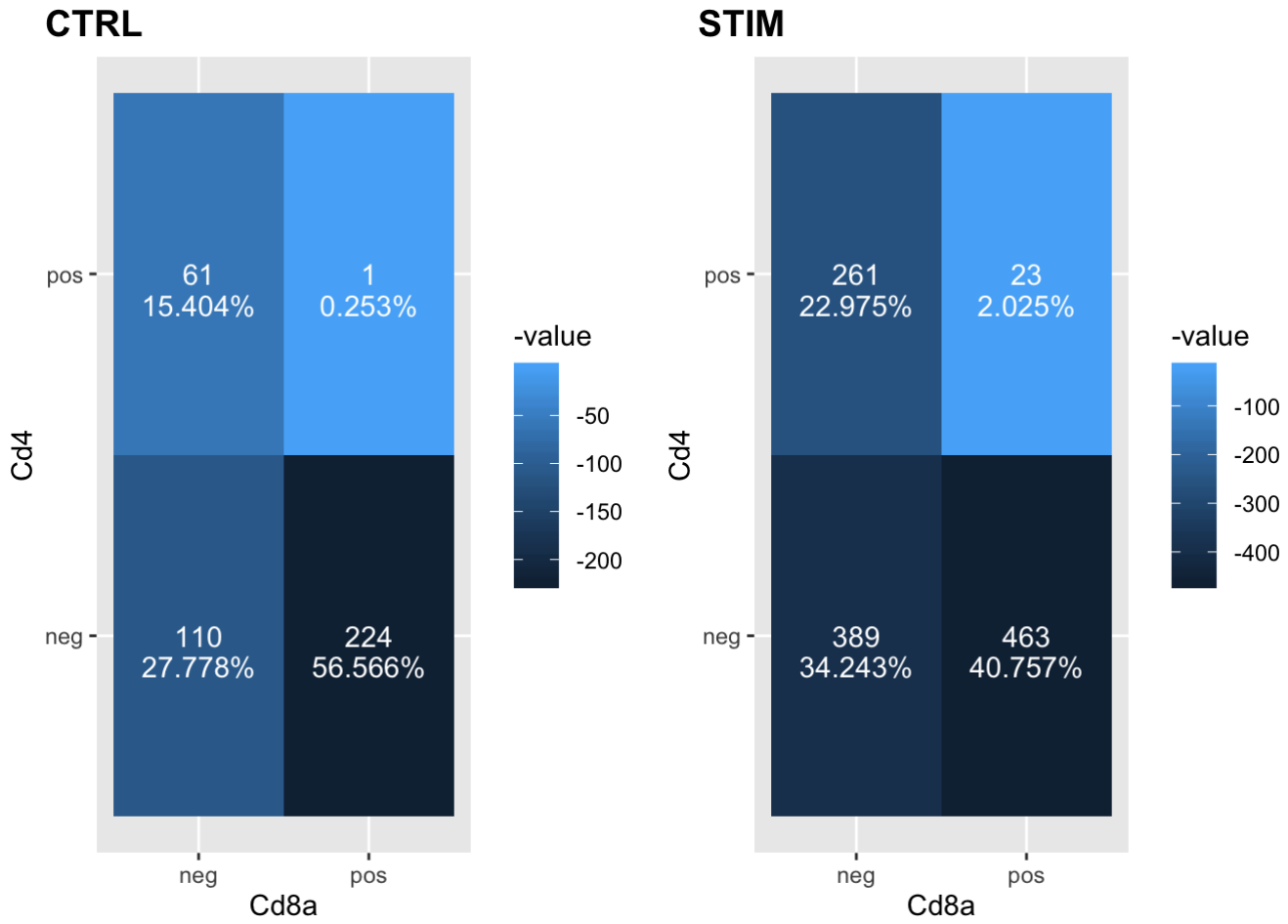
Build a cytometry object with CD8 and CD4 as axes

```
DefaultAssay(Tcell) <- "RNA"
p = Plot.FeatureScatter(Tcell, x = 'Cd8a', y = 'Cd4')
p
```

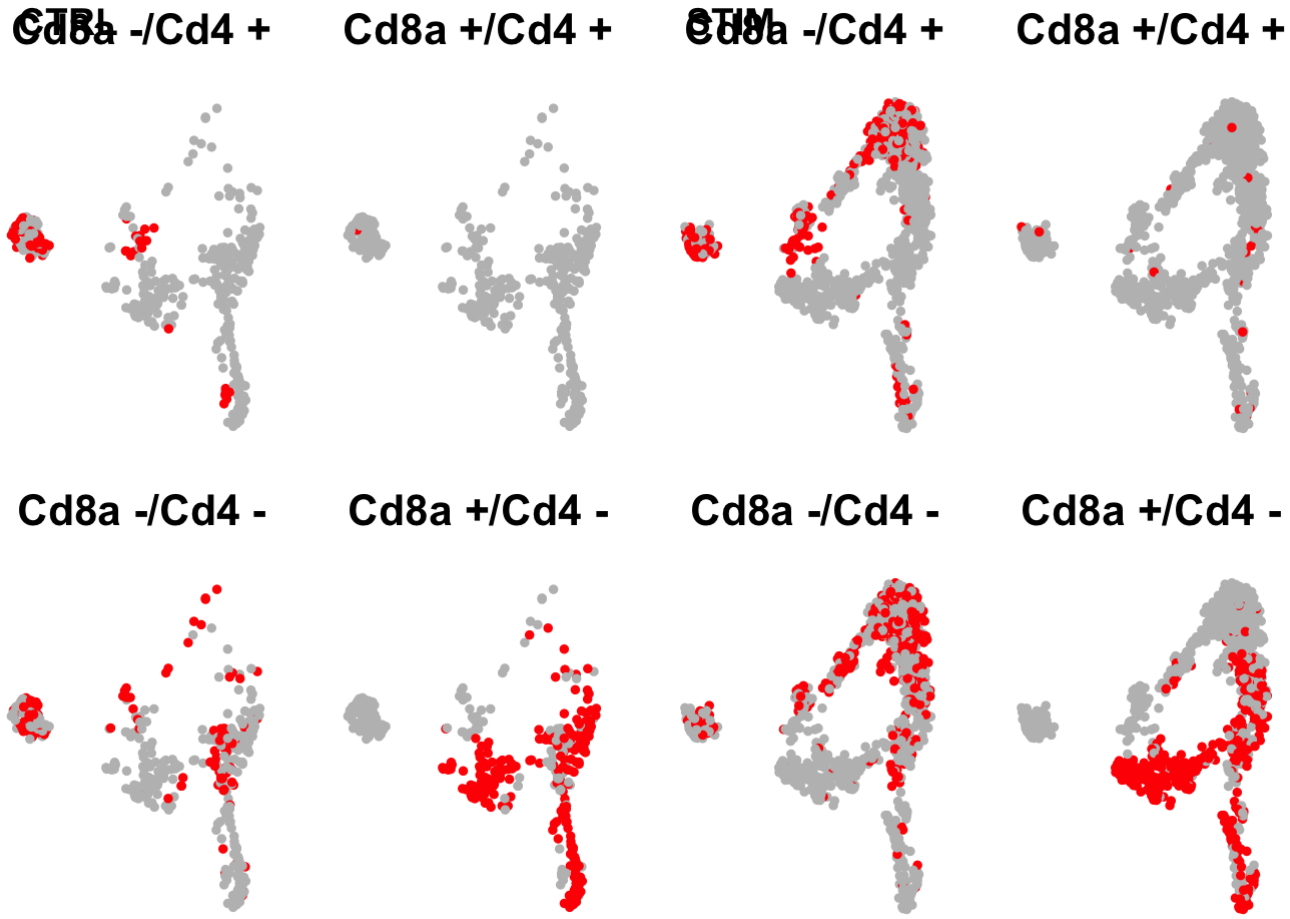



```
Tcell.cyto = Build.Cyto(Tcell,x = 'Cd8a', y = 'Cd4',x.thresh = 0,y.thresh = 0)
Plot.Cyto.Count(seurat.ob = Tcell.cyto,x = 'Cd8a', y = 'Cd4',split = T)
```

```
## Using x.label, y.label as id variables
## Using x.label, y.label as id variables
```




```
p = Plot.Cyto.Cluster(seurat.ob = Tcell.cyto,x = 'Cd8a', y = 'Cd4',split = T)
plot_grid(p$ctrl,p$stim,labels = c('CTRL','STIM'))
```



Extract the cd8, cd4 and double negative population

```
cd8 = Recluster.Quadrant(Tcell.cyto,n.quadrant = 2)
cd4 = Recluster.Quadrant(Tcell.cyto,n.quadrant = 4)
dn = Recluster.Quadrant(Tcell.cyto,n.quadrant = 3)
```

Perform reclustering of the 3 subpopulations

```
DefaultAssay(cd8) <- "integrated"
DefaultAssay(cd4) <- "integrated"
DefaultAssay(dn) <- "integrated"
cd8 = Build.Seurat.Cluster(cd8)
```

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

```
## Warning in PrepDR(object = object, features = features, verbose = verbose):
## The following 83 features requested have zero variance (running reduction
## without them): Cd209a, Rufy4, Klk1, Npy, D13Ertd608e, Cd209d, Slc40a1,
## Fcna, Lrg1, Retnlg, Paqr5, Lyve1, Fam198b, Gm5483, Ackr3, Ffar2, Akap6,
## Gm5861, Nyap2, Ccl24, Tmem204, Steap4, Siglec15, Stfa2l1, Plxna4, Ankrd2,
## Retnla, Cited1, Serpinb10, Fam57b, Rab38, Kazn, Dio2, Gsta3, Ephb2, Tenm4,
## Gm14161, Ptk6, Plpp3, Btbd17, Eci3, Sdpr, Clec2g, Tfcp2l1, Tcaf1, Emp2,
## Zfp462, Maats1, Shbg, Rptoros, Elavl2, Adgra3, Fbln2, Elavl3, Tmem246,
## Sbk2, Bmp2, Ak1, Steap1, Fzd1, Aqp1, F3, Dkk3, Gm9873, Btnl1, Tmem178,
## Rtn4r, Scgb1a1, Sema3a, Retn, Itgad, Hpgd, Gm38335, 6330403N20Rik, Pdgfra,
## Slc6a12, Calm4, Gng4, Fgf7, Scel, Tvp23a, Gulp1, Gstm2
```

```
## PC_ 1
## Positive:  Birc5, Stmn1, Cdk1, 2810417H13Rik, Cdca8, Top2a, Cdca3, Cks1b, Mki67, Nusa
p1
##     Ccna2, Rrm2, Tacc3, Asf1b, Spc24, Tuba1b, Smc2, Tyms, Ube2c, Tubb5
##     Fbxo5, Tk1, Hist1h1b, Tpx2, Ccnb2, Hmgb2, Hist1h2ap, Prc1, Hmmr, Cenpf
## Negative:  Nrg4, Angptl2, Ednrb, Pgf, Atcayos, Sept3, Cdr2l, Mcpt8, Gmpr, Vps37b
##     Obsl1, Cd3e, Txnip, Ccl5, Cpt1c, Cd8a, Rln3, Hopx, Mgl2, Itk
##     5830411N06Rik, Gls2, Ccnd2, Flrt2, Npl, Serpinb8, Wnt11, Mycl, Cd274, Hcst
## PC_ 2
## Positive:  S100a4, Lgals3, Capg, Pdcd1, Lag3, Bcl2a1d, S100a10, Bhlhe40, Bcl2a1b, Cxc
r6
##     Lgals1, Cd200r3, Tnfrsf9, Anxa2, Rora, Cd8b1, Cd3g, AA467197, Ppp1r1a, Ifitm2
##     Klrk1, Dcn, S100a11, Lsp1, Cd4, Cd82, Ube2l6, Hilpda, Ctla4, Ptprf
## Negative:  Satb1, 5830411N06Rik, Ppp1r14b, Lef1, Clec9a, Txk, Gimap6, Cd207, Xist, Gr
amd3
##     Ccr7, Socs3, Dkc1, Samd3, Ocstamp, Bcl2, Plaur, Bbs9, Tcf7, Gimap7
##     Nhp2, Itm2a, Ffar4, Rpl29, Ldlrad4, Cd28, Vps37b, Xcl1, Nme1, Abi3bp
## PC_ 3
## Positive:  Marcks, Bex6, Il15, Fndc7, C3ar1, Ly6i, Cd83, Ifi205, Mafb, Fcgr1
##     Lefty1, Pla2g7, Nupr1, Dpep2, B430306N03Rik, Klra2, Lyz1, Gm5150, Lrp1, C5ar1
##     Dnah2, Serpinb8, Clec4d, Ndufa4l2, Gpr141, Fcgr2b, Dppa3, Rgl1, Pi16, Col6a1
## Negative:  Edn1, Prr15, Fpr1, Sox9, Matn4, Col5a1, Gp6, Osr1, Armc4, Prss46
##     Slc7a2, Tnfsf15, Kdr, Thbd, Grem1, Serpinh1, Cd59a, Nrg4, Adgrg6, Ltf
##     Wnt11, Slco2b1, Tmem150c, Aldh1a2, Etnk2, Rerg, Dll4, H2-M2, Asprv1, Edil3
## PC_ 4
## Positive:  Nrg1, Fabp7, Edil3, Etnk2, Rerg, H2-M2, Asprv1, Aldh1a2, Dll4, Ltf
##     Tmem150c, Adgrg6, Wnt11, Adcy6, Ppp1r14a, Adh4, Fam178b, Agap1, Fabp4, Pnp2
##     Rhou, Stxbp6, Uchl1, Hepacam2, Rnf152, Flt1, Cd68, Clec4d, Clec9a, Cd180
## Negative:  Thbd, Flt3, Scin, Il6, Kdr, Laptm4b, Mgl2, Rogdi, Batf3, Cacnb3
##     Slc27a3, Cx3cl1, Pdlim4, Ccl22, Tspan33, Gmpr, Fscn1, Edn1, Armc4, Matn4
##     Sox9, Prr15, Gp6, Fpr1, Prss46, Col5a1, Osr1, Nostrin, Slc7a2, Tbc1d4
## PC_ 5
## Positive:  Gm26812, Rab15, Gins2, Dscc1, E2f1, Mcm7, Mcm3, Clec1a, Nme1, Fam71a
##     Nxpe5, Cdca7, Pgf, Ephx2, Ranbp1, Rln3, Cpt1c, Wnt11, Angptl2, Srm
##     Siva1, Wfdc18, Cisd1, Hells, Ltc4s, Prim1, Erh, Npm1, Syce2, Ung
## Negative:  Cenpf, Rhou, Grem1, Hmmr, Cdc20, Cdkn3, Apoc1, Stxbp6, Ube2c, Kdr
##     Ccnb1, Slc7a2, Sox9, Osr1, Armc4, Matn4, Fpr1, Prr15, Col5a1, Gp6
##     Prss46, Aspm, Tshz2, Thbd, Cenpe, Fam64a, Kif2c, Fndc7, Gbx1, Nfib
```

```
## 14:38:13 Read 687 rows and found 20 numeric columns
```

```
## 14:38:13 Using Annoy for neighbor search, n_neighbors = 30
```

```
## 14:38:13 Building Annoy index with metric = cosine, n_trees = 50
```

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

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

```
## **************************************************|
## 14:38:13 Writing NN index file to temp file /var/folders/pw/6v34dsx97wjdql3g4qhfjpj80
000gn/T//RtmpdrjIGQ/file518f5b474ec6
## 14:38:13 Searching Annoy index using 1 thread, search_k = 3000
## 14:38:13 Annoy recall = 100%
## 14:38:13 Commencing smooth kNN distance calibration using 1 thread
## 14:38:13 Initializing from normalized Laplacian + noise
## 14:38:13 Commencing optimization for 500 epochs, with 26506 positive edges
## 14:38:14 Optimization finished
## Computing nearest neighbor graph
## Computing SNN
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 687
## Number of edges: 22979
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8020
## Number of communities: 6
## Elapsed time: 0 seconds
```

```
cd4 = Build.Seurat.Cluster(cd4)
```

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

```
## Warning in PrepDR(object = object, features = features, verbose = verbose):
## The following 91 features requested have zero variance (running reduction
## without them): Rufy4, Dmkn, Lcn2, Klk1, Ifnb1, D13Ertd608e, Sh3bgr, Cd300c,
## Slc40a1, Lrg1, Retnlg, Paqr5, Lyve1, Prss50, Ackr3, Akap6, Speg, Nyap2,
## Tmsb15a, Sh2d1b2, Tmem204, Steap4, Siglec15, Prss46, Stfa2l1, Cndp1,
## Plxna4, Ankrd2, Retnla, Serpinb10, Tspan7, Fam57b, Rab38, Kazn, Cxcr2,
## Dio2, Gsta3, Tenm4, Serpinb2, Gm14161, Col16a1, Plpp3, Il12a, Eci3, Sdpr,
## Clec2g, Gal, Tcaf1, Emp2, Zfp462, Maats1, Shbg, Rptoros, Hao1, Osr1,
## Elavl2, Adgra3, Fbln2, Gp6, Sema6a, Cd177, Elavl3, Tmem246, Plcb1, Col5a1,
## Sbk2, Ak1, Steap1, Aqp1, Scx, Ppbp, Matn4, Tmem178, Rtn4r, Khdc1a, Gm38335,
## Six1, Cpne9, 6330403N20Rik, Pdgfra, Slc6a12, Olfr658, Pcdh7, Calm4, Gng4,
## Armc4, Pdgfrb, C9, Gulp1, Phactr3, Gstm2
```

```
## PC_ 1
## Positive:  Angpt2, Hist1h1b, Mki67, Spc24, Nid1, Nusap1, Cdk1, Gm15056, Tyms, 2810417
H13Rik
##     Tk1, Birc5, Ccna2, Cdca8, Fam64a, Gm43291, H1fx, Spc25, Aurkb, Hist1h2ap
##     Hmmr, Cenpe, Hist1h2ab, Sgms2, Tnfrsf4, Hist1h2ae, Asf1b, Foxa3, Inhba, Neil3
## Negative:  Cd8a, Rab27b, Lipg, Cd160, Tuba8, Vps37b, Tnfaip6, Hk3, Dppa3, Gls2
##     Bcl2, Sapcd1, Satb1, Mcpt8, Slamf7, Gm4208, Fstl1, Bhlhe41, Nes, Plet1
##     Serpinb9b, Ly6c2, Pdzd4, Aif1l, Klrk1, B230344G16Rik, Dmrta1, Klrc2, Gm15518, Ami
ca1
## PC_ 2
## Positive:  Tnfrsf9, S100a4, Ly6a, Traf1, Tnfrsf18, Tnfrsf4, Fam46a, Itgav, Icos, Rgs1
6
##     Ikzf2, AW112010, Maf, Capg, Ctla4, S100a10, Dhrs9, Gm15056, S100a11, Gm42835
##     Itgb8, Glrx, Foxp3, Cd74, Tnfaip6, Pdcd1, Il2ra, Tmem123, Hopx, Mcpt8
## Negative:  Cd36, Ncr1, Klra1, Akr1b8, Adm, Klra3, Hk3, Xcl1, H1fx, Pmp22
##     2810417H13Rik, Rab27b, Cd160, Gm43291, Cdc20, Xlr, Fam64a, Tuba8, Cenpm, Stmn1
##     Hist1h1b, Drc1, Ahnak2, Npm1, Gm4208, Spc24, Ptma, Meikin, Cdk1, Dppa3
## PC_ 3
## Positive:  Tns3, Cbfa2t3, Rogdi, Apod, Cyp7b1, Arhgef40, Adrbk2, Sh2d1b1, Apol7c, Ccl
22
##     Ccl17, Tspan33, Il4i1, Lima1, Pkib, Emid1, Gcsam, Cd63, Cacnb3, Fscn1
##     Sct, Fnbp1l, Ckap4, Srgap1, Tbc1d8, Mgp, Nudt17, Tmem150c, Gstt1, Uchl1
## Negative:  Stap2, 1700066B17Rik, Gm37422, Wfdc9, Asprv1, Rerg, Fgf7, Dll4, Slc7a2, Cc
l24
##     Fabp7, Plekhs1, Aldh1a2, Flt1, Gm10851, Fbxl13, Sphk1, Il6, Wnt11, Clec9a
##     Gm3336, Col27a1, Eps8, Ltc4s, Ppp1r14a, Met, Anpep, Ptcra, Adam23, Tpm2
## PC_ 4
## Positive:  Serpind1, Bex6, Prr15, Tvp23a, Fzd1, Ephb2, Fam198b, Adam23, Cldn1, Apof
##     Ppp1r14a, Sept3, Nes, Gm26812, Plppr4, Cfh, Tnni2, Adam11, Clec10a, Fam71a
##     Phkg1, Tnfsf15, Xlr, Aldh1a2, Aif1l, Zmynd15, 1700066B17Rik, Edil3, H2-Eb2, Batf3
## Negative:  Ltc4s, Anpep, Smagp, Klra17, 1700024P16Rik, Kdr, Cd59a, Myo1b, Ffar4, Klrb
1b
##     Tppp3, H2-DMb1, Cd300a, BC035044, Nccrp1, Vegfc, Spint1, Sectm1a, Trem3, Cd34
##     Hpgd, Dpys, Bmp2, Cd209d, Itgad, Cgnl1, Smim5, Scel, Sox9, Gfra2
## PC_ 5
## Positive:  Abi3bp, Il17a, Cd68, Cd163l1, C2, Tmem176a, Tmem176b, Crispld2, Gcnt2, Ser

pinb1a
##     5830411N06Rik, Aqp3, 5430421N21Rik, Gm12253, Tktl1, Plxdc2, Cxcr6, Blk, Bcl2a1b,
Dixdc1
##     Ppp1r14c, Bcl2a1d, Klrb1f, Tnp2, Ddx60, Tnfsf8, Ms4a8a, Nebl, Bdh2, Spink2
## Negative:  Derl3, Epha2, Gm15518, Cxcl1, Ccl5, Ecscr, Gzmb, Gm42835, Dppa3, Slco2b1
##     Serpinh1, Foxp3, Gm9873, Cd27, Cystm1, Il2ra, Gm6634, Arl5a, Ar, Itgb8
##     Hk3, Klrg1, Ikzf2, Tnfrsf9, Ddit4, Gimap3, Sdc4, Pglyrp1, Gpr137b-ps, Cd81
## 14:38:14 Read 322 rows and found 20 numeric columns
## 14:38:14 Using Annoy for neighbor search, n_neighbors = 30
## 14:38:14 Building Annoy index with metric = cosine, n_trees = 50
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
## 14:38:14 Writing NN index file to temp file /var/folders/pw/6v34dsx97wjdql3g4qhfjpj80
000gn/T//RtmpdrjIGQ/file518f2684d7fb
## 14:38:14 Searching Annoy index using 1 thread, search_k = 3000
## 14:38:14 Annoy recall = 100%
```

```
## 14:38:14 Commencing smooth kNN distance calibration using 1 thread
## 14:38:15 Initializing from normalized Laplacian + noise
## 14:38:15 Commencing optimization for 500 epochs, with 12606 positive edges
## 14:38:15 Optimization finished
## Computing nearest neighbor graph
## Computing SNN
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 322
## Number of edges: 11701
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7462
## Number of communities: 5
## Elapsed time: 0 seconds
```

```
dn = Build.Seurat.Cluster(dn)
```

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

```
## Warning in PrepDR(object = object, features = features, verbose = verbose):
## The following 35 features requested have zero variance (running reduction
## without them): Rufy4, Klk1, Npy, D13Ertd608e, Slc40a1, Lrg1, Gm5483, Akap6,
## Nyap2, Steap4, Siglec15, Stfa2l1, Ankrd2, Retnla, Cited1, Gm156, Fam57b,
## Shisa9, Dio2, Tenm4, Serpinb2, Gm14161, Plpp3, Sdpr, Maats1, Sbk2, Scx,
## Dll4, Gm38335, 6330403N20Rik, Olfr658, Calm4, C9, Gulp1, Gstm2
```

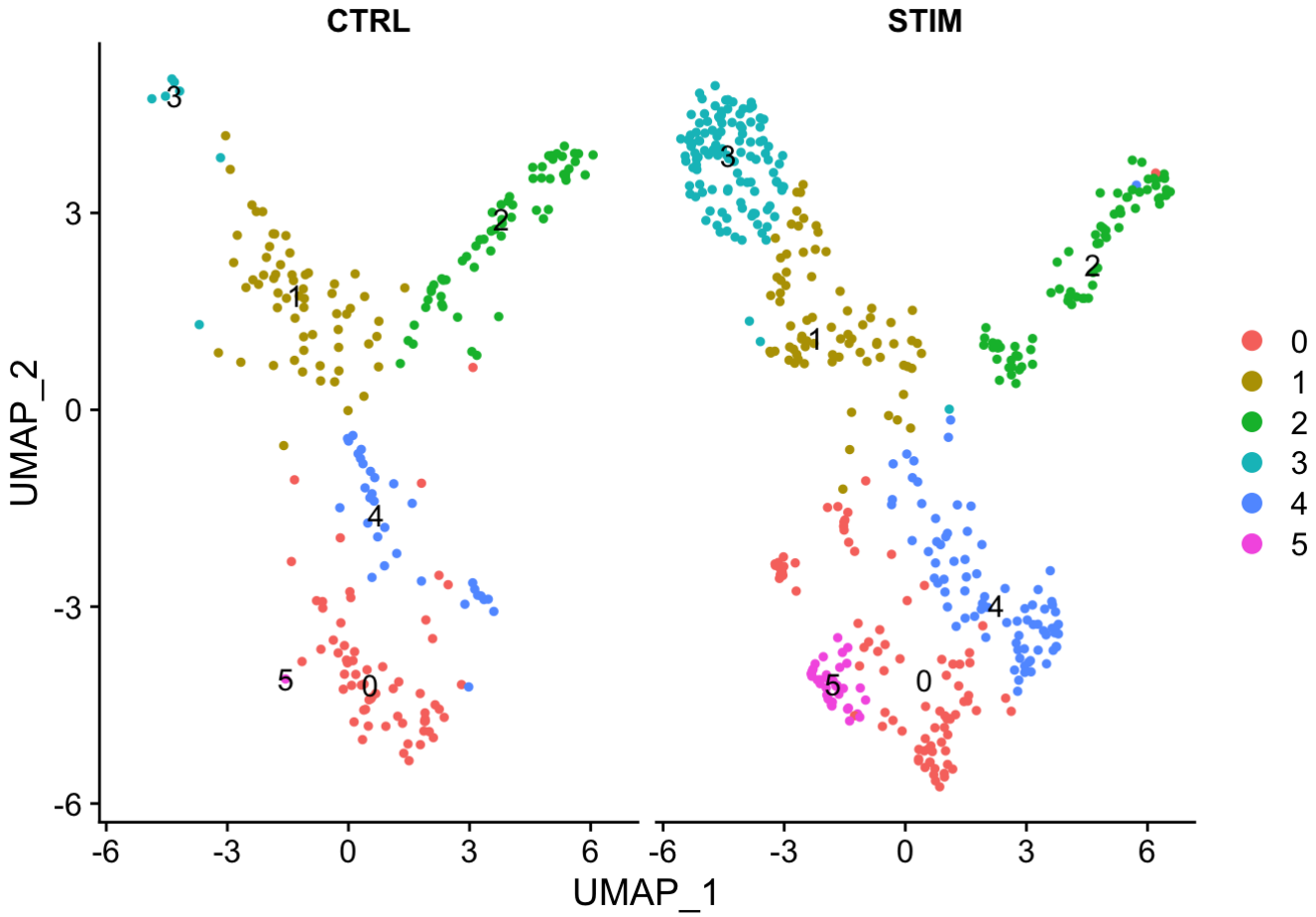
```
## PC_ 1
## Positive:  Tnfrsf4, Tnfrsf9, Nid1, Gm15056, Ikzf2, Tnfrsf18, Il2ra, Capg, Ctla4, Fam1
10a
##     Icos, Glrx, S100a10, Sdc4, Traf1, Hilpda, Maf, Itgav, S100a4, Foxp3
##     Arl5a, Klrg1, Slc16a3, Tmem123, Odc1, Areg, Fam46a, Rora, Ly6a, Itgb8
## Negative:  Cd8a, Satb1, Strip2, Tuba8, Ffar4, Vps37b, Bcl2, Adm, Gm13889, Wdr95
##     Gm13594, Ndrg4, Gramd3, Gm12589, Ovol2, Meikin, Lef1, Fbxl13, Slamf7, Il6
##     Ctsw, Cdc42ep2, Trim30c, Amica1, Cgnl1, Cd160, Zfp462, Nhp2, Slc6a12, Atp8b5
## PC_ 2
## Positive:  Fkbp11, Lima1, Lmo7, Rai14, Pde10a, Bcat1, Dclk1, Serpinf1, Lyz1, Adamtsl5
##     Oaf, Apoc4, Apod, Fn1, Col6a3, Tmeff1, Vcam1, Gcsam, Wfdc18, Fam71a
##     Ppbp, Plpp1, Clu, Fstl1, Col14a1, Oscar, Mgp, Serpinh1, Apold1, Sectm1a
## Negative:  Elavl2, Cndp1, Ifnb1, Car8, Tph1, Slc6a12, Slc6a4, Atp8b5, Tspan7, Mustn1
##     Zfp462, Pls3, Cgnl1, Pcdh7, 1700012B09Rik, Prr15, Fbxl13, Il6, Mcpt4, Rptoros
##     Fzd1, Flt1, Cd59a, Cpa3, Rhou, Spic, Gm13889, Eps8l2, Tnfrsf4, Tnfrsf18
## PC_ 3
## Positive:  Gm26737, Pard3b, Apobec2, Gcsam, Armcx6, Cpt1c, Gldc, Tmeff1, Dcstamp, Cd7
4
##     Adgrl2, Ly6a, Sdc4, Nfkbia, Gm15608, Gm8369, Cd3e, Tfcp2l1, Tnfrsf4, Itgb8
##     Lyz1, Dgat1, Icos, Apol9b, Vcam1, Efcab6, Ttc39c, Cd209a, Sdcbp2, Rora
## Negative:  Nusap1, Ccna2, 2810417H13Rik, Hist1h1b, Kif4, Ccnb1, Rrm2, Cdk1, Esco2, Bi
rc5
##     Smc2, Hist1h2ap, Cdca8, Cdca5, Asf1b, Tyms, Mki67, Spc24, Ncapg, Cdca3
##     Ckap2l, Dhfr, Stmn1, Neil3, Cenpf, Cenpe, Ccnb2, Hist1h2bj, Ube2c, Ttk
## PC_ 4
## Positive:  Gstt1, Stap2, Rerg, Plekhs1, Gm37422, Ppp1r14a, Eps8l2, Uchl1, Cx3cl1, Sph
k1
##     Fam198b, Cd300c, Prima1, Tvp23a, Eci3, Strip2, Gls2, Slc7a2, Sept3, Fabp4
##     Serpind1, Cd3d, Cd3g, Ctnnd2, Fndc7, Dcstamp, Tmsb10, Tnfrsf18, Ephx2, Mgl2
## Negative:  Il27, Tbc1d8, Olr1, Cxcl9, Wdr86, Cldn1, Mcpt4, Gpr137b-ps, Proser2, Cyp7b
1
##     Adrbk2, Sowahc, H2-Aa, Cpa3, Gcnt2, Irf8, H2-Eb1, Gpr137b, Gna14, Ifi30
##     Pkib, Mreg, Xcr1, Cd40, Marcks, Tubb6, Wfdc17, Tspan33, Siglecg, A530032D15Rik
## PC_ 5
## Positive:  Tnni2, Serpind1, Rassf8, Mgl2, Eci3, Cd300c, Prima1, Tvp23a, Sept3, Timd4
##     Flt3, Arhgef40, Fam198b, Flrt3, Spic, Il4i1, Il15, Batf3, Sct, Apof

##     Gfra2, Tnfsf15, Nudt17, Trim47, Ppp1r14a, Mmp25, Aldh1a2, Pard3b, Creb5, Dnase1l3
## Negative:  Fndc7, Sphk1, Gls2, Gm37422, Rerg, Plekhs1, Slc7a2, Uchl1, Cx3cl1, Stap2
##     Etnk2, Ephx2, 1700024P16Rik, Klra17, Eps8l2, Gna14, Strip2, Il6, Wdfy4, H2-DMb2
##     H2-Ab1, Fbxl13, Alpk2, Edn1, Ccdc88a, Nlrp3, Bank1, Fcmr, Bex6, Ascl2
## 14:38:15 Read 499 rows and found 20 numeric columns
## 14:38:15 Using Annoy for neighbor search, n_neighbors = 30
## 14:38:15 Building Annoy index with metric = cosine, n_trees = 50
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
## 14:38:15 Writing NN index file to temp file /var/folders/pw/6v34dsx97wjdql3g4qhfjpj80
000gn/T//RtmpdrjIGQ/file518f5b7fb37e
## 14:38:15 Searching Annoy index using 1 thread, search_k = 3000
## 14:38:16 Annoy recall = 100%
## 14:38:16 Commencing smooth kNN distance calibration using 1 thread
## 14:38:16 Initializing from normalized Laplacian + noise
## 14:38:16 Commencing optimization for 500 epochs, with 19596 positive edges
```

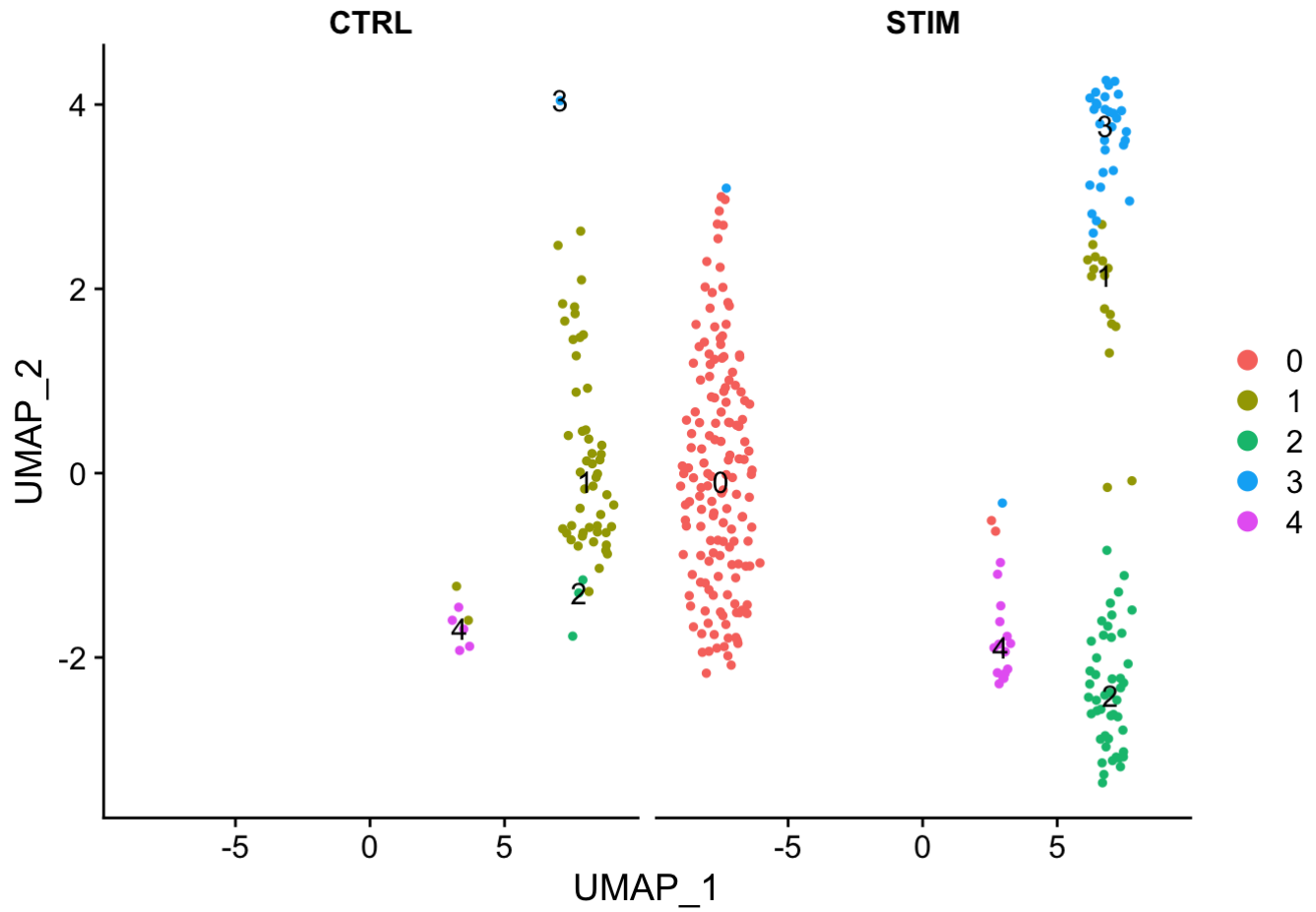
```
## 14:38:17 Optimization finished
## Computing nearest neighbor graph
## Computing SNN
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 499
## Number of edges: 17610
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7674
## Number of communities: 5
## Elapsed time: 0 seconds
```

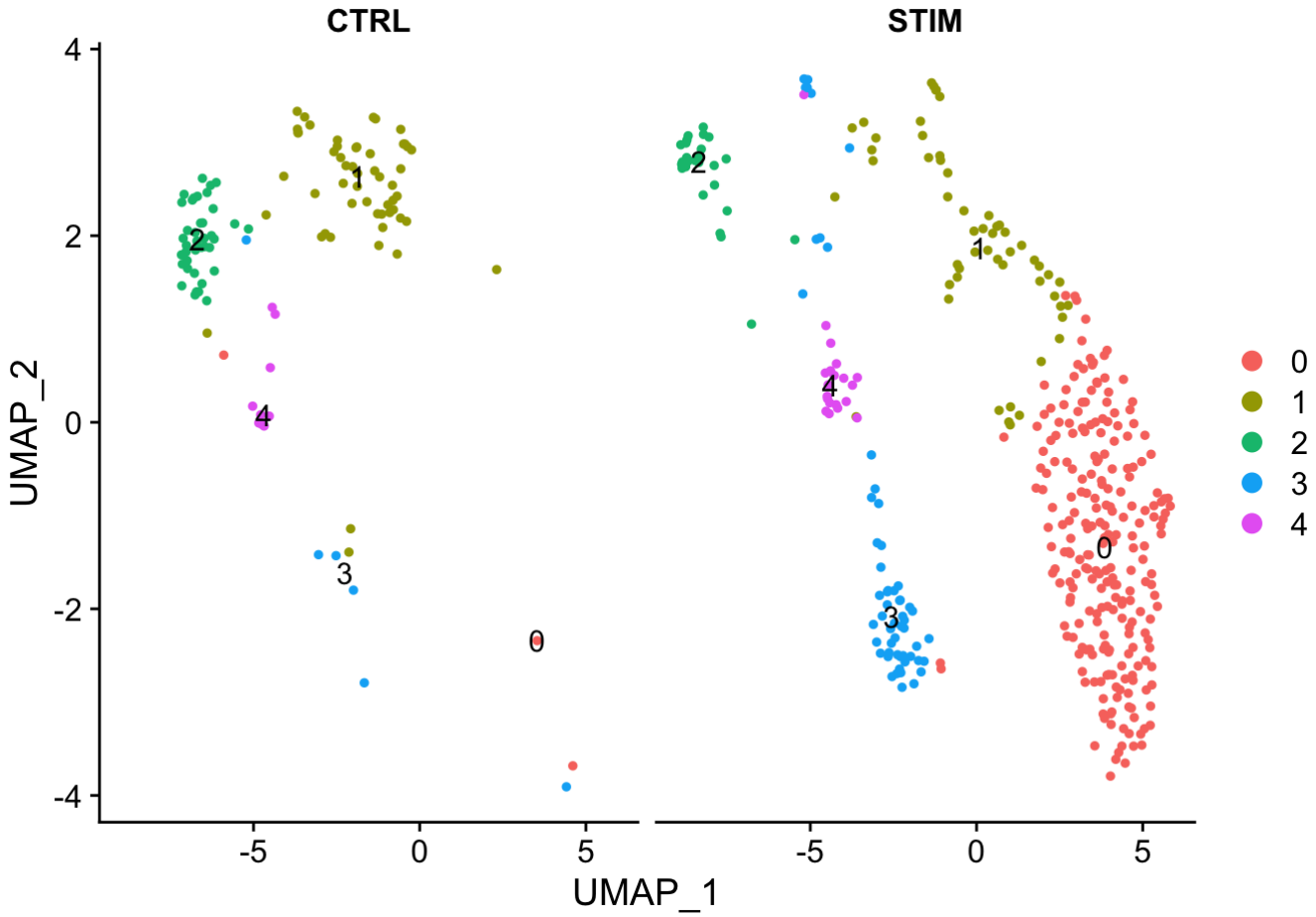
```
DimPlot(cd8, reduction = "umap", split.by = "stim",label = T)
```




```
DimPlot(cd4, reduction = "umap", split.by = "stim",label = T)
```




```
DimPlot(dn, reduction = "umap", split.by = "stim",label = T)
```



Store the ShinyApp output of desired sub population (CD8)

```
cd8.markers <- Build.ConserveMarkers.All(cd8)
cd8.markers.each = Find.Markers.Each(cd8)
cd8.diff = DE.Each.Cluster(cd8)
cd8.out = Shine.Out(ob = cd8, diff = cd8.diff, markers.each = cd8.markers.each, markers.
conserved = cd8.markers.flat)
```

```
saveRDS(object = cd8.out,file = './Shiny_diff/input/cd8_out_0820.rds')
```