ChIP-seq Data Analysis

Leila Taher

WS 2022/2023

Contents

1 Transcription Factor POU5F1

The ultimate aim of this tutorial is to compare the binding profiles of the transcription factor POU5F1 in H1-hESC and K562 cells (see Table 1). We will use ChIP-seq data generated by the ENCODE consortium. In addition to the ChIP experiments performed in duplicates, ENCODE prepared a matching input control experiment.

- The BAM files in the "bam" folder contain the resulting 36bp-long reads mapped to the human reference genome assembly (hg19). To reduce the time and computational resources needed to analyze the data, we will only work with the reads from chromosome 12.

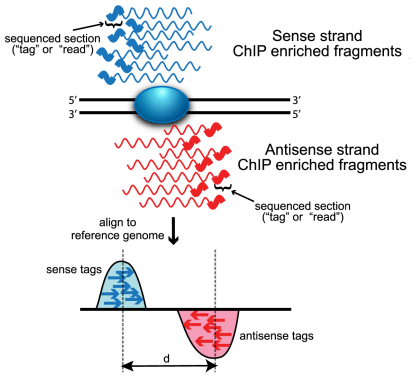
Sample	Identifier	Source
H1-hESC	ENCFF181MED	`https://www.encodeproject.org/experiments/ENCSR000BMU/`
H1-hESC	ENCFF696NWL	`https://www.encodeproject.org/experiments/ENCSR000BMU/`
K562	ENCFF032JWA	`https://www.encodeproject.org/experiments/ENCSR364SNE/`
K562	ENCFF252WJR	`https://www.encodeproject.org/experiments/ENCSR364SNE/`
Control h1-hESC	ENCFF356DXM	`https://www.encodeproject.org/experiments/ENCSR000BHL/`
Control K562	ENCFF335IPE	`https://www.encodeproject.org/experiments/ENCSR080EOT/`

Datasets analyzed in this tutorial.

2 Peak calling with MACS2

definition 2.1 (Peak calling)**. Peak calling** is a computational method used to identify areas in the genome that have been enriched with aligned reads as a consequence of performing a ChIP-sequencing experiment.

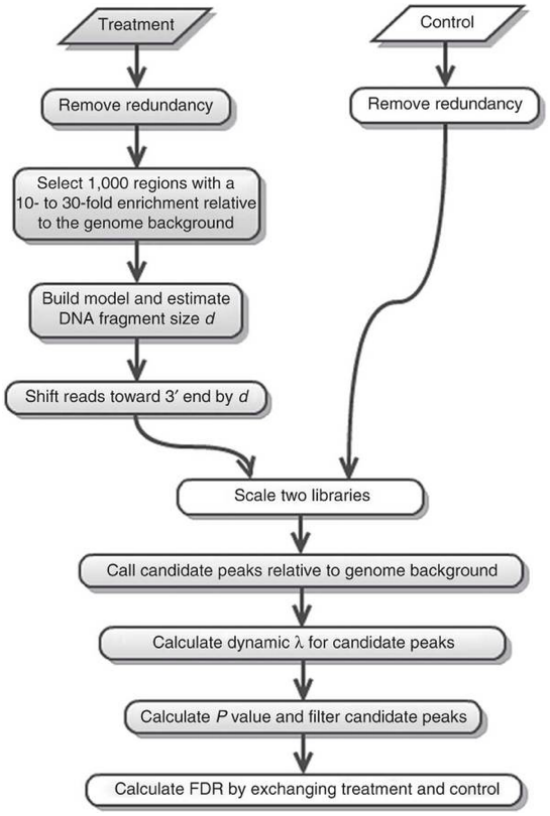
The sequencing reads from ChIP-seq experiments are expected to be centered around the location of the protein of interest. The 5' ends of the fragments that are associated with the protein will normally form two groups: one on the forward strand (sense strand) and one on the reverse strand (antisense strand). The distributions of these groups can be assessed statistically and compared against the background (input or mock IP samples).



Wilbanks and Faccioti, PLoS One, 2010

There are various tools available for peak calling, commonly referred to as *peak callers*. One of the most frequently used ones is the "Model-based Analysis of ChIP-seq (MACS)" python tool, which we will demonstrate here.

Our dataset is investigating a transcription factor and so our focus is on identifying short degenerate sequences that present as punctate binding sites. ChIP-seq analysis algorithms are specialized in identifying one of two types of enrichment (or have specific methods for each): **broad peaks** or broad domains (i.e. histone modifications that cover entire gene bodies) or **narrow peaks** (i.e. a transcription factor binding). MACS performs several steps:



Wilbanks and Faccioti, PLoS One, 2010

i. **Redundancy removal**: By default, MACS keeps a single read at each genomic location. The "`auto`" option tells MACS to calculate the maximum tags at the exact same location based on binomial distribution using 1×10^{-5} as the P-value cutoff. The `all` option keeps every tag. If an integer is specified, then at most that many tags will be kept at the same location.

ii. **Model building**: MACS first scans the whole dataset searching for highly significant enriched regions in the sample. Given a sonication size (*bandwidth*) and a high-confidence fold-enrichment (*mfold*), MACS slides two bandwidth windows across the genome to find regions with enriched with *mfold*

more tags relative to a random tag genome distribution. MACS randomly samples 1,000 of these high-quality peaks, separates their positive and negative strand tags, and aligns them by the midpoint between their centers. The distance between the modes of the two peaks in the alignment is defined as "d" and represents the estimated fragment length. MACS shifts all the tags by $d/2$ toward the 3' ends to the most likely protein-DNA interaction sites.

iii. **Library scaling**: MACS linearly scales the total control tag count to be the same as the total ChIP tag count. The default behavior is for the larger sample to be scaled down.

iv. **Peak detection**: After MACS shifts every tag by $d/2$, it then slides across the genome using a window size of $2d$ to find candidate peaks. The number of tags arising from a particular genomic region X, can be modeled by a Poisson random variable:

$$P(X = k) = \frac{\lambda^k}{k!\, e^{-\lambda}}.$$

The parameter λ is also the expected number of tags. MACS uses a different λ for each genomic region. λ is estimated from the control sample and is deduced by taking the maximum value across various window sizes:

$$\lambda_{\text{local}} = max(\lambda_{BG}, \lambda_{1k}, \lambda_{5k}, \lambda_{10k}).$$

Thus, λ captures the influence of local biases, and is robust against occasional low tag counts at small local regions.

v. **Estimation of the FDR**: Each peak is considered an independent test and thus we need to correct for multiple comparisons.

There are seven major functions available in MACS2 serving as sub-commands. Here we will only cover `callpeak`, but you can use "`macs2 SUBCOMMAND -h`" to find out more, if you are interested. For additional information, you can also visit `https://github.com/taoliu/MACS/wiki/`.
`callpeak` is the main function in MACS2 and can be invoked by typing "`macs2 callpeak`". If you type this command without parameters, you will see a full description of command line options.

Here is a shorter list of the commonly used ones:

i. Input:

- `-t`: IP data file(required).
- `-c`: control or mock data file.
- `-f`: format of the input file; by default, MACS will decide on the format automatically.
- `-g`: *mappable* genome size.

definition 2.2 (Mappability)**. Mappability** is a measure of uniqueness of a specific sequence in the reference genome, and depends on the length of the sequence and the number of mismatches allowed. If $F_k(x)$ is the frequency at which the k-mer sequence at position x is observed in the genome and its reverse complement, the mappability of this position is defined as $M_k(x) = \frac{1}{F_k(x)}$.

ii. Output:

- `-outdir`: output folder.
- `-n`: prefix string for output files.
- `-B/-bdg`: store the fragment pileup, control lambda, -log10pvalue and -log10qvalue scores in bedGraph (`https://genome.ucsc.edu/goldenPath/help/bedgraph.html`) files.

iii. Shifting model:

- `-s`: size of sequencing *tags* (reads); by default MACS will use the first 10 reads in the treatment input file to determine it.
- `-bw`: bandwidth which is used to scan the genome (only for model building); can be set to the expected sonication fragment size.

iv. Peak calling:

- `-q`: Q-value (minimum FDR) cutoff
- `-p`: P-value cutoff (instead of Q-value cutoff)
- `-nolambda`: do not consider the local bias/lambda at peak candidate regions
- `-broad`: broad peak calling NOTE: Relaxing the Q-value does not behave as you would expect in this case since it is partially tied to peak widths. Ideally, if you relaxed the thresholds, you would simply get more peaks but with MACS2 relaxing thresholds also results in wider peaks.

1. Install MACS2

```
pip3 install MACS2
```

2. Create a folder `macs2/results/` for the output generated from MACS2.

3. Run MACS2 as follows

```
macs2 callpeak -t < CHIP_FN.bam > \
        -c < CONTROL_FN.bam > \
        -f BAM -g 2.8e9 \
        -n < OUT_FN.bam >
```

MACS2 is quite verbose so you should see lines of text being printed to the terminal, describing each step that is being carried out. Run MACS2 for all samples.

2.1 MACS2 Output files

4. Examine MACS2's output files:

There should be six output files for each sample:

- `_peaks.narrowPeak`: A BED 6+4 formatted file that contains the peak locations together with peak summit.
  The BED (Browser Extensible Data) format consists of one line per feature, each containing 3-12 fields. The first three fields in each line are required:

```
chr1  213941196  213942363
chr2  158364697  158365864
chr3  127479365  127480532
chr3  127480532  127481699
```

  First 6 columns in `_peaks.narrowPeak` are standard for BED formatted files:

  1 the name of the chromosome (or contig, scaffold, etc.)
  2 start coordinate (the first base in a chromosome is numbered 0)
  3 end coordinate
  4 name given to a region. Use "." if no name is assigned
  5 score of the peak.
  6 strand or orientation. +, - or "." if no orientation is assigned.

  The remaining columns in `_peaks.narrowPeak` are narrowPeak specific fields:

7 `signalValue`: overall enrichment for the region.
8 `pValue`: statistical significance ($-log_{10}$).
9 `qValue`: false discovery rate ($-log_{10}$).
10 `peak`: Point-source called for this peak; 0-based offset from chromStart.

- `_peaks.xls`: a tabular file which contains information about called peaks. Additional information includes pileup and fold enrichment.
- `_summits.bed`: peak summits locations for every peak. To find the motifs at the binding sites, this file is recommended.
- `_model.R`: an R script to visualize the model.

5. How many peaks were called in each sample?

6. Start `R` and run the `_model.R` scripts from the command line. The script should produce a `.pdf` file in your output directory. Open the file.
   - The first plot illustrates the distance between the modes from which the shift was determined.
   - The second plot is the cross-correlation plot.

3 Differential *binding* analysis with `R`

The R/Bioconductor `DiffBind` package works primarily with *peaksets*. For each sample, there is an associated peakset consisting of a set of genomic intervals representing candidate protein binding sites.

- Each peak consists of a chromosome, a start and end position, and usually a score indicating the *strength* of the peak.

Processing ChIP-seq data with `DiffBind` involves several steps:

(i) **Reading the sample peaksets and computing an *occupancy score* for each peak**. For each sample, the occupancy score of a peak is calculated by dividing its score by the score of the peak with the maximum score in the corresponding peakset.

(ii) **Constructing a consensus peakset and assigning scores to the corresponding consensus peaks.** Overlapping peaks in two or more samples are merged into *consensus peaks*. In the consensus peaks, the peaks in the original peaksets have been extended to encompass overlapping peaks in other peaksets. By default, only peaks in two or more samples are considered for the consensus peakset. Each peak in the consensus peakset is assigned a score in each sample. For a given sample:
   - If the consensus peak overlaps with only one peak in the peakset of the sample, it is assigned its occupancy score.
   - If the consensus peak overlaps with two or more peaks in the peakset of the sample, it is assigned the maximum of their occupancy scores.
   - If the consensus peak does not overlap with any peak in the peakset of the sample, it is assigned a score of -1.

(iii) **Quantifying enrichment for each consensus peak and each sample.** For each consensus peak and sample, `DiffBind` computes a normalized read count or *affinity score*. The peaks in the consensus peakset may be re-centered and trimmed based on their summits (point of greatest read overlap) in order to provide more standardized peak intervals. The result is a *binding affinity matrix*.

(iv) **Analyzing differential *binding*.** The *differential binding analysis* identifies statistical differences between two sample groups. By default, this is executed using the `DESeq` R/Bioconductor package.

7. If necessary, install and load the `DiffBind` R/Bioconductor package.

3.1 Consensus peakset (Occupancy Analysis)

Peaksets, especially those generated by peak callers, provide an insight into the potential occupancy of the protein being ChIPed for at specific genomic loci. After the peaksets have been loaded, it can be useful to perform some exploratory plotting to determine how these occupancy maps agree with each other, e.g. between experimental replicates (re-doing the ChIP under the same conditions), between different peak callers on the same experiment, and within groups of samples representing a common experimental condition. DiffBind provides functions to enable overlaps to be examined, as well as functions to determine how well similar samples cluster together. Beyond quality control, the product of an occupancy analysis may be a consensus peakset, representing an overall set of candidate binding sites to be used in further analysis.

8. Generate a comma-separated value (.csv) metafile `samples.csv` containing following columns, and one line for each sample:

   `SampleID,Condition,Replicate,bamReads,ControlID,bamControl,Peaks,PeakCaller`

   - SampleID ... sample identifier (Tab.1)
   - Condition ... Cell type
   - Replicate ... 1 or 2
   - bamReads ... path to bam file
   - ControlID ... corresponding control identifier (Tab.1)
   - bamReads ... path to control bam file
   - Peaks ... path to output .xls file
   - PeakCaller ... macs

The actual peaksets can be read using the `dba()` function of the `DiffBind` package:

```
peaksets <- "samples.csv"
encode <- dba(sampleSheet=peaksets)
```

The result is a `DBA object`. Type "`encode`" to retrieve the *metadata* associated with this object.

9. How many peaksets have you read?
   How many peaks are contained in each peakset?
   How many consensus peaks have been computed?

- If you are curious, the individual peaksets are stored in a list:

```
encode$peaks
```

- The coordinates of the peaks in the consensus peakset are stored in a table:

```
encode$merged
```

10. Based on the consensus peaks and their scores (here, $-\log_{10}$ of the MACS2 P-values), generate a heatmap displaying the Pearson correlation coefficients between pairs of samples:

```
dba.plotHeatmap(encode,margin=15)
```

- Do the samples cluster as expected?

3.2 Counting reads - Binding affinity matrix

Once a consensus peakset has been derived, DiffBind can use the supplied sequence read files (`.bam`) to count how many reads overlap each interval for each unique sample. The result of this is a binding affinity matrix containing a (normalized) read count for each sample at every potential binding site. With this matrix, the samples can be re-clustered using affinity, rather than occupancy, data. The binding affinity matrix is used for QC plotting as well as for subsequent differential analysis. It can be calculated using the `dba.count()` function. The "`score`" option of `dba.count()` specifies how the read counts are to be normalized. The default value for "`score`" is "`DBA_SCORE_TMM_MINUS_FULL`" (see Table 2).

score value	affinity score
DBA_SCORE_READS	raw read count from ChIP
DBA_SCORE_READS_FOLD	raw read count from ChIP divided by raw read count for control
DBA_SCORE_READS_MINUS	raw read count from ChIP minus raw read count for control
DBA_SCORE_RPKM	RPKM from ChIP
DBA_SCORE_RPKM_FOLD	RPKM from ChIP divided by RPKM from control
DBA_SCORE_TMM_READS_FULL	TMM normalized (using edgeR), using ChIP read counts and Full Library size
DBA_SCORE_TMM_READS_EFFECTIVE	TMM-normalized reads from ChIP, using the effective library size (computed by edgeR)
DBA_SCORE_TMM_MINUS_FULL	TMM-normalized reads from ChIP minus control, using the effective library size (computed by edgeR)
DBA_SCORE_TMM_MINUS_EFFECTIVE	TMM normalized (using edgeR), using ChIP read counts minus Control read counts and Effective Library size
DBA_SCORE_TMM_READS_FULL_CPM	same as DBA_SCORE_TMM_READS_FULL, but reported as counts per million.
DBA_SCORE_TMM_READS_EFFECTIVE_CPM	same as DBA_SCORE_TMM_READS_EFFECTIVE, but reported as counts per million.
DBA_SCORE_TMM_MINUS_FULL_CPM	same as DBA_SCORE_TMM_MINUS_FULL, but reported as counts per million.
DBA_SCORE_TMM_MINUS_EFFECTIVE_CPM	same as DBA_SCORE_TMM_MINUS_EFFECTIVE, but reported as counts per million.

Possible values for the "`score`" parameter of `dba.count()`. RPKM: Reads per Kilo base Per Million mapped reads; TMM: Trimmed Mean of M-values.

Typically, the peaks obtained when using an antibody against a transcription factor are relatively *narrow*.

- The "`summits`" option re-centers the consensus peaks around their point of greatest enrichment and adjust the width of the peaks to a specific value (e.g., 500 bp: 250 bp upstream and 250 bp downstream of the summit).
  - Note that this is not advisable for *broad signals*.

```
encode <- dba.count(encode, summits=250)
encode
```

The `dba.count()` function also computes the "Fraction of Reads in Peaks" (FRiP).

11. What is the FRiP? How do you interpret this result?
12. Based on the binding affinities of the consensus peaks, generate a heatmap displaying the Pearson correlation coefficients between pairs of samples:

```
dba.plotHeatmap(encode,margin=15)
```

- Do the samples cluster as expected? Are there any differences compared to the analysis you performed in the previous section?

13. A principal component analysis (PCA) based on the affinity scores of all consensus peaks can be obtained with the `dba.plotPCA` function:

```
dba.plotPCA(encode, label=DBA_CONDITION)
```

- What do you conclude from the PCA?

3.3 Differential binding affinity analysis

Differential binding analysis enables binding sites to be identified that are statistically significantly differentially bound between sample groups. To accomplish this, first a contrast is established, dividing the samples into groups to be compared. Next, a p-value and FDR to each candidate binding site is assigned.

To specify the sample groups you can use the `dba.contrast()` function:

```
encode <- dba.contrast(encode, categories=DBA_CONDITION, minMembers=2)

## Computing results names...
```

The "`categories`" option of the `dba.contrast()` function can take the following values:

- DBA_ID
- DBA_TISSUE
- DBA_FACTOR
- DBA_CONDITION
- DBA_TREATMENT
- DBA_REPLICATE
- DBA_CALLER

`minMembers` specifies the minimum number of replicates in each sample group required by `DiffBind`. By default, it is 3.
The suffix of the chosen value should be a column in "samples.csv". "`categories=DBA_CONDITION`" implies that the values in the "Condition" column will be used to separate the samples into groups.

14. How many sample groups do you expect? Which samples will be in each group?

    Verify your answer by typing:

    ```
    dba.show(encode, bContrasts=TRUE)
    ```

    The differential analysis is executed by the `dba.analyze()` function:

    ```
    encode <- dba.analyze(encode)
    encode
    ```

    - 121 (out of 143) peaks are identified as being significantly differentially bound (DB) using the default FDR threshold of 0.05.

15. Generate a heatmap based on the DB peaks:

```
dba.plotHeatmap(encode,margin=15,contrast=1)
```

16. A PCA based on the affinity scores of the DB peaks can be obtained as follows:

```
dba.plotPCA(encode, contrast=1, label=DBA_CONDITION)
```

- What do you conclude from the heatmap and the PCA?

You can use the `dba.report()` function to retrieve the differentially bound (DB) peaks:

```
encode.DB <- dba.report(encode, contrast=1)
encode.DB %>% as.data.frame() %>% dim #number of DB sites

## [1] 113  11
```

`encode.DB` is a `GRanges` object:

- Conc: mean read *concentration* over all the samples (the default calculation uses $\log_2$ normalized ChIP read counts with control read counts subtracted).
- Conc_hESC: mean concentration in the first group.
- Conc_K562: mean concentration over the second group.
- Fold: Difference in the mean concentrations of group 1 and 2.
- The final two columns give confidence measures for identifying these sites as differentially bound (raw *P*-value and false discovery rate).

17. Convert the object `encode.DB` into a data frame and print to a file.

18. How many of the consensus peaks are more strongly bound by POU5F1 in hESC cells? How many of the consensus peaks are more strongly bound by POU5F1 in K562 cells?

19. Create a BED file for each set of differentially bound peaks. We will write these regions to file and use as input for downstream visualization.

```
# Create bed files for each keeping only significant peaks (p < 0.05)
enrich <- encode.DB %>% as.data.frame() %>%
        select(FDR,seqnames, start, end) %>%
        filter(FDR < 0.05) %>%
        select(seqnames, start, end)

# Write to file
write.table(enrich, file="enriched.bed", sep="\t", quote=F,
        row.names=F, col.names=F)
```

20. To visualize our results in igv, first we need to generate index files with samtools. For each sample, generate a corresponding index file with

    `samtools index IN.bam OUT.bai`

21. Load the BED file containing the differentially bound peaks as well as one mapping bam file for each cell line and the controls into igv (`https://software.broadinstitute.org/software/igv/download`). Zoom into the first DB region in the genome: `chr12:11,761,870-11,762,370`

- Can you make sense of what you see when comparing to the `encode.DB` Granges object?

- Explore other positions on Chr12 according to the bed file

4 Version information R and associated packages

```
macs2 -version
macs2 2.2.7.1
```

```
sessionInfo()

## R version 4.0.5 (2021-03-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.2 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_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] forcats_0.5.2               stringr_1.4.1
##  [3] dplyr_1.0.10                purrr_0.3.5
##  [5] readr_2.1.3                 tidyr_1.2.1
##  [7] tibble_3.1.8                ggplot2_3.4.0
##  [9] tidyverse_1.3.2             DiffBind_3.0.15
## [11] SummarizedExperiment_1.20.0 Biobase_2.50.0
## [13] MatrixGenerics_1.2.1        matrixStats_0.62.0
## [15] GenomicRanges_1.42.0        GenomeInfoDb_1.26.7
## [17] IRanges_2.24.1              S4Vectors_0.28.1
## [19] BiocGenerics_0.36.1         knitr_1.40
##
## loaded via a namespace (and not attached):
##   [1] readxl_1.4.1             backports_1.4.1          GOstats_2.56.0
##   [4] BiocFileCache_1.14.0     plyr_1.8.7               GSEABase_1.52.1
##   [7] splines_4.0.5            BiocParallel_1.24.1      amap_0.8-19
##  [10] digest_0.6.30            invgamma_1.1             GO.db_3.12.1
##  [13] SQUAREM_2021.1           fansi_1.0.3              magrittr_2.0.3
##  [16] checkmate_2.1.0          memoise_2.0.1            BSgenome_1.58.0
##  [19] base64url_1.4            googlesheets4_1.0.1      tzdb_0.3.0
##  [22] limma_3.46.0             Biostrings_2.58.0        annotate_1.68.0
##  [25] modelr_0.1.9             systemPipeR_1.24.6       timechange_0.1.1
##  [28] askpass_1.1              bdsmatrix_1.3-6          prettyunits_1.1.1
##  [31] jpeg_0.1-9               colorspace_2.0-3         rvest_1.0.3
##  [34] blob_1.2.3               rappdirs_0.3.3           apeglm_1.12.0
```

```
##  [37] ggrepel_0.9.2            haven_2.5.1              xfun_0.34
##  [40] crayon_1.5.2             RCurl_1.98-1.9           jsonlite_1.8.3
##  [43] graph_1.68.0             genefilter_1.72.1        brew_1.0-8
##  [46] survival_3.4-0           VariantAnnotation_1.36.0 glue_1.6.2
##  [49] gargle_1.2.1             gtable_0.3.1             zlibbioc_1.36.0
##  [52] XVector_0.30.0           DelayedArray_0.16.3      V8_4.2.2
##  [55] Rgraphviz_2.34.0         scales_1.2.1             pheatmap_1.0.12
##  [58] mvtnorm_1.1-3            DBI_1.1.3                edgeR_3.32.1
##  [61] Rcpp_1.0.9               xtable_1.8-4             progress_1.2.2
##  [64] emdbook_1.3.12           bit_4.0.4                rsvg_2.3.2
##  [67] AnnotationForge_1.32.0   truncnorm_1.0-8          httr_1.4.4
##  [70] gplots_3.1.3             RColorBrewer_1.1-3       ellipsis_0.3.2
##  [73] pkgconfig_2.0.3          XML_3.99-0.12            dbplyr_2.2.1
##  [76] deldir_1.0-6             locfit_1.5-9.4           utf8_1.2.2
##  [79] tidyselect_1.2.0         rlang_1.0.6              AnnotationDbi_1.52.0
##  [82] cellranger_1.1.0         munsell_0.5.0            tools_4.0.5
##  [85] cachem_1.0.6             cli_3.4.1                generics_0.1.3
##  [88] RSQLite_2.2.18           broom_1.0.1              evaluate_0.18
##  [91] fastmap_1.1.0            yaml_2.3.6               fs_1.5.2
##  [94] bit64_4.0.5              caTools_1.18.2           RBGL_1.66.0
##  [97] xml2_1.3.3               biomaRt_2.46.3           compiler_4.0.5
## [100] curl_4.3.3               png_0.1-7                reprex_2.0.2
## [103] geneplotter_1.68.0       stringi_1.7.8            highr_0.9
## [106] GenomicFeatures_1.42.3   lattice_0.20-45          Matrix_1.5-1
## [109] vctrs_0.5.0              pillar_1.8.1             lifecycle_1.0.3
## [112] irlba_2.3.5.1            data.table_1.14.4        bitops_1.0-7
## [115] rtracklayer_1.50.0       R6_2.5.1                 latticeExtra_0.6-30
## [118] hwriter_1.3.2.1          ShortRead_1.48.0         KernSmooth_2.23-20
## [121] MASS_7.3-58.1            gtools_3.9.3             assertthat_0.2.1
## [124] DESeq2_1.30.1            openssl_2.0.4            Category_2.56.0
## [127] rjson_0.2.21             withr_2.5.0              GenomicAlignments_1.26.0
## [130] batchtools_0.9.15        Rsamtools_2.6.0          GenomeInfoDbData_1.2.4
## [133] hms_1.1.2                grid_4.0.5               DOT_0.1
## [136] coda_0.19-4              googledrive_2.0.0        GreyListChIP_1.22.0
## [139] ashr_2.2-54              mixsqp_0.3-43            bbmle_1.0.25
## [142] lubridate_1.9.0          numDeriv_2016.8-1.1      interp_1.1-3
```