Monocytes can Proliferate in Vacant Tissue Niches prior to Differentiation into Macrophages

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

2022-01-28 13:43:44 +0100

Abstract

Resident tissue macrophages (RTM) are differentiated immune cells populating distinct niches and exhibiting important tissue-supportive functions. RTM maintenance is thought to depend either on monocyte engraftment and differentiation, or on the self-renewal of mature RTM. Here, we discovered that monocytes can re-enter cell cycle and proliferate locally before their differentiation into RTM. We developed a mouse model of inducible lung interstitial macrophage (IM) depletion in which the vacant niche is repopulated by BM-derived monocytes giving rise to fully differentiated IM subsets. By performing time-course single-cell RNA-sequencing analyses of myeloid cells during niche refilling, we found that few Ly6C+ classical monocytes could self-renew locally in a CSF1R-dependent manner. We further showed that the transcription factor MafB restricted such proliferation and was essential to mediate RTM specification and identity in our model. Our data 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

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 pepline build Seurat object from matrix, celltyping and contaminated cell removal were identical to the previous anlayses (Analysis 2, 3 and 4) using Seurat pakcage (Hao et al., 2021) and SingleR package (Aran et al., 2019). 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-CTTTGTCTTTGTGAG" as it's not added to the sample.
immune.htos <- immune.htos[rownames(immune.htos) != "HT5_CTL-
    CTTTGTCTTTGTGAG", ]

# 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

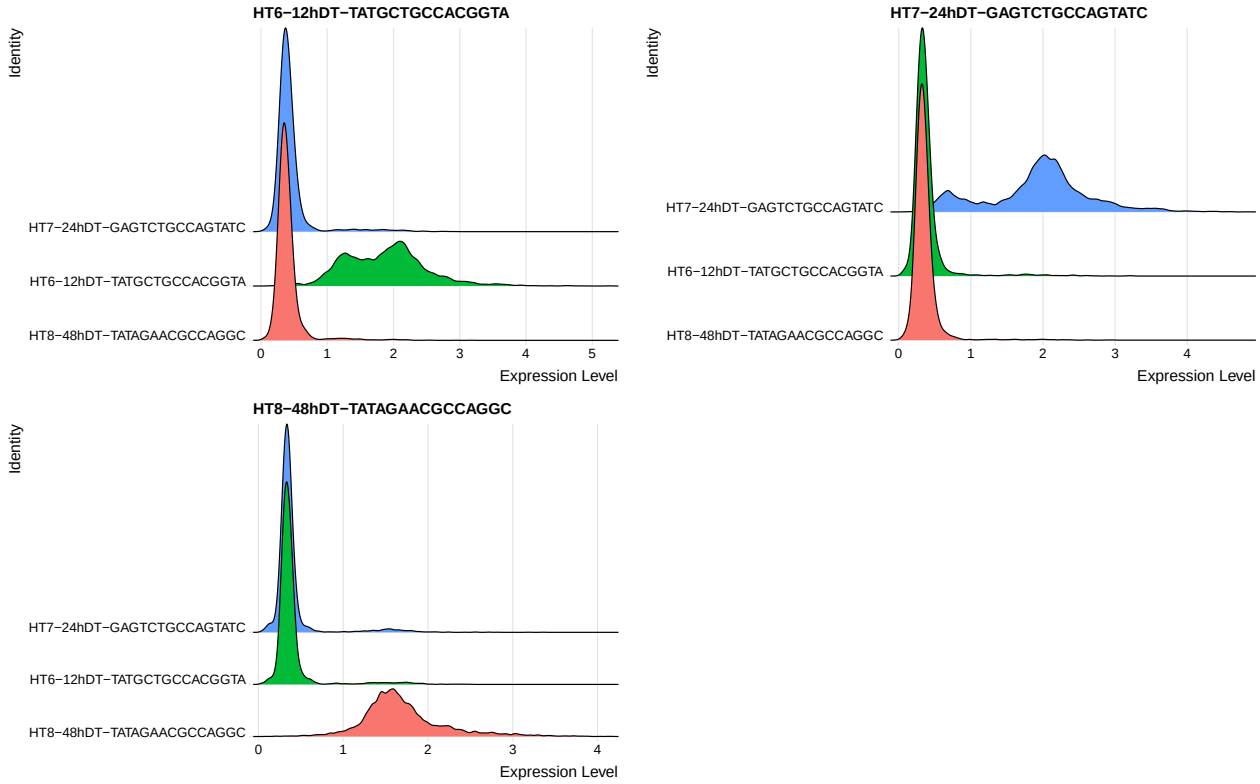
Output from running `HTODemux()` is saved in the object metadata. We can visualize how many cells are classified as singlets, doublets and negative/ambiguous cells.

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

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

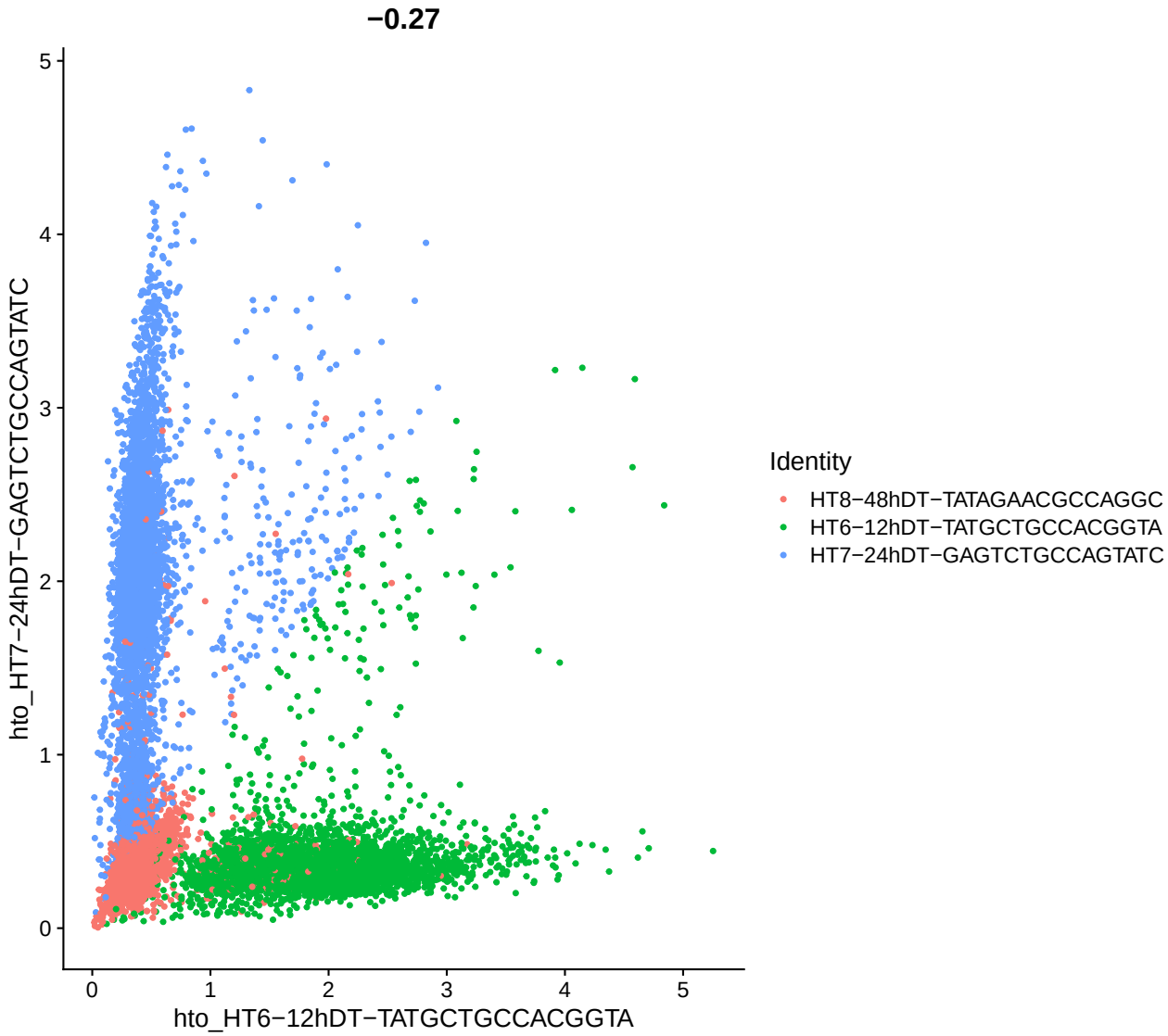
Visualize enrichment for selected HTOs with ridge plots

```
# 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)
```

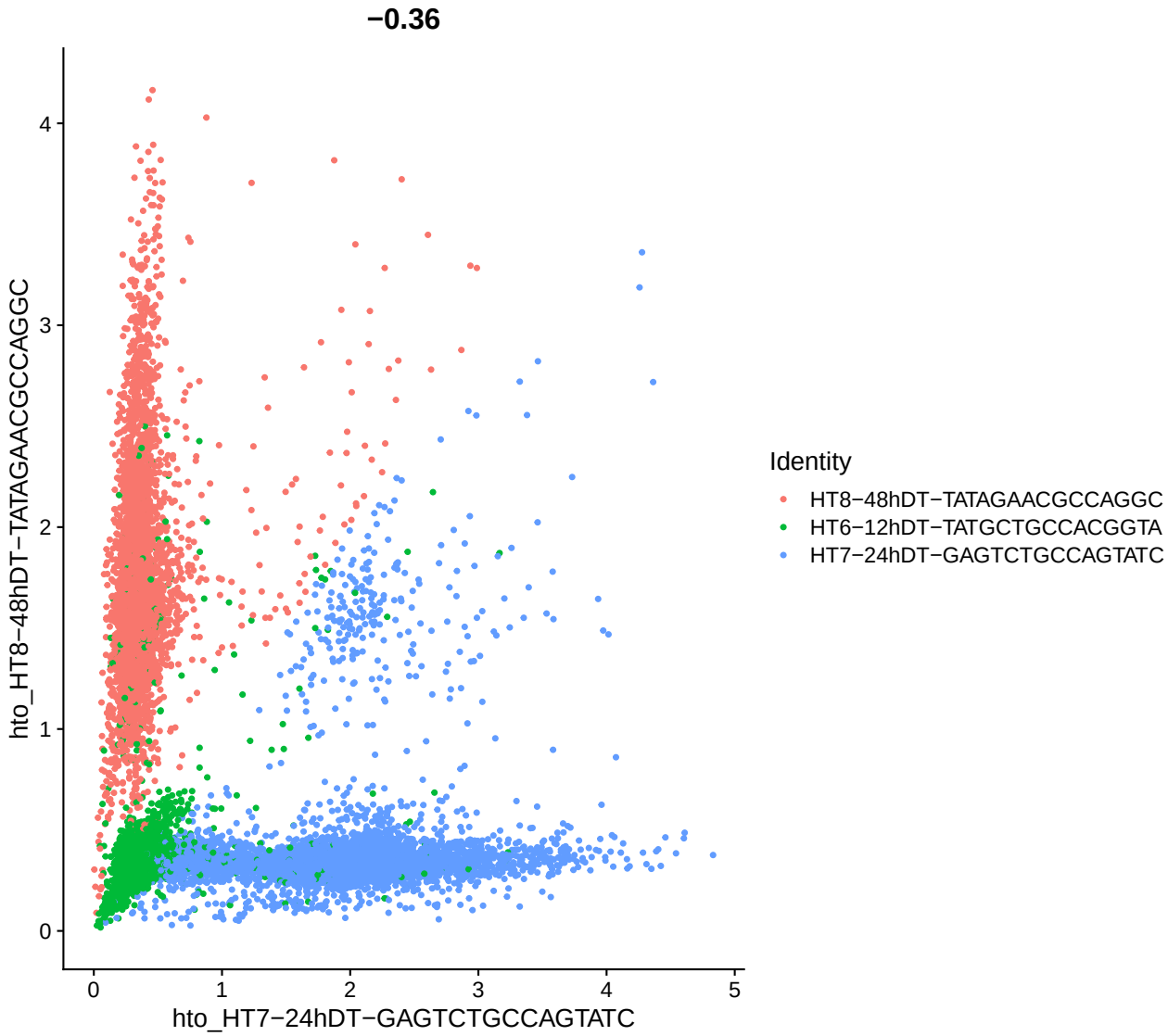


Visualize pairs of HTO signals to confirm mutual exclusivity in singlets

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

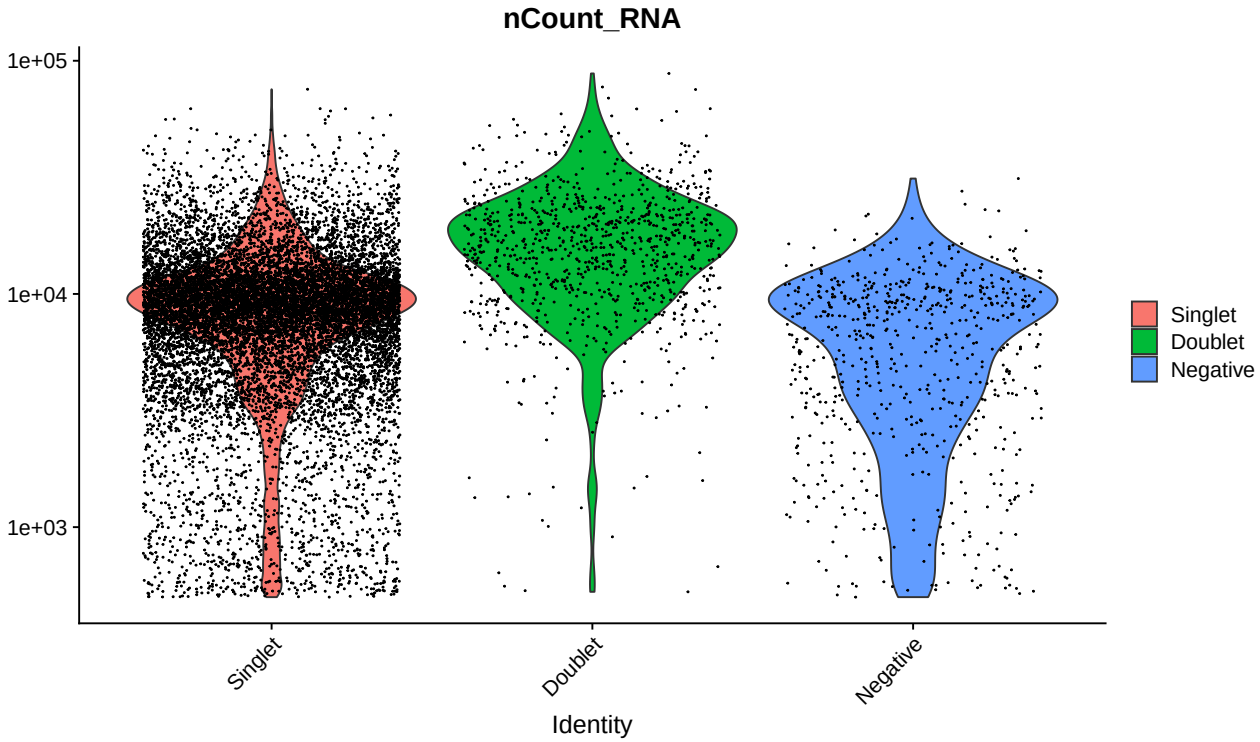



```
FeatureScatter(immune.hashtag, feature1 = 'HT7-24hDT-GAGTCTGCCAGTATC',
    feature2 = 'HT8-48hDT-TATAGAACGCCAGGC')
```



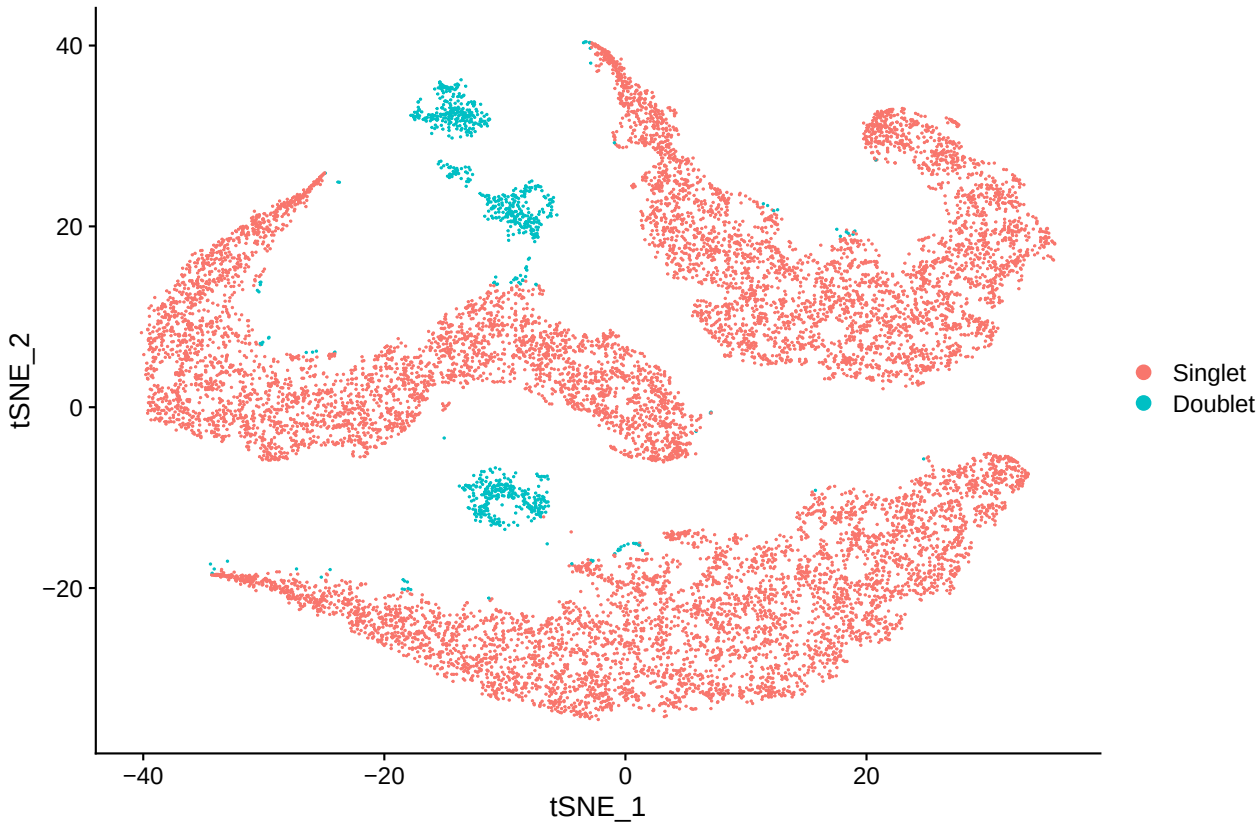
Compare number of UMIs for singlets, doublets and negative cells

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



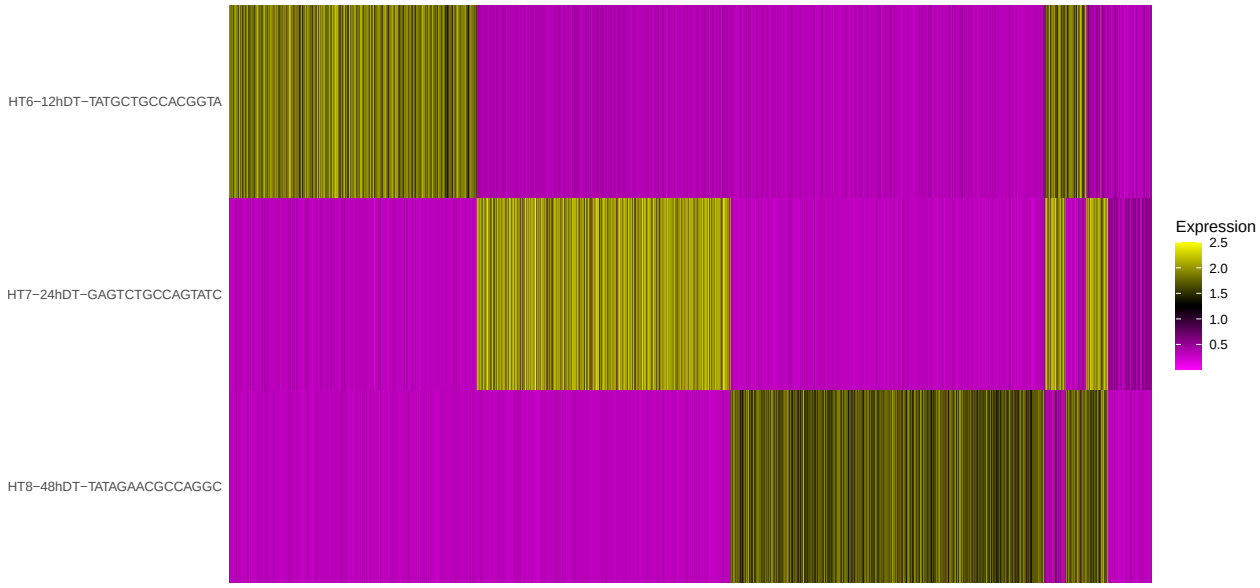
Generate a two dimensional tSNE embedding for HTOs.Here we are grouping cells by singlets and doublets for simplicity.

```
#First, we will remove negative cells from the object
immune.hashtag.subset <- subset(immune.hashtag, idents = 'Negative',
    invert = TRUE)

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

DimPlot(immune.hashtag.subset)
```



Create an HTO heatmap

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

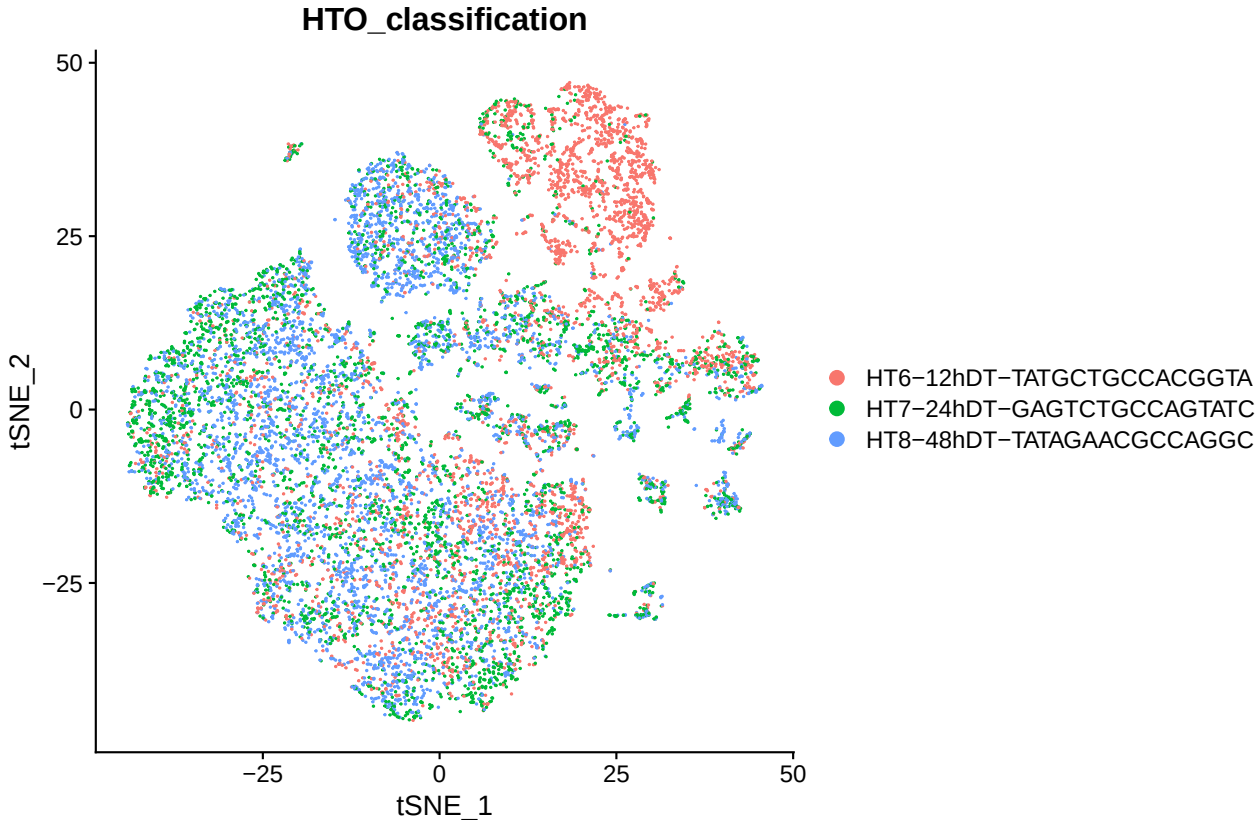


Cluster and visualize cells using the usual scRNA-seq workflow, and examine for the potential presence of batch effects.

```
# 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:

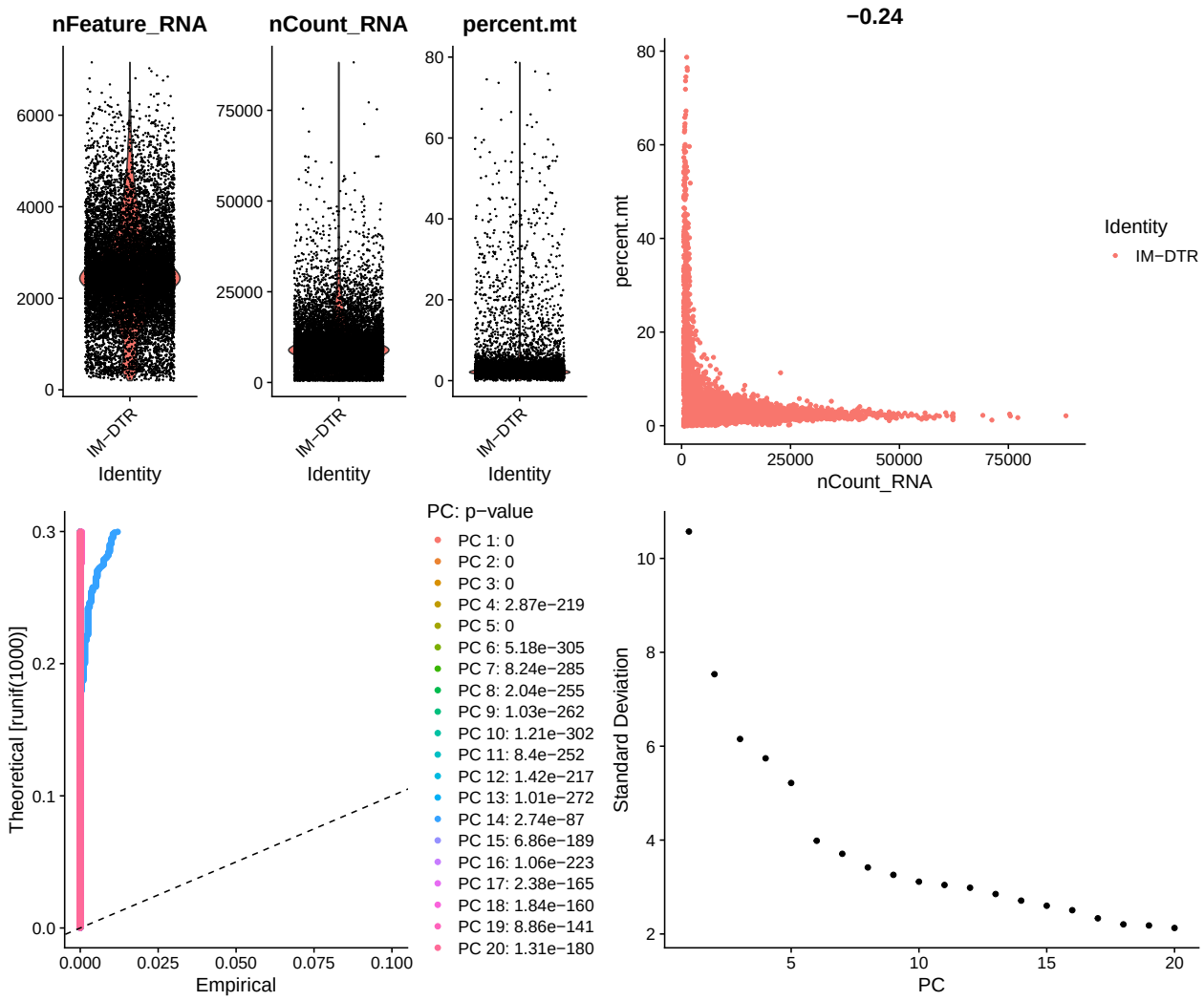
```
# load package and data
library(ggpubr)
source("../R/seurat.setup.R")
dir.10x <- "/mnt/Data/Single-cell_Analysis/Projects/IPL/IM_DTR/IM_DTR_exp2/counts/scRNAseq/chromium"
```

3.1 Sample: CD45Plus_NGS21-S229

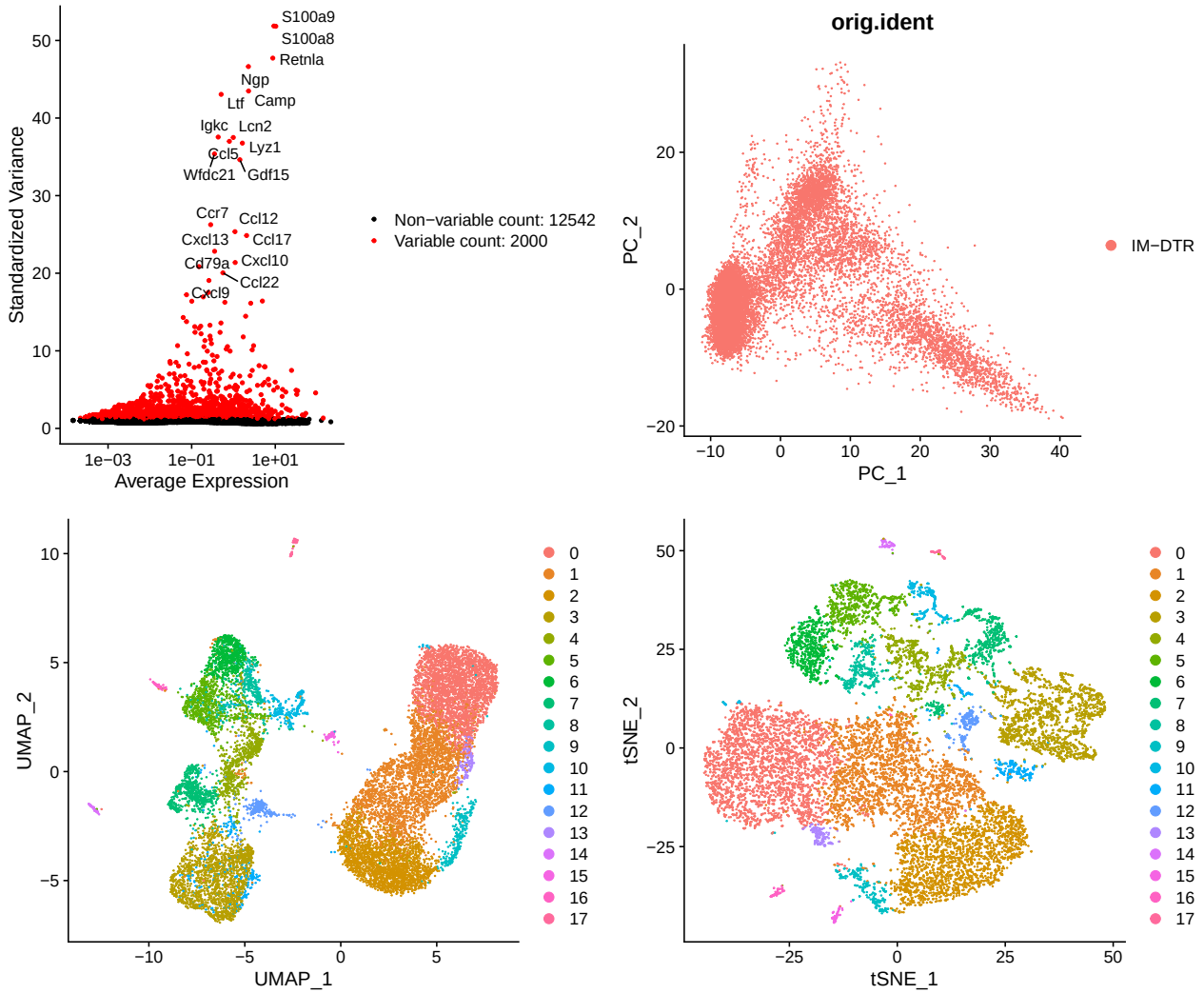
```
CD45Plus_NGS21_S229 <- seurat.setup(path.10x = file.path(dir.10x, "CD45Plus_NGS21-S229/outs/filtered_feature_bc_matrix/") , project = "IM-DTR", dimensionality = 1:20, mt.percentage = 20, human = FALSE)
```

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

```
ggarrange(CD45Plus_NGS21_S229$plots$feature_vln, CD45Plus_NGS21_S229$plots$RNA_mt.pct_scatter, CD45Plus_NGS21_S229$plots$JackStrawPlot, CD45Plus_NGS21_S229$plots$ElbowPlot, ncol = 2, nrow = 2)
```




```
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 = "")

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

4.1 Build metadata table

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

save individual samples for other use:

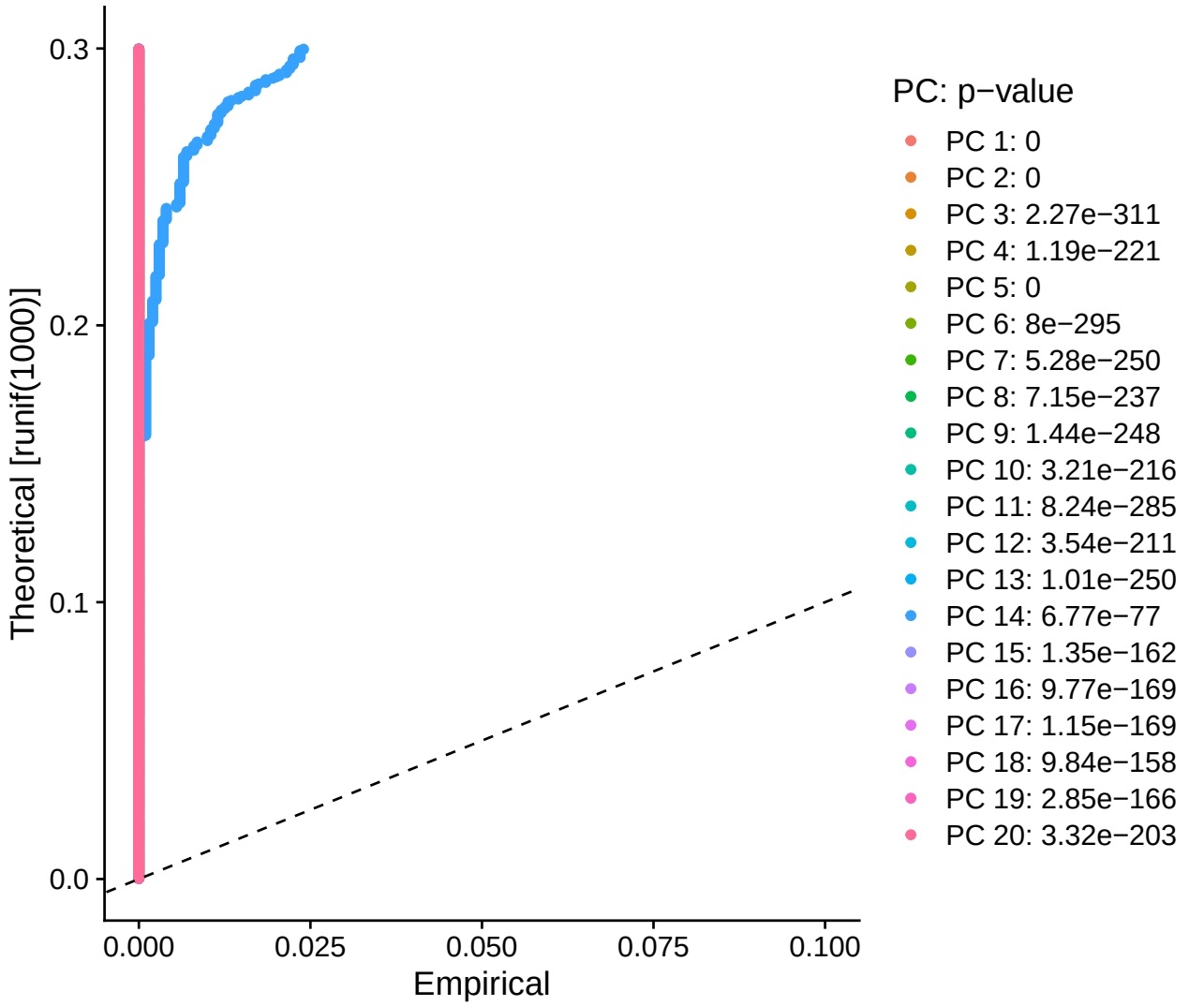
```
obj <- CD45Plus_NGS21_S229.seuratObject
for (i in unique(obj$treatment)) {
  obj.sub <- subset(obj, subset = treatment == i)
  file.name <- paste("CD45Plus_NGS21_S229", i, "seuratObject.rds", sep = ".")
  # saveRDS(object = obj.sub, file = file.path(".", file.name)) # Here we can also save the individual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

4.2 Data processing for CD45Plus_NGS21_S229

```
CD45Plus_NGS21_S229.seuratObject <- NormalizeData(CD45Plus_NGS21_S229.seuratObject)
CD45Plus_NGS21_S229.seuratObject <- FindVariableFeatures(CD45Plus_NGS21_S229.seuratObject,
                                                         selection.method = "vst", nfeatures = 2000)
CD45Plus_NGS21_S229.seuratObject <- ScaleData(CD45Plus_NGS21_S229.seuratObject,
                                              features = rownames(CD45Plus_NGS21_S229.seuratObject))
```

4.3 Linear dimension reduction

```
CD45Plus_NGS21_S229.seuratObject <- RunPCA(CD45Plus_NGS21_S229.seuratObject,
                                           features = VariableFeatures(object = CD45Plus_NGS21_S229.seuratObject))
CD45Plus_NGS21_S229.seuratObject <- JackStraw(CD45Plus_NGS21_S229.seuratObject, num.replicate = 100)
CD45Plus_NGS21_S229.seuratObject <- ScoreJackStraw(CD45Plus_NGS21_S229.seuratObject, dims = 1:20)
JackStrawPlot(CD45Plus_NGS21_S229.seuratObject, dims = 1:20)
```

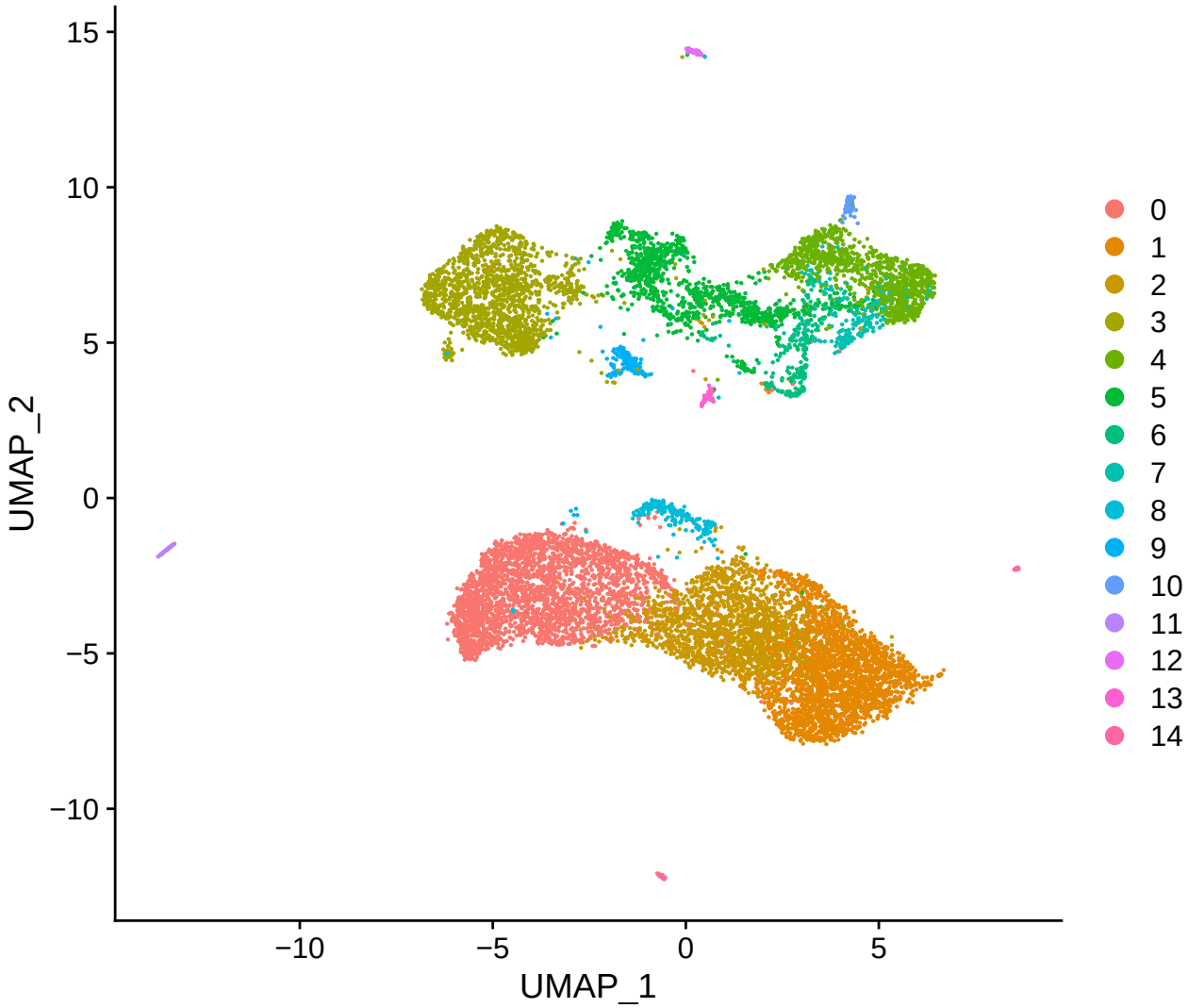



```
CD45Plus_NGS21_S229.seuratObject <- FindNeighbors(CD45Plus_NGS21_S229.
    seuratObject, dims = 1:30)
CD45Plus_NGS21_S229.seuratObject <- FindClusters(CD45Plus_NGS21_S229.
    seuratObject, resolution = 0.5)
```

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

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

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



5 Identify cell types

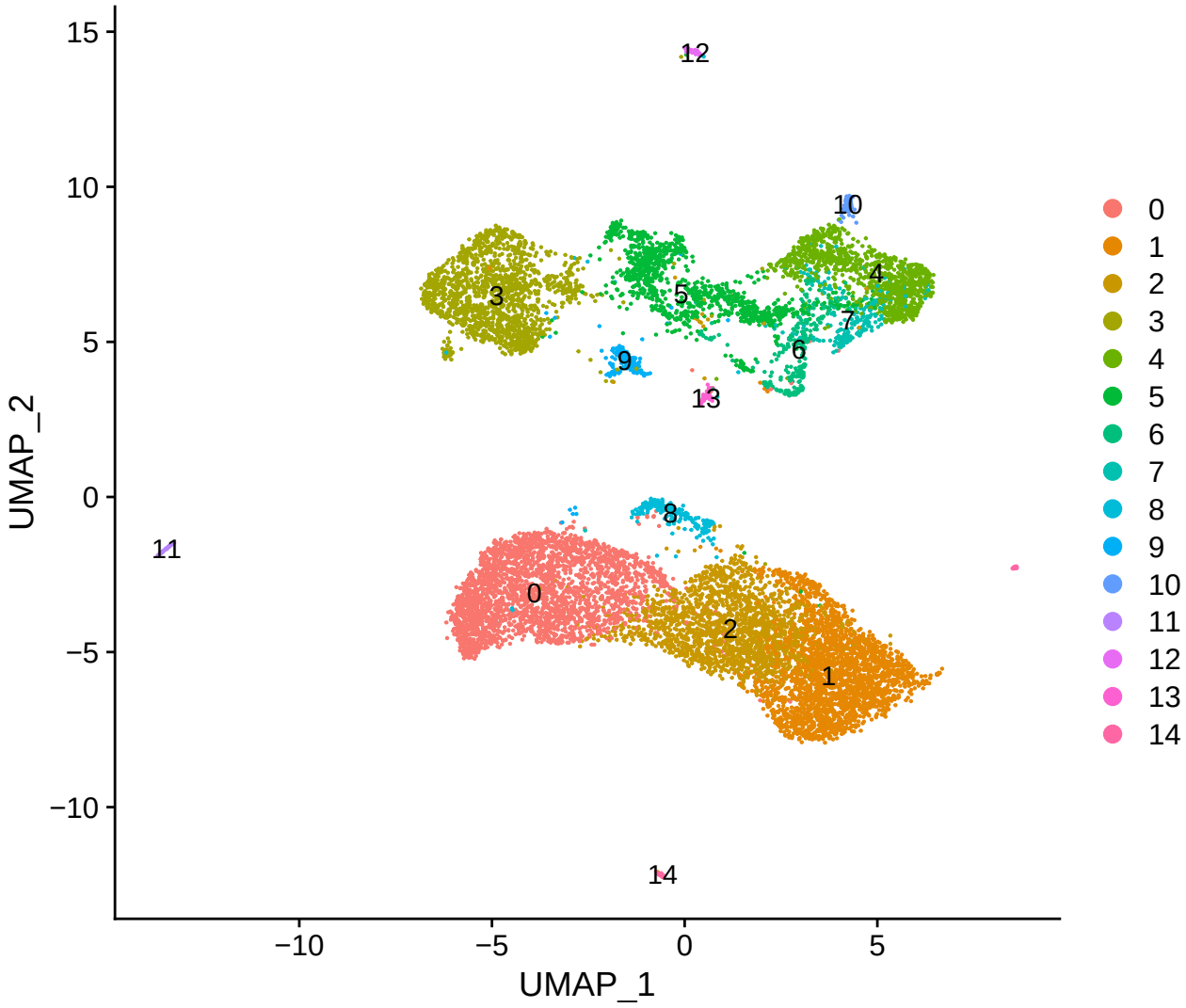
5.1 SingleR analysis

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

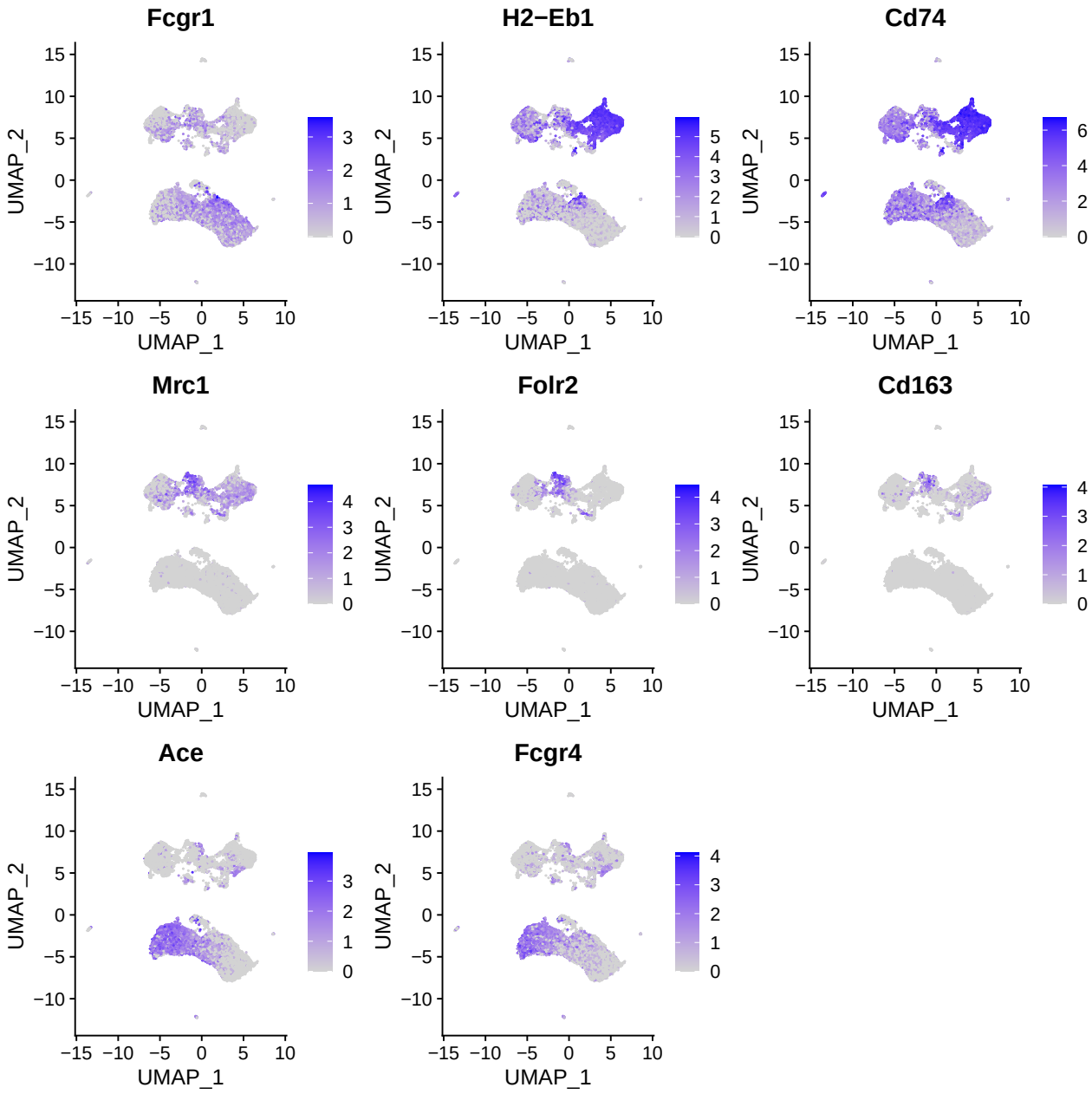
```
saveRDS(results.singleR, file = "./CD45Plus_NGS21_S229.ImmGenData.singleR.
    Rds")
saveRDS(CD45Plus_NGS21_S229.seuratObject, file = "./CD45Plus_NGS21_S229.
    seuratObject.rds")
```

5.2 Marker expression check

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

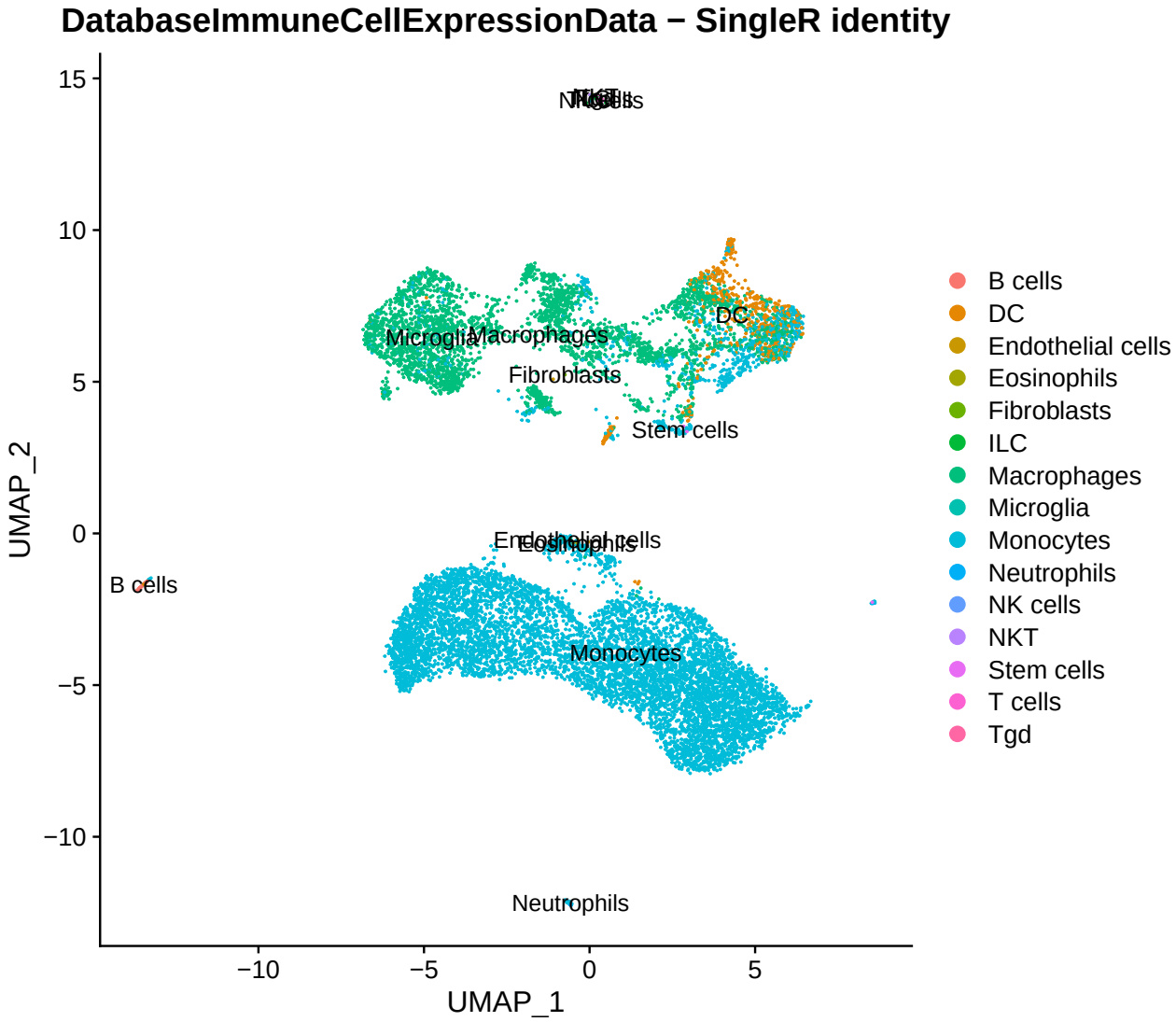



```
DefaultAssay(results) <- "RNA"
FeaturePlot(results, features = c("Fcgr1", "H2-Eb1", "Cd74", "Mrc1", "
    Folr2", "Cd163", "Ace", "Fcgr4"), reduction = "umap")
```

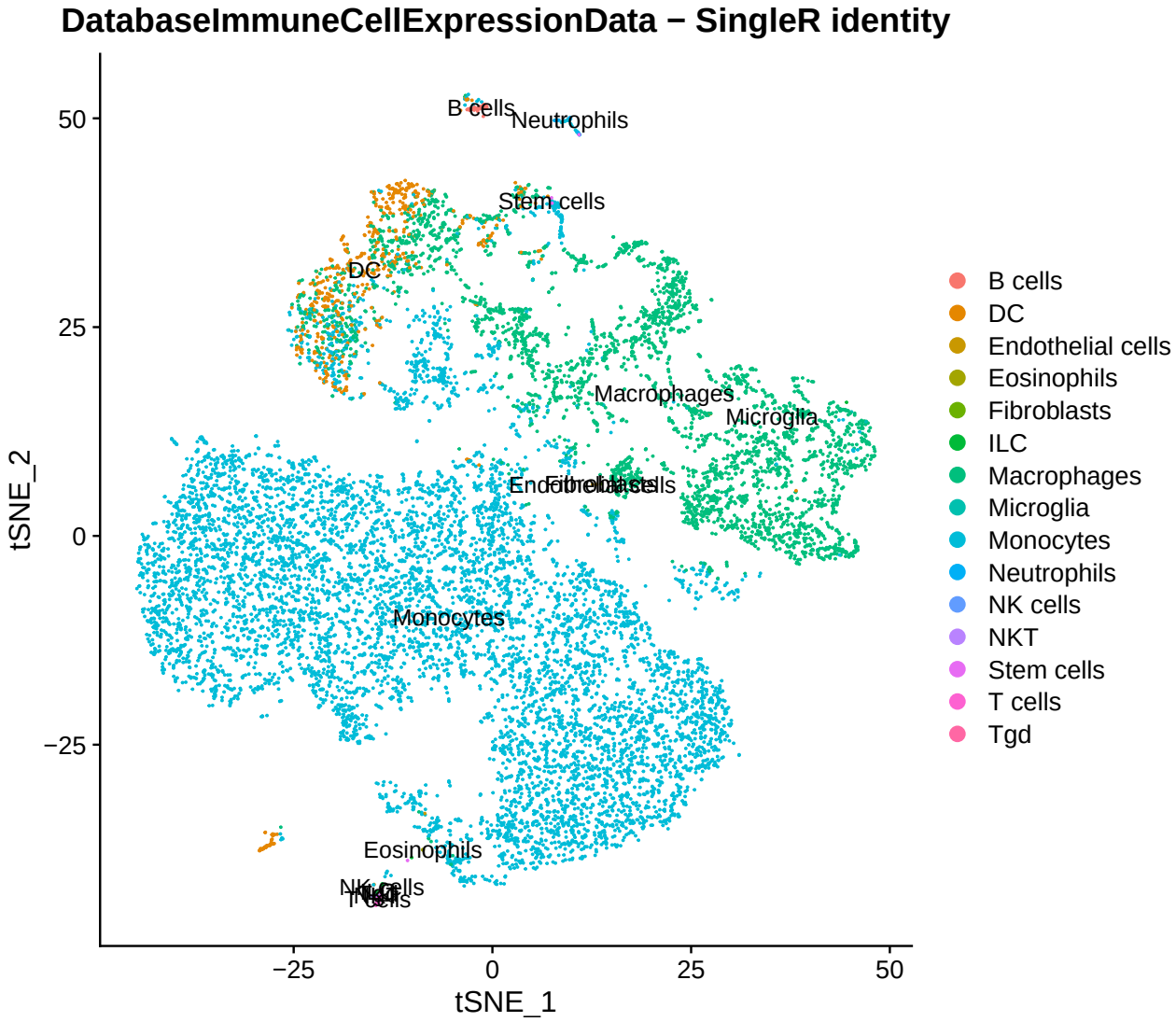


5.3 Combine marker expression with SingleR results

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

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




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



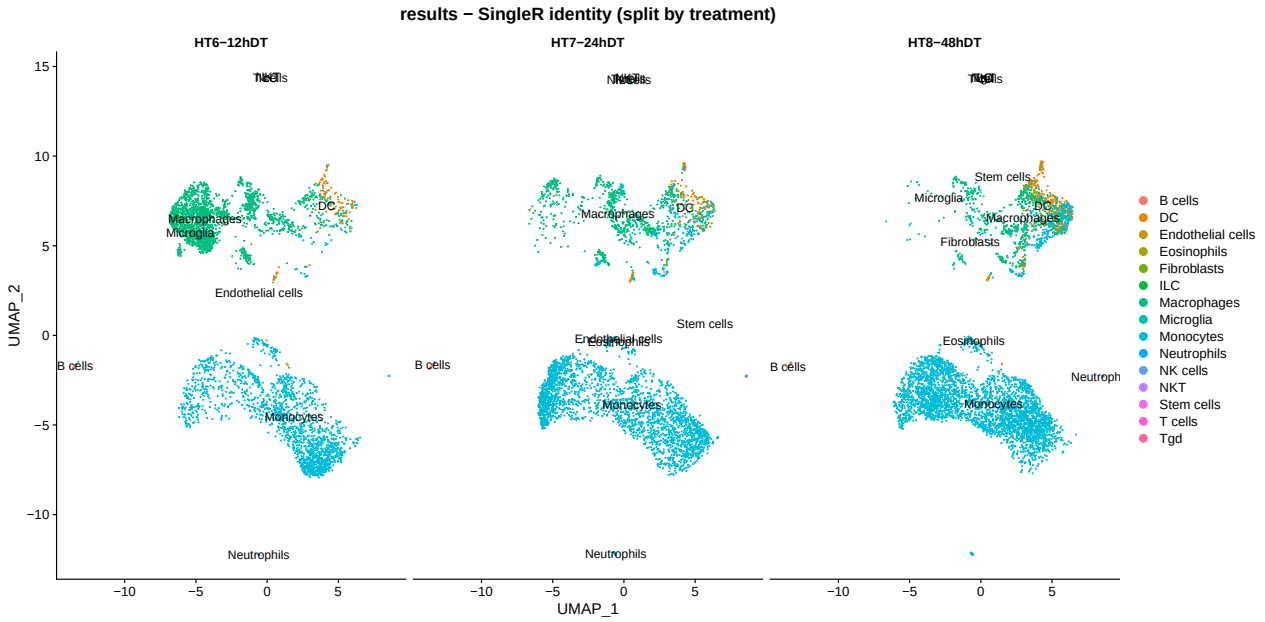
Cell numbers of each type:

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

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

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", "
    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:

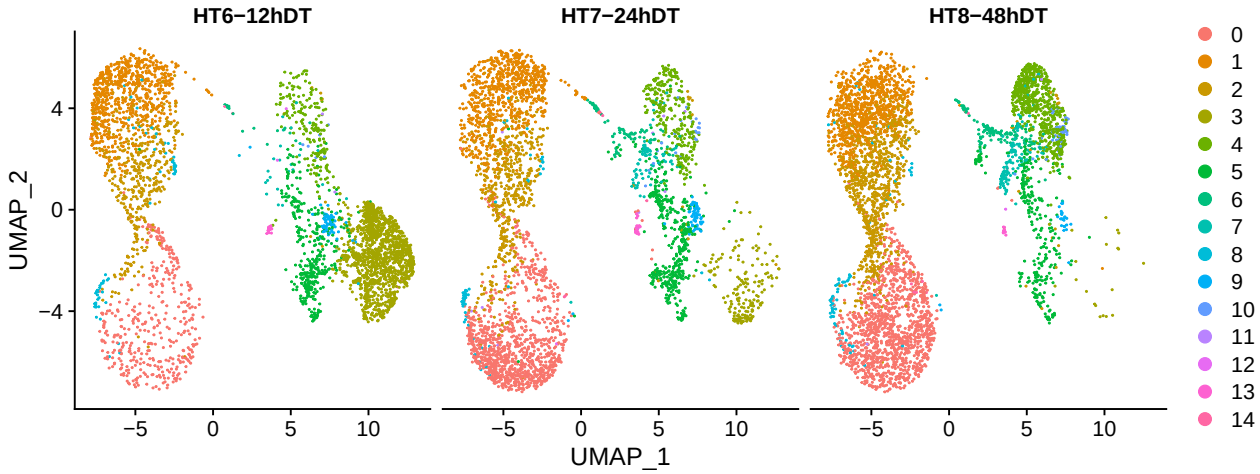
```
results$RNA_snn_res.0.5 <- NULL
```

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

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

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

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

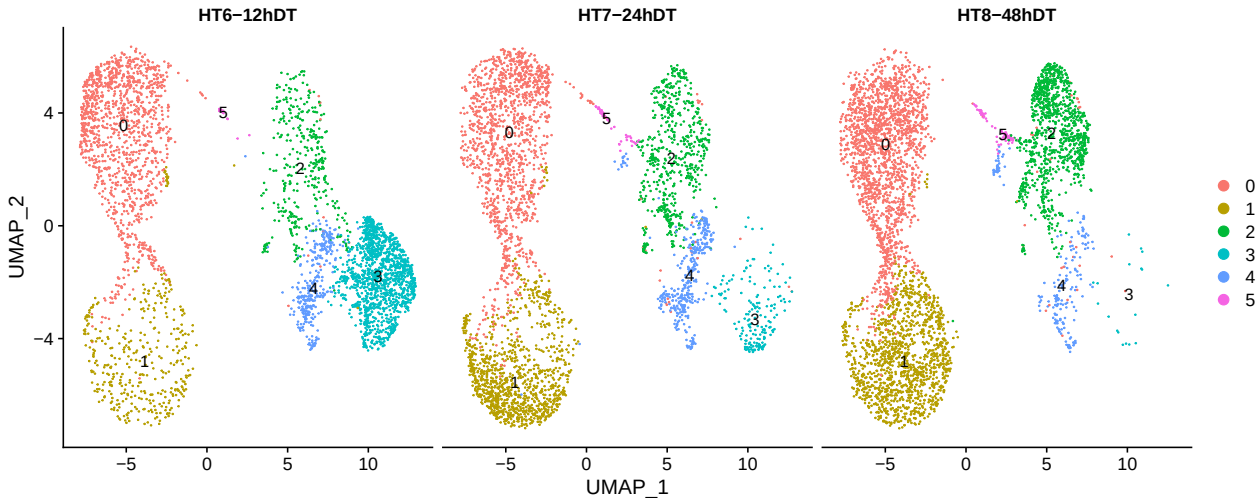


6.2 Clustering of cells

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

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

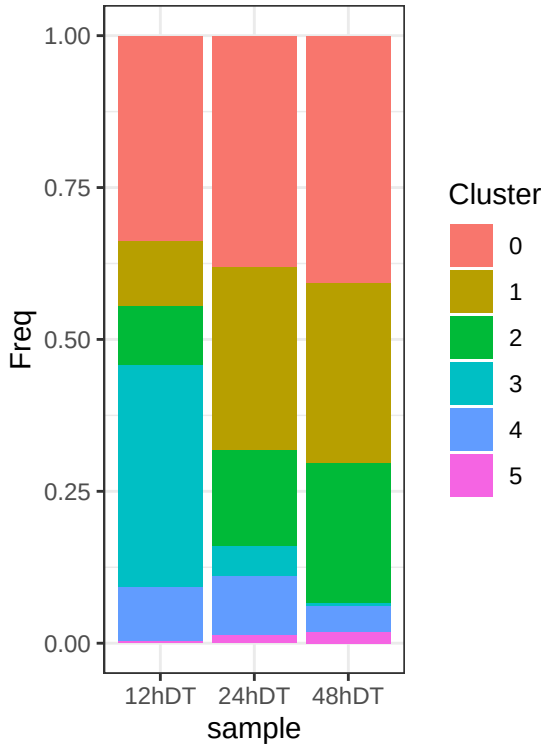
```
Idents(results) <- "RNA_snn_res.0.14"
DimPlot(results, label = TRUE, split.by = "treatment")
```



Distribution of clusters across samples:

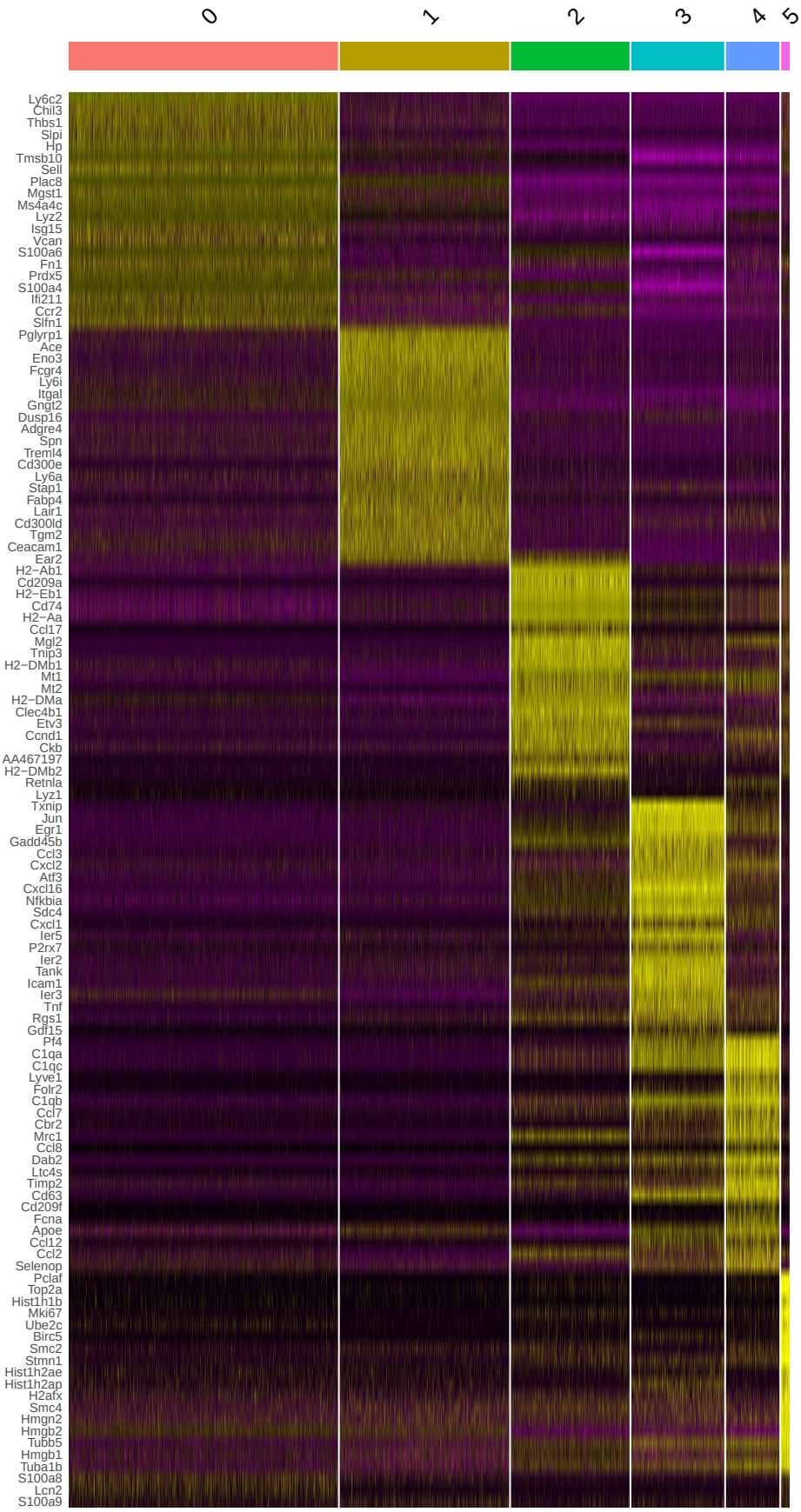
```
source("../R/SeuratFreqTable.R")
freq.celltype.list <- list(
```

```
  `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")
```



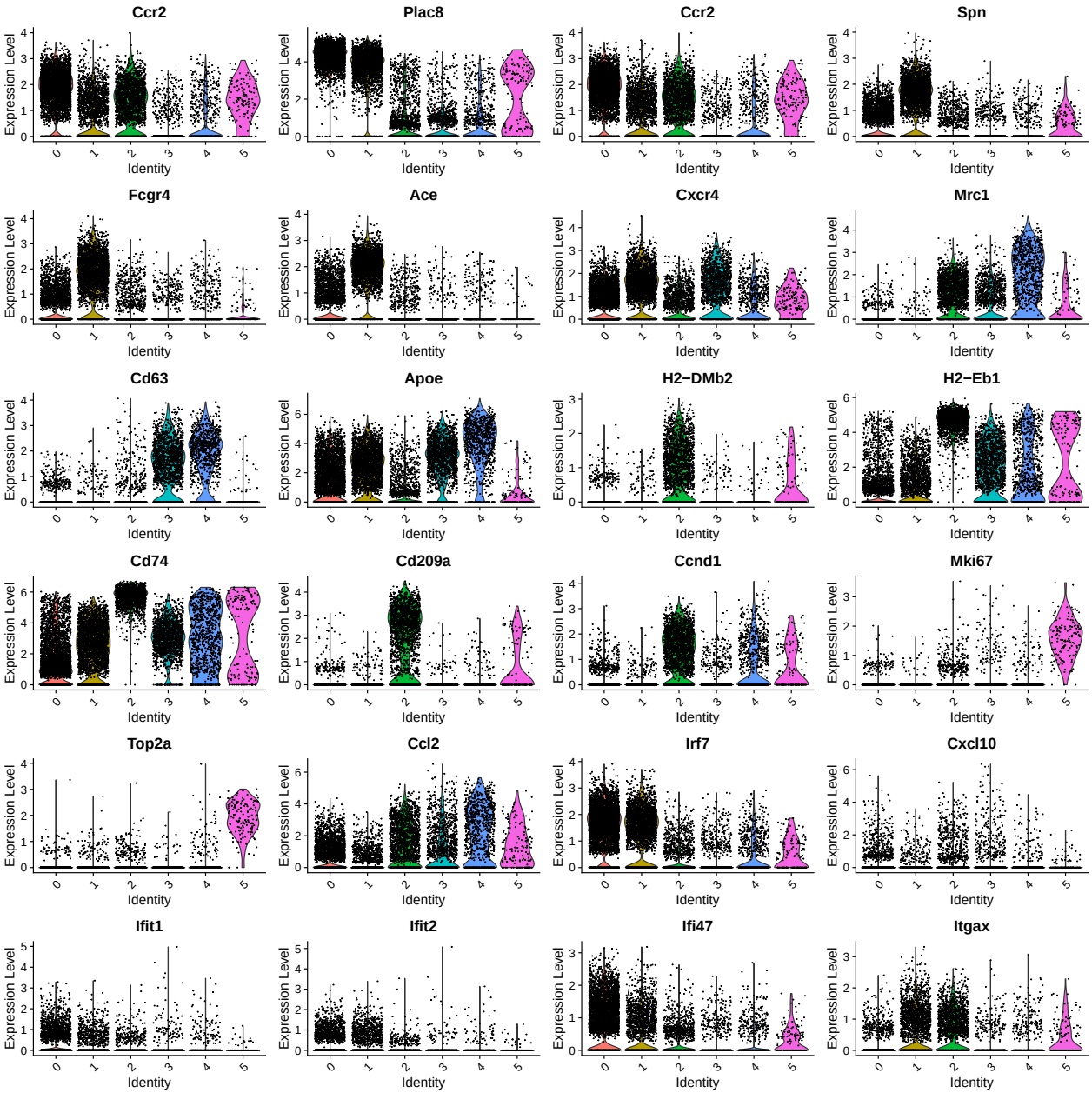
6.3 Population characterization

```
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()
```


0
1
2
3
4
5
Ly6c2
Chil3
Thbs1
Slpi
Hp
Tmsb10
Sell
Plac8
Mgst1
Ms4a4c
Lyz2
Isg15
Vcan
S100a6
Fn1
Prdx5
S100a4
Ifi211
Ccr2
Slfn1
Pglyrp1
Ace
Eno3
Fcgr4
Ly6i
Itgal
Gngt2
Dusp16
Adgre4
Spn
Treml4
Cd300e
Ly6a
Stap1
Fabp4
Lair1
Cd300ld
Tgm2
Ceacam1
Ear2
H2−Ab1
Cd209a
H2−Eb1
Cd74
H2−Aa
Ccl17
Mgl2
Tnip3
H2−DMb1
Mt1
Mt2
H2−DMa
Clec4b1
Etv3
Ccnd1
Ckb
AA467197
H2−DMb2
Retnla
Lyz1
Txnip
Jun
Egr1
Gadd45b
Ccl3
Cxcl2
Atf3
Cxcl16
Nfkbia
Sdc4
Cxcl1
Ier5
P2rx7
Ier2
Tank
Icam1
Ier3
Tnf
Rgs1
Gdf15
Pf4
C1qa
C1qc
Lyve1
Folr2
C1qb
Ccl7
Cbr2
Mrc1
Ccl8
Dab2
Ltc4s
Timp2
Cd63
Cd209f
Fcna
Apoe
Ccl12
Ccl2
Selenop
Pclaf
Top2a
Hist1h1b
Mki67
Ube2c
Birc5
Smc2
Stmn1
Hist1h2ae
Hist1h2ap
H2afx
Smc4
Hmgn2
Hmgb2
Tubb5
Hmgb1
Tuba1b
S100a8
Lcn2
S100a9

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)
```

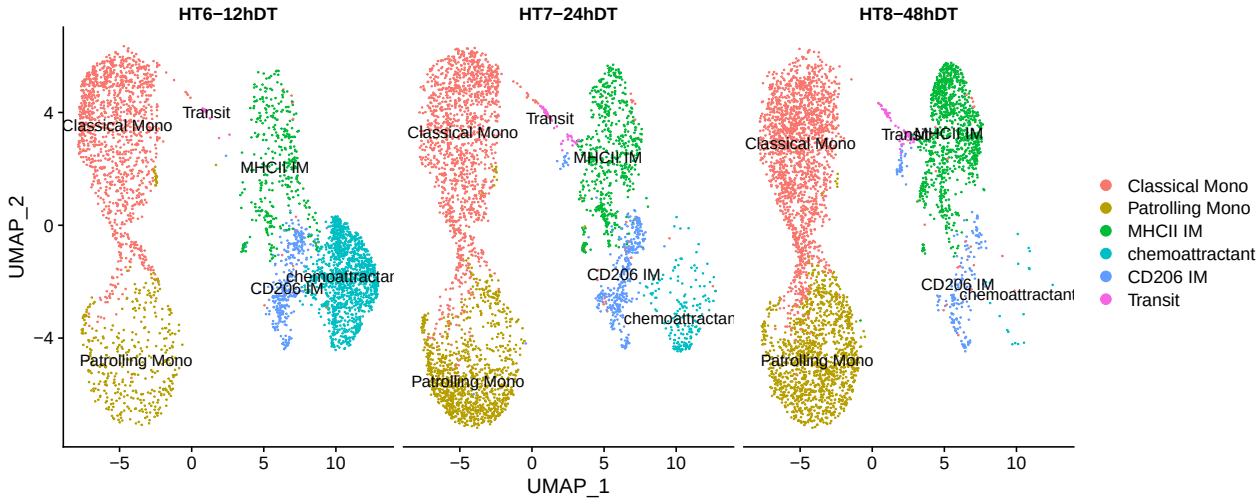


Here's the cell type annotation: cluster 0: Classical Mono cluster 1: Patrolling Mono cluster 2: MHCII IM cluster 3: chemoattractant cluster 4: CD206 IM cluster 5: Transit

```
levels(results)
```

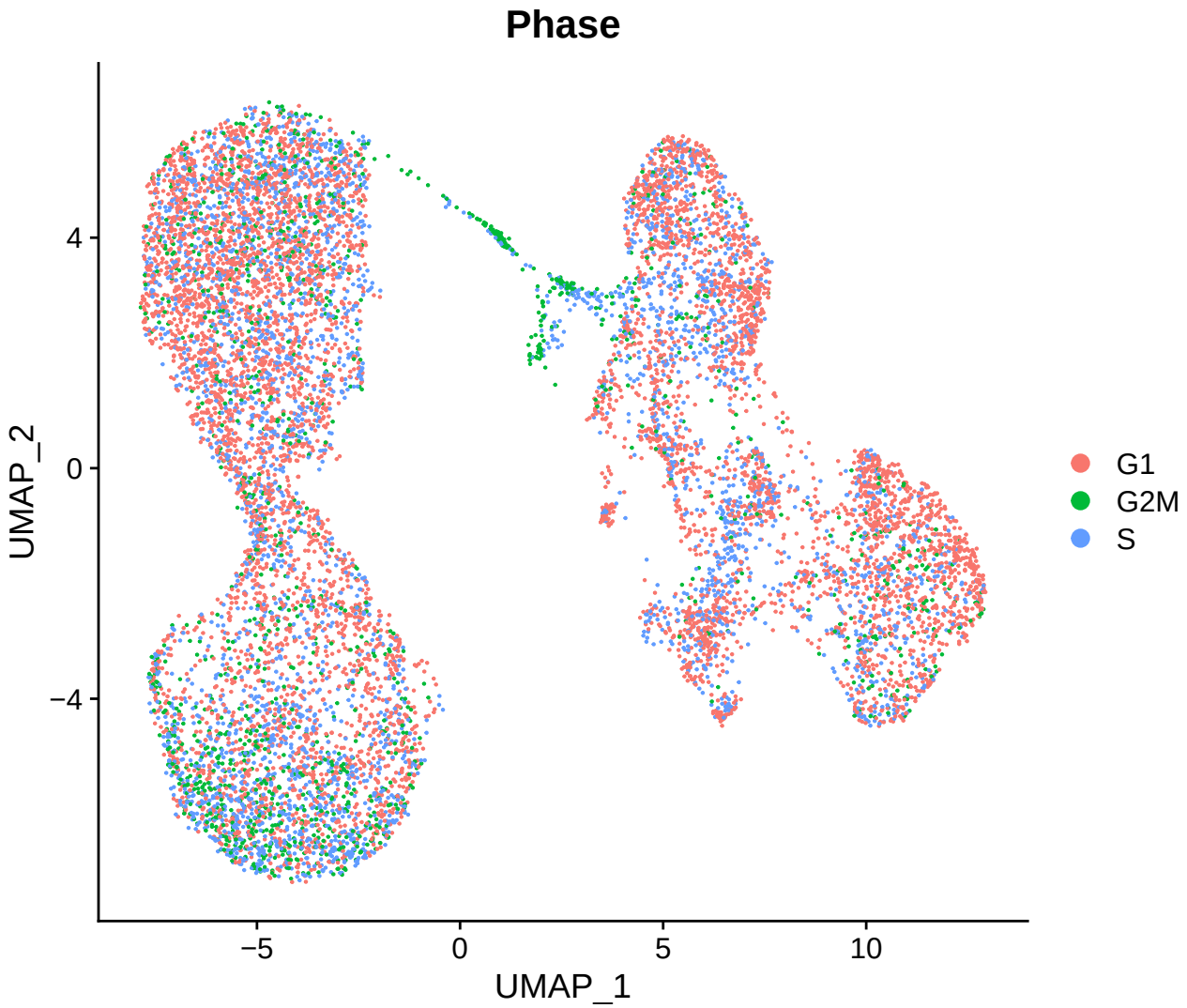
```
## [1] "0" "1" "2" "3" "4" "5"
```

```
results$cell.type2 <- factor(Idents(results), labels = c("Classical Mono",
    "Patrolling Mono", "MHCII IM", "chemoattractant", "CD206 IM", "Transit
    "))
Idents(results) <- "cell.type2"
DimPlot(results, label = TRUE, split.by = "treatment")
```

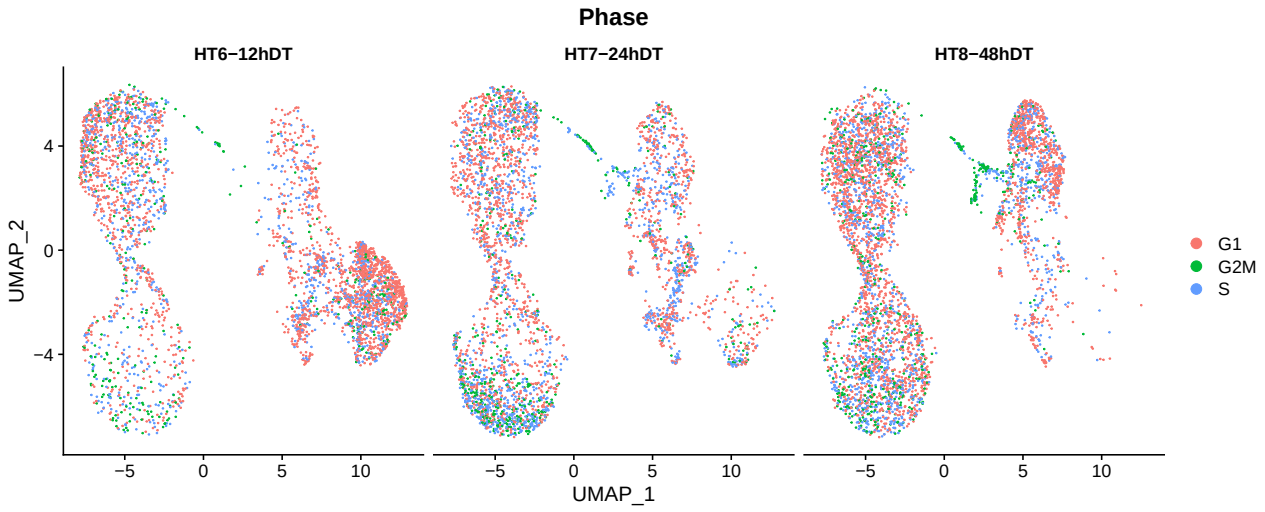


6.4 Cell cycle analysis

```
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")
```

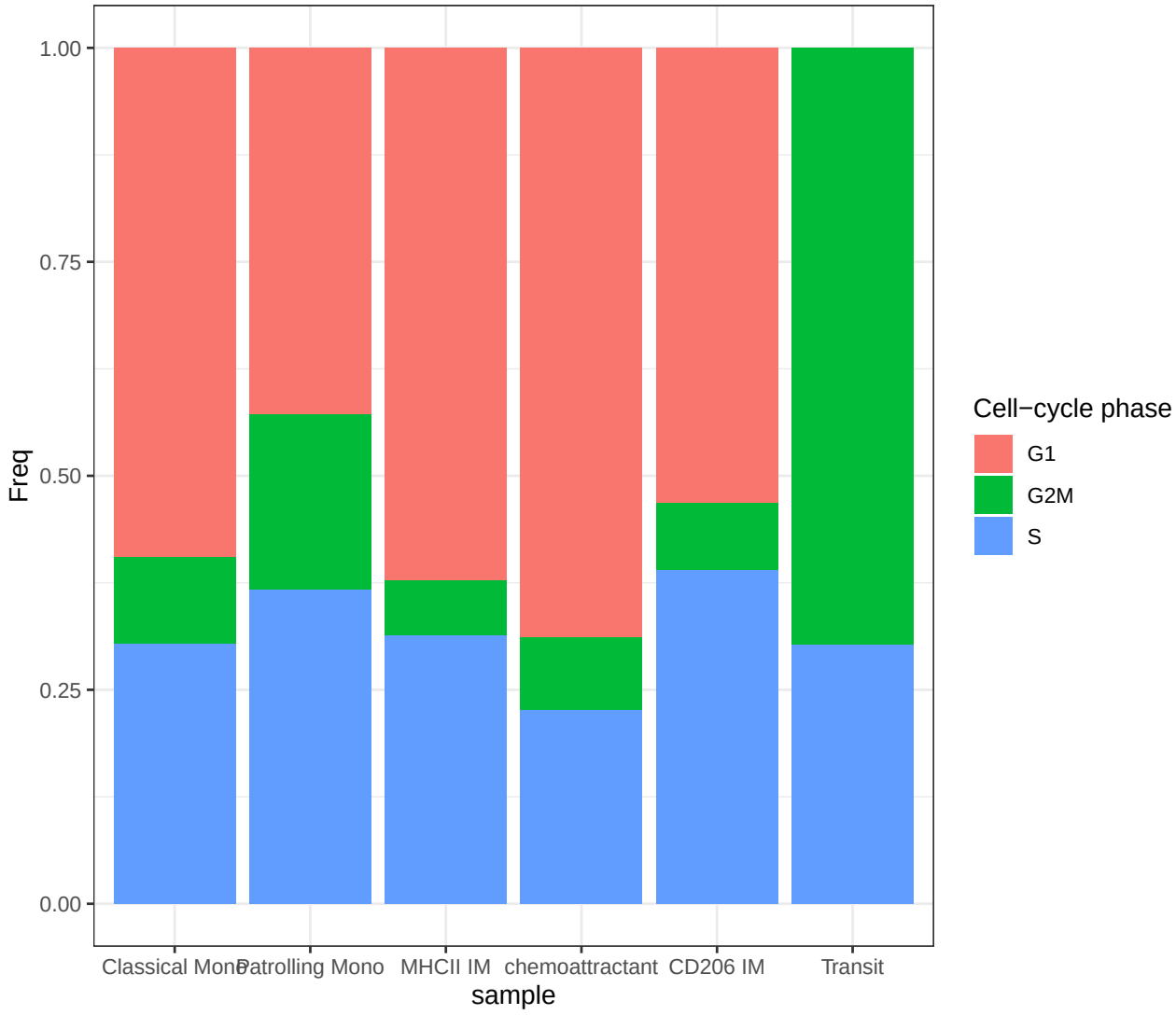



```
DimPlot(results, group.by = "Phase", reduction = "umap", split.by = "
    treatment")
```



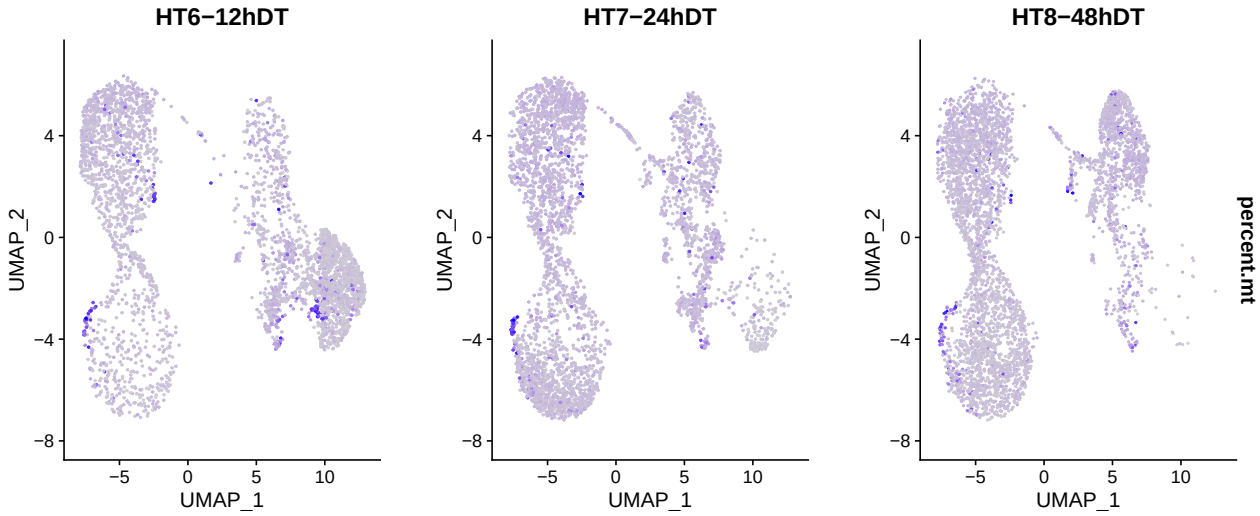
cluster 0: Classical Mono cluster 1: Patrolling Mono cluster 2: MHCII IM cluster 3: chemoattractant cluster 4: CD206 IM cluster 5: Transit

```
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 mitochodrial 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)
## 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
##  [3] celldex_1.0.0               SingleR_1.4.1
```

```
##  [5] SummarizedExperiment_1.20.0 Biobase_2.50.0
##  [7] GenomicRanges_1.42.0        GenomeInfoDb_1.26.7
##  [9] IRanges_2.24.1              S4Vectors_0.28.1
## [11] BiocGenerics_0.36.1         MatrixGenerics_1.2.1
## [13] matrixStats_0.61.0          dplyr_1.0.7
## [15] ggpubr_0.4.0                ggplot2_3.3.5
## [17] SeuratObject_4.0.4          Seurat_4.0.5
##
## loaded via a namespace (and not attached):
##   [1] utf8_1.2.2                    reticulate_1.22
##   [3] tidyselect_1.1.1              RSQLite_2.2.9
##   [5] AnnotationDbi_1.52.0          htmlwidgets_1.5.4
##   [7] grid_4.0.3                    BiocParallel_1.24.1
##   [9] Rtsne_0.15                    pROC_1.18.0
##  [11] munsell_0.5.0                 codetools_0.2-18
##  [13] ica_1.0-2                     future_1.23.0
##  [15] miniUI_0.1.1.1                withr_2.4.3
##  [17] colorspace_2.0-2              highr_0.9
##  [19] knitr_1.36                    rstudioapi_0.13
##  [21] ROCR_1.0-11                   ggsignif_0.6.3
##  [23] tensor_1.5                    listenv_0.8.0
##  [25] labeling_0.4.2                GenomeInfoDbData_1.2.4
##  [27] polyclip_1.10-0               bit64_4.0.5
##  [29] farver_2.1.0                  parallelly_1.29.0
##  [31] vctrs_0.3.8                   generics_0.1.1
##  [33] ipred_0.9-12                  xfun_0.28
##  [35] BiocFileCache_1.14.0          randomForest_4.6-14
##  [37] R6_2.5.1                      rsvd_1.0.5
##  [39] bitops_1.0-7                  spatstat.utils_2.2-0
##  [41] cachem_1.0.6                  DelayedArray_0.16.3
##  [43] assertthat_0.2.1              promises_1.2.0.1
##  [45] scales_1.1.1                  nnet_7.3-14
##  [47] gtable_0.3.0                  beachmat_2.6.4
##  [49] globals_0.14.0                goftest_1.2-3
##  [51] timeDate_3043.102             rlang_0.4.12
##  [53] splines_4.0.3                 rstatix_0.7.0
##  [55] lazyeval_0.2.2                ModelMetrics_1.2.2.2
##  [57] checkmate_2.0.0               spatstat.geom_2.3-0
##  [59] broom_0.7.10                  BiocManager_1.30.16
##  [61] yaml_2.2.1                    reshape2_1.4.4
##  [63] abind_1.4-5                   backports_1.4.0
##  [65] httpuv_1.6.3                  Hmisc_4.6-0
##  [67] DiagrammeR_1.0.6.1            caret_6.0-90
##  [69] lava_1.6.10                   tools_4.0.3
##  [71] ellipsis_0.3.2                spatstat.core_2.3-2
##  [73] proxy_0.4-26                  ggridges_0.5.3
##  [75] Rcpp_1.0.7                    plyr_1.8.6
##  [77] base64enc_0.1-3               visNetwork_2.1.0
##  [79] sparseMatrixStats_1.2.1       zlibbioc_1.36.0
##  [81] purrr_0.3.4                   RCurl_1.98-1.5
##  [83] rpart_4.1-15                  deldir_1.0-6
##  [85] pbapply_1.5-0                 zoo_1.8-9
##  [87] nichenetr_1.0.0               ggrepel_0.9.1
##  [89] cluster_2.1.0                 magrittr_2.0.1
```

```
##  [91] data.table_1.14.2            RSpectra_0.16-0                  78
##  [93] scattermore_0.7              lmtest_0.9-39                    79
##  [95] RANN_2.6.1                   fitdistrplus_1.1-6               80
##  [97] hms_1.1.1                    patchwork_1.1.1                  81
##  [99] mime_0.12                    evaluate_0.14                    82
## [101] xtable_1.8-4                 jpeg_0.1-9                       83
## [103] gridExtra_2.3                compiler_4.0.3                   84
## [105] tibble_3.1.6                 KernSmooth_2.23-20               85
## [107] crayon_1.4.2                 htmltools_0.5.2                  86
## [109] tzdb_0.2.0                   mgcv_1.8-33                      87
## [111] later_1.3.0                  Formula_1.2-4                    88
## [113] tidyr_1.1.4                  lubridate_1.8.0                  89
## [115] DBI_1.1.1                    ExperimentHub_1.16.1             90
## [117] dbplyr_2.1.1                 MASS_7.3-53                      91
## [119] rappdirs_0.3.3               readr_2.1.1                      92
## [121] Matrix_1.3-4                 car_3.0-12                       93
## [123] gower_0.2.2                  igraph_1.2.9                     94
## [125] pkgconfig_2.0.3              foreign_0.8-81                   95
## [127] plotly_4.10.0                spatstat.sparse_2.0-0            96
## [129] recipes_0.1.17               foreach_1.5.1                    97
## [131] XVector_0.30.0               prodlim_2019.11.13               98
## [133] stringr_1.4.0                digest_0.6.29                    99
## [135] sctransform_0.3.2            RcppAnnoy_0.0.19                 100
## [137] spatstat.data_2.1-0          rmarkdown_2.11                   101
## [139] leiden_0.3.9                 htmlTable_2.3.0                  102
## [141] uwot_0.1.11                  DelayedMatrixStats_1.12.3        103
## [143] curl_4.3.2                   shiny_1.7.1                      104
## [145] lifecycle_1.0.1              nlme_3.1-153                     105
## [147] jsonlite_1.7.2               carData_3.0-4                    106
## [149] BiocNeighbors_1.8.2          viridisLite_0.4.0                107
## [151] limma_3.46.0                 fansi_0.5.0                      108
## [153] pillar_1.6.4                 lattice_0.20-41                  109
## [155] fastmap_1.1.0                httr_1.4.2                       110
## [157] survival_3.2-7               interactiveDisplayBase_1.28.0    111
## [159] glue_1.5.1                   fdrtool_1.2.17                   112
## [161] png_0.1-7                    iterators_1.0.13                 113
## [163] BiocVersion_3.12.0           bit_4.0.4                        114
## [165] class_7.3-17                 stringi_1.7.6                    115
## [167] blob_1.2.2                   BiocSingular_1.6.0               116
## [169] AnnotationHub_2.22.1         caTools_1.18.2                   117
## [171] latticeExtra_0.6-29          memoise_2.0.1                    118
## [173] e1071_1.7-9                  irlba_2.3.5                      119
## [175] future.apply_1.8.1                                            120
```

References

Aran, D., Looney, A.P., Liu, L., Wu, E., Fong, V., Hsu, A., Chak, S., Naikawadi, R.P., Wolters, P.J., Abate, A.R., et al. (2019). Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. Nature Immunology *20*, 163–172.

Hao, Y., Hao, S., Andersen-Nissen, E., III, W.M.M., Zheng, S., Butler, A., Lee, M.J., Wilk, A.J., Darby, C., Zagar, M., et al. (2021). Integrated analysis of multimodal single-cell data. Cell.