SLBP analysis

Thomas Schwarzl, Sudeep Sahadevan and Thileepan Sekaran

3/12/2021

Contents

1 SLBP analysis

This markdown file contains the complete analysis of SLBP example dataset and follows the same analysis steps and file naming patterns as it is described in the chapter.

1.1 Installation of DEWSeq and setup

```
if(!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("DEWSeq")
```

1.2 Load libraries

```
suppressPackageStartupMessages(require(DEWSeq))
suppressPackageStartupMessages(require(IHW))
suppressPackageStartupMessages(require(tidyverse))
suppressPackageStartupMessages(require(data.table))
suppressPackageStartupMessages(require(ggplot2))
suppressPackageStartupMessages(require(ggrepel))
```

`suppressPackageStartupMessages` ignores all start up messages

1.3 Data import

```
setwd('/path/to/SLBP_analysis')
```

```
count_matrix <- fread('counts/SLBP_K562_w50s20_counts.txt.gz',
                      stringsAsFactors=FALSE, sep="\t", header=TRUE)
count_matrix <- column_to_rownames(count_matrix,'unique_id')
head(count_matrix)
```

```
##                                    ENCFF218ZEI ENCFF511HSJ ENCFF879UID
## ENSG00000227232.5:intron0008W00093           1           0           0
## ENSG00000227232.5:intron0008W00094           1           0           0
## ENSG00000268903.1:exon0001W00003             1           0           2
## ENSG00000268903.1:exon0001W00004             3           0           2
## ENSG00000268903.1:exon0001W00005             3           0           2
## ENSG00000268903.1:exon0001W00006             2           0           0
```

```
col_data <- data.frame(type=c('IP','IP','SMI'),
                       row.names=colnames(count_matrix))
head(col_data)
```

```
##             type
## ENCFF218ZEI   IP
## ENCFF511HSJ   IP
## ENCFF879UID  SMI
```

```
annotation_file <- 'annotation/SLBP_K562_w50s20_annotation.txt.gz'
```

1.3.1 Create DESeqDataset

```
ddw <- DESeqDataSetFromSlidingWindows(countData=count_matrix, colData=col_data,
                                      annotObj=annotation_file, design=~type)
```

```
## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
```

```
head(ddw)
```

```
## class: DESeqDataSet
## dim: 6 3
## metadata(1): version
## assays(1): counts
## rownames(6): ENSG00000225630.1:exon0001W00013
##   ENSG00000225630.1:exon0001W00014 ... ENSG00000225630.1:exon0001W00017
##   ENSG00000225630.1:exon0001W00018
## rowData names(8): unique_id gene_id ... Total_nr_of_region
##   window_number
## colnames(3): ENCFF218ZEI ENCFF511HSJ ENCFF879UID
## colData names(1): type
```

1.4 Estimate size factors

```
ddw <- estimateSizeFactors(ddw)
sizeFactors(ddw)
```

```
## ENCFF218ZEI ENCFF511HSJ ENCFF879UID
##   0.8735805   0.7272363   1.5874011
```

1.4.1 Estimate size factors for only protein coding genes

```
ddw_mRNAs <- ddw[ rowData(ddw)[,"gene_type"] == "protein_coding", ]
ddw_mRNAs <- estimateSizeFactors(ddw_mRNAs)
sizeFactors(ddw) <- sizeFactors(ddw_mRNAs)
```

```
sizeFactors(ddw_mRNAs)
```

```
## ENCFF218ZEI ENCFF511HSJ ENCFF879UID
##   0.8735805   0.7937005   1.5874011
```

1.4.2 Estimate size factors without significant windows

In this context, the aim is to use only the windows that which do not show a large difference between IP and SMI samples for normalization.

This step uses `local` fit for dispersion estimation and Likelihood-ratio test (LRT)

```
ddw_tmp <- ddw
ddw_tmp <- estimateDispersions(ddw_tmp, fitType = "local", quiet = TRUE)
ddw_tmp <- nbinomLRT(ddw_tmp,full = ~type,reduced = ~1)
tmp_significant_windows <-
                results(ddw_tmp,
                    contrast = c("type", "IP", "SMI"),
                    tidy = TRUE,
                    filterFun = ihw) %>%
                dplyr::filter(padj < 0.05) %>%
                .[["row"]]
rm(ddw_tmp)
```

Now estimate the size factors without the significant windows.

```
ddw_mRNAs <- ddw_mRNAs[ !rownames(ddw_mRNAs) %in% tmp_significant_windows, ]
ddw_mRNAs <- estimateSizeFactors(ddw_mRNAs)
sizeFactors(ddw) <- sizeFactors(ddw_mRNAs)
```

```
sizeFactors(ddw_mRNAs)
```

```
## ENCFF218ZEI ENCFF511HSJ ENCFF879UID
##   0.8735805   0.7937005   1.5874011
```

```
rm(ddw_mRNAs)
```

1.5 Prefiltering

```
keep <- which(rowSums(counts(ddw)>1)>=2)
ddw <- ddw[keep,]
ddw
```

```
## class: DESeqDataSet
## dim: 35601 3
## metadata(1): version
## assays(1): counts
## rownames(35601): ENSG00000225630.1:exon0001W00014
##   ENSG00000225630.1:exon0001W00015 ... ENSG00000210196.2:exon0001W00001
##   ENSG00000210196.2:exon0001W00002
## rowData names(8): unique_id gene_id ... Total_nr_of_region
##   window_number
## colnames(3): ENCFF218ZEI ENCFF511HSJ ENCFF879UID
## colData names(2): type sizeFactor
```

1.6 Estimate dispersion

By default, DESeq2 uses `parametric` fit for dispersion estimation

```
ddw <- estimateDispersions(ddw,quiet=TRUE)
```

Users can also opt for `local` dispersion estimation

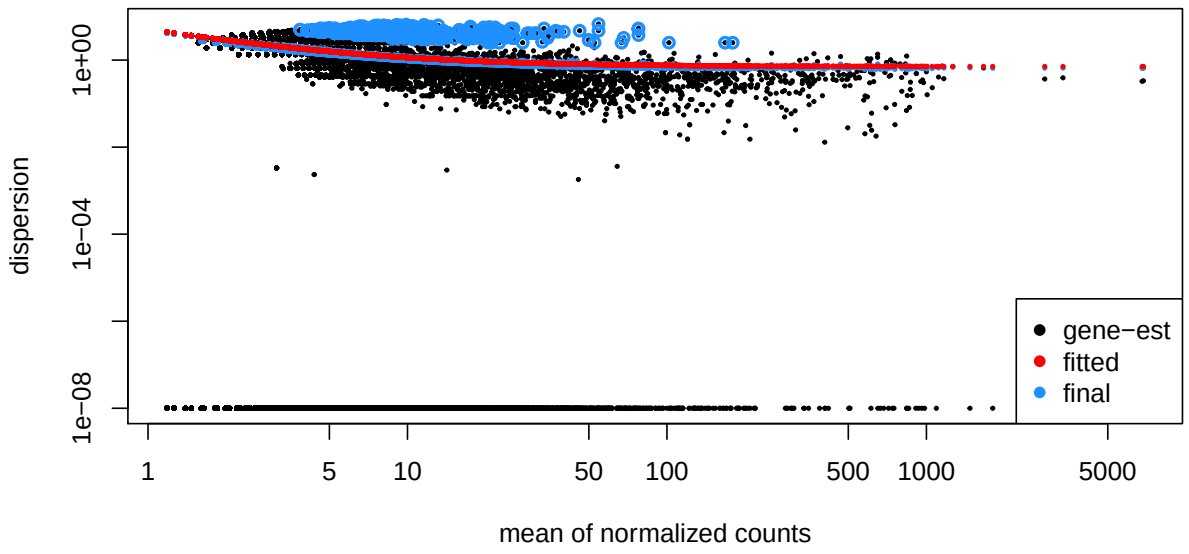
```
ddw <- estimateDispersions(ddw,fitType='local',quiet=TRUE)
```

1.6.1 Decide fit

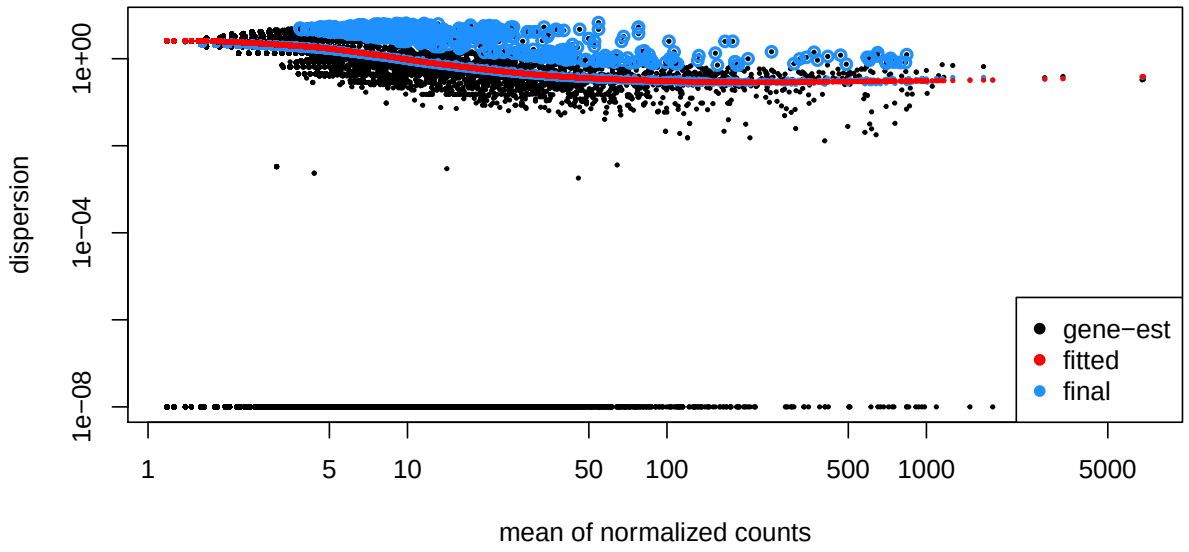
Instead of picking a fit type at random, it is possible to decide the best fit based on the residuals after calculating the dispersion estimates with each type. This step is based on a bioconductor post

```
ddw_param <- estimateDispersions(ddw,quiet=TRUE)
ddw_local <- estimateDispersions(ddw,fitType='local',quiet=TRUE)
```

```
plotDispEsts(ddw_param)
```



```
plotDispEsts(ddw_local)
```




```
parametric_resid <- na.omit(with(mcols(ddw_param),
                                 abs(log(dispGeneEst)-log(dispFit))))
local_resid <- na.omit(with(mcols(ddw_local),
                            abs(log(dispGeneEst)-log(dispFit))))
```

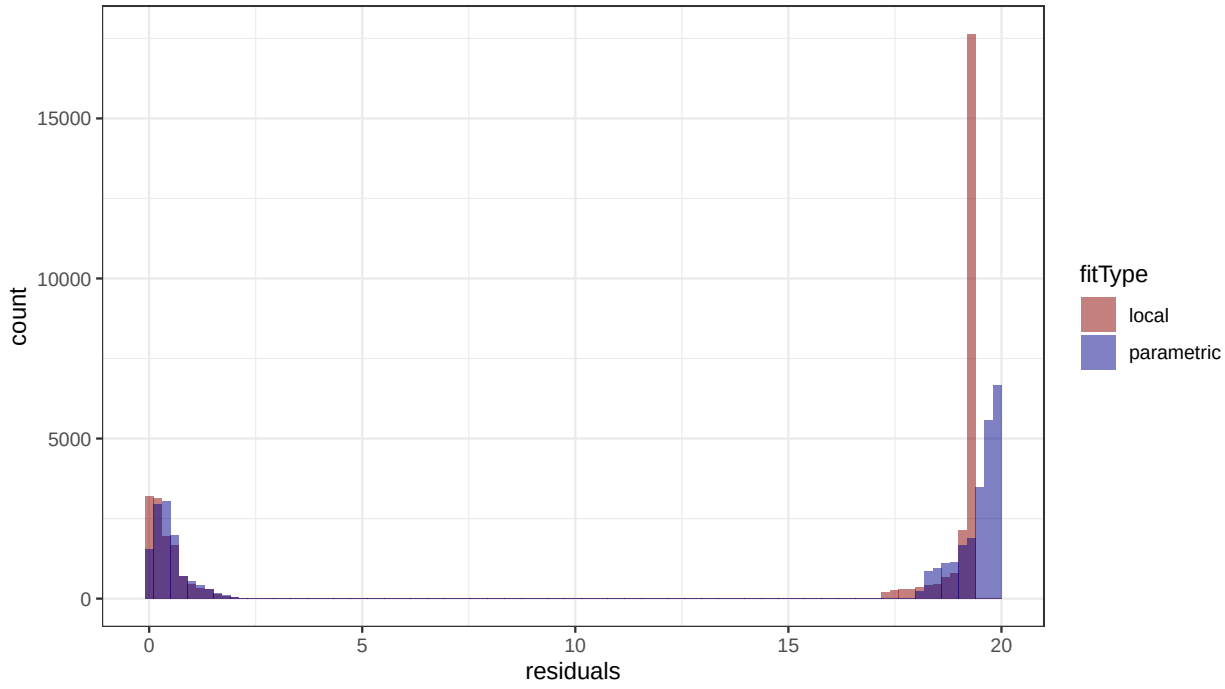
```
summary(parametric_resid)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
##   0.000612   0.632932 19.172591 13.050483 19.761154 19.896069
```

```
summary(local_resid)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
##   0.000115   0.550802 19.178385 12.827602 19.362438 19.376290
```

```
resid_df <- data.frame(residuals=c(parametric_resid,local_resid),
                       fitType=c(rep("parametric",length(parametric_resid)),
                                 rep("local",length(local_resid))))
ggplot(resid_df, aes(x = residuals, fill = fitType)) +
  scale_fill_manual(values = c("darkred", "darkblue")) +
  geom_histogram(alpha = 0.5, position='identity', bins = 100) +
  theme_bw()
```



The residual summaries and histogram clearly indicates that the `local` fit produces a better fit, hence smaller residuals, and therefore we proceed with using local fit for the rest of the analysis.

```
ddw <- ddw_local
```

this can also be done just based on the median values as:

```
if (median(local_resid) <= median(parametric_resid)){
    cat("chosen fitType: local")
    ddw <- ddw_local
  }else{
    cat("chosen fitType: parametric")
    ddw <- ddw_param
  }
```

```
rm(ddw_local,ddw_param,resid_df,parametric_resid,local_resid)
```

1.7 Differential expressed windows analysis

Using LRT

```
ddw <- nbinomLRT(ddw, full = ~type, reduced = ~1)
```

Using Wald test

```
ddw <-  nbinomWaldTest(ddw)
```

1.7.1 Extract significant windows

```
resultWindows <- resultsDEWSeq(ddw, contrast = c("type", "IP", "SMI"),
                               tidy = TRUE) %>% as_tibble
resultWindows
```

```
## # A tibble: 9,211 x 20
##    chromosome  begin    end width strand unique_id gene_id gene_name gene_type
##    <chr>       <dbl>  <int> <int> <chr>  <chr>     <chr>   <chr>     <chr>
##  1 chr1       6.30e5 6.30e5    50 +      ENSG0000~ ENSG00~ MTND2P28  unproces~
##  2 chr1       1.06e6 1.06e6    50 +      ENSG0000~ ENSG00~ AL645608~ sense_in~
##  3 chr1       1.48e6 1.48e6    50 +      ENSG0000~ ENSG00~ ATAD3B    protein_~
##  4 chr1       1.52e6 1.52e6    50 +      ENSG0000~ ENSG00~ ATAD3A    protein_~
##  5 chr1       2.23e6 2.23e6    50 +      ENSG0000~ ENSG00~ SKI       protein_~
##  6 chr1       3.63e6 3.63e6    50 +      ENSG0000~ ENSG00~ TPRG1L    protein_~
##  7 chr1       6.21e6 6.21e6    50 +      ENSG0000~ ENSG00~ RNF207    protein_~
##  8 chr1       6.58e6 6.58e6    39 +      ENSG0000~ ENSG00~ ZBTB48    protein_~
##  9 chr1       1.04e7 1.04e7    50 +      ENSG0000~ ENSG00~ PGD       protein_~
## 10 chr1       1.05e7 1.05e7    50 +      ENSG0000~ ENSG00~ PEX14     protein_~
## # ... with 9,201 more rows, and 11 more variables: gene_region <chr>,
## #   Nr_of_region <int>, Total_nr_of_region <int>, window_number <int>,
## #   baseMean <dbl>, log2FoldChange <dbl>, lfcSE <dbl>, stat <dbl>,
## #   pvalue <dbl>, pSlidingWindows <dbl>, pSlidingWindows.adj <dbl>
```

1.8 Multiple hypothesis correction

1.8.1 FDR correction using BH

```
resultWindows[,'p_adj'] <- p.adjust(resultWindows$pvalue, method="BH")
```

1.8.2 Using IHW

```
resultWindows[,"p_adj_IHW"] <- adj_pvalues(ihw(pvalue ~ baseMean,
                                               data = resultWindows,
                                               alpha = 0.05, nfolds = 10))
```

1.9 Combining windows to regions

```
resultRegions <- extractRegions(windowRes=resultWindows, padjCol="p_adj_IHW",
                                padjThresh = 0.05,
                                log2FoldChangeThresh=1) %>% as_tibble
```

```
## Warning in extractRegions(windowRes = resultWindows, padjCol = "p_adj_IHW", :
## windowRes is a data.table or tibble object, converting it to data.frame
```

```
##   |                                                                              |
```

```
resultRegions
```

```
## # A tibble: 111 x 21
##    chromosome region_begin region_end strand windows_in_regi~ region_length
##    <chr>            <dbl>      <int> <fct>              <dbl>         <int>
##  1 chr1          149813270  149813380 -                      4           110
##  2 chr1          149840697  149840767 -                      2            70
##  3 chr1          149842207  149842357 -                      6           150
##  4 chr1          149843440  149843533 +                      4            93
##  5 chr1          149850292  149850362 -                      2            70
```

```
##  6 chr1          149886138  149886388 -                    11           250
##  7 chr1          149886498  149886628 -                     5           130
##  8 chr1          149886974  149887024 +                     1            50
##  9 chr1          149887254  149887344 +                     3            90
## 10 chr1          149887523  149887673 -                     5           150
## # ... with 101 more rows, and 15 more variables: padj_min <dbl>,
## #   padj_mean <dbl>, padj_max <dbl>, log2FoldChange_min <dbl>,
## #   log2FoldChange_mean <dbl>, log2FoldChange_max <dbl>, regionStartId <chr>,
## #   gene_id <fct>, gene_name <chr>, gene_type <chr>, gene_region <chr>,
## #   Nr_of_region <dbl>, Total_nr_of_region <dbl>, window_number <dbl>,
## #   unique_ids <chr>
```

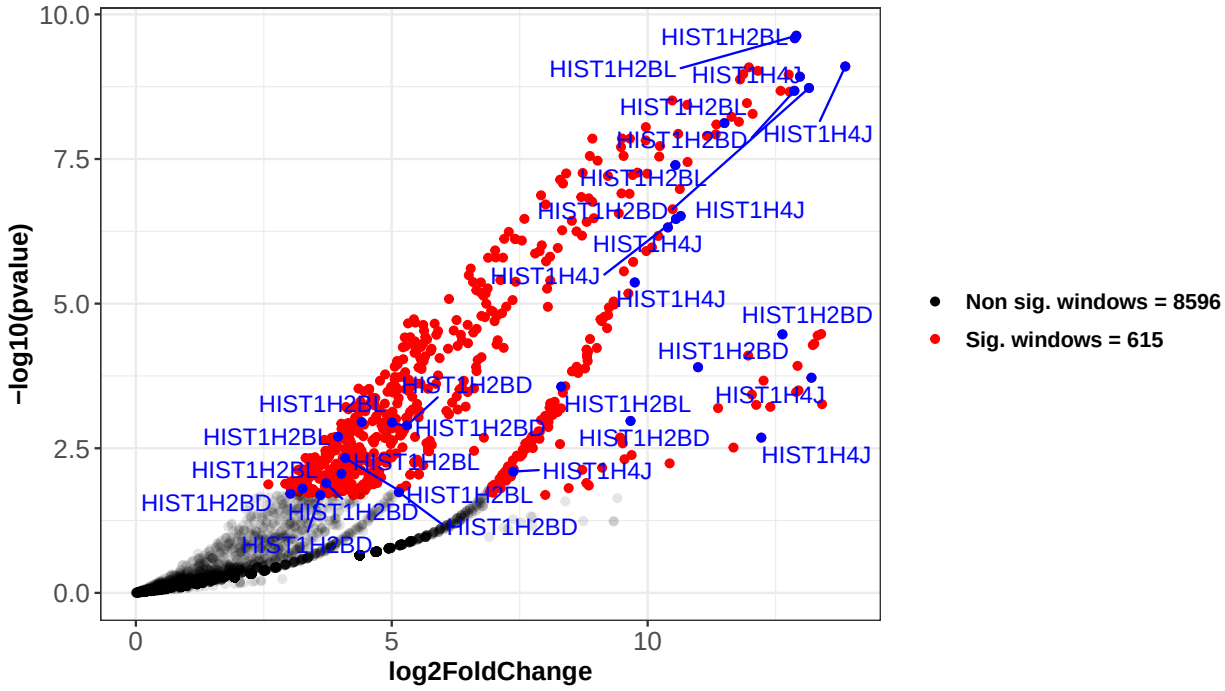
1.10 Exporting results

```
toBED(windowRes=resultWindows, regionRes=resultRegions, padjThresh=0.05,
      padjCol="p_adj_IHW", fileName="SLBP_regions_w50s20.bed")
```

1.11 Visualization

An R function to visualize enriched regions as volcano plots is available in this github repo. Please refer to README files in this repo for additional details.

```
fnSrc <- paste0('https://raw.githubusercontent.com/EMBL-Hentze-group/',
                'DEWSeq_analysis_helpers/master/Volcano_plot/volcanoplot.R')
source(fnSrc)
# subset of all results
tophits <- resultWindows %>% filter( p_adj_IHW<=1e-5 & log2FoldChange>=10 )%>%
                  select(gene_name,log2FoldChange) %>%
                  arrange(-log2FoldChange) %>%
                  select(gene_name) %>%
                  unlist() %>% unique()
volcanoplot(resultWindows,padj_col = 'p_adj_IHW',gene_names = tophits[c(1:3)])
```



1.12 Session info

```
sessionInfo()
```

```
## R version 4.0.0 (2020-04-24)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: CentOS Linux 7 (Core)
##
## Matrix products: default
## BLAS/LAPACK: /g/easybuild/x86_64/CentOS/7/haswell/software/OpenBLAS/0.3.9-GCC-9.3.0/lib/libopenblas_
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] ggrepel_0.8.2               data.table_1.12.8
##  [3] forcats_0.5.0               stringr_1.4.0
##  [5] dplyr_0.8.5                 purrr_0.3.4
##  [7] readr_1.3.1                 tidyr_1.0.2
##  [9] tibble_3.0.1                ggplot2_3.3.0
## [11] tidyverse_1.3.0             IHW_1.16.0
```

```
## [13] DEWSeq_1.2.0                BiocParallel_1.22.0
## [15] DESeq2_1.28.1               SummarizedExperiment_1.18.2
## [17] DelayedArray_0.14.0         matrixStats_0.56.0
## [19] Biobase_2.48.0              GenomicRanges_1.40.0
## [21] GenomeInfoDb_1.24.2         IRanges_2.22.2
## [23] S4Vectors_0.26.1            BiocGenerics_0.34.0
## [25] R.utils_2.9.2               R.oo_1.23.0
## [27] R.methodsS3_1.8.0
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-147           fs_1.4.1               bitops_1.0-6
##  [4] lubridate_1.7.8        bit64_0.9-7            RColorBrewer_1.1-2
##  [7] httr_1.4.1             tools_4.0.0            backports_1.1.6
## [10] utf8_1.1.4             R6_2.4.1               DBI_1.1.0
## [13] colorspace_1.4-1       withr_2.2.0            tidyselect_1.0.0
## [16] bit_1.1-15.2           compiler_4.0.0         fdrtool_1.2.15
## [19] cli_2.0.2              rvest_0.3.5            xml2_1.3.2
## [22] labeling_0.3           slam_0.1-47            scales_1.1.0
## [25] genefilter_1.70.0      digest_0.6.25          rmarkdown_2.6
## [28] XVector_0.28.0         pkgconfig_2.0.3        htmltools_0.4.0
## [31] lpsymphony_1.16.0      dbplyr_1.4.3           rlang_0.4.5
## [34] readxl_1.3.1           rstudioapi_0.11        RSQLite_2.2.0
## [37] farver_2.0.3           generics_0.0.2         jsonlite_1.6.1
## [40] RCurl_1.98-1.2         magrittr_1.5           GenomeInfoDbData_1.2.3
## [43] Matrix_1.2-18          fansi_0.4.1            Rcpp_1.0.4.6
## [46] munsell_0.5.0          lifecycle_0.2.0        stringi_1.4.6
## [49] yaml_2.2.1             zlibbioc_1.34.0        grid_4.0.0
## [52] blob_1.2.1             crayon_1.3.4           lattice_0.20-41
## [55] haven_2.2.0            splines_4.0.0          annotate_1.66.0
## [58] hms_0.5.3              locfit_1.5-9.4         knitr_1.28
## [61] pillar_1.4.3           geneplotter_1.66.0     reprex_0.3.0
## [64] XML_3.99-0.3           glue_1.4.0             evaluate_0.14
## [67] modelr_0.1.6           vctrs_0.2.4            cellranger_1.1.0
## [70] gtable_0.3.0           assertthat_0.2.1       xfun_0.21
## [73] xtable_1.8-4           broom_0.5.6            survival_3.1-12
## [76] AnnotationDbi_1.50.1   memoise_1.1.0          ellipsis_0.3.0
```