PAPER TITLE TO BE DEFINED (in common.yaml)

10-DE genes across pseudotime

2021-12-08 16:24:59 +0100

Abstract

Lung interstitium macrophages (IMs) are non-alveolar resident tissue macrophages which contribute to the lung homeostasis. These cells were reported to be heterogeneous by our group and other teams, which contains two main distinct subpopulations: CD206+ IMs and CD206- IMs. However, the exact origin of IMs and the transcriptional programs that control IM differentiation remains unclear. In recent report, we analyzed the refilled IMs in the course of time after induced IM depletion with single-cell RNA sequencing (10X Genomics Chromium) and bulk RNA sequencing.

Contents

1 Description

2 Prepare data

```
suppressMessages(
{
library(Seurat)
library(ComplexHeatmap)
library(ggplot2)
library(dplyr)
library(RColorBrewer)
library(circlize)
library(monocle3)
})

mo <- readRDS(file = "../9-Monocle_analysis_and_pseudotime_estimation/Mono
   _to_IM.cds")
```

3 DE gene expression across IM-differentiation

DE genes across pseudotime of IM differentiation ## Across pseudotime of IM differentiation

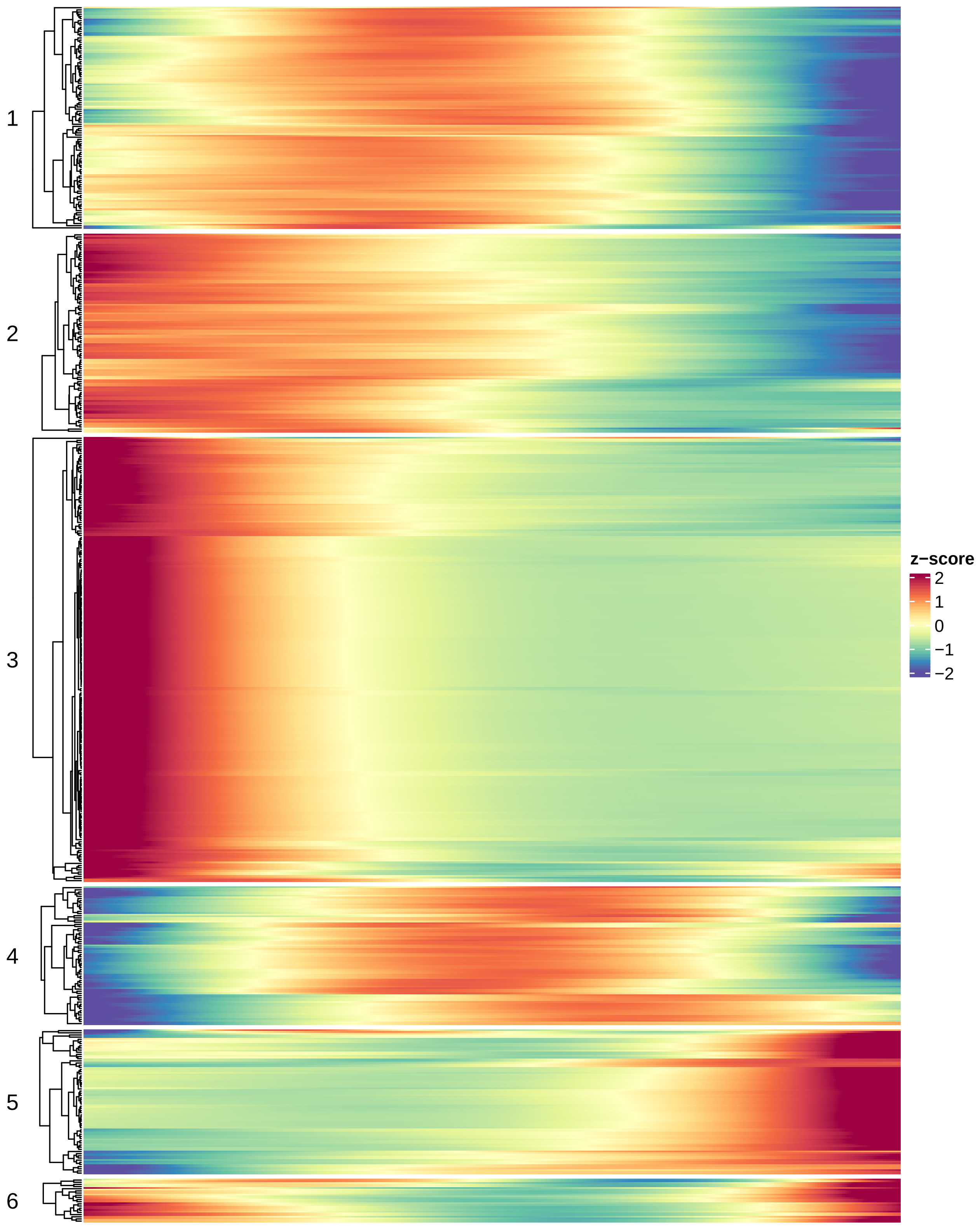
Prepare matrix with z-scores, smoothened and scaled data across pseudotime for heatmap.

```
pt.matrix <- exprs(mo)[match(genes,rownames(rowData(mo))),order(pseudotime
   (mo))]
cellnames <- colnames(pt.matrix)
#Can also use "normalized_counts" instead of "exprs" to use various
   normalization methods, for example:
#normalized_counts(cds, norm_method = "log")

pt.matrix <- t(apply(pt.matrix,1,function(x){smooth.spline(x,df=3)$y}))
pt.matrix <- t(apply(pt.matrix,1,function(x){(x-mean(x))/sd(x)}))
rownames(pt.matrix) <- genes
colnames(pt.matrix) <- cellnames
```

Show DE genes in unsupervised heatmap.

```
#K means with 6 groups
htkm <- Heatmap(
  pt.matrix,
  # use_raster = FALSE, # use FALSE to export to vector image.
  name                         = "z-score",
  col                          = colorRamp2(seq(from=-2,to=2,length=11),
     rev(brewer.pal(11, "Spectral"))),
  show_row_names               = FALSE,
  show_column_names            = FALSE,
  row_names_gp                 = gpar(fontsize = 3),
  row_km = 6,
  row_km_repeats = 31,
  row_dend_reorder = TRUE,
  row_title_rot                = 0,
  cluster_rows                 = TRUE,
  cluster_row_slices           = FALSE,
```

```r
  cluster_columns                  = FALSE,
)

htkm <- draw(htkm)
```



In this heatmap, the x axis is pseudotime, which represents differentiation state from monocytes (left) to IMs (right).

3.1 Annotate the cells associated to either differentiation of CD206+ IMs or CD206- IMs

```r
library(magrittr)
# Get the closest vertice for every cell
y_to_cells <-  mo@principal_graph_aux$UMAP$pr_graph_cell_proj_closest_
   vertex%>%as.data.frame()

y_to_cells$cells <- rownames(y_to_cells)
y_to_cells$Y <- y_to_cells$V1


# Get the root vertices
# It is the same node as above
root <- mo@principal_graph_aux$UMAP$root_pr_nodes

principalgraph <- mo@principal_graph$UMAP

# Get the other endpoints
endpoints <- names(which(igraph::degree(principalgraph ) == 1))
endpoints <- endpoints[!endpoints %in% root]

# For each endpoint
cellWeights <- lapply(endpoints, function(endpoint) {
  # We find the path between the endpoint and the root
  path <- igraph::shortest_paths(principalgraph, root, endpoint)$vpath
     [[1]]
  path <- as.character(path)
  # We find the cells that map along that path
  df <- y_to_cells[y_to_cells$Y %in% path, ]
  df <- data.frame(weights = as.numeric(colnames(mo) %in% df$cells))
  colnames(df) <- endpoint
  return(df)
  }) %>% do.call(what = 'cbind', args = .) %>%
    as.matrix()
rownames(cellWeights) <- colnames(mo)
colnames(cellWeights) <- c("CD206_IM␣branch", "MHCII_IM␣branch")
pseudotime <- matrix(mo@principal_graph_aux$UMAP$pseudotime, ncol = ncol(
   cellWeights),
                     nrow = ncol(mo), byrow = FALSE)
rownames(pseudotime) <- colnames(mo)
```

4 TradeSeq analysis for the differentiation of monocytes to either of IM subsets

4.1 Construct sce object for TradeSeq

```r
suppressMessages(library(tradeSeq))
```

```r
# this step is VERY time consuming
sce <- fitGAM(counts = mo@assays@data$counts,
              pseudotime = pseudotime,
              cellWeights = cellWeights)
```

```

saveRDS(sce, file = "./sce.4339cells.newversion.Rds")
```

Between-lineage comparisons (CD206+ IM vs CD206- IM differentiation) ## Between-lineage comparisons (CD206+ IM vs CD206- IM differentiation)

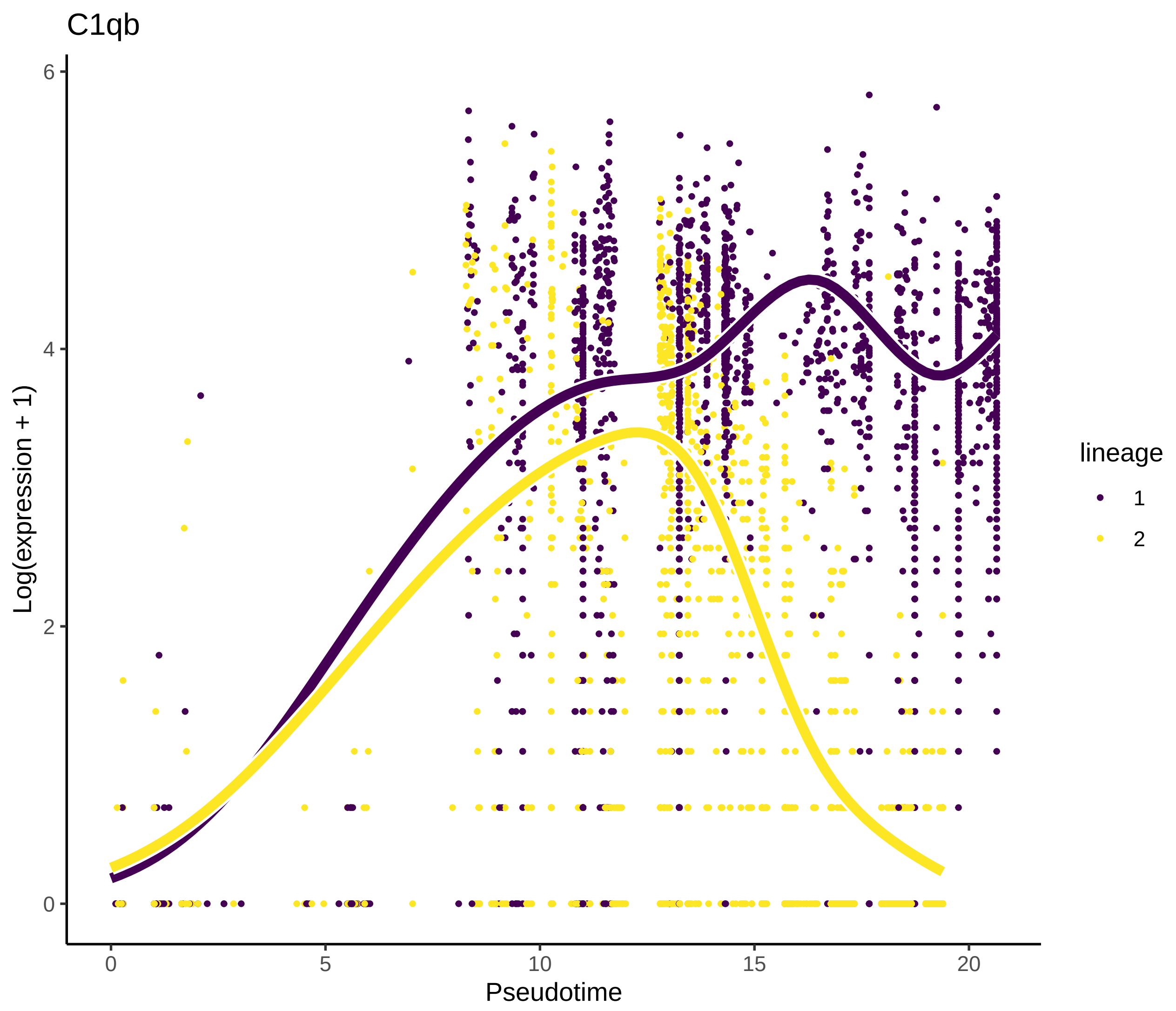
Association of gene expression with pseudotime (find significant DE genes along pseudotime).

```
assoRes <- associationTest(sce)
endRes <- diffEndTest(sce)
head(assoRes)
```

```
## # A tibble: 6 x 4
##   waldStat    df      pvalue meanLogFC
##      <dbl> <dbl>       <dbl>     <dbl>
## 1    210.      9  0              0.222
## 2     28.4     9  0.000815       0.113
## 3     NA      NA NA              0.121
## 4     41.8     9  0.00000360     0.0958
## 5     36.4     9  0.0000330      0.163
## 6     45.1     9  0.000000880    0.160
```

Plot the most sig gene:

```
library(ggplot2)
o <- order(endRes$waldStat, decreasing = TRUE)
sigGene <- names(sce)[o[1]]
plotSmoothers(sce, counts = counts(sce), gene = sigGene
              #, curvesCol = c("#33A02C", "#B2DF8A")
              ) + ggtitle(sigGene)
```



```
#+ scale_color_manual(
#   name = "Lineage",

   labels = c("CD206_IM branch", "MHCII_IM branch"),
#   values = c("#33A02C", "#B2DF8A"))
```

What's the top genes?

```
names(sce)[o[1:20]]
```

```
##  [1] "C1qb"     "Ctsb"     "C1qa"     "Selenop" "Csf1r"    "Timp2"    "Pf4"
##  [8] "C1qc"     "Serinc3" "Cd209a"  "Lsp1"     "Lgmn"     "Apoe"     "Blvrb
   "
## [15] "Olfm1"    "Tnip3"    "Rpl13"    "Ninj1"    "Rpl28"    "H2-DMb1"
```

4.2 Clustering using RSEC, clusterExperiment

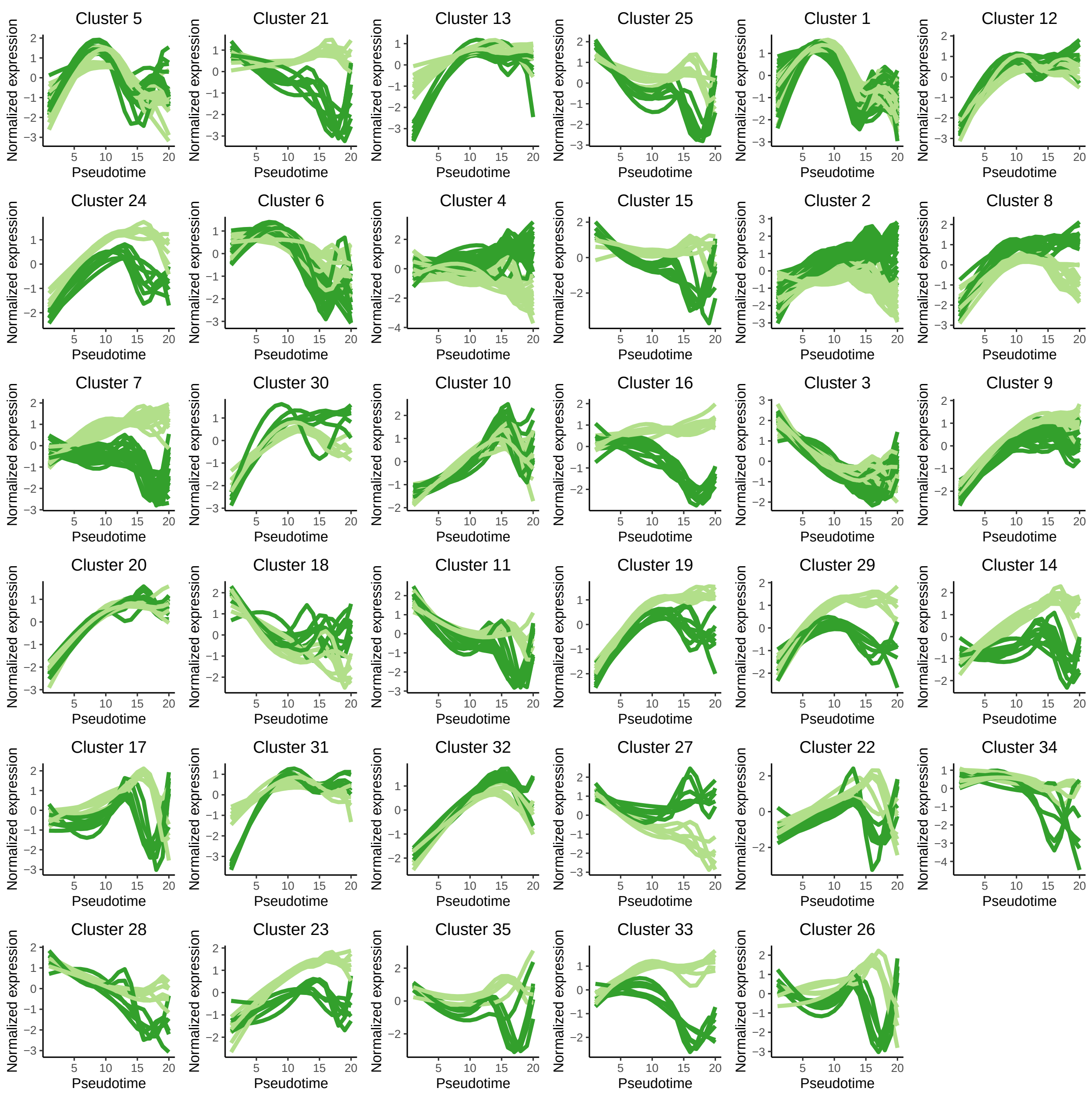
tradeSeq provides the functionality to cluster genes according to their expression pattern along the lineages with the clusterExpressionPatterns function. A number of equally spaced points for every lineage are selected to perform the clustering, and the number of points can be selected with the nPoints argument. (from `vignette("tradeSeq")`)

```
library(clusterExperiment)
nPointsClus <- 20 # The number of points to use for clustering the
   expression patterns..
clusPat <- clusterExpressionPatterns(sce,
                                     nPoints = nPointsClus,
                                     genes = genes,
                                     random.seed = 43,
                                     beta = 0.2
                                     )
```

```
## 36 parameter combinations, 36 use sequential method, 36 use subsampling
    method
## Running Clustering on Parameter Combinations...
## done.
```

```
clusterLabels <- primaryCluster(clusPat$rsec)
```

```
cUniq <- unique(clusterLabels) #
cUniq <- cUniq[!cUniq == -1] # remove unclustered genes

# cUniq <- cUniq[cUniq == -1]
#Any samples not found as part of a homogenous set of clusters at that
   point will be classified as unclustered (given a value of -1)

# beta: value between 0 and 1 to decide how stable clustership membership
   has to be before 'finding' and removing the cluster.
if (exists("p.total")) { rm(p.total)}

for (xx in cUniq) {
  cId <- which(clusterLabels == xx)
  p <- ggplot(data = data.frame(x = 1:nPointsClus,
                                y = rep(range(clusPat$yhatScaled[cId, ]),
                                        nPointsClus / 2)),
              aes(x = x, y = y)) +
    geom_point(alpha = 0) +
    labs(title = paste0("Cluster␣", xx),  x = "Pseudotime", y = "
       Normalized␣expression") +
    theme_classic() +
    theme(plot.title = element_text(hjust = 0.5))
  for (ii in 1:length(cId)) {
    geneId <- rownames(clusPat$yhatScaled)[cId[ii]]
    p <- p +
      geom_line(data = data.frame(x = rep(1:nPointsClus, 2),
                                  y = clusPat$yhatScaled[geneId, ],
                                  lineage = rep(0:1, each = nPointsClus)),
                aes(col = as.character(lineage), group = lineage), lwd =
                   1.5)
  }
  p <- p + guides(color = "none") +
    scale_color_manual(values = c("#33A02C", "#B2DF8A"),
                       breaks = c("0", "1"))
  if (exists("p.total")) { p.total <- p.total + p} else {p.total <- p}
}
  print(p.total)
```



5 Show gene expression pattern calculated by TradeSeq in heatmap

5.1 Data preparation

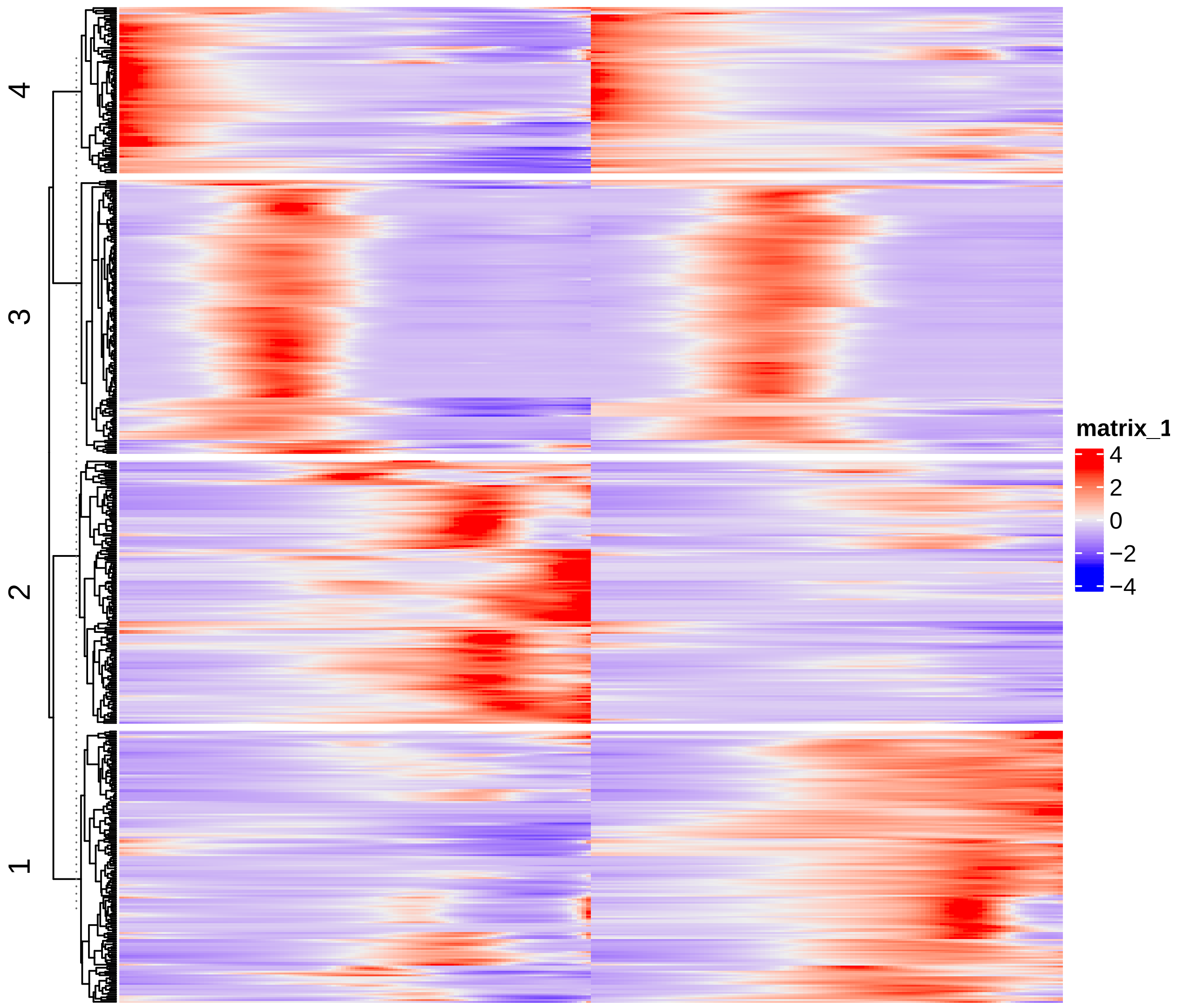
Here we use the DE genes calculated in DE genes across pseudotime of IM differentiation.

```
yhatSmooth <- predictSmooth(sce, gene = genes, nPoints = 100, tidy = FALSE
   )
```

```
yhatSmoothScaled <- t(scale(t(yhatSmooth)))
```

5.2 Draw heatmap

```
heatSmooth <- Heatmap(yhatSmoothScaled, cluster_columns = FALSE, show_row_
   names = FALSE, show_column_names = FALSE, row_km = 4)
heatSmooth <- draw(heatSmooth)
```



Two IM differentiation show similar patterns but some genes (especially cluster 2 and 1) are different in CD206+ and CD206-.

5.3 Annotate DE genes as CD206+/CD206- IM differentiation specific or common genes

According to the heatmap above, some of DE genes should remain unchanged (common) and half of them are specific to one of two IM differentiation.

We use wald statistic calculated in diffEndTest to annotate the "common" genes and "specific" genes. (in Between-lineage comparisons (CD206+ IM vs CD206- IM differentiation))

```r
endRes.DE <- endRes[rownames(yhatSmooth), ]
summary(endRes.DE$waldStat)
```

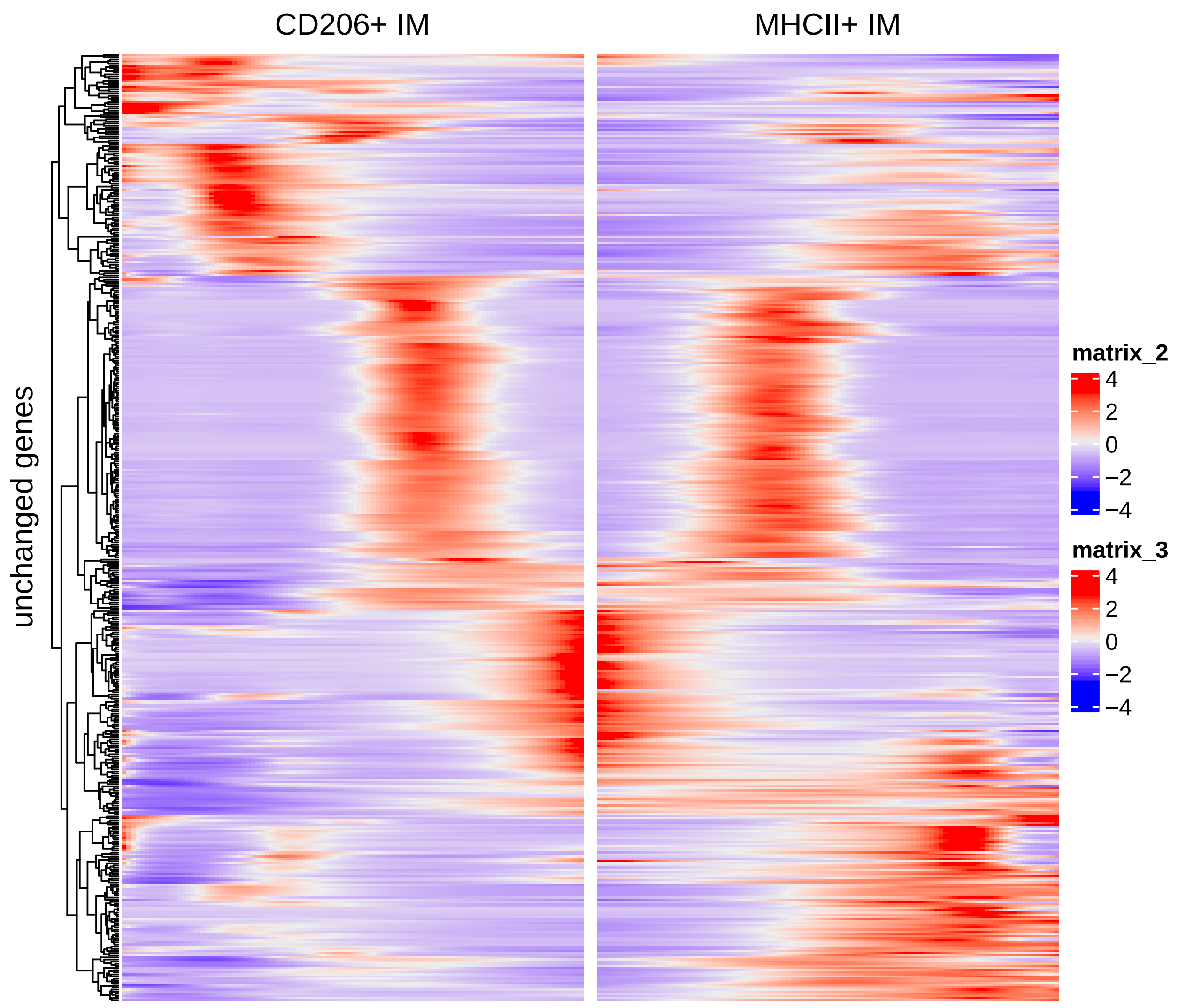
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##    0.0027    3.8599   41.4571  123.7353  168.7434 1263.9587
```

Let's use waldStat > 40 and logFC > 2 as cut threshold.

```r
genes.changed <- rownames(filter(endRes.DE, waldStat > 70 & (logFC1_2 > 2
    | logFC1_2 < -2) ))
genes.noChange <- setdiff(rownames(endRes.DE) , genes.changed)
```

Make heatmap with unchanged/common genes.

```r
heatSmooth_cd206.unchanged <- Heatmap(yhatSmoothScaled[genes.noChange,
    100:1], cluster_columns = FALSE,  show_row_names = FALSE, show_column_
    names = FALSE, column_title = "CD206+ IM")

heatSmooth_mhcii.unchanged <- Heatmap(yhatSmoothScaled[genes.noChange,
    101:200], cluster_columns = FALSE,  show_row_names = FALSE, show_column
    _names = FALSE, column_title = "MHCII+ IM")

heatSmooth_combined.unchanged <- draw ( heatSmooth_cd206.unchanged +
    heatSmooth_mhcii.unchanged, row_title = "unchanged genes", auto_adjust
    = FALSE)
```



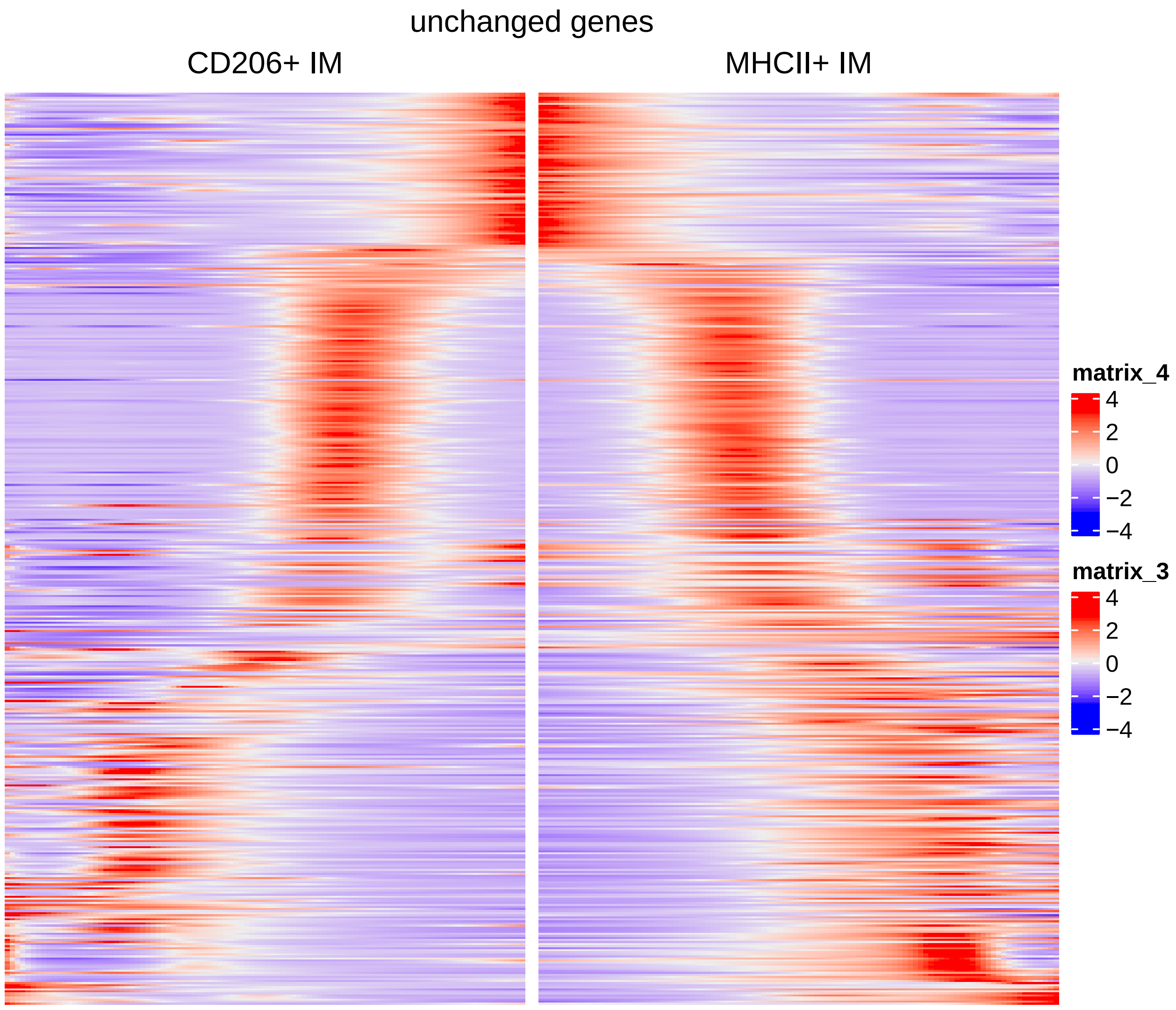
5.4 Daw heatmap with expression patterns of unchanged/common genes in the order of pseudotime

Let's find the expression peak of each gene:

```r
orderbyExpressionPeak <- function(x, # matrix
                                  decreasing = FALSE,
                                  output.position = FALSE # if true, give
                                     relative position 0 - 1, or output
                                     order.
                                  ) {
  indx.peak <- apply(x, 1 , which.max)

  if(output.position) {
    po <- indx.peak/nrow(x)
    if (! length(rownames(x)) == 0) {names(po) <- rownames(x)}
    return(po)
  } else {
    o <- order(indx.peak)
    if (! length(rownames(x)) == 0) {names(o) <- rownames(x)[o]}
    return (o)}
}
```

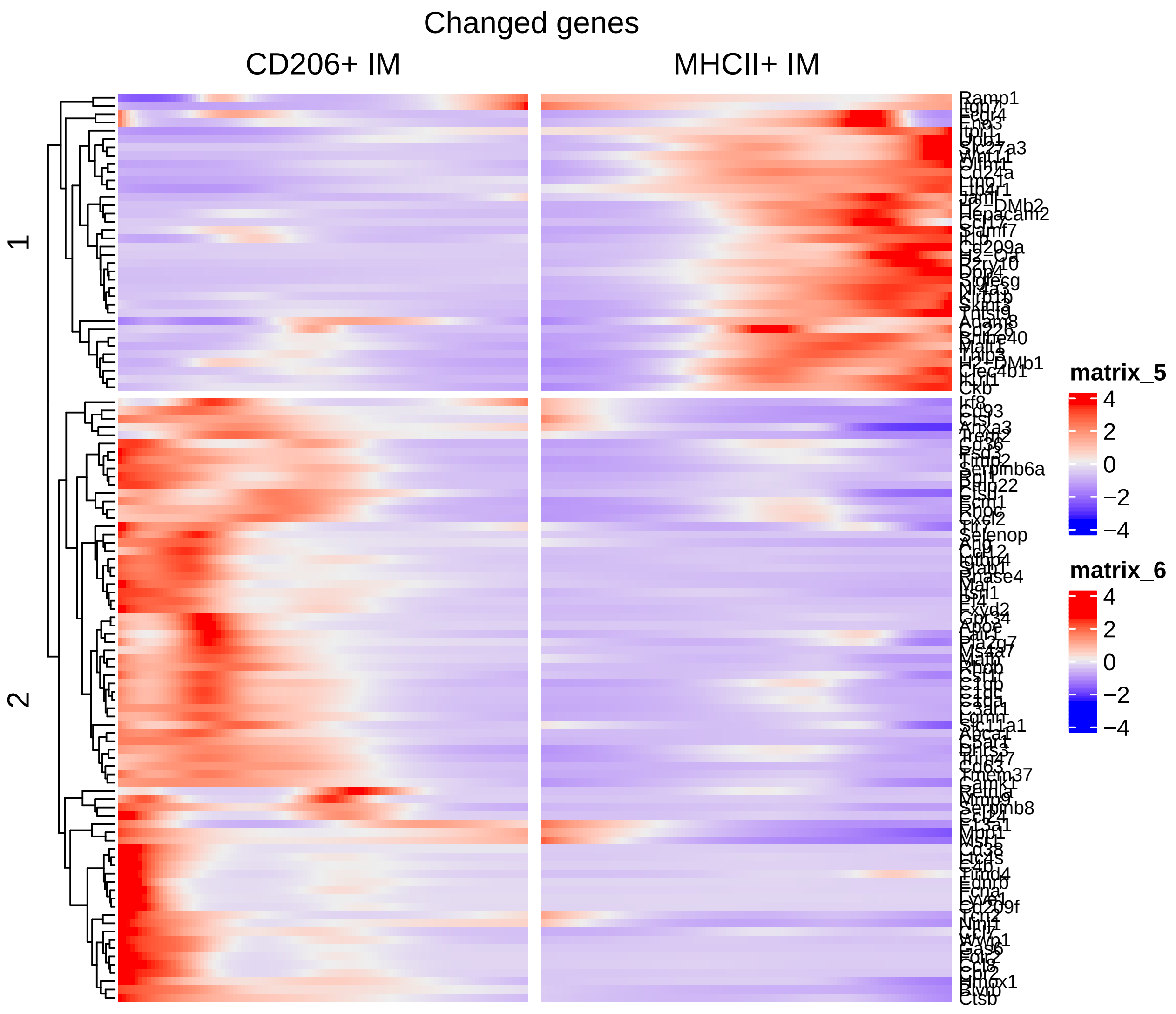
Make average peak pseudotime peak for each gene:

```
po.cd206 <- orderbyExpressionPeak(yhatSmoothScaled[genes.noChange, 1:100],
    output.position = TRUE)
po.mhcii <- orderbyExpressionPeak(yhatSmoothScaled[genes.noChange,
   101:200], output.position = TRUE)
order.mean <- order ( ( po.cd206 + po.mhcii ) /2)

heatSmooth_cd206.unchanged.ordered <- Heatmap(yhatSmoothScaled[genes.
   noChange, 100:1], cluster_columns = FALSE,  show_row_names = FALSE,
   show_column_names = FALSE, row_order = order.mean,  column_title = "
   CD206+␣IM")

heatSmooth_combined.unchanged.ordered <- draw ( heatSmooth_cd206.unchanged
   .ordered + heatSmooth_mhcii.unchanged, column_title = "unchanged␣genes"
   , auto_adjust = FALSE)
```



5.5 Make with changed/specific genes

```
heatSmooth_cd206.changed <- Heatmap(yhatSmoothScaled[genes.changed,
    100:1], cluster_columns = FALSE,  show_row_names = FALSE, cluster_rows
    = hclust(dist(yhatSmoothScaled[genes.changed, ])), show_column_names =
    FALSE, column_title = "CD206+␣IM")

heatSmooth_mhcii.changed <- Heatmap(yhatSmoothScaled[genes.changed,
    101:200], cluster_columns = FALSE,  show_row_names = TRUE,row_names_gp
    = gpar(fontsize = 8), show_column_names = FALSE, column_title = "MHCII+
    ␣IM")

heatSmooth_combined.changed <- draw ( heatSmooth_cd206.changed +
    heatSmooth_mhcii.changed, column_title = "Changed␣genes", split = 2)
```



6 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] stats4    parallel  grid      stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] clusterExperiment_2.11.2    tradeSeq_1.4.0
##  [3] magrittr_2.0.1              monocle3_1.0.0
##  [5] SingleCellExperiment_1.12.0 SummarizedExperiment_1.20.0
##  [7] GenomicRanges_1.42.0        GenomeInfoDb_1.26.7
##  [9] IRanges_2.24.1              S4Vectors_0.28.1
## [11] MatrixGenerics_1.2.1        matrixStats_0.61.0
## [13] Biobase_2.50.0              BiocGenerics_0.36.1
## [15] circlize_0.4.13             RColorBrewer_1.1-2
## [17] dplyr_1.0.7                 ggplot2_3.3.5
## [19] ComplexHeatmap_2.6.2        SeuratObject_4.0.4
## [21] Seurat_4.0.5
##
## loaded via a namespace (and not attached):
##   [1] scattermore_0.7       princurve_2.1.6       coda_0.19-4
##   [4] pkgmaker_0.32.2       tidyr_1.1.4           bit64_4.0.5
##   [7] knitr_1.36            irlba_2.3.5           DelayedArray_0.16.3
##  [10] data.table_1.14.2     rpart_4.1-15          RCurl_1.98-1.5
##  [13] doParallel_1.0.16     generics_0.1.1        terra_1.4-22
##  [16] cowplot_1.1.1         RSQLite_2.2.9         RANN_2.6.1
##  [19] VGAM_1.1-5            combinat_0.0-8        proxy_0.4-26
##  [22] future_1.23.0         bit_4.0.4             phylobase_0.8.10
##  [25] spatstat.data_2.1-0   xml2_1.3.3            httpuv_1.6.3
##  [28] wk_0.5.0              assertthat_0.2.1      viridis_0.6.2
##  [31] xfun_0.28             hms_1.1.1             evaluate_0.14
##  [34] promises_1.2.0.1      fansi_0.5.0           progress_1.2.2
##  [37] igraph_1.2.9          DBI_1.1.1             htmlwidgets_1.5.4
##  [40] sparsesvd_0.2         spatstat.geom_2.3-0   spdep_1.1-12
##  [43] purrr_0.3.4           ellipsis_0.3.2        DDRTree_0.1.5
##  [46] annotate_1.68.0       gridBase_0.4-7        locfdr_1.1-8
##  [49] deldir_1.0-6          vctrs_0.3.8           Cairo_1.5-12.2
##  [52] ROCR_1.0-11           abind_1.4-5           cachem_1.0.6
##  [55] withr_2.4.3           sctransform_0.3.2      prettyunits_1.1.1
##  [58] goftest_1.2-3         softImpute_1.4-1       cluster_2.1.0
##  [61] ape_5.5               lazyeval_0.2.2         crayon_1.4.2
##  [64] genefilter_1.72.1     edgeR_3.32.1           pkgconfig_2.0.3
##  [67] slam_0.1-49           labeling_0.4.2         units_0.7-2
##  [70] nlme_3.1-153          rlang_0.4.12           globals_0.14.0
##  [73] lifecycle_1.0.1       miniUI_0.1.1.1         registry_0.5-1
##  [76] rsvd_1.0.5            polyclip_1.10-0        lmtest_0.9-39
##  [79] rngtools_1.5.2        Matrix_1.3-4           raster_3.5-2
##  [82] Rhdf5lib_1.12.1       boot_1.3-25            zoo_1.8-9
##  [85] ggridges_0.5.3        GlobalOptions_0.1.2    pheatmap_1.0.12
##  [88] png_0.1-7             viridisLite_0.4.0      rjson_0.2.20
##  [91] bitops_1.0-7          rhdf5filters_1.2.1     rncl_0.8.4
##  [94] KernSmooth_2.23-20    blob_1.2.2             shape_1.4.6
##  [97] classInt_0.4-3        stringr_1.4.0          zinbwave_1.12.0
## [100] slingshot_1.8.0       s2_1.0.7               parallelly_1.29.0
## [103] beachmat_2.6.4        scales_1.1.1           memoise_2.0.1
## [106] plyr_1.8.6            ica_1.0-2              howmany_0.3-1
## [109] gdata_2.18.0          zlibbioc_1.36.0        compiler_4.0.3
## [112] HSMMSingleCell_1.10.0 clue_0.3-60            fitdistrplus_1.1-6
## [115] cli_3.1.0             ade4_1.7-18            XVector_0.30.0
## [118] LearnBayes_2.15.1     listenv_0.8.0          patchwork_1.1.1
## [121] pbapply_1.5-0         MASS_7.3-53            mgcv_1.8-33
## [124] tidyselect_1.1.1      stringi_1.7.6          highr_0.9
## [127] densityClust_0.3      yaml_2.2.1             BiocSingular_1.6.0
## [130] locfit_1.5-9.4        ggrepel_0.9.1          pbmcapply_1.5.0
## [133] tools_4.0.3           future.apply_1.8.1     rstudioapi_0.13
## [136] uuid_1.0-3            monocle_2.18.0         foreach_1.5.1
## [139] RNeXML_2.4.5          gridExtra_2.3          farver_2.1.0
## [142] Rtsne_0.15            digest_0.6.29          FNN_1.1.3
## [145] shiny_1.7.1           qlcMatrix_0.9.7        Rcpp_1.0.7
## [148] later_1.3.0           RcppAnnoy_0.0.19       AnnotationDbi_1
   .52.0
## [151] httr_1.4.2            sf_1.0-4               kernlab_0.9-29
## [154] colorspace_2.0-2      XML_3.99-0.8           tensor_1.5
## [157] reticulate_1.22       splines_4.0.3          uwot_0.1.11
## [160] expm_0.999-6          spatstat.utils_2.2-0   sp_1.4-6
## [163] plotly_4.10.0         spData_2.0.1           xtable_1.8-4
## [166] jsonlite_1.7.2        R6_2.5.1               gmodels_2.18.1
## [169] pillar_1.6.4          htmltools_0.5.2        mime_0.12
## [172] NMF_0.23.0            glue_1.5.1             fastmap_1.1.0
## [175] BiocParallel_1.24.1   class_7.3-17           codetools_0.2-18
## [178] utf8_1.2.2            lattice_0.20-41        spatstat.sparse_2
   .0-0
## [181] tibble_3.1.6          leiden_0.3.9           gtools_3.9.2
## [184] magick_2.7.3          survival_3.2-7         limma_3.46.0
## [187] rmarkdown_2.11        docopt_0.7.1           fastICA_1.2-3
## [190] munsell_0.5.0         rhdf5_2.34.0           e1071_1.7-9
## [193] GetoptLong_1.0.5      GenomeInfoDbData_1.2.4 iterators_1.0.13
## [196] HDF5Array_1.18.1      reshape2_1.4.4         gtable_0.3.0
## [199] spatstat.core_2.3-2
```

7 References