PAPER TITLE TO BE DEFINED (in common.yaml)

12 - cMAF- and Mafb-deficient IM

2022-01-14 11:22:38 +0100

Abstract

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 control 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.

Contents

1 Description

We would like to know the functions of cMAF and MAFb transcription factors in IM differentiation or IM refilling in empty niche. We analyze the IM with cMAF-KO, MAFb-KO, cMAF/MAF-dKO and control phenotype using scRNAseq (10X Genomics). These mice are Ms4a3-Cre induced recombination KO mice, thus gene KO is dependent on the Ms4a3 expression. We know in 6-week old Ms4a3-dTomato mice, dTomato+ cells don't present in all the whole IM population (~80%).

The solution is to use the BM chimera with BM of Ms4a3-cMAF, Ms4a3-MAFb, Ms4a3-cMAF/MAFb and littermate control transferred into CD45.1-IM-DTR mice after thorax-protection irradiation. The monocytes with Ms4a3-induced KO will take over the lung after the IM niche is emptied by DT treatment. After 1 week, the CD45.2+CD45.1- (originated from transferred BM) IM and monocytes should be IM and monocytes with right gene KO.

In this experiment, we have 4 groups of chimera mice (host = CD45.1-IM-DTR):

```
1. Three chimera with littermate control BM (control group);
2. Three chimera with Ms4a3-cMAF BM  (cMAF-KO group);
3. Three chimera with Ms4a3-MAFb BM  (MAFb-KO group);
4. Three chimera with Ms4a3-cMAF/MAFb BM  (cMAF/MAFb-dKO group).
```

Lung monocytes/macrophages will be sorted. For each sample, input 5K cells will be needed for scRNA-seq library construction to target recovered cell number of 3K cells. Sample from each time point should be stained with anti-MHCI-hashtag (Biolegend Hashtag) before pooled sequencing.

2 Demultiplexing by hashtag sequences

```
suppressMessages({
  library(Seurat)
  library(ggpubr)
})
```

2.1 Read data

```
# Load in the UMI matrix (from CR)

umis <- Read10X("../../../counts/scRNAseq/Experiment-7-12-21-ScRNA_NGS21-U976/outs/filtered_feature_bc_matrix/")

# For generating a hashtag count matrix from FASTQ files, please refer to https://github.com/Hoohm/CITE-seq-Count.
# Load in the HTO count matrix
htos <- Read10X("../../../counts/HTO/read_count/", gene.column=1)
```

```
# change cell names to "-1" mode.
colnames(htos) <- paste0(colnames(htos), "-1")

# remove the unmapped as it's not a barcode. It's the last row.
htos <- htos[-nrow(htos), ]

# 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(umis), colnames(htos))
```

```
# Subset RNA and HTO counts by joint cell barcodes
umis <- umis[, joint.bcs]
htos <- as.matrix(htos[, joint.bcs])


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

```
## [1] "HT5_Control-CTTTGTCTTTGTGAG" "HT6_cMAF_KO-TATGCTGCCACGGTA"
## [3] "HT7_MAFb_KO-GAGTCTGCCAGTATC" "HT8_dKO-TATAGAACGCCAGGC"
```

Setup Seurat object and add in the HTO data

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

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

2.2 Adding HTO data as an independent assay

You can read more about working with multi-modal data here

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

2.3 Demultiplex cells based on HTO enrichment

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

```
# If you have a very large dataset we suggest using k_function = "clara".
    This is a k-medoid clustering function for large applications
# You can also play with additional parameters (see documentation for
    HTODemux()) to adjust the threshold for classification
# Here we are using the default settings
hashtag <- HTODemux(hashtag, assay = "HTO", positive.quantile = 0.94)
```

2.4 Visualize demultiplexing results

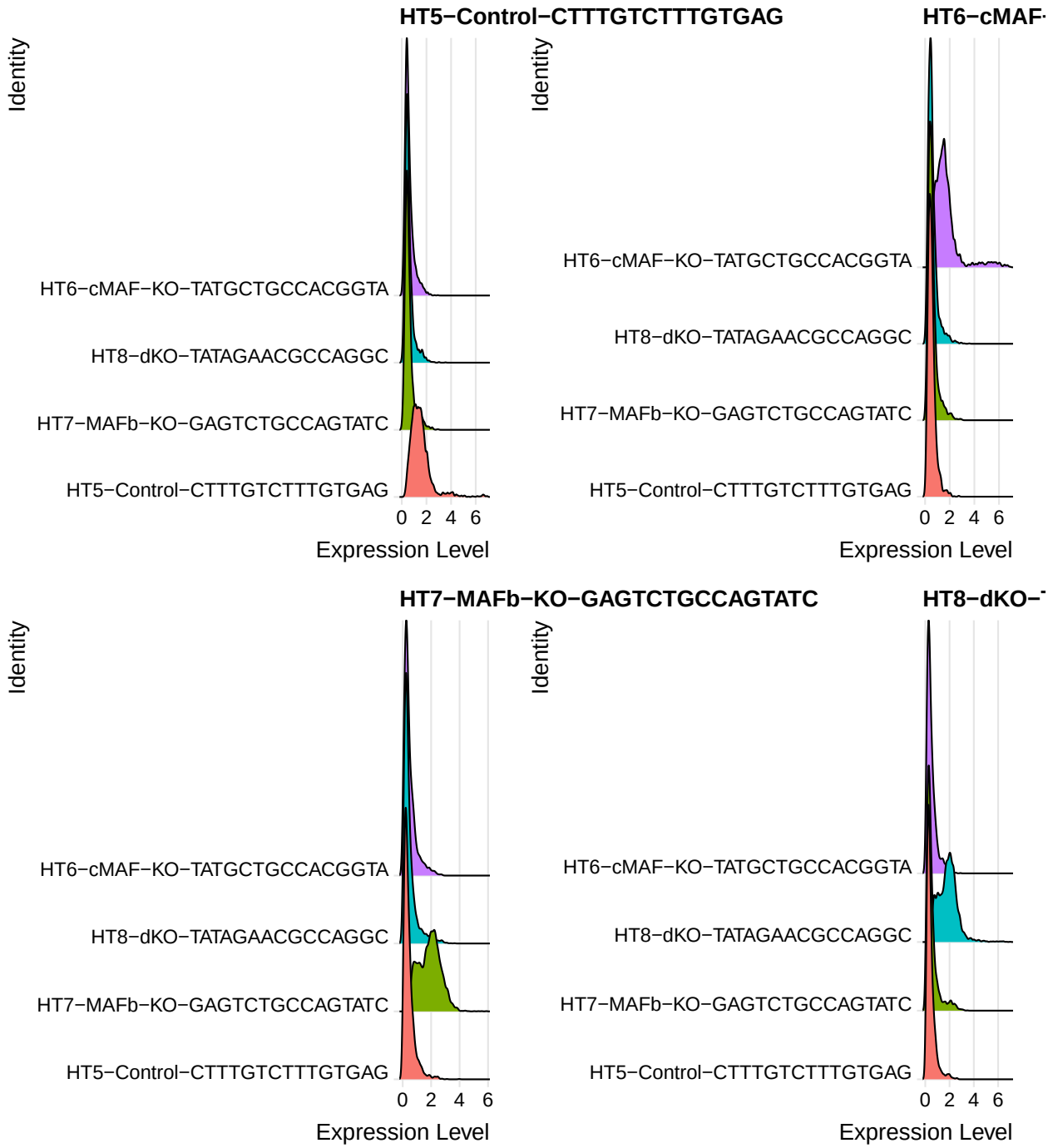
Output from running `HTODemux()` is saved in the object metadata. We can visualize how many cells are classified as singlets, doublets and negative/ambiguous cells.

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

```
##
##  Doublet Negative  Singlet
##     2094     2591     6653
```

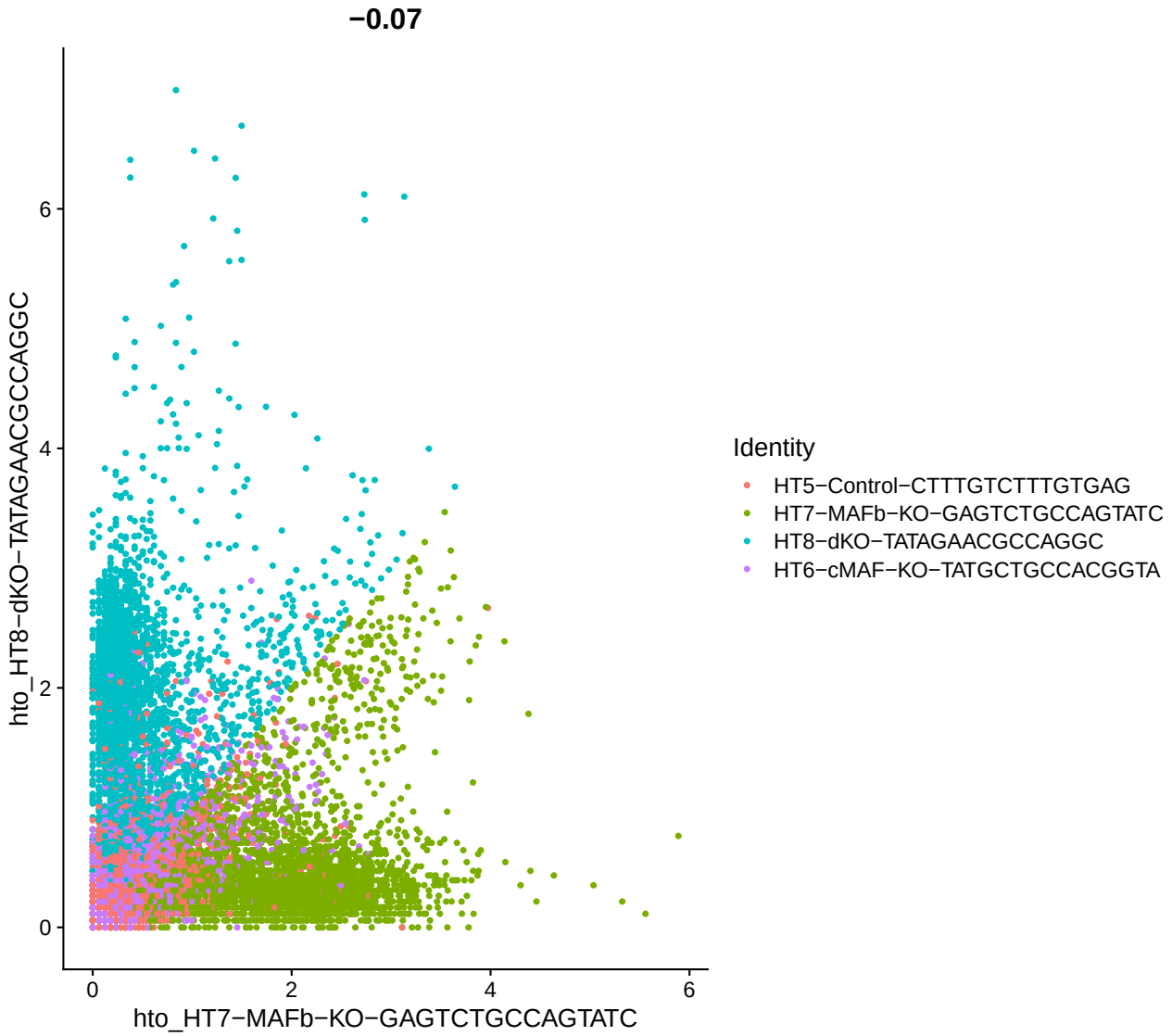
Visualize enrichment for selected HTOs with ridge plots

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

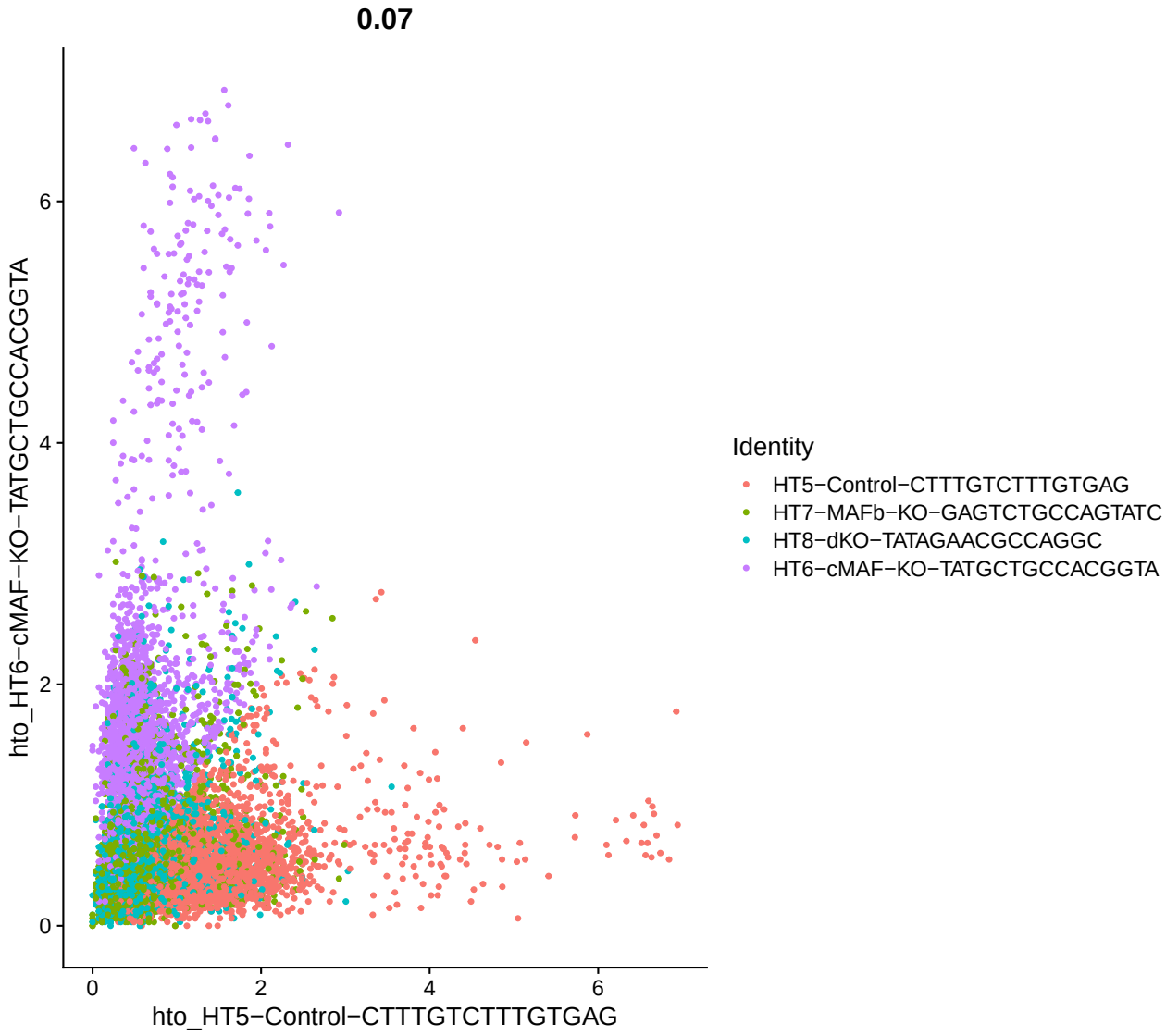


Visualize pairs of HTO signals to confirm mutual exclusivity in singlets

```
FeatureScatter(hashtag, feature1 = 'HT7-MAFb-KO-GAGTCTGCCAGTATC', feature2 = 'HT8-dKO-TATAGAACGCCAGGC')
```

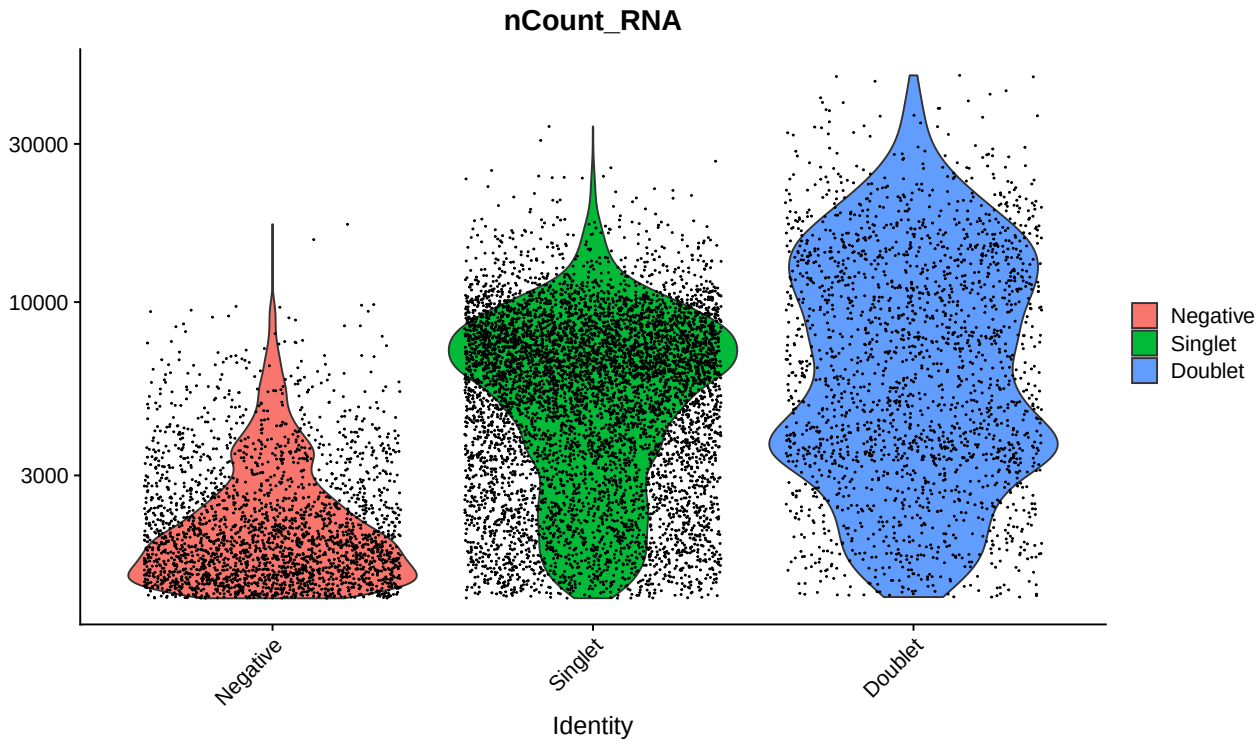



```
FeatureScatter(hashtag, feature1 = 'HT5-Control-CTTTGTCTTTGTGAG', feature2
    = 'HT6-cMAF-KO-TATGCTGCCACGGTA')
```



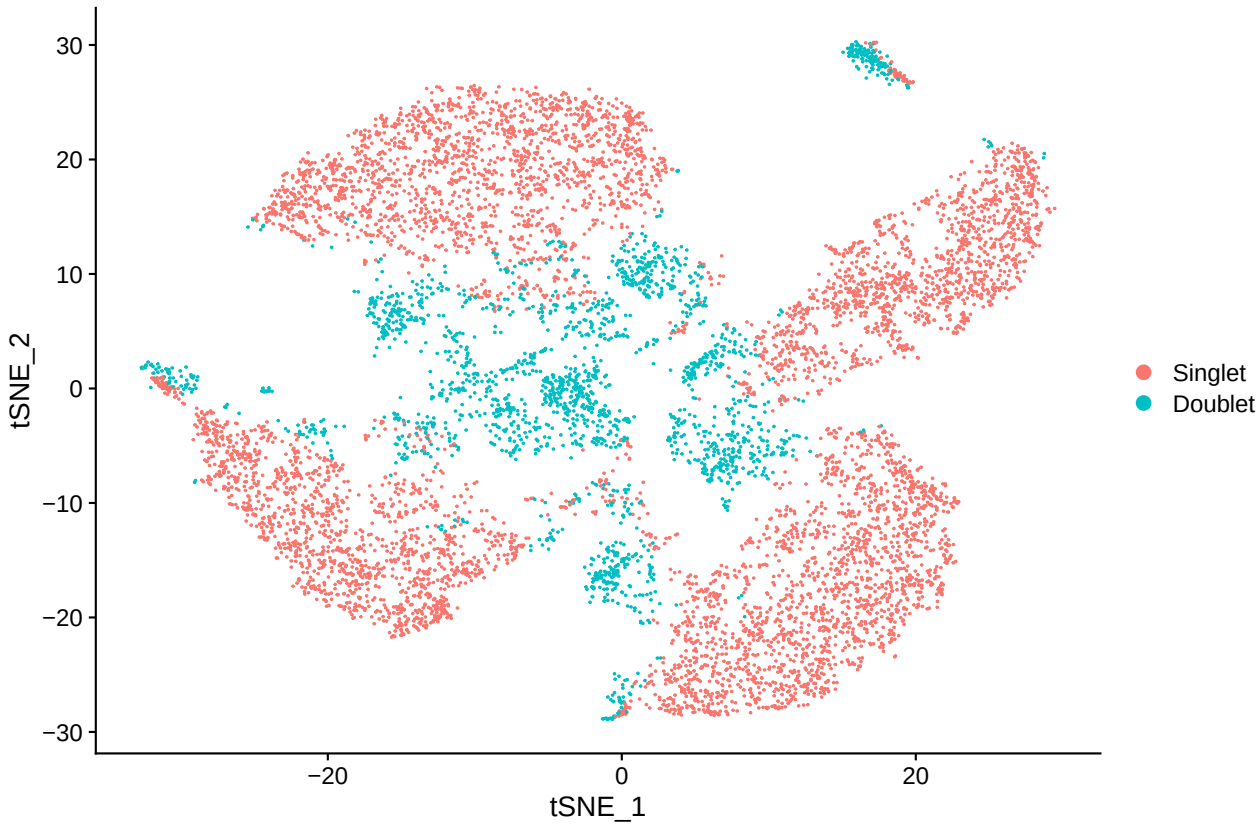
Compare number of UMIs for singlets, doublets and negative cells

```
Idents(hashtag) <- 'HTO_classification.global'
VlnPlot(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
hashtag.subset <- subset(hashtag, idents = 'Negative', invert = TRUE)

# Calculate a tSNE embedding of the HTO data
DefaultAssay(hashtag.subset) <- "HTO"
hashtag.subset <- ScaleData(hashtag.subset, features = rownames(hashtag.
    subset), verbose = FALSE)
hashtag.subset <- RunPCA(hashtag.subset, features = rownames(hashtag.
    subset), approx = FALSE)
hashtag.subset <- RunTSNE(hashtag.subset, dims = 1:8, perplexity = 100,
    check_duplicates = FALSE)

DimPlot(hashtag.subset)
```



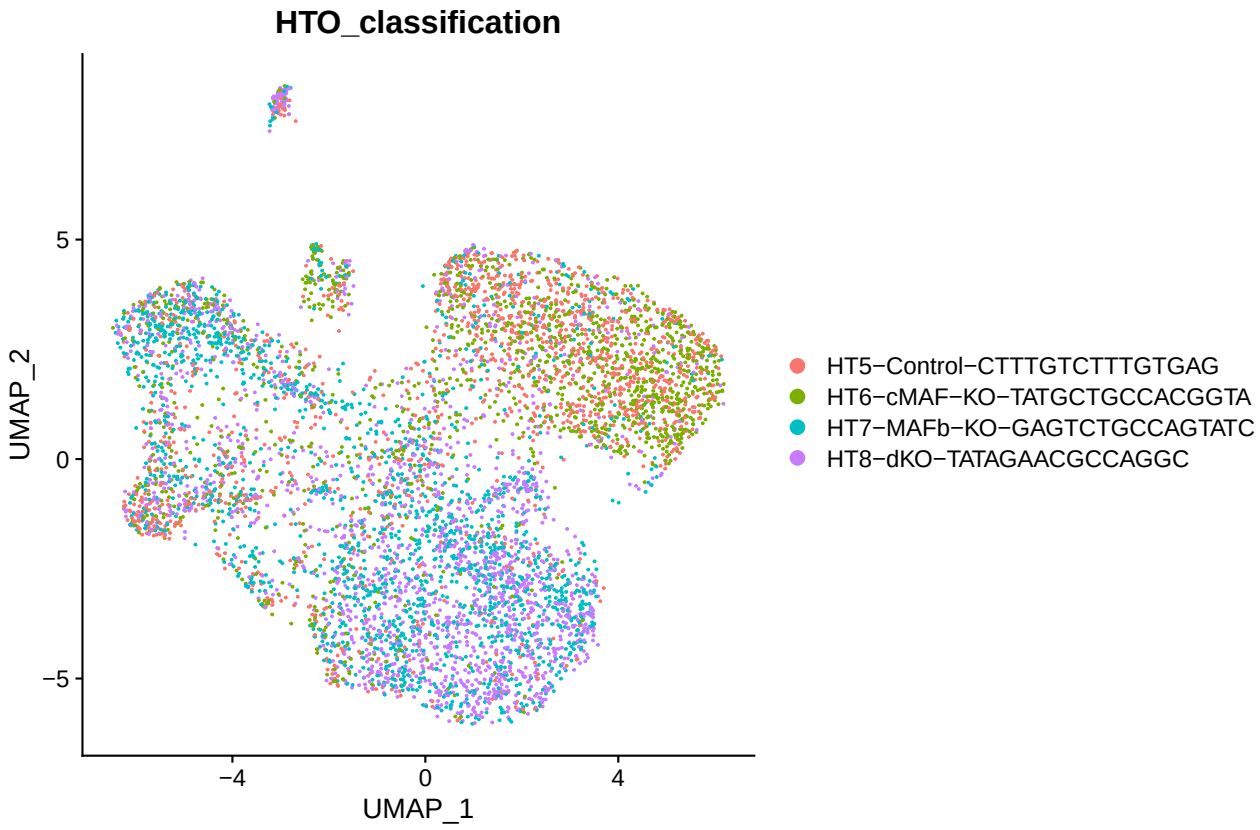
Create an HTO heatmap, based on Figure 1C in the Cell Hashing paper.

```
#To increase the efficiency of plotting, you can subsample cells using the
    num.cells argument
HTOHeatmap(hashtag, assay = 'HTO', ncells = 1000)
```

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

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

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

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

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

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

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



save to seurat object:

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

3 scRNAseq initiation - QC

Given the low apoptotic rate in the sample, we consider cells with >10% mt genes as apoptotic cells and filter them out from the further analyses.

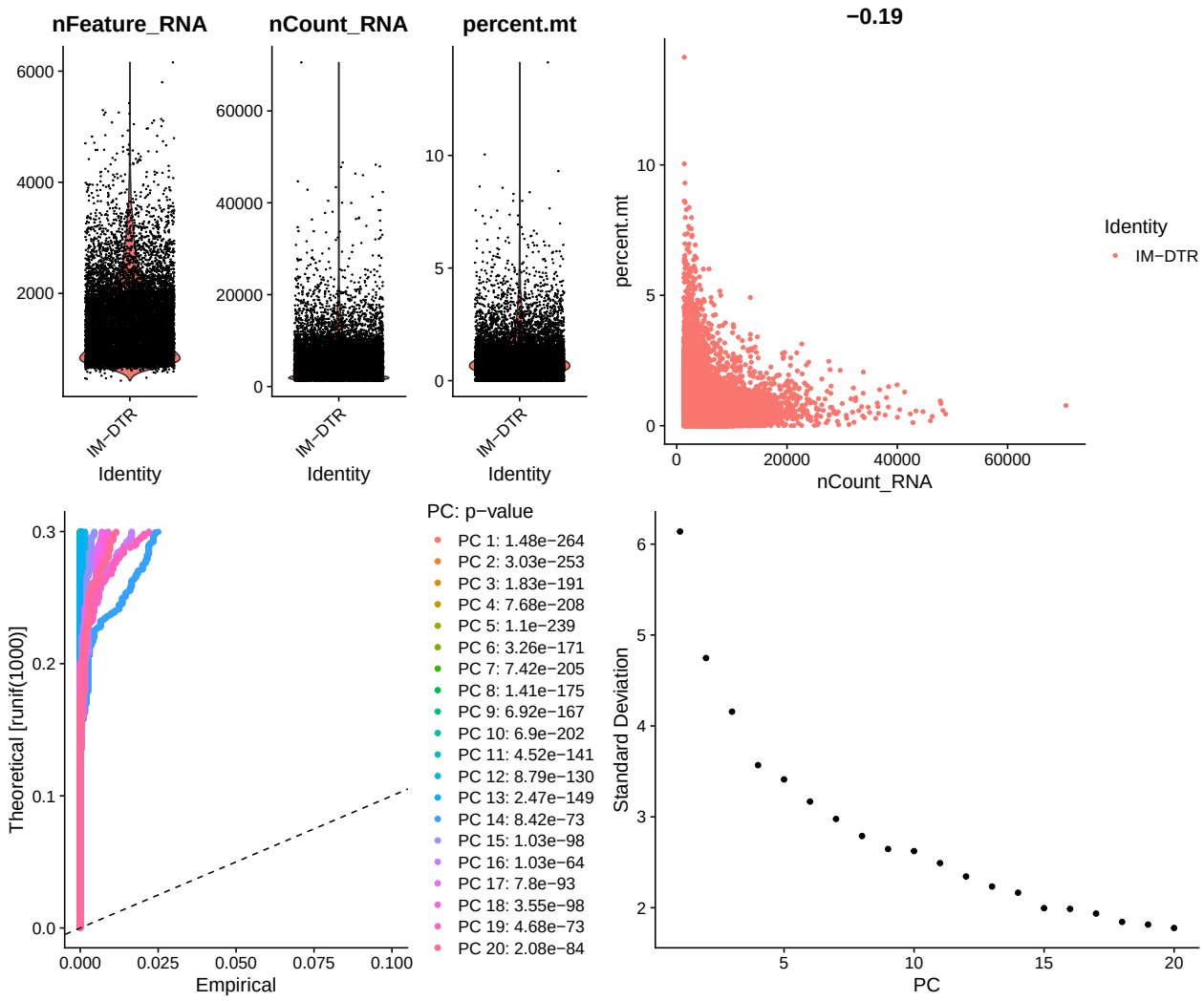
```
# load package and data
library(Seurat)
source("../R/seurat.setup.R")
mt.percentage <- 10
dimensionality <- 1:20
```

3.1 QC

```
IM_Maf <- seurat.setup(path.10x = "/mnt/Data/Single-cell_Analysis/Projects
   /IPL/IM_DTR/IM-DTR_MAF/counts/scRNAseq/Experiment-7-12-21-ScRNA_NGS21-
   U976/outs/filtered_feature_bc_matrix/", project = "IM-DTR",
   dimensionality = dimensionality, mt.percentage = mt.percentage, human =
    FALSE)
```

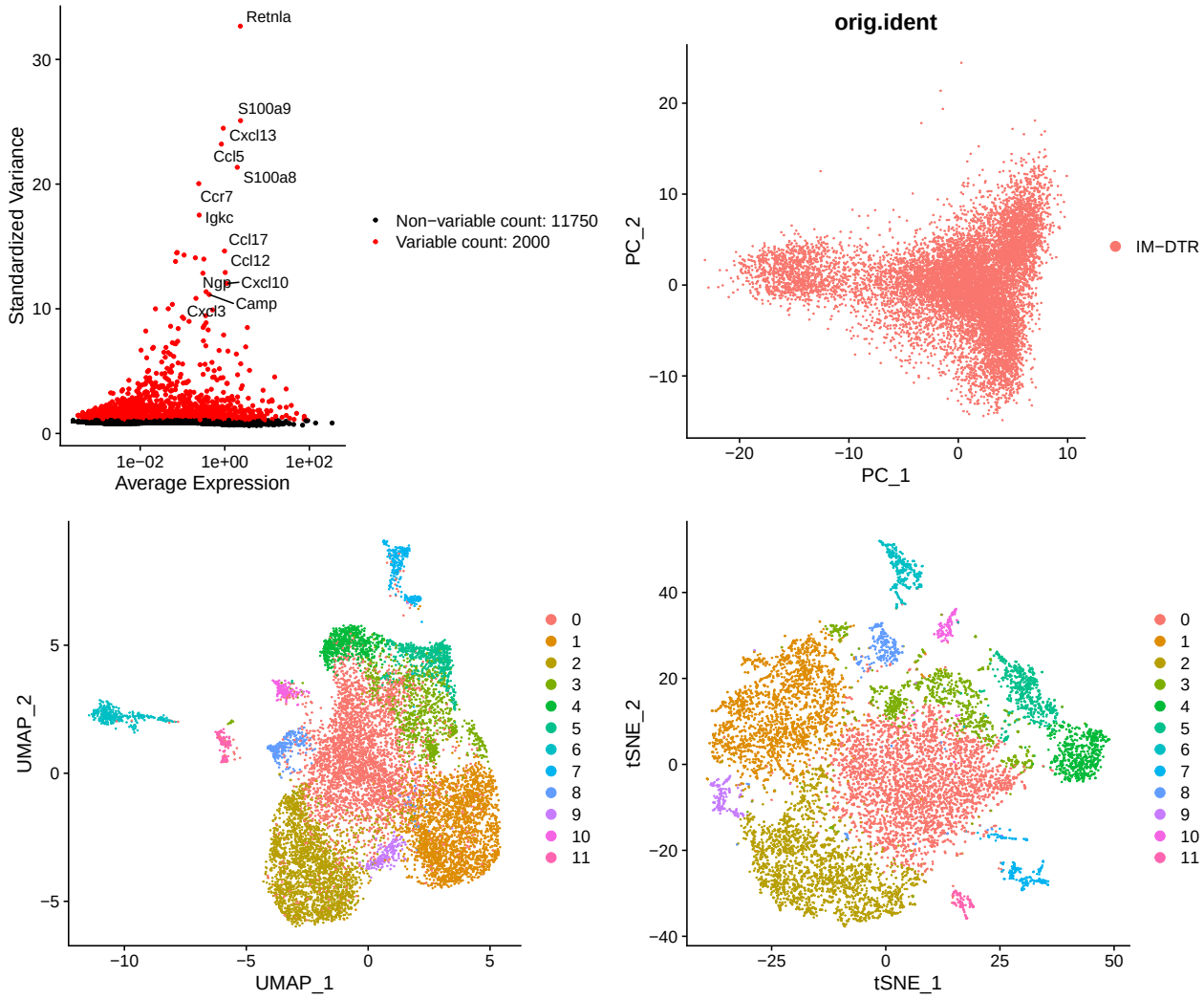
```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 11452
## Number of edges: 367228
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8751
## Number of communities: 12
## Elapsed time: 1 seconds
```

```
ggarrange(IM_Maf$plots$feature_vln, IM_Maf$plots$RNA_mt.pct_scatter, IM_
   Maf$plots$JackStrawPlot, IM_Maf$plots$ElbowPlot, ncol = 2, nrow = 2)
```




```
ggarrange(IM_Maf$plots$variable_features, IM_Maf$plots$PCA_plot, IM_Maf$
    plots$UMAP_plot, IM_Maf$plots$TSNE_plot, ncol = 2, nrow = 2)
```

1



4 Annotate cells by hashtags

4.1 Load Chromium/HTO data

```
IM_Maf.seuratObject <- IM_Maf$seuratObject
demultiplexed.seuratObject <- singlet
```

Assign HTO annotation to cells

```
common <- intersect(colnames(IM_Maf.seuratObject), colnames(demultiplexed.
    seuratObject))

IM_Maf.seuratObject <- subset(IM_Maf.seuratObject, cells = common)
```

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

# remove the sequence chars
hto <- sub(hto, pattern = "-[C,T,G,A]{0,15}$", replacement = "")
```

```
                                                                    7
# assign to seurat object:                                          8
IM_Maf.seuratObject$group <- hto[colnames(IM_Maf.seuratObject)]     9
```

4.2 Make metadata for samples

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

Save individual samples for other use

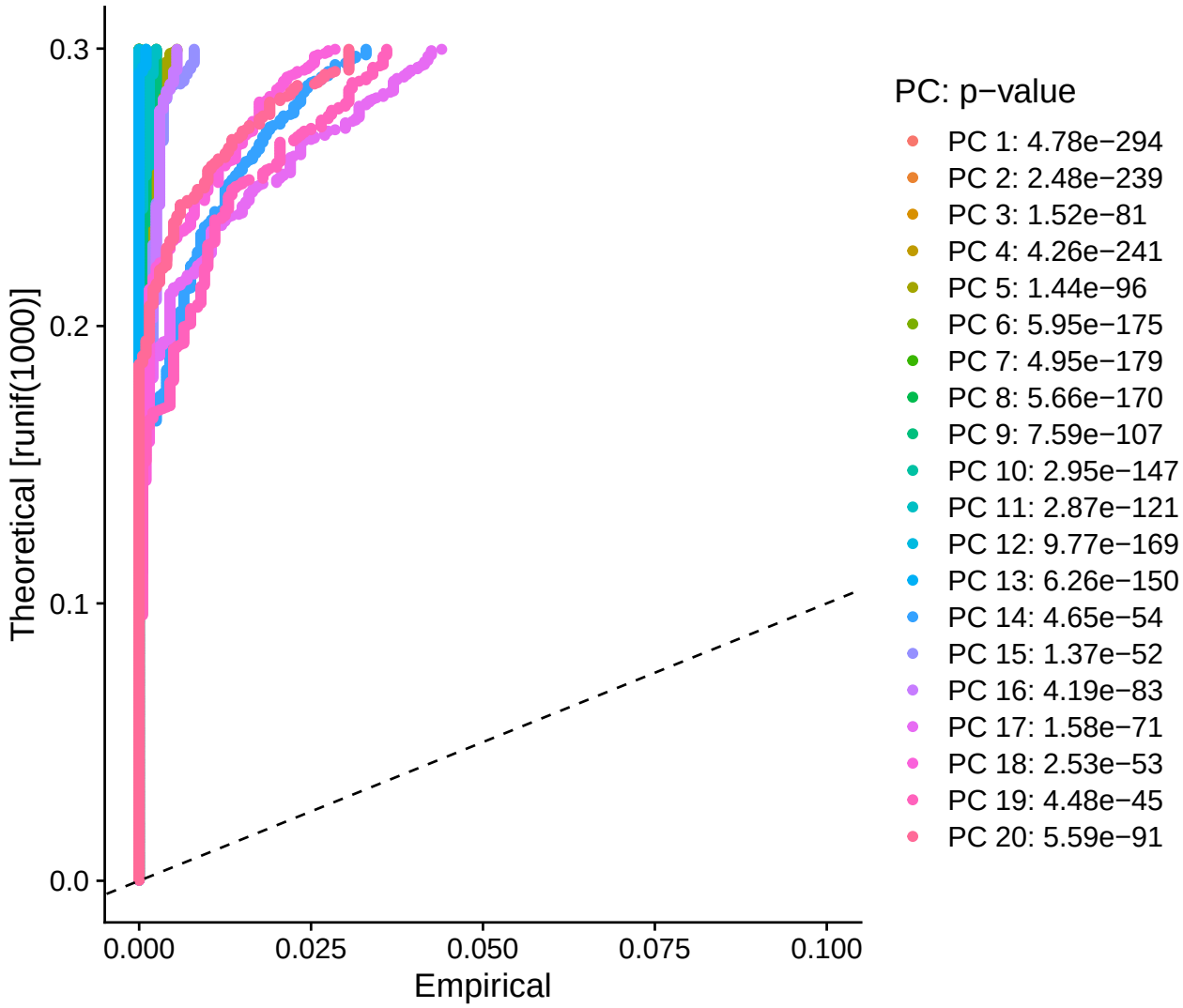
```
obj <- IM_Maf.seuratObject                                                  1
for (i in unique(obj$group)) {                                              2
  obj.sub <- subset(obj, subset = group == i)                               3
  file.name <- paste("IM_mono", i, "seuratObject.rds", sep = ".")           4
  saveRDS(object = obj.sub, file = file.path(".", file.name))               5
}                                                                           6
```

4.3 Data processing and cell clustering

```
IM_Maf.seuratObject <- NormalizeData(IM_Maf.seuratObject)                   1
IM_Maf.seuratObject <- FindVariableFeatures(IM_Maf.seuratObject, selection  2
   .method = "vst", nfeatures = 2000)
IM_Maf.seuratObject <- ScaleData(IM_Maf.seuratObject, features = rownames(  3
   IM_Maf.seuratObject))
```

Linear dimension reduction:

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

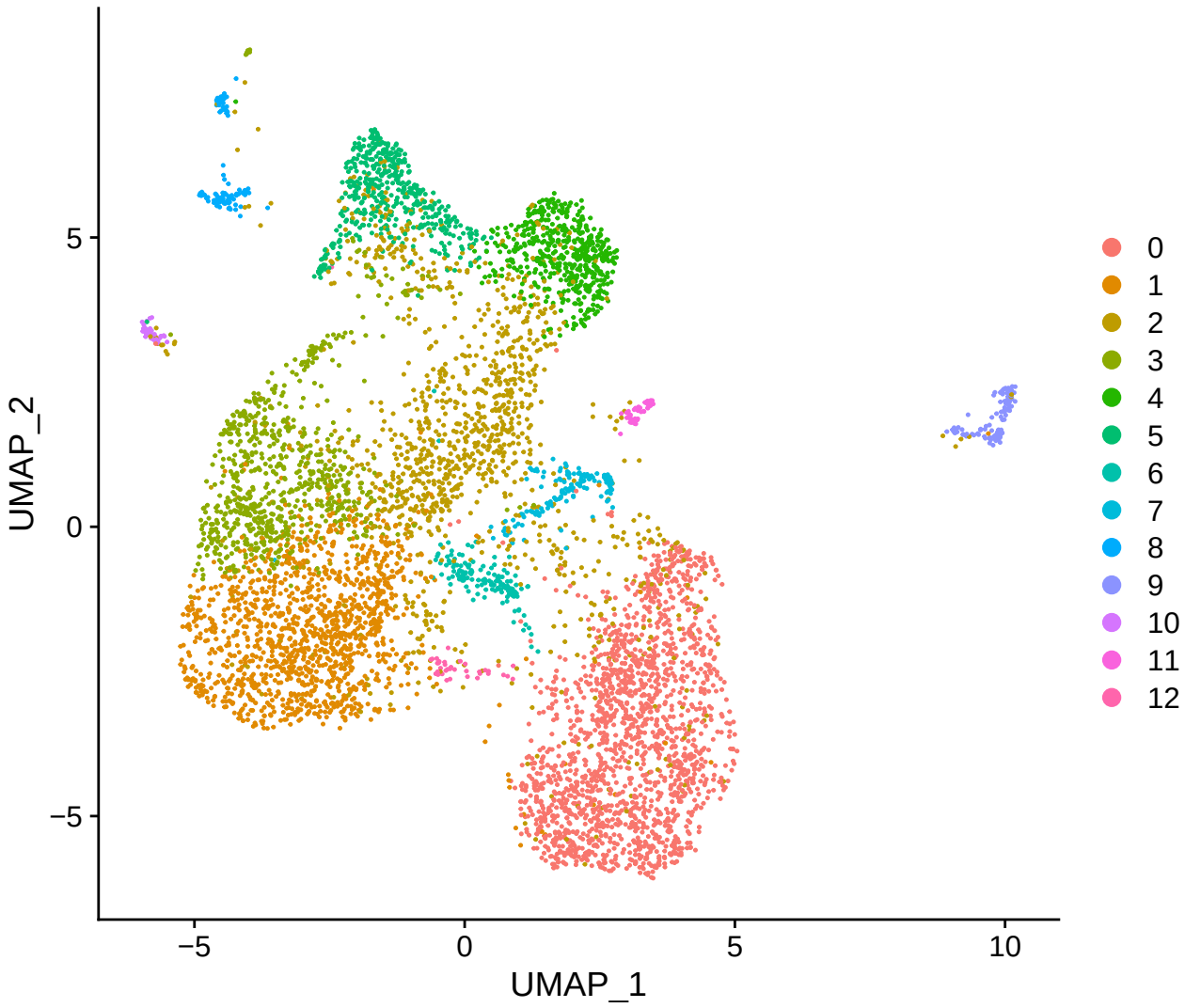



```
IM_Maf.seuratObject <- FindNeighbors(IM_Maf.seuratObject, dims = 1:30)
IM_Maf.seuratObject <- FindClusters(IM_Maf.seuratObject, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 6652
## Number of edges: 247502
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8791
## Number of communities: 13
## Elapsed time: 0 seconds
```

```
IM_Maf.seuratObject <- RunTSNE(IM_Maf.seuratObject, dims = 1:30)
IM_Maf.seuratObject <- RunUMAP(IM_Maf.seuratObject, dims = 1:30)
```

```
DimPlot(IM_Maf.seuratObject)
```



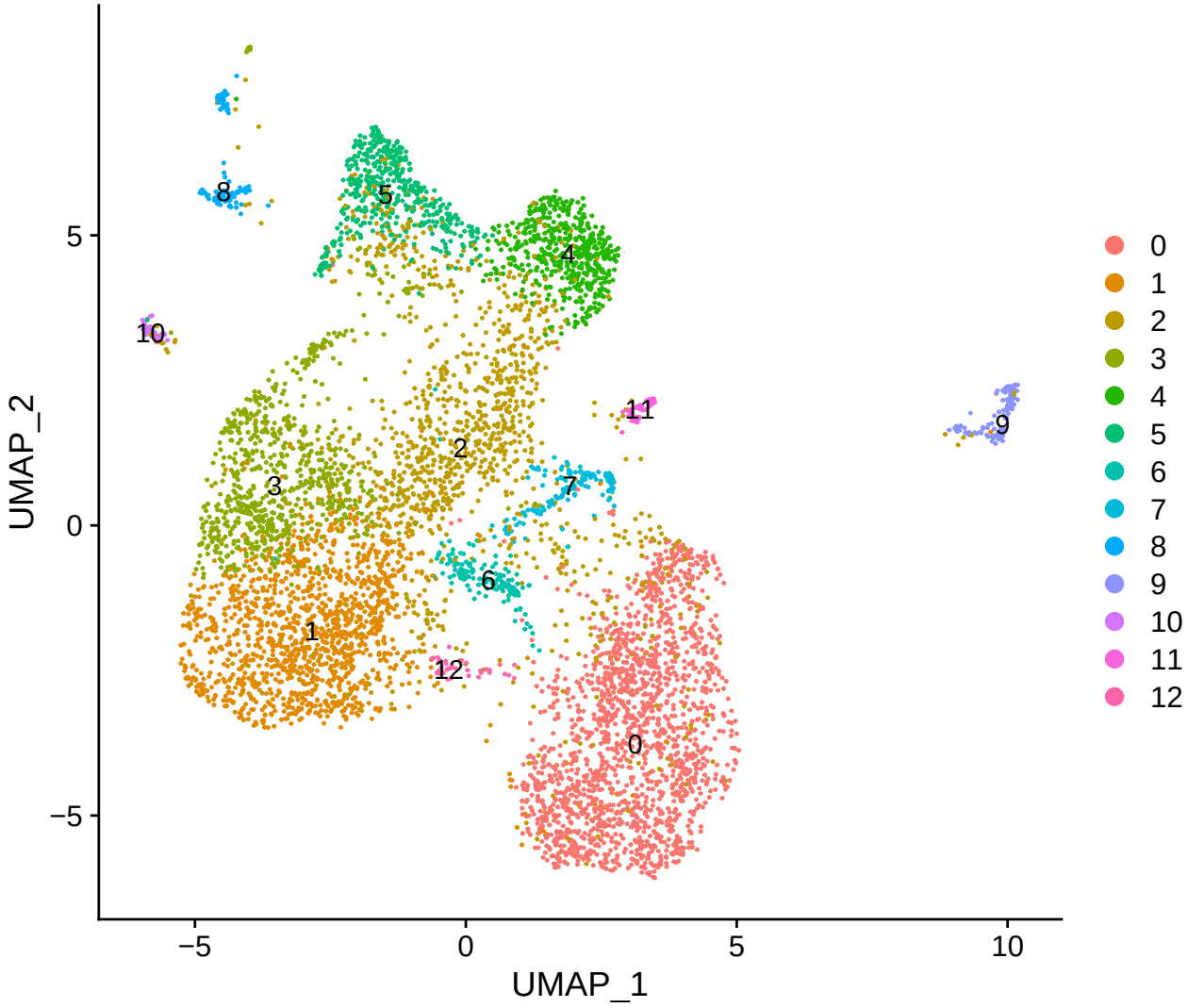
4.4 Celltyping

```
source("../R/seurat2singleR.R")

results.singleR <- seurat2singleR(IM_Maf.seuratObject, ref = "ImmGenData")

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

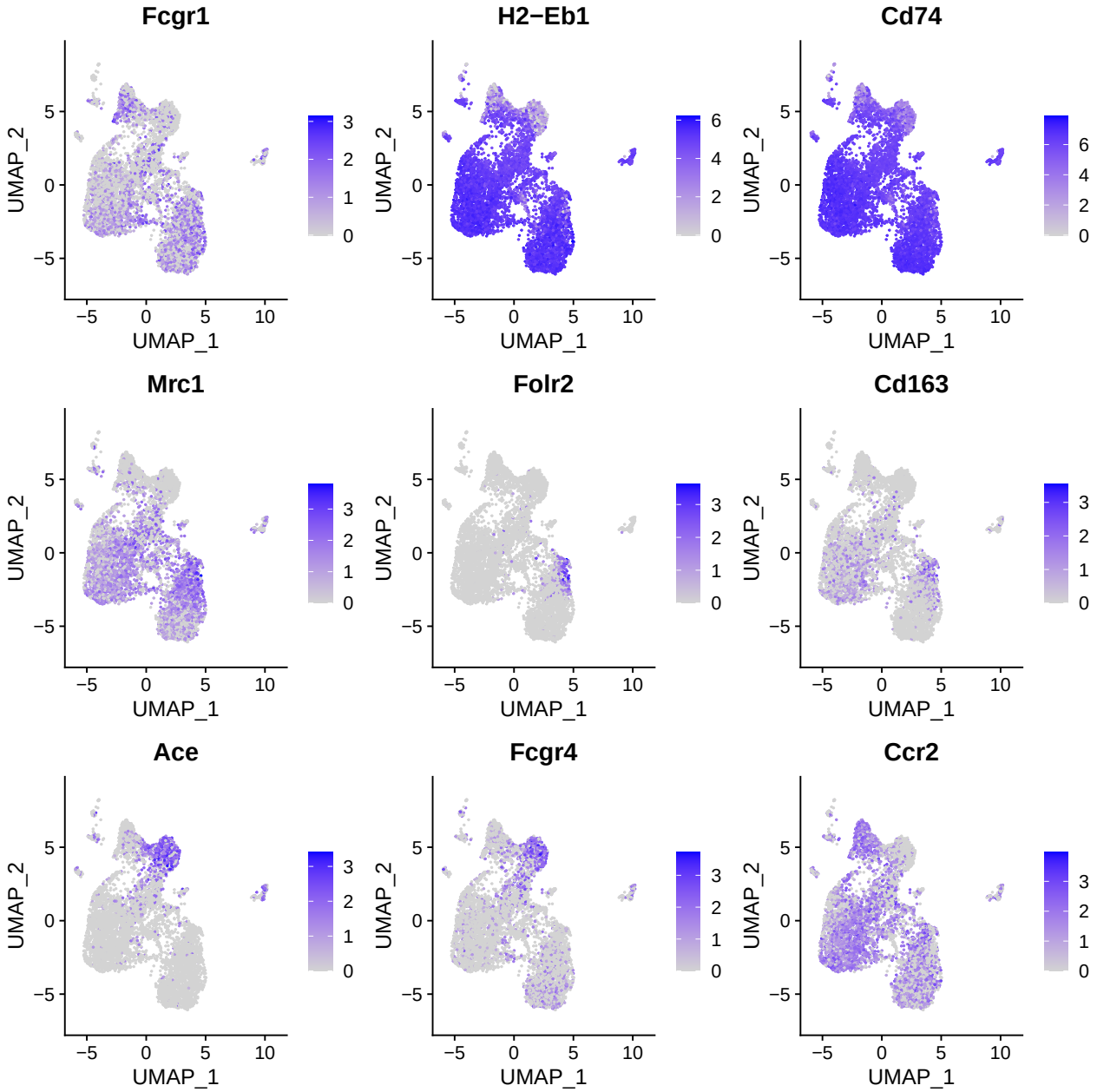
5 Remove contaminated cell types

5.1 Plot with key markers

```
library(ggplot2)
library(dplyr)
results <- IM_Maf.seuratObject
DimPlot(results, reduction = "umap", label = TRUE)
```




```
DefaultAssay(results) <- "RNA"
FeaturePlot(results, features = c("Fcgr1", "H2-Eb1", "Cd74", "Mrc1", "
   Folr2", "Cd163", "Ace", "Fcgr4", "Ccr2"), reduction = "umap")
```



5.2 Use results of SingleR celltyping to annotate cells

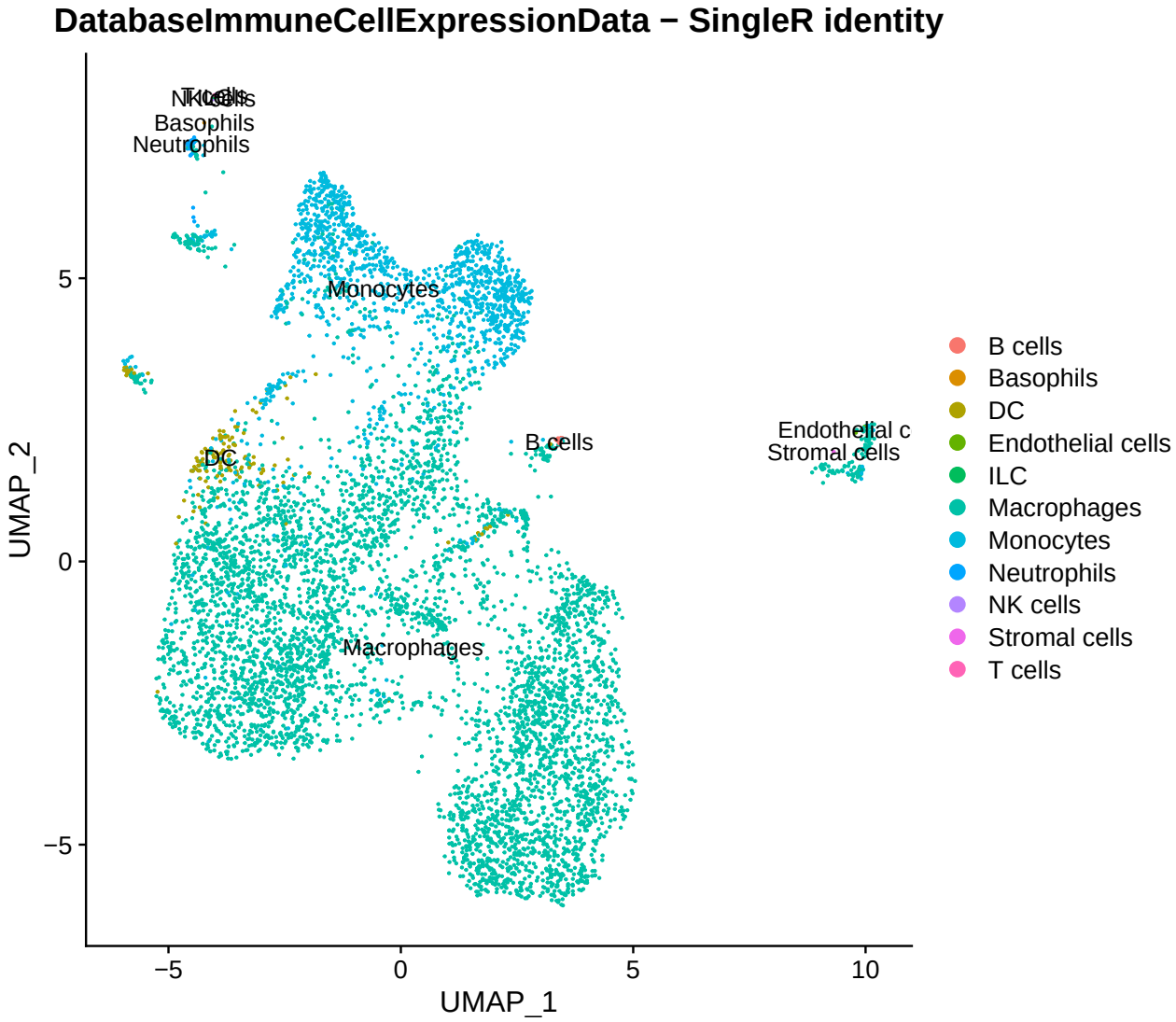
Check if SingleR results contain the same cells:

```
identical(colnames(results), rownames(results.singleR))
```

```
## [1] TRUE
```

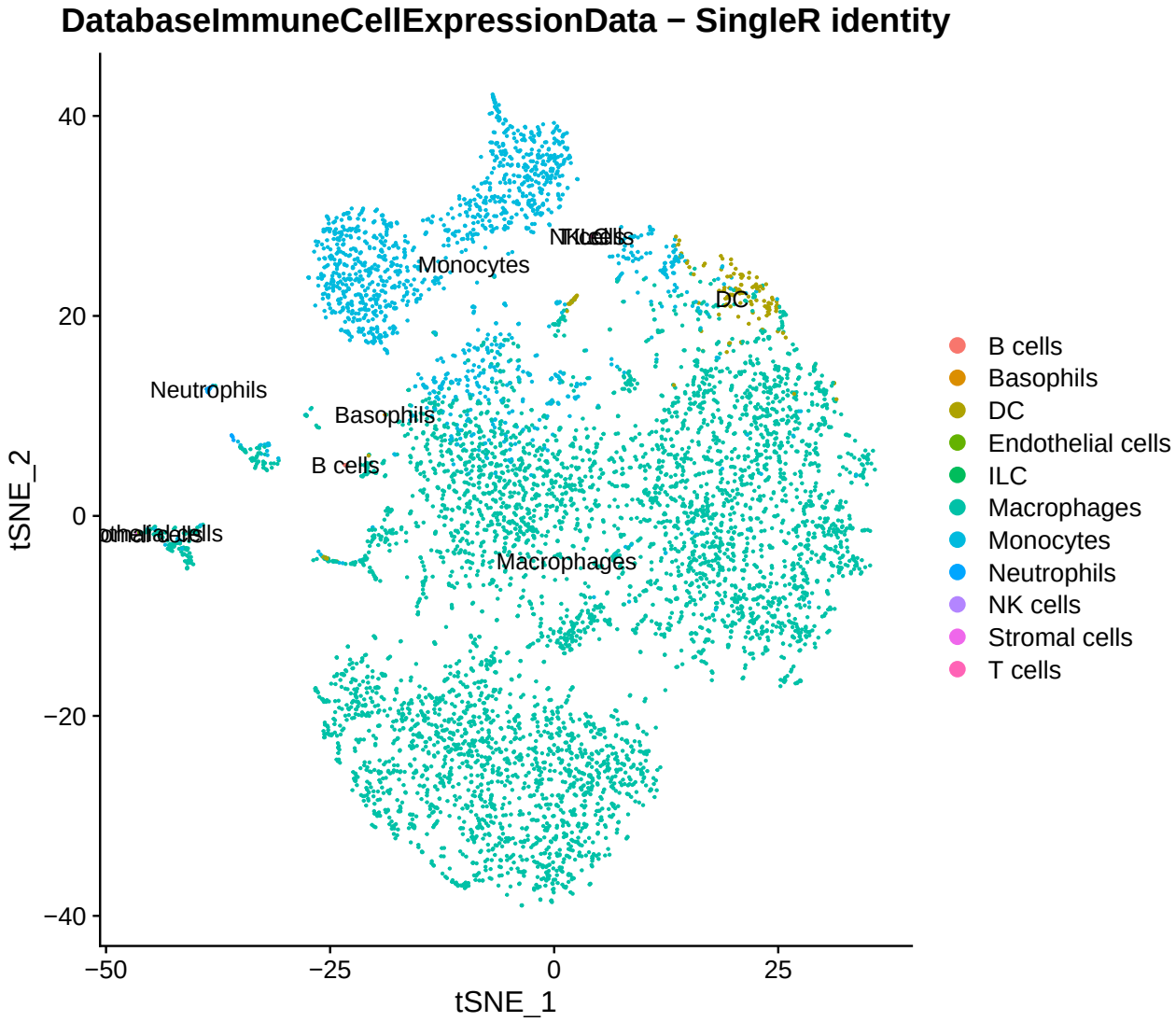
UMAP plot show cell types:

```
results$singleR.celltype <- results.singleR$labels

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

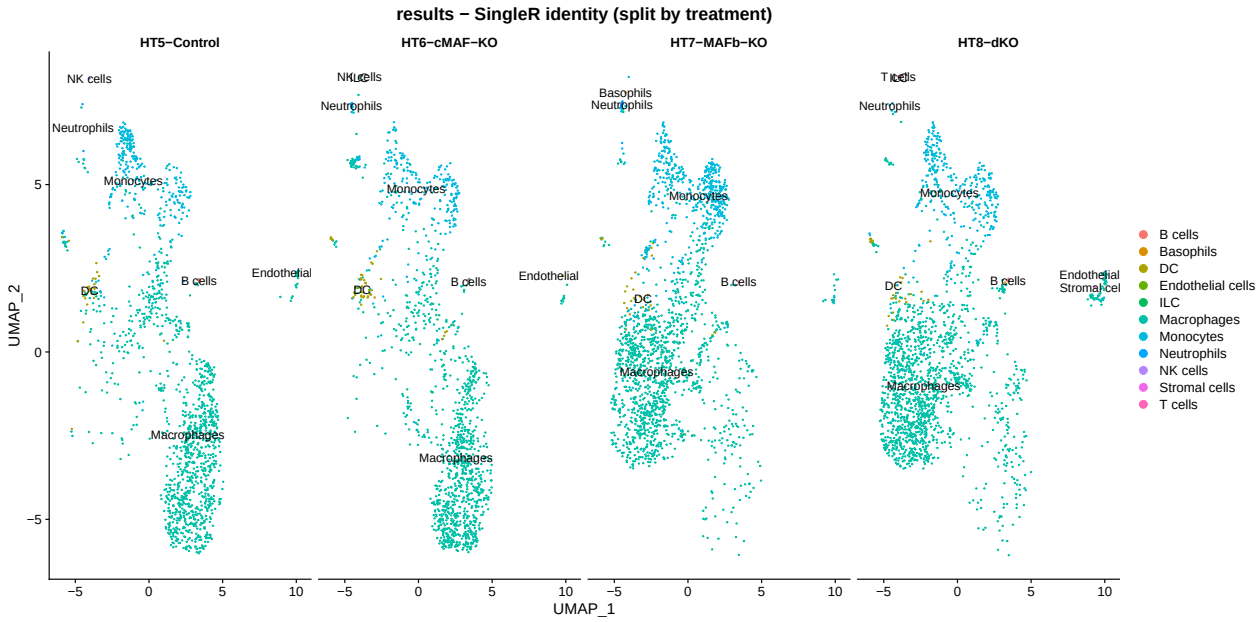


TSNE plot show cell types:

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




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



5.3 Remove contamination

Here's all the cell types and number:

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

```
##
##          B cells         Basophils                DC Endothelial cells
##               13                 1               145                   3
##              ILC       Macrophages         Monocytes        Neutrophils
##                2              5049              1403                  30
##          NK cells     Stromal cells           T cells
##                4                 1                 1
```

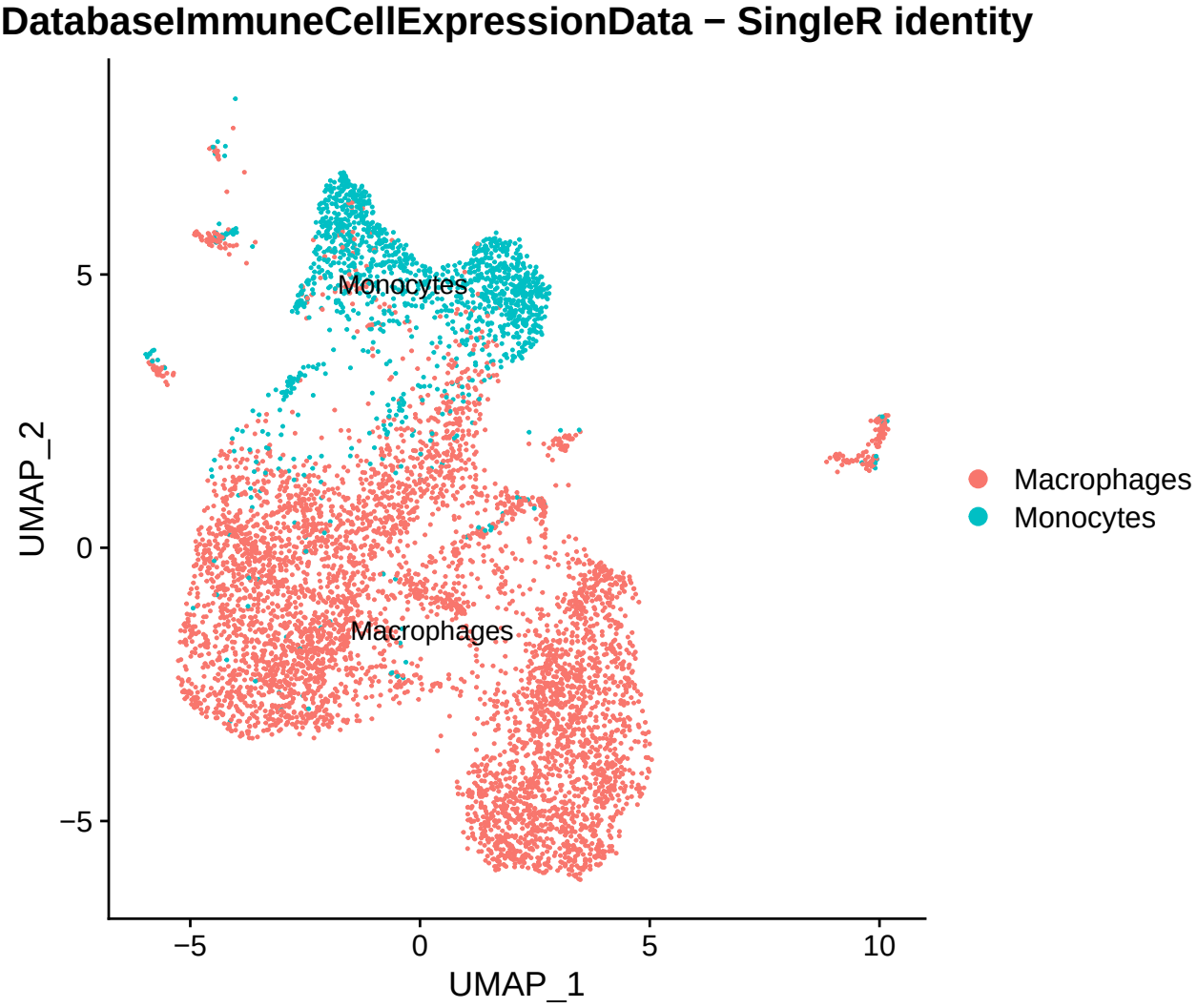
Remove all the other cell types and keep only monocytes and macrophages.

```
results <- subset(results, subset = singleR.celltype %in% c("B cells", "
    Basophils", "Endothelial cells", "Stromal cells", "Neutrophils", "NK
    cells", "DC", "ILC", "T cells"), invert = TRUE) # DC removed!!!
```

```
table(results$group)
```

```
##
## HT5-Control HT6-cMAF-KO HT7-MAFb-KO     HT8-dKO
##        1429        1340        1815        1868
```

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



5.4 Reanalysis data after contamination removal

Remove old snn:

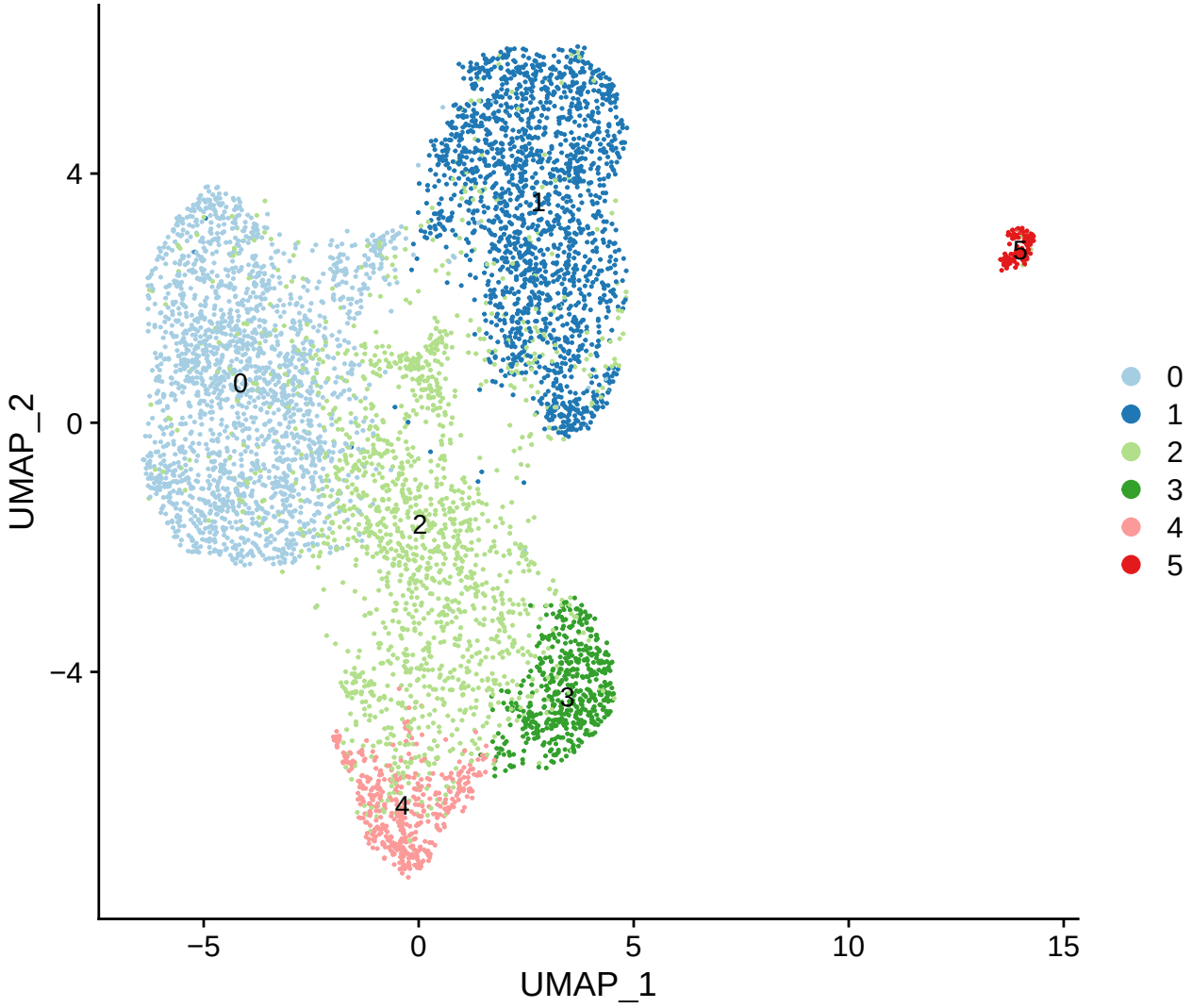
```
meta.names <- names(results@meta.data)
old.snn <- meta.names[startsWith(meta.names, "RNA_snn")]
for (i in old.snn) {
  results[[i]] <- NULL
}
```

```
# DefaultAssay(results) <- "RNA"
results <- NormalizeData(results, verbose=FALSE)
results <- FindVariableFeatures(results, selection.method = "vst",
    nfeatures = 2000, verbose=FALSE)
results <- ScaleData(results, features = rownames(results), verbose=FALSE)
results <- RunPCA(results, features = VariableFeatures(results), verbose=
    FALSE)
results <- RunTSNE(results, dims = 1:10, verbose=FALSE)
results <- RunUMAP(results, dims = 1:10, verbose=FALSE)
```

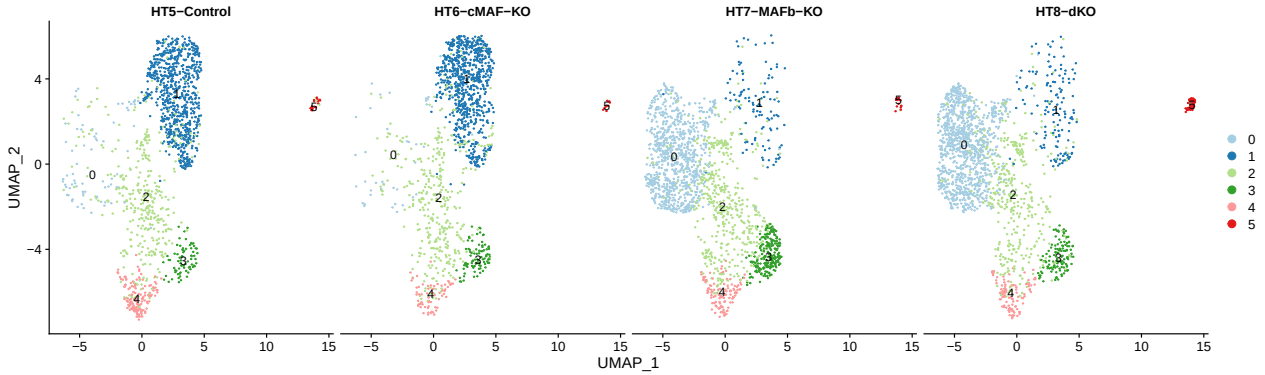
5.5 Cell clustering

```
results <- FindNeighbors(results, dims = 1:10, verbose = FALSE)
results <- FindClusters(results, resolution = 0.12, verbose = FALSE)
```

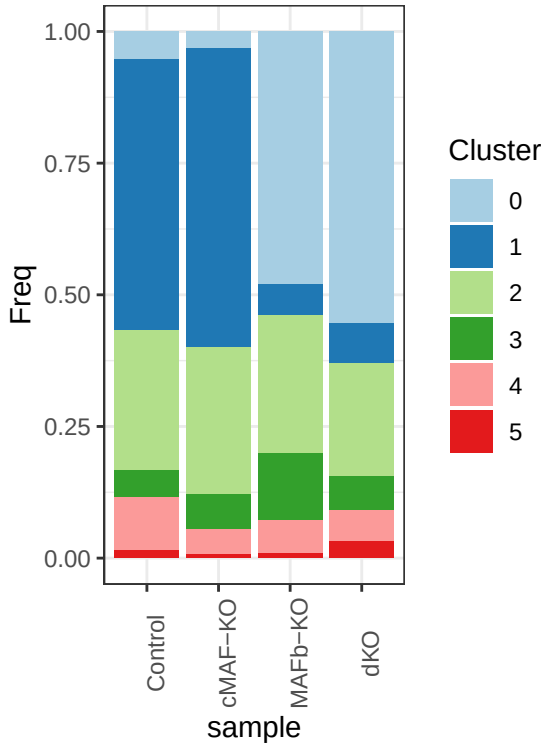
```
DimPlot(results, label = TRUE, cols = "Paired", reduction = "umap")
```




```
DimPlot(results, label = TRUE, split.by = "group", cols = "Paired")
```



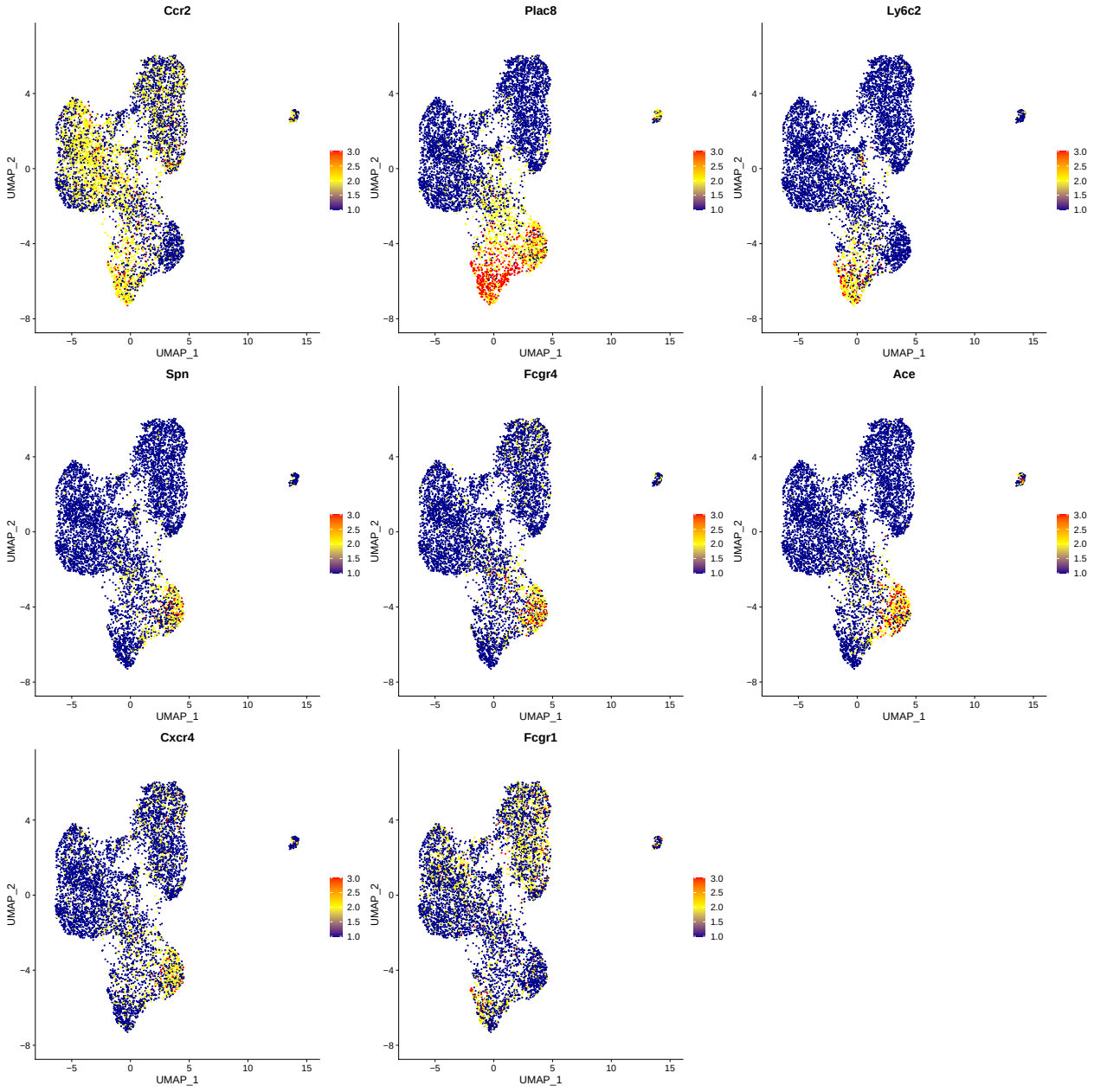

```
source("../R/SeuratFreqTable.R")
freq.celltype.list <- list(
  `Control` = Seurat2CellFreqTable(subset(results, subset = group == "HT5-
     Control"), slotName = "RNA_snn_res.0.12"),
  `cMAF-KO` = Seurat2CellFreqTable(subset(results, subset = group == "HT6-
     cMAF-KO"), slotName = "RNA_snn_res.0.12"),
  `MAFb-KO` = Seurat2CellFreqTable(subset(results, subset = group == "HT7-
     MAFb-KO"), slotName = "RNA_snn_res.0.12"),
  `dKO` = Seurat2CellFreqTable(subset(results, subset = group == "HT8-dKO"
     ), slotName = "RNA_snn_res.0.12")
)

source("../R/barChart.R")
barChart(freq.celltype.list) + labs(fill = "Cluster") + scale_fill_manual(
    values = brewer.pal(6, "Paired")) + theme(axis.text.x = element_text(
    angle = 90))
```



5.6 population characterization

5.6.1 Show expresssion of important monocyte markers

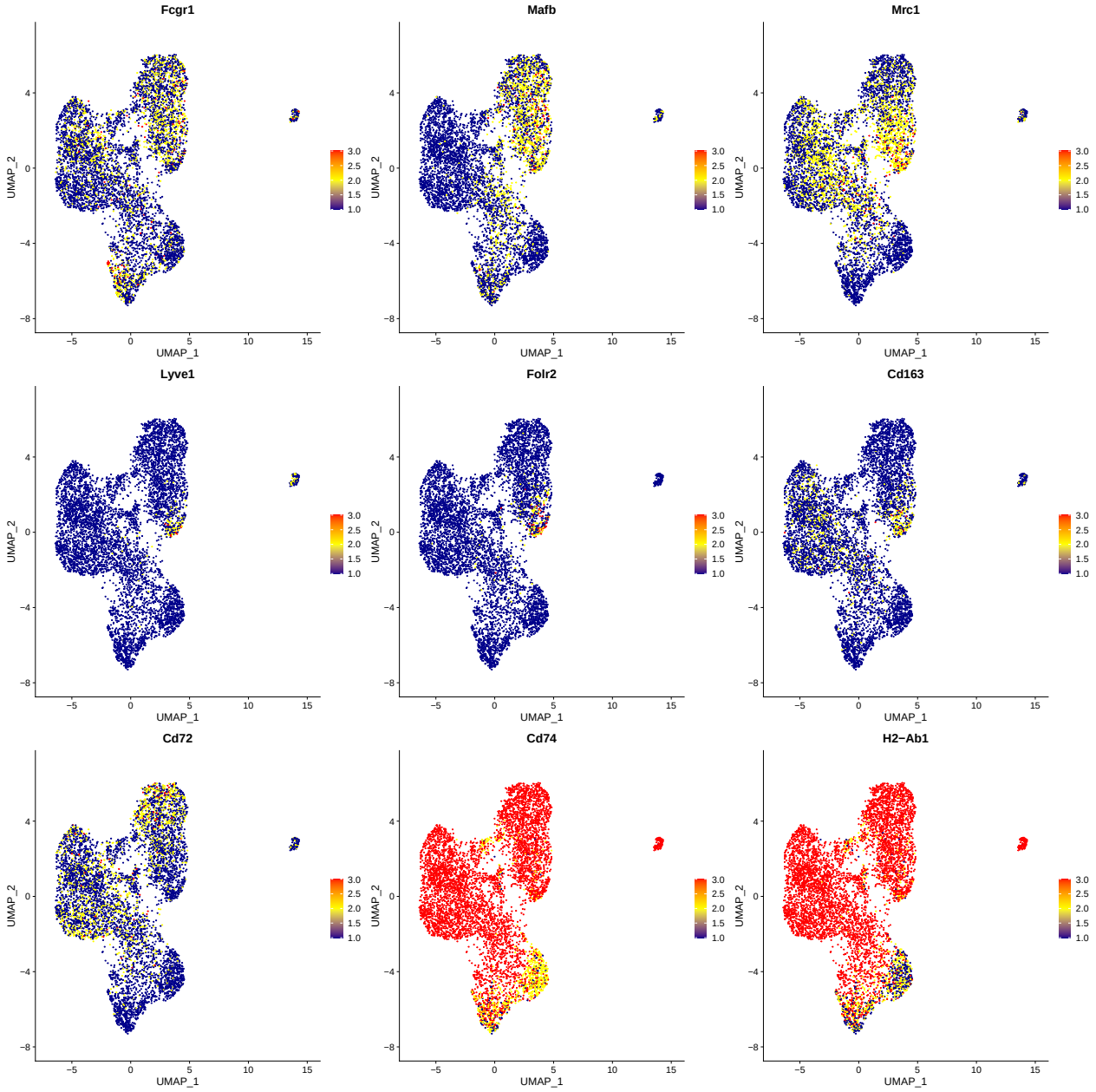
```
FeaturePlot(results, features = c("Ccr2", "Plac8" ,"Ly6c2", "Spn",
                                  "Fcgr4", "Ace", "Cxcr4",
                                  "Fcgr1") ,
            ncol = 3, reduction = "umap", cols = c("dark blue", "yellow", "
               red"))
```



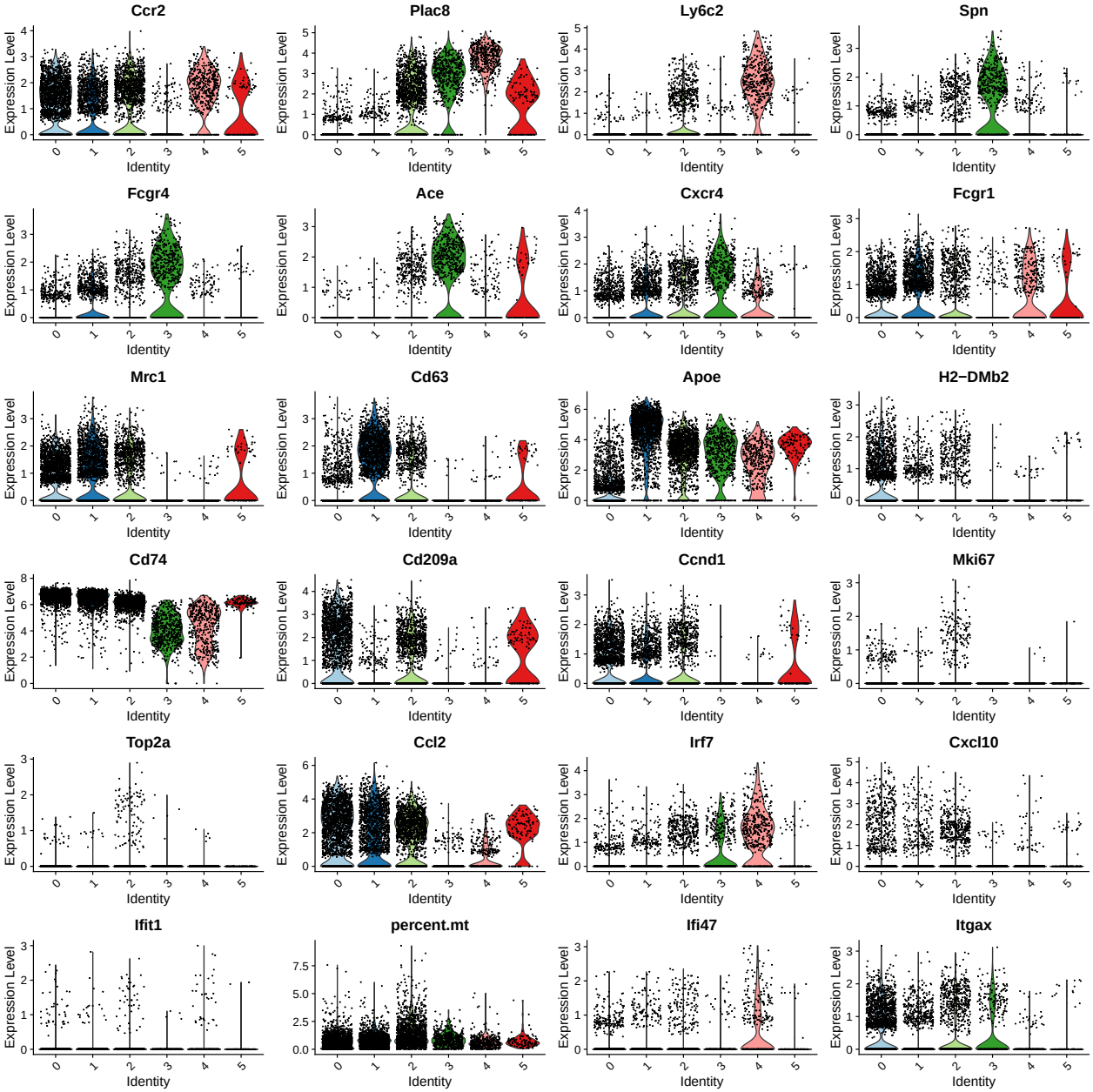
5.6.2 Show expresssion of important IM markers

```
FeaturePlot(results, features = c(
                                  "Fcgr1", "Mafb" ,"Mrc1", "Lyve1", "Folr2",
                                     "Cd163",
```

```
                                "Cd72", "Cd74", "H2-Ab1") ,
          ncol = 3, cols = c("dark blue", "yellow", "red"))
```



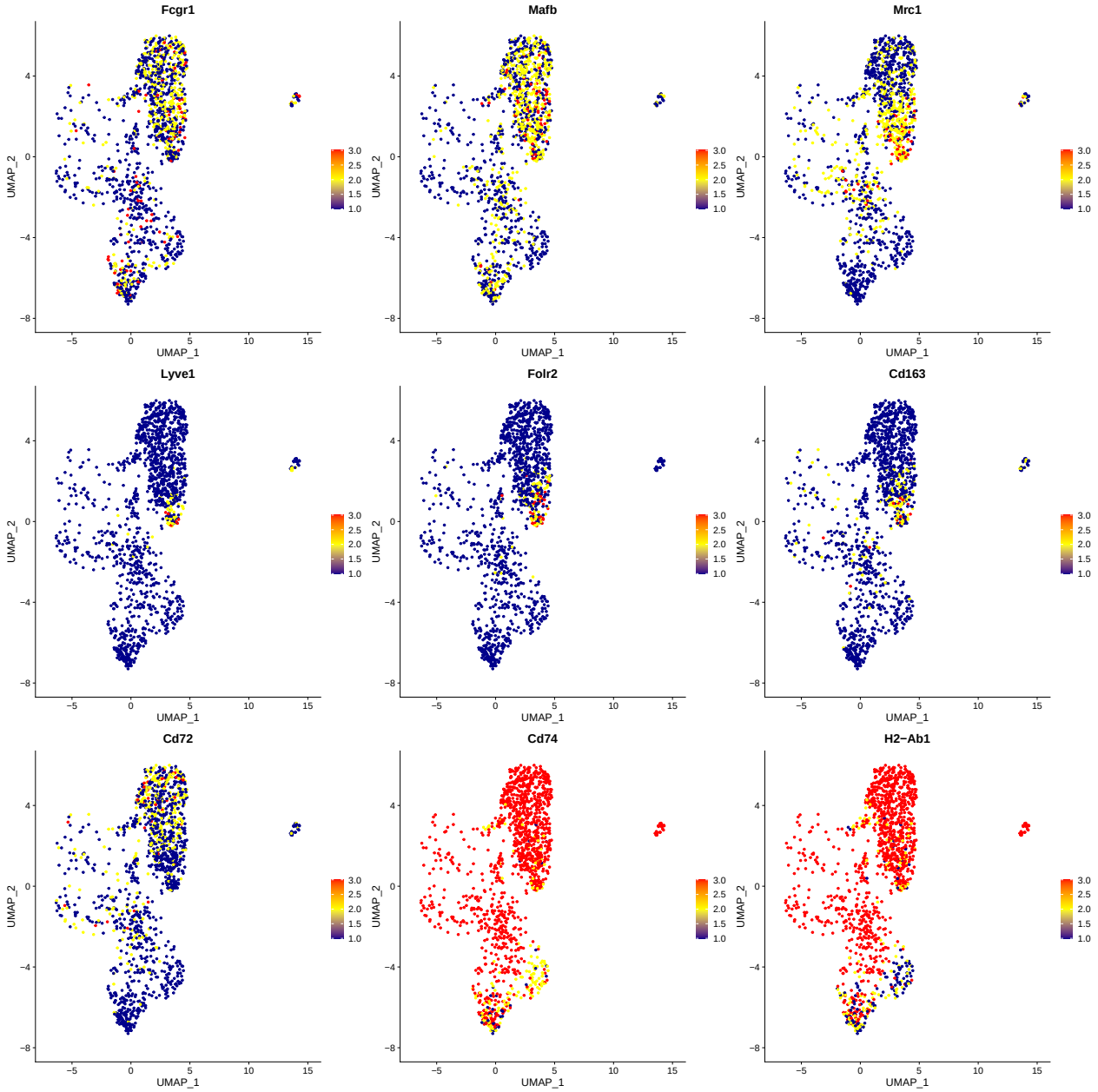

```
VlnPlot(results, features = c("Ccr2", "Plac8" ,"Ly6c2", "Spn",
                              "Fcgr4", "Ace", "Cxcr4",
                              "Fcgr1", "Mrc1", "Cd63", "Apoe",
                              "H2-DMb2", "Cd74", "Cd209a",
                              "Ccnd1","Mki67", "Top2a", "Ccl2",
                              "Irf7", "Cxcl10", "Ifit1", "percent.mt", "
                                 Ifi47", "Itgax") ,
        ncol = 4, cols = brewer.pal(6, "Paired"))
```



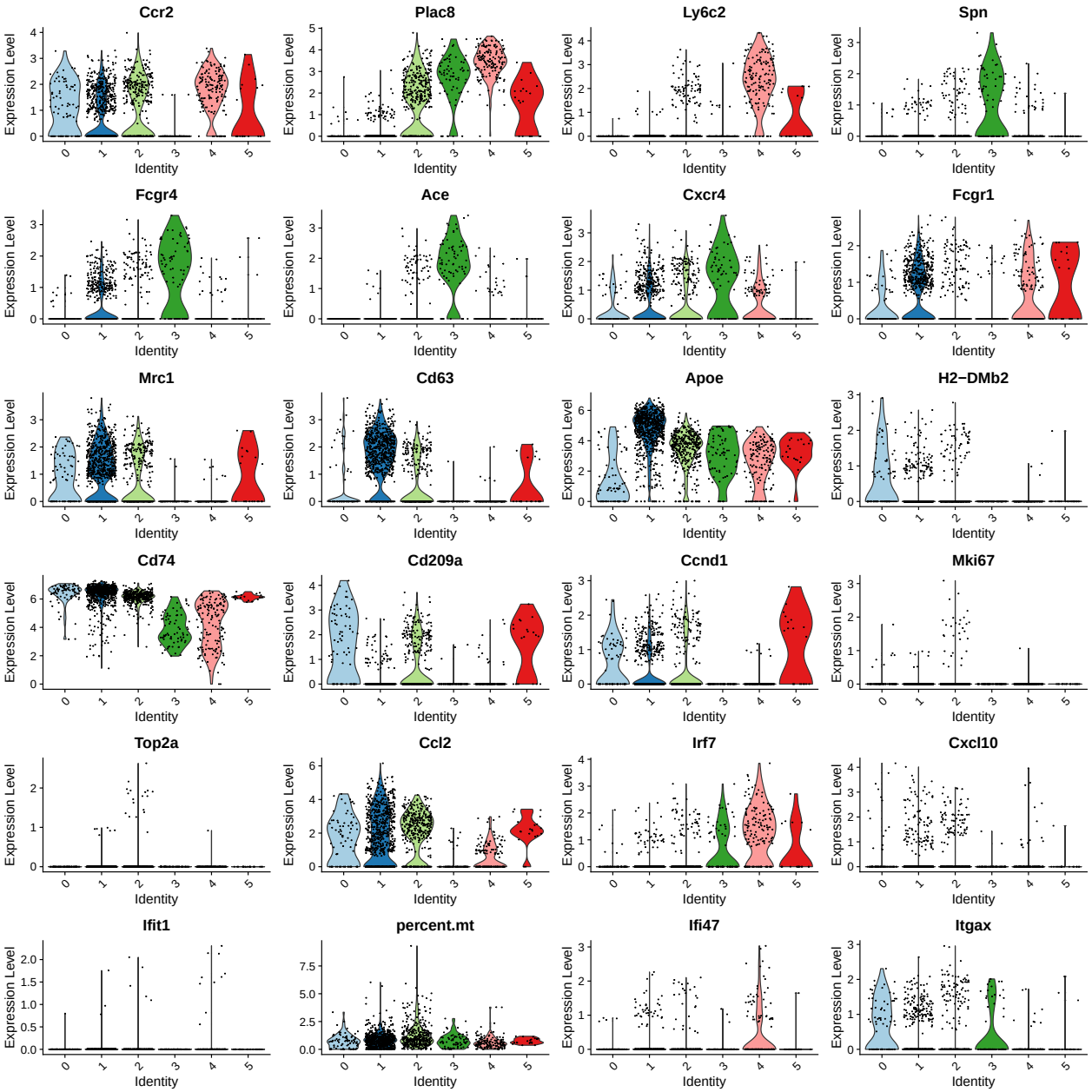
All the IMs are in cluster 1 while cluster 0 only presents Mafb- cells. To confirm we try to identify the populations only in control sample.

5.6.3 Show expresssion of important IM markers in Control sample

```
FeaturePlot(
    subset(results, subset = group == "HT5-Control"),
            features = c("Fcgr1", "Mafb" ,"Mrc1", "Lyve1", "Folr2", "Cd163
               ",
                                "Cd72", "Cd74", "H2-Ab1") ,
          ncol = 3, cols = c("dark blue", "yellow", "red"))
```




```
VlnPlot(
  subset(results, subset = group == "HT5-Control"),
  features = c("Ccr2", "Plac8" ,"Ly6c2", "Spn",
                           "Fcgr4", "Ace", "Cxcr4",
                           "Fcgr1", "Mrc1", "Cd63", "Apoe",
                           "H2-DMb2", "Cd74", "Cd209a",
                           "Ccnd1","Mki67", "Top2a", "Ccl2",
                           "Irf7", "Cxcl10", "Ifit1", "percent.mt", "
                              Ifi47", "Itgax") ,
        ncol = 4, cols = brewer.pal(6, "Paired"))
```



5.6.4 Subset charaterization

cluster 0: Mafb- trapped cluster 1: All IMs (both CD206+ and CD206- IM) cluster 2: Intermediate cluster 3: Patrolling Mono cluster 4: Classical Mono cluster 5: unknown

5.7 Clustering and Annotate CD206+ and CD206- IMs

As Mafb-deficiency introduced a bigger variance which made CD206+/CD206- IMs unable to be clustered separately. We will isolate cluster 1 (contains all IMs) and redo cluster.

```
ims <- subset(results, idents = "1")
```

Normalize and find variable genes

```
ims <- NormalizeData(ims, verbose=FALSE)
```

```
ims <- FindVariableFeatures(ims, selection.method = "vst", nfeatures =
    2000, verbose=FALSE)
ims <- ScaleData(ims, features = rownames(ims), verbose=FALSE)
ims <- RunPCA(ims, features = VariableFeatures(ims), verbose=FALSE)
ims <- RunTSNE(ims, dims = 1:5, verbose=FALSE)
ims <- RunUMAP(ims, dims = 1:5, verbose=FALSE)
```

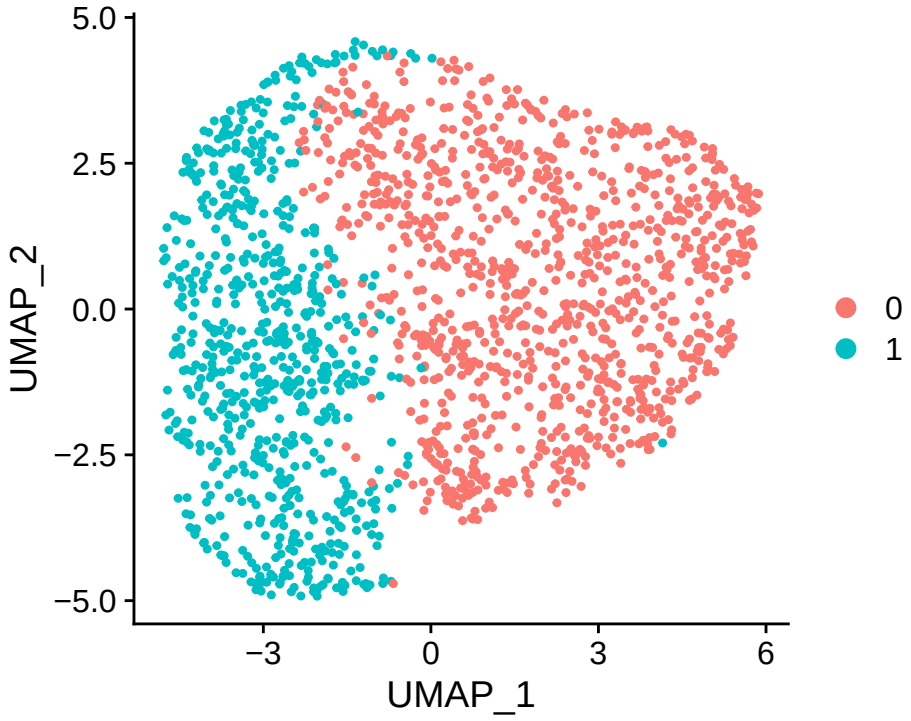
Cell clustering within cluster 1

```
ims <- FindNeighbors(ims, dims = 1:5, verbose = FALSE)
ims <- FindClusters(ims, resolution = 0.2, verbose = FALSE)
```

Show CD206+ and CD206- IMs:

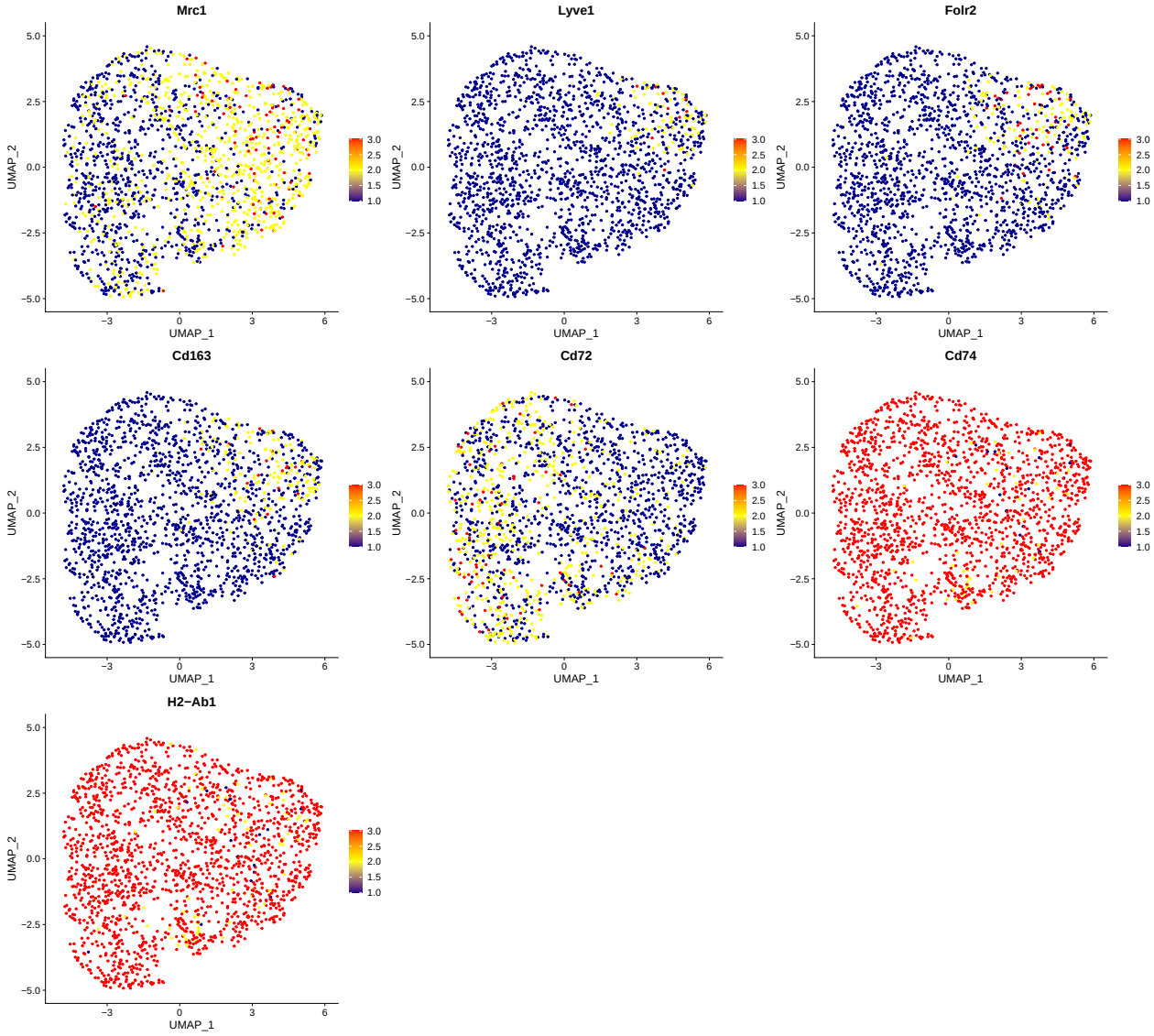
Clusters:

```
DimPlot(ims)
```



Show marker expression:

```
FeaturePlot(ims,
            features = c("Mrc1", "Lyve1", "Folr2", "Cd163",
                         "Cd72", "Cd74", "H2-Ab1") ,
        ncol = 3, cols = c("dark␣blue", "yellow", "red"))
```



Within the cluster1: subcluster 0: CD206+ IMs subcluster 1: CD206- (MHCII hi) IMs

5.8 Annotate with cell types

```
results$cell.type2 <- factor(Idents(results), labels = c("Mafb-␣neo", "IM"
    , "Intermediate", "Patrolling␣Mono", "Classical␣Mono", "Unknown"))

results$cell.type2 <- as.character(results$cell.type2)
```

Overide IM cluster with subcluster annotatinons in new cell.type3 annotation.

```
results$cell.type3 <- results$cell.type2

# overide
results$cell.type3[colnames(ims)] <- as.character(factor(Idents(ims),
    labels = c("CD206+␣IMs", "CD206-␣IMs")))

# Now make annotation into proper factors:
```

```
results$cell.type2 <- as.factor(results$cell.type2)
results$cell.type3 <- factor(results$cell.type3, levels = c("Classical␣
   Mono",
                                                            "Patrolling␣
                                                               Mono",
                                                            "Intermediate"
                                                               ,
                                                            "CD206-␣IMs",
                                                            "CD206+␣IMs",
                                                            "Mafb-␣neo",
                                                            "Unknown"))

Idents(results) <- "cell.type3"
levels(results)
```

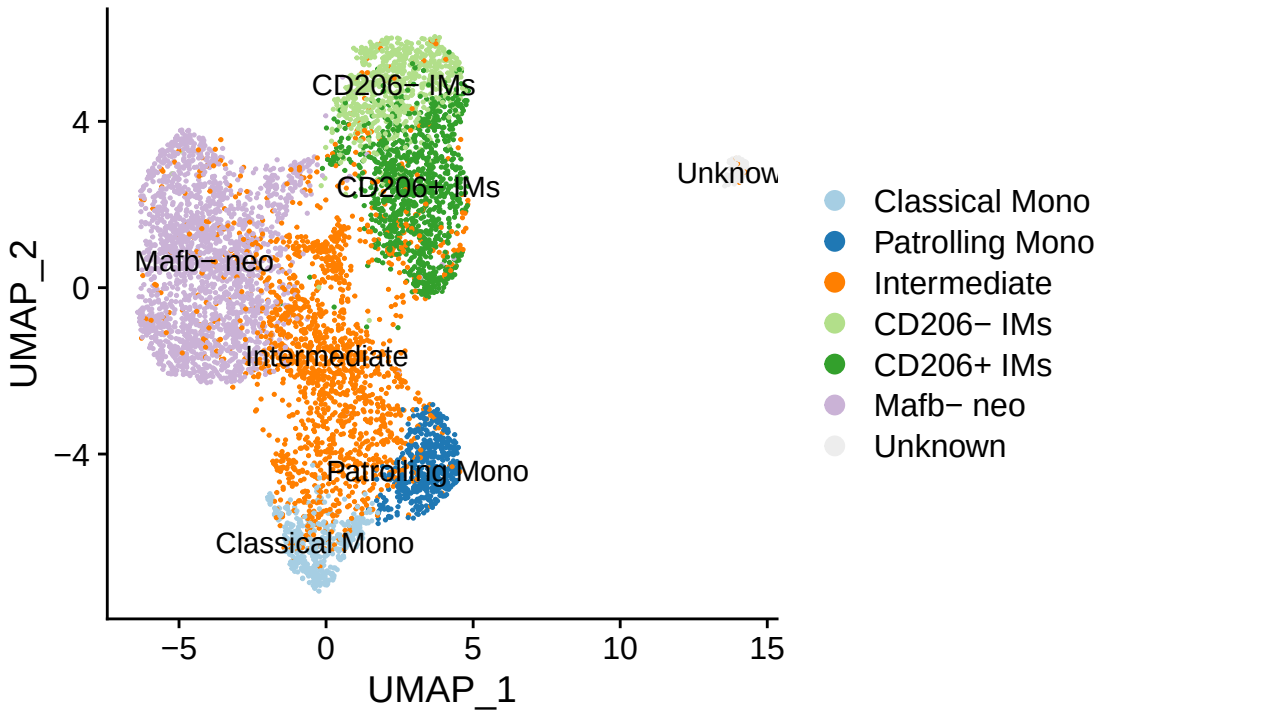
```
## [1] "Classical Mono"  "Patrolling Mono" "Intermediate"    "CD206- IMs"
## [5] "CD206+ IMs"      "Mafb- neo"       "Unknown"
```

5.9 Make plots with annotated cells

Make color palette for plot (3 new colors were Paired number 8 and 9 from RColorBrewer, and light grey for unknown population):

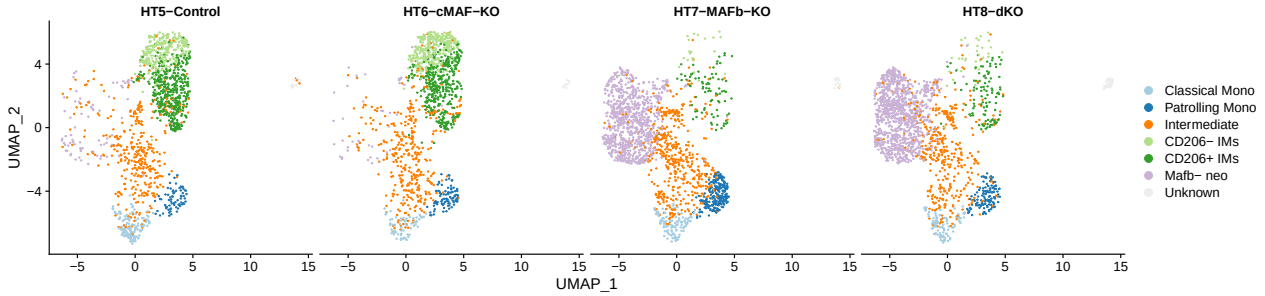
```
pal3 <- c(
  "#A6CEE3", # cMo
  "#1F78B4", # pMo
  "#FF7F00", # Intermediate
  "#B2DF8A", # MHCII IM
  "#33A02C", # CD206 IM
  "#CAB2D6", # Mafb- neo
  "#ededed"# Unknown
)
```

```
DimPlot(results, label = TRUE, cols = pal3)
```




```
ggsave(filename = "../Figures/UMAPplot_All_samplesMaf_label.pdf", width =
    6, height = 4)
```

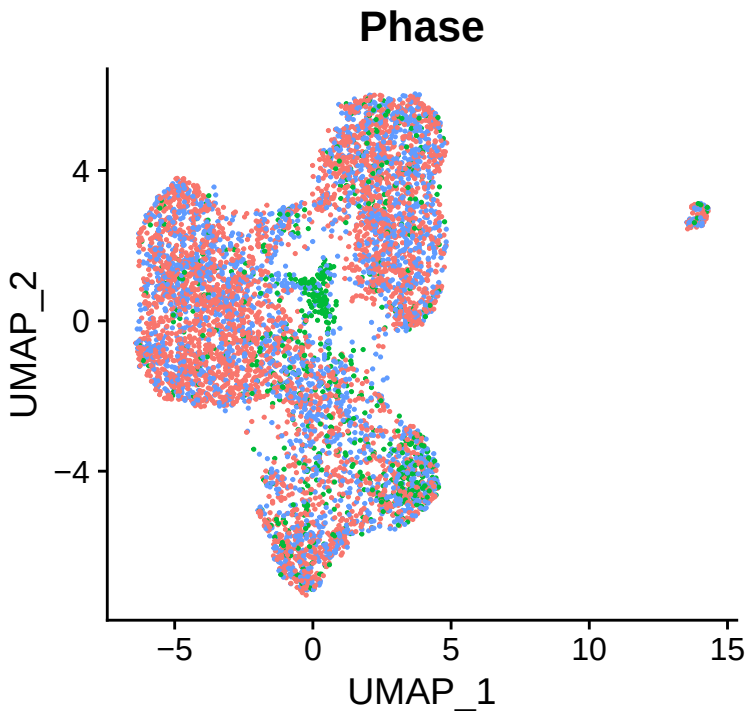
```
DimPlot(results, cols = pal3, split.by = "group")
```




```
ggsave(filename = "../Figures/UMAPplot_all_separate_samplesMaf.pdf", width
    = 16, height = 4)
```

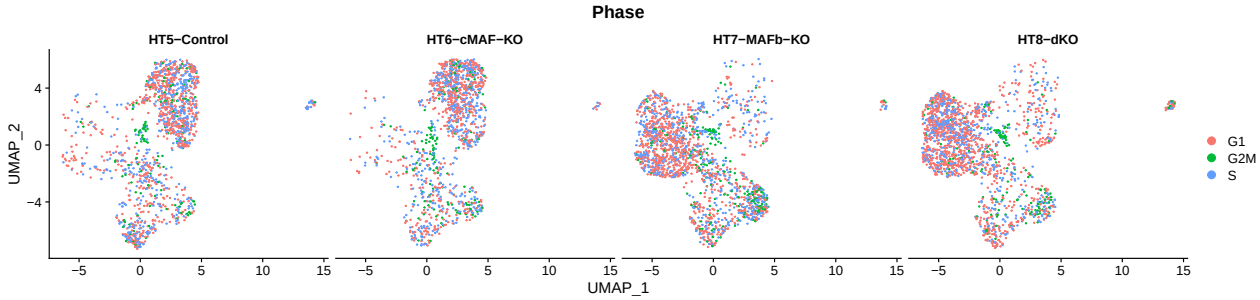
5.10 Cell cycle analysis

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



We see a clear cycling core in Intermediate population.

```
DimPlot(results, group.by = "Phase", reduction = "umap", split.by = "group")
```




```
ggsave(filename = "../Figures/UMAPplot_all_separate_samplesMaf_Phase.pdf",
    width = 16, height = 4)
```

```
saveRDS(results, file = "./All_samples_Maf.seuratObject.Rds")
```

6 Session information

R sesssion:

```
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.3 LTS
##
```

```
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/liblapack.so.3
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_GB.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils
   datasets
## [8] methods   base
##
## other attached packages:
## [1] cowplot_1.1.1       RColorBrewer_1.1-2  S4Vectors_0.28.1
## [4] BiocGenerics_0.36.1 dplyr_1.0.7         ggpubr_0.4.0
## [7] ggplot2_3.3.5       SeuratObject_4.0.4  Seurat_4.0.5
##
## loaded via a namespace (and not attached):
##   [1] backports_1.4.0       Hmisc_4.6-0           systemfonts_1.0.3
##   [4] plyr_1.8.6            igraph_1.2.9          lazyeval_0.2.2
##   [7] splines_4.0.3         listenv_0.8.0         scattermore_0.7
##  [10] digest_0.6.29         foreach_1.5.1         htmltools_0.5.2
##  [13] fansi_0.5.0           checkmate_2.0.0       magrittr_2.0.1
##  [16] tensor_1.5            cluster_2.1.0         ROCR_1.0-11
##  [19] limma_3.46.0          tzdb_0.2.0            readr_2.1.1
##  [22] recipes_0.1.17        globals_0.14.0        gower_0.2.2
##  [25] matrixStats_0.61.0    spatstat.sparse_2.0-0 jpeg_0.1-9
##  [28] colorspace_2.0-2      ggrepel_0.9.1         textshaping_0.3.6
##  [31] xfun_0.28             crayon_1.4.2          jsonlite_1.7.2
##  [34] spatstat.data_2.1-0   survival_3.2-7        zoo_1.8-9
##  [37] iterators_1.0.13      glue_1.5.1            polyclip_1.10-0
##  [40] gtable_0.3.0          ipred_0.9-12          leiden_0.3.9
##  [43] car_3.0-12            future.apply_1.8.1    abind_1.4-5
##  [46] scales_1.1.1          DBI_1.1.1             rstatix_0.7.0
##  [49] miniUI_0.1.1.1        Rcpp_1.0.7            htmlTable_2.3.0
##  [52] viridisLite_0.4.0     xtable_1.8-4          reticulate_1.22
##  [55] spatstat.core_2.3-2   foreign_0.8-81        proxy_0.4-26
##  [58] Formula_1.2-4         lava_1.6.10           prodlim_2019.11.13
##  [61] htmlwidgets_1.5.4     httr_1.4.2            DiagrammeR_1.0.6.1
##  [64] ellipsis_0.3.2        ica_1.0-2             pkgconfig_2.0.3
##  [67] farver_2.1.0          nnet_7.3-14           uwot_0.1.11
##  [70] deldir_1.0-6          utf8_1.2.2            caret_6.0-90
##  [73] tidyselect_1.1.1      labeling_0.4.2        rlang_0.4.12
##  [76] reshape2_1.4.4        later_1.3.0           visNetwork_2.1.0
##  [79] munsell_0.5.0         tools_4.0.3           generics_0.1.1
##  [82] broom_0.7.10          ggridges_0.5.3        fdrtool_1.2.17
##  [85] evaluate_0.14         stringr_1.4.0         fastmap_1.1.0
##  [88] yaml_2.2.1            ragg_1.2.1            goftest_1.2-3
##  [91] ModelMetrics_1.2.2.2  knitr_1.36            fitdistrplus_1.1-6
```

```
##  [94] caTools_1.18.2         randomForest_4.6-14    purrr_0.3.4
##  [97] RANN_2.6.1             pbapply_1.5-0          future_1.23.0
## [100] nlme_3.1-153           mime_0.12              rstudioapi_0.13
## [103] compiler_4.0.3         plotly_4.10.0          png_0.1-7
## [106] e1071_1.7-9            ggsignif_0.6.3         spatstat.utils_2.2-0
## [109] tibble_3.1.6           stringi_1.7.6          highr_0.9
## [112] RSpectra_0.16-0        lattice_0.20-41        Matrix_1.3-4
## [115] vctrs_0.3.8            pillar_1.6.4           lifecycle_1.0.1
## [118] spatstat.geom_2.3-0    lmtest_0.9-39          RcppAnnoy_0.0.19
## [121] bitops_1.0-7           data.table_1.14.2      irlba_2.3.5
## [124] httpuv_1.6.3           patchwork_1.1.1        latticeExtra_0.6-29
## [127] R6_2.5.1               promises_1.2.0.1       KernSmooth_2.23-20
## [130] gridExtra_2.3          parallelly_1.29.0      codetools_0.2-18
## [133] MASS_7.3-53            assertthat_0.2.1       withr_2.4.3
## [136] sctransform_0.3.2      hms_1.1.1              mgcv_1.8-33
## [139] grid_4.0.3             rpart_4.1-15           nichenetr_1.0.0
## [142] timeDate_3043.102      tidyr_1.1.4            class_7.3-17
## [145] rmarkdown_2.11         carData_3.0-4          Rtsne_0.15
## [148] pROC_1.18.0            base64enc_0.1-3        shiny_1.7.1
## [151] lubridate_1.8.0
```

7 References