

Mafb-restricted local monocyte proliferation precedes lung interstitial macrophage differentiation

5-scRNAseq initiation for Exp2: the IMs refilling during post deletion

2022-03-09 09:27:16 +0100

Abstract

Resident tissue macrophages (RTM) are differentiated immune cells populating distinct niches and exhibiting important tissue-supportive functions. RTM maintenance is thought to rely on either monocyte engraftment and differentiation, or RTM self-renewal. Here, we developed an inducible mouse model of lung interstitial macrophage (IM) niche depletion and repopulation to investigate IM development *in vivo*. Using time-course single-cell RNA-sequencing analyses, bone marrow chimeras and gene targeting, we found that engrafted Ly6C+ classical monocytes could self-renew locally in a CSF1R-dependent manner before their differentiation into RTM. We further showed that the switch from monocyte proliferation towards IM subset specification was controlled by MafB, while c-Maf specifically regulated the identity of the CD206+ IM subset. Our data shed new light on the transcriptional regulation of IM development and provide evidence that, in the mononuclear phagocyte system, self-renewal is not merely restricted to myeloid progenitor cells and mature macrophages, but is also a tightly regulated capability of mature monocytes developing into RTM *in vivo*.

Contents

1 Description	3
2 Demultiplexing (de-hashtag)	3
2.1 Load packages and data	3
2.2 Setup Seurat object and add in the HTO data	4
2.3 Adding HTO data as an independent assay	4
2.4 Demultiplex cells based on HTO enrichment	4
2.5 Visualize demultiplexing results	4
3 QC of pooled sample	11
3.1 Sample: CD45Plus_NGS21-S229	11
4 Identify the cells from each samples (de-hashtagging)	13
4.1 Build metadata table	14
4.2 Data processing for CD45Plus_NGS21_S229	14
4.3 Linear dimension reduction	14
5 Identify cell types	16
5.1 SingleR analysis	16
5.2 Marker expression check	16
5.3 Combine marker expression with SingleR results	18
5.4 Remove contaminated cell types other than monocytes/macrophages	21
6 Phenotypic characterization of monocytes/macrophages	21
6.1 Data processing after filtering:	21
6.2 Clustering of cells	22
6.3 Population characterization	23

6.4	Cell cycle analysis	26
6.5	Apoptosis rate by mitochondrial genes	29
7	Session information	29
	References	31

1 Description

Lung interstitium macrophages (IMs) are non-alveolar resident tissue macrophages which contribute to the lung homeostasis. These cells were reported to be heterogeneous by our group and other teams, which contains two main distinct subpopulations: CD206+ IMs and CD206- IMs. However, the exact origin of IMs and the transcriptional programs that regulate IM differentiation remains unclear. In recent report, we analyzed the refilled IMs in the course of time after induced IM depletion with single-cell RNA sequencing (10X Genomics Chromium) and bulk RNA sequencing.

The lung IMs and monocytes from the mice at 12 hours (DT12h), 24 hours (DT24h) and 48 hours (DT48h) after diphtheria toxin (DT)-induced IM depletion were analyzed and compared using single-cell RNA sequencing. A subpopulation was found to be a transit differentiating cells from monocytes to IMs. Transcription factor activity analysis and trajectory showed cMAF and MAFb transcription factors played important roles in monocyte-IM differentiation.

Over design of experiment:

The lung IMs and monocytes were sorted from three groups of mice:

- Group “DT12h” contains 5 IM-DTR mice which were treated with 50 ng diphtheria toxin (DT) 12 hours before sacrifice.
- Group “DT24h” contains 5 IM-DTR mice which were treated with 50 ng DT 24 hours before sacrifice.
- Group “DT48h” contains 5 IM-DTR mice which were treated with 50 ng DT 48 hours before sacrifice.

Cells from each group were barcoded with different Biolegend anti-mouse Hastags before being pooled for library construction. Processed data then were analyzed with Bioconductor and Seurat package. The cells from different time points were demultiplexed with the barcode detected in each cell.

The pipeline build Seurat object from matrix, celltyping and contaminated cell removal were identical to the previous analyses (Analysis 2, 3 and 4) using Seurat package¹ and SingleR package.² Finally, the filtered cells were saved into a Seurat object for further analysis.

2 Demultiplexing (de-hashtag)

2.1 Load packages and data

```
library(Seurat)  
  
# change cell names to "-1" mode.  
colnames(immune.htos) <- paste0(colnames(immune.htos), "-1")  
  
# remove the unmapped as it's not a barcode. It's the last row.  
immune.htos <- immune.htos[-nrow(immune.htos), ]  
  
# remove the "HT5_CTL-CTTTGTCTTGAG" as it's not added to the sample.  
immune.htos <- immune.htos[rownames(immune.htos) != "HT5_CTL-  
CTTTGTCTTGAG", ]  
  
# Select cell barcodes detected by both RNA and HTO  
# In the example datasets we have already filtered the cells for you, but  
# perform this step for clarity.  
joint.bcs <- intersect(colnames(immune.umis), colnames(immune.htos))  
  
# Subset RNA and HTO counts by joint cell barcodes  
immune.umis <- immune.umis[, joint.bcs]  
immune.htos <- as.matrix(immune.htos[, joint.bcs])
```

```
# Confirm that the HTO have the correct names
rownames(immune.htos)
```

```
## [1] "HT6_12hDT-TATGCTGCCACGGTA" "HT7_24hDT-GAGTCTGCCAGTATC"
## [3] "HT8_48hDT-TATAGAACGCCAGGC"
```

2.2 Setup Seurat object and add in the HTO data

```
# Setup Seurat object
immune.hashtag <- CreateSeuratObject(counts = immune.umis)

# Normalize RNA data with log normalization
immune.hashtag <- NormalizeData(immune.hashtag)
# Find and scale variable features
immune.hashtag <- FindVariableFeatures(immune.hashtag, selection.method =
  'mean.var.plot')
immune.hashtag <- ScaleData(immune.hashtag, features = VariableFeatures(
  immune.hashtag))
```

2.3 Adding HTO data as an independent assay

```
# Add HTO data as a new assay independent from RNA
immune.hashtag[['HTO']] <- CreateAssayObject(counts = immune.htos)
# Normalize HTO data, here we use centered log-ratio (CLR) transformation
immune.hashtag <- NormalizeData(immune.hashtag, assay = 'HTO',
  normalization.method = 'CLR')
```

2.4 Demultiplex cells based on HTO enrichment

Here we use the Seurat function HTODemux() to assign single cells back to their sample origins.

```
# Here we are using the default settings
immune.hashtag <- HTODemux(immune.hashtag, assay = "HTO", positive.
  quantile = 0.99)
```

2.5 Visualize demultiplexing results

Output from running HTODemux() is saved in the object metadata. We can visualize how many cells are classified as singlets, doublets and negative/ambiguous cells.

```
# Global classification results
table(immune.hashtag$HTO_classification.global)
```

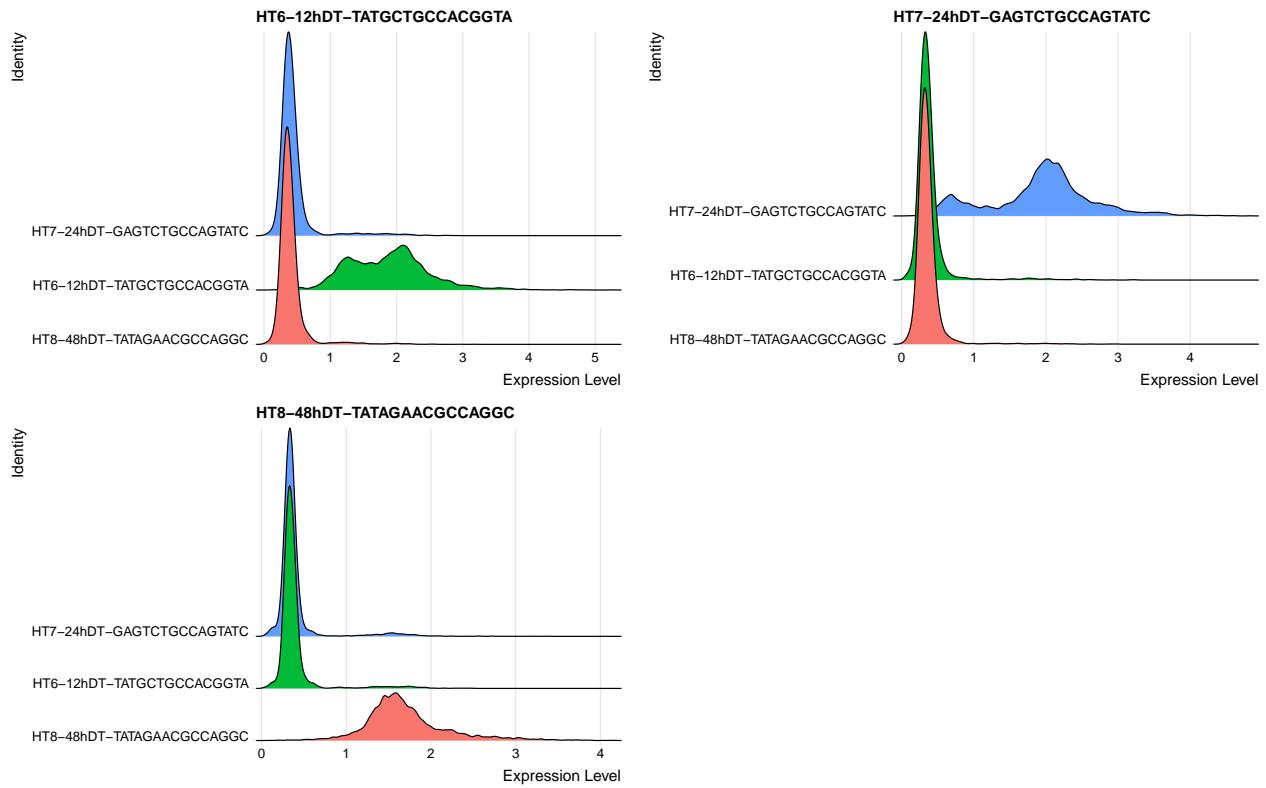
```
##
## Doublet Negative Singlet
##      963      679    12528
```

Visualize enrichment for selected HTOs with ridge plots

```

# Group cells based on the max HTO signal
Idents(immune.hashtag) <- 'HTO_maxID'
RidgePlot(immune.hashtag, assay = 'HTO', features = rownames(immune.
hashtag [['HTO']]), ncol = 2)

```

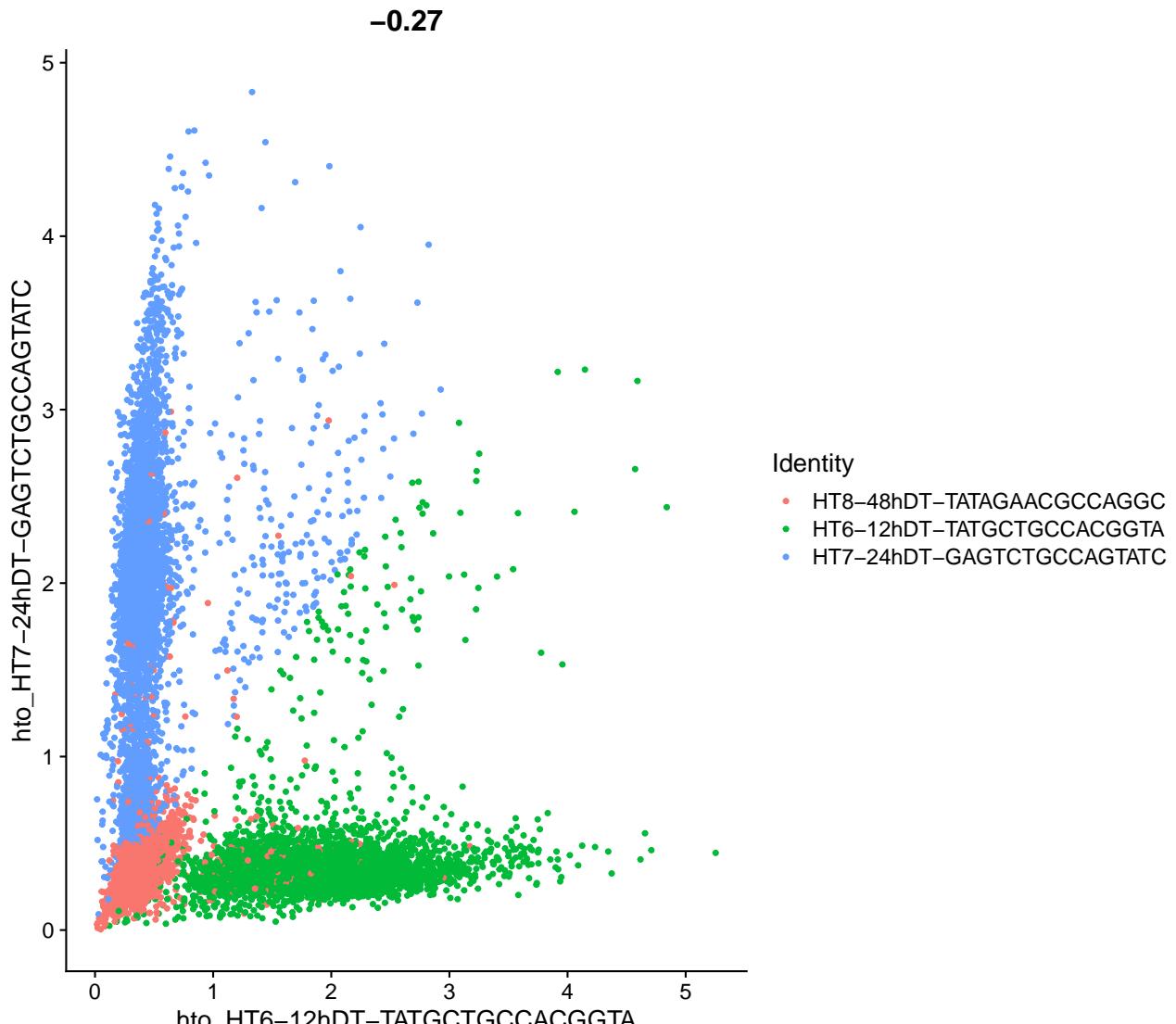


Visualize pairs of HTO signals to confirm mutual exclusivity in singlets

```

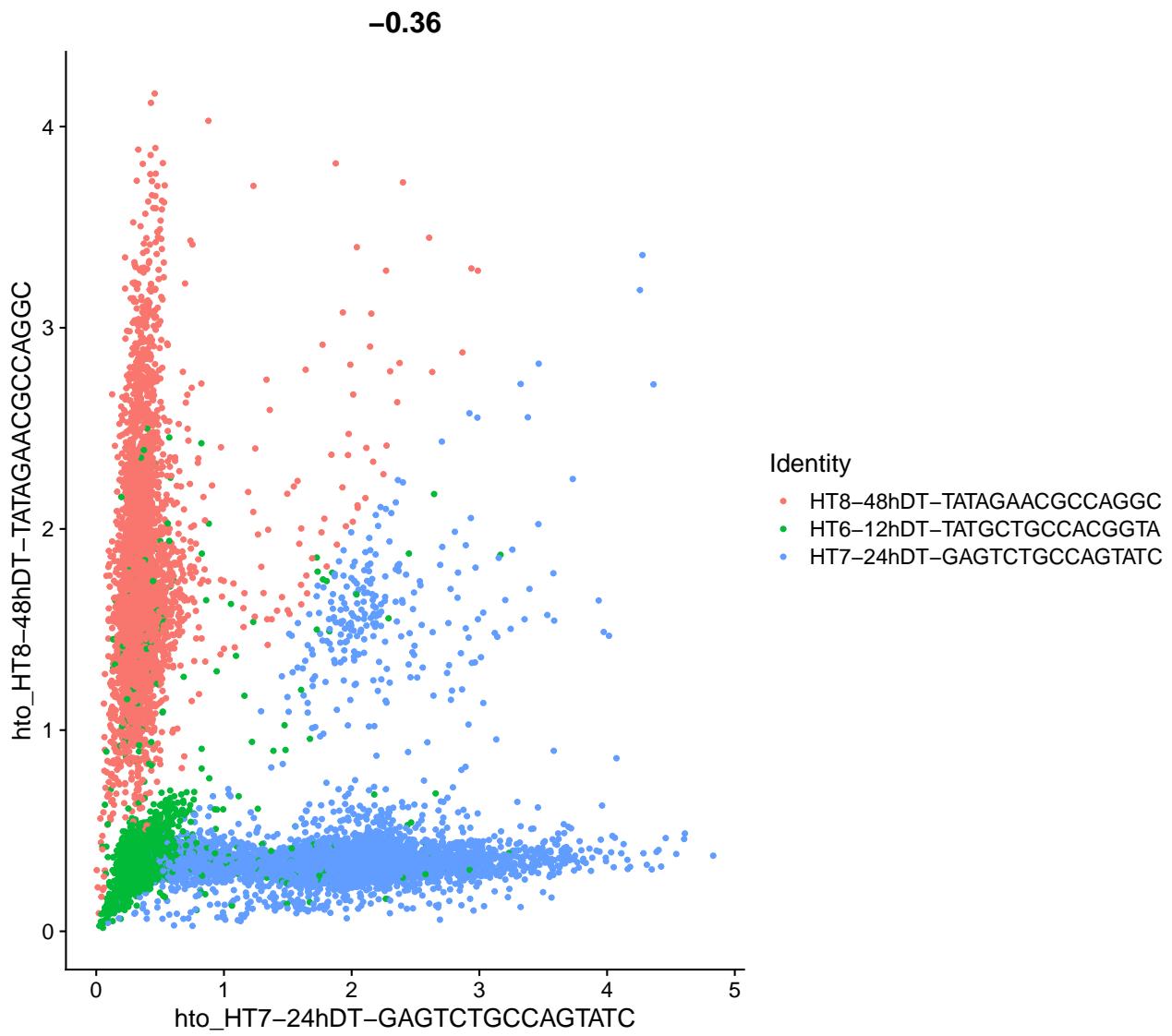
FeatureScatter(immune.hashtag, feature1 = 'HT6-12hDT-TATGCTGCCACGGTA',
               feature2 = 'HT7-24hDT-GAGTCTGCCAGTATC')

```



```
FeatureScatter(immune.hashtag, feature1 = 'HT7-24hDT-GAGTCTGCCAGTATC',
               feature2 = 'HT8-48hDT-TATAGAACGCCAGGC')
```

1

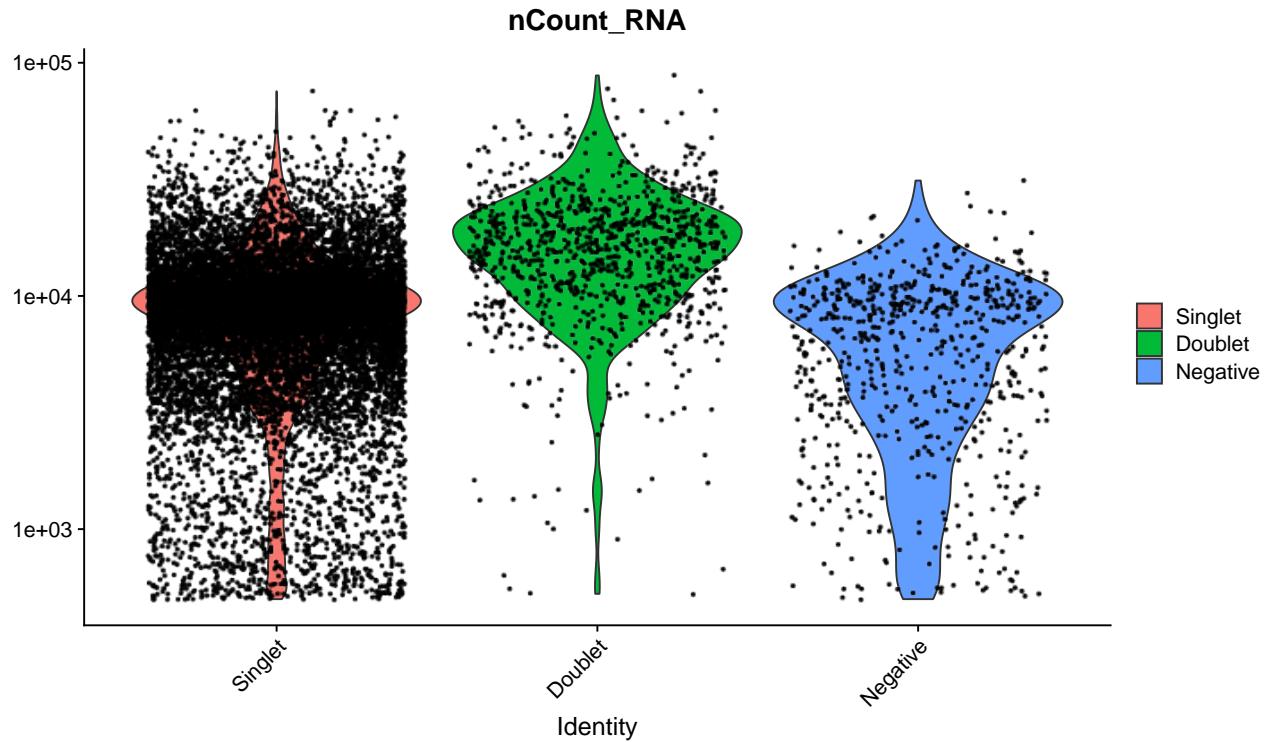


Compare number of UMIs for singlets, doublets and negative cells

```

1 Idents(immune.hashtag) <- 'HT0_classification.global'
2 VlnPlot(immune.hashtag, features = 'nCount_RNA', pt.size = 0.1, log = TRUE
  )

```

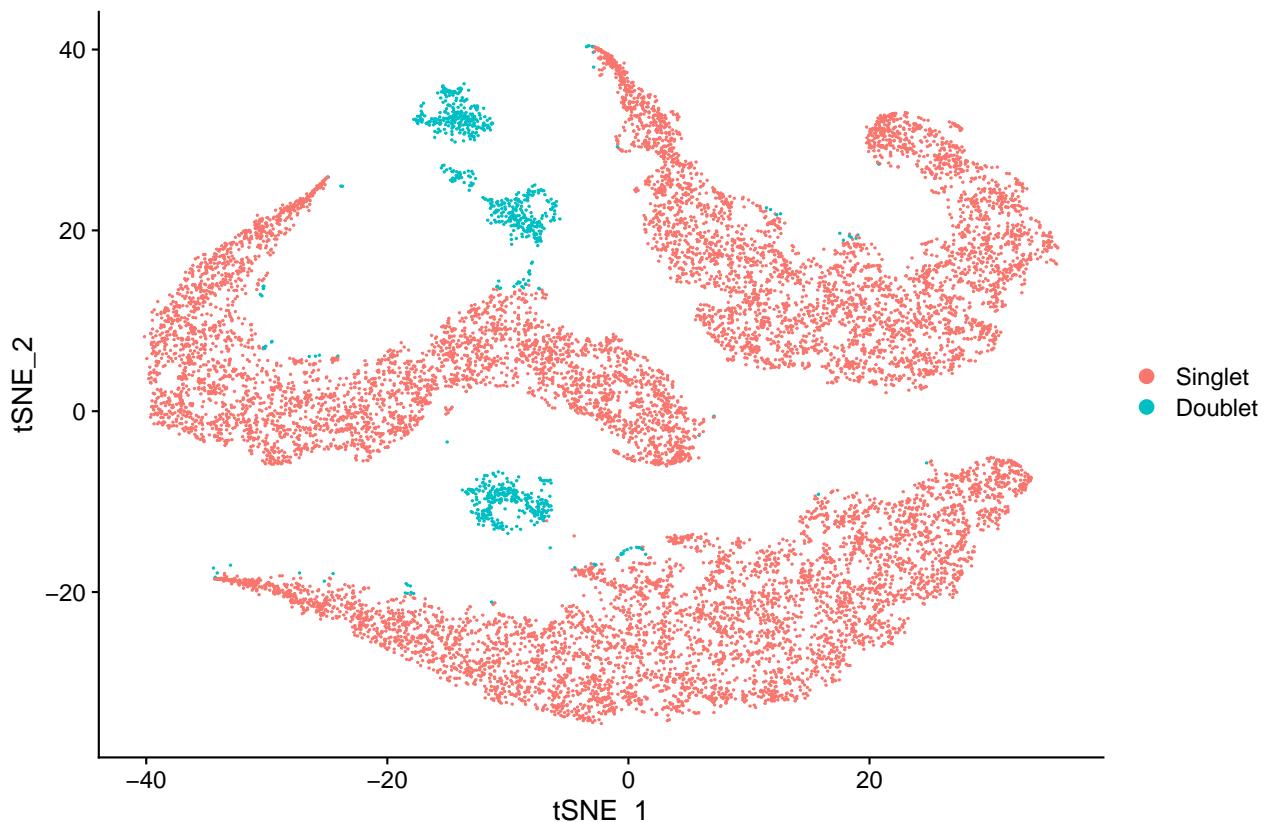


Generate a two dimensional tSNE embedding for HTOs. Here we are grouping cells by singlets and doublets for simplicity.

```

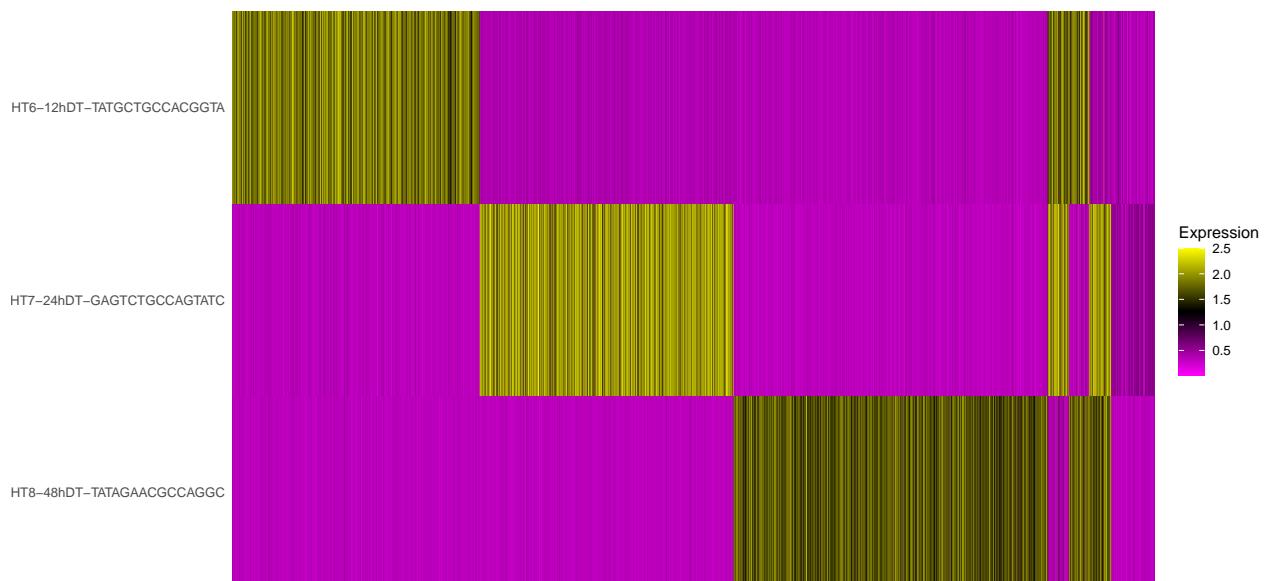
#First, we will remove negative cells from the object
immune.hashtag.subset <- subset(immune.hashtag, idents = 'Negative',
                                invert = TRUE)                                     1
                                                               2
                                                               3
# Calculate a tSNE embedding of the HTO data
DefaultAssay(immune.hashtag.subset) <- "HTO"                         4
immune.hashtag.subset <- ScaleData(immune.hashtag.subset, features =
                                    rownames(immune.hashtag.subset), verbose = FALSE) 5
immune.hashtag.subset <- RunPCA(immune.hashtag.subset, features = rownames
                                 (immune.hashtag.subset), approx = FALSE)           6
immune.hashtag.subset <- RunTSNE(immune.hashtag.subset, dims = 1:8,      7
                                  perplexity = 100, check_duplicates = FALSE)    8
                                                               9
DimPlot(immune.hashtag.subset)                                         10

```



Create an HTO heatmap

```
HTOHeatmap(immune.hashtag, assay = 'HTO', ncells = 5000)
```



Cluster and visualize cells using the usual scRNA-seq workflow, and examine for the potential presence of batch effects.

```
# Extract the singlets
immune.singlet <- subset(immune.hashtag, idents = 'Singlet')
```

```

# Select the top 1000 most variable features
immune.singlet <- FindVariableFeatures(immune.singlet, selection.method =
  'mean.var.plot')

# Scaling RNA data, we only scale the variable features here for
# efficiency
immune.singlet <- ScaleData(immune.singlet, features = VariableFeatures(
  immune.singlet))

# Run PCA
immune.singlet <- RunPCA(immune.singlet, features = VariableFeatures(
  immune.singlet))

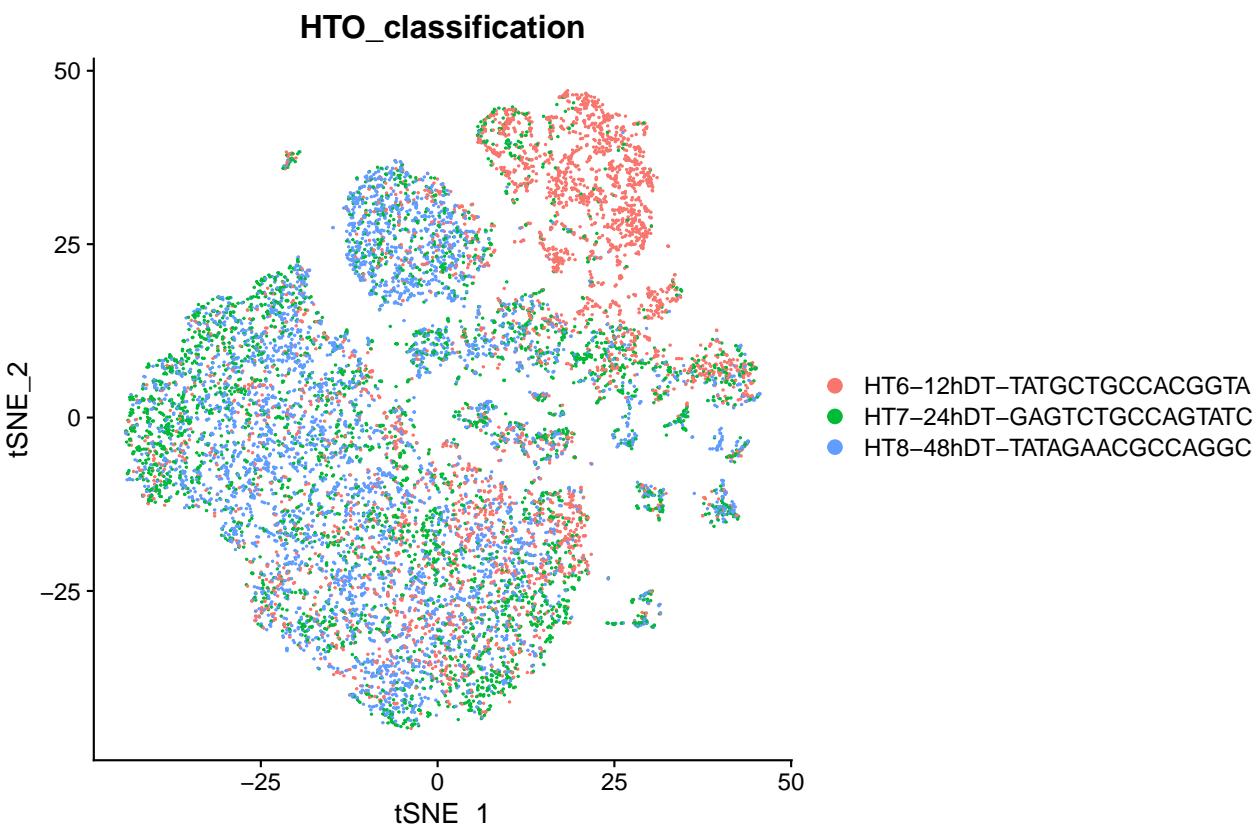
```

```

# We select the top 10 PCs for clustering and tSNE based on PCElbowPlot
immune.singlet <- FindNeighbors(immune.singlet, reduction = 'pca', dims =
  1:10)
immune.singlet <- FindClusters(immune.singlet, resolution = 0.6, verbose =
  FALSE)
immune.singlet <- RunTSNE(immune.singlet, reduction = 'pca', dims = 1:10)

# Projecting singlet identities on TSNE visualization
DimPlot(immune.singlet, group.by = "HTO_classification")

```



save to seurat object:

```
saveRDS(immune.singlet, file = "./immune.demultiplexed.seuratObject.rds")
```

3 QC of pooled sample

Load data:

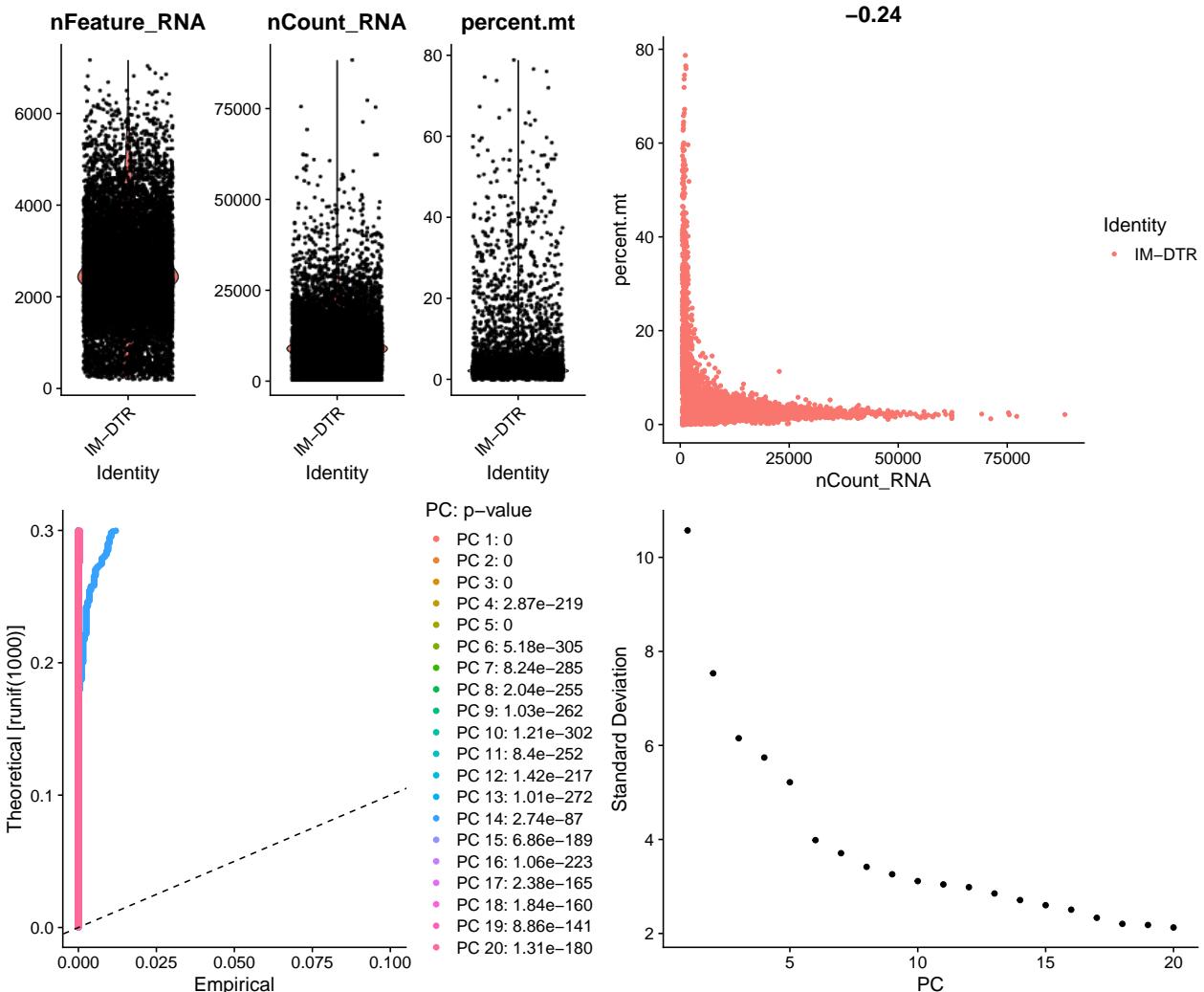
```
# load package and data 1
library(ggpubr) 2
source("../R/seurat.setup.R") 3
dir.10x <- "/mnt/Data/Single-cell_Analysis/Projects/IPL/IM_DTR/IM_DTR_exp2 4
/counts/scRNAseq/chromium"
```

3.1 Sample: CD45Plus_NGS21-S229

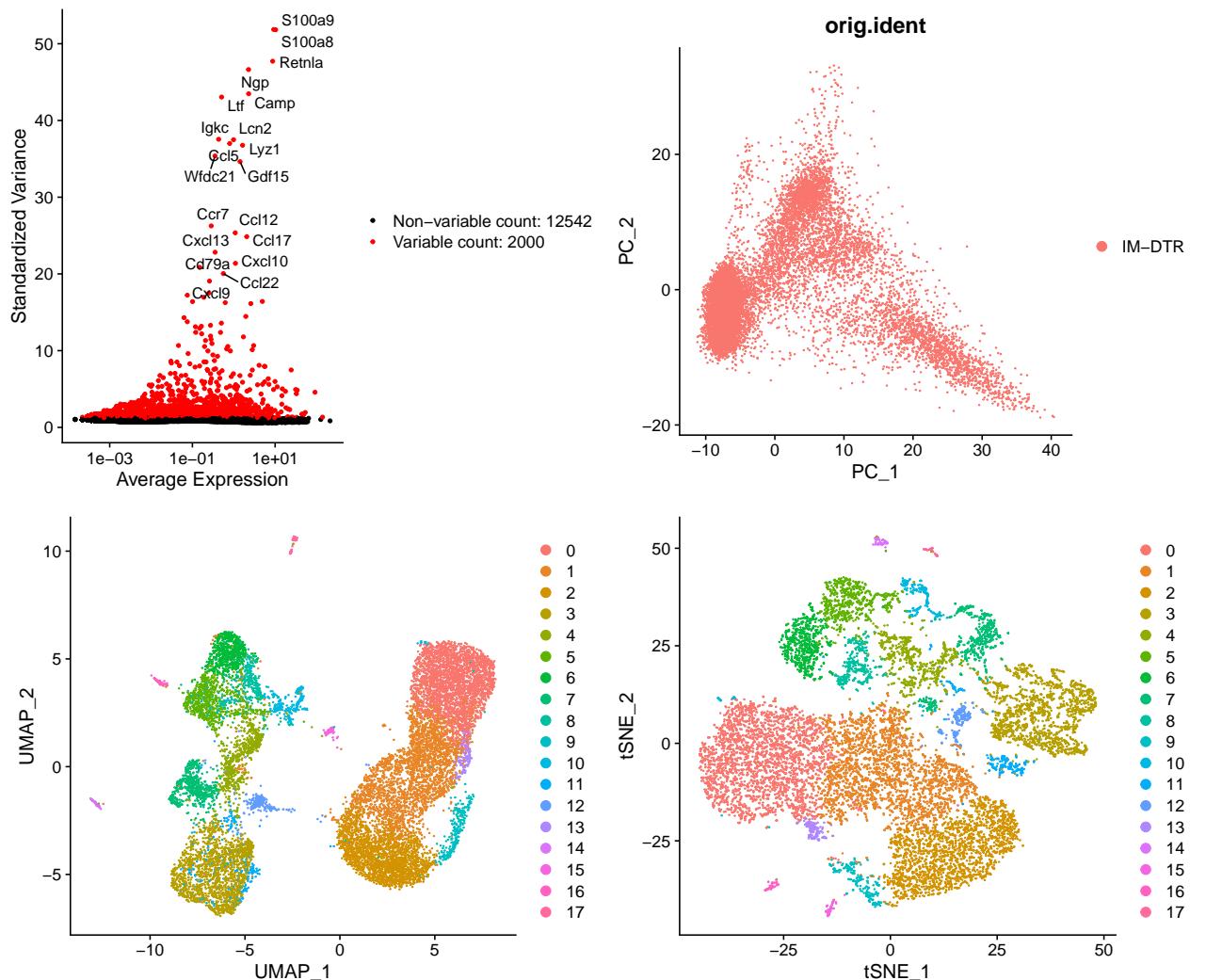
```
CD45Plus_NGS21_S229 <- seurat.setup(path.10x = file.path(dir.10x, " 1
CD45Plus_NGS21-S229/outs/filtered_feature_bc_matrix/"), project = "IM- 2
DTR", dimensionality = 1:20, mt.percentage = 20, human = FALSE) 3
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck 1
## 2
## Number of nodes: 13765 3
## Number of edges: 471965 4
## 5
## Running Louvain algorithm... 6
## Maximum modularity in 10 random starts: 0.8934 7
## Number of communities: 18 8
## Elapsed time: 1 seconds 9
```

```
ggarrange(CD45Plus_NGS21_S229$plots$feature_vln, CD45Plus_NGS21_S229$plots 1
$RNA_mt_pct_scatter, CD45Plus_NGS21_S229$plots$JackStrawPlot, CD45Plus_ 2
NGS21_S229$plots$ElbowPlot, ncol = 2, nrow = 2) 3
```



```
ggarrange(CD45Plus_NGS21_S229$plots$variable_features, CD45Plus_NGS21_S229
  $plots$PCA_plot, CD45Plus_NGS21_S229$plots$UMAP_plot, CD45Plus_NGS21_
  S229$plots$TSNE_plot, ncol = 2, nrow = 2)
```



```
# save data for next use:
saveRDS(CD45Plus_NGS21_S229$seuratObject, file = "./CD45Plus_NGS21_S229_seuratObject.rds")
```

4 Identify the cells from each samples (de-hashtagging)

Assign HTO calling to the Seurat object

```
common.immune <- intersect(colnames(CD45Plus_NGS21_S229.seuratObject),
  colnames(immune.demultiplexed.seuratObject))
CD45Plus_NGS21_S229.seuratObject <- subset(CD45Plus_NGS21_S229.seuratObject,
  cells = common.immune)
```

```
# make intersect between hto and rna cells:
hto.immune <- immune.demultiplexed.seuratObject@meta.data$HTO_classification
names(hto.immune) <- colnames(immune.demultiplexed.seuratObject)
# remove the sequence chars
```

```

hto.immune <- sub(hto.immune, pattern = "-[C,T,G,A]{0,15}$", replacement =
  "") 6

# assign to seurat object: 7
CD45Plus_NGS21_S229.seuratObject$treatment <- hto.immune[colnames(CD45Plus_8
  _NGS21_S229.seuratObject)] 9

```

4.1 Build metadata table

```

# cell type 1
CD45Plus_NGS21_S229.seuratObject$cell.type0 <- "CD45+" 2

```

save individual samples for other use:

```

obj <- CD45Plus_NGS21_S229.seuratObject 1
for (i in unique(obj$treatment)) { 2
  obj.sub <- subset(obj, subset = treatment == i) 3
  file.name <- paste("CD45Plus_NGS21_S229", i, "seuratObject.rds", sep = " 4
    ".") 5
  # saveRDS(object = obj.sub, file = file.path(".", file.name)) # Here we
    can also save the individual object. 6
}

```

4.2 Data processing for CD45Plus_NGS21_S229

```

CD45Plus_NGS21_S229.seuratObject <- NormalizeData(CD45Plus_NGS21_S229. 1
  seuratObject)
CD45Plus_NGS21_S229.seuratObject <- FindVariableFeatures(CD45Plus_NGS21_ 2
  _S229.seuratObject,
                                         selection.method = "vst", 3
                                         nfeatures = 2000)
CD45Plus_NGS21_S229.seuratObject <- ScaleData(CD45Plus_NGS21_S229. 4
  seuratObject,
                                         features = rownames(CD45Plus_ 5
    _NGS21_S229.seuratObject)
)

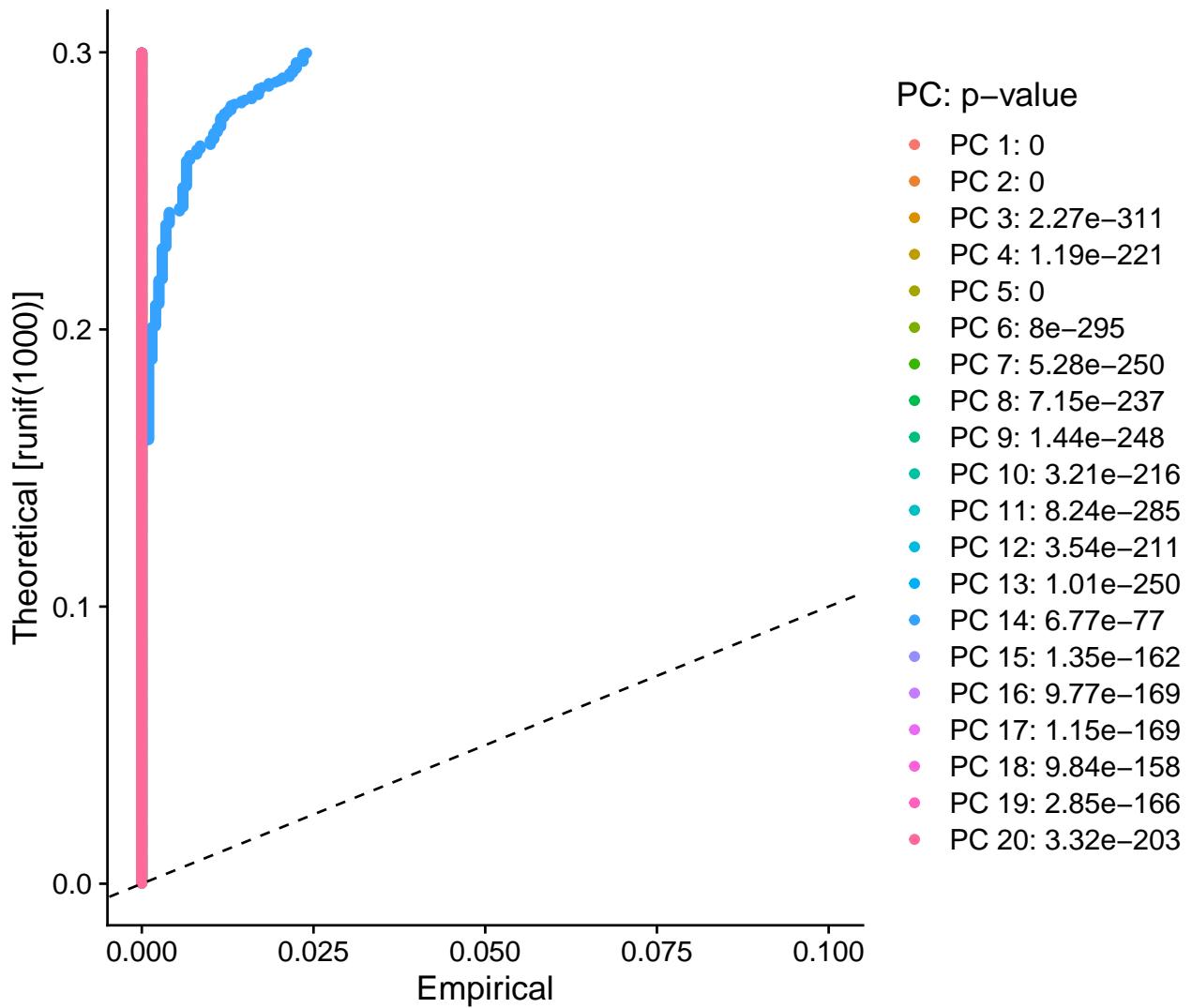
```

4.3 Linear dimension reduction

```

CD45Plus_NGS21_S229.seuratObject <- RunPCA(CD45Plus_NGS21_S229. 1
  seuratObject,
                                         features = VariableFeatures( 2
                                           object = CD45Plus_NGS21_S229_
                                             .seuratObject))
CD45Plus_NGS21_S229.seuratObject <- JackStraw(CD45Plus_NGS21_S229. 3
  seuratObject, num.replicate = 100)
CD45Plus_NGS21_S229.seuratObject <- ScoreJackStraw(CD45Plus_NGS21_S229. 4
  seuratObject, dims = 1:20)
JackStrawPlot(CD45Plus_NGS21_S229.seuratObject, dims = 1:20) 5

```

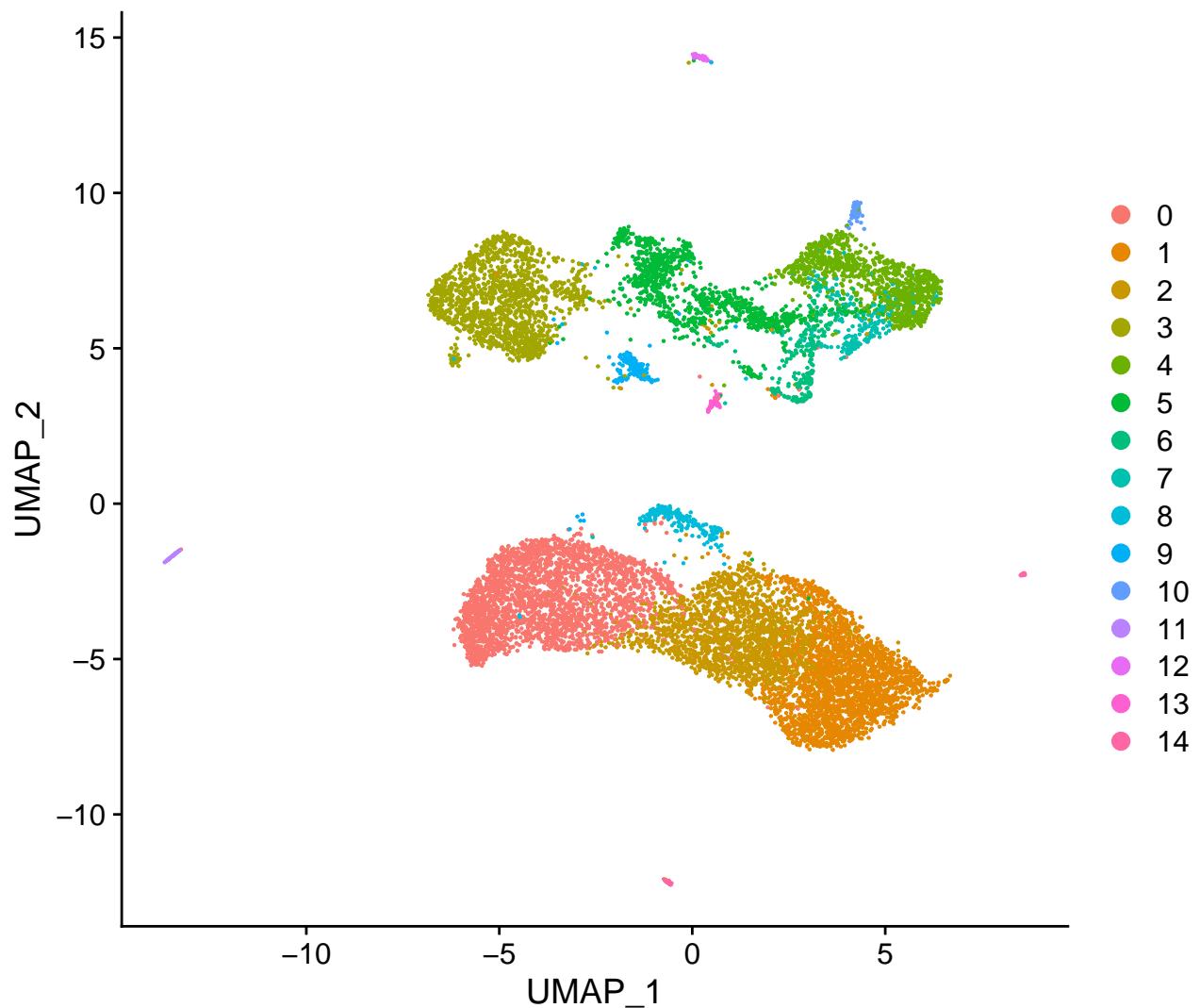


```
CD45Plus_NGS21_S229.seuratObject <- FindNeighbors(CD45Plus_NGS21_S229.  
seuratObject, dims = 1:30) 1  
CD45Plus_NGS21_S229.seuratObject <- FindClusters(CD45Plus_NGS21_S229.  
seuratObject, resolution = 0.5) 2
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck 1  
## 2  
## Number of nodes: 12127 3  
## Number of edges: 434299 4  
## 5  
## Running Louvain algorithm... 6  
## Maximum modularity in 10 random starts: 0.8877 7  
## Number of communities: 15 8  
## Elapsed time: 1 seconds 9
```

```
CD45Plus_NGS21_S229.seuratObject <- RunUMAP(CD45Plus_NGS21_S229.  
seuratObject, dims = 1:30) 1  
CD45Plus_NGS21_S229.seuratObject <- RunUMAP(CD45Plus_NGS21_S229.  
seuratObject, dims = 1:30) 2
```

```
DimPlot(CD45Plus_NGS21_S229.seuratObject)
```



5 Identify cell types

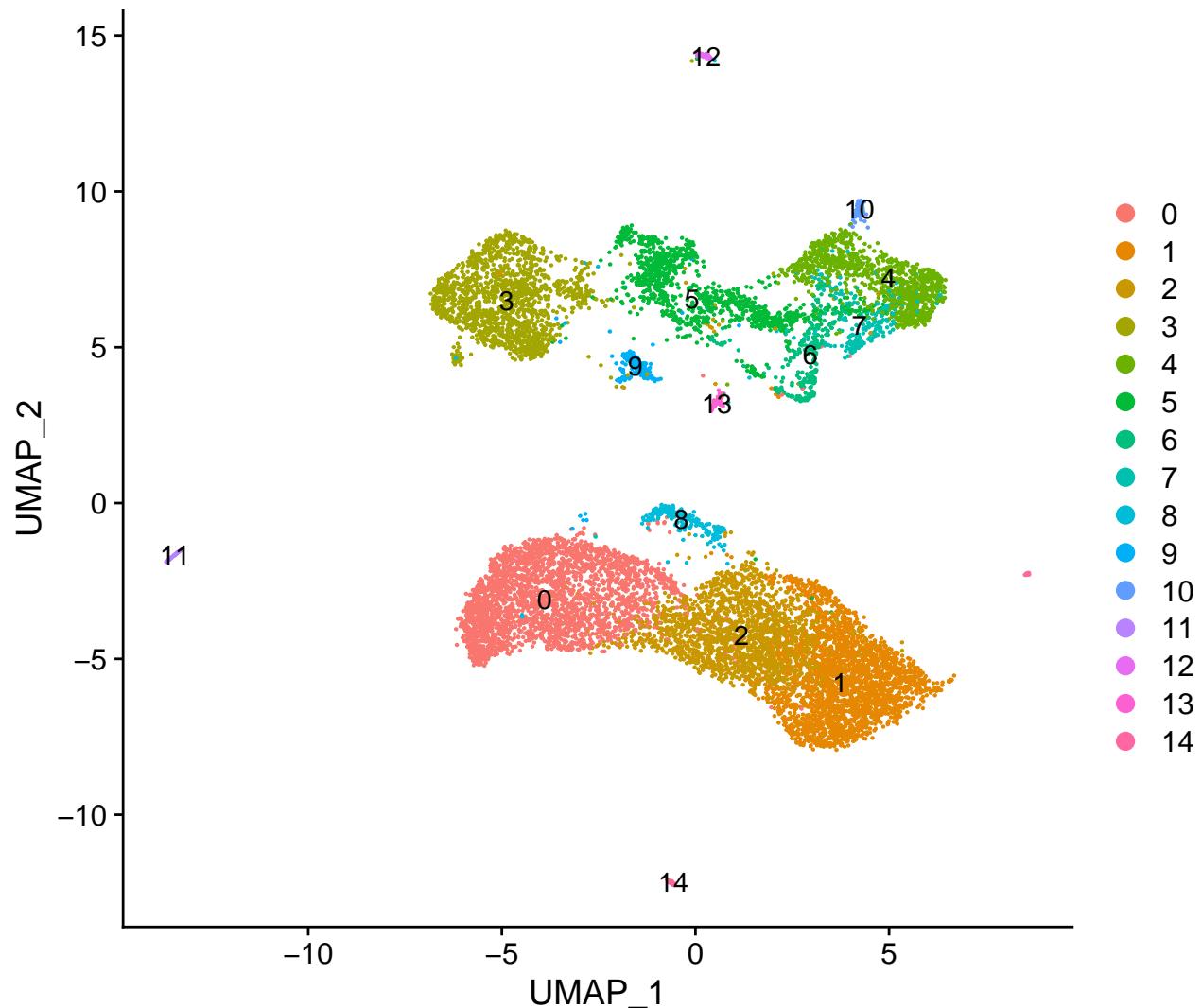
5.1 SingleR analysis

```
source("../R/seurat2singleR.R")
results.singleR <- seurat2singleR(CD45Plus_NGS21_S229.seuratObject, ref = 1
"ImmGenData") 2
```

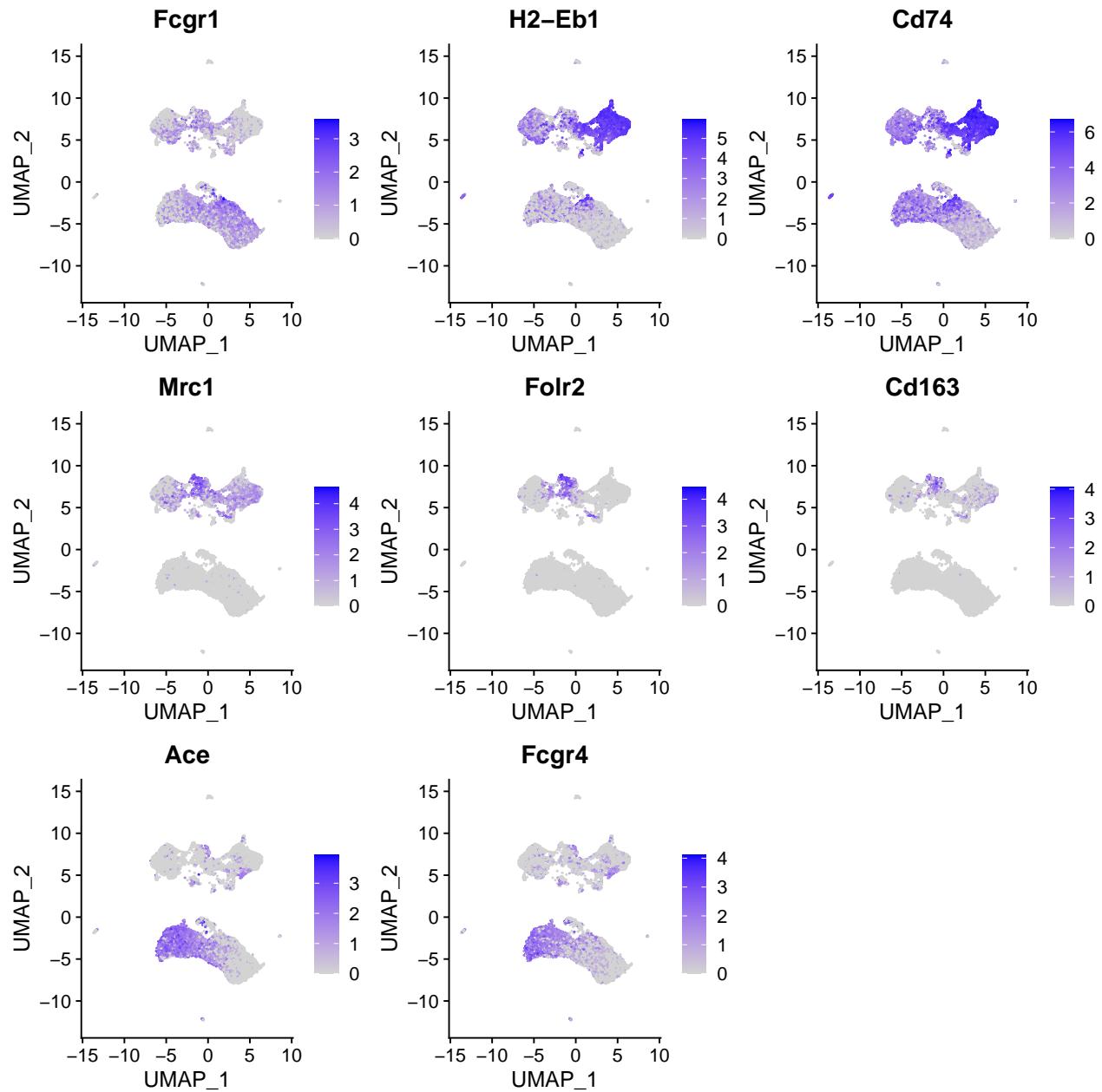
```
saveRDS(results.singleR, file = "./CD45Plus_NGS21_S229.ImmGenData.singleR. 1
Rds")
saveRDS(CD45Plus_NGS21_S229.seuratObject, file = "./CD45Plus_NGS21_S229. 2
seuratObject.rds")
```

5.2 Marker expression check

```
results <- CD45Plus_NGS21_S229.seuratObject  
DimPlot(results, reduction = "umap", label = TRUE)
```



```
DefaultAssay(results) <- "RNA"  
FeaturePlot(results, features = c("Fcgr1", "H2-Eb1", "Cd74", "Mrc1", "Folr2",  
"Cd163", "Ace", "Fcgr4"), reduction = "umap")
```



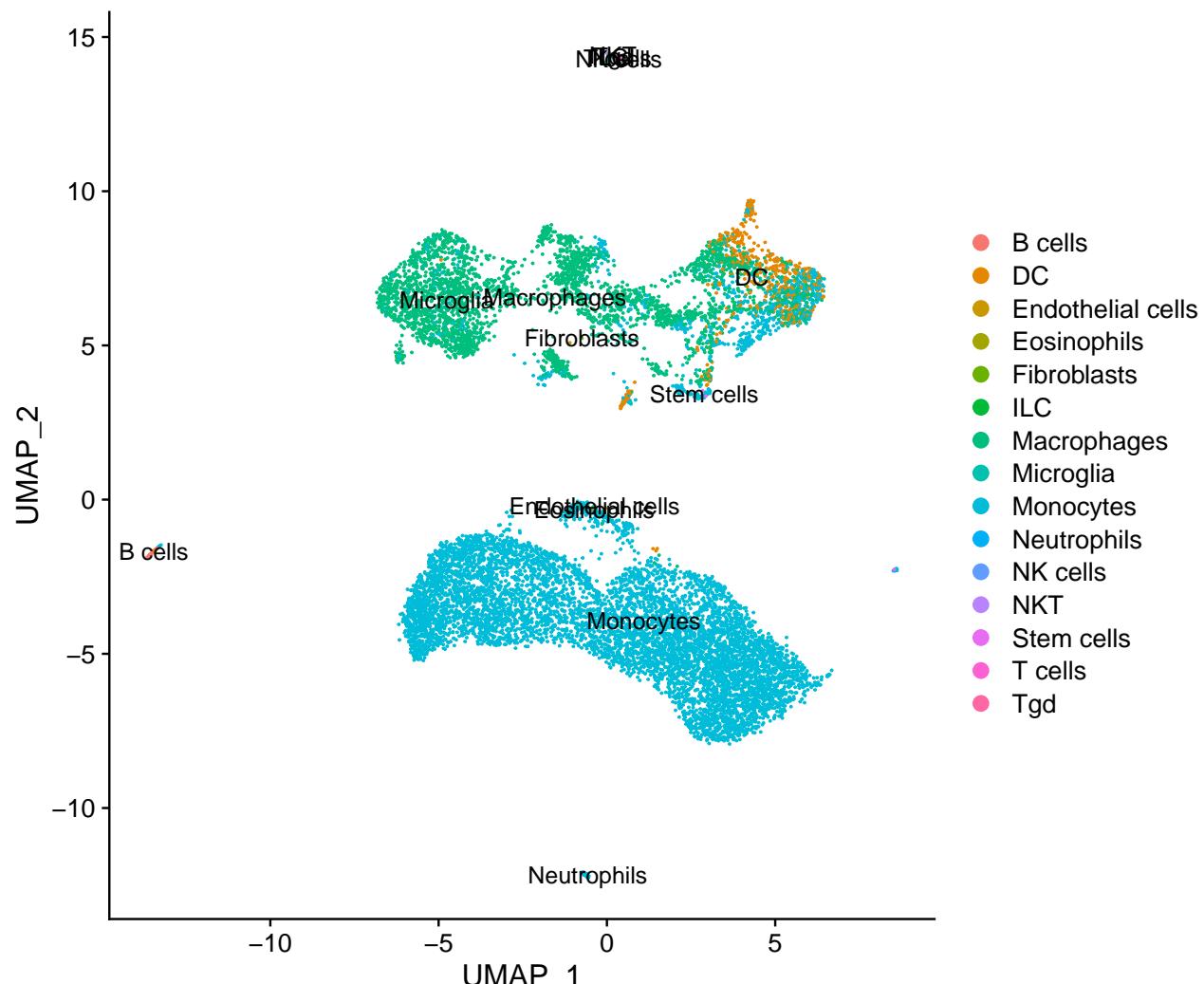
5.3 Combine marker expression with SingleR results

```

results$singleR.celltype <- results.singleR$labels
1
2
DimPlot(results, group.by = "singleR.celltype", reduction = "umap", label
= T) + ggttitle("DatabaseImmuneCellExpressionData - SingleR identity")
3

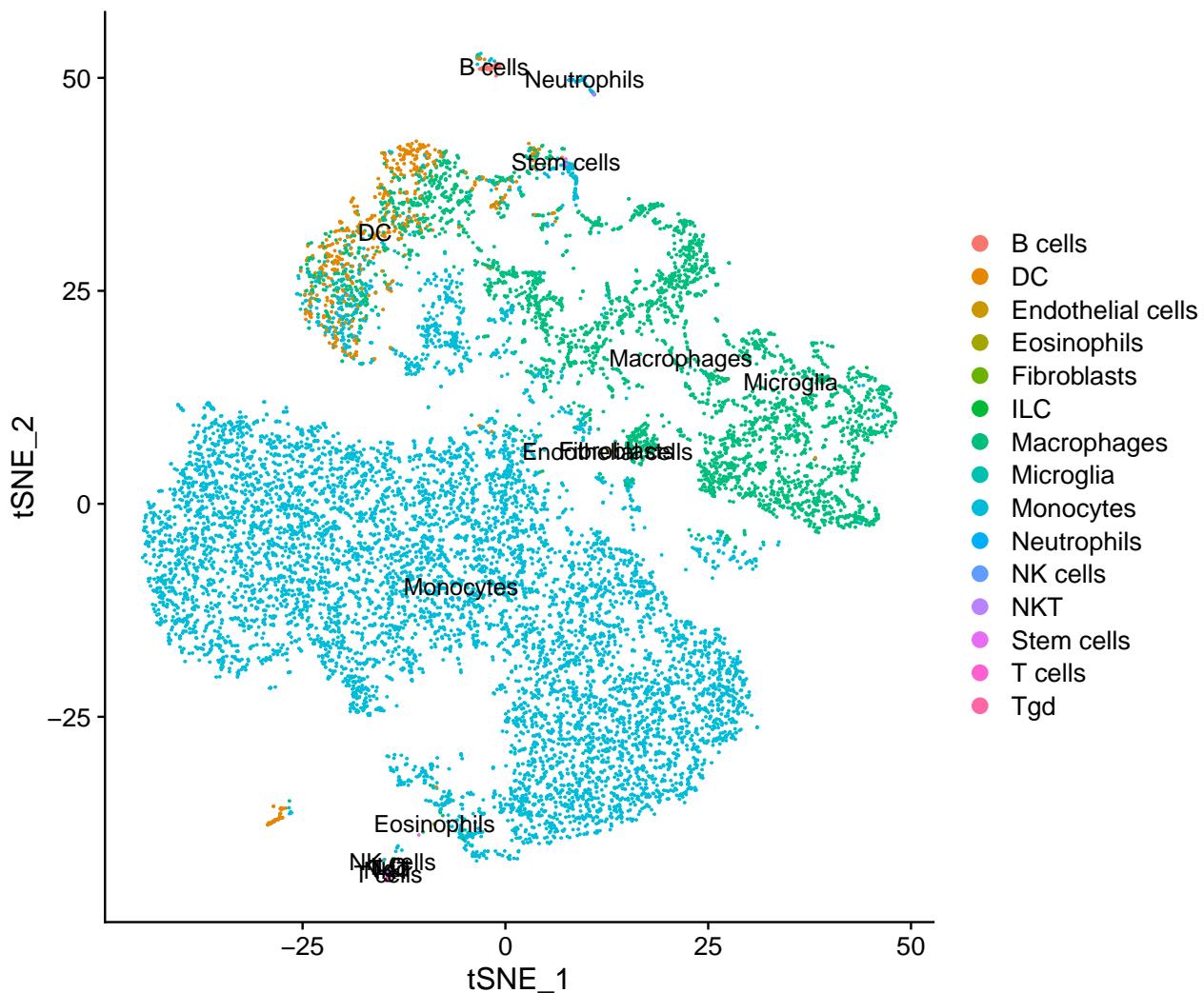
```

DatabaseImmuneCellExpressionData – SingleR identity



```
DimPlot(results, group.by = "singleR.celltype", reduction = "tsne", label  
= T) + ggtitle("DatabaseImmuneCellExpressionData - SingleR identity") 1
```

DatabaseImmuneCellExpressionData – SingleR identity



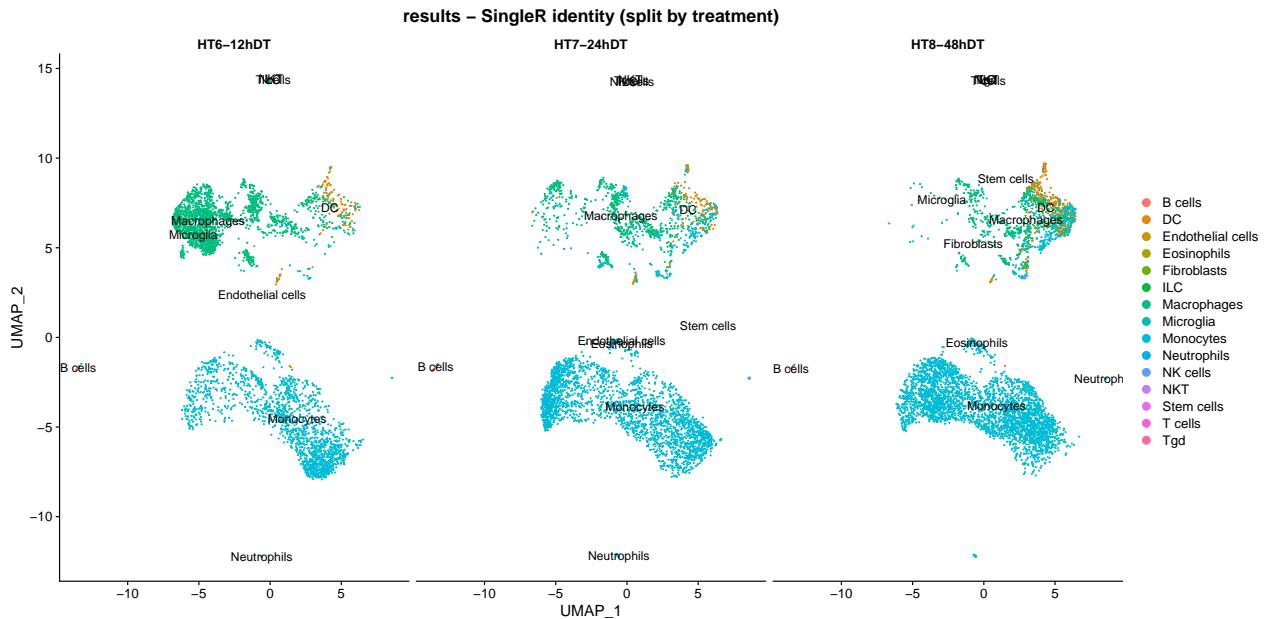
Cell numbers of each type:

```
table(results$singleR.celltype)
```

##	B cells	DC	Endothelial cells	Eosinophils	1
##	48	568	3	3	2
##	Fibroblasts	ILC	Macrophages	Microglia	3
##	1	12	3273	6	5
##	Monocytes	Neutrophils	NK cells	NKT	6
##	8136	29	2	9	7
##	Stem cells	T cells	Tgd		8
##	4	32	1		9

Split by treatment: contaminated cells came from the same sample?

```
DimPlot(results, group.by = "singleR.celltype", reduction = "umap", label = T, split.by = "treatment") + ggtitle("results - SingleR identity (split by treatment)")
```



5.4 Remove contaminated cell types other than monocytes/macrophages

```
results <- subset(results, subset = singleR.celltype %in% c("B_cells", "1
  Endothelial_cells", "Eosinophils", "Fibroblasts", "ILC", "Neutrophils",
  "NK_cells", "NKT", "Stem_cells", "T_cells", "Tgd"), invert = TRUE)
```

Here we keep some closely-related cell types, like DC, microglia etc.

6 Phenotypic characterization of monocytes/macrophages

6.1 Data processing after filtering:

Remove old snn:

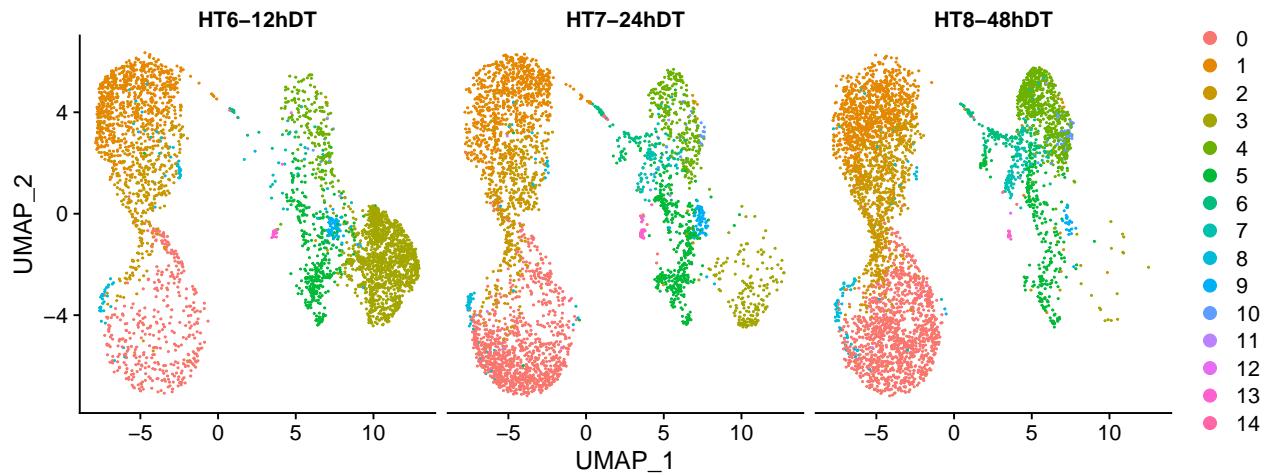
```
results$RNA_snn_res.0.5 <- NULL 1
```

```
DefaultAssay(results) <- "RNA" 1
results <- NormalizeData(results)
results <- FindVariableFeatures(results, selection.method = "vst", 2
  nfeatures = 2000)
results <- ScaleData(results, features = rownames(results)) 3
results <- RunPCA(results, features = VariableFeatures(results)) 4
  5
```

Non-linear dimension reduction: Let's choose $dim = 1:10$, showing very decent bridge between monocytes and macrophages.

```
results <- RunTSNE(results, dims = 1:10) 1
results <- RunUMAP(results, dims = 1:10) 2
```

```
DimPlot(results, reduction = "umap", split.by = "treatment") 1
```

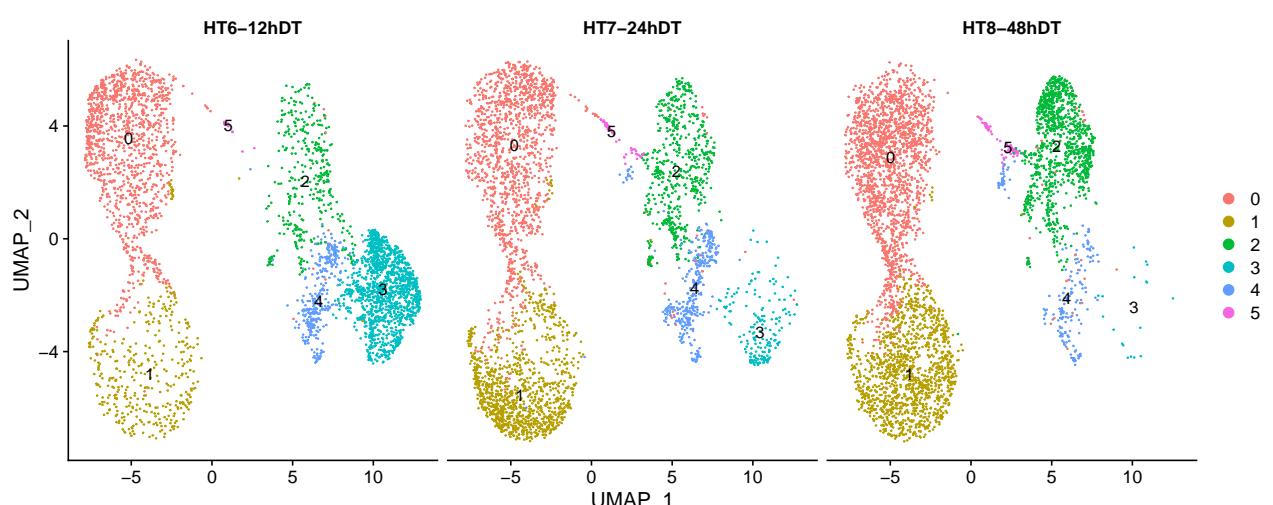


6.2 Clustering of cells

```
results <- FindNeighbors(results, dims = 1:10) 1
results <- FindClusters(results, resolution = 0.14) 2
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck 1
## 2
## Number of nodes: 11983 3
## Number of edges: 389445 4
## 5
## Running Louvain algorithm... 6
## Maximum modularity in 10 random starts: 0.9492 7
## Number of communities: 6 8
## Elapsed time: 1 seconds 9
```

```
Idents(results) <- "RNA_snn_res.0.14" 1
DimPlot(results, label = TRUE, split.by = "treatment") 2
```



Distribution of clusters across samples:

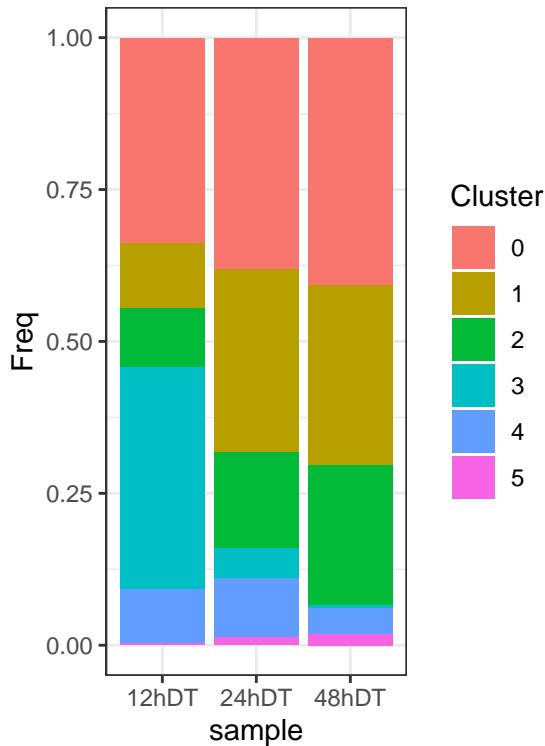
```
source("../R/SeuratFreqTable.R") 1
freq.celltype.list <- list( 2
```

```

`12hDT` = Seurat2CellFreqTable(subset(results, subset = treatment == "HT6-12hDT"), slotName = "RNA_snn_res.0.14"),
`24hDT` = Seurat2CellFreqTable(subset(results, subset = treatment == "HT7-24hDT"), slotName = "RNA_snn_res.0.14"),
`48hDT` = Seurat2CellFreqTable(subset(results, subset = treatment == "HT8-48hDT"), slotName = "RNA_snn_res.0.14")
)

source("../R/barChart.R")
barChart(freq.celltype.list) + labs(fill = "Cluster")

```

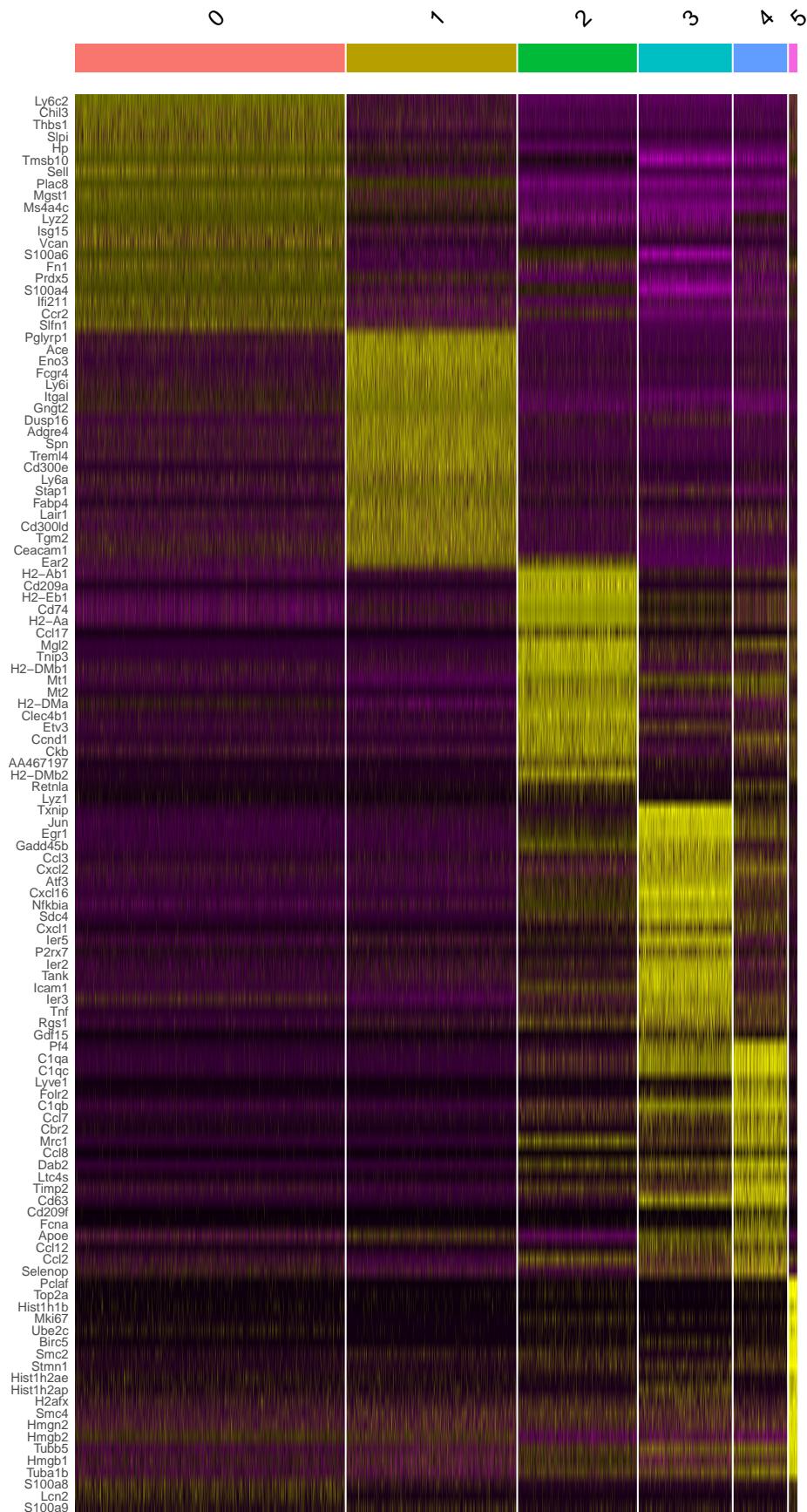


6.3 Population characterization

```

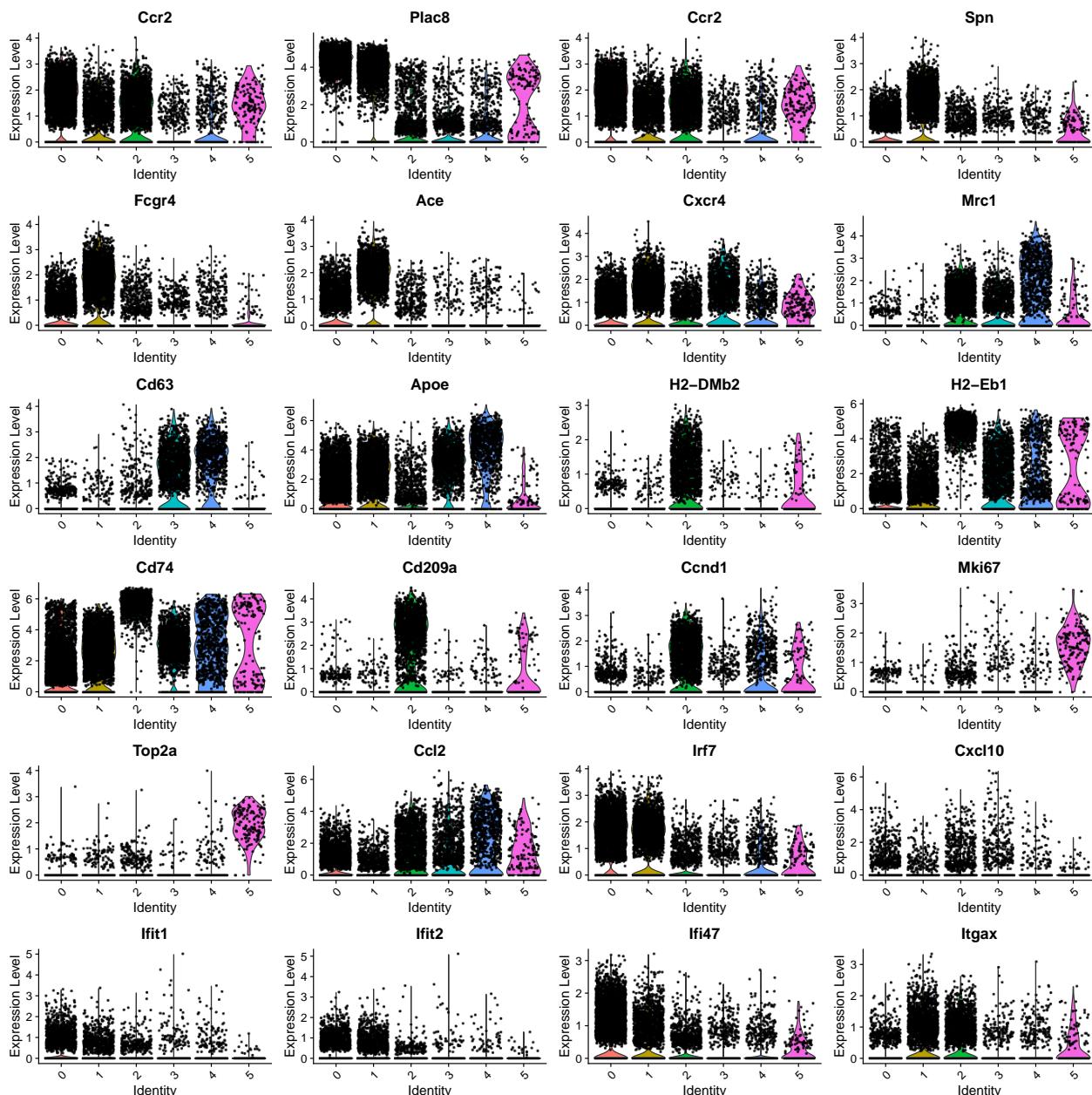
library(dplyr)
all_cluster.markers <- FindAllMarkers(results)
top20 <- all_cluster.markers %>% group_by(cluster) %>% top_n(n = 20, wt =
  avg_log2FC)
DoHeatmap(results, features = top20$gene) + NoLegend()

```



Expression of relevant genes

```
VlnPlot(results, features = c("Ccr2", "Plac8", "Ccr2", "Spn",
                               "Fcgr4", "Ace", "Cxcr4",
                               "Mrc1", "Cd63", "Apoe",
                               "H2-DMb2", "H2-Eb1", "Cd74", "Cd209a",
                               "Ccnd1", "Mki67", "Top2a", "Ccl2",
                               "Irf7", "Cxcl10", "Ifit1", "Ifit2", "Ifi47",
                               "Itgax"),
# split.by = "treatment",
ncol = 4)
```

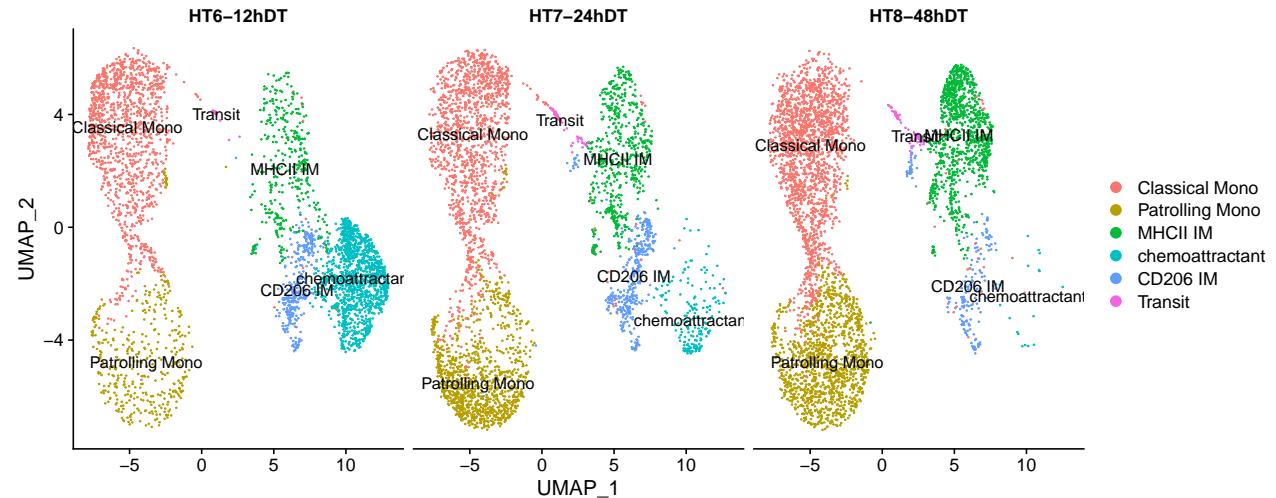


Here's the cell type annotation: cluster 0: Classical Mono cluster 1: Patrolling Mono cluster 2: MHCII IM
cluster 3: chemoattractant cluster 4: CD206 IM cluster 5: Transit

```

levels(results)                                1
## [1] "0" "1" "2" "3" "4" "5"                  1
results$cell.type2 <- factor(Idents(results), labels = c("Classical_Mono", 1
  "Patrolling_Mono", "MHCII_IM", "chemoattractant", "CD206_IM", "Transit" 1
  ""))
Idents(results) <- "cell.type2"                2
DimPlot(results, label = TRUE, split.by = "treatment") 3

```

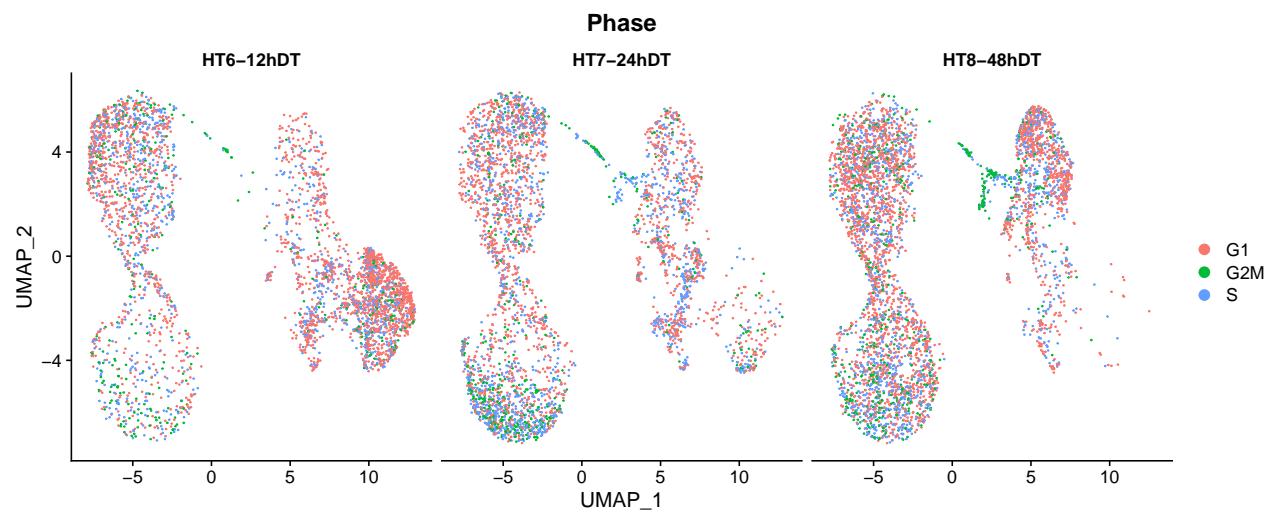
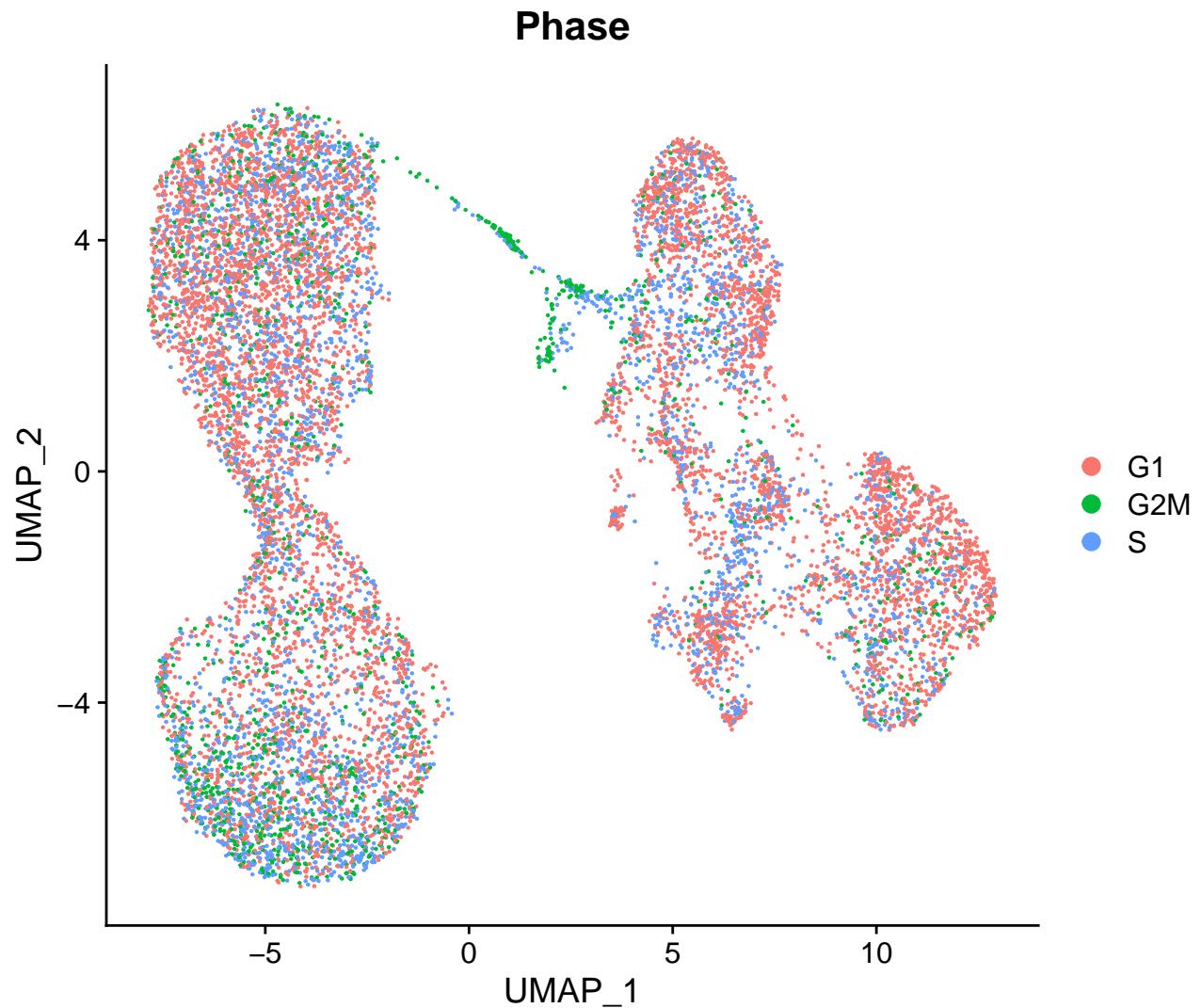


6.4 Cell cycle analysis

```

library(cowplot)
data("geneinfo_human", package = "nichenetr")
s.genes <- nichenetr::convert_human_to_mouse_symbols(cc.genes.updated.2019 1
  $s.genes)
g2m.genes <- nichenetr::convert_human_to_mouse_symbols(cc.genes.updated 2
  .2019$g2m.genes)
results <- CellCycleScoring(results, s.features = s.genes, g2m.features = 3
  g2m.genes, set.ident = FALSE)
DimPlot(results, group.by = "Phase")           4

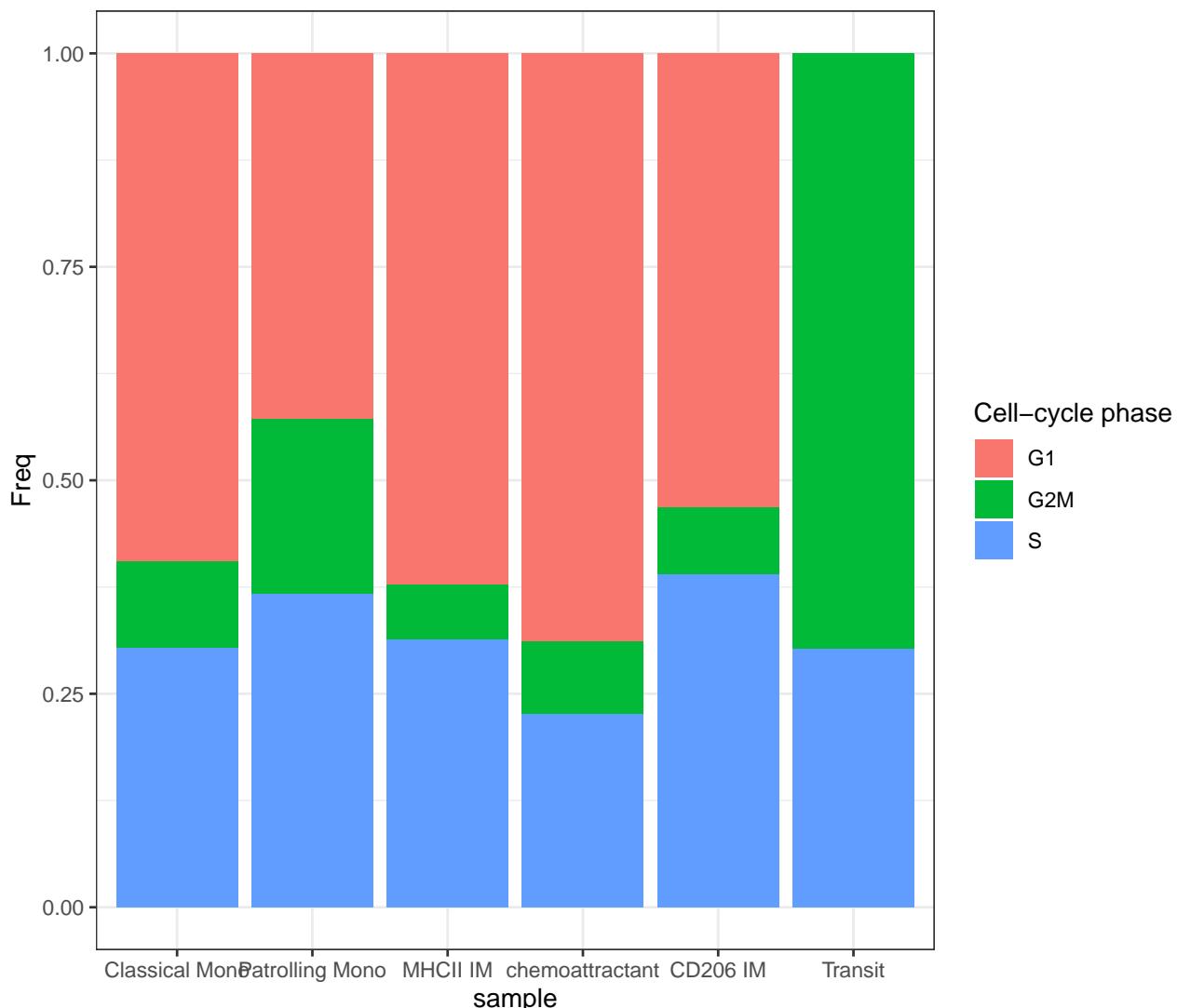
```



cluster 0: Classical Mono
 cluster 1: Patrolling Mono
 cluster 2: MHCII IM
 cluster 3: chemoattractant
 cluster 4: CD206 IM
 cluster 5: Transit

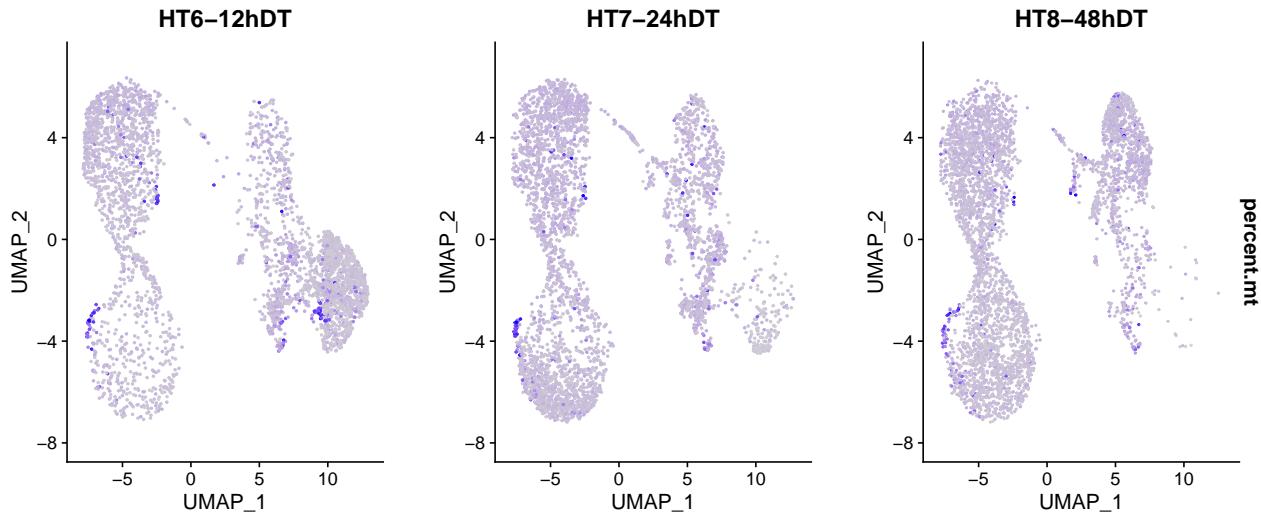
```

freq.celltype.list <- list(
  `Classical Mono` = Seurat2CellFreqTable(subset(results, ident = "Classical_Mono"), slotName = "Phase"),
  `Patrolling Mono` = Seurat2CellFreqTable(subset(results, ident = "Patrolling_Mono"), slotName = "Phase"),
  `MHCII IM` = Seurat2CellFreqTable(subset(results, ident = "MHCII_IM"), slotName = "Phase"),
  `chemoattractant` = Seurat2CellFreqTable(subset(results, ident = "chemoattractant"), slotName = "Phase"),
  `CD206 IM` = Seurat2CellFreqTable(subset(results, ident = "CD206_IM"), slotName = "Phase"),
  `Transit` = Seurat2CellFreqTable(subset(results, ident = "Transit"), slotName = "Phase")
)
barChart(freq.celltype.list) + labs(fill = "Cell-cycle phase")
  
```



6.5 Apoptosis rate by mitochondrial genes

```
FeaturePlot(results, features = "percent.mt", reduction = "umap", split.by = "treatment")
```



```
saveRDS(results, file = "./immune.cellType_filtered.withSingleR.seuratObject.Rds")
```

7 Session information

```
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)                                         1
## Platform: x86_64-pc-linux-gnu (64-bit)                                2
## Running under: Ubuntu 20.04.3 LTS                                       3
##
## Matrix products: default                                              4
## BLAS:    /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3          5
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/liblapack.so.3        6
##
## locale:                                                               7
##   [1] LC_CTYPE=en_US.UTF-8          LC_NUMERIC=C                      8
##   [3] LC_TIME=en_GB.UTF-8          LC_COLLATE=en_US.UTF-8           9
##   [5] LC_MONETARY=en_GB.UTF-8      LC_MESSAGES=en_US.UTF-8          10
##   [7] LC_PAPER=en_GB.UTF-8         LC_NAME=C                        11
##   [9] LC_ADDRESS=C                 LC_TELEPHONE=C                  12
##  [11] LC_MEASUREMENT=en_GB.UTF-8   LC_IDENTIFICATION=C             13
##
## attached base packages:                                                 14
## [1] parallel      stats        graphics     grDevices    utils
## datasets
## [8] methods       base
##
## other attached packages:                                                15
## [1] cowplot_1.1.1            RColorBrewer_1.1-2               16
## [3] celldex_1.0.0            SingleR_1.4.1                   17
## [4] dplyr_1.0.2               ggplot2_3.3.5                   18
## [5] gridExtra_2.3.0            ggrepel_0.9.1                   19
## [6] knitr_1.33                magrittr_2.0.1                  20
## [7] rlang_0.4.10              readr_2.0.1                     21
## [8] rmarkdown_2.5              rvest_0.3.6                     22
## [9] tibble_3.1.2               tidyverse_1.3.1                 23
```

## [5] SummarizedExperiment_1.20.0	Biobase_2.50.0	24
## [7] GenomicRanges_1.42.0	GenomeInfoDb_1.26.7	25
## [9] IRanges_2.24.1	S4Vectors_0.28.1	26
## [11] BiocGenerics_0.36.1	MatrixGenerics_1.2.1	27
## [13] matrixStats_0.61.0	dplyr_1.0.8	28
## [15] ggpubr_0.4.0	ggplot2_3.3.5	29
## [17] SeuratObject_4.0.4	Seurat_4.1.0	30
##		31
## loaded via a namespace (and not attached):		32
## [1] rappdirs_0.3.3	scattermore_0.8	33
## [3] ModelMetrics_1.2.2.2	nichenetr_1.0.0	34
## [5] tidyR_1.2.0	bit64_4.0.5	35
## [7] knitr_1.37	irlba_2.3.5	36
## [9] DelayedArray_0.16.3	data.table_1.14.2	37
## [11] rpart_4.1-15	hardhat_0.2.0	38
## [13] RCurl_1.98-1.6	generics_0.1.2	39
## [15] RSQLite_2.2.10	RANN_2.6.1	40
## [17] proxy_0.4-26	future_1.24.0	41
## [19] DiagrammeR_1.0.8	bit_4.0.4	42
## [21] tzdb_0.2.0	spatstat.data_2.1-2	43
## [23] lubridate_1.8.0	httpuv_1.6.5	44
## [25] assertthat_0.2.1	gower_1.0.0	45
## [27] xfun_0.29	hms_1.1.1	46
## [29] evaluate_0.15	promises_1.2.0.1	47
## [31] fansi_1.0.2	caTools_1.18.2	48
## [33] dbplyr_2.1.1	igraph_1.2.11	49
## [35] DBI_1.1.2	htmlwidgets_1.5.4	50
## [37] spatstat.geom_2.3-2	purrr_0.3.4	51
## [39] ellipsis_0.3.2	RSpectra_0.16-0	52
## [41] backports_1.4.1	deldir_1.0-6	53
## [43] sparseMatrixStats_1.2.1	vctrs_0.3.8	54
## [45] Cairo_1.5-14	ROCR_1.0-11	55
## [47] abind_1.4-5	caret_6.0-90	56
## [49] cachem_1.0.6	withr_2.4.3	57
## [51] checkmate_2.0.0	sctransform_0.3.3	58
## [53] fdrtool_1.2.17	goftest_1.2-3	59
## [55] cluster_2.1.0	ExperimentHub_1.16.1	60
## [57] lazyeval_0.2.2	crayon_1.5.0	61
## [59] recipes_0.2.0	pkgconfig_2.0.3	62
## [61] labeling_0.4.2	nlme_3.1-155	63
## [63] viper_0.4.5	nnet_7.3-14	64
## [65] rlang_1.0.1	globals_0.14.0	65
## [67] lifecycle_1.0.1	miniUI_0.1.1.1	66
## [69] BiocFileCache_1.14.0	rsvd_1.0.5	67
## [71] AnnotationHub_2.22.1	randomForest_4.6-14	68
## [73] ggrastr_1.0.1	polyclip_1.10-0	69
## [75] lmtest_0.9-39	Matrix_1.4-0	70
## [77] carData_3.0-5	zoo_1.8-9	71
## [79] base64enc_0.1-3	beeswarm_0.4.0	72
## [81] ggridges_0.5.3	png_0.1-7	73
## [83] viridisLite_0.4.0	bitops_1.0-7	74
## [85] KernSmooth_2.23-20	visNetwork_2.1.0	75
## [87] pROC_1.18.0	blob_1.2.2	76
## [89] DelayedMatrixStats_1.12.3	stringr_1.4.0	77

## [91] parallely_1.30.0	spatstat.random_2.1-0	78
## [93] readr_2.1.2	jpeg_0.1-9	79
## [95] rstatix_0.7.0	ggsignif_0.6.3	80
## [97] beachmat_2.6.4	scales_1.1.1	81
## [99] memoise_2.0.1	magrittr_2.0.2	82
## [101] plyr_1.8.6	ica_1.0-2	83
## [103] zlibbioc_1.36.0	compiler_4.0.3	84
## [105] fitdistrplus_1.1-6	cli_3.2.0	85
## [107] XVector_0.30.0	listenv_0.8.0	86
## [109] patchwork_1.1.1	pbapply_1.5-0	87
## [111] htmlTable_2.4.0	Formula_1.2-4	88
## [113] MASS_7.3-53	mgcv_1.8-33	89
## [115] tidyselect_1.1.1	stringi_1.7.6	90
## [117] highr_0.9	yaml_2.3.5	91
## [119] BiocSingular_1.6.0	latticeExtra_0.6-29	92
## [121] ggrepel_0.9.1	grid_4.0.3	93
## [123] tools_4.0.3	future.apply_1.8.1	94
## [125] rstudioapi_0.13	foreach_1.5.2	95
## [127] foreign_0.8-82	gridExtra_2.3	96
## [129] prodlim_2019.11.13	farver_2.1.0	97
## [131] Rtsne_0.15	digest_0.6.29	98
## [133] BiocManager_1.30.16	shiny_1.7.1	99
## [135] lava_1.6.10	Rcpp_1.0.8	100
## [137] car_3.0-12	broom_0.7.12	101
## [139] BiocVersion_3.12.0	later_1.3.0	102
## [141] RcppAnnoy_0.0.19	httr_1.4.2	103
## [143] AnnotationDbi_1.52.0	colorspace_2.0-3	104
## [145] tensor_1.5	reticulate_1.24	105
## [147] splines_4.0.3	uwot_0.1.11	106
## [149] spatstat.utils_2.3-0	plotly_4.10.0	107
## [151] xtable_1.8-4	jsonlite_1.7.3	108
## [153] timeDate_3043.102	ipred_0.9-12	109
## [155] R6_2.5.1	Hmisc_4.6-0	110
## [157] pillar_1.7.0	htmltools_0.5.2	111
## [159] mime_0.12	glue_1.6.1	112
## [161] fastmap_1.1.0	BiocParallel_1.24.1	113
## [163] BiocNeighbors_1.8.2	class_7.3-17	114
## [165] interactiveDisplayBase_1.28.0	codetools_0.2-18	115
## [167] utf8_1.2.2	lattice_0.20-41	116
## [169] spatstat.sparse_2.1-0	tibble_3.1.6	117
## [171] curl_4.3.2	ggbeeswarm_0.6.0	118
## [173] leiden_0.3.9	survival_3.2-7	119
## [175] limma_3.46.0	rmarkdown_2.11	120
## [177] munsell_0.5.0	e1071_1.7-9	121
## [179] GenomeInfoDbData_1.2.4	iterators_1.0.14	122
## [181] reshape2_1.4.4	gttable_0.3.0	123
## [183] spatstat.core_2.4-0		124

References

1. Hao, Y. *et al.* Integrated analysis of multimodal single-cell data. *Cell* (2021) doi:10.1016/j.cell.2021.04.048.

-
2. Aran, D. *et al.* Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. *Nature Immunology* **20**, 163–172 (2019).