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

4-Compare Refilled IMs in Day4 depletion to control

2022-01-28 13:20:16 +0100

Abstract

Resident tissue macrophages (RTM) are differentiated immune cells populating distinct niches and exhibiting important tissue-supportive functions. RTM maintenance is thought to 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

We established an in vivo inducible IM-depletion model by generating the IM-DTR mouse line. The IM in these mice expressed specifically the diphtheria toxin receptor (DTR) and 50 ng diphtheria toxin (DT) treatment was showed to be effective to deplete the lung IM population after 24 hours. Then IM population was regenerated from D3. On D7 post-treatment the IM population could recover to a similar size of intact lung IM population.

In this analysis, the lung IM from D4 post-treatment were compared to the intact IM of littermate control mice without DT treatment using 10X single-cell RNAsequencing.

Data processing, cell clustering, linear- or non-linear dimension reduction were made in Seurat package (Hao et al., 2021) and the celltyping was made by SingleR pakcage (Aran et al., 2019). Cells were firstly clustered with high resolution in order that the contaminated cells would be clustered together and identified with SingleR package. Once the contaminated cells removed, data of the filtered cells were then re-filtered and clustered for further analyses.

2 Load data and packages

```
library(Seurat)
source("../R/seurat.make.integrated.R")

initiation.analysis.folder <- "../3-scRNAseq_initiation_the_IMs_in_Day4_depletion/"
file.names <- list.files(initiation.analysis.folder, pattern = "*.rds")
sample.names <- sub(".rds", "", file.names)

table.samples <- data.frame(sample.names, file.names)

i <- sapply(table.samples, is.factor)
table.samples[i] <- lapply(table.samples[i], as.character) # this is for convert the factor to character.

rm("file.names", "sample.names") # remove individual vector to avoid confusion.

for (i in 1:length(table.samples$sample.names)) {
  assign(table.samples$sample.names[i], readRDS(file = file.path(
     initiation.analysis.folder, table.samples$file.names[i])))
}
```

3 Build up sample metadata

Here's sample order:

```
table.samples$sample.names
```

```
## [1] "CPlus_NGS20_Q147.seuratObject"    "Plusplus_NGS20_Q148.seuratObject"
```

Make metadata table

```
# sample.names
```

```
sample.metadata <- data.frame(
  group = c("Cre+", "++"),
  cell.type = rep("lung IM niche (including CD11c)", length(table.samples$
     sample.names)),  # nolint
  origin = rep("lung", length(table.samples$sample.names)),
  sample_prefix = gsub("_NGS[a-zA-Z0-9_.-]*", "", table.samples$sample.
     names))
rownames(sample.metadata) <- table.samples$sample.names
sample.metadata
```

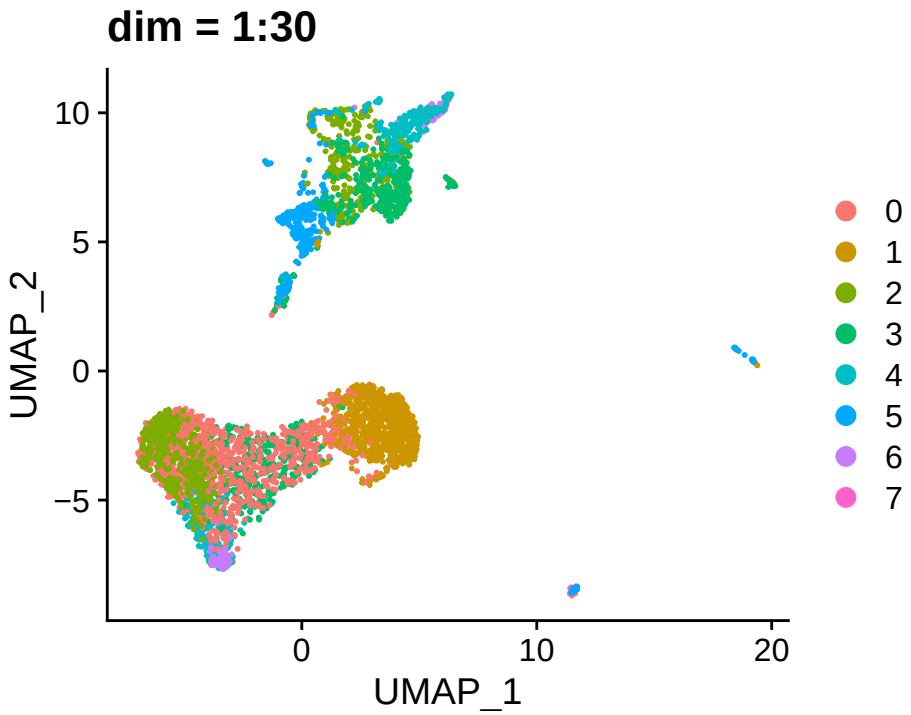
```
## # A tibble: 2 x 4
##   group cell.type                       origin sample_prefix
##   <chr> <chr>                           <chr>  <chr>
## 1 Cre+  lung IM niche (including CD11c) lung   CPlus
## 2 ++    lung IM niche (including CD11c) lung   Plusplus
```

```
# add sample information:

for (i in table.samples$sample.names) {
  obj <- get(i)
  obj$treatment <- sample.metadata[i, ]$group
  assign(i, obj)
}

for (i in table.samples$sample.names) {
  obj <- get(i)
  obj$therapy <- sample.metadata[i, ]$cell.type
  assign(i, obj)
}

for (i in table.samples$sample.names) {
  obj <- get(i)
  obj$therapy <- sample.metadata[i, ]$origin
  assign(i, obj)
}


# add pre-fix to cellnames:

for (i in table.samples$sample.names) {
  obj <- get(i)
  obj <- RenameCells(obj,
    add.cell.id = sample.metadata[i, ]$sample_prefix)
  assign(i, obj)
}
```

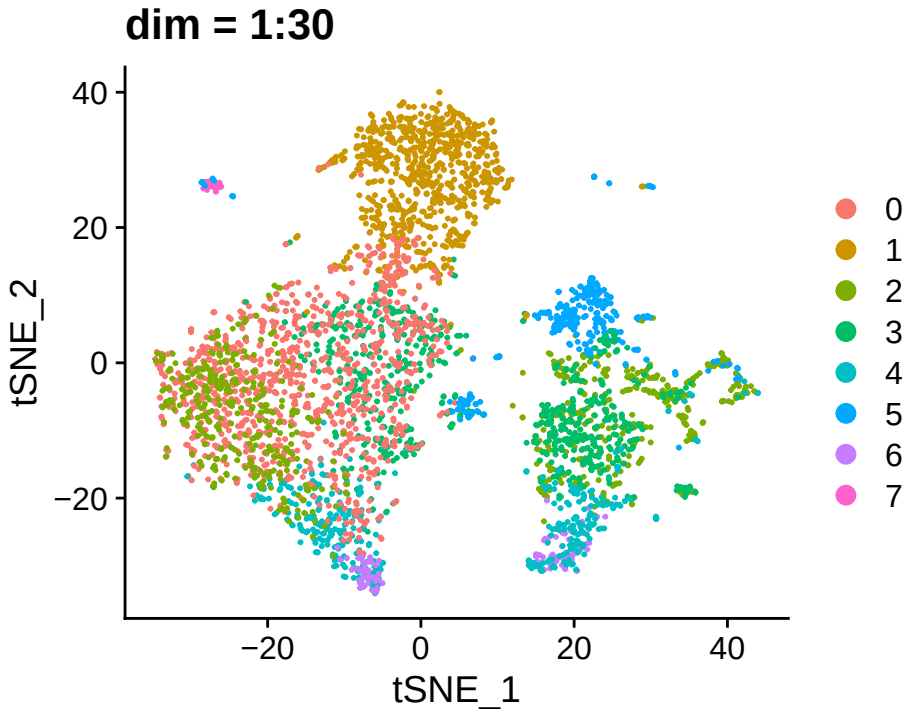
Integrate:

```
results.dim30 <- seurat.make.integrated(
  seuratObjects = sapply(table.samples$sample.names, get),
  prefix = table.samples$sample.names, dimensionality = 1:30,
  SCTransf = FALSE)
```

```
results.dim30$plots$plot.umap + ggtitle("dim = 1:30")
```




```
results.dim30$plots$plot.tsne + ggtitle("dim = 1:30")
```




```
results <- results.dim30$integrated.seuratObject
```

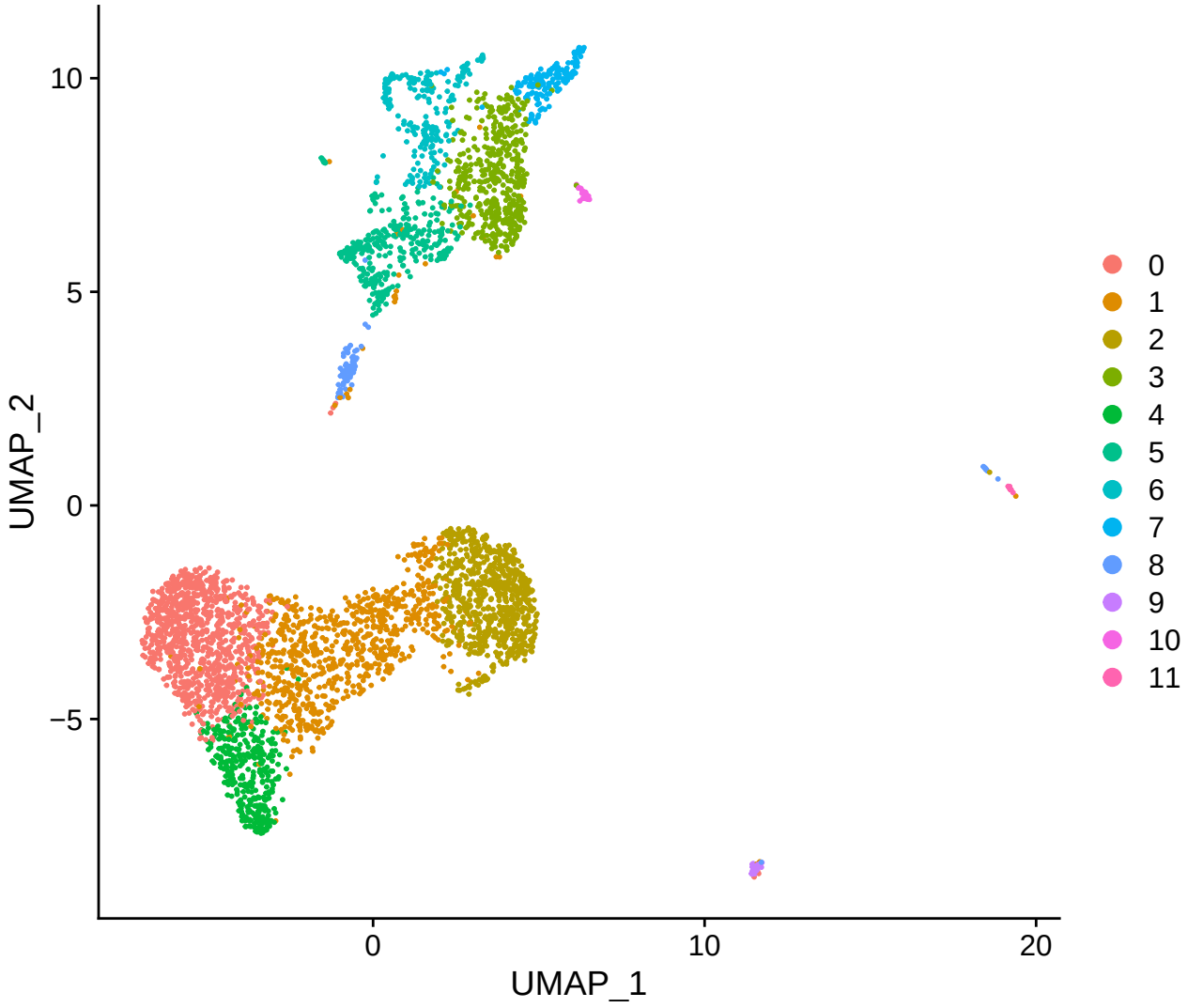
4 Celltype identification

4.1 Clustering cells

```
results <- FindClusters(results, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 4169
## Number of edges: 161254
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8723
## Number of communities: 12
## Elapsed time: 0 seconds
```

```
DimPlot(results)
```




```
# save data
```

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

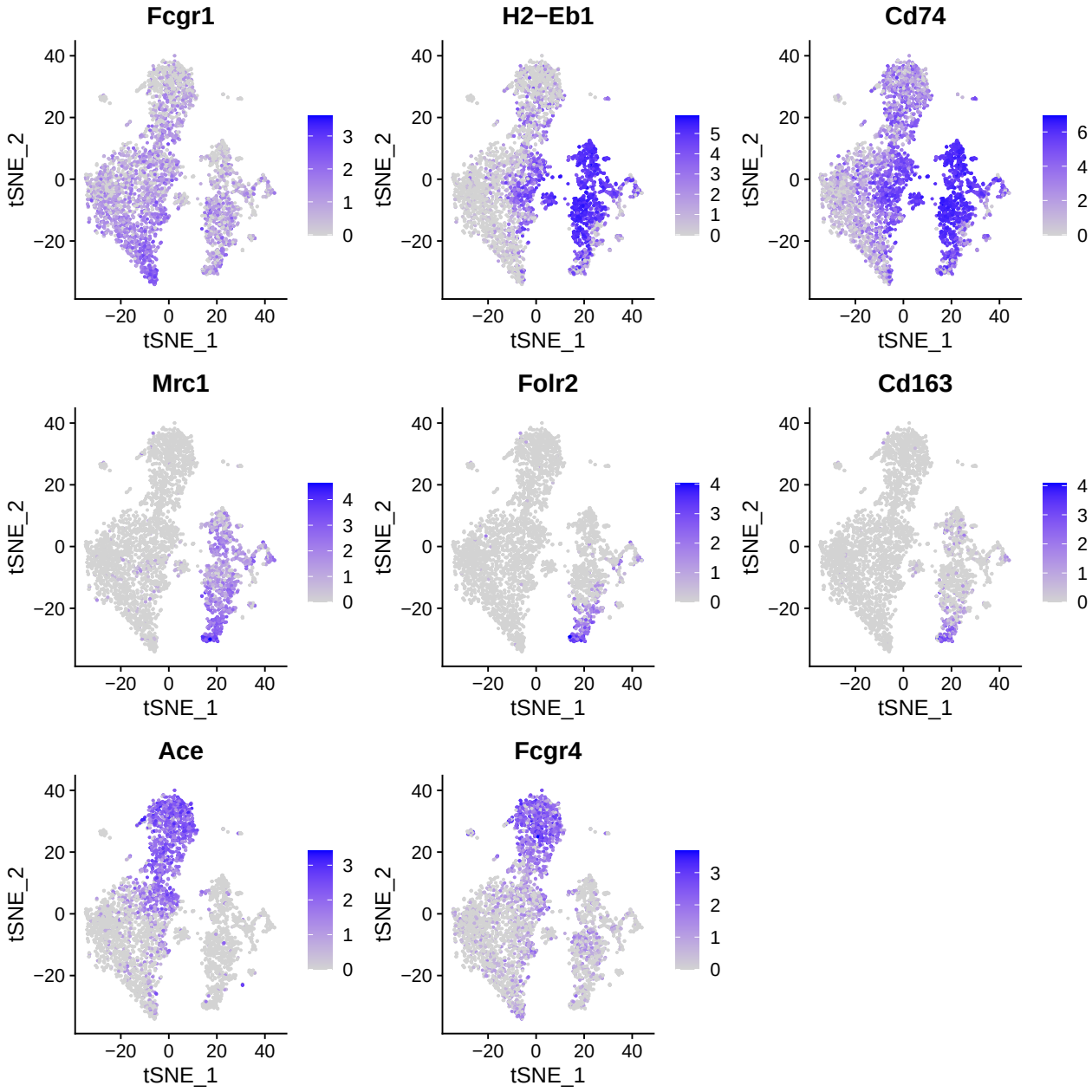
4.2 Identify cells with SingleR

```
source("../R/seurat2singleR.R")
Idents(results) <- "integrated_snn_res.0.5"
results.singleR <- seurat2singleR(results, ref = "ImmGenData")
```

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

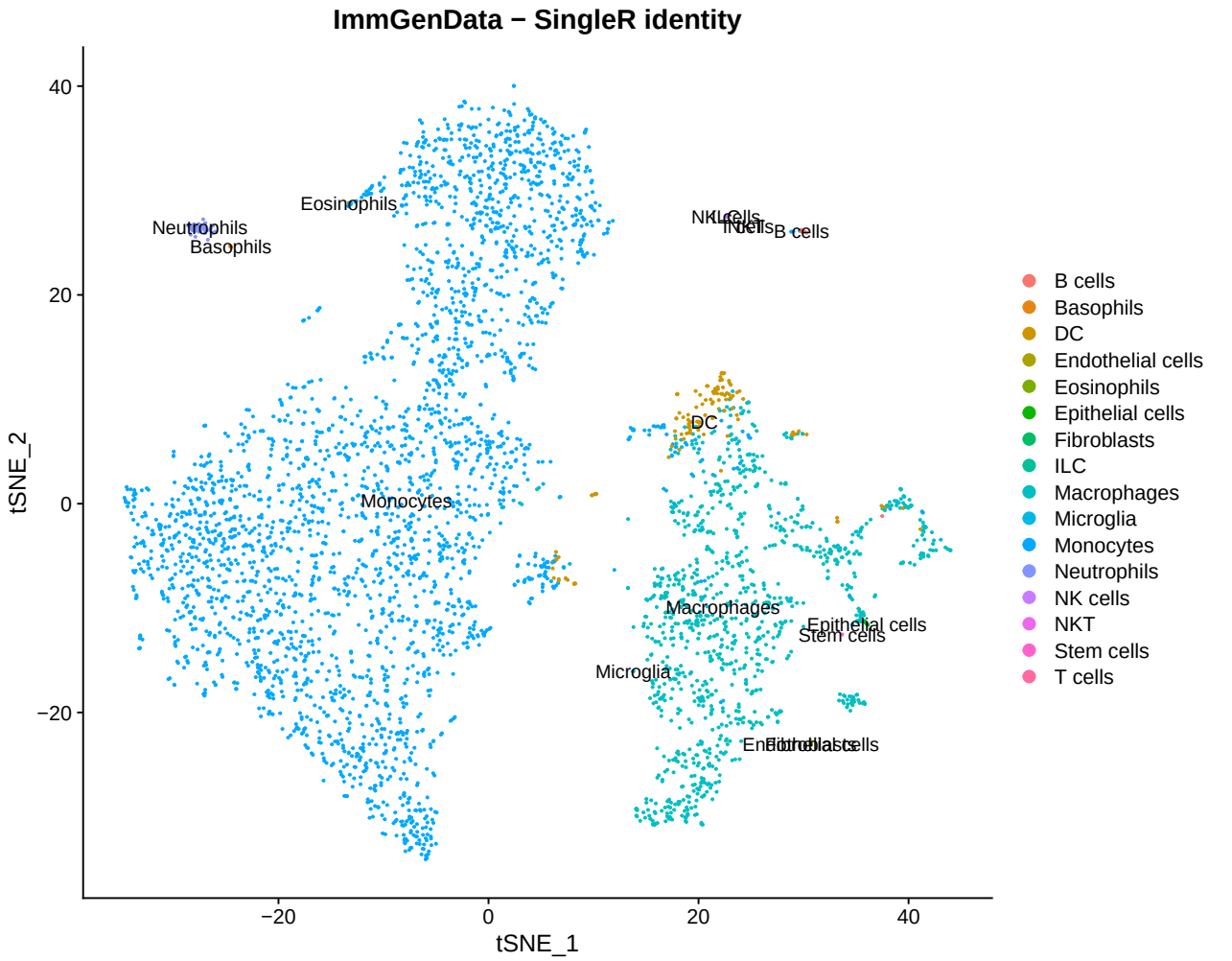
4.3 Identify cells with marker expression

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

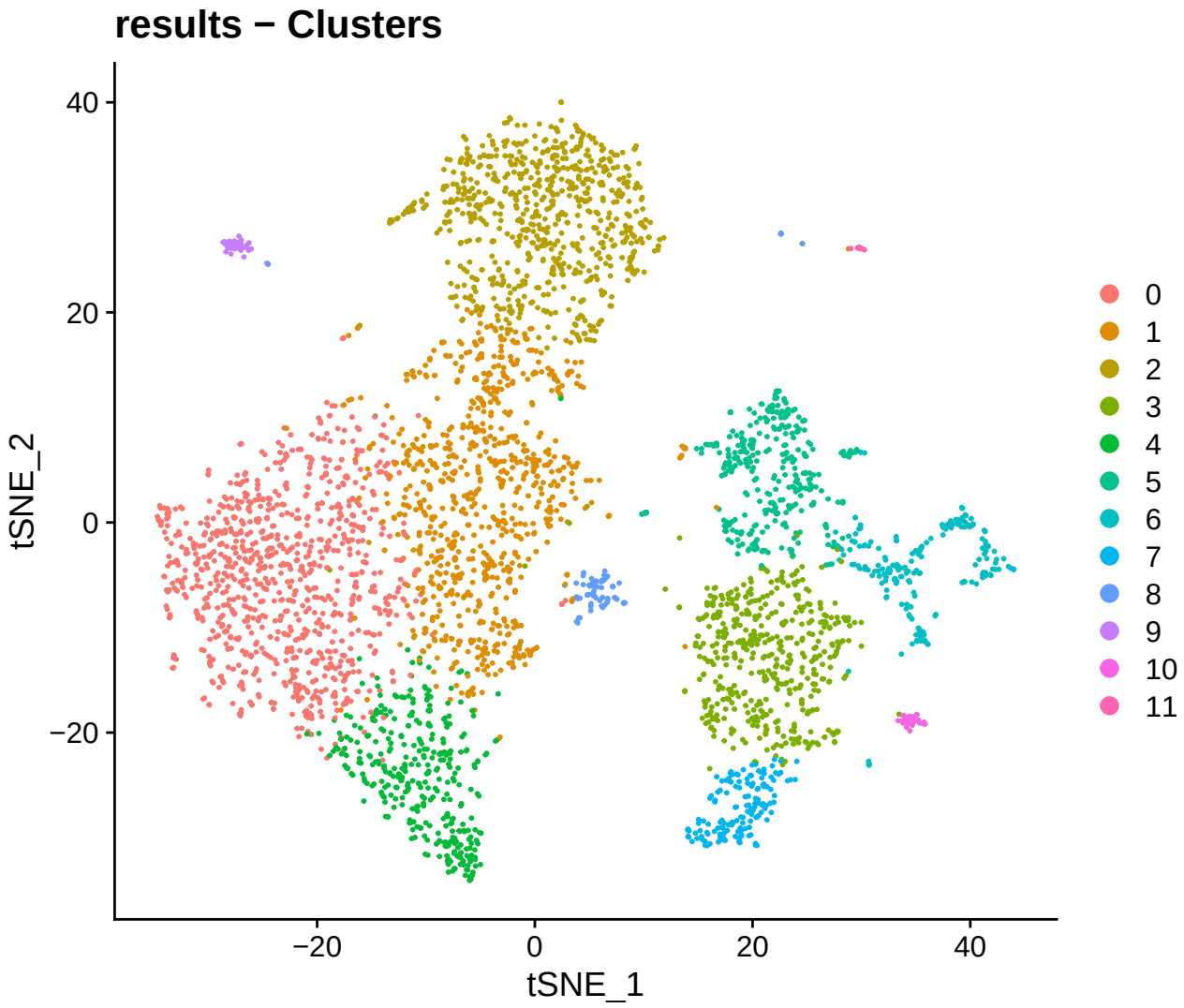


4.4 Combine SingleR and marker expression results

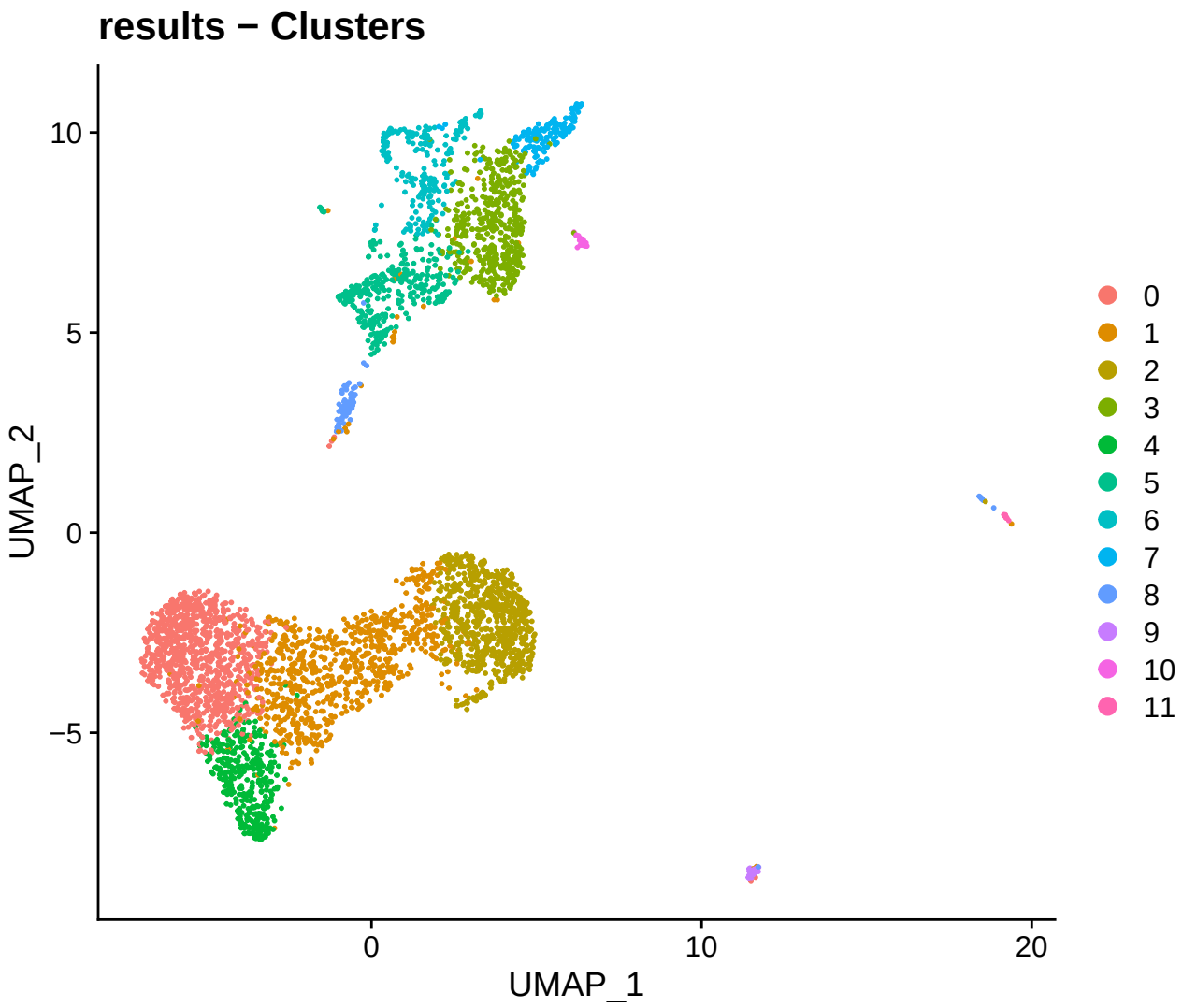
```
results$singleR.celltype <- results.singleR$labels
DimPlot(results, group.by = "singleR.celltype", reduction = "tsne", label
    = T) + ggtitle("ImmGenData␣-␣SingleR␣identity")
```



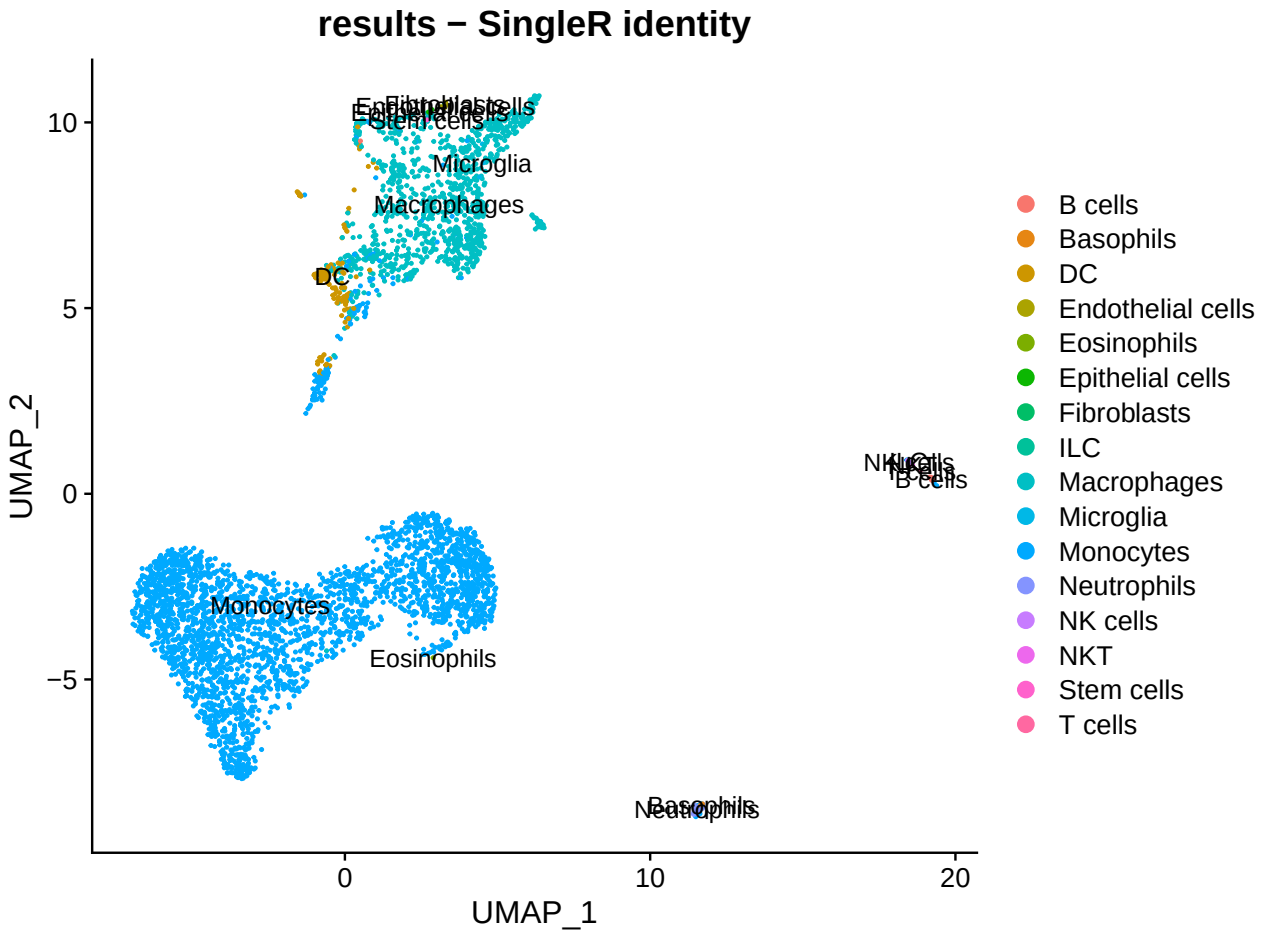

```
DimPlot(results, reduction = "tsne") + ggtitle("results - Clusters")
```



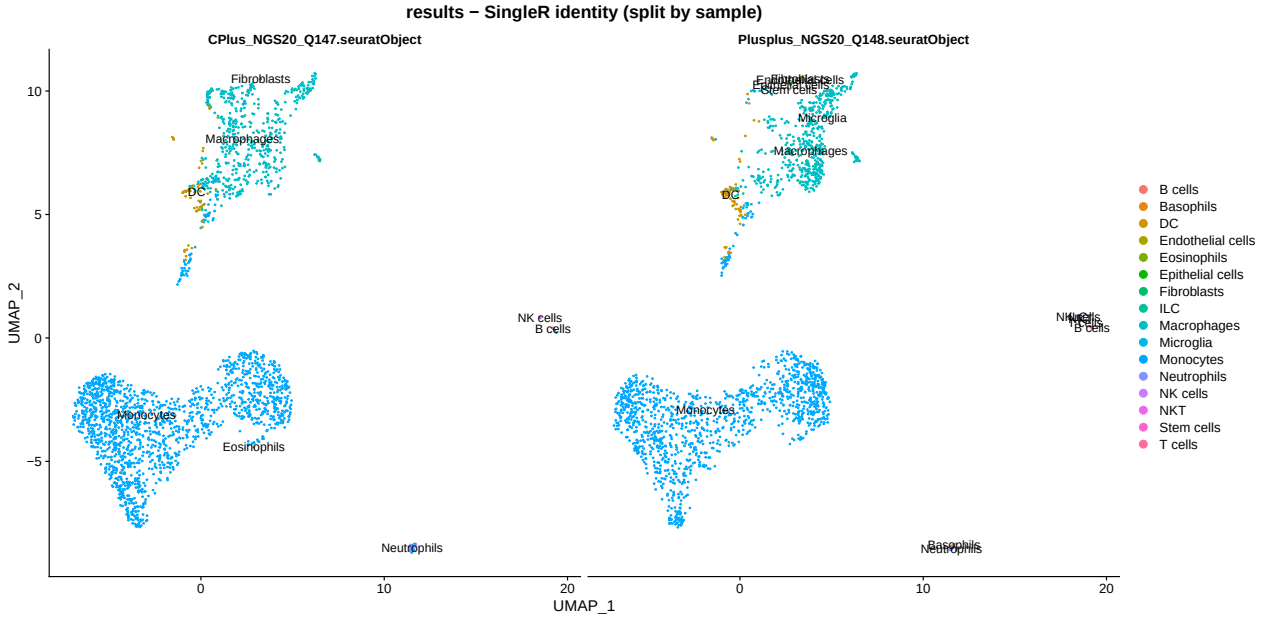

```
DimPlot(results, reduction = "umap") + ggtitle("results - Clusters")
```




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




```
DimPlot(results, group.by = "singleR.celltype", reduction = "umap", label
    = T, split.by = "object_before_integrated", ncol = 2) + ggtitle("
    results - SingleR identity (split by sample)")
```



5 Remove other cell types than monocytes/macrophages

Cell numbers per cell type:

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

```
##
##           B cells         Basophils                DC Endothelial cells
##                16                 3               126                   6
##       Eosinophils  Epithelial cells       Fibroblasts               ILC
##                 1                 3                 2                 2
##       Macrophages          Microglia         Monocytes       Neutrophils
##              1034                 3              2921                42
##          NK cells               NKT        Stem cells           T cells
##                 7                 1                 1                 1
```

Keep only "Monocytes," "Macrophages" and "Microglia." As microglia are highly similar to lung IMs and we can exclude the possibility of contamination of brain cells in this experiement, we should keep them for the following analysis.

```
results <- results[, WhichCells(results, expression = singleR.celltype %in
   % c("Monocytes", "Macrophages", "Microglia"))]
```

6 Phenotyping of lung monocytes/IMs

6.1 Clean data after filtering

Remove old snn:

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

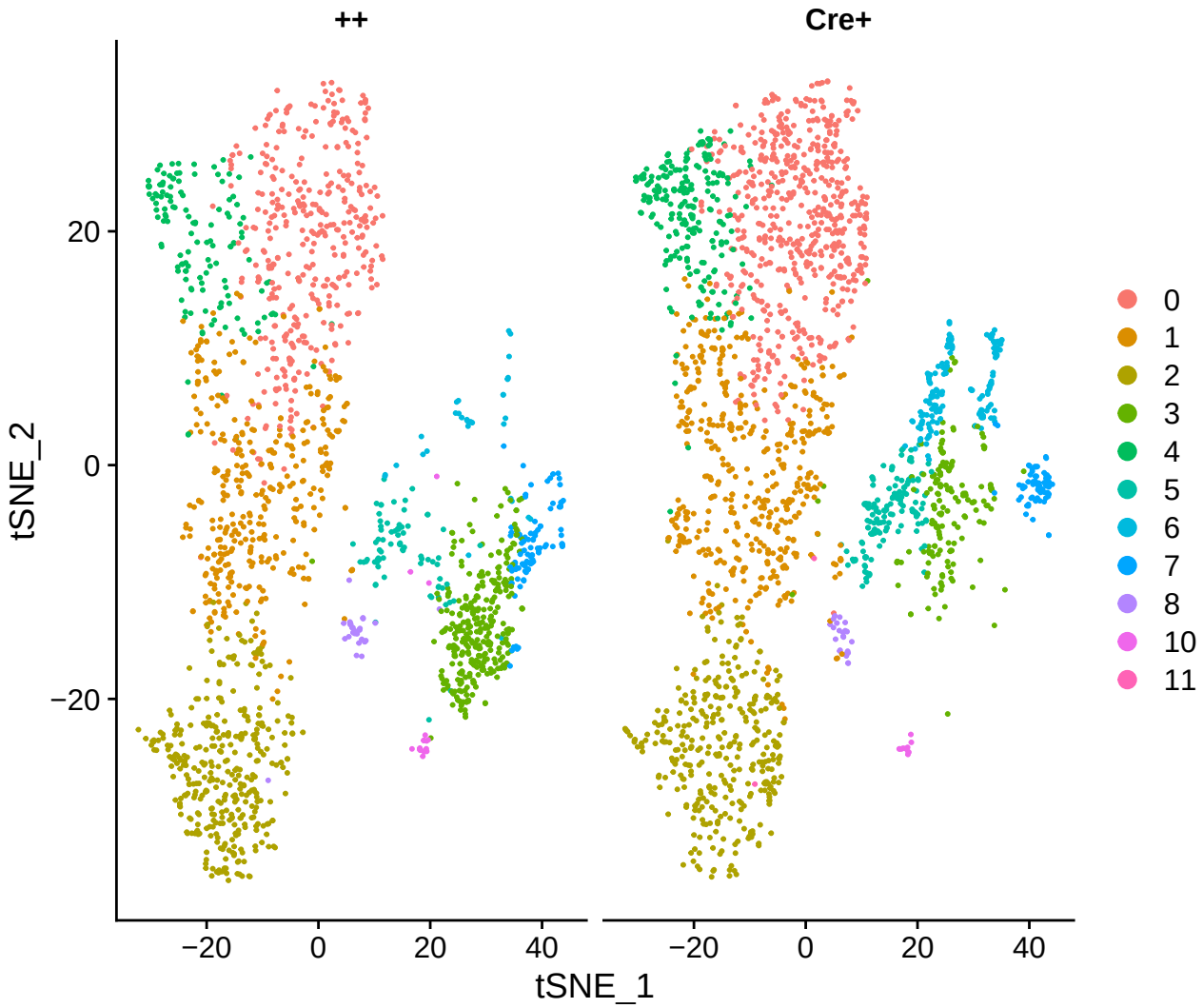
Save data

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

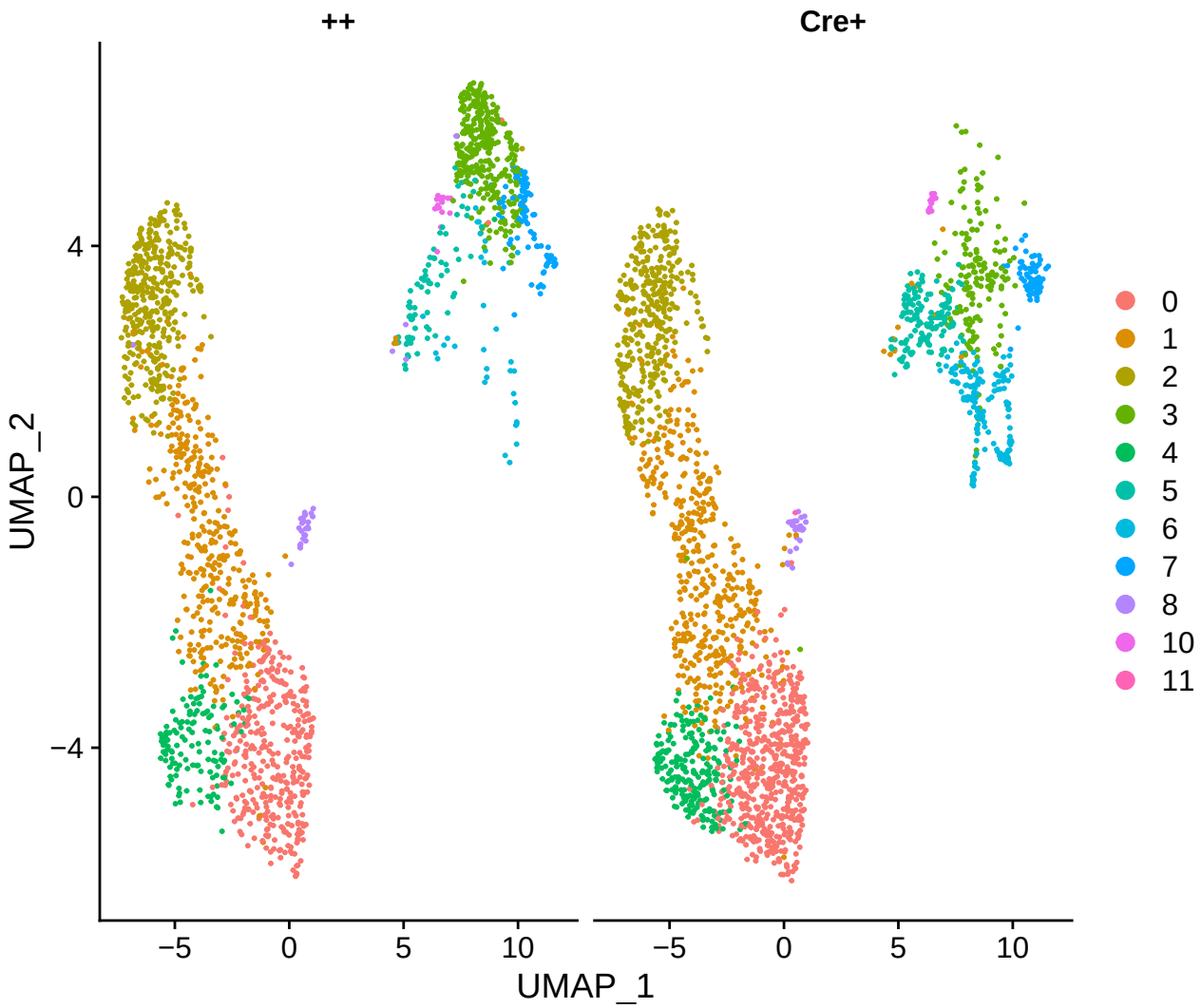
6.2 Data re-processing

```
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))
results <- RunTSNE(results, dims = 1:20)
results <- RunUMAP(results, dims = 1:20)
```

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




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

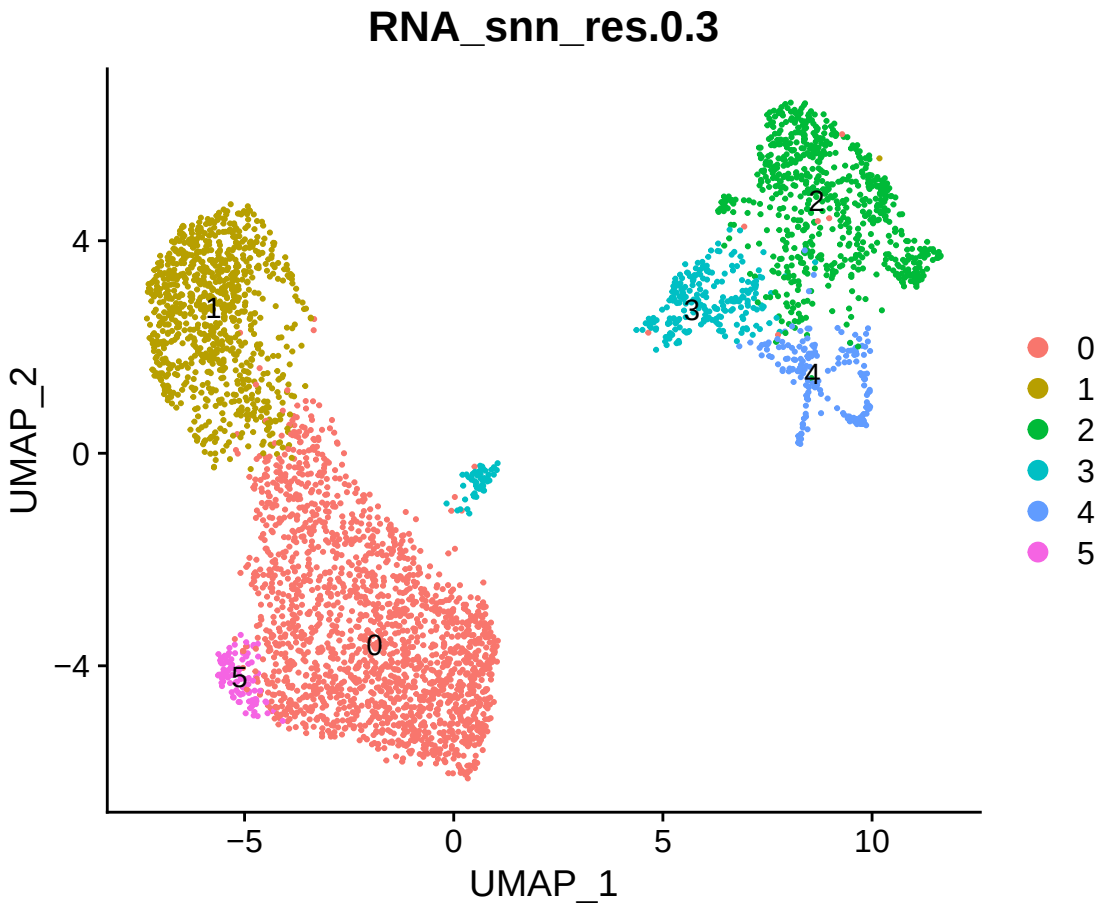


Clustering of cells

```
  results <- FindNeighbors(results, dims = 1:20)
  results <- FindClusters(results, resolution = 0.3)
```

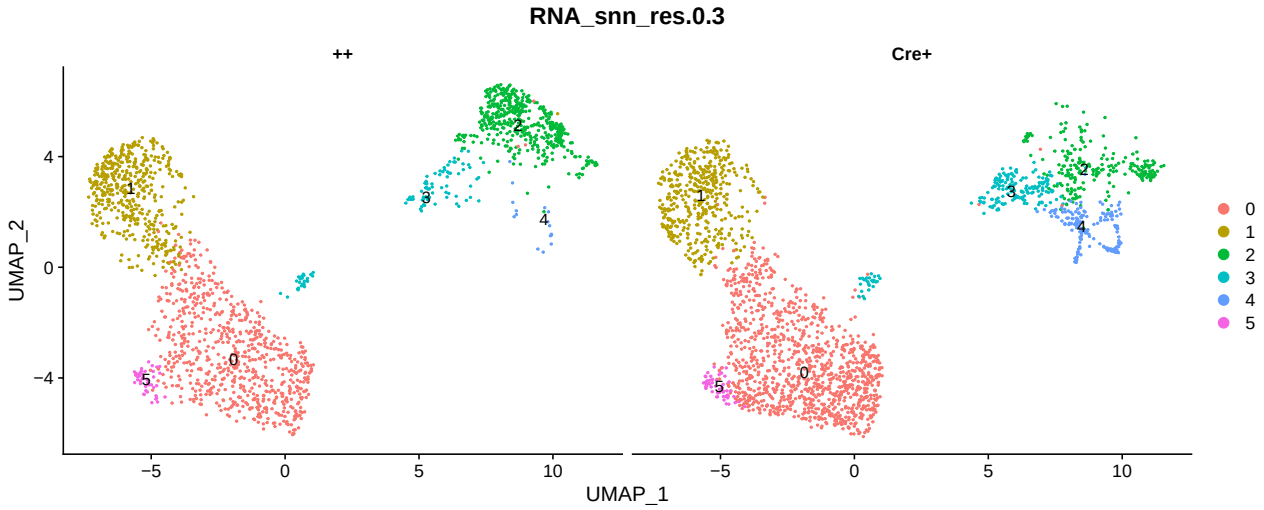
```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 3958
## Number of edges: 139307
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8872
## Number of communities: 6
## Elapsed time: 0 seconds
```

```
Idents(results) <- "RNA_snn_res.0.3"
DimPlot(results, group.by = "RNA_snn_res.0.3", label = TRUE)
```



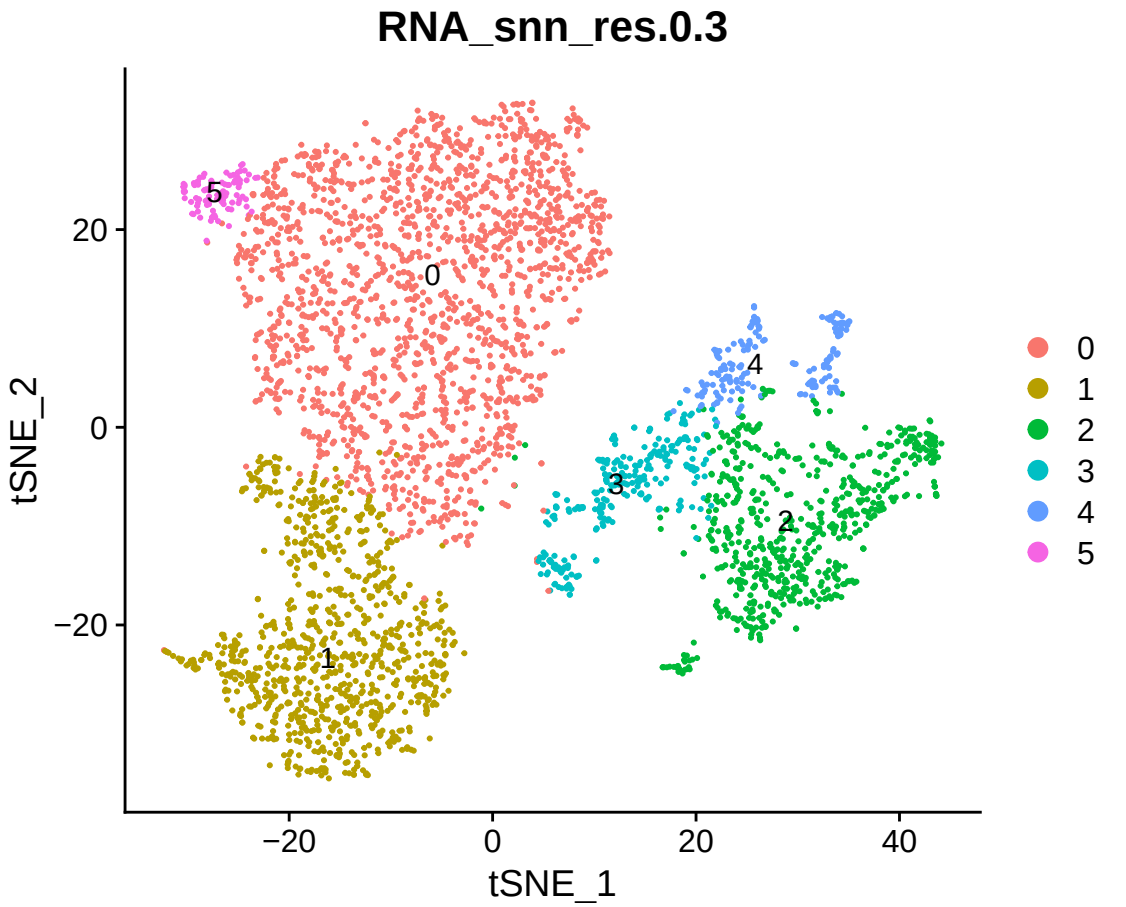

```
DimPlot(results, split.by = "treatment", group.by = "RNA_snn_res.0.3",
    label = TRUE)
```

1

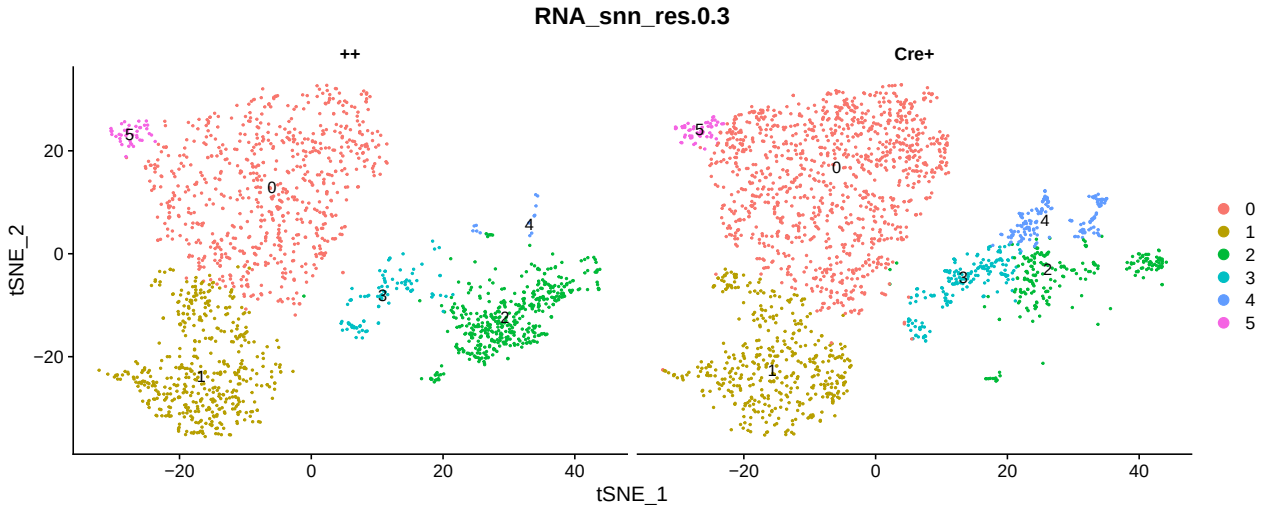



```
DimPlot(results, group.by = "RNA_snn_res.0.3", reduction = "tsne", label =
    TRUE)
```

1



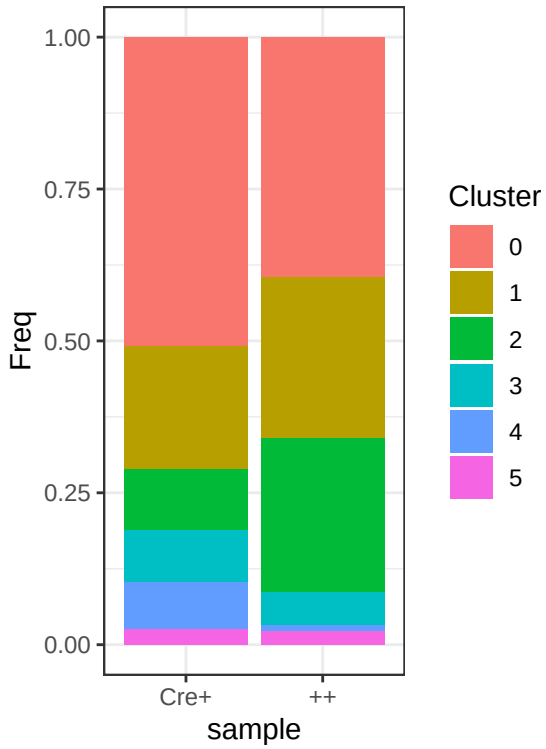

```
DimPlot(results, split.by = "treatment", group.by = "RNA_snn_res.0.3",
   reduction = "tsne", label = TRUE)
```



Distribution of each cluster across the samples

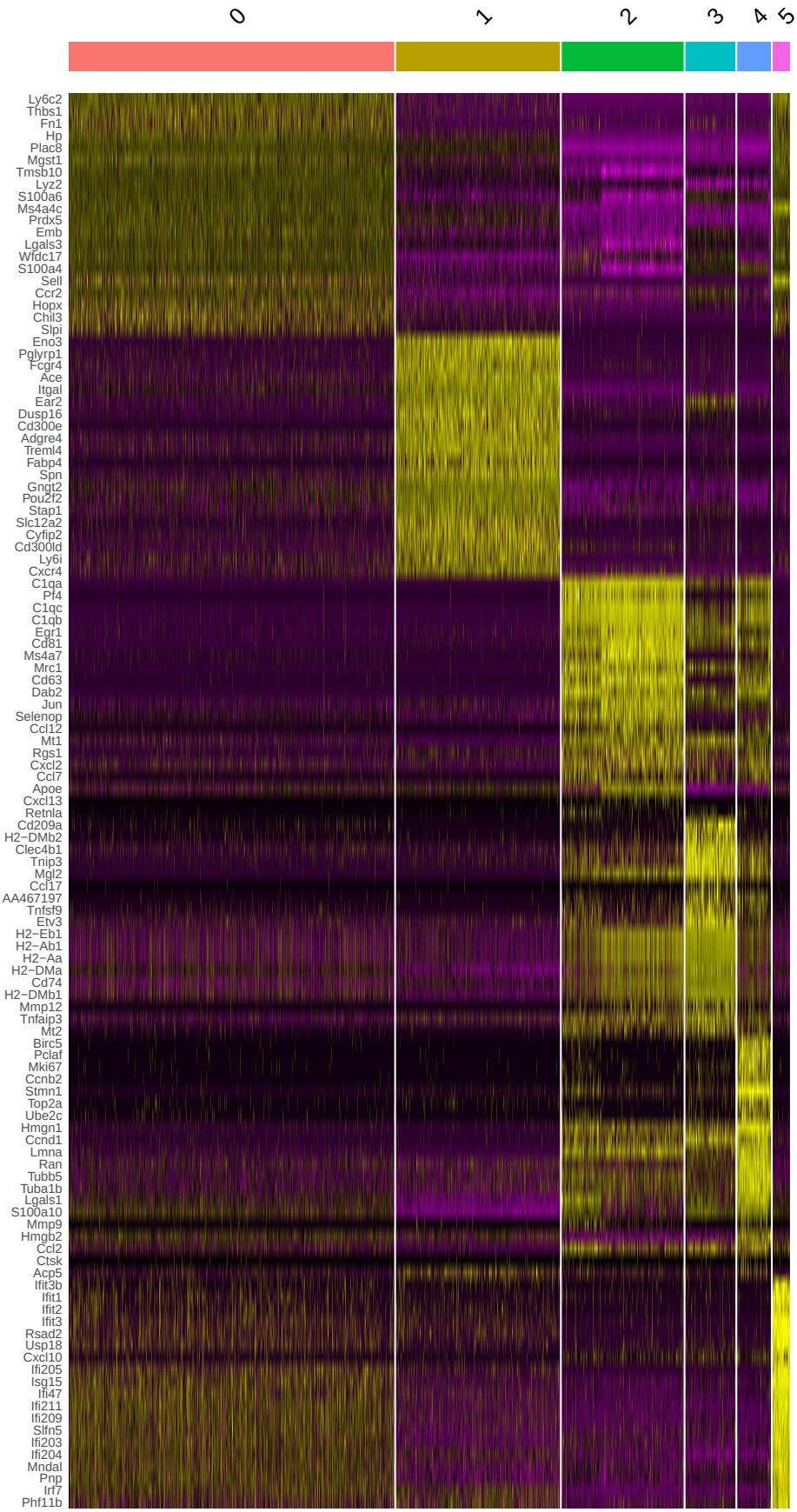
```
source("../R/SeuratFreqTable.R")
freq.celltype.list <- list(
  `Cre+` = Seurat2CellFreqTable(subset(results, subset = treatment == "Cre
     +"), slotName = "RNA_snn_res.0.3"),
```

```
  `++` = Seurat2CellFreqTable(subset(results, subset = treatment == "++"),
      slotName = "RNA_snn_res.0.3")
)

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



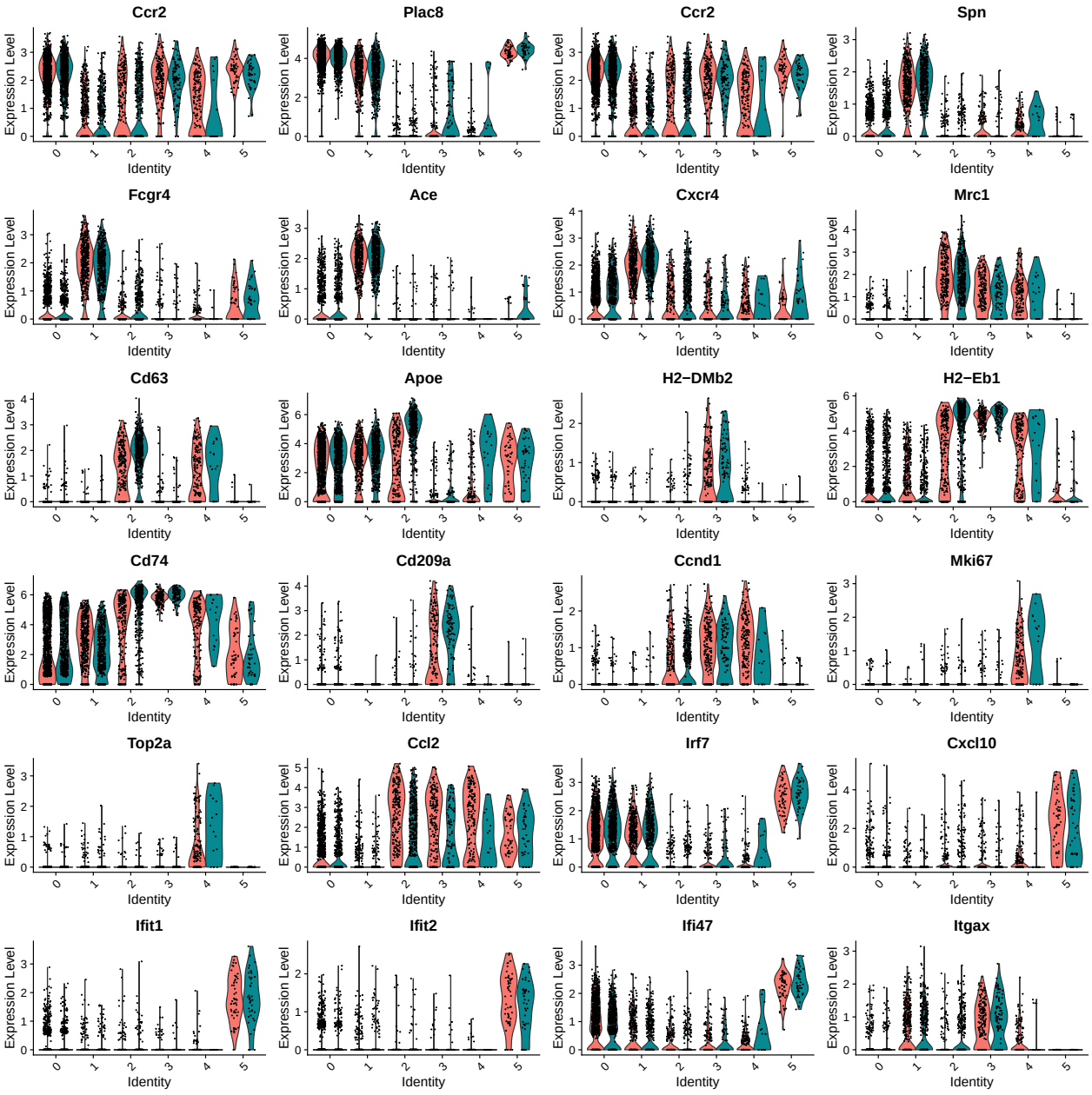
6.3 Population characterization

```
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
Thbs1
Fn1
Hp
Plac8
Mgst1
Tmsb10
Lyz2
S100a6
Ms4a4c
Prdx5
Emb
Lgals3
Wfdc17
S100a4
Sell
Ccr2
Hopx
Chil3
Slpi
Eno3
Pglyrp1
Fcgr4
Ace
Itgal
Ear2
Dusp16
Cd300e
Adgre4
Treml4
Fabp4
Spn
Gngt2
Pou2f2
Stap1
Slc12a2
Cyfip2
Cd300ld
Ly6i
Cxcr4
C1qa
Pf4
C1qc
C1qb
Egr1
Cd81
Ms4a7
Mrc1
Cd63
Dab2
Jun
Selenop
Ccl12
Mt1
Rgs1
Cxcl2
Ccl7
Apoe
Cxcl13
Retnla
Cd209a
H2−DMb2
Clec4b1
Tnip3
Mgl2
Ccl17
AA467197
Tnfsf9
Etv3
H2−Eb1
H2−Ab1
H2−Aa
H2−DMa
Cd74
H2−DMb1
Mmp12
Tnfaip3
Mt2
Birc5
Pclaf
Mki67
Ccnb2
Stmn1
Top2a
Ube2c
Hmgn1
Ccnd1
Lmna
Ran
Tubb5
Tuba1b
Lgals1
S100a10
Mmp9
Hmgb2
Ccl2
Ctsk
Acp5
Ifit3b
Ifit1
Ifit2
Ifit3
Rsad2
Usp18
Cxcl10
Ifi205
Isg15
Ifi47
Ifi211
Ifi209
Slfn5
Ifi203
Ifi204
Mndal
Pnp
Irf7
Phf11b

Expression of markers:

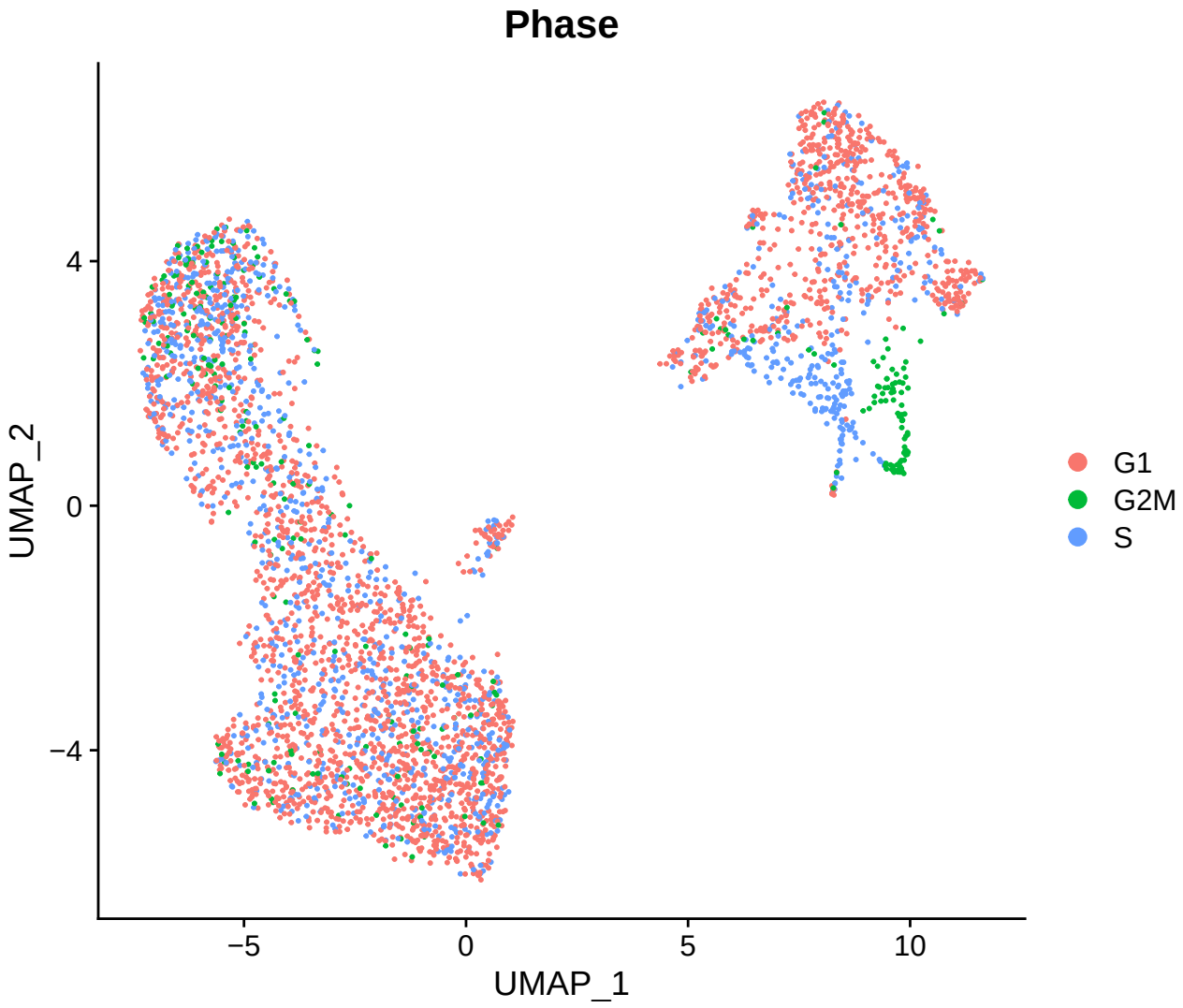
```
VlnPlot(results, features = c("Ccr2", "Plac8" ,"Ccr2", "Spn",
                              "Fcgr4", "Ace", "Cxcr4",
                              "Mrc1", "Cd63", "Apoe",
                              "H2-DMb2", "H2-Eb1", "Cd74", "Cd209a",
                              "Ccnd1","Mki67", "Top2a", "Ccl2",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



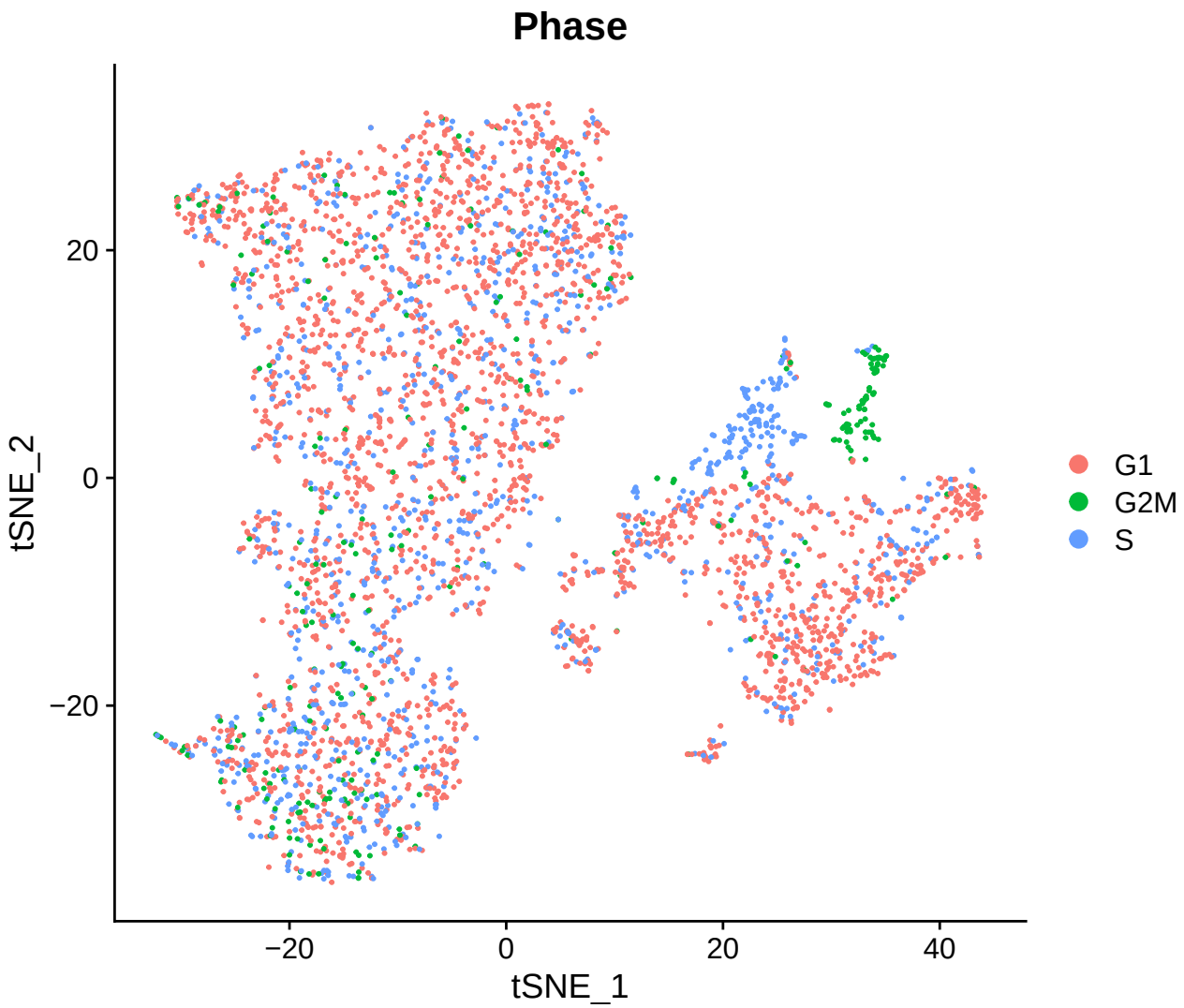
6.4 Cell-cycle analysis

```
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")
```

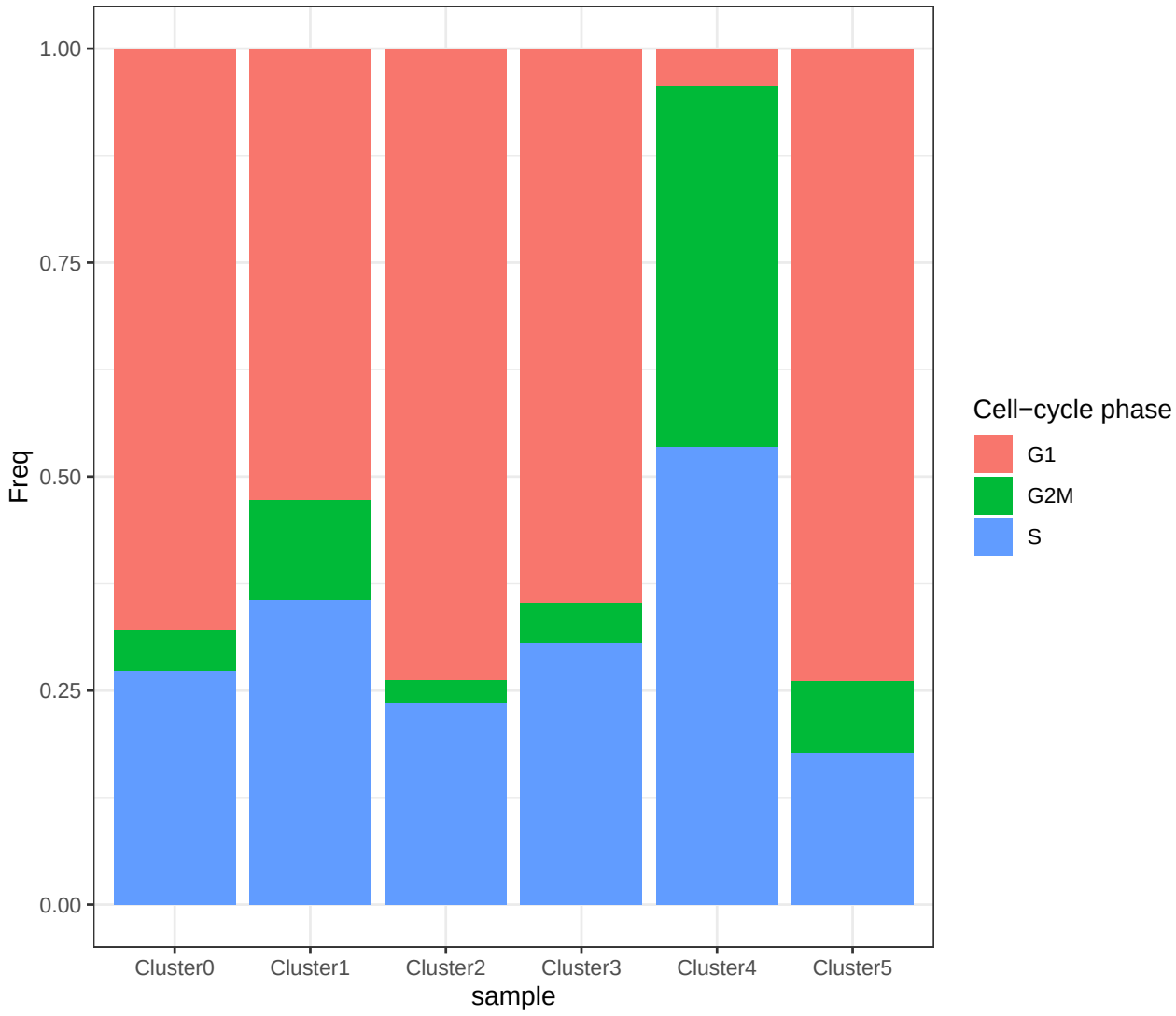



```
DimPlot(results, group.by = "Phase", reduction = "tsne")
```



Distribution of cell phases

```
freq.celltype.list <- list(
  Cluster0 = Seurat2CellFreqTable(subset(results, ident = 0), slotName = "
     Phase"),
  Cluster1 = Seurat2CellFreqTable(subset(results, ident = 1), slotName = "
     Phase"),
  Cluster2 = Seurat2CellFreqTable(subset(results, ident = 2), slotName = "
     Phase"),
  Cluster3 = Seurat2CellFreqTable(subset(results, ident = 3), slotName = "
     Phase"),
  Cluster4 = Seurat2CellFreqTable(subset(results, ident = 4), slotName = "
     Phase"),
  Cluster5 = Seurat2CellFreqTable(subset(results, ident = 5), slotName = "
     Phase")
)
barChart(freq.celltype.list) + labs(fill = "Cell-cycle␣phase")
```



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               dplyr_1.0.7
##  [3] RColorBrewer_1.1-2          celldex_1.0.0
##  [5] SingleR_1.4.1               SummarizedExperiment_1.20.0
##  [7] Biobase_2.50.0              GenomicRanges_1.42.0
##  [9] GenomeInfoDb_1.26.7         IRanges_2.24.1
## [11] S4Vectors_0.28.1            BiocGenerics_0.36.1
## [13] MatrixGenerics_1.2.1        matrixStats_0.61.0
## [15] ggplot2_3.3.5               SeuratObject_4.0.4
## [17] 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                  tensor_1.5
##  [23] listenv_0.8.0                labeling_0.4.2
##  [25] GenomeInfoDbData_1.2.4       polyclip_1.10-0
##  [27] bit64_4.0.5                  farver_2.1.0
##  [29] parallelly_1.29.0            vctrs_0.3.8
##  [31] generics_0.1.1               ipred_0.9-12
##  [33] xfun_0.28                    BiocFileCache_1.14.0
##  [35] randomForest_4.6-14          R6_2.5.1
##  [37] rsvd_1.0.5                   bitops_1.0-7
##  [39] spatstat.utils_2.2-0         cachem_1.0.6
##  [41] DelayedArray_0.16.3          assertthat_0.2.1
##  [43] promises_1.2.0.1             scales_1.1.1
##  [45] nnet_7.3-14                  gtable_0.3.0
##  [47] beachmat_2.6.4               globals_0.14.0
##  [49] goftest_1.2-3                timeDate_3043.102
##  [51] rlang_0.4.12                 splines_4.0.3
##  [53] lazyeval_0.2.2               ModelMetrics_1.2.2.2
##  [55] checkmate_2.0.0              spatstat.geom_2.3-0
##  [57] BiocManager_1.30.16          yaml_2.2.1
##  [59] reshape2_1.4.4               abind_1.4-5
##  [61] backports_1.4.0              httpuv_1.6.3
##  [63] Hmisc_4.6-0                  DiagrammeR_1.0.6.1
##  [65] caret_6.0-90                 tools_4.0.3
##  [67] lava_1.6.10                  ellipsis_0.3.2
##  [69] spatstat.core_2.3-2          proxy_0.4-26
```

```
##  [71] ggridges_0.5.3               Rcpp_1.0.7                          68
##  [73] plyr_1.8.6                   base64enc_0.1-3                     69
##  [75] visNetwork_2.1.0             sparseMatrixStats_1.2.1             70
##  [77] zlibbioc_1.36.0              purrr_0.3.4                         71
##  [79] RCurl_1.98-1.5               rpart_4.1-15                        72
##  [81] deldir_1.0-6                 pbapply_1.5-0                       73
##  [83] zoo_1.8-9                    nichenetr_1.0.0                     74
##  [85] ggrepel_0.9.1                cluster_2.1.0                       75
##  [87] magrittr_2.0.1               data.table_1.14.2                   76
##  [89] RSpectra_0.16-0              scattermore_0.7                     77
##  [91] lmtest_0.9-39                RANN_2.6.1                          78
##  [93] fitdistrplus_1.1-6           hms_1.1.1                           79
##  [95] patchwork_1.1.1              mime_0.12                           80
##  [97] evaluate_0.14                xtable_1.8-4                        81
##  [99] jpeg_0.1-9                   gridExtra_2.3                       82
## [101] compiler_4.0.3               tibble_3.1.6                        83
## [103] KernSmooth_2.23-20           crayon_1.4.2                        84
## [105] htmltools_0.5.2              tzdb_0.2.0                          85
## [107] mgcv_1.8-33                  later_1.3.0                         86
## [109] Formula_1.2-4                tidyr_1.1.4                         87
## [111] lubridate_1.8.0              DBI_1.1.1                           88
## [113] ExperimentHub_1.16.1         dbplyr_2.1.1                        89
## [115] MASS_7.3-53                  rappdirs_0.3.3                      90
## [117] readr_2.1.1                  Matrix_1.3-4                        91
## [119] cli_3.1.0                    gower_0.2.2                         92
## [121] igraph_1.2.9                 pkgconfig_2.0.3                     93
## [123] foreign_0.8-81               plotly_4.10.0                       94
## [125] spatstat.sparse_2.0-0        recipes_0.1.17                      95
## [127] foreach_1.5.1                XVector_0.30.0                      96
## [129] prodlim_2019.11.13           stringr_1.4.0                       97
## [131] digest_0.6.29                sctransform_0.3.2                   98
## [133] RcppAnnoy_0.0.19             spatstat.data_2.1-0                 99
## [135] rmarkdown_2.11               leiden_0.3.9                        100
## [137] htmlTable_2.3.0              uwot_0.1.11                         101
## [139] DelayedMatrixStats_1.12.3    curl_4.3.2                          102
## [141] shiny_1.7.1                  lifecycle_1.0.1                     103
## [143] nlme_3.1-153                 jsonlite_1.7.2                      104
## [145] BiocNeighbors_1.8.2          viridisLite_0.4.0                   105
## [147] limma_3.46.0                 fansi_0.5.0                         106
## [149] pillar_1.6.4                 lattice_0.20-41                     107
## [151] fastmap_1.1.0                httr_1.4.2                          108
## [153] survival_3.2-7               interactiveDisplayBase_1.28.0       109
## [155] glue_1.5.1                   fdrtool_1.2.17                      110
## [157] png_0.1-7                    iterators_1.0.13                    111
## [159] BiocVersion_3.12.0           bit_4.0.4                           112
## [161] class_7.3-17                 stringi_1.7.6                       113
## [163] blob_1.2.2                   BiocSingular_1.6.0                  114
## [165] AnnotationHub_2.22.1         caTools_1.18.2                      115
## [167] latticeExtra_0.6-29          memoise_2.0.1                       116
## [169] e1071_1.7-9                  irlba_2.3.5                         117
## [171] future.apply_1.8.1                                               118
```

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.