Mafb-restricted local monocyte proliferation precedes lung interstitial macrophage differentiation

14 - Compare with cMaf-KO IM

2022-03-16 17:56:50 +0100

Abstract

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

Contents

1 Description

In this analysis, we compared cMAF-KO sample to Control sample. All four samples were included, but dKO sample was found to be highly similar to the Mafb-KO sample. So we focus on the difference between cMaf-KO and Control samples.

2 Load packages and data

```
suppressMessages({
  library(Seurat)
  library(ggplot2)
  library(RColorBrewer)
})

so <- readRDS(file = "../12-cMAF_and_Mafb_deficient_IM/All_samples_Maf.
   seuratObject.Rds")
```

We will take into account the cMAF-KO and dKO samples into analysis.

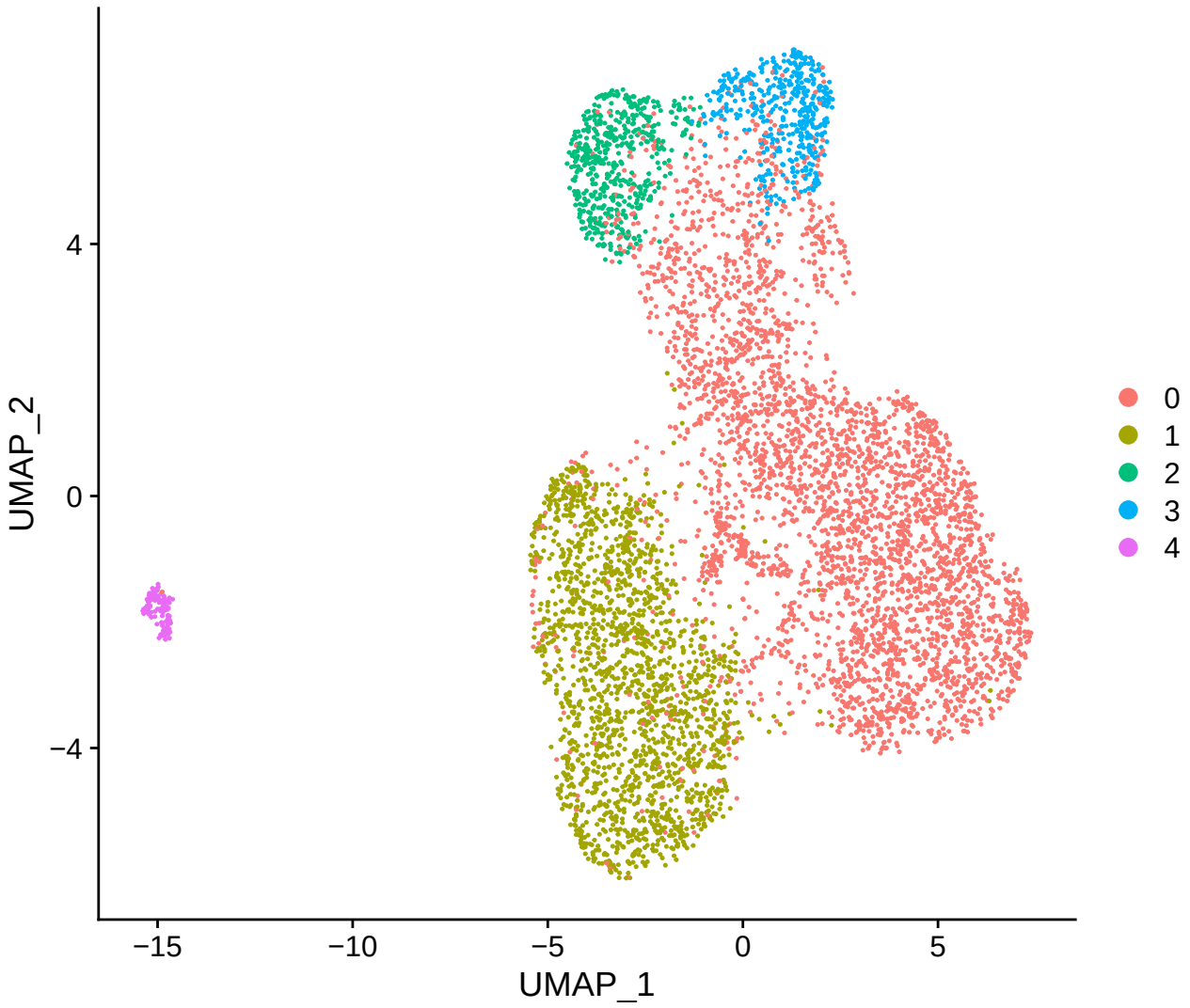
3 Compare populations

3.1 Re-process data

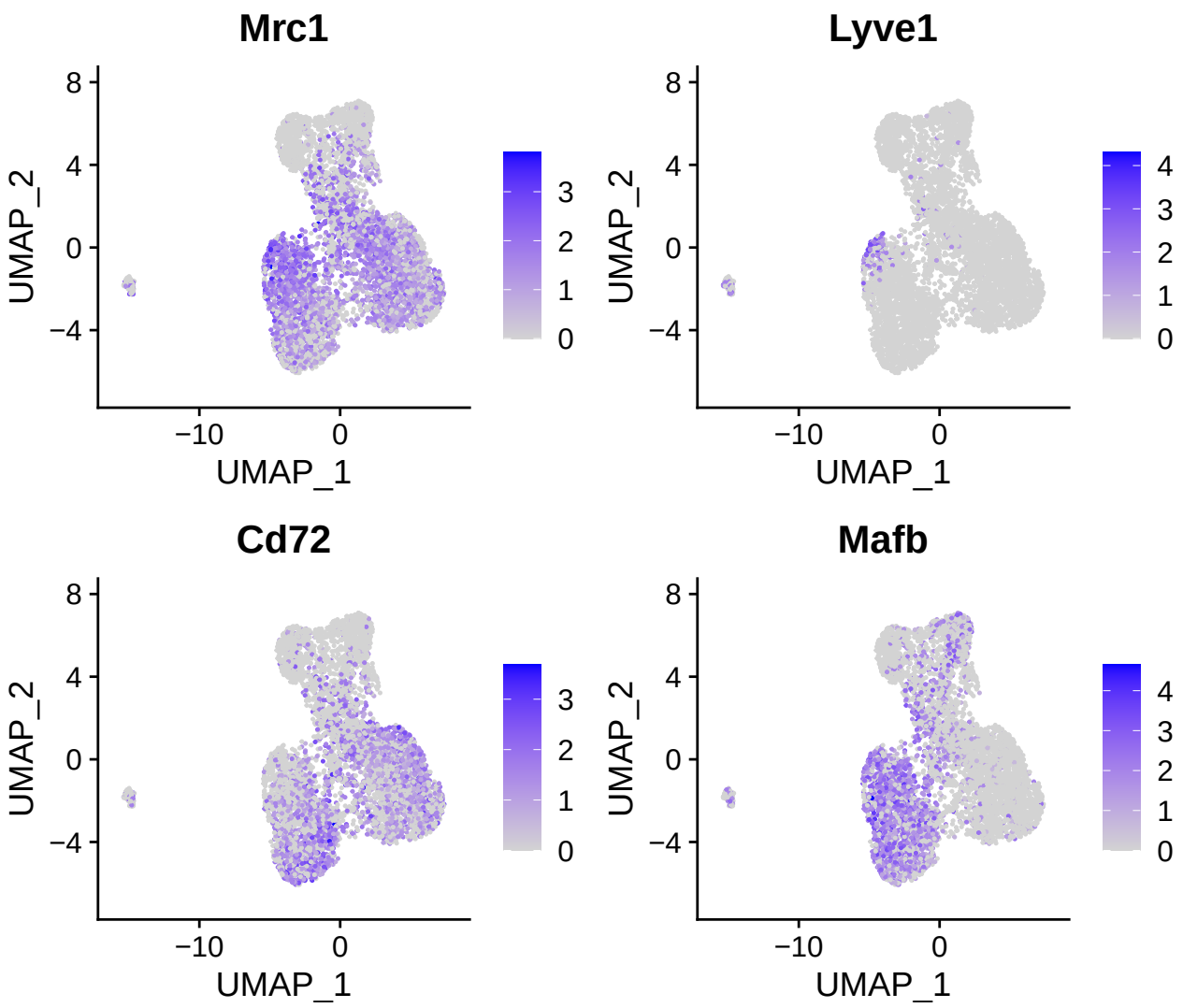
```
so <- NormalizeData(so, verbose=FALSE)
so <- FindVariableFeatures(so, selection.method = "vst", nfeatures = 2000,
    verbose=FALSE) # we focus less variable genes.
so <- ScaleData(so, features = rownames(so), verbose=FALSE)
so <- RunPCA(so, features = VariableFeatures(so), verbose=FALSE)
so <- RunTSNE(so, dims = 1:8, verbose=FALSE)
so <- RunUMAP(so, dims = 1:8, verbose=FALSE)
```

```
so <- FindNeighbors(so, dims = 1:8, verbose = FALSE)
so <- FindClusters(so, resolution = 0.12, verbose = FALSE)
```

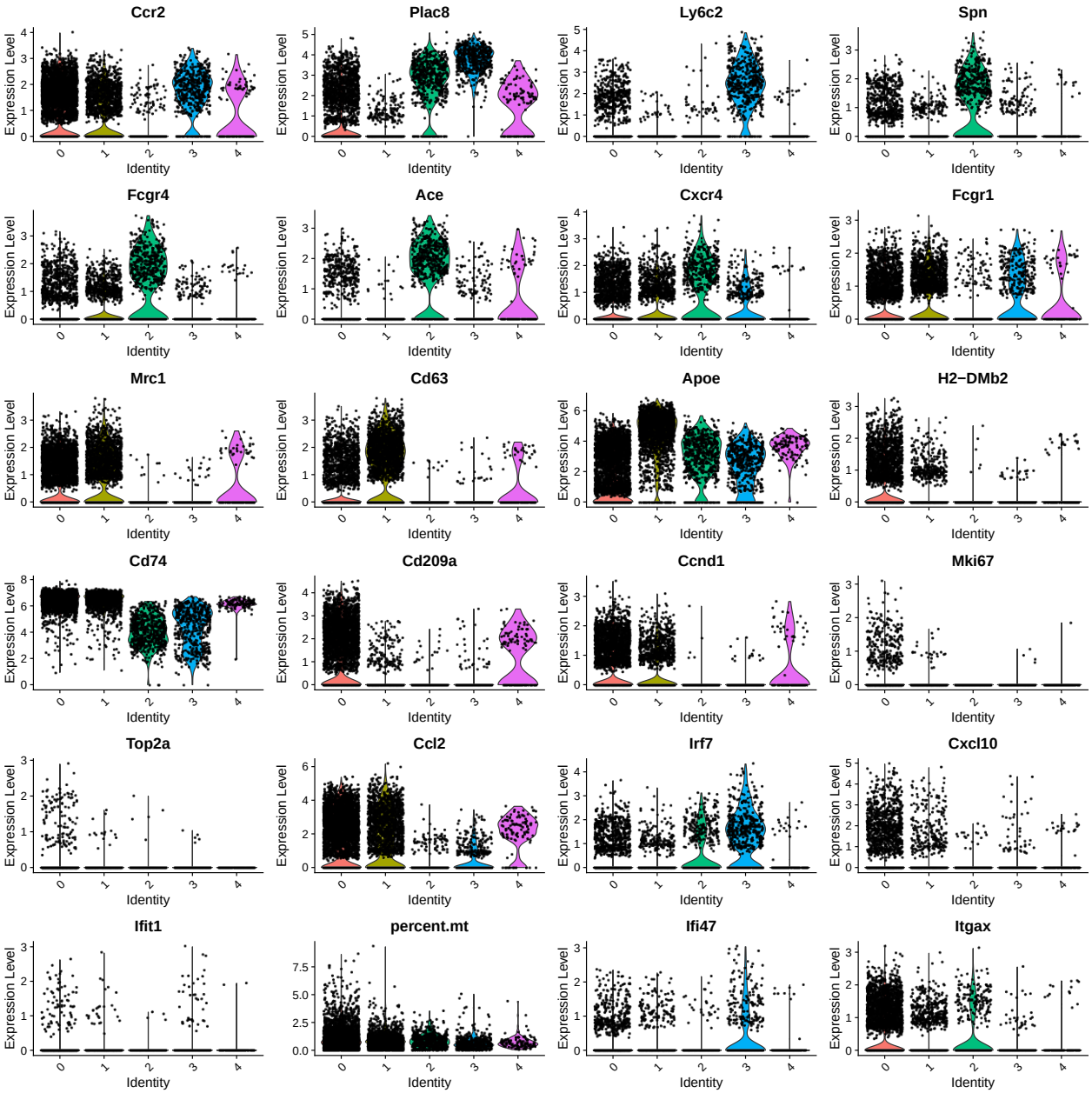
```
DimPlot(so, reduction = "umap")
```




```
FeaturePlot(so, features = c("Mrc1", "Lyve1", "Cd72", "Mafb"))
```




```
VlnPlot(
  so,
  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)
```

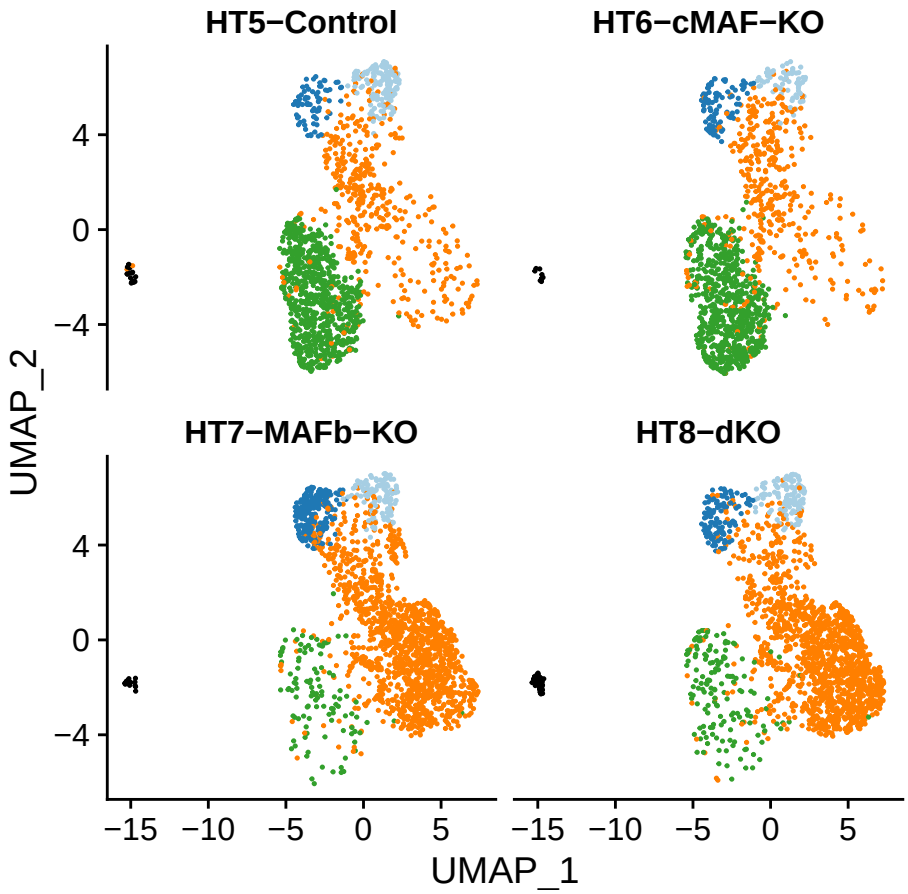


cluster 0: Mafb-KO Cluster 1: IM Cluster 2: Patrolling Monocytes Cluster 3: Classical Monocytes Cluster 4: Unknown

```
so$cell.type2 <- factor(Idents(so), labels = c("Mafb-independent", "IM", "
    Patrolling␣Mono", "Classical␣Mono", "Unknown"))
so$cell.type2 <- factor(so$cell.type2, levels = c("Classical␣Mono", "
    Patrolling␣Mono", "IM", "Mafb-independent", "Unknown"))
Idents(so) <- "cell.type2"
```

```
pal4 <- c(
  "#A6CEE3", # cMo
  "#1F78B4", # pMo
  "#33A02C", # CD206 IM
   "#FF7F00", # Mafb- neo
  "black" # unknown
```

```
)
DimPlot(so, cols = pal4, split.by = "group", ncol = 2
) + NoLegend()
```




```
ggsave(filename = "../Figures/UMAPplot_All_samplesMaf_separate_2colunns.
    pdf", width = 5, height = 5)
```

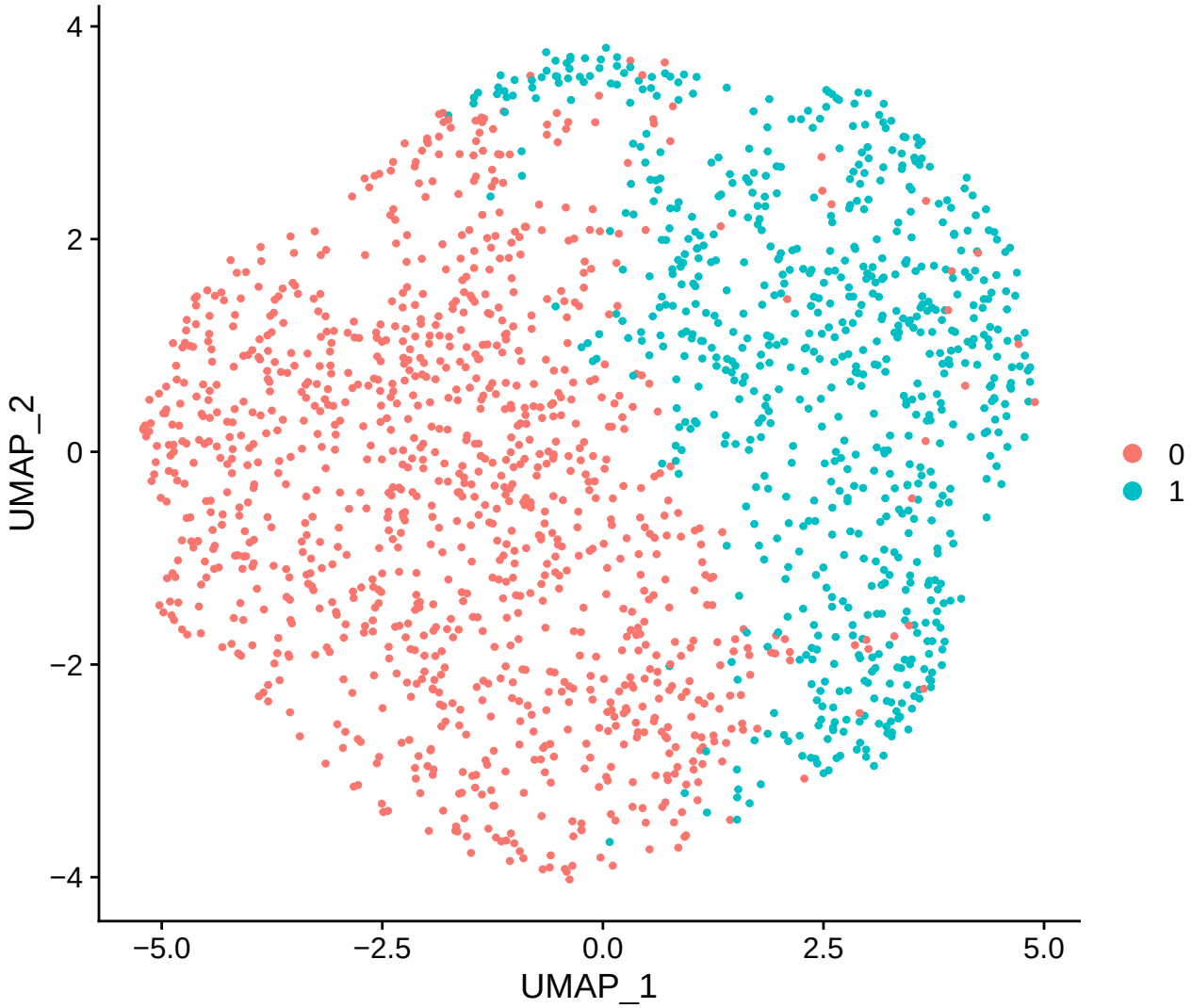
4 Identify the CD206+ and CD206- IMs

```
ims <- subset(so, idents = "IM")

ims <- NormalizeData(ims, verbose=FALSE)
ims <- FindVariableFeatures(ims, selection.method = "vst", nfeatures =
    2000, verbose=FALSE) # we focus less variable genes.
ims <- ScaleData(ims, features = rownames(ims), verbose=FALSE)
ims <- RunPCA(ims, features = VariableFeatures(ims), verbose=FALSE)
ims <- RunTSNE(ims, dims = 1:8, verbose=FALSE)
ims <- RunUMAP(ims, dims = 1:8, verbose=FALSE)
ims <- FindNeighbors(ims, dims = 1:8, verbose = FALSE)
ims <- FindClusters(ims, resolution = 0.15, verbose = T)
```

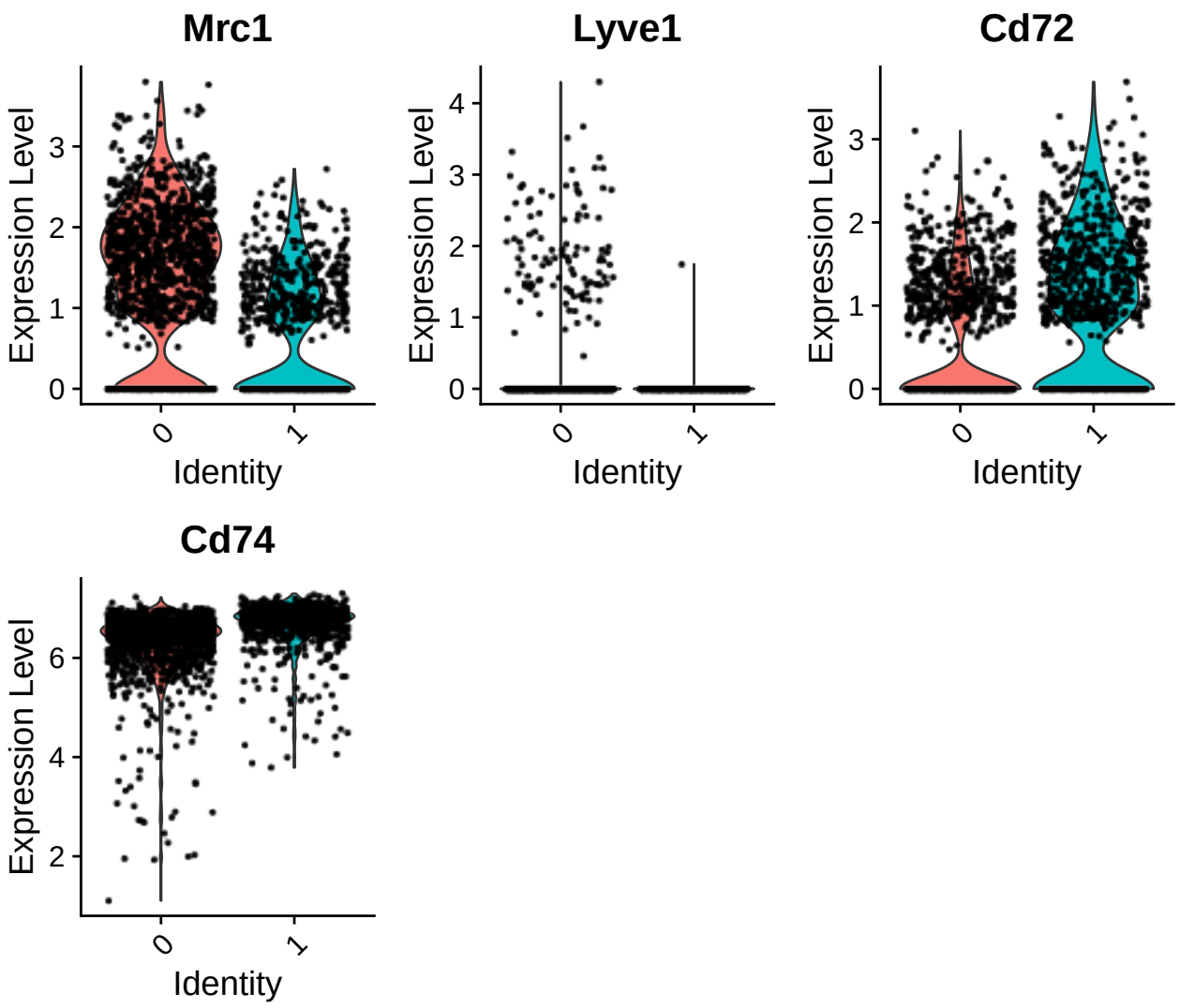
```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
```

```
## Number of nodes: 1802
## Number of edges: 57482
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8545
## Number of communities: 2
## Elapsed time: 0 seconds
```

```
DimPlot(ims)
```



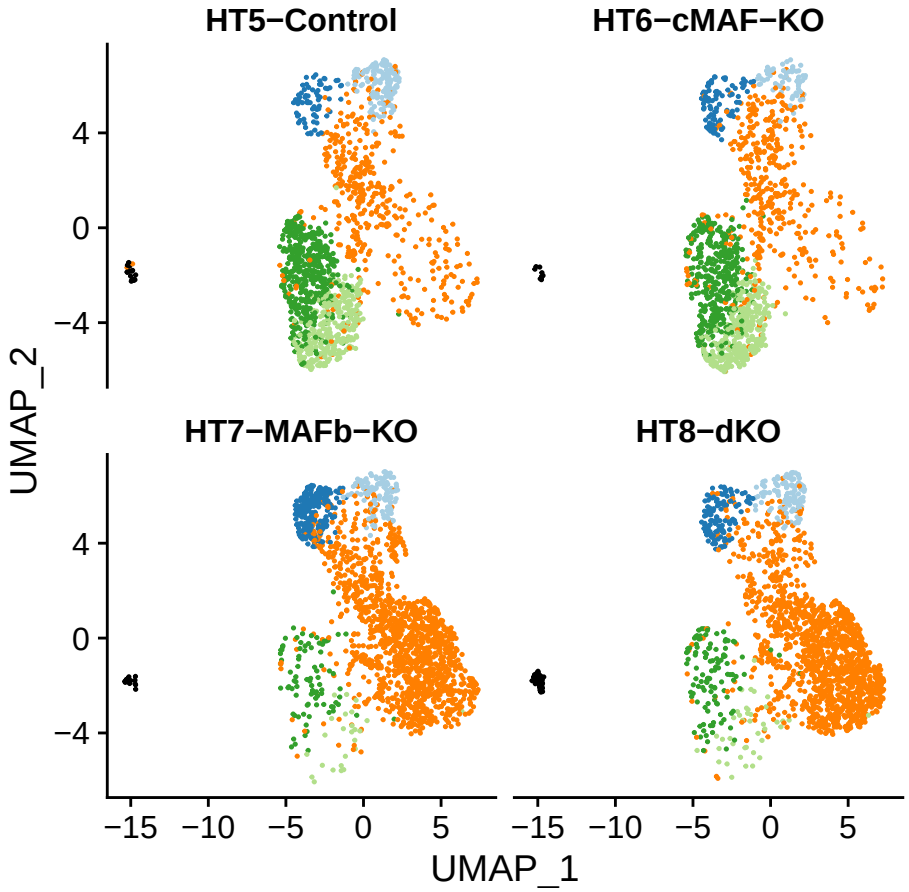

```
VlnPlot(ims, features = c("Mrc1", "Lyve1", "Cd72", "Cd74"))
```



cluster 0: CD206+ IM cluster 1: CD206- IM

```
so$cell.type3 <- as.character(so$cell.type2)
so$cell.type3[WhichCells(ims, idents = "0")] <- "CD206+ IM"
so$cell.type3[WhichCells(ims, idents = "1")] <- "CD206- IM"
so$cell.type3 <- factor(so$cell.type3, levels = c("Classical Mono", "
    Patrolling Mono", "CD206+ IM", "CD206- IM", "Mafb-independent", "
    Unknown"))
Idents(so) <- "cell.type3"
```

```
pal4 <- c(
  "#A6CEE3", # cMo
  "#1F78B4", # pMo
  "#33A02C", # CD206 IM
  "#B2DF8A", # MHCII IM
  "#FF7F00", # Mafb- neo
  "black" # unknown
)
DimPlot(so, cols = pal4, split.by = "group", ncol = 2
) + NoLegend()
```




```
ggsave(filename = "../Figures/UMAPplot_All_samplesMaf_with_IMsubsets_
    separate_2colunns.pdf", width = 5, height = 5)
```

5 Focus on the cMAF-KO vs Control

```
so <- subset(so, subset = group == c("HT5-Control", "HT6-cMAF-KO"))
```

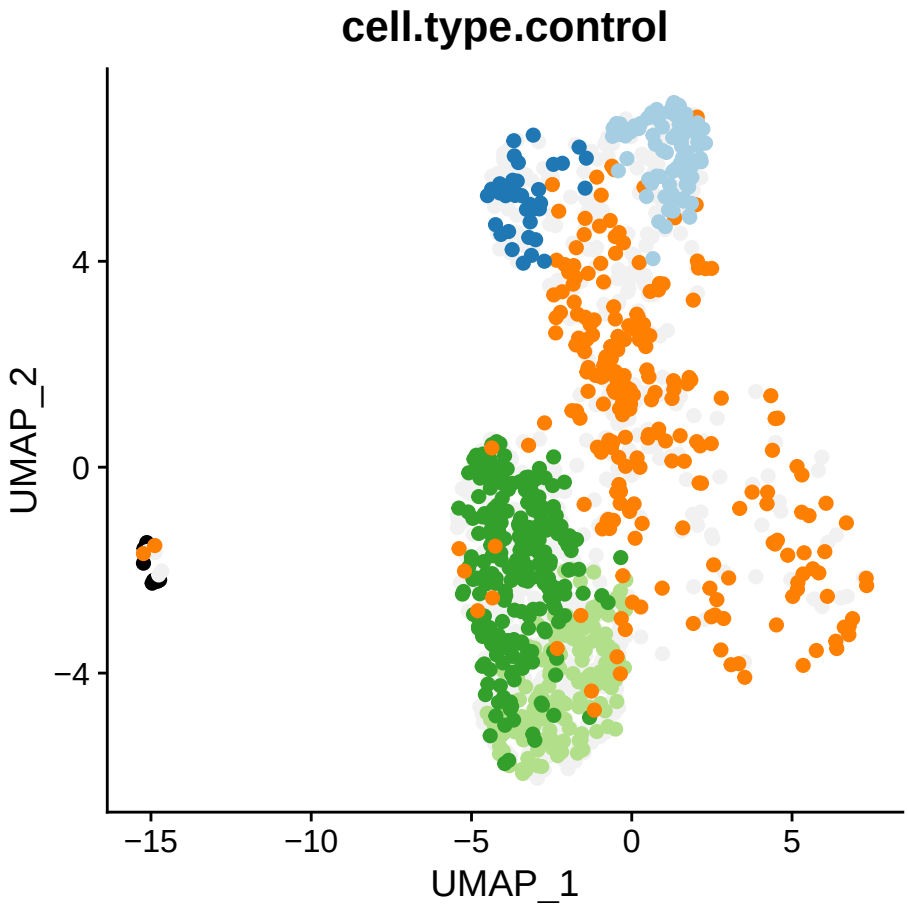
Plot cell in colors but only for one of two samples

```
so$cell.type.control <- as.character(so$cell.type3)
so$cell.type.control[WhichCells(so, expression = group == "HT6-cMAF-KO")]
    <- "ZZ"
so$cell.type.control <- factor(so$cell.type.control, levels = c("Classical
    Mono", "Patrolling Mono", "CD206+ IM", "CD206- IM", "Mafb-independent"
    , "Unknown", "ZZ"))
Idents(so) <- "cell.type.control"
```

```
pal4.control <- c(
  "black", # unknown
  "#F1F1F1", # grey for another sample
  "#B2DF8A", # MHCII IM
  "#33A02C", # CD206 IM
  "#FF7F00", # Mafb- neo
    "#A6CEE3", # cMo
```

```
  "#1F78B4" # pMo
)
```

```
DimPlot(so, cols = pal4.control, group.by = "cell.type.control", pt.size =
    2,
order = c("IM", "Patrolling Mono", "Classical Mono", "Mafb-independent", "
    CD206+ IM", "CD206- IM", "ZZ")
) + NoLegend()
```



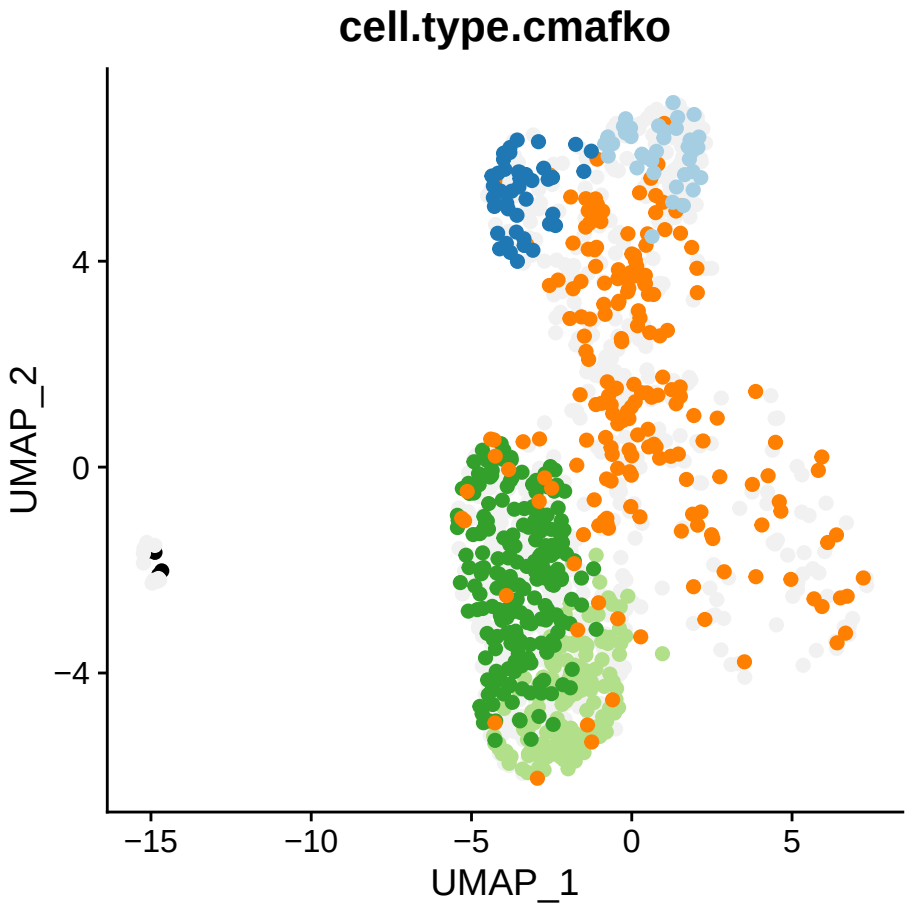

```
ggsave(filename = "../Figures/UMAPplot_Ctl_(vscMafKO)_no_legend.pdf",
       width = 5, height = 5)
```

Plot Mafb-KO:

```
so$cell.type.cmafko <- as.character(so$cell.type3)
so$cell.type.cmafko[WhichCells(so, expression = group == "HT5-Control")]
    <- "ZZ"
so$cell.type.cmafko <- factor(so$cell.type.cmafko, levels = c("Classical
    Mono", "Patrolling Mono", "CD206+ IM", "CD206- IM", "Mafb-independent",
     "Unknown", "ZZ"))
Idents(so) <- "cell.type.cmafko"
```

```
DimPlot(so, cols = pal4.control, group.by = "cell.type.cmafko", pt.size =
    2,
```

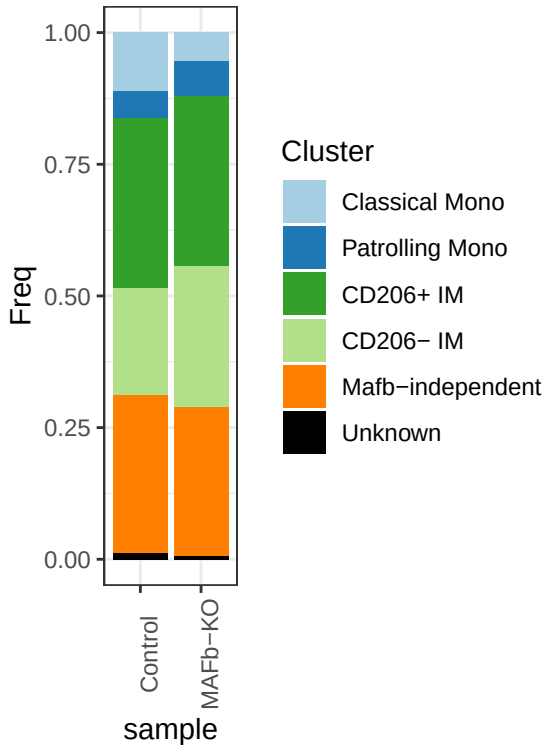
```
order = c("IM", "Patrolling␣Mono", "Classical␣Mono", "Mafb-independent", "
    CD206+␣IM", "CD206-␣IM", "ZZ")
) + NoLegend()
```




```
ggsave(filename = "../Figures/UMAPplot_cMafKO_(vsControl)_no_legend.pdf",
       width = 5, height = 5)
```

See population frequencies:

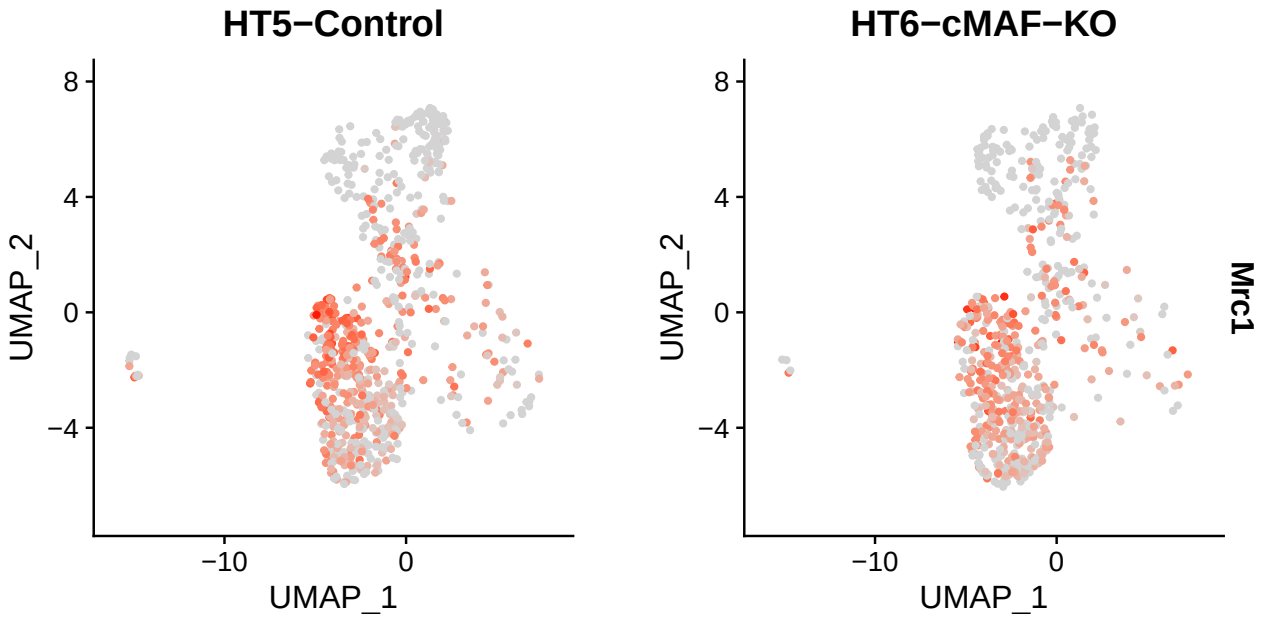
```
source("../R/SeuratFreqTable.R")
freq.celltype.list <- list(
  `Control` = Seurat2CellFreqTable(subset(so, subset = group == "HT5-
     Control"), slotName = "cell.type3"),
  `MAFb-KO` = Seurat2CellFreqTable(subset(so, subset = group == "HT6-cMAF-
     KO"), slotName = "cell.type3")
)

source("../R/barChart.R")
barChart(freq.celltype.list) + labs(fill = "Cluster") + scale_fill_manual(
    values = pal4) + theme(axis.text.x = element_text(angle = 90))
```




```
ggsave(filename = "../Figures/Barplot_Ctl_cMafKO_population_frequency.pdf"
    ,
       width = 3, height = 4)
```

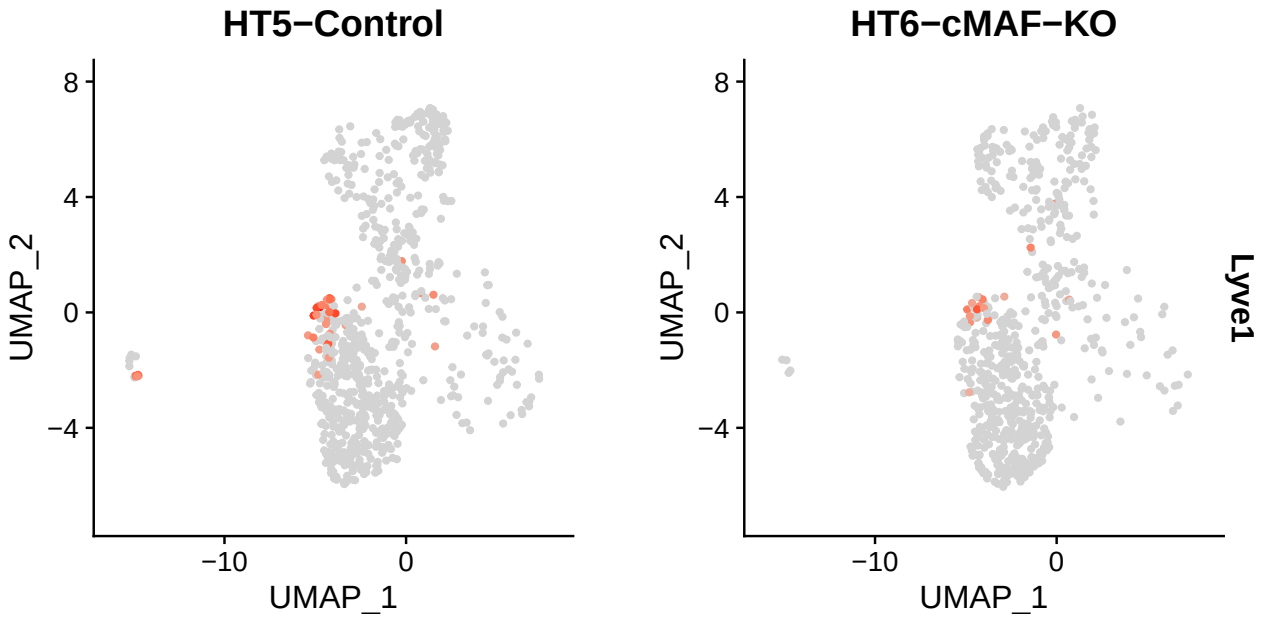
Show CD206+ IM markers

```
FeaturePlot(so, features = "Mrc1", cols = c("lightgray", "red"), min.
    cutoff = 0.5, split.by = "group")
```



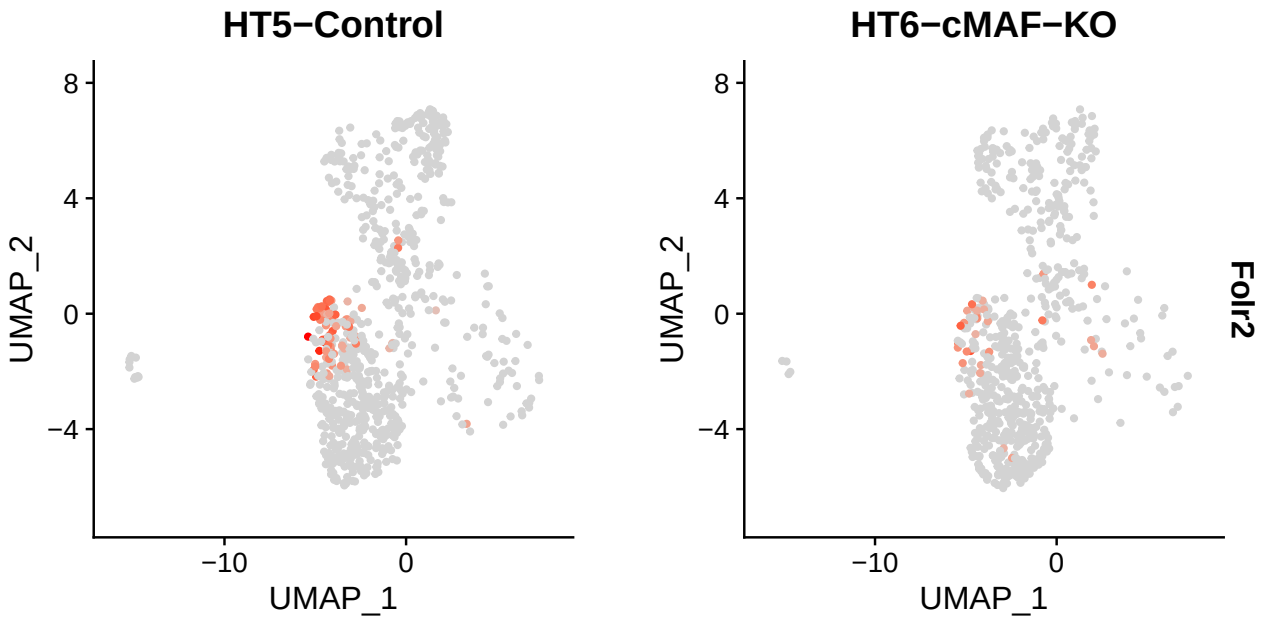
```
ggsave(filename = "../Figures/VlnPlot_Mrc1_in_cMafKO_Control.pdf", width =
    8, height = 4)
```

```
FeaturePlot(so, features = "Lyve1", cols = c("lightgray", "red"), split.by
    = "group")
```



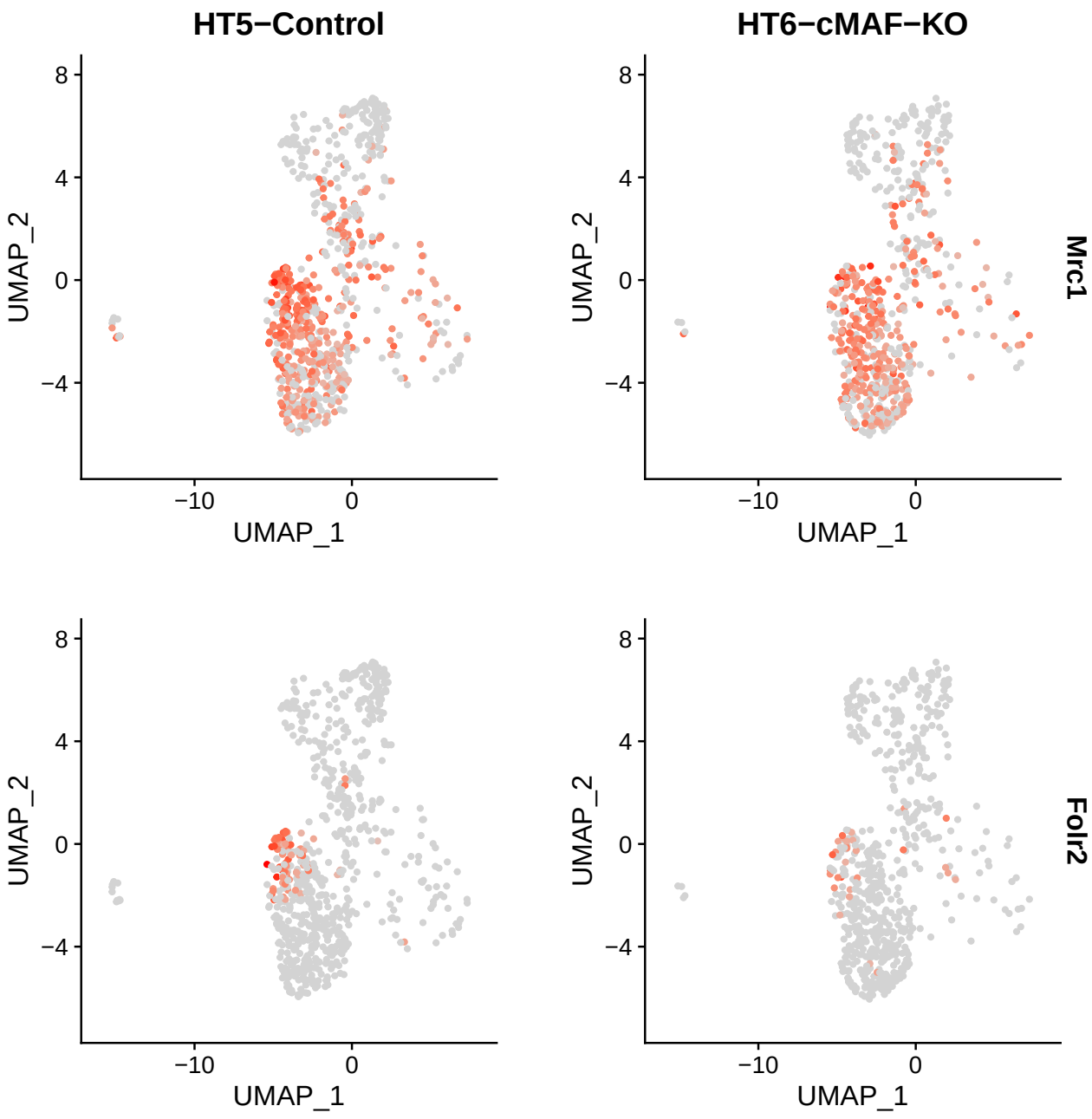

```
ggsave(filename = "../Figures/VlnPlot_Lyve1_in_cMafKO_control.pdf", width
    = 8, height = 4)
```

```
FeaturePlot(so, features = "Folr2", cols = c("lightgray", "red"), split.by
    = "group")
```



```
ggsave(filename = "../Figures/VlnPlot_Folr2_in_cMafKO_control.pdf", width
    = 8, height = 4)
```

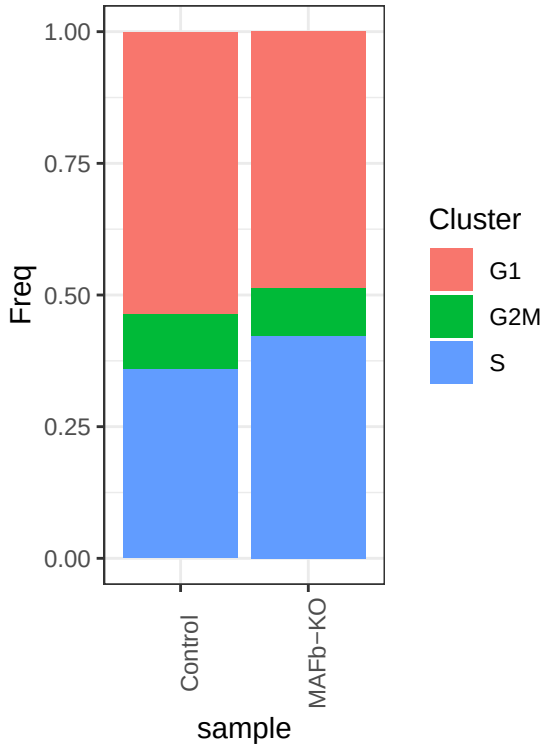
```
FeaturePlot(so, features = c("Mrc1", "Folr2"), cols = c("lightgray", "red"
    ), split.by = "group")
```



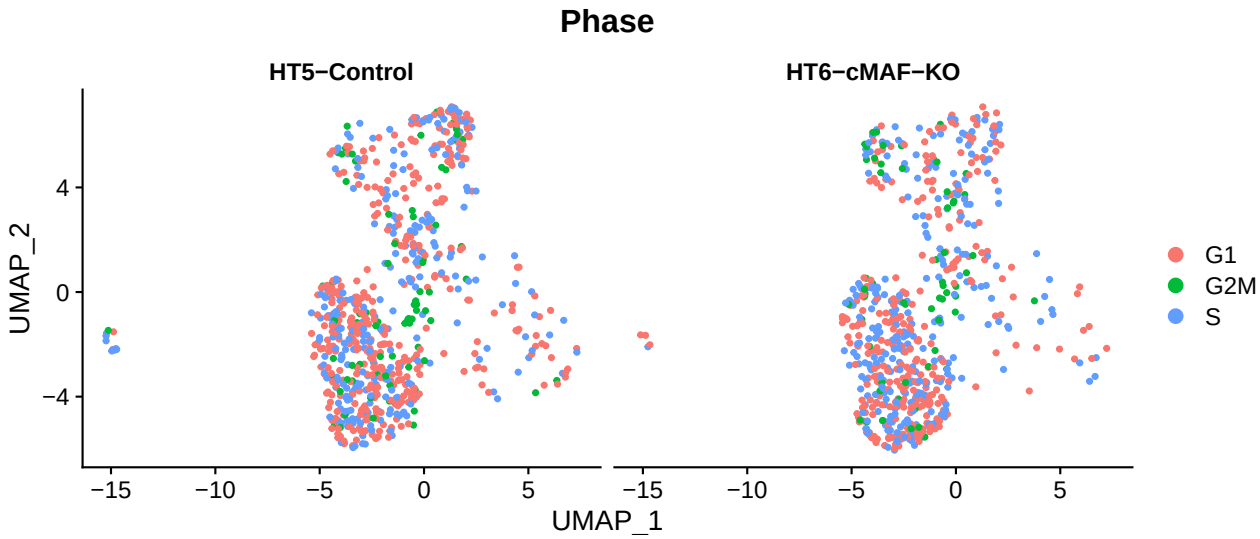

```
ggsave(filename = "../Figures/Featureplot_Mrc1_Folr2_in_cMafKO_control.pdf
    ", height = 8, width = 8)
```

6 Proliferation comparison

```
freq.celltype.list <- list(
  `Control` = Seurat2CellFreqTable(subset(so, subset = group == "HT5-Control"), slotName = "Phase"),
  `MAFb-KO` = Seurat2CellFreqTable(subset(so, subset = group == "HT6-cMAF-KO"), slotName = "Phase")
)

barChart(freq.celltype.list) + labs(fill = "Cluster") +  theme(axis.text.x = element_text(angle = 90))
```




```
DimPlot(so, group.by = "Phase", split.by = "group")
```



7 Comparison between cMAF-deficient IMs population and control IMs

Let's focus on the Mafb-deficient population in Mafb-deficient sample.

```
ims <- subset(so, subset = cell.type2 == "IM")
Idents(ims) <- "group"
```

7.1 DE genes between Mafb- neo and IM population

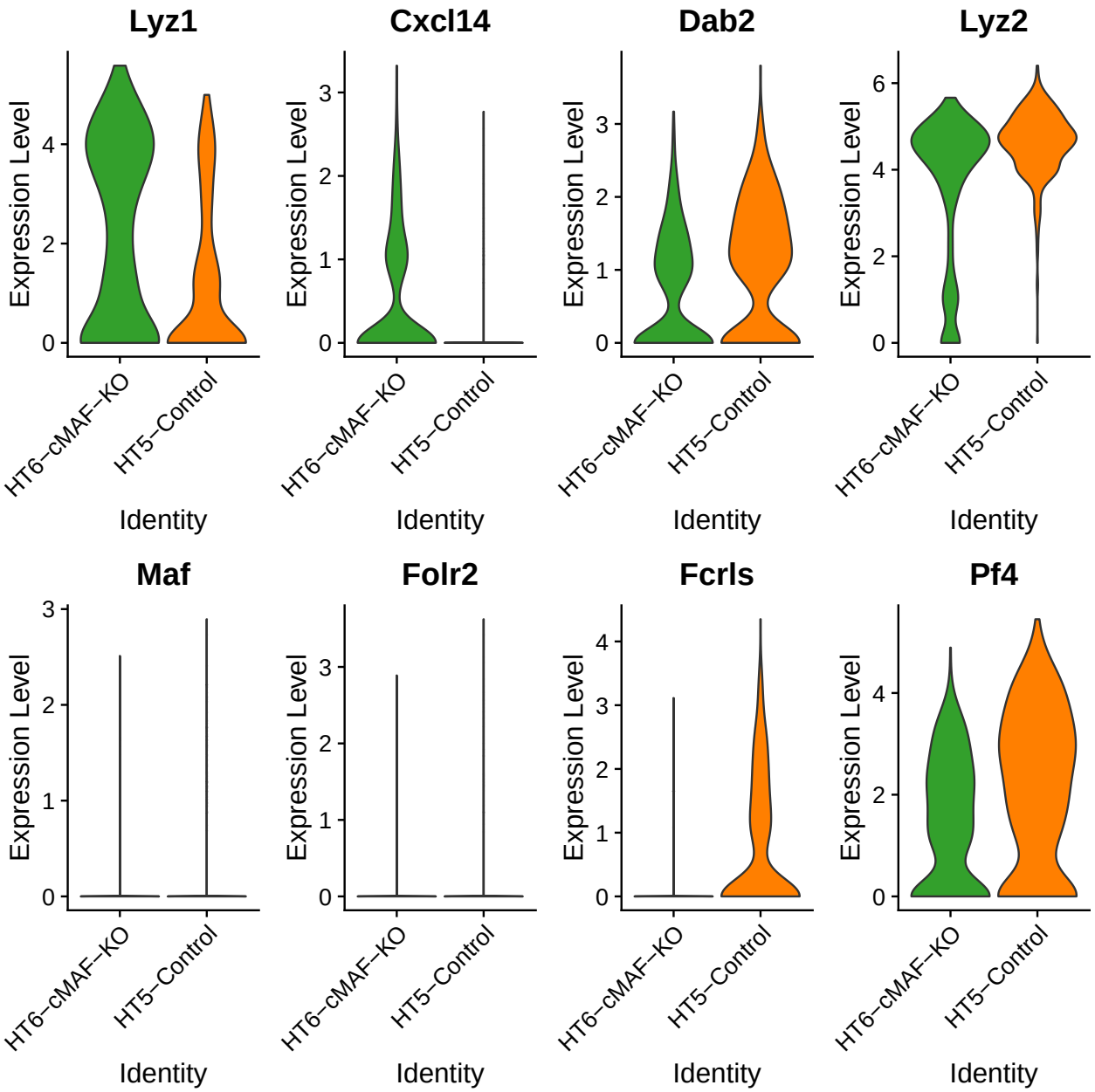
```
library(dplyr)
cMafKO_vs_IM <- FindMarkers(ims,
                            ident.1 = "HT6-cMAF-KO",
                            ident.2 = "HT5-Control",
                            logfc.threshold = 0,
                            verbose = FALSE)


# keep only adj p value < 0.05 and logFC > 0.5 as significant markers.
cMafKO_vs_IM.markers <- cMafKO_vs_IM[cMafKO_vs_IM$p_val_adj < 0.05 & abs(
   cMafKO_vs_IM$avg_log2FC) > 0.5, ]
cMafKO_vs_IM.markers <- cMafKO_vs_IM.markers[order(cMafKO_vs_IM.markers$
   avg_log2FC, decreasing = TRUE), ]
nrow(cMafKO_vs_IM.markers)
```

```
## [1] 8
```

```
write.csv(cMafKO_vs_IM.markers ,file = "./cMAFKO_vs_IM.DEgenes.results.csv
   ", quote = FALSE)
```

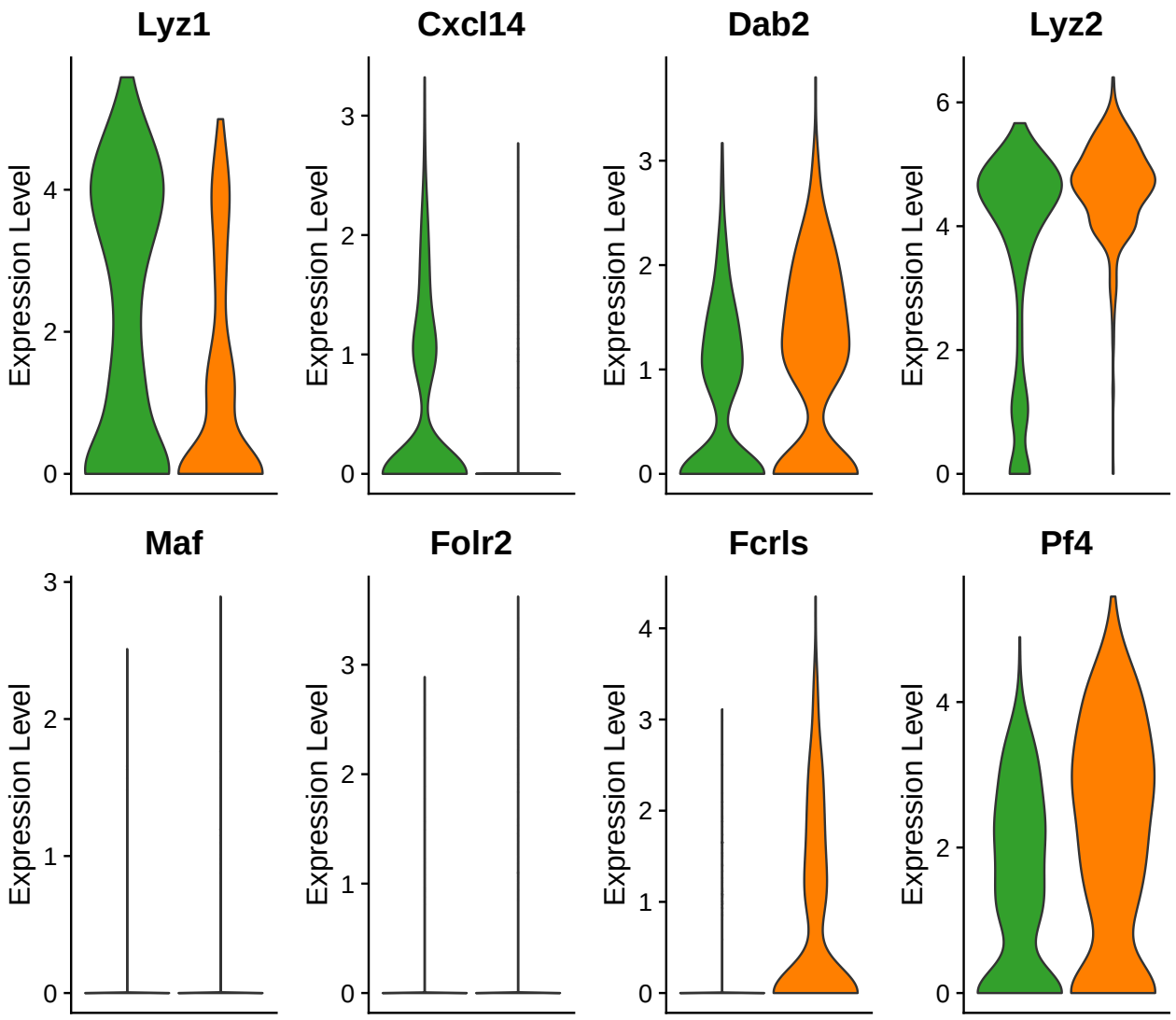
Show in vlnplot

```
p <- VlnPlot(ims, features = c(rownames(cMafKO_vs_IM.markers)), cols = c("
   #33A02C", "#FF7F00"), ncol = 4, pt.size = 0)
p
```



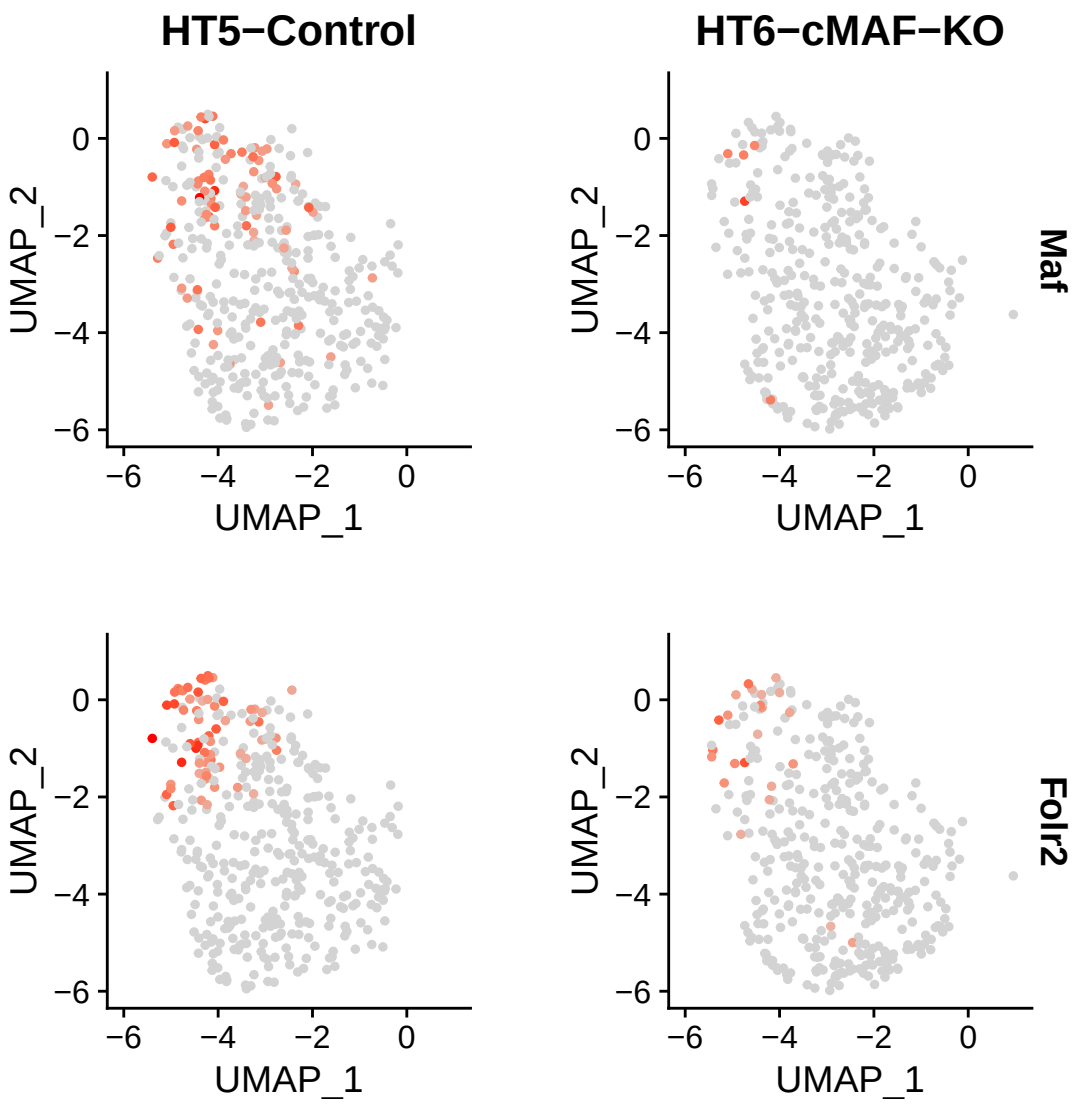
Show vlnplot without label:

```
p & theme(axis.title.x=element_blank(),
          axis.text.x=element_blank(),
          axis.ticks.x=element_blank())
```




```
ggsave(filename = "../Figures/Vlnplot_DE_genes_in_Ctrl_cMafKO.pdf",
       width = 8, height = 8)
```

```
FeaturePlot(ims, features = c("Maf", "Folr2"), split.by = "group", cols =
   c("lightgray", "red"))
```



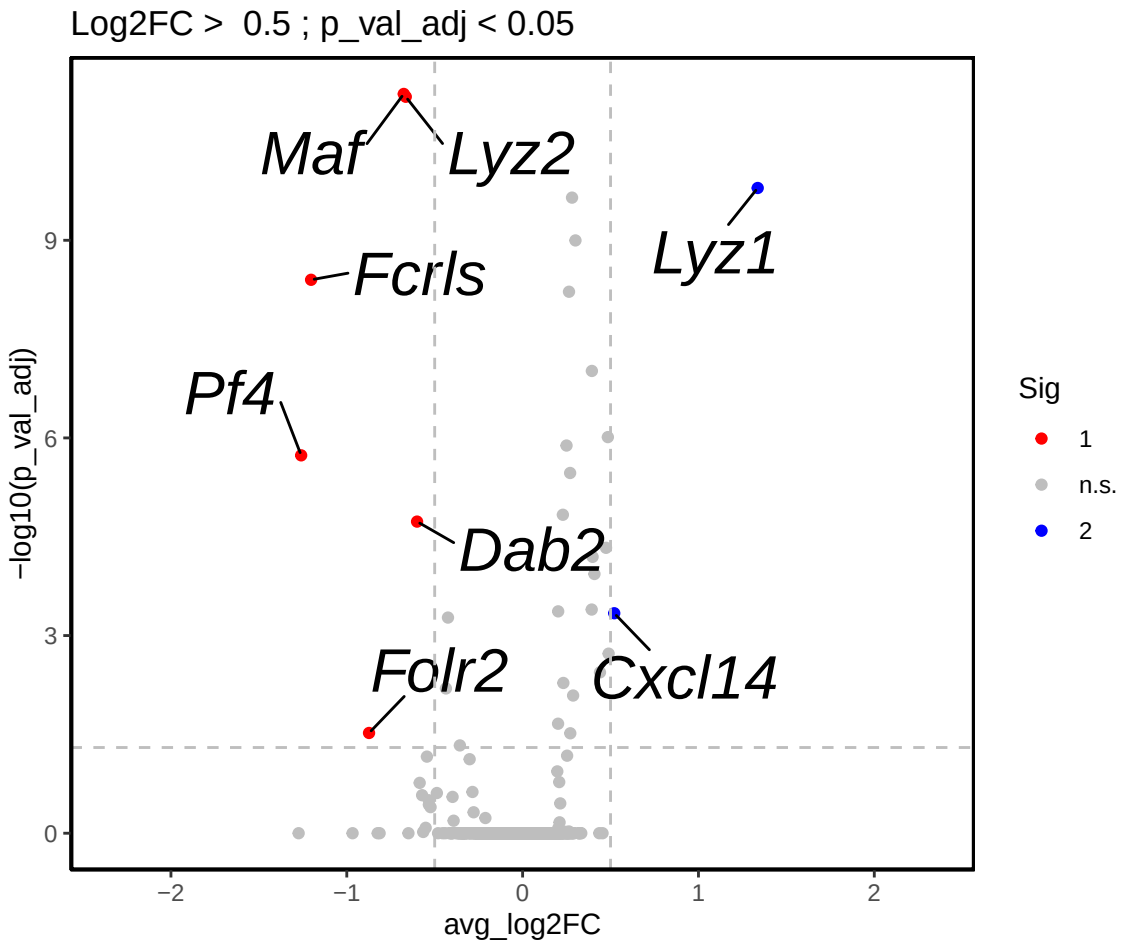
7.1.1 Volcano plot of DE genes

```
suppressMessages({
  library(dplyr)
  library(ggrepel)
  })
```

Let's set a threshold of log2FC and p_val_adj and plot them all:

```
threshold.log2fc <- 0.5
threshold.adjp <- 0.05
cMafKO_vs_IM.volcano = mutate(cMafKO_vs_IM,
 Sig=ifelse((abs(cMafKO_vs_IM$avg_log2FC) > threshold.log2fc)&(cMafKO_vs_
    IM$p_val_adj < threshold.adjp), "Sig", "n.s."))
# add two colors to 2 sig lists
cMafKO_vs_IM.volcano$Sig [ cMafKO_vs_IM.volcano$avg_log2FC < -threshold.
   log2fc & cMafKO_vs_IM.volcano$p_val_adj < threshold.adjp ] <- "1"

```

```
cMafKO_vs_IM.volcano$Sig [ cMafKO_vs_IM.volcano$avg_log2FC > threshold.
    log2fc & cMafKO_vs_IM.volcano$p_val_adj < threshold.adjp ] <- "2"

cMafKO_vs_IM.volcano$Gene <- rownames(cMafKO_vs_IM.volcano)
Gene.to.show.ValcanoPlot <- rownames(cMafKO_vs_IM.volcano[cMafKO_vs_IM.
    volcano$Sig != "n.s.", ])

p <- ggplot(cMafKO_vs_IM.volcano, aes(avg_log2FC, -log10(p_val_adj))) +
    geom_point(aes(col=Sig)) + scale_color_manual(values=c( `1`="red", `n.s
    .`="grey", `2`="blue"))

# set axis lim:
axis.lim <- max(abs(cMafKO_vs_IM.volcano$avg_log2FC)) + 1

p + geom_text_repel(data=filter(cMafKO_vs_IM.volcano, Gene %in% Gene.to.
    show.ValcanoPlot), size = 8, aes(label=Gene, fontface = "italic"), box.
    padding = 1) + xlim(c(-axis.lim, axis.lim)) + theme_classic() + theme(
    panel.border = element_rect(colour = "black", fill = NA, size = 1)) +
    geom_hline(yintercept = -log10(threshold.adjp), linetype='dashed', col
    = 'grey') + geom_vline(xintercept = c(-threshold.log2fc, threshold.
    log2fc),  linetype='dashed', col = 'grey') + ggtitle(paste("Log2FC > ",
     threshold.log2fc, "; p_val_adj <", threshold.adjp))
```



```
write.csv(cMafKO_vs_IM.volcano[Gene.to.show.ValcanoPlot, ] %>% arrange(.,
   desc(avg_log2FC)) ,
          file = paste("./Mafb-deficient_vs_IM.DEgenes.Log2FC", threshold.
             log2fc, ".adjPval", threshold.adjp, ".results.csv", sep = "")
              )
```

```
ggsave(filename = "../Figures/VolcanoPlot_DE_IM_ctrol_vs_cMafKO.pdf",
       width = 6, height = 5)
```

8 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] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] ggrepel_0.9.1      dplyr_1.0.8        RColorBrewer_1.1-2 ggplot2_3
   .3.5
## [5] SeuratObject_4.0.4 Seurat_4.1.0
##
## loaded via a namespace (and not attached):
##   [1] ggbeeswarm_0.6.0      Rtsne_0.15            colorspace_2.0-3
##   [4] deldir_1.0-6          ellipsis_0.3.2        ggridges_0.5.3
##   [7] rstudioapi_0.13       spatstat.data_2.1-2   farver_2.1.0
##  [10] leiden_0.3.9          listenv_0.8.0         RSpectra_0.16-0
##  [13] fansi_1.0.2           codetools_0.2-18      splines_4.0.3
##  [16] knitr_1.37            polyclip_1.10-0       jsonlite_1.7.3
##  [19] Cairo_1.5-14          ica_1.0-2             cluster_2.1.0
##  [22] png_0.1-7             uwot_0.1.11           shiny_1.7.1
##  [25] sctransform_0.3.3     spatstat.sparse_2.1-0 compiler_4.0.3
##  [28] httr_1.4.2            assertthat_0.2.1      Matrix_1.4-0
##  [31] fastmap_1.1.0         lazyeval_0.2.2        limma_3.46.0
##  [34] cli_3.2.0             later_1.3.0           htmltools_0.5.2
##  [37] tools_4.0.3           igraph_1.2.11         gtable_0.3.0
```

```
##  [40] glue_1.6.1             RANN_2.6.1             reshape2_1.4.4          38
##  [43] Rcpp_1.0.8             scattermore_0.8        vctrs_0.3.8             39
##  [46] nlme_3.1-155           lmtest_0.9-39          spatstat.random_2.1-0   40
##  [49] xfun_0.29              stringr_1.4.0          globals_0.14.0          41
##  [52] mime_0.12              miniUI_0.1.1.1         lifecycle_1.0.1         42
##  [55] irlba_2.3.5            goftest_1.2-3          future_1.24.0           43
##  [58] MASS_7.3-53            zoo_1.8-9              scales_1.1.1            44
##  [61] spatstat.core_2.4-0    promises_1.2.0.1       spatstat.utils_2.3-0    45
##  [64] parallel_4.0.3         yaml_2.3.5             reticulate_1.24         46
##  [67] pbapply_1.5-0          gridExtra_2.3          ggrastr_1.0.1           47
##  [70] rpart_4.1-15           stringi_1.7.6          highr_0.9               48
##  [73] rlang_1.0.1            pkgconfig_2.0.3        matrixStats_0.61.0      49
##  [76] evaluate_0.15          lattice_0.20-41        ROCR_1.0-11             50
##  [79] purrr_0.3.4            tensor_1.5             patchwork_1.1.1         51
##  [82] htmlwidgets_1.5.4      labeling_0.4.2         cowplot_1.1.1           52
##  [85] tidyselect_1.1.1       parallelly_1.30.0      RcppAnnoy_0.0.19        53
##  [88] plyr_1.8.6             magrittr_2.0.2         R6_2.5.1                54
##  [91] generics_0.1.2         DBI_1.1.2              pillar_1.7.0            55
##  [94] withr_2.4.3            mgcv_1.8-33            fitdistrplus_1.1-6      56
##  [97] survival_3.2-7         abind_1.4-5            tibble_3.1.6            57
## [100] future.apply_1.8.1     crayon_1.5.0           KernSmooth_2.23-20      58
## [103] utf8_1.2.2             spatstat.geom_2.3-2    plotly_4.10.0           59
## [106] rmarkdown_2.11         grid_4.0.3             data.table_1.14.2       60
## [109] digest_0.6.29          xtable_1.8-4           tidyr_1.2.0             61
## [112] httpuv_1.6.5           munsell_0.5.0          beeswarm_0.4.0          62
## [115] viridisLite_0.4.0      vipor_0.4.5                                    63
```