

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

14 - Compare with cMaf-KO IM

2022-01-31 00:37:09 +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	2
2 Load packages and data	2
3 Compare populations	2
3.1 Re-process data	2
4 Identify the CD206+ and CD206- IMs	6
5 Focus on the cMAF-KO vs Control	9
6 Proliferation comparison	14
7 Comparison between cMAF-deficient IMs population and control IMs	16
7.1 DE genes between Mafb- neo and IM population	16
8 Session information	21

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.

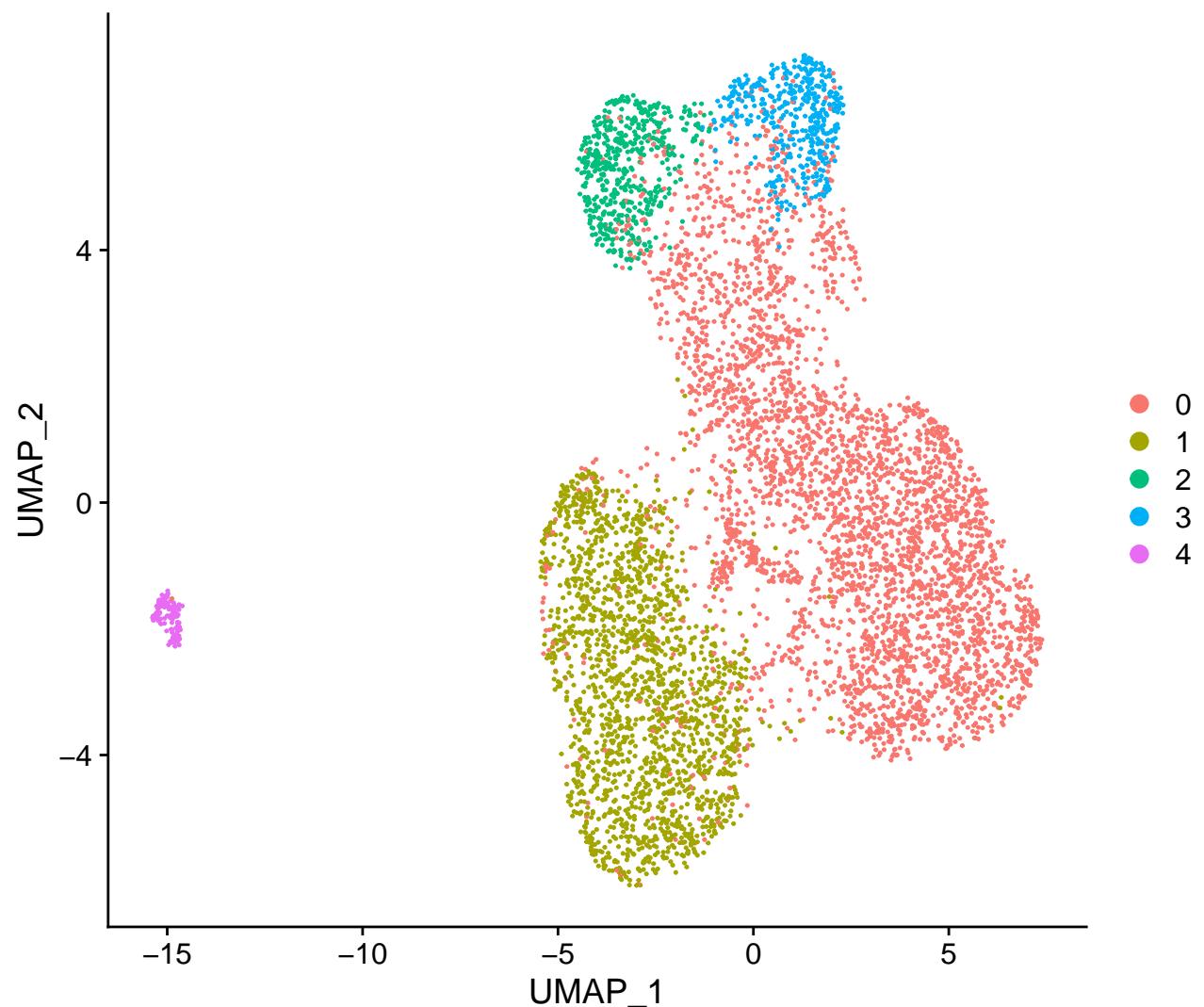
3 Compare populations

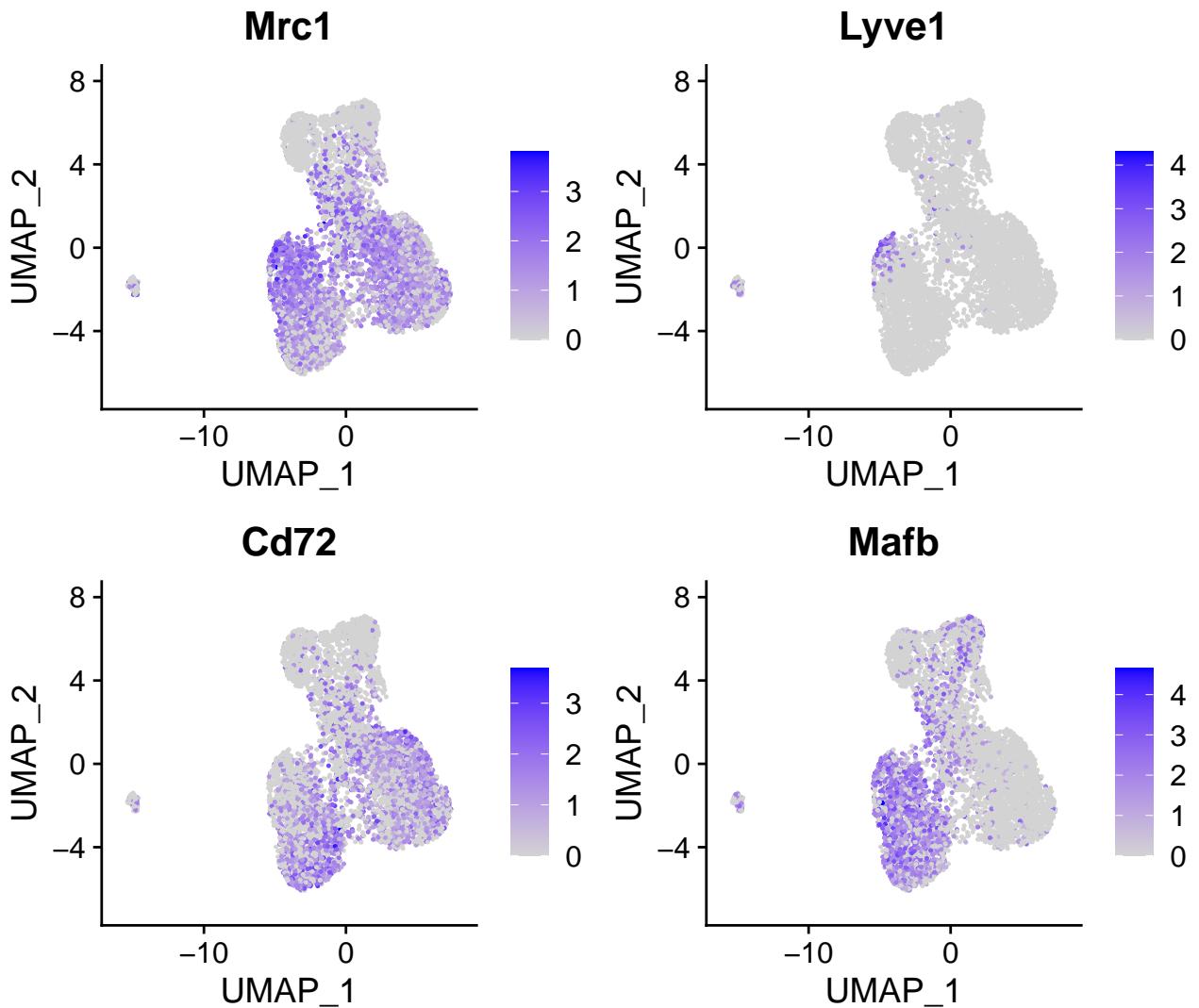
3.1 Re-process data

```
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)
```

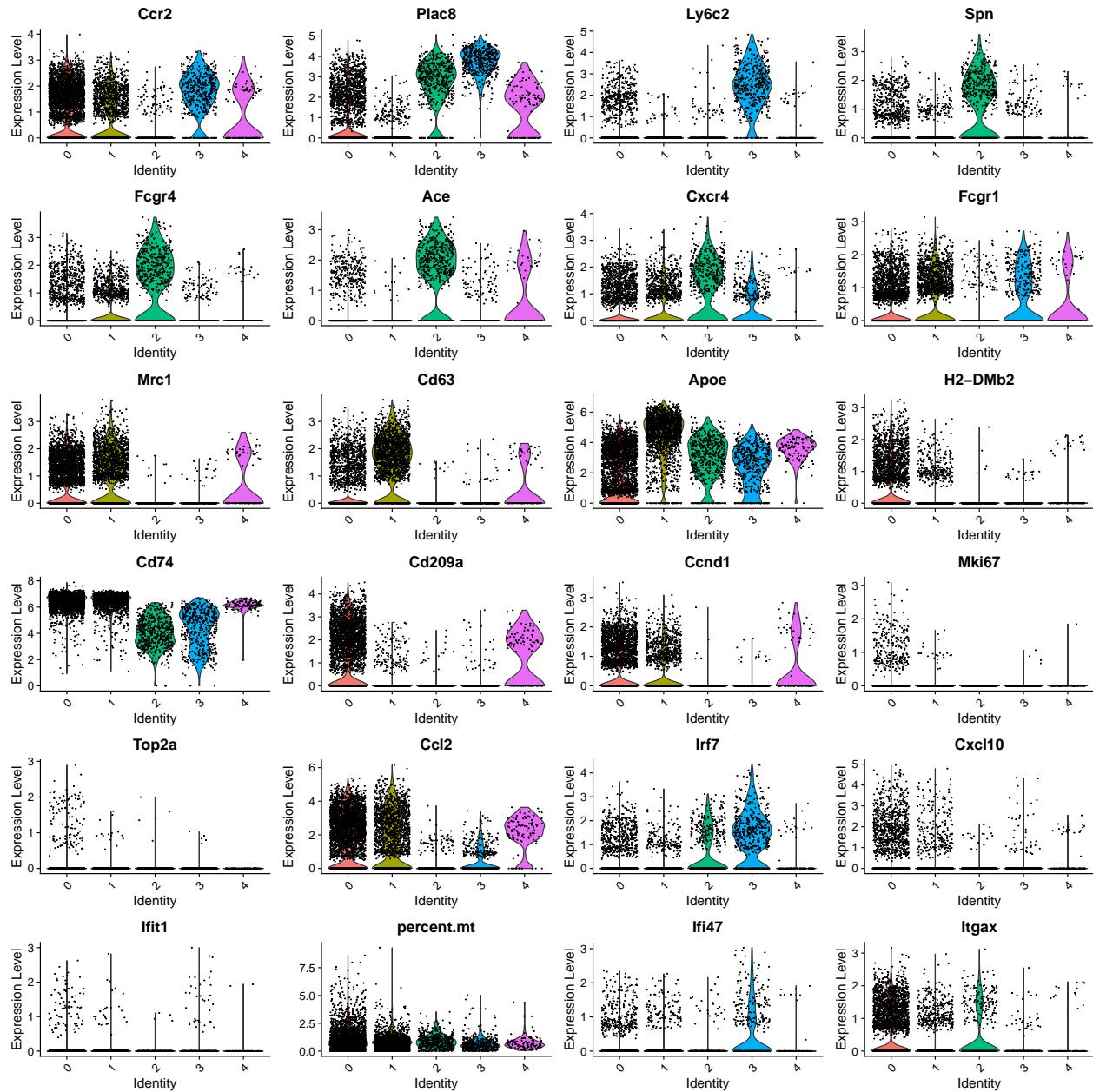
```
so <- FindNeighbors(so, dims = 1:8, verbose = FALSE)  
so <- FindClusters(so, resolution = 0.12, verbose = FALSE)
```

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





```
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"
  
```

```

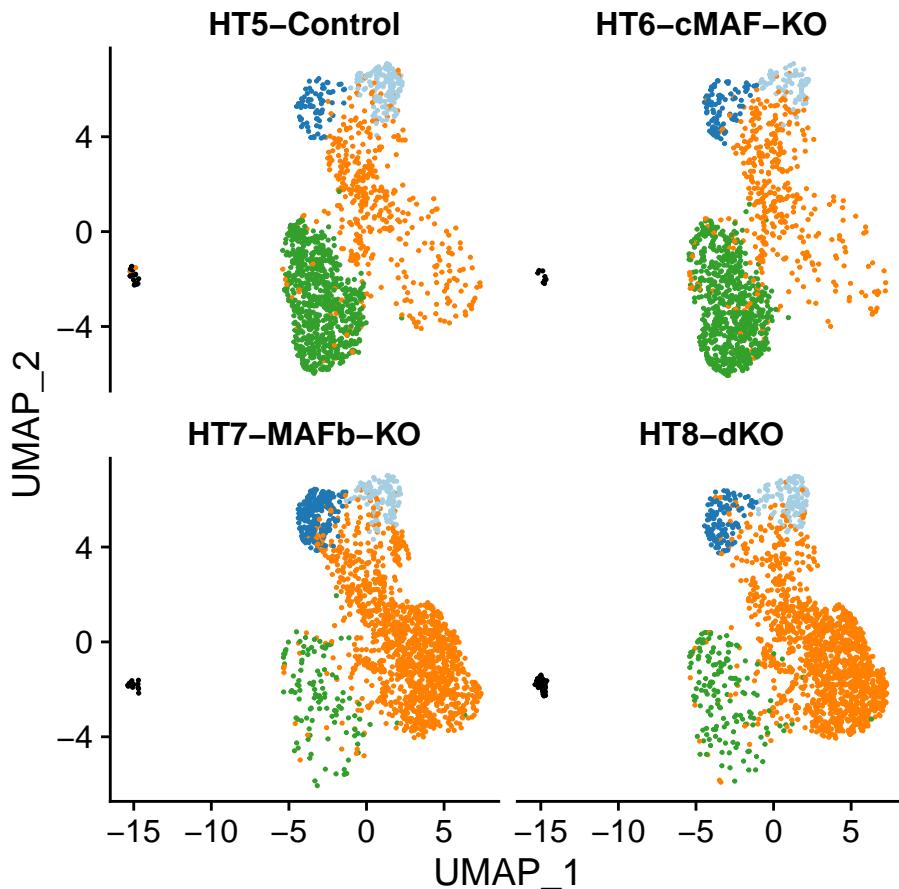
pal4 <- c(
  "#A6CEE3", # cMo
  "#1F78B4", # pMo
  "#33A02C", # CD206 IM
  "#FF7F00", # Mafb - neo
  "black" # unknown
  
```

```

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

```

7
8
9



```

ggsave(filename = "../Figures/UMAPplot_All_samplesMaf_separate_2columns.
pdf", width = 5, height = 5)

```

1
2

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)

```

1
2
3
4
5
6
7
8
9
10

```

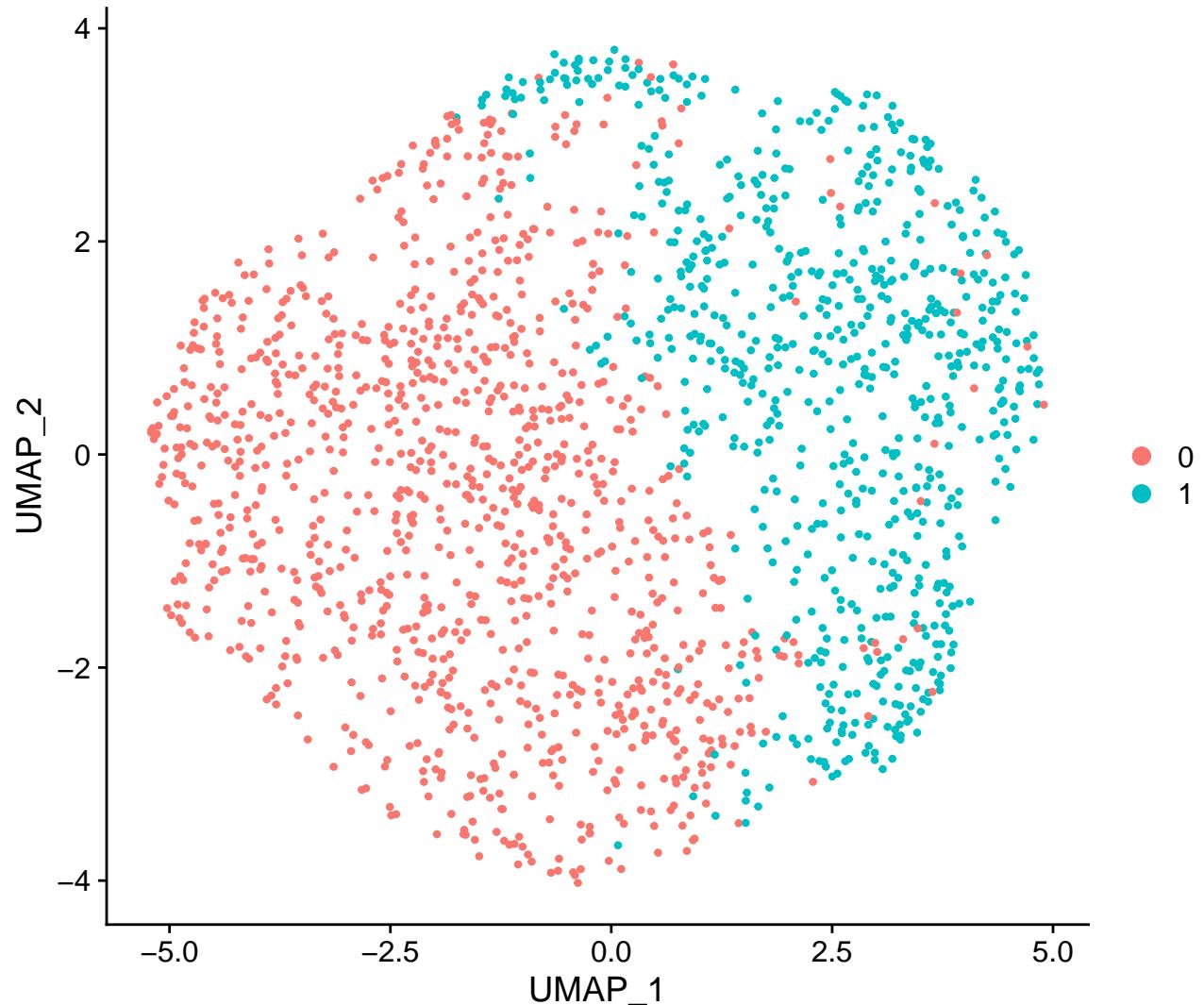
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
## 

```

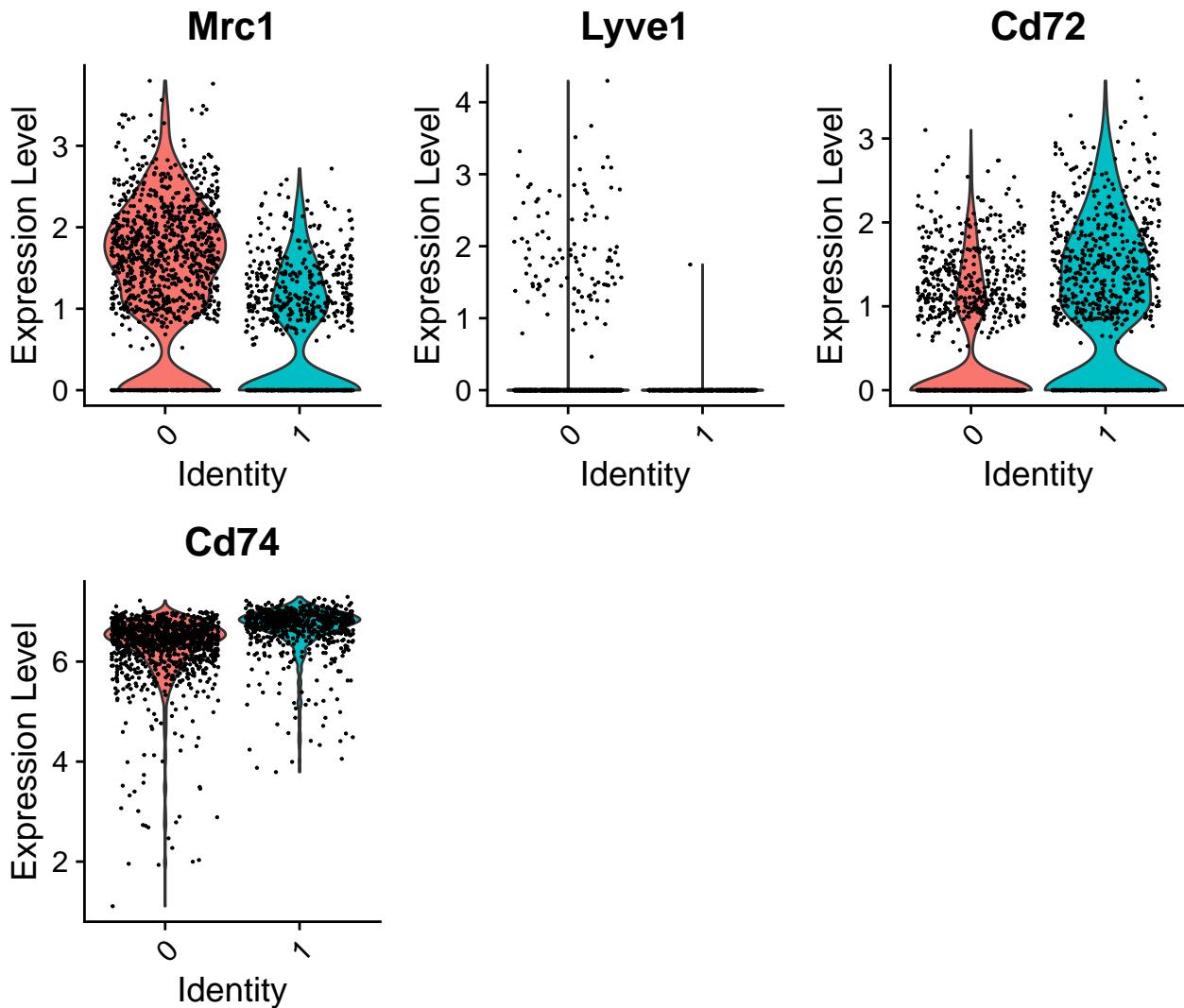
1
2

```
## 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"))
```



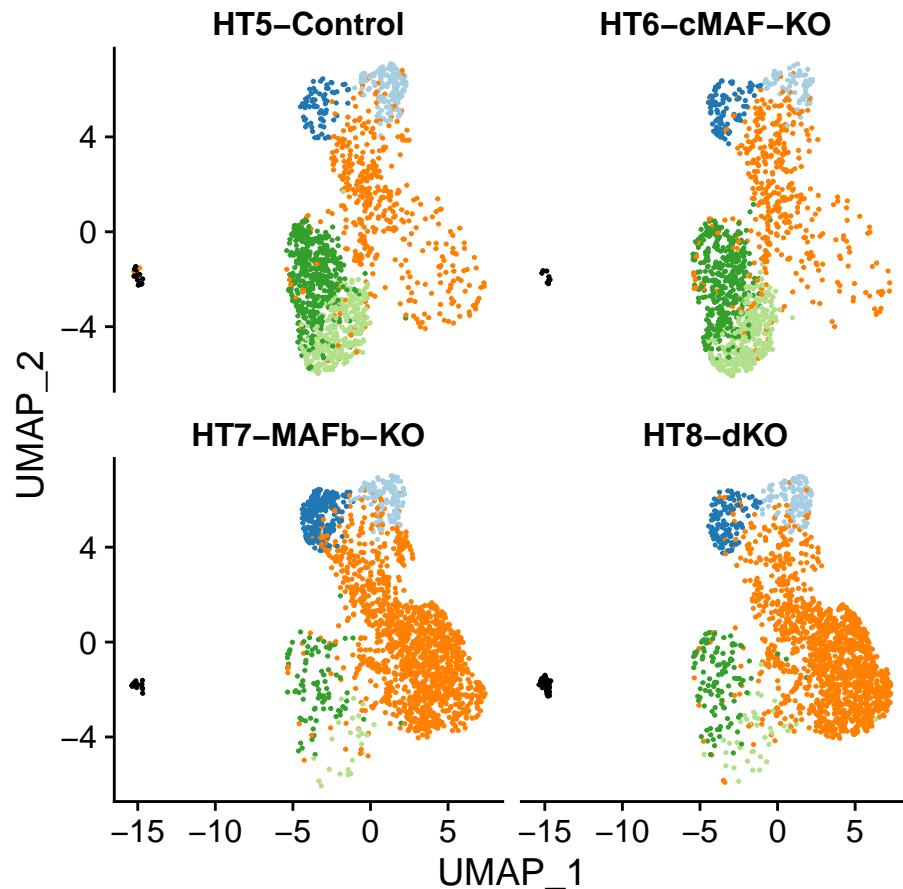
cluster 0: CD206+ IM cluster 1: CD206- IM

```

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_2columns.pdf", width = 5, height = 5) 1
```

5 Focus on the cMAF-KO vs Control

```
so <- subset(so, subset = group == c("HT5-Control", "HT6-cMAF-KO")) 1
```

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" 1
2
3
4
```

```
pal4.control <- c(
  "black", # unknown
  "#F1F1F1", # grey for another sample
  "#B2DF8A", # MHCII IM
  "#33A02C", # CD206 IM
  "#FF7F00", # Mafb-neo
  "#A6CEE3", # cMo 1
2
3
4
5
6
7
```

```

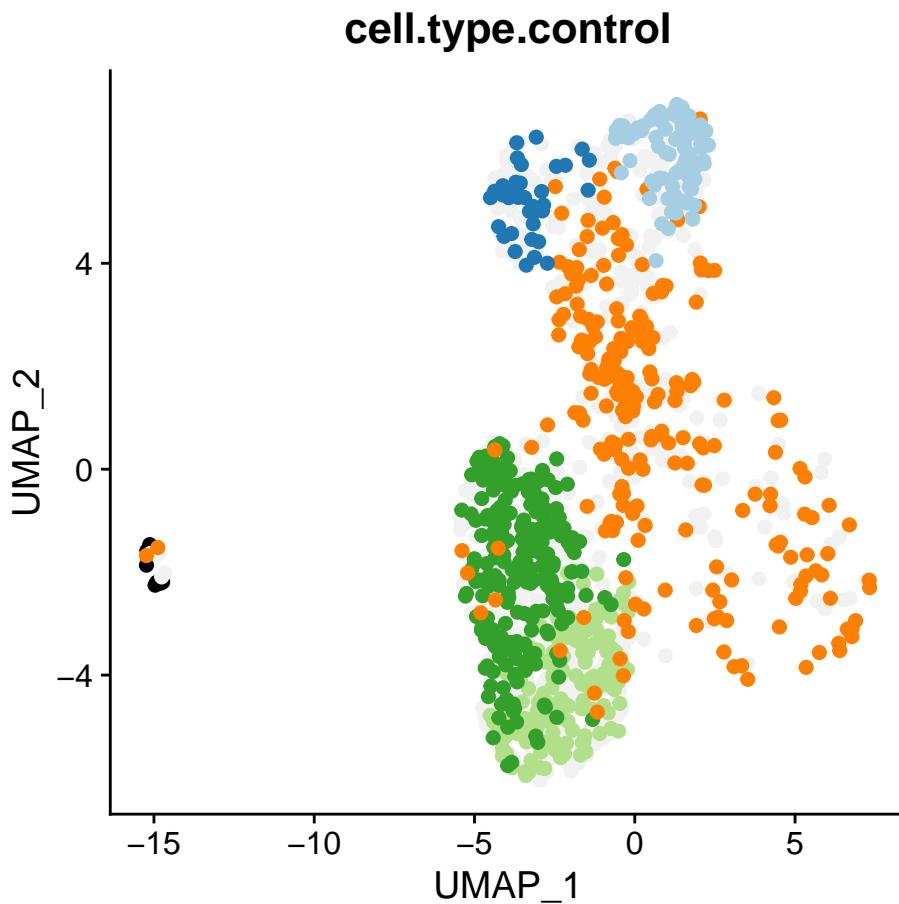
  "#1F78B4" # pMo
)

```

```

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"

```

```

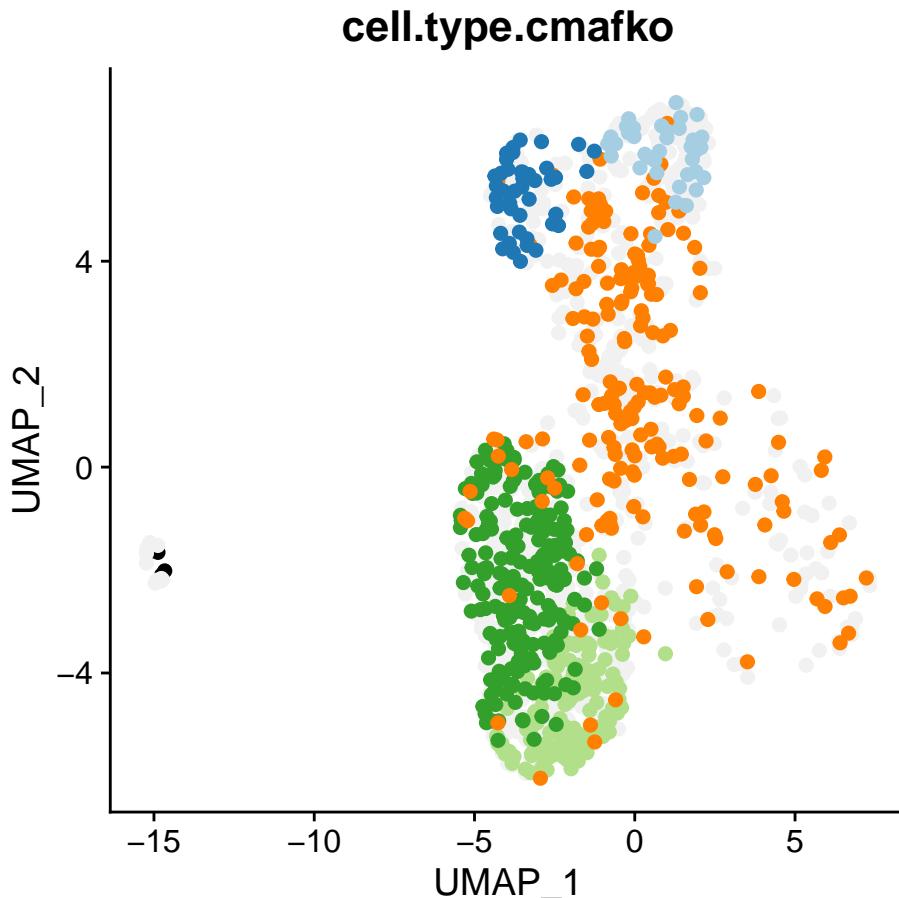
DimPlot(so, cols = pal4.control, group.by = "cell.type.cmafko", pt.size =
  2,

```

```

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)

```

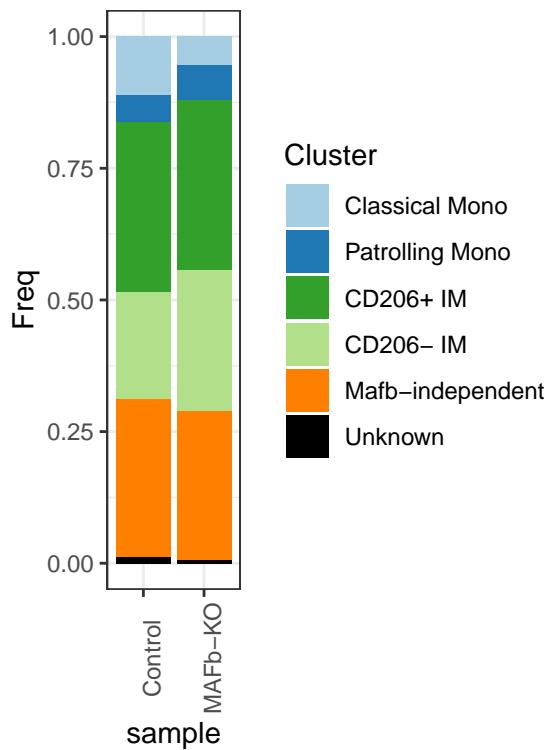
See population frequencies:

```

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))

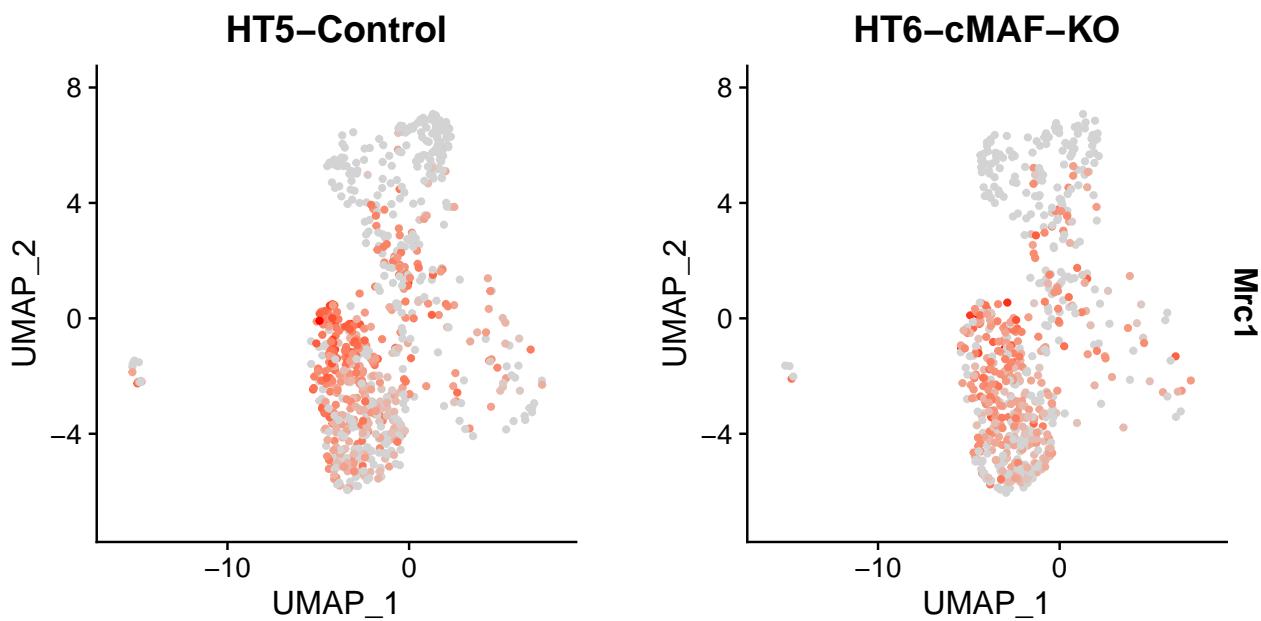
```



```
ggsave(filename = ".../Figures/Barplot_Ctl_cMafKO_population_frequency.pdf" 1
      ,  
      width = 3, height = 4) 2
```

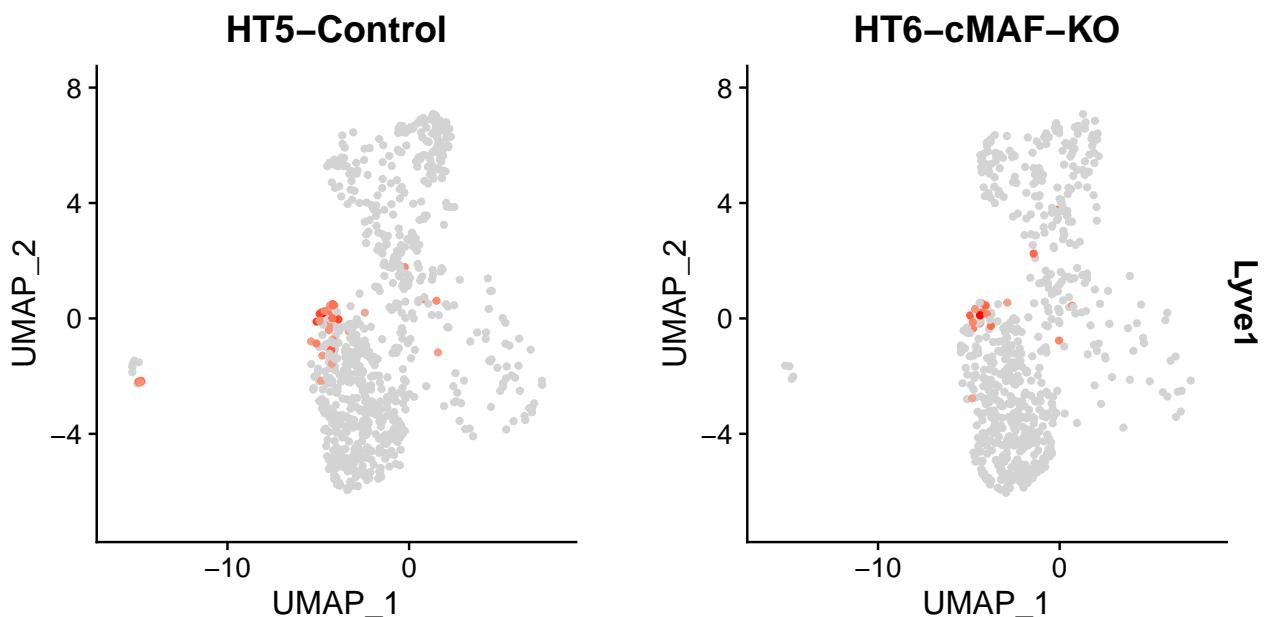
Show CD206+ IM markers

```
FeaturePlot(so, features = "Mrc1", cols = c("lightgray", "red"), min. 1
cutoff = 0.5, split.by = "group") 2
```



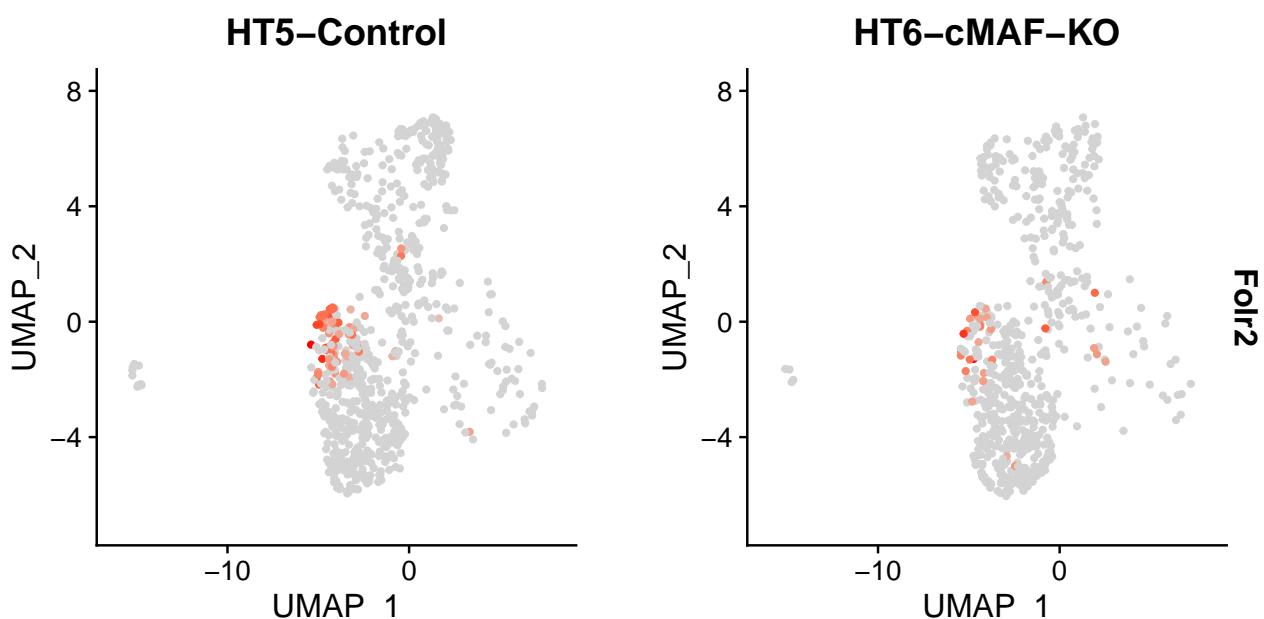
```
ggsave(filename = "../Figures/VlnPlot_Mrc1_in_cMafKO_Control.pdf", width = 1  
       8, height = 4)
```

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



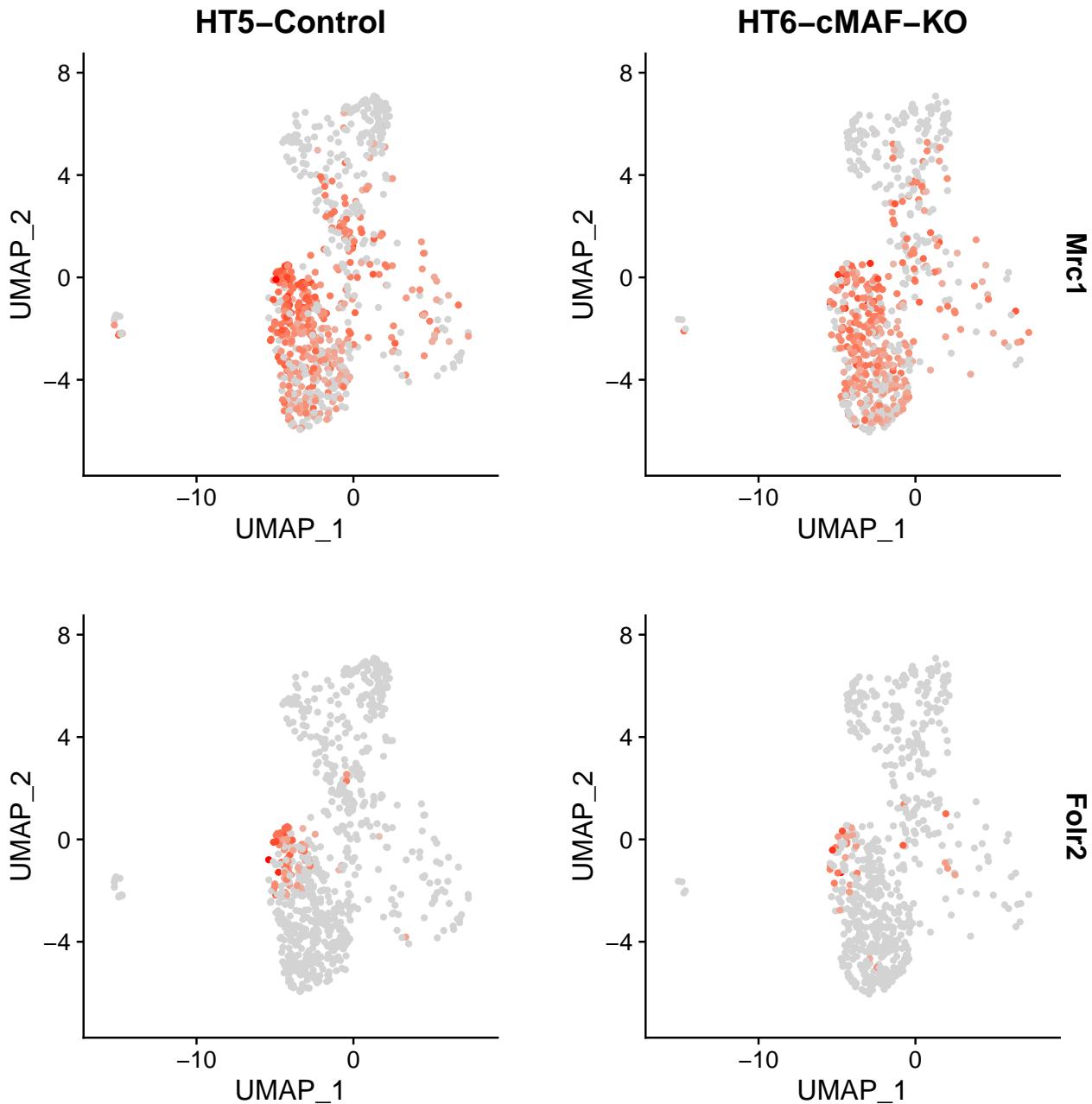
```
ggsave(filename = "../Figures/VlnPlot_Lyve1_in_cMafKO_control.pdf", width = 1  
       8, height = 4)
```

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



```
ggsave(filename = "../Figures/VlnPlot_Folr2_in_cMafKO_control.pdf", width = 8, height = 4) 1
```

```
FeaturePlot(so, features = c("Mrc1", "Folr2"), cols = c("lightgray", "red"), split.by = "group") 1
```



```
ggsave(filename = "../Figures/Featureplot_Mrc1_Folr2_in_cMafKO_control.pdf", height = 8, width = 8) 1
```

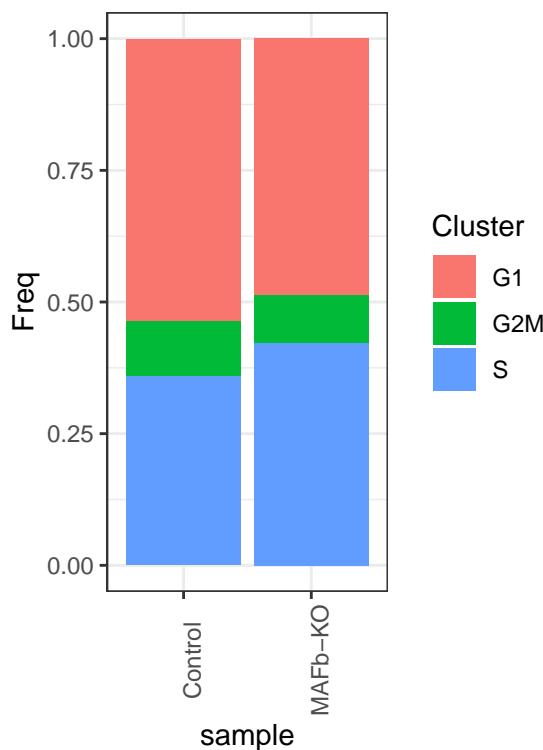
6 Proliferation comparison

```

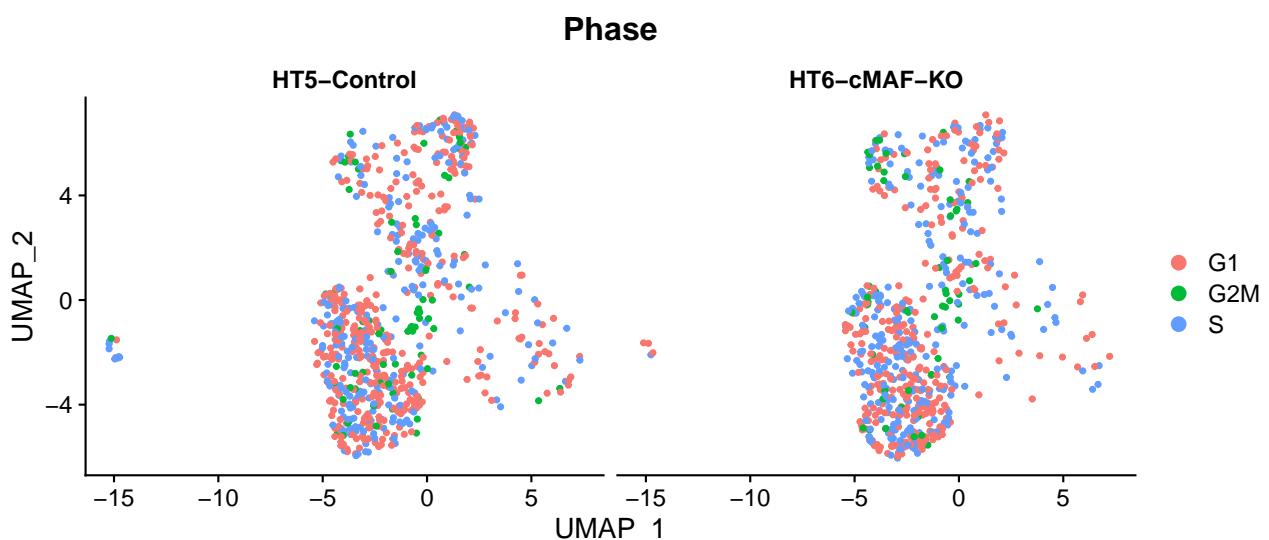
freq.celltype.list <- list(
  `Control` = Seurat2CellFreqTable(subset(so, subset = group == "HT5-
  Control"), slotName = "Phase"),
  `MAFb-KO` = Seurat2CellFreqTable(subset(so, subset = group == "HT6-cMAF-
  KO"), slotName = "Phase")
)

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.

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

7.1 DE genes between Mafb- neo and IM population

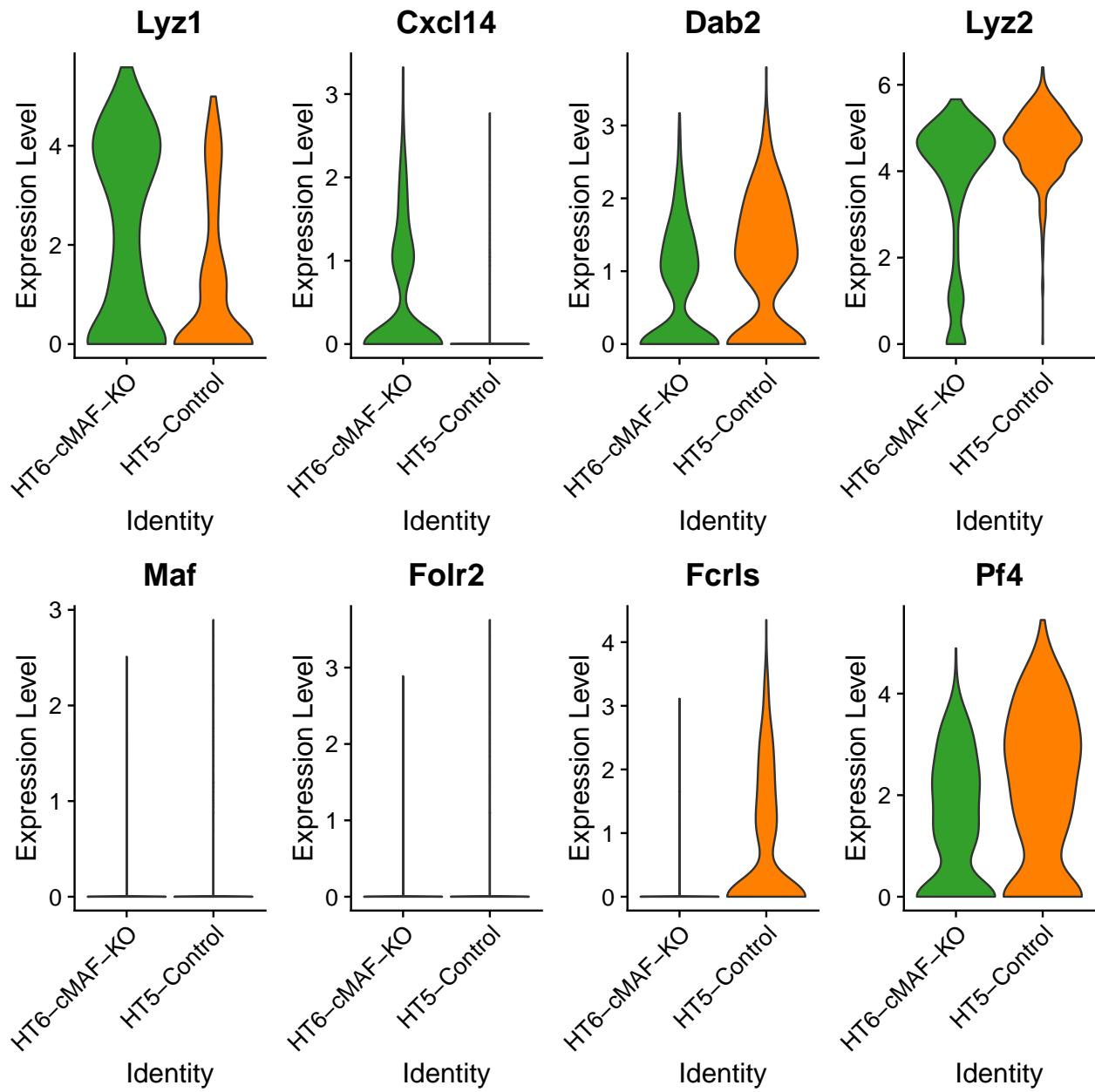
```
1 library(dplyr)
2 cMafKO_vs_IM <- FindMarkers(ims,
3                               ident.1 = "HT6-cMAF-KO",
4                               ident.2 = "HT5-Control",
5                               logfc.threshold = 0,
6                               verbose = FALSE)
7
8
9 # keep only adj p value < 0.05 and logFC > 0.5 as significant markers.
10 cMafKO_vs_IM.markers <- cMafKO_vs_IM[cMafKO_vs_IM$p_val_adj < 0.05 & abs(
11   cMafKO_vs_IM$avg_log2FC) > 0.5, ]
12 cMafKO_vs_IM.markers <- cMafKO_vs_IM.markers[order(cMafKO_vs_IM.markers$avg_log2FC, decreasing = TRUE), ]
13 nrow(cMafKO_vs_IM.markers)
```

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

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

Show in vlnplot

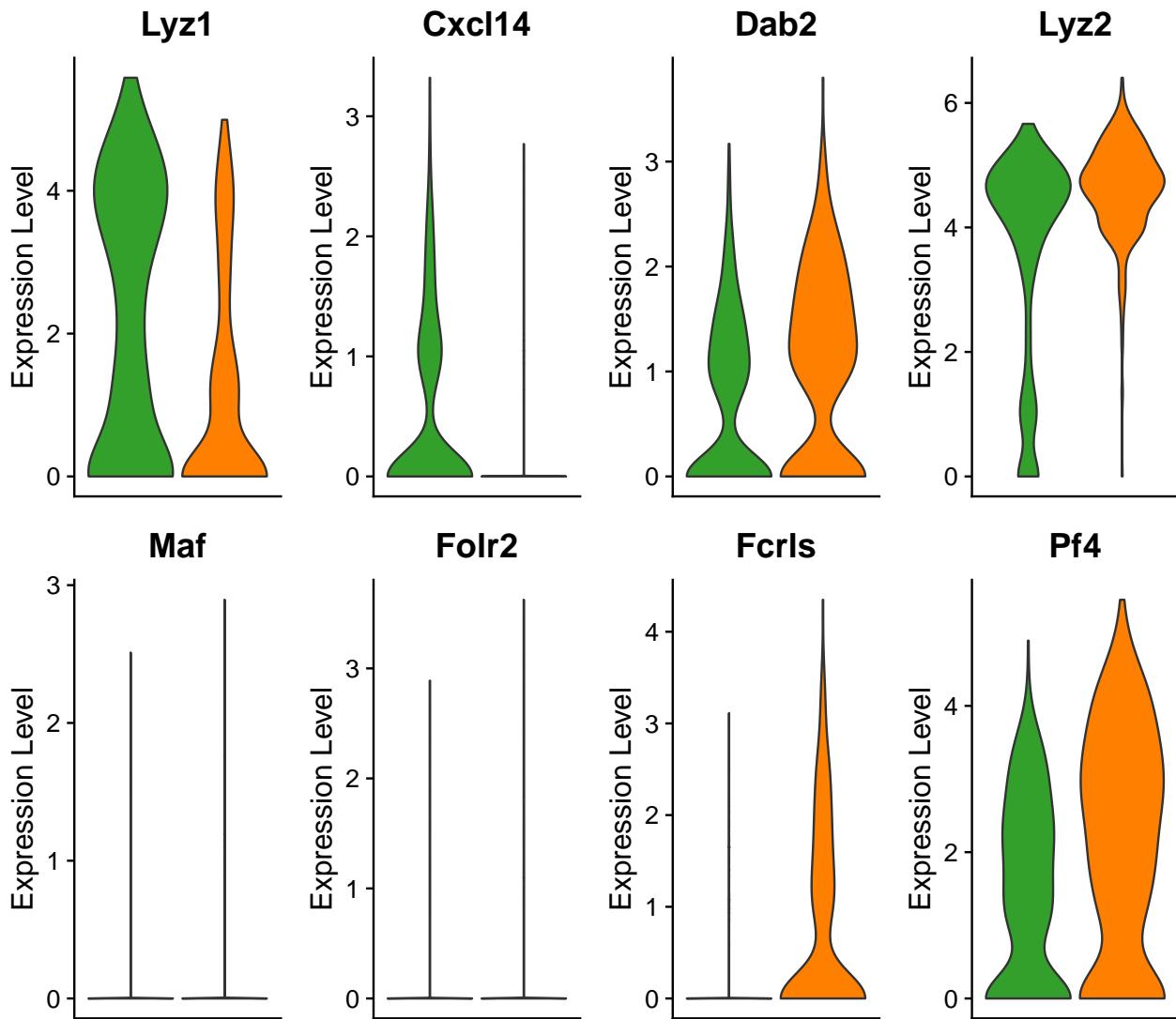
```
1 p <- VlnPlot(ims, features = c(rownames(cMafKO_vs_IM.markers)), cols = c(
2   "#33A02C", "#FF7F00"), ncol = 4, pt.size = 0)
3 p
```



Show vlnplot without label:

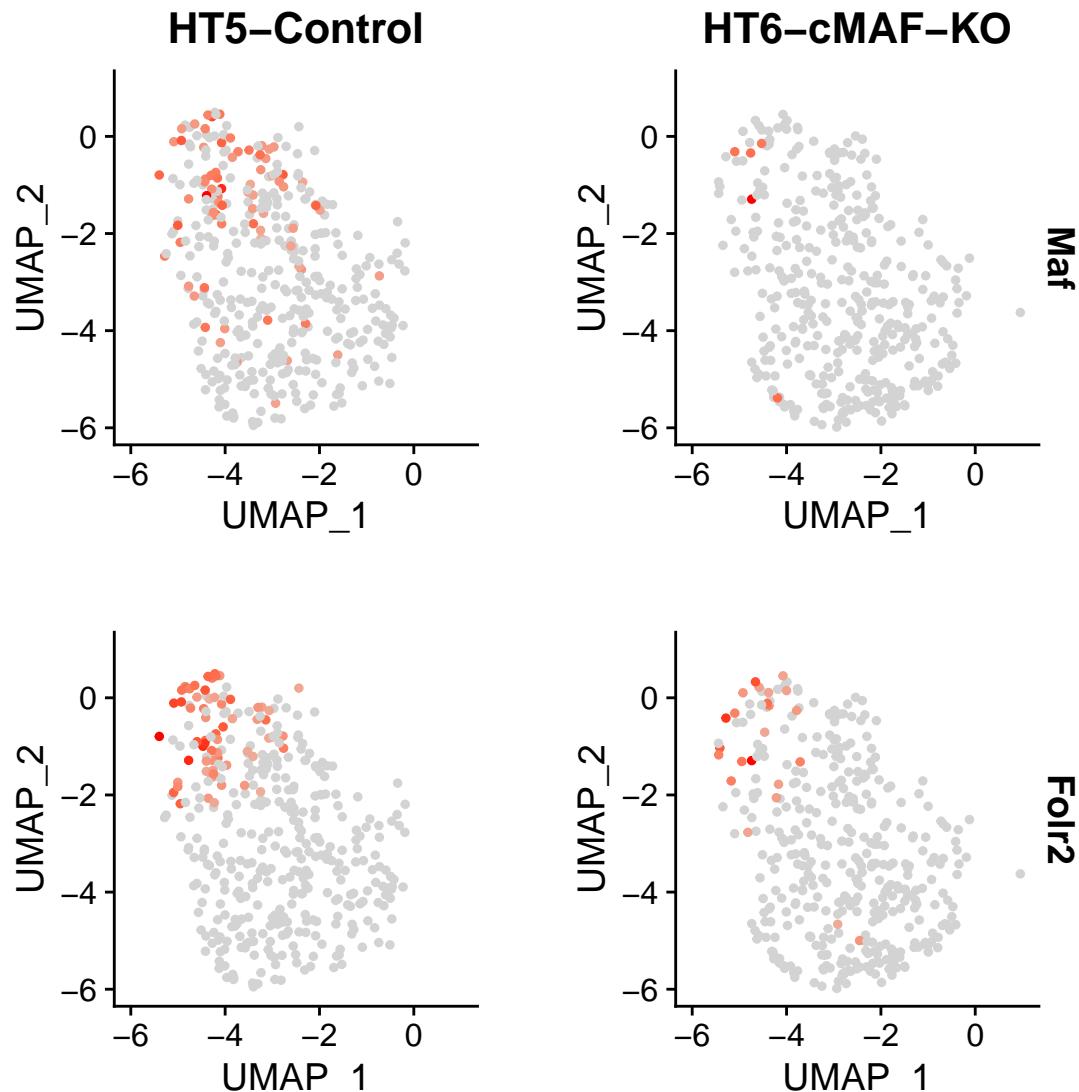
```
p & theme(axis.title.x=element_blank(),
          axis.text.x=element_blank(),
          axis.ticks.x=element_blank())
```

1
2
3



```
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)
})
```

1
2
3
4

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

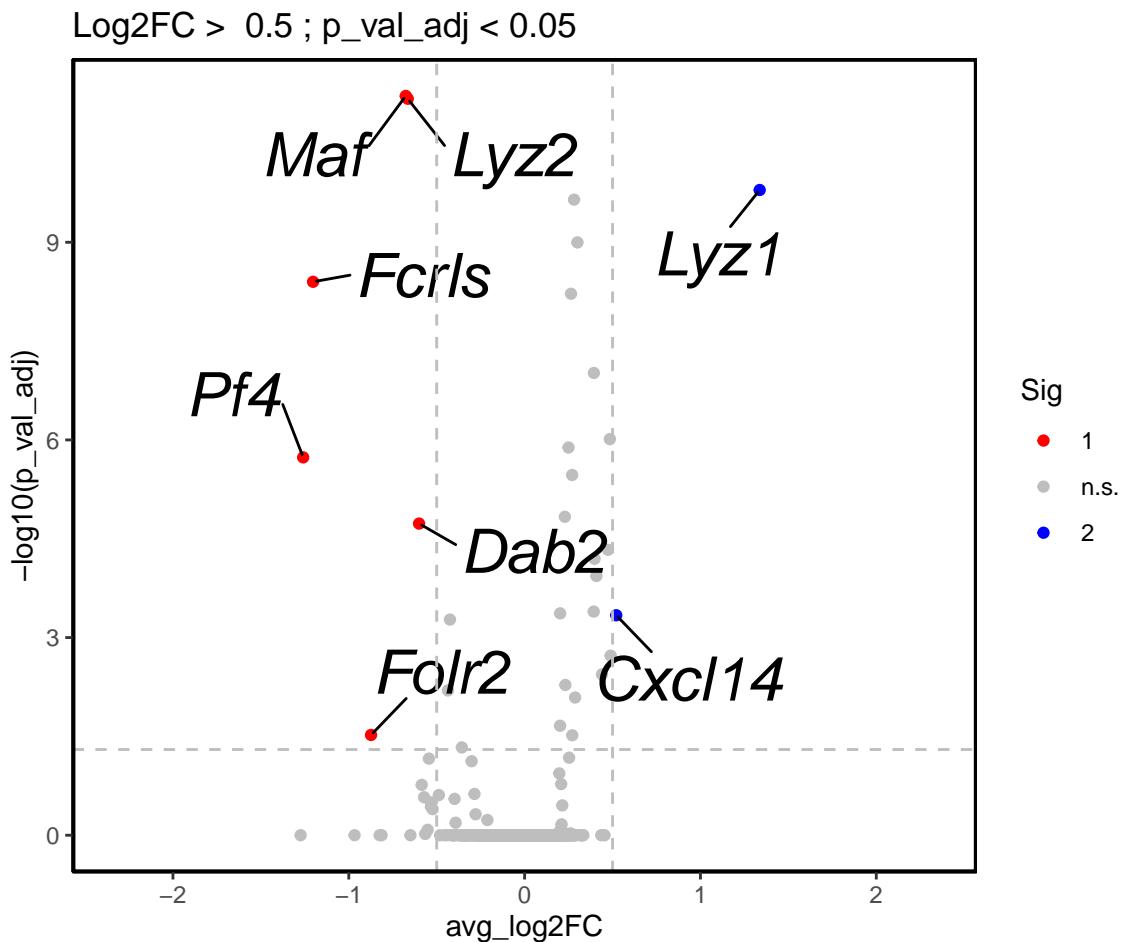
```
threshold.log2fc <- 0.5
threshold.adjp <- 0.05
cMafKO_vs_IM.volcano = mutate(cMafKO_vs_IM,
  Sig=ifelse((abs(cMafKO_vs_IM$avg_log2FC) > threshold.log2fc)&(cMafKO_vs_
  IM$p_val_adj < threshold.adjp), "Sig", "n.s."))
# add two colors to 2 sig lists
cMafKO_vs_IM.volcano$Sig [ cMafKO_vs_IM.volcano$avg_log2FC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

1
2
3
4
5
6
7

```

cMafKO_vs_IM.volcano$Sig [ cMafKO_vs_IM.volcano$avg_log2FC > threshold.
8   log2fc & cMafKO_vs_IM.volcano$p_val_adj < threshold.adjp ] <- "2"
9
cMafKO_vs_IM.volcano$Gene <- rownames(cMafKO_vs_IM.volcano)
10 Gene.to.show.ValcanoPlot <- rownames(cMafKO_vs_IM.volcano[cMafKO_vs_IM.
11   volcano$Sig != "n.s.", ])
12
p <- ggplot(cMafKO_vs_IM.volcano, aes(avg_log2FC, -log10(p_val_adj))) +
13   geom_point(aes(col=Sig)) + scale_color_manual(values=c(`1`="red",
14   `n.s.`="grey", `2`="blue"))
15
# set axis lim:
16 axis.lim <- max(abs(cMafKO_vs_IM.volcano$avg_log2FC)) + 1
17
18 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(., 1
desc(avg_log2FC)) ,
          file = paste("./Mafb-deficient_vs_IM.DEgenes.Log2FC", threshold. 2
log2fc, ".adjPval", threshold.adjp, ".results.csv", sep = "") )

```

```

ggsave(filename = "../Figures/VolcanoPlot_DE_IM_ctrl_vs_cMafKO.pdf", 1
width = 6, height = 5) 2

```

8 Session information

R sesssion:

```
sessionInfo() 1
```

```

## R version 4.0.3 (2020-10-10) 1
## Platform: x86_64-pc-linux-gnu (64-bit) 2
## Running under: Ubuntu 20.04.3 LTS 3
## 4
## Matrix products: default 5
## BLAS: /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3 6
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/liblapack.so.3 7
## 8
## locale: 9
## [1] LC_CTYPE=en_US.UTF-8           LC_NUMERIC=C 10
## [3] LC_TIME=en_GB.UTF-8          LC_COLLATE=en_US.UTF-8 11
## [5] LC_MONETARY=en_GB.UTF-8       LC_MESSAGES=en_US.UTF-8 12
## [7] LC_PAPER=en_GB.UTF-8         LC_NAME=C 13
## [9] LC_ADDRESS=C                  LC_TELEPHONE=C 14
## [11] LC_MEASUREMENT=en_GB.UTF-8    LC_IDENTIFICATION=C 15
## 16
## attached base packages: 17
## [1] stats      graphics   grDevices  utils      datasets   methods    base 18
## 19
## other attached packages: 20
## [1] ggrepel_0.9.1      dplyr_1.0.7        RColorBrewer_1.1-2 ggplot2_3 21
## .3.5
## [5] SeuratObject_4.0.4  Seurat_4.0.5 22
## 23
## loaded via a namespace (and not attached): 24
## [1] nlme_3.1-153          spatstat.sparse_2.0-0 matrixStats_0.61.0 25
## [4] RcppAnnoy_0.0.19      httr_1.4.2            sctransform_0.3.2 26
## [7] tools_4.0.3           utf8_1.2.2           R6_2.5.1 27
## [10] irlba_2.3.5          rpart_4.1-15        KernSmooth_2.23-20 28
## [13] uwot_0.1.11          mgcv_1.8-33         DBI_1.1.1 29
## [16] lazyeval_0.2.2         colorspace_2.0-2    withr_2.4.3 30
## [19] tidyselect_1.1.1       gridExtra_2.3       compiler_4.0.3 31
## [22] plotly_4.10.0          labeling_0.4.2      scales_1.1.1 32
## [25] spatstat.data_2.1-0    lmtest_0.9-39       ggridges_0.5.3 33
## [28] pbapply_1.5-0          goftest_1.2-3       stringr_1.4.0 34
## [31] digest_0.6.29          spatstat.utils_2.2-0 rmarkdown_2.11 35
## [34] pkgconfig_2.0.3         htmltools_0.5.2     parallelly_1.29.0 36
## [37] limma_3.46.0           highr_0.9           fastmap_1.1.0 37

```

##	[40]	htmlwidgets_1.5.4	rlang_0.4.12	shiny_1.7.1	38
##	[43]	farver_2.1.0	generics_0.1.1	zoo_1.8-9	39
##	[46]	jsonlite_1.7.2	ica_1.0-2	magrittr_2.0.1	40
##	[49]	patchwork_1.1.1	Matrix_1.3-4	Rcpp_1.0.7	41
##	[52]	munsell_0.5.0	fansi_0.5.0	abind_1.4-5	42
##	[55]	reticulate_1.22	lifecycle_1.0.1	stringi_1.7.6	43
##	[58]	yaml_2.2.1	MASS_7.3-53	Rtsne_0.15	44
##	[61]	plyr_1.8.6	grid_4.0.3	parallel_4.0.3	45
##	[64]	listenv_0.8.0	promises_1.2.0.1	crayon_1.4.2	46
##	[67]	deldir_1.0-6	miniUI_0.1.1.1	lattice_0.20-41	47
##	[70]	cowplot_1.1.1	splines_4.0.3	tensor_1.5	48
##	[73]	knitr_1.36	pillar_1.6.4	igraph_1.2.9	49
##	[76]	spatstat.geom_2.3-0	future.apply_1.8.1	reshape2_1.4.4	50
##	[79]	codetools_0.2-18	leiden_0.3.9	glue_1.5.1	51
##	[82]	evaluate_0.14	data.table_1.14.2	png_0.1-7	52
##	[85]	vctrs_0.3.8	httpuv_1.6.3	polyclip_1.10-0	53
##	[88]	gttable_0.3.0	RANN_2.6.1	purrrr_0.3.4	54
##	[91]	spatstat.core_2.3-2	tidyR_1.1.4	scattermore_0.7	55
##	[94]	future_1.23.0	assertthat_0.2.1	xfun_0.28	56
##	[97]	mime_0.12	xtable_1.8-4	RSpectra_0.16-0	57
##	[100]	later_1.3.0	survival_3.2-7	viridisLite_0.4.0	58
##	[103]	tibble_3.1.6	cluster_2.1.0	globals_0.14.0	59
##	[106]	fitdistrplus_1.1-6	ellipsis_0.3.2	ROCR_1.0-11	60