cytometry_reclustering

Changhua Yu

8/16/2019

```
library(dplyr)
library(Seurat)
library(purrr)
library(cowplot)
library(parallel)
library(roxygen2)
library(reshape2)
library(tibble)
library(ggplot2)
source('../R/cell_cytometry.R')
source("../R/DE_analysis.R")
```

Read in the Count matrix

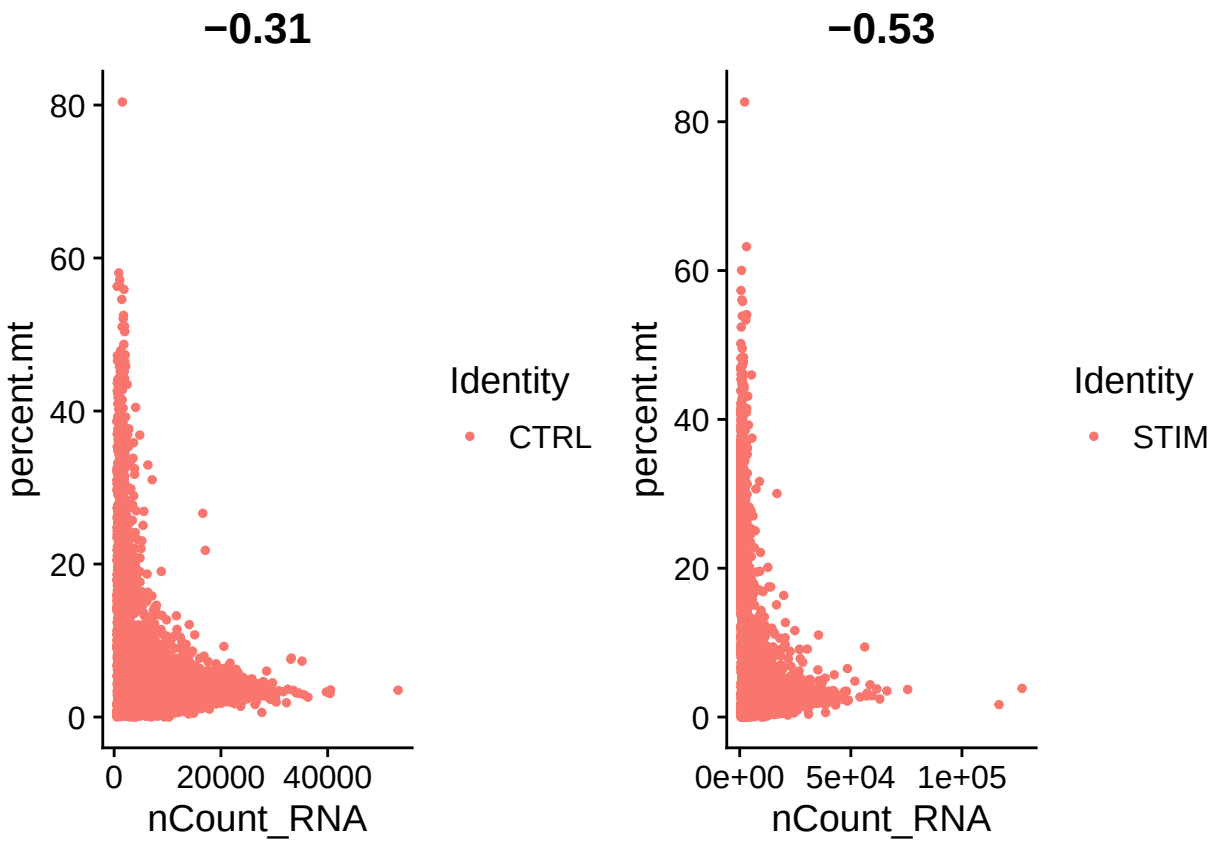
```
path1 = "../data/KO0531/"
path2 = "../data/WT0531/"
ko.mat = Read10X(data.dir = path1)
wt.mat = Read10X(data.dir = path2)

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)
```

Mitochondria Percentage calculation and assigning conditions to "stim" field

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

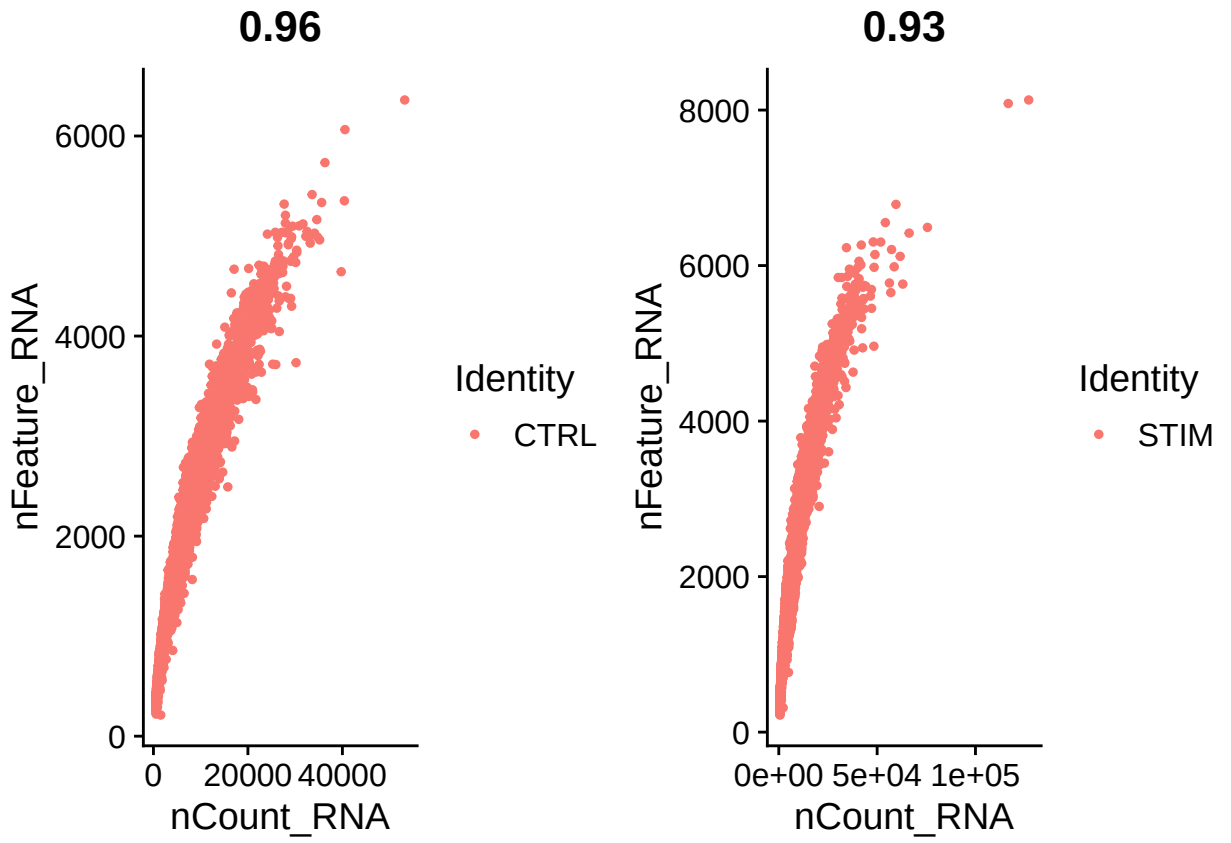
Plot out mitochondria percent distribution

```
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)
```

Normalize Data and find variable 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
```

Save data for future usage

```
save(ctrl, stim,immune.combined, file="../data/CD45POS.RData")
```

Proceed the anchored clustering

```
load("../data/CD45POS.RData")
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 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

## 17:16:09 Read 11659 rows and found 15 numeric columns

## 17:16:09 Using Annoy for neighbor search, n_neighbors = 30

## 17:16:09 Building Annoy index with metric = cosine, n_trees = 50

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

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

## **************************************************|
## 17:16:10 Writing NN index file to temp file /var/folders/pw/6v34dsx97wjdql3g4qhfjpj80000gn/T//Rtmpni
## 17:16:10 Searching Annoy index using 1 thread, search_k = 3000
## 17:16:13 Annoy recall = 100%
## 17:16:13 Commencing smooth kNN distance calibration using 1 thread
## 17:16:14 Initializing from normalized Laplacian + noise
## 17:16:14 Commencing optimization for 200 epochs, with 477176 positive edges
## 17:16:20 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)
```

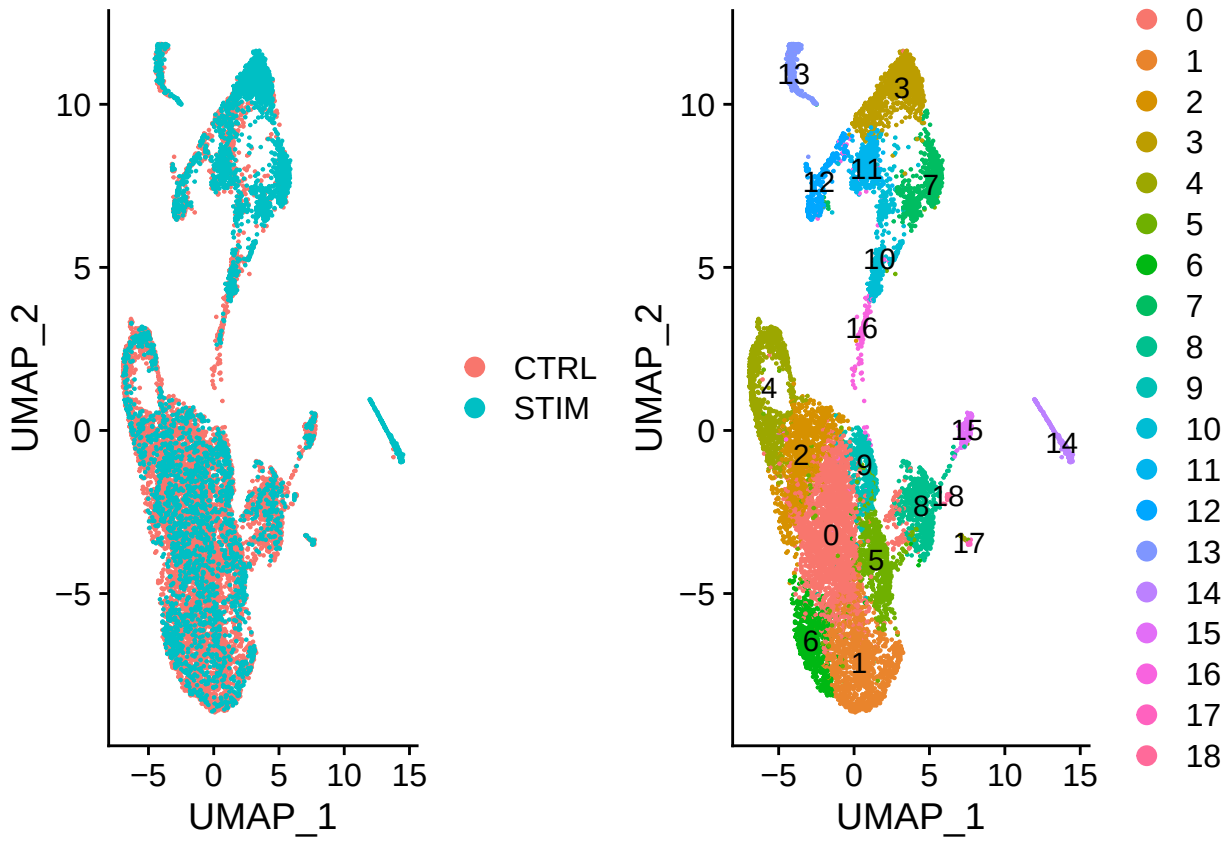
```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 11659
## Number of edges: 392061
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8841
## Number of communities: 19
## Elapsed time: 1 seconds
```

Visualizing UMAP clustering

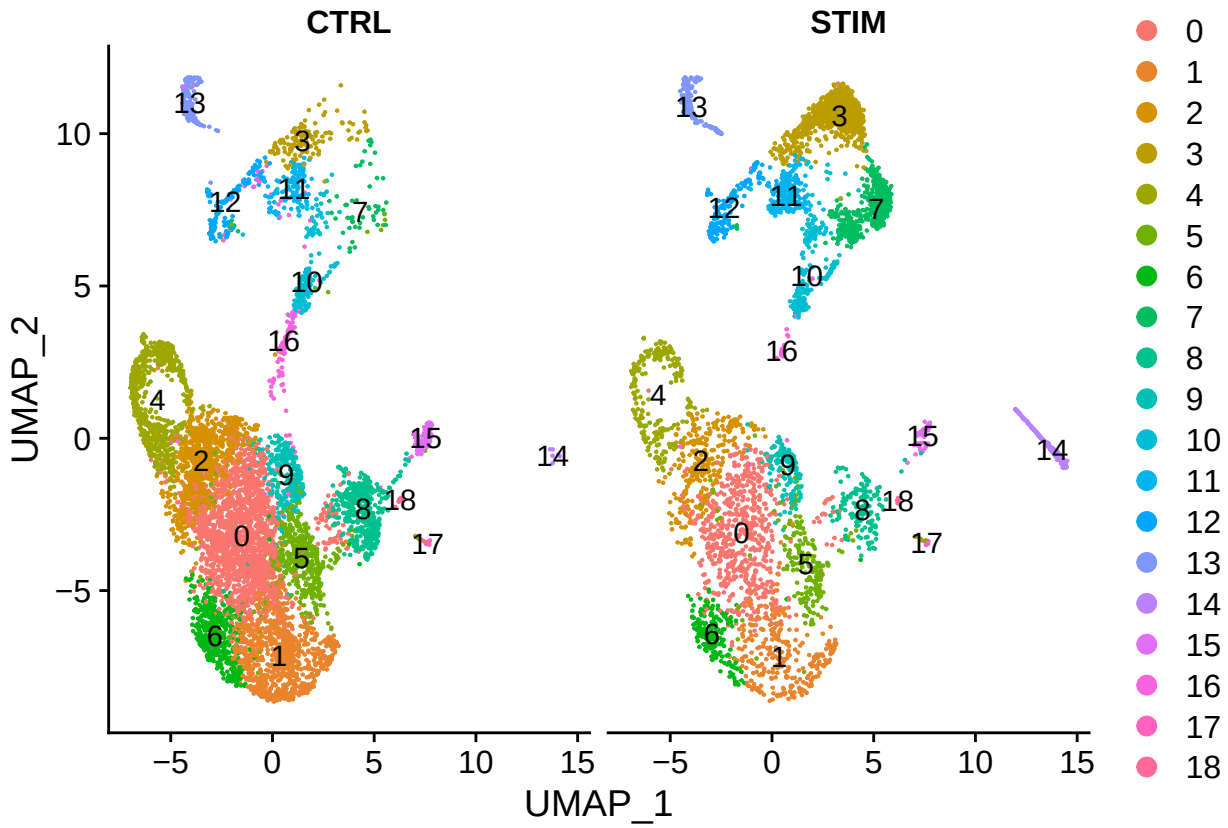
```
p1 <- DimPlot(immune.combined, reduction = "umap", group.by = "stim")
p2 <- DimPlot(immune.combined, reduction = "umap", label = TRUE)
```

```
## 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)
```



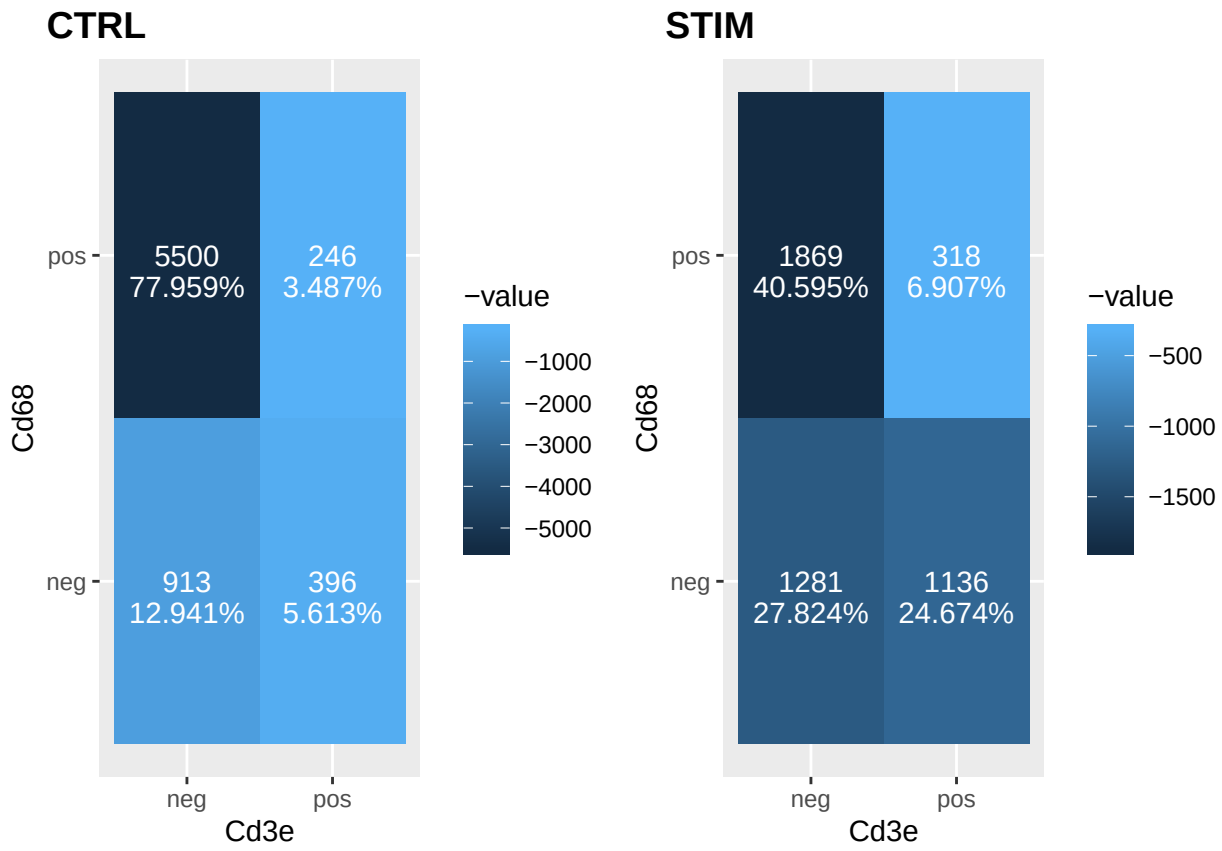

```
DimPlot(immune.combined, reduction = "umap", split.by = "stim",label = T)
```



Create a Cytometry object with the given markers and threshold on markers

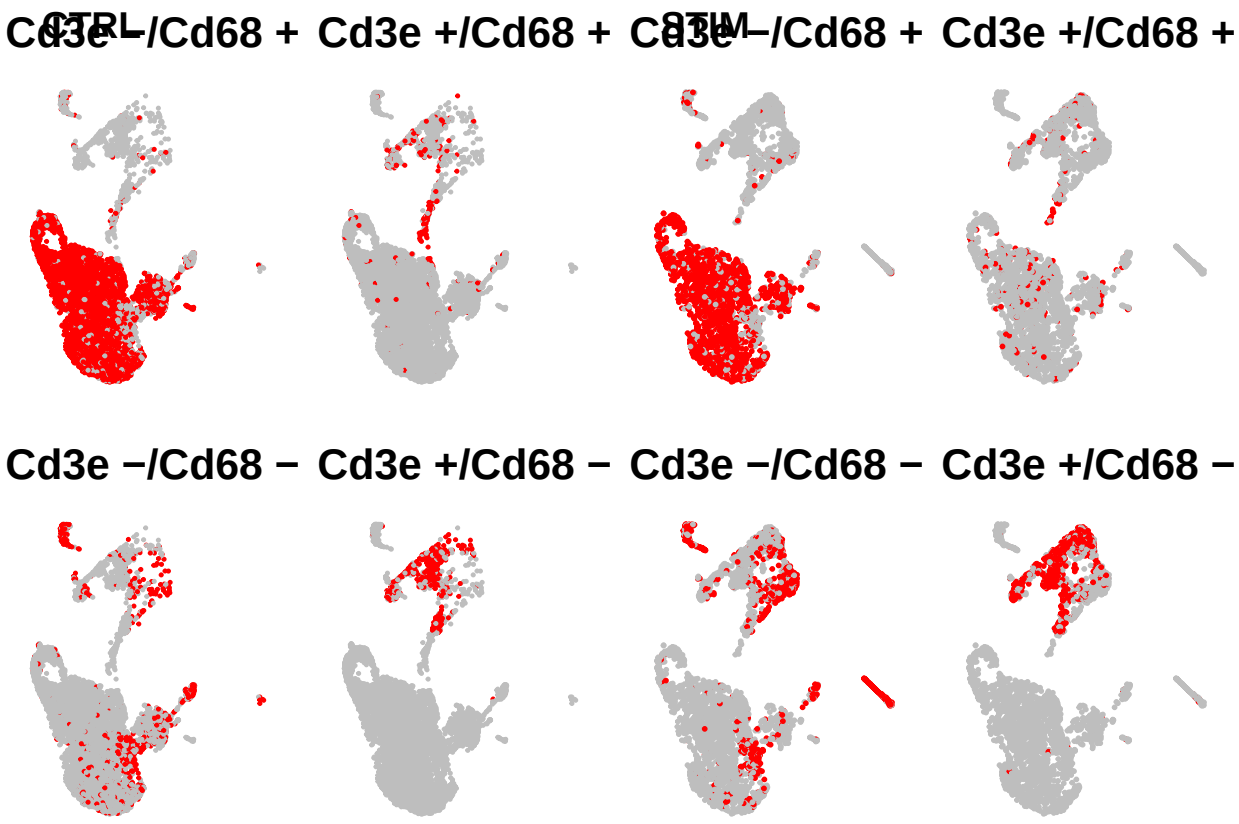
```
immune.combined.cyto.cd68 = Build.Cyto(immune.combined,"Cd3e","Cd68",x.thresh = 0.1,y.thresh = 0.1)
Plot.Cyto.Count(seurat.ob = immune.combined.cyto.cd68,
                x= "Cd3e",
                y = "Cd68",split = T)
```

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



Plot out the clusters mapping back to the UMAP clustering plot

```
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'))
```



Extracting out the cd3e positive T cells (the second qudrant)

```
Tcell = Recluster.Quadrant(immune.combined.cyto.cd68,n.quadrant = 2)
```

perform the reclustering on T Cell object

```
DefaultAssay(Tcell) <- "integrated"
Tcell <- RunPCA(Tcell, npcs = 30, verbose = FALSE)
# t-SNE and Clustering
Tcell <- RunUMAP(Tcell, reduction = "pca",dims = 1:15)
```

```
## 17:16:29 Read 1532 rows and found 15 numeric columns

## 17:16:29 Using Annoy for neighbor search, n_neighbors = 30

## 17:16:29 Building Annoy index with metric = cosine, n_trees = 50

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

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

```
## **************************************************|
## 17:16:29 Writing NN index file to temp file /var/folders/pw/6v34dsx97wjdql3g4qhfjpj80000gn/T//Rtmpni
## 17:16:29 Searching Annoy index using 1 thread, search_k = 3000
## 17:16:29 Annoy recall = 100%
## 17:16:29 Commencing smooth kNN distance calibration using 1 thread
## 17:16:30 Initializing from normalized Laplacian + noise
## 17:16:30 Commencing optimization for 500 epochs, with 64154 positive edges
## 17:16:32 Optimization finished
```

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

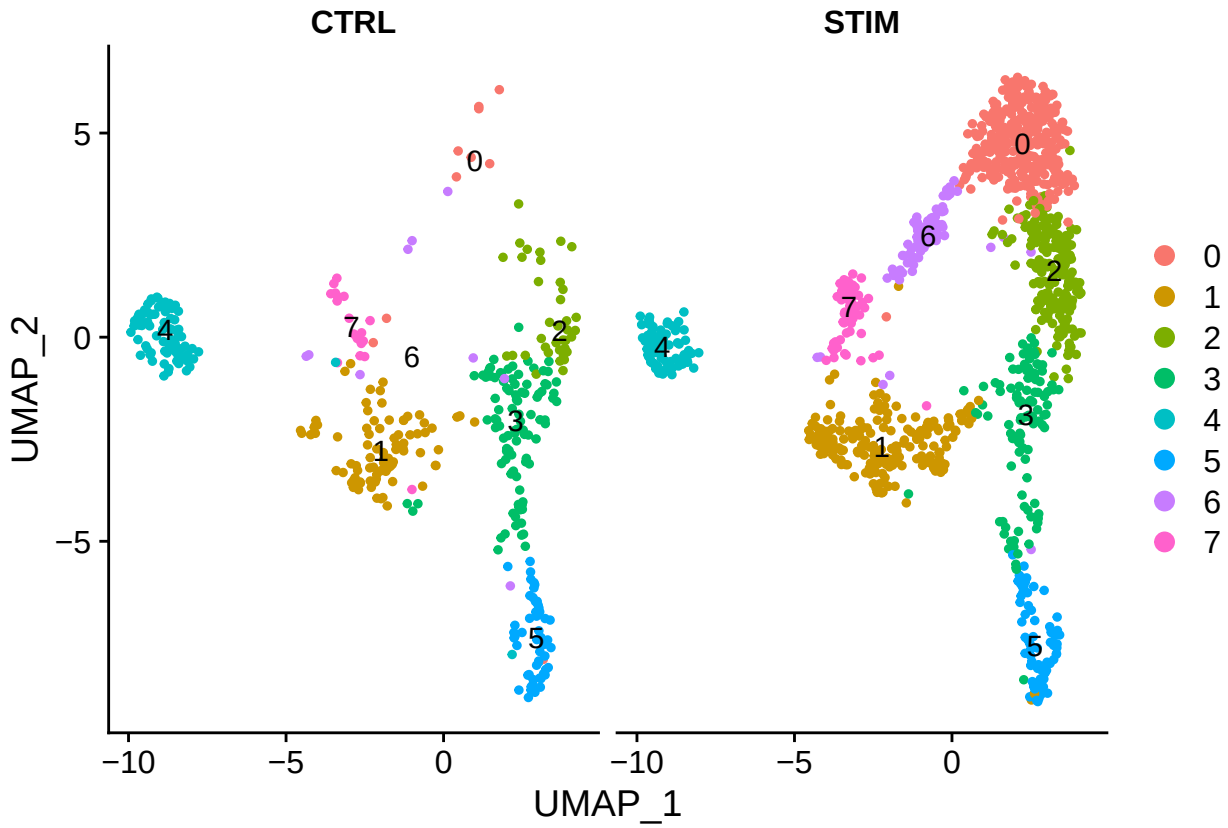
```
## Computing nearest neighbor graph
## Computing SNN
```

```
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
```

Inspect on the reclusted T cell populations

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


CTRL
STIM
UMAP_2
UMAP_1
5
0
−5
−10
−5
0
0
1
2
3
4
5
6
7