# Differential Expression

Ahmed Mahfouz and Indu Khatri

11/16/2021

## Overview

In this tutorial we will explore different methods to perform differential expression analysis on scRNA-seq data. The exercises are based on Seurat's differential expression testing vignette.

Load required packages:

```
require(Seurat)
require(scran)
require(scater)
require(pheatmap)
```

We will continue with our PBMC dataset.

```
pbmc <- readRDS(file = "pbmc3k_Clust.rds")
```

### Differential expression testing in Seurat

In Seurat, differential expression analysis can be performed using the `FindMarkers` function. As a default, Seurat performs differential expression based on the non-parameteric Wilcoxon rank sum test. Differential expression is performed between groups of cells. To test for differential expression between two specific groups of cells, specify the `ident.1` and `ident.2` parameters. The function will automatically retrieve the cluster identities from the Seurat object using the `Idents()` function.

Before applying the function, we first have to change the identities to the original `celltype` column, as we have changed them in the clustering lab.

```
levels(pbmc)

## [1] "NK and T cells" "Monocytes"      "B cells"

Idents(pbmc) <- 'celltype'
levels(pbmc)

## [1] "Monocyte"       "B cell"         "CD8 T cell"     "CD4 T cell"
## [5] "NK cell"        "Dendritic cell"

# Find differentially expressed features between CD8 and CD4 T-cells
tcell.de.markers <- FindMarkers(pbmc, ident.1 = "CD8 T cell", ident.2 = "CD4
T cell")
```

```
# View results
head(tcell.de.markers)

##             p_val avg_log2FC pct.1 pct.2    p_val_adj
## NKG7 7.901233e-61   3.843329 0.741 0.027 2.649915e-56
## CD8A 2.501225e-55   1.986264 0.695 0.020 8.388608e-51
## CTSW 1.230991e-54   1.991361 0.925 0.314 4.128497e-50
## CCL5 2.573606e-50   3.067194 0.764 0.109 8.631360e-46
## GZMA 2.840748e-45   2.570760 0.707 0.109 9.527300e-41
## CST7 4.187390e-41   2.194723 0.609 0.058 1.404367e-36
```

The results data frame has the following columns:

- `p_val`: Unadjusted p-value
- `avg_log2FC`: log fold-change of the average expression between the two groups. Positive values indicate that the feature is more highly expressed in the first group.
- `pct.1`: The percentage of cells where the feature is detected in the first group.
- `pct.2`: The percentage of cells where the feature is detected in the second group.
- `p_val_adj`: Adjusted p-value, based on Bonferroni correction using all features in the dataset.

If the ident.2 parameter is omitted or set to NULL, `FindMarkers` will test for differentially expressed features between the group specified by ident.1 and all other cells.

```
# Find differentially expressed features between CD8 T-cells and all other ce
lls, only
# search for positive markers
tcell.de.markers <- FindMarkers(pbmc, ident.1 = "CD8 T cell", ident.2 = NULL,
only.pos = TRUE)
# view results
head(tcell.de.markers)

##                p_val avg_log2FC pct.1 pct.2     p_val_adj
## CD8A  1.068397e-126   1.961412 0.695 0.022 3.583191e-122
## CD8B   4.247617e-98   1.901227 0.523 0.011  1.424566e-93
## CCL5   6.657331e-89   2.550836 0.764 0.101  2.232736e-84
## GZMK   9.705468e-88   2.580187 0.615 0.050  3.255020e-83
## CTSW   6.102467e-87   1.482927 0.925 0.184  2.046645e-82
## TRGC2  3.555947e-72   1.823488 0.546 0.053  1.192593e-67
```

To increase the speed of marker discovery, particularly for large datasets, Seurat allows for pre-filtering of features or cells. For example, features that are very infrequently detected in either group of cells, or features that are expressed at similar average levels, are unlikely to be differentially expressed. Example use cases of the `min.pct`, `logfc.threshold`, `min.diff.pct`, and `max.cells.per.ident` parameters are demonstrated below.

```
# Pre-filter features that are detected at <50% frequency in either CD8 T-cel
ls or CD4 T-cells.
head(FindMarkers(pbmc, ident.1 = "CD8 T cell", ident.2 = "CD4 T cell", min.pc
t = 0.5))

##             p_val avg_log2FC pct.1 pct.2    p_val_adj
## NKG7 7.901233e-61   3.843329 0.741 0.027 2.649915e-56
## CD8A 2.501225e-55   1.986264 0.695 0.020 8.388608e-51
## CTSW 1.230991e-54   1.991361 0.925 0.314 4.128497e-50
## CCL5 2.573606e-50   3.067194 0.764 0.109 8.631360e-46
## GZMA 2.840748e-45   2.570760 0.707 0.109 9.527300e-41
## CST7 4.187390e-41   2.194723 0.609 0.058 1.404367e-36
```

***Question 7:*** *Find markers using different parameters: Find markers 1) using filter for less than a two-fold change between the average expression of CD8 T-cells vs CD4 T-cells ( HINT: logfc.threshold = log(2) ) and 2) detection percentages across the two groups are similar (within 0.25) (HINT: min.diff.pct = 0.25)*

Finally, you can also identify all cluster markers in one go using `FindAllMarkers`.

```
head(FindAllMarkers(pbmc, logfc.threshold = log(2), min.pct = 0.5, min.diff.p
ct = 0.25))

## Calculating cluster Monocyte

## Calculating cluster B cell

## Calculating cluster CD8 T cell

## Calculating cluster CD4 T cell

## Calculating cluster NK cell

## Calculating cluster Dendritic cell

##                  p_val avg_log2FC pct.1 pct.2     p_val_adj  cluster     g
ene
## FCN1     2.665083e-214   4.058224 0.994 0.012 8.938154e-210 Monocyte     F
CN1
## SERPINA1 3.087393e-212   2.907682 0.982 0.008 1.035450e-207 Monocyte SERPI
NA1
## FGL2     3.210232e-203   2.984234 0.991 0.042 1.076648e-198 Monocyte     F
GL2
## MNDA     9.642581e-203   3.649053 0.979 0.032 3.233929e-198 Monocyte     M
NDA
## CSTA     1.058587e-200   2.552211 0.955 0.012 3.550290e-196 Monocyte     C
STA
## VCAN     2.667989e-199   4.023919 0.943 0.012 8.947902e-195 Monocyte     V
CAN
```

### Alternative DE tests in Seurat

The following differential expression tests are currently supported by Seurat:

- `wilcox`: Wilcoxon rank sum test (default)
- `bimod`: Likelihood-ratio test for single cell feature expression, (McDavid et al., Bioinformatics, 2013)
- `roc`: Standard AUC classifier
- `t`: Student's t-test
- `poisson`: Likelihood ratio test assuming an underlying negative binomial distribution.
- `negbinom`: Likelihood ratio test assuming an underlying negative binomial distribution.
- `LR`: Uses a logistic regression framework to determine differentially expressed genes. Constructs a logistic regression model predicting group membership based on each feature individually and compares this to a null model with a likelihood ratio test.
- `MAST`: GLM-framework that treates cellular detection rate as a covariate (Finak et al, Genome Biology, 2015)
- `DESeq2`: DE based on a model using the negative binomial distribution (Love et al, Genome Biology, 2014)

For MAST and DESeq2 please ensure that these packages are installed separately in order to use them as part of Seurat. Once installed, the `test.use` parameter can be used to specify which DE test to use.

## Differential expression analysis using scran

The `findMarkers()` function in scran uses a different approach to identify marker genes compared to Seurat. While in Seurat the default is to perform one vs all comparisons, `findMarkers()` in scran performs pairwise comparisons between clusters for each gene. The default test in `findMarkers()` is the Welch t-test.

Scran intentionally uses pairwise comparisons between clusters rather than comparing each cluster to the average of all other cells. The latter approach is sensitive to the population composition, potentially resulting in substantially different sets of markers when cell type abundances change in different contexts. In the worst case, the presence of a single dominant subpopulation will drive the selection of top markers for every other cluster, pushing out useful genes that can resolve the various minor subpopulations.

First, let's convert our Seurat object to a SingleCellExperiment object.

```
pbmc.sce <- as.SingleCellExperiment(pbmc)
```

`findMarkers()` returns a list of data frames containing ranked candidate markers for each cluster.

```
markers.pbmc <- findMarkers(pbmc.sce, groups=pbmc.sce$ident)
```

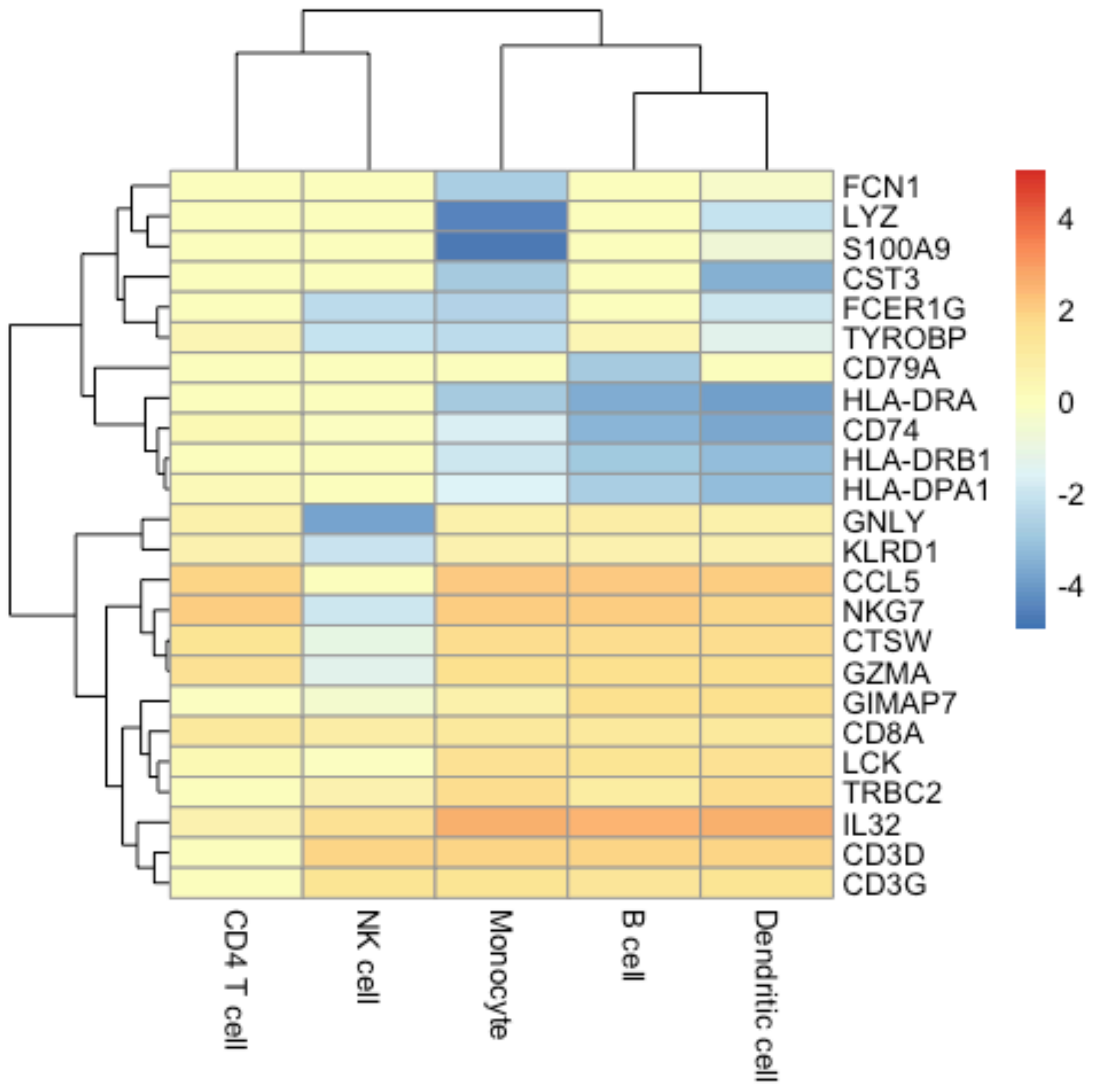
You can then choose one data frame (in this example, corresponding to CD8 T-cells). This data frame contains log2-fold changes of expression in the chosen cluster over each other cluster as well as several statistics obtained by combining p-values across the pairwise comparisons involving the cluster of interest.

```
chosen <- "CD8 T cell"
interesting <- markers.pbmc[[chosen]]
interesting[1:10,1:4]

## DataFrame with 10 rows and 4 columns
##                Top      p.value          FDR summary.logFC
##          <integer>    <numeric>    <numeric>     <numeric>
## HLA-DRB1         1 6.03248e-176 1.12398e-172      -2.95127
## CTSW             1  2.43221e-72  4.77027e-70       1.65306
## CD3D             1  5.83999e-96  1.99859e-93       1.83222
## IL32             1  1.13157e-97  4.12505e-95       2.47244
## CST3             1  0.00000e+00  0.00000e+00      -2.86031
## LCK              2  2.74067e-75  5.81750e-73       1.44093
## FCER1G           2 7.94116e-310 1.33165e-305      -2.58399
## CD3G             2  4.02478e-61  5.79326e-59       1.29761
## CCL5             2  1.35929e-47  1.36490e-45       2.03777
## CD79A            2 6.39837e-171 1.12941e-167      -2.84229
```

The `summary.logFC` field provides a summary of the direction and effect size for each gene. `logFC` is defined here as the log-fold change from the comparison with the lowest p-value. The `p.value` field contains the combined p-value that is obtained by applying Simes' method to the pairwise p-values for each gene. Of particular interest is the `Top` field. The set of genes with `Top` $\leq X$ is the union of the top $X$ genes (ranked by p-value) from each pairwise comparison involving the cluster of interest.

Let's plot a heatmap of the top 5 genes for CD8 T-cells.

```
best.set <- interesting[interesting$Top <= 5,]
logFCs <- getMarkerEffects(best.set)
pheatmap(logFCs, breaks=seq(-5, 5, length.out=101))
```



## Wilcoxon vs t-test

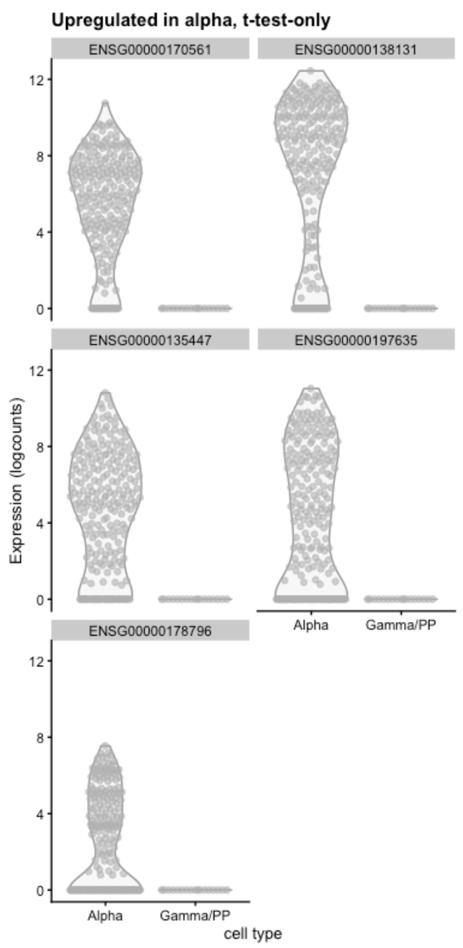
Also in scran, you can use different DE tests. Beside the default Welch t-test, you can also use a Wilcoxon rank-sum test or a binomial test.

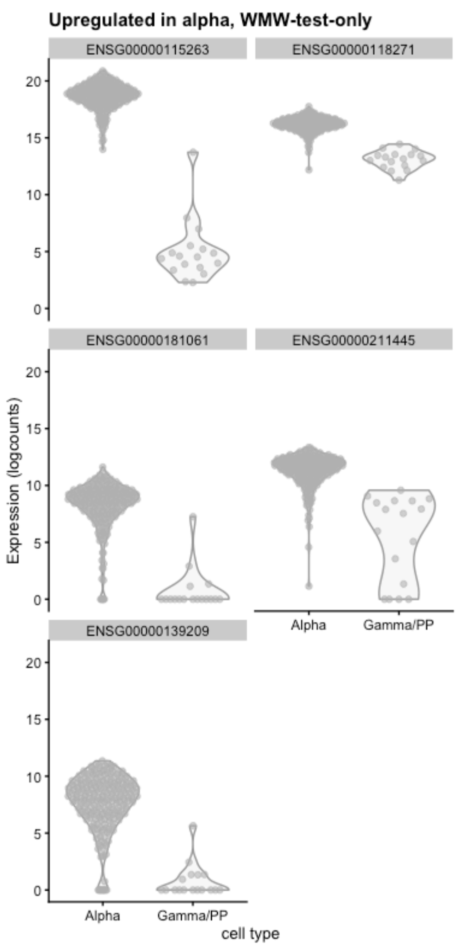
```
markers.pbmc.wrs <- findMarkers(pbmc.sce, groups=pbmc.sce$ident, test="wilcox
")
interesting.wrs <- markers.pbmc.wrs[[chosen]]
interesting.wrs[1:10,1:4]

## DataFrame with 10 rows and 4 columns
##                Top      p.value         FDR summary.AUC
##          <integer>    <numeric>   <numeric>   <numeric>
## FCER1G           1  1.16092e-78 3.89351e-75 1.72586e-05
## MARCH1           1  2.87753e-64 9.94913e-62 5.70571e-02
## CD3E             1 3.22427e-103 1.08136e-98 9.97101e-01
## CD79A            1  8.03182e-66 3.28502e-63 0.00000e+00
## NKG7           1  3.95062e-60 1.01920e-57 8.66100e-01
## CD8A           2  1.13117e-65 4.57073e-63 8.47701e-01
## KLRF1          2  2.29872e-32 1.59947e-30 4.91071e-02
## RNASE6         2  9.78777e-40 9.71190e-38 2.94118e-02
## IL32           2  1.48340e-99 1.72695e-95 9.88092e-01
## TGFBI          3  1.60505e-62 5.03085e-60 6.32270e-02
```

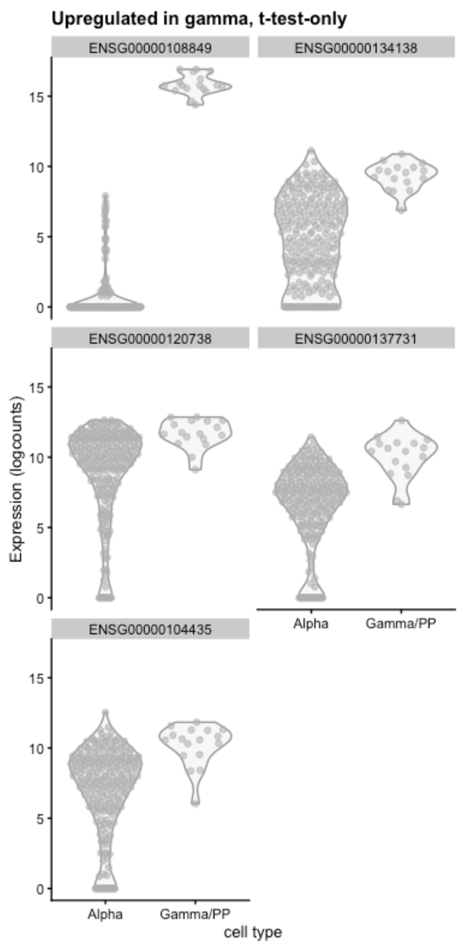
One advantage of the Wilcoxon rank-sum test over the Welch t-test is that it is symmetric with respect to differences in the size of the groups being compared. In other words, it is less affected by the number of cells in each group. On the other hand, the t-test will favor genes where the larger group has the higher relative variance as this increases the estimated degrees of freedom and decreases the resulting p-value.

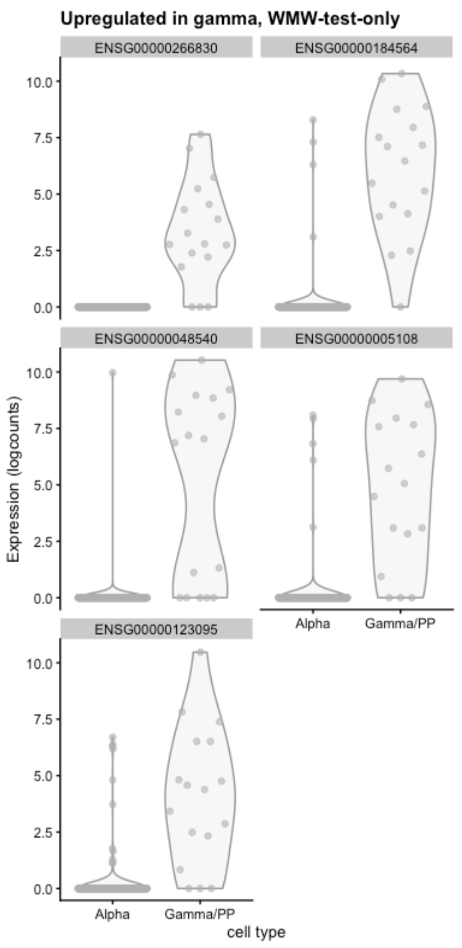
To illustrate this we will use an example from "Orchestrating Single-Cell Analysis with Bioconductor". In this example, we will compare alpha and gamma cells in the human pancreas data set from Lawlor et al. (2017)

```
sce.lawlor <- readRDS(file = "sce_lawlor.rds")
marker.lawlor.t <- findMarkers(sce.lawlor, groups=sce.lawlor$`cell type`,
                               direction="up", restrict=c("Alpha", "Gamma/PP"
))
marker.lawlor.w <- findMarkers(sce.lawlor, groups=sce.lawlor$`cell type`,
                               direction="up", restrict=c("Alpha", "Gamma/PP"
), test.type="wilcox")
# Upregulated in alpha:
marker.alpha.t <- marker.lawlor.t$Alpha
marker.alpha.w <- marker.lawlor.w$Alpha
chosen.alpha.t <- rownames(marker.alpha.t)[1:5]
chosen.alpha.w <- rownames(marker.alpha.w)[1:5]
u.alpha.t <- setdiff(chosen.alpha.t, chosen.alpha.w)
u.alpha.w <- setdiff(chosen.alpha.w, chosen.alpha.t)
# Upregulated in gamma:
marker.gamma.t <- marker.lawlor.t$`Gamma/PP`
marker.gamma.w <- marker.lawlor.w$`Gamma/PP`
chosen.gamma.t <- rownames(marker.gamma.t)[1:5]
chosen.gamma.w <- rownames(marker.gamma.w)[1:5]
u.gamma.t <- setdiff(chosen.gamma.t, chosen.gamma.w)
u.gamma.w <- setdiff(chosen.gamma.w, chosen.gamma.t)
# Examining all uniquely detected markers in each direction.
subset <- sce.lawlor[,sce.lawlor$`cell type` %in% c("Alpha", "Gamma/PP")]
gridExtra::grid.arrange(
  plotExpression(subset, x="cell type", features=u.alpha.t, ncol=2) +
    ggtitle("Upregulated in alpha, t-test-only"),
  plotExpression(subset, x="cell type", features=u.alpha.w, ncol=2) +
    ggtitle("Upregulated in alpha, WMW-test-only"),
  plotExpression(subset, x="cell type", features=u.gamma.t, ncol=2) +
    ggtitle("Upregulated in gamma, t-test-only"),
  plotExpression(subset, x="cell type", features=u.gamma.w, ncol=2) +
    ggtitle("Upregulated in gamma, WMW-test-only"),
  ncol=2
)
```


Upregulated in alpha, t-test-only
ENSG00000170561
ENSG00000138131
ENSG00000135447
ENSG00000197635
ENSG00000178796
Expression (logcounts)
Alpha
Gamma/PP
cell type


Upregulated in alpha, WMW-test-only
ENSG00000115263
ENSG00000118271
ENSG00000181061
ENSG00000211445
ENSG00000139209
Expression (logcounts)
Alpha
Gamma/PP
cell type


Upregulated in gamma, t-test-only
ENSG00000108849
ENSG00000134138
ENSG00000120738
ENSG00000137731
ENSG00000104435
Expression (logcounts)
Alpha
Gamma/PP
cell type


Upregulated in gamma, WMW-test-only
ENSG00000266830
ENSG00000184564
ENSG00000048540
ENSG00000005108
ENSG00000123095
Expression (logcounts)
Alpha
Gamma/PP
cell type

***Question 8:*** *Can you observe the effects of the tests in the resulting genes?*

## Session info

```
sessionInfo()

## R version 4.1.0 (2021-05-18)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Mojave 10.14.6
## 
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRbla
s.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlap
ack.dylib
## 
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
## 
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets 
## [8] methods   base     
## 
## other attached packages:
##  [1] pheatmap_1.0.12             scater_1.20.1              
##  [3] ggplot2_3.3.5               scran_1.20.1               
##  [5] scuttle_1.2.1               SingleCellExperiment_1.14.1
##  [7] SummarizedExperiment_1.22.0 Biobase_2.52.0             
##  [9] GenomicRanges_1.44.0        GenomeInfoDb_1.28.4        
## [11] IRanges_2.26.0              S4Vectors_0.30.0           
## [13] BiocGenerics_0.38.0         MatrixGenerics_1.4.3       
## [15] matrixStats_0.60.1          SeuratObject_4.0.2         
## [17] Seurat_4.0.4               
## 
## loaded via a namespace (and not attached):
##   [1] plyr_1.8.6                igraph_1.2.6             
##   [3] lazyeval_0.2.2            splines_4.1.0            
##   [5] BiocParallel_1.26.2       listenv_0.8.0            
##   [7] scattermore_0.7           digest_0.6.27            
##   [9] htmltools_0.5.2           viridis_0.6.1            
##  [11] fansi_0.5.0               magrittr_2.0.1           
##  [13] ScaledMatrix_1.0.0        tensor_1.5               
##  [15] cluster_2.1.2             ROCR_1.0-11              
##  [17] limma_3.48.3              globals_0.14.0           
##  [19] spatstat.sparse_2.0-0     colorspace_2.0-2         
##  [21] ggrepel_0.9.1             xfun_0.26                
##  [23] dplyr_1.0.7               crayon_1.4.1             
##  [25] RCurl_1.98-1.5            jsonlite_1.7.2           
##  [27] spatstat.data_2.1-0       survival_3.2-13          
##  [29] zoo_1.8-9                 glue_1.4.2               
##  [31] polyclip_1.10-0           gtable_0.3.0             
##  [33] zlibbioc_1.38.0            XVector_0.32.0
##  [35] leiden_0.3.9               DelayedArray_0.18.0
##  [37] BiocSingular_1.8.1         future.apply_1.8.1
##  [39] abind_1.4-5                scales_1.1.1
##  [41] DBI_1.1.1                  edgeR_3.34.1
##  [43] miniUI_0.1.1.1             Rcpp_1.0.7
##  [45] viridisLite_0.4.0          xtable_1.8-4
##  [47] reticulate_1.22            spatstat.core_2.3-0
##  [49] dqrng_0.3.0                rsvd_1.0.5
##  [51] metapod_1.0.0              htmlwidgets_1.5.4
##  [53] httr_1.4.2                 RColorBrewer_1.1-2
##  [55] ellipsis_0.3.2             ica_1.0-2
##  [57] farver_2.1.0               pkgconfig_2.0.3
##  [59] uwot_0.1.10                deldir_0.2-10
##  [61] locfit_1.5-9.4             utf8_1.2.2
##  [63] labeling_0.4.2             tidyselect_1.1.1
##  [65] rlang_0.4.11               reshape2_1.4.4
##  [67] later_1.3.0                munsell_0.5.0
##  [69] tools_4.1.0                generics_0.1.0
##  [71] ggridges_0.5.3             evaluate_0.14
##  [73] stringr_1.4.0              fastmap_1.1.0
##  [75] yaml_2.2.1                 goftest_1.2-2
##  [77] knitr_1.34                 fitdistrplus_1.1-5
##  [79] purrr_0.3.4                RANN_2.6.1
##  [81] pbapply_1.5-0              future_1.22.1
##  [83] nlme_3.1-153               sparseMatrixStats_1.4.2
##  [85] mime_0.11                  compiler_4.1.0
##  [87] beeswarm_0.4.0             plotly_4.9.4.1
##  [89] png_0.1-7                  spatstat.utils_2.2-0
##  [91] statmod_1.4.36             tibble_3.1.4
##  [93] stringi_1.7.4              highr_0.9
##  [95] lattice_0.20-44            bluster_1.2.1
##  [97] Matrix_1.3-4               vctrs_0.3.8
##  [99] pillar_1.6.2               lifecycle_1.0.0
## [101] spatstat.geom_2.2-2        lmtest_0.9-38
## [103] RcppAnnoy_0.0.19           BiocNeighbors_1.10.0
## [105] data.table_1.14.0          cowplot_1.1.1
## [107] bitops_1.0-7               irlba_2.3.3
## [109] httpuv_1.6.3               patchwork_1.1.1
## [111] R6_2.5.1                   promises_1.2.0.1
## [113] KernSmooth_2.23-20         gridExtra_2.3
## [115] vipor_0.4.5                parallelly_1.28.1
## [117] codetools_0.2-18           MASS_7.3-54
## [119] assertthat_0.2.1           withr_2.4.2
## [121] sctransform_0.3.2          GenomeInfoDbData_1.2.6
## [123] mgcv_1.8-36                grid_4.1.0
## [125] rpart_4.1-15               beachmat_2.8.1
## [127] tidyr_1.1.3                rmarkdown_2.11
## [129] DelayedMatrixStats_1.14.3 Rtsne_0.15
## [131] shiny_1.6.0                ggbeeswarm_0.6.0
```