Mafb-restricted local monocyte proliferation precedes lung interstitial macrophage differentiation

8-SCENIC analysis

2022-03-09 16:33:03 +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 **2**

2 Prepare data **2**
2.1 Load Seurat objects . . . 2
2.2 Filter only the cells in classic monocytes, patrolling monocytes and interstitial macrophages . 2
2.3 Prepare input data for SCENIC. . . . 2
2.4 Cell info/phenodata . . . 3

3 Initialize settings **3**

4 Co-expression network **3**
4.1 Gene filter/selection . . . 3
4.2 Correlation . . . 4
4.3 GENIE3 . . . 4

5 Build and score the GRN **4**

6 Demonstration of SCENIC results with Seurat **5**

7 Make plots **9**
7.1 Show TF activity . . . 9
7.2 Show TF expression . . . 20

8 Session information **22**

References **24**

1 Description

To predict the potential active transcription factors (TF), Ly6C+ cMo, transit cells, CD206- and CD206+ IM were subjected to SCENIC analysis using SCENIC package.[1] The normalized counts, nFeature_RNA, nCount_RNA in merged Seurat object were used for the initial SCENIC analysis. The genes expressed with a value of 3 in 0.5% of the cells and detected in 1% of the cells were kept for following SCENIC analysis. Co-expression network analysis was made with GENIE3 in the SCENIC package. To represent the SCENIC results, the results of 3.4_regulonsAUC were added to the metadata of Seurat object so that regulon AUC scores could be plot using FeaturePlot function. The top 50 regulons with highest variance were showed in the heatmap with their Z-scores.

2 Prepare data

2.1 Load Seurat objects

The 3D file store 3D embedding for UMAP and TSNE.

```
suppressMessages({
  library(Seurat)
  library(SCENIC)
  library(AUCell)
  library(RcisTarget)
  library(SCopeLoomR)
  library(ggplot2)})

obj <- readRDS("../6-Merge_two_experiments/immune_imdtr3.seuratObject.Rds"
   )
```

2.2 Filter only the cells in classic monocytes, patrolling monocytes and interstitial macrophages

```
obj <- subset(obj, subset = cell.type2 %in% c("Classical_Mono",
                                              "MHCII_IM",
                                              "CD206_IM",
                                              "Transit"))
obj$cell.type2 <- as.character(obj$cell.type2)
```

2.3 Prepare input data for SCENIC.

```
obj.sce <- as.SingleCellExperiment(obj) # we indeed don't need this.
dir.create("data")
saveRDS(obj.sce, file="data/IM-DTR3_3D.sce.Rds")
```

2.3.1 Matrix construction:

```
exprMat <- as.matrix(obj@assays$RNA@data)
exprMat[1:3,1:3]
```

```
dir.create("int")
saveRDS(exprMat, file="int/exprMat.Rds")
```

2.4 Cell info/phenodata

Cell information:

```
cellInfo <- data.frame(CellType = obj@meta.data$cell.type2, nFeature_RNA =
    obj@meta.data$nFeature_RNA, nCount_RNA = obj@meta.data$nCount_RNA)
rownames(cellInfo) <- colnames(obj)
head(cellInfo)
```

Save data for later:

```
saveRDS(cellInfo, file="int/cellInfo.Rds")
```

Color to assign to the variables:

```
colVars <- list(CellType=c("Classical_Mono"="#A6CEE3",
                           "Patrolling_Mono"="#1F78B4",
                           "MHCII_IM"="#B2DF8A",
                           "CD206_IM"="#7E893D",
                           "chemoattractant"="#FB9A99",
                           "Transit"="#E31A1C",
                           "CCR7_IM"="#FDBF6F"))
colVars$CellType <- colVars$CellType[intersect(names(colVars$CellType),
   cellInfo$CellType)]
plot.new(); legend(0,1, fill=colVars$CellType, legend=names(colVars$
   CellType))
```

3 Initialize settings

```
scenicOptions <- initializeScenic(org="mgi",
                                  dbDir="./Data/Scenic_database/cisTarget_
                                     databases/",  # this should be
                                     prepared before, see SCENIC package
                                     instructions
                                  nCores=22 # use appropriate core number
                                  )
  # NOTE: nCores is the number of cores used for calculating. For GENIE3
     analysis is time-consuming and calculation-consuming, make this
     nCores maximum.
```

```
scenicOptions@inputDatasetInfo$cellInfo <- "int/cellInfo.Rds"
scenicOptions@inputDatasetInfo$colVars <- "int/colVars.Rds"
# update scenicOptions object.
saveRDS(scenicOptions, file="int/scenicOptions.Rds")
```

4 Co-expression network

4.1 Gene filter/selection

1. Filter by the total number of reads per gene. minCountsPerGene: By default it keeps only the genes with at least 6 UMI counts across all samples (e.g. the total number the gene would have, if it was expressed with a value of 3 in 1% of the cells). ***NOTE: here we use value of 3 in 0.5% cells as our data is bigger (avoid removing too many useful genes)***

2. Filter by the number of cells in which the gene is detected** (e.g. >0 UMI, or >1 log2(TPM)). minSamples: detected in at least 1% of the cells

```
genesKept <- geneFiltering(exprMat, scenicOptions=scenicOptions,
                           minCountsPerGene=3*.005*ncol(exprMat),
                           minSamples=ncol(exprMat)*.01)
exprMat_filtered <- exprMat[genesKept, ]
dim(exprMat_filtered)
```

```
saveRDS(exprMat_filtered, file = "int/exprMat_filtered.Rds")
saveRDS(genesKept, file = "int/geneKept.Rds")
```

4.2 Correlation

```
source("/home/qiang/Desktop/velocyto/Script/runCorrelation.R") # package
    does not contain this function.
runCorrelation(exprMat_filtered, scenicOptions)
```

4.3 GENIE3

This step is time consuming.

Run GENIE3:

```
# here you should run the following codes:
runGenie3(exprMat_filtered, scenicOptions)
```

5 Build and score the GRN

- Build the gene regulatory network:
  1. Get co-expression modules
  2. Get regulons (with RcisTarget): TF motif analysis)
- Identify cell states:
  3. Score GRN (regulons) in the cells (with AUCell)
  4. Cluster cells according to the GRN activity

Prepare setting:

```
scenicOptions@settings$verbose <- TRUE
scenicOptions@settings$seed <- 123 # this is important to have a
    reproducible result, because the algothritm is built on Ramdom Forest
    model.
scenicOptions@settings$nCores <- 10 # when run with 22 core, runSCENIC_2_
    createRegulons will overflow memory.
```

Run the remaining steps using the wrapper functions:

```
runSCENIC_1_coexNetwork2modules(scenicOptions)
runSCENIC_2_createRegulons(scenicOptions)
runSCENIC_3_scoreCells(scenicOptions, exprMat_filtered_log)
```

6 Demonstration of SCENIC results with Seurat

```
regulonAUC <- readRDS("int/3.4_regulonAUC.Rds")
```

Load regulon AUC score:

```
regulonAUC <- readRDS("../../../IM_DTR_exp2/analyses/20210505_SCENIC_only_
   IM_DIfferentiation/int/3.4_regulonAUC.Rds")
```

Extract the useful information:

```
regulonAUC.mat <- regulonAUC@assays@data@listData$AUC
dim(regulonAUC.mat)
```

```
## [1]   398 10566
```

Now take the top 50 most variable Regulons:

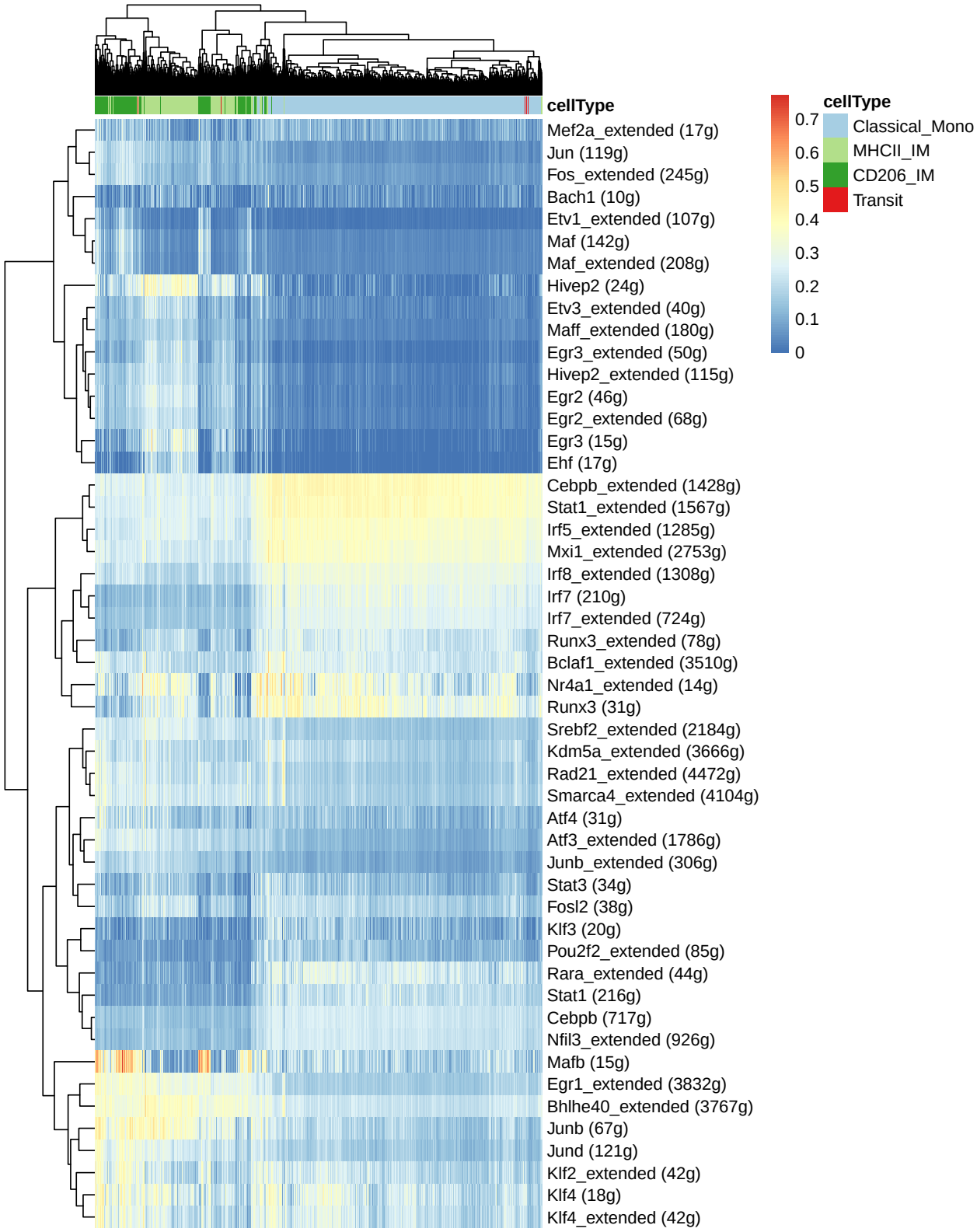
```
suppressMessages(library(genefilter))
rv <- rowVars(regulonAUC.mat)
idx <- order(-rv)[1:50]
```

```
sce <- readRDS("./data/IM-DTR3_3D.sce.Rds")
```

```
suppressMessages({
  library(pheatmap)
  library(RColorBrewer)
  library(SingleCellExperiment)
})

pal <- brewer.pal(length(levels(sce@colData$ident)), name = "Paired")

cellInfo.cellType <- data.frame(cellType = sce@colData$ident)
rownames(cellInfo.cellType) <- colnames(sce)
  annotationColors <- list(cellType = c(`Classical_Mono`="#A6CEE3",
                                        `MHCII_IM`="#B2DF8A",
                                        `CD206_IM`="#33A02C",
                                        `Transit` = "#E31A1C"))
pheatmap(regulonAUC.mat[idx,], show_colnames = FALSE, annotation_col =
   cellInfo.cellType, annotation_colors = annotationColors)
```




```
suppressMessages(library(ComplexHeatmap))
```

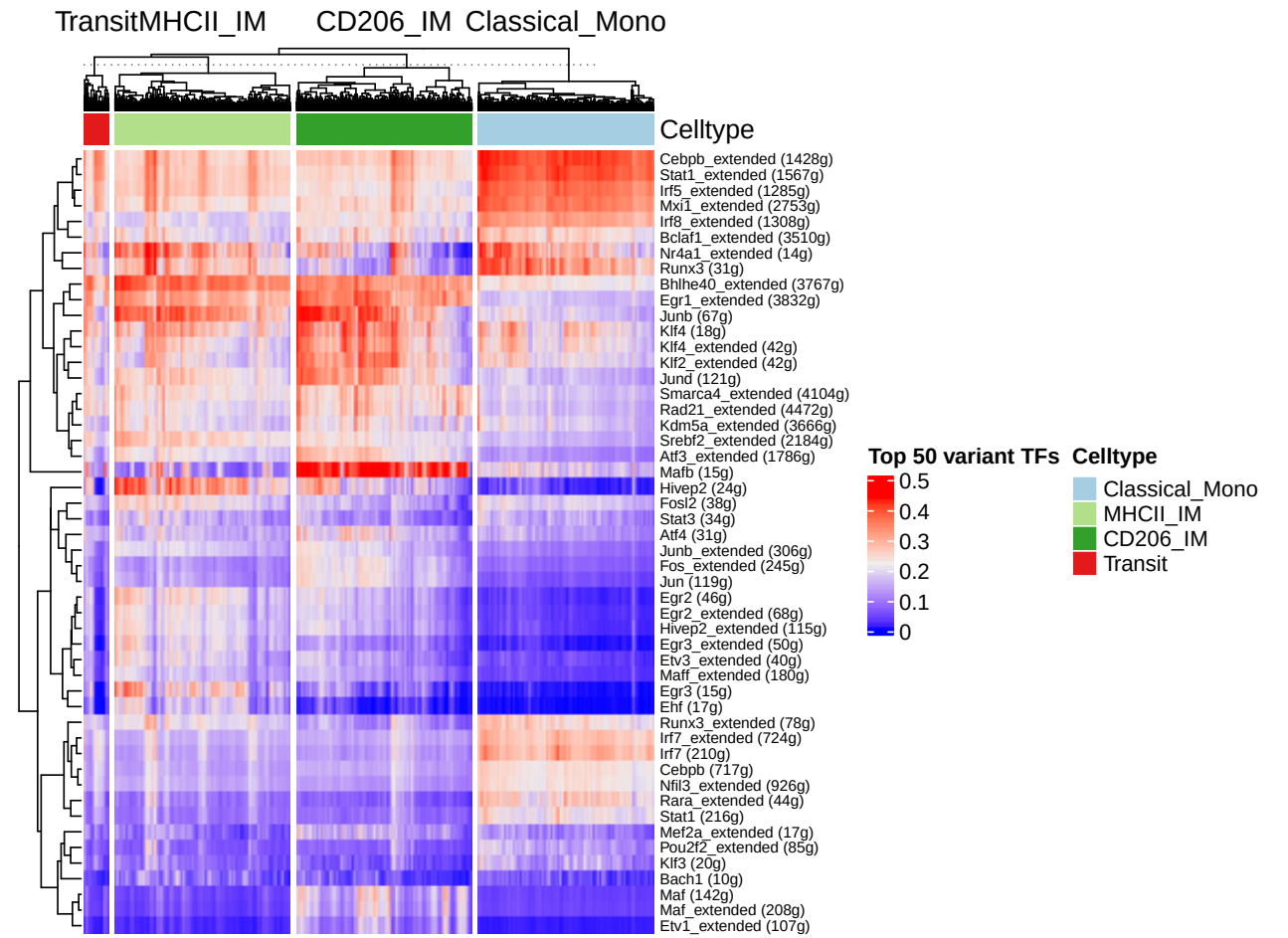
1

Randomly select 1500 cells in monocytes and IMs:

```
set.seed(41)
cellnames.15k <- unlist(sapply(unique(sce$cell.type2), function(x) {
  cellnames <- colnames(sce)[sce$cell.type2== x ]
  if (length(cellnames)>1500) {
    cellnames <- cellnames[sample(1:length(cellnames), 1500)]
  }
  return(cellnames)
} ))
```

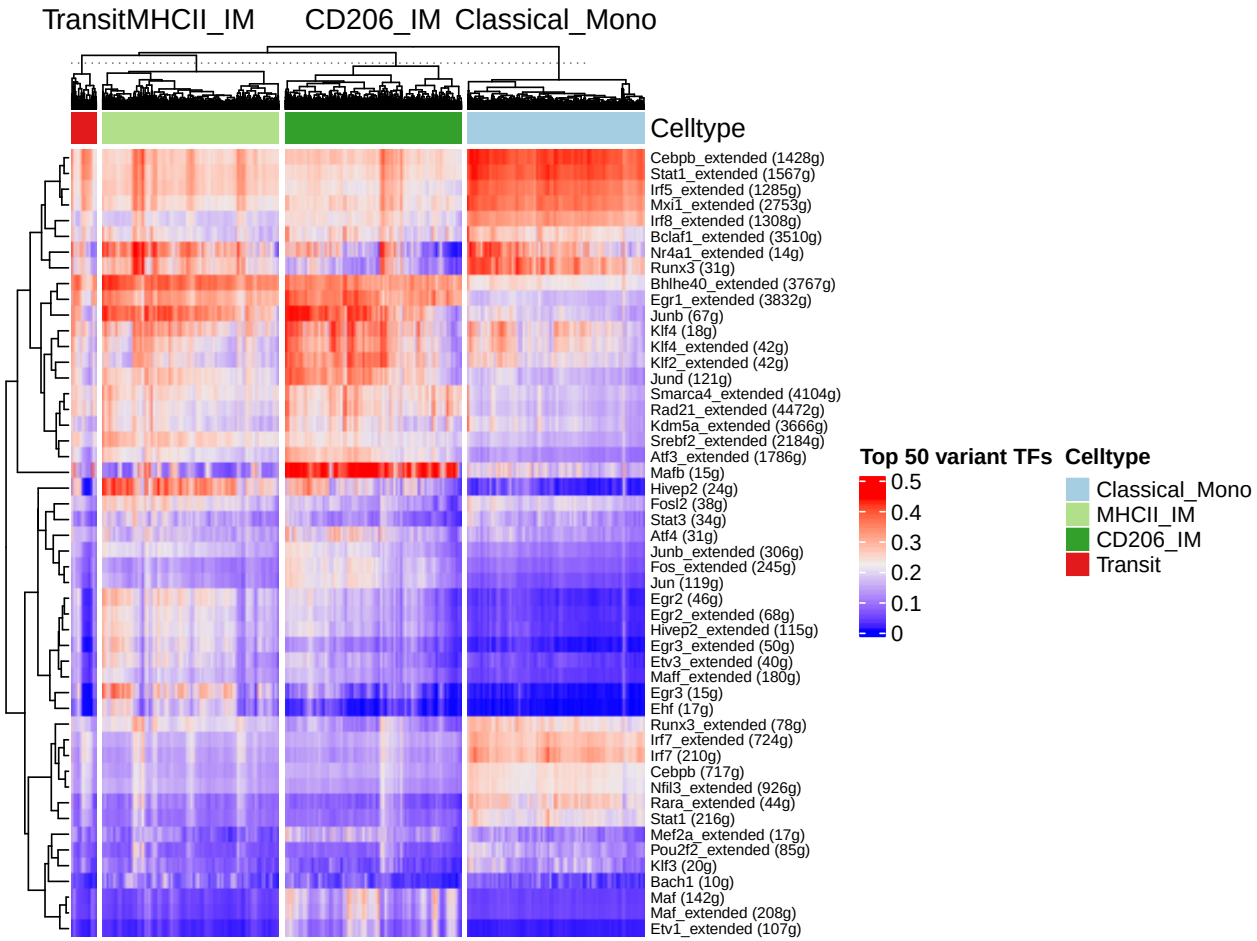
```
top50.15k.mat <- regulonAUC.mat[idx,cellnames.15k]
sce.15k <- sce[, cellnames.15k]
```

```
hp <- Heatmap(top50.15k.mat, name = "Top␣50␣variant␣TFs",
        show_row_names=TRUE, show_column_names = FALSE,
        row_names_gp = gpar ( fontsize = 7),
        column_split = factor(sce.15k$cell.type2),
        top_annotation = HeatmapAnnotation(Celltype=sce.15k$cell.type2,
                                           col = list(Celltype =
                                  c(`Classical_Mono`="#A6CEE3",
                                    `MHCII_IM`="#B2DF8A",
                                    `CD206_IM`="#33A02C",
                                    `Transit` = "#E31A1C")))
        )
hp <- draw(hp)
```


TransitMHCII_IM
CD206_IM
Classical_Mono
Celltype
Cebpb_extended (1428g)
Stat1_extended (1567g)
Irf5_extended (1285g)
Mxi1_extended (2753g)
Irf8_extended (1308g)
Bclaf1_extended (3510g)
Nr4a1_extended (14g)
Runx3 (31g)
Bhlhe40_extended (3767g)
Egr1_extended (3832g)
Junb (67g)
Klf4 (18g)
Klf4_extended (42g)
Klf2_extended (42g)
Jund (121g)
Smarca4_extended (4104g)
Rad21_extended (4472g)
Kdm5a_extended (3666g)
Srebf2_extended (2184g)
Atf3_extended (1786g)
Mafb (15g)
Hivep2 (24g)
Fosl2 (38g)
Stat3 (34g)
Atf4 (31g)
Junb_extended (306g)
Fos_extended (245g)
Jun (119g)
Egr2 (46g)
Egr2_extended (68g)
Hivep2_extended (115g)
Egr3_extended (50g)
Etv3_extended (40g)
Maff_extended (180g)
Egr3 (15g)
Ehf (17g)
Runx3_extended (78g)
Irf7_extended (724g)
Irf7 (210g)
Cebpb (717g)
Nfil3_extended (926g)
Rara_extended (44g)
Stat1 (216g)
Mef2a_extended (17g)
Pou2f2_extended (85g)
Klf3 (20g)
Bach1 (10g)
Maf (142g)
Maf_extended (208g)
Etv1_extended (107g)
Top 50 variant TFs
0.5
0.4
0.3
0.2
0.1
0
Celltype
Classical_Mono
MHCII_IM
CD206_IM
Transit

```
hp
```

1




```
pdf(file = "../Figures/HeatMap_top50_TF_SCENIC_in_15kcells.pdf",
    width = 8, height = 6)
hp
dev.off()
```

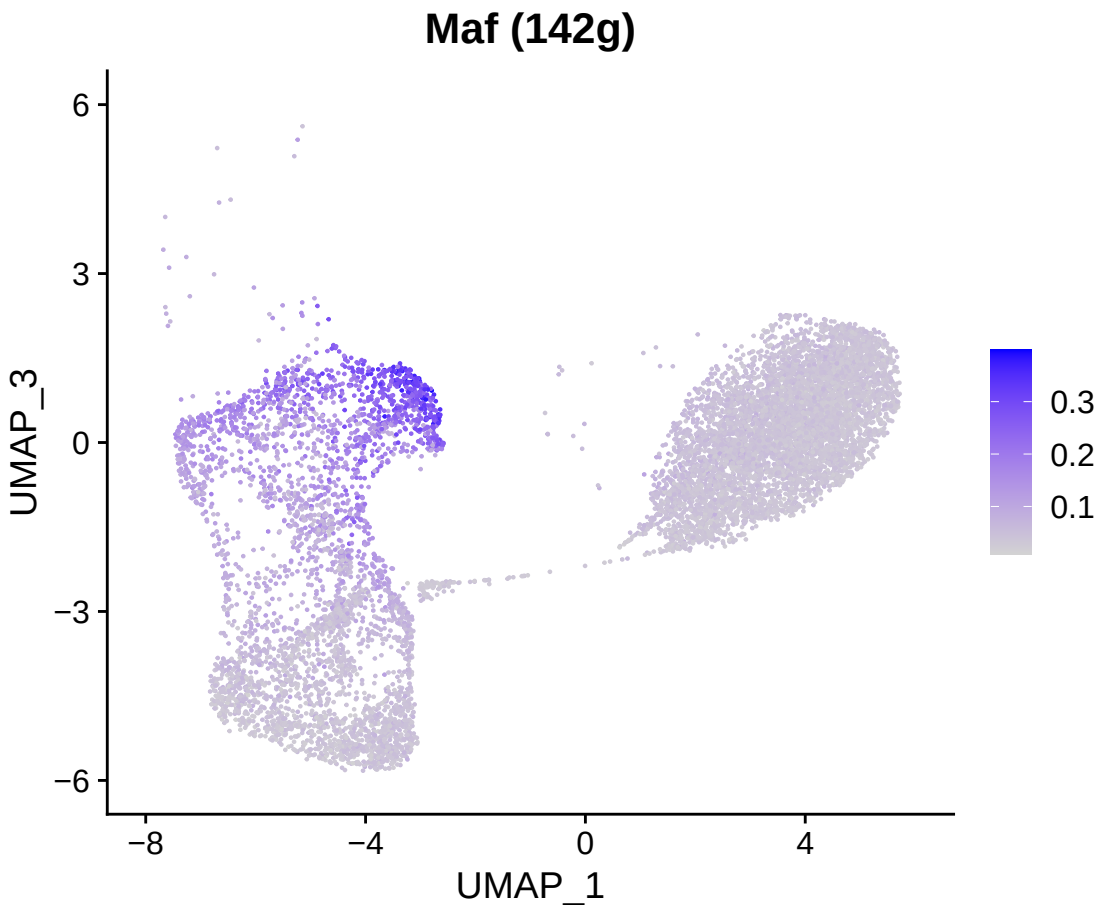
7 Make plots

Let's show the expression and activity of each TF in the pop:

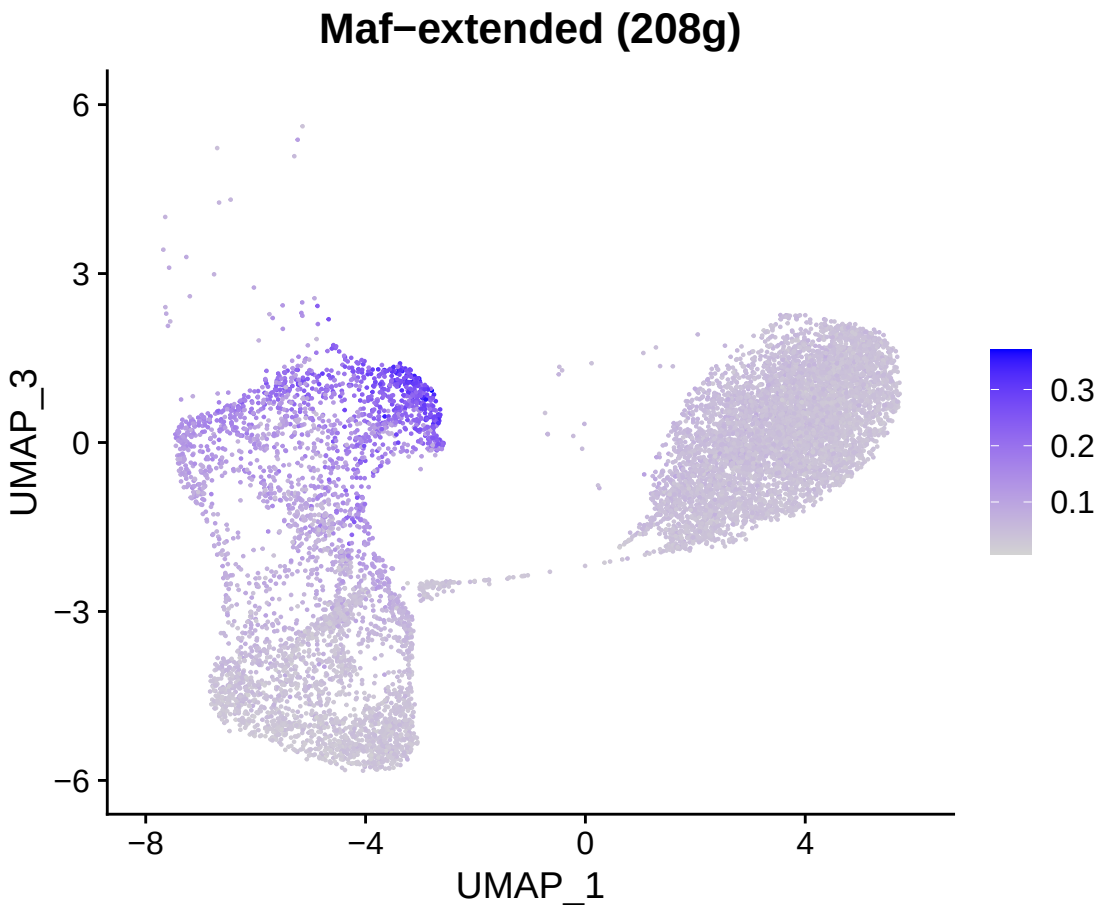
7.1 Show TF activity

```
obj[["scenic"]] <- CreateAssayObject(counts = regulonAUC.mat)
DefaultAssay(obj) <- "scenic"
```

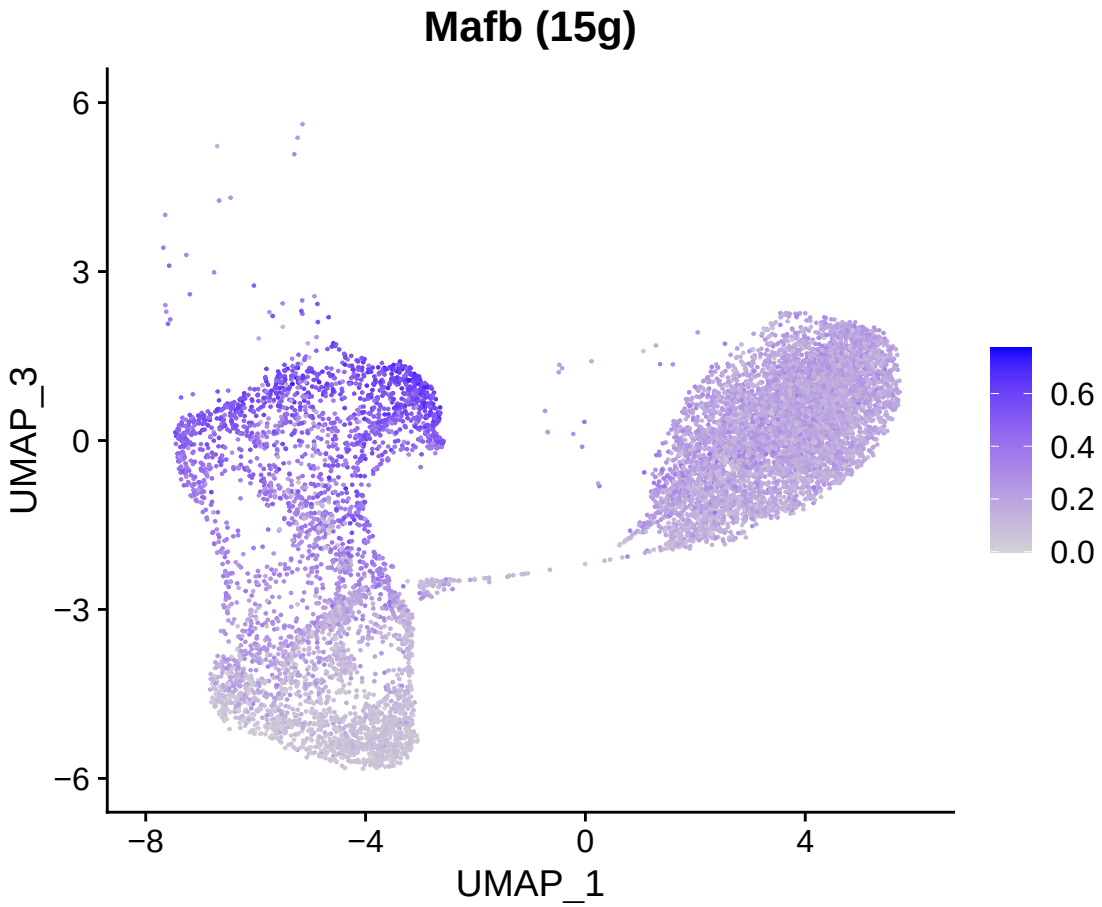
```
  FeaturePlot(obj, features = "Maf␣(142g)", dims = c(1,3))
```



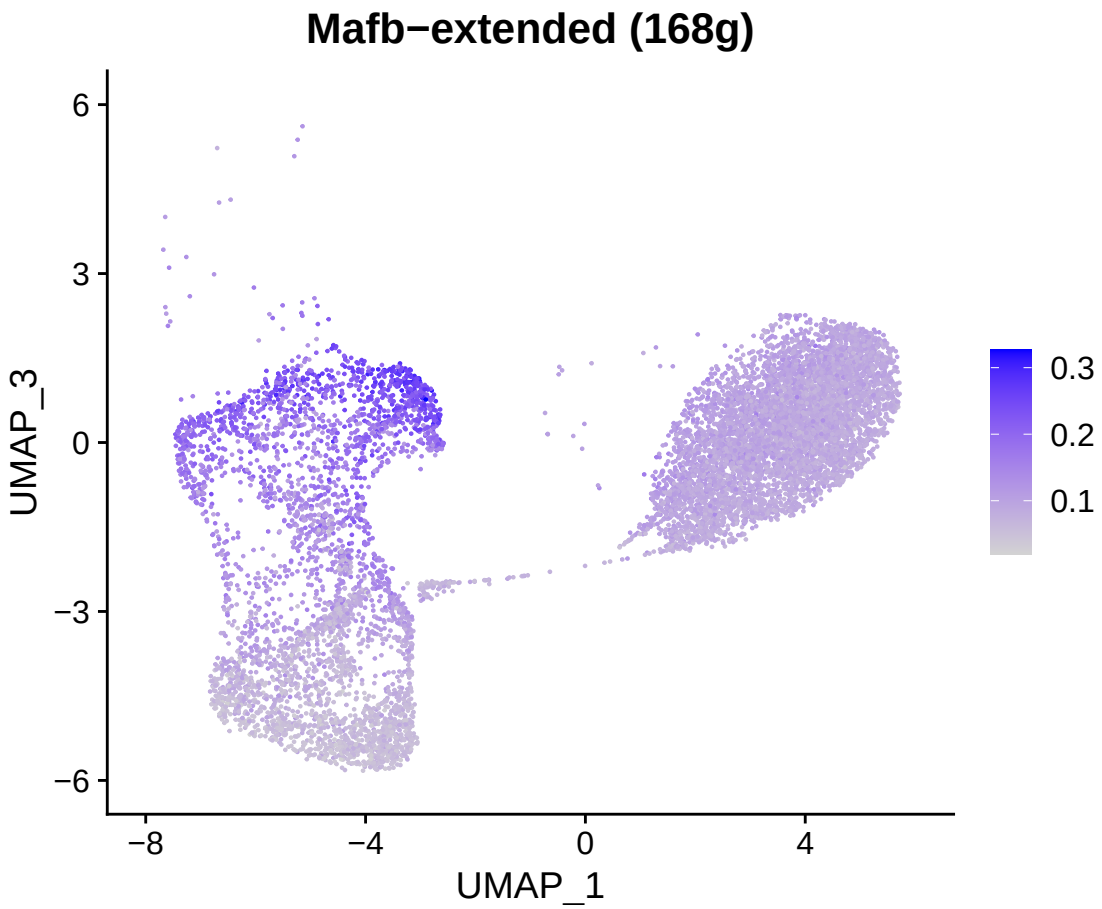

```
FeaturePlot(obj, features = "Maf-extended␣(208g)", dims = c(1,3))
```



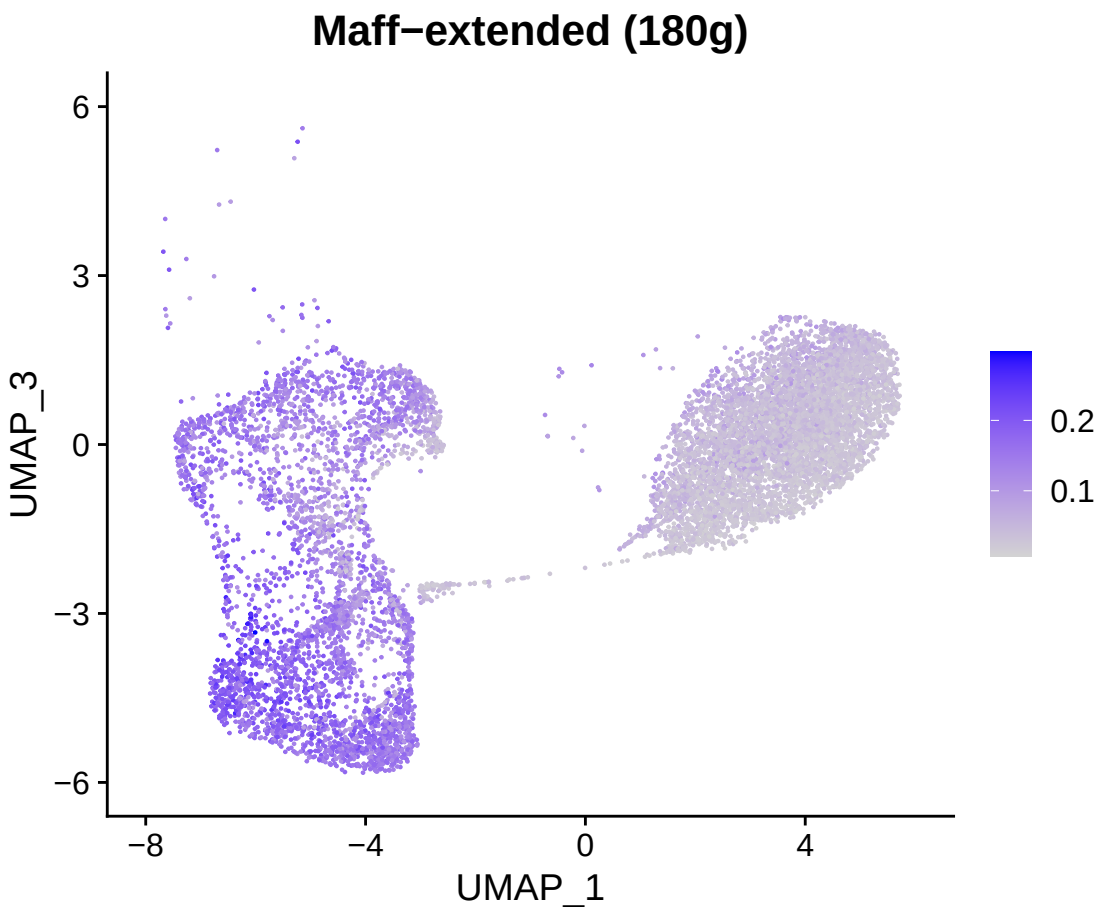

```
FeaturePlot(obj, features = "Mafb␣(15g)", dims = c(1,3))
```



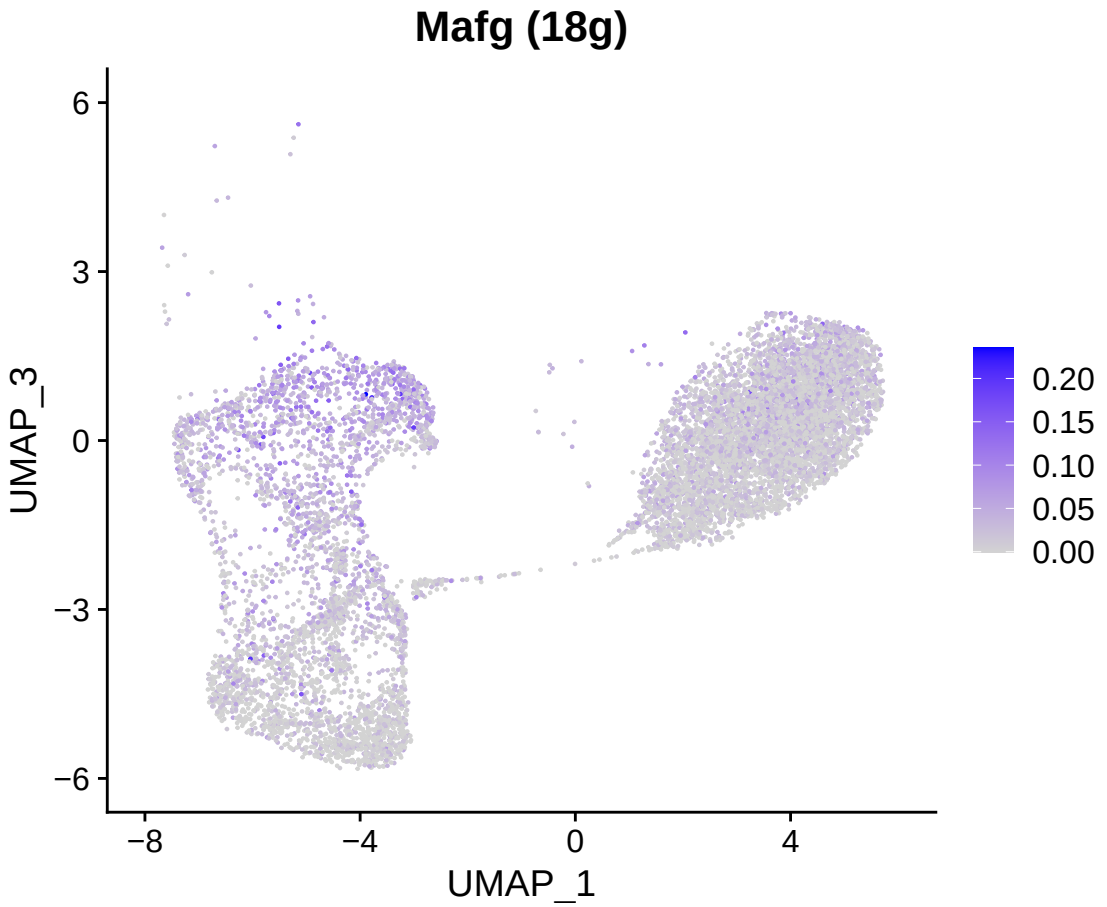

```
FeaturePlot(obj, features = "Mafb-extended (168g)", dims = c(1,3))
```



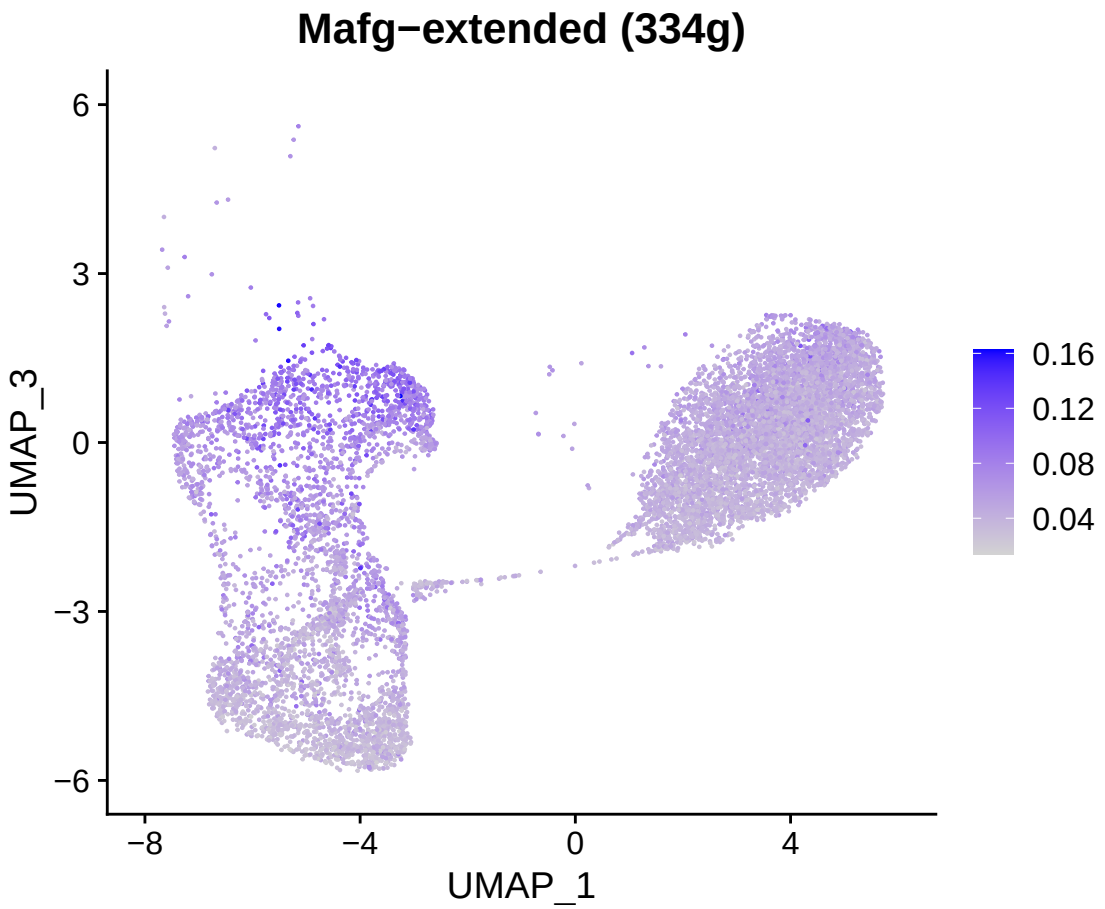

```
FeaturePlot(obj, features = "Maff-extended␣(180g)", dims = c(1,3))
```



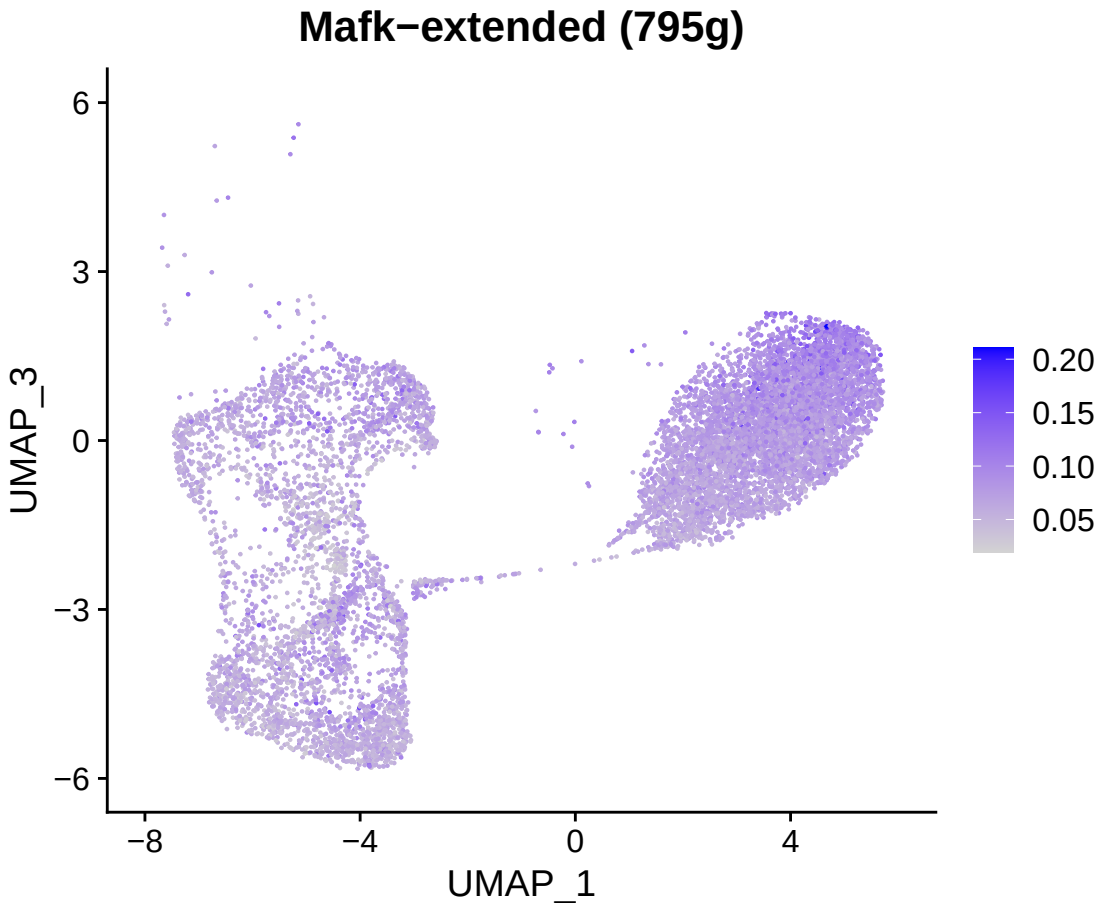

```
FeaturePlot(obj, features = "Mafg␣(18g)", dims = c(1,3))
```




```
FeaturePlot(obj, features = "Mafg-extended␣(334g)", dims = c(1,3))
```

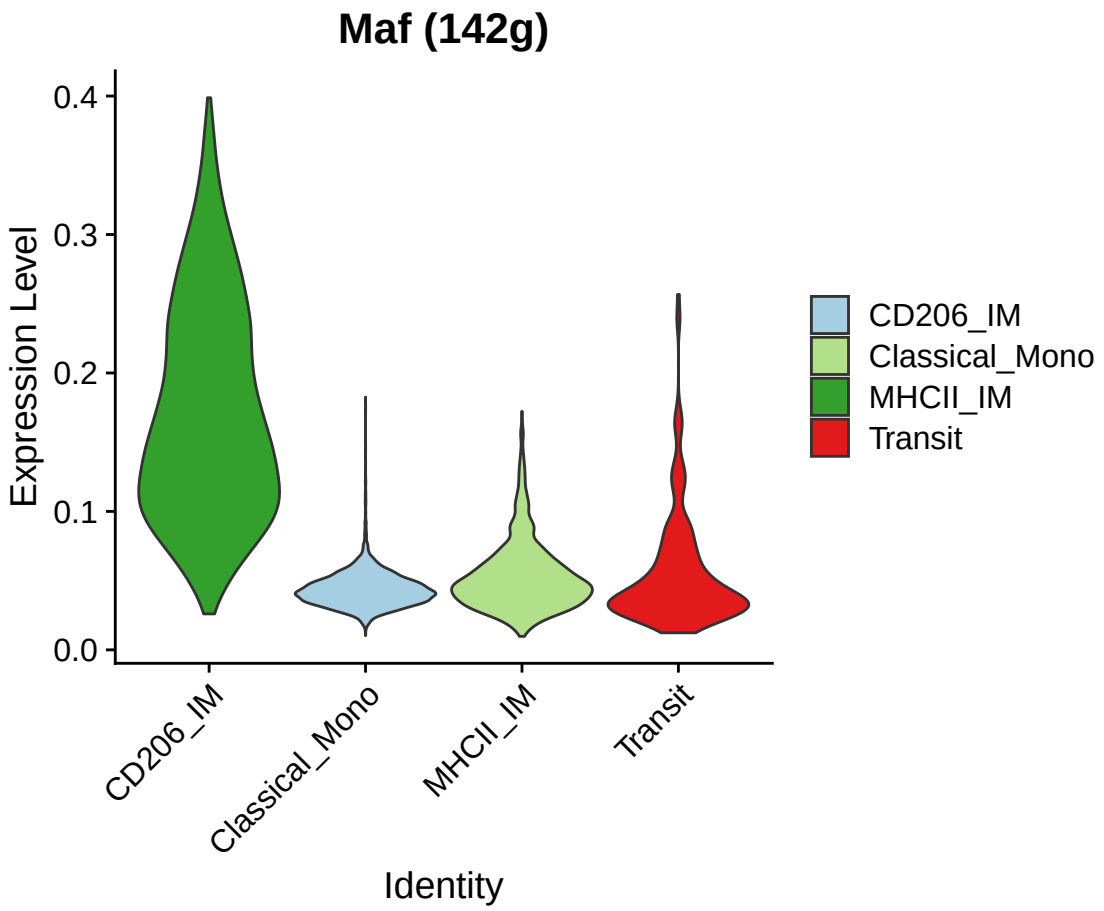



```
FeaturePlot(obj, features = "Mafk-extended␣(795g)", dims = c(1,3))
```

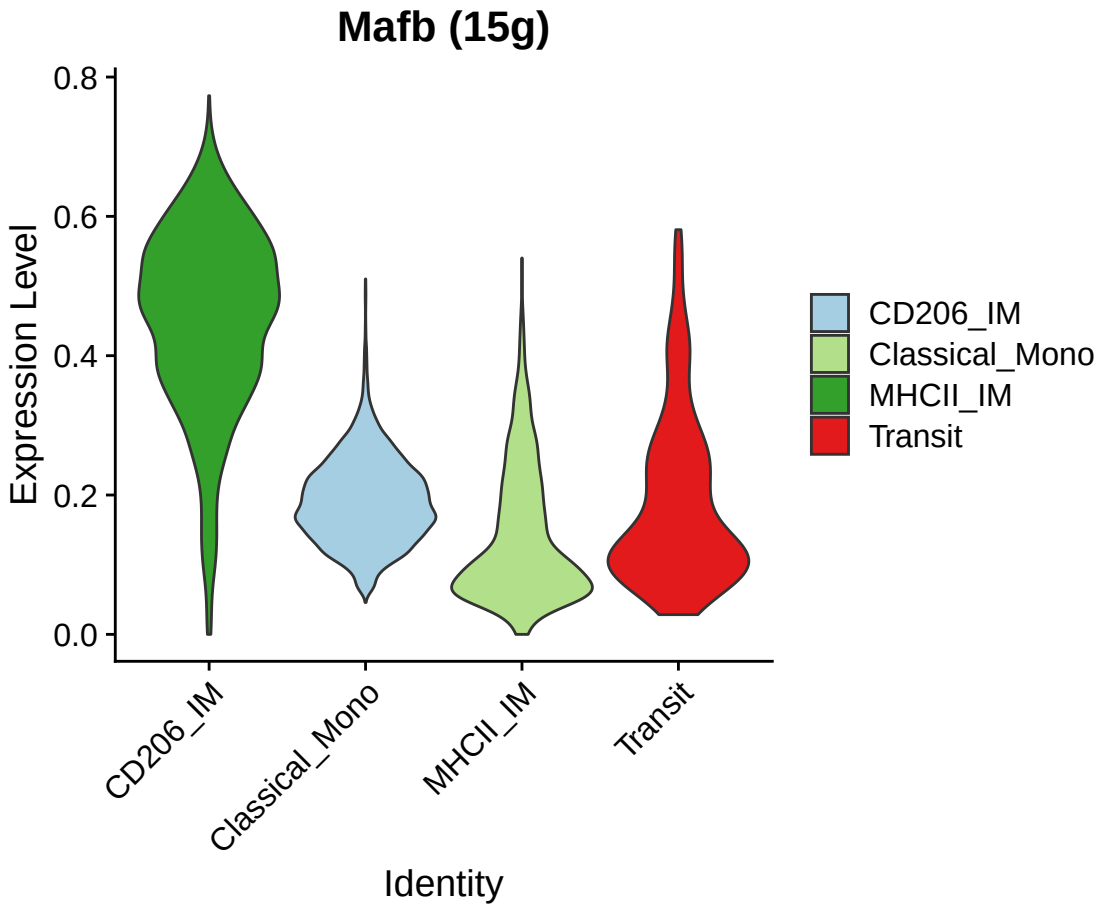



```
pal2 <- c(`Classical_Mono`="#A6CEE3",                                    1
                                        `MHCII_IM`="#B2DF8A",            2
                                        `CD206_IM`="#33A02C",            3
                                        `Transit` = "#E31A1C")           4
```

```
  VlnPlot(obj, features = "Maf␣(142g)", group.by  = "cell.type2", pt.size   1
     = 0, cols = pal2)
```

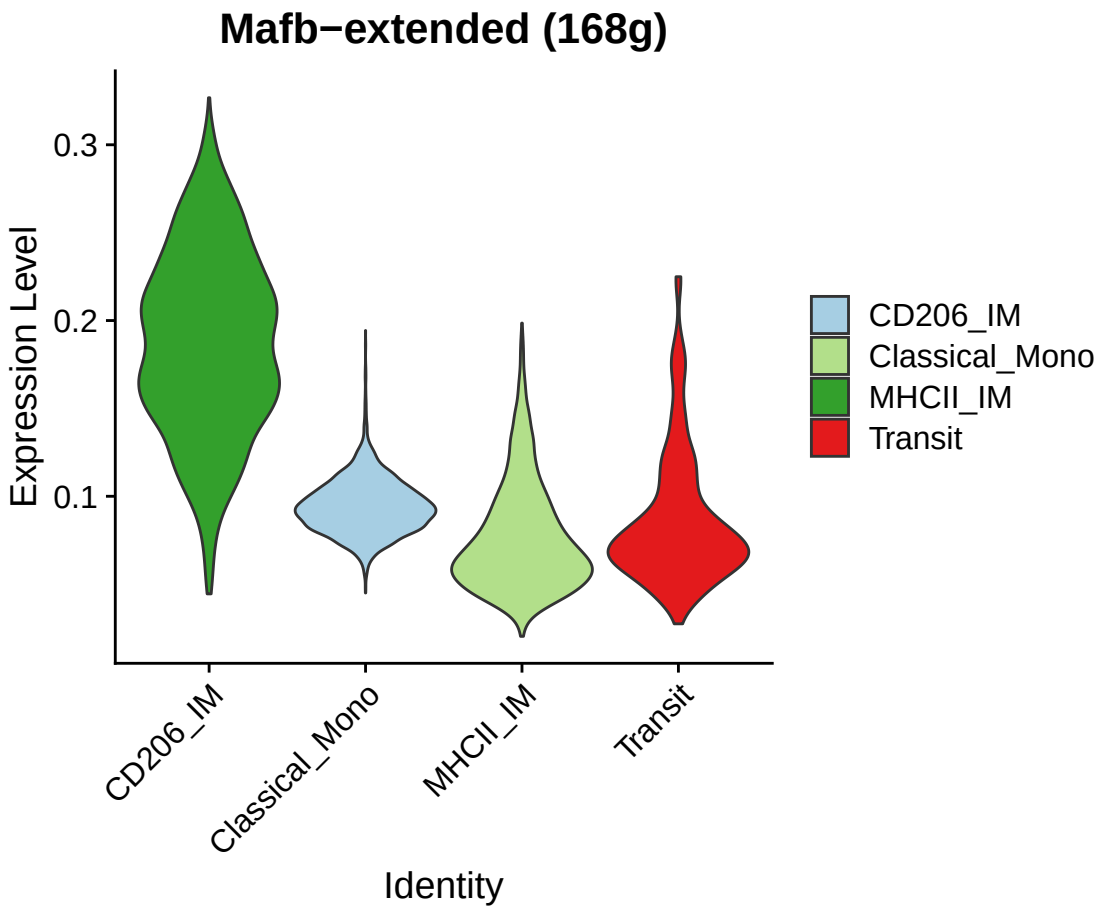



```
VlnPlot(obj, features = "Mafb␣(15g)", group.by  = "cell.type2", pt.size
   = 0, cols = pal2)
```




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

```
  VlnPlot(obj, features = "Mafb-extended␣(168g)", group.by  = "cell.type2"
     , pt.size = 0, cols = pal2)
```



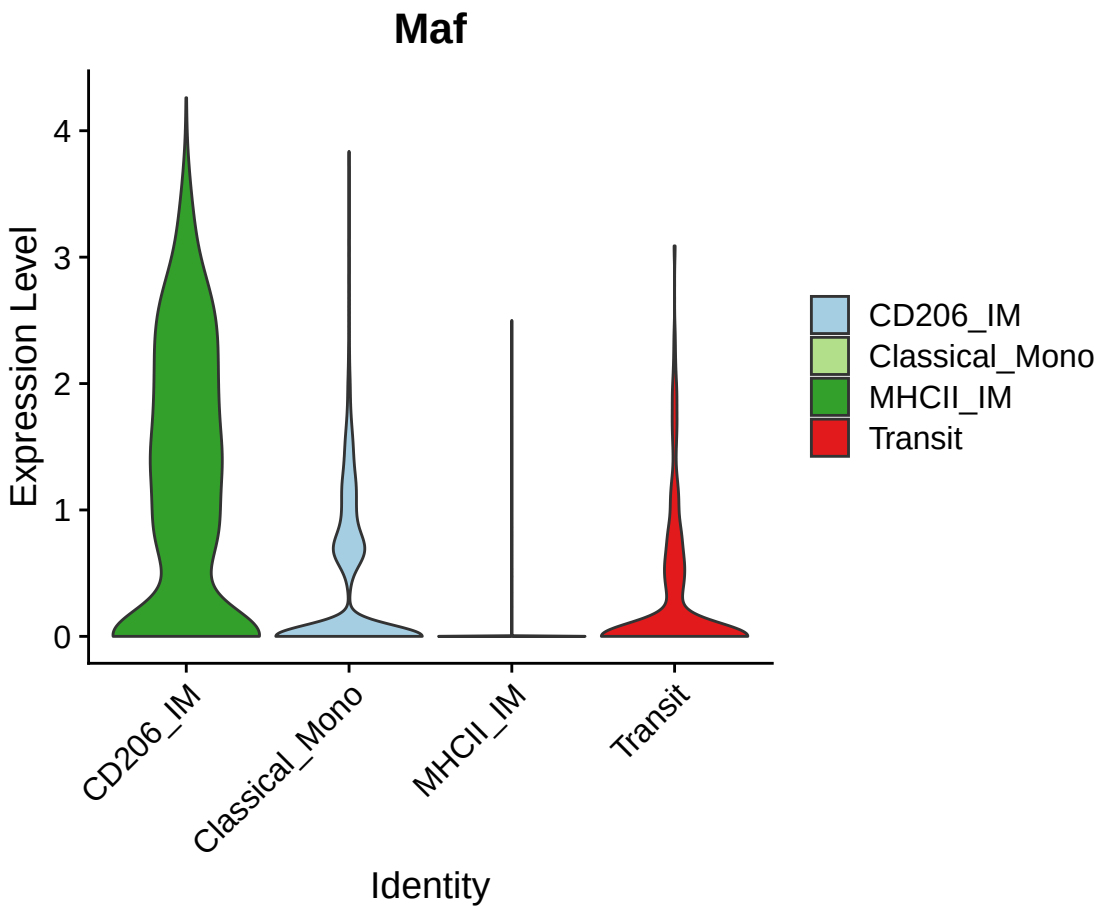
7.2 Show TF expression

```
DefaultAssay(obj) <- "RNA"
```

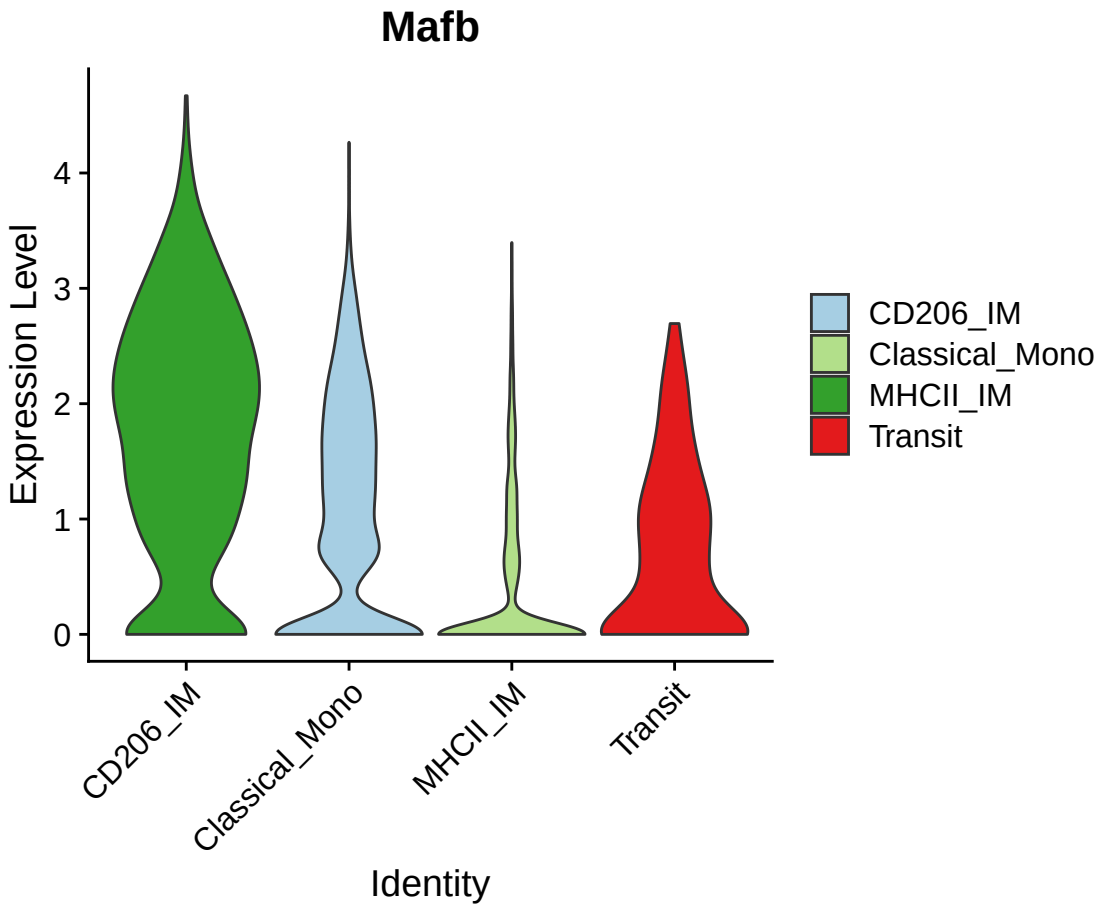
1

```
  VlnPlot(obj, features = "Maf", group.by  = "cell.type2", pt.size = 0,
     cols = pal2)
```

1




```
VlnPlot(obj, features = "Mafb", group.by  = "cell.type2", pt.size = 0,
    cols = pal2)
```




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

Save data

```
saveRDS(obj, file = "./only_IM_differentiation.with_SCENIC.seuratObject.
   Rds")
```

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                                 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] grid      parallel  stats4    stats     graphics  grDevices utils      18
##  [8] datasets  methods   base                                               19
##                                                                              20
## other attached packages:                                                     21
##  [1] ComplexHeatmap_2.6.2        SingleCellExperiment_1.12.0                 22
##  [3] SummarizedExperiment_1.20.0 Biobase_2.50.0                              23
##  [5] GenomicRanges_1.42.0        GenomeInfoDb_1.26.7                         24
##  [7] IRanges_2.24.1              S4Vectors_0.28.1                            25
##  [9] BiocGenerics_0.36.1         MatrixGenerics_1.2.1                        26
## [11] matrixStats_0.61.0          RColorBrewer_1.1-2                          27
## [13] pheatmap_1.0.12             genefilter_1.72.1                           28
## [15] ggplot2_3.3.5               SCopeLoomR_0.10.4                           29
## [17] RcisTarget_1.10.0           AUCell_1.13.3                               30
## [19] SCENIC_1.2.4                SeuratObject_4.0.4                          31
## [21] Seurat_4.1.0                                                            32
##                                                                              33
## loaded via a namespace (and not attached):                                   34
##   [1] circlize_0.4.14        plyr_1.8.6             igraph_1.2.11          35
##   [4] lazyeval_0.2.2         GSEABase_1.52.1        splines_4.0.3          36
##   [7] listenv_0.8.0          feather_0.3.5          scattermore_0.8        37
##  [10] digest_0.6.29          htmltools_0.5.2        magick_2.7.3           38
##  [13] fansi_1.0.2            magrittr_2.0.2         memoise_2.0.1          39
##  [16] tensor_1.5             cluster_2.1.0          ROCR_1.0-11            40
##  [19] globals_0.14.0         annotate_1.68.0        R.utils_2.11.0         41
##  [22] spatstat.sparse_2.1-0  colorspace_2.0-3       blob_1.2.2             42
##  [25] ggrepel_0.9.1          xfun_0.29              dplyr_1.0.8            43
##  [28] crayon_1.5.0           RCurl_1.98-1.6         jsonlite_1.7.3         44
##  [31] graph_1.68.0           spatstat.data_2.1-2    survival_3.2-7         45
##  [34] zoo_1.8-9              glue_1.6.1             polyclip_1.10-0        46
##  [37] gtable_0.3.0           zlibbioc_1.36.0        XVector_0.30.0         47
##  [40] leiden_0.3.9           GetoptLong_1.0.5       DelayedArray_0.16.3    48
##  [43] shape_1.4.6            future.apply_1.8.1     abind_1.4-5            49
##  [46] scales_1.1.1           DBI_1.1.2              spatstat.random_2      50
    .1-0
##  [49] miniUI_0.1.1.1         Rcpp_1.0.8             viridisLite_0.4.0      51
##  [52] xtable_1.8-4           clue_0.3-60            reticulate_1.24        52
##  [55] spatstat.core_2.4-0    bit_4.0.4              htmlwidgets_1.5.4      53
##  [58] httr_1.4.2             ellipsis_0.3.2         ica_1.0-2              54
##  [61] farver_2.1.0           pkgconfig_2.0.3        XML_3.99-0.8           55
##  [64] R.methodsS3_1.8.1      uwot_0.1.11            deldir_1.0-6           56
##  [67] utf8_1.2.2             labeling_0.4.2         tidyselect_1.1.1       57
##  [70] rlang_1.0.1            reshape2_1.4.4         later_1.3.0            58
##  [73] AnnotationDbi_1.52.0   munsell_0.5.0          tools_4.0.3            59
##  [76] cachem_1.0.6           cli_3.2.0              generics_0.1.2         60
##  [79] RSQLite_2.2.10         ggridges_0.5.3         evaluate_0.15          61
##  [82] stringr_1.4.0          fastmap_1.1.0          yaml_2.3.5             62
```

```
##  [85] goftest_1.2-3          knitr_1.37             bit64_4.0.5
##  [88] fitdistrplus_1.1-6     purrr_0.3.4            RANN_2.6.1
##  [91] pbapply_1.5-0          future_1.24.0          nlme_3.1-155
##  [94] mime_0.12              ggrastr_1.0.1          R.oo_1.24.0
##  [97] hdf5r_1.3.5            compiler_4.0.3         rstudioapi_0.13
## [100] beeswarm_0.4.0         plotly_4.10.0          png_0.1-7
## [103] spatstat.utils_2.3-0   tibble_3.1.6           stringi_1.7.6
## [106] highr_0.9              lattice_0.20-41        Matrix_1.4-0
## [109] vctrs_0.3.8            pillar_1.7.0           lifecycle_1.0.1
## [112] GlobalOptions_0.1.2    spatstat.geom_2.3-2    lmtest_0.9-39
## [115] RcppAnnoy_0.0.19       data.table_1.14.2      cowplot_1.1.1
## [118] bitops_1.0-7           irlba_2.3.5            httpuv_1.6.5
## [121] patchwork_1.1.1        R6_2.5.1               promises_1.2.0.1
## [124] KernSmooth_2.23-20     gridExtra_2.3          vipor_0.4.5
## [127] parallelly_1.30.0      codetools_0.2-18       MASS_7.3-53
## [130] assertthat_0.2.1       rjson_0.2.21           withr_2.4.3
## [133] sctransform_0.3.3      GenomeInfoDbData_1.2.4 hms_1.1.1
## [136] mgcv_1.8-33            rpart_4.1-15           tidyr_1.2.0
## [139] rmarkdown_2.11         Cairo_1.5-14           Rtsne_0.15
## [142] shiny_1.7.1            ggbeeswarm_0.6.0
```

References

1. Aibar, S. *et al.* SCENIC: Single-cell regulatory network inference and clustering. *Nature Methods 2017 14:11* **14**, 1083–1086 (2017).