TFBSfindR: Variant analysis of Transcription Factor Binding Sites

Douwe J. Spaanderman [1]

[1]Division of Gene Regulation, de Wit E. Group, the Netherlands Cancer Institute

25 June 2019

Abstract

TFBSfindR investigates changes between Transcription Factor (TFs) binding sites in single nucleotide polymorphisms. TFBSfindR uses several methods to calculate changes in TF motifs, first calculating Position Weight Matrixes scores for both reference and variant. Secondly, two methods can be assessed for the identification of important variants, suggested by Kumasaka et al. 2019 and Touzet et al. 2007, TFBSfindR also comes with some plot functions for single variant score analysis.

Contents

1 Introduction . . . 3
2 Using TFBSfindR: A quick overview . . . 3
3 In depth overview . . . 4
3.1 Step 1| Object initialization . . . 4
3.2 Step 2| Motif selection . . . 5
3.3 Step 3| Motif analysis . . . 6
4 Data analysis . . . 7
5 Readable dataframe . . . 8
6 Further filtering | Preliminary from here . . . 9
7 Variant plot . . . 10
8 SessionInfo . . . 10

1 Introduction

This document offers an introduction for using TFBSfindR for the identification of Transcription Factor (TF) binding sites in a variant dataset. Using surrounding reference seqeunce, input transcription factor motifs are compared for their likeness score to the reference and variant sequence. This Motif score is calculate by the use of Position Weight Matrixes (PWMs), using the sum of this log probability matrix. Position Weight Matrixes are widely used for representing Transcription Factor binding motifs and are computed using either Position Probability Matrixes (PPM) or Position Frequency Matrixes (PFM). All three of these Matrixes can be retrieved from well known TF Motif databases like Hocomoco !!LINK!! and Jaspar !!LINK!!. TFBSfindR works with all three, calculating PWM using log probability as suggested by Hocomoco !!LINK!!.

Following PWM score calculations, TFBSfindR can be used for analysing motifs based on two additional scores. First kumusaka et al., suggested calculating motif binding based on the prior probability of a transcription factor binding. Secondly, TFBSfindR uses p-value calculation by Touzet et al. 2007, in order to discover pvalue differences in the reference and variant.

Lastly, in the last segment we will discus method of identifying important variants from either the Kumusaka score as well as with the p-value calculations.

https://almob.biomedcentral.com/articles/10.1186/1748-7188-2-15

2 Using TFBSfindR: A quick overview

Here TFBSfindR is run in its most simplified way. Analysing the example variant dataset provided with TFBSfindR.

```
library(TFBSfindR)
data <- system.file("extdata", "variant.dataset.fasta", package = "TFBSfindR")
data <- read.input.file(input = data, ref.genome = BSgenome.Hsapiens.UCSC.hg19)
data <- TFBS.findR(data, motiflist = MotifDb)
```

3 In depth overview

3.1 Step 1| Object initialization

Here we look more into detail in TFBSfindR and it's customizable functions. First we select the example variant dataset provided with TFBSfindR. Either a fasta file or a vcf file should be provided for TFBSfindR in order to read the format.

```
library(TFBSfindR)
data <- system.file("extdata", "variant.dataset.fasta", package = "TFBSfindR")
```

Next, we select the reference genome, we want to compare the variant data to.

```
library(BSgenome.Hsapiens.UCSC.hg19)
ref.genome <- BSgenome.Hsapiens.UCSC.hg19
```

Finally, we give our sample a name, which can be anything, and read the input file and output a GRangesobject. ranges.filter can be used to filter variants in GRangesobject of there presence in these ranges peaks. In order to do these filter steps, ranges.filter needs to be a string with the location in a BED file, which consists of variants present in ATAC data. For now, we set ranges.filter to FALSE as it is set as default. Additionally, single or multiple chromosomes can be exclusively be selected for using chr.filter, which is set to FALSE as default. read.input.file outputs a GRangesobject with sample.name, rs number, allel, reference and alternative nucleotide and their sequences including 20 nucleotides before and after the variant, which is for the longest TF motif.

```
sample.name <- "example.dataset"
data <- read.input.file(input = data, ref.genome = ref.genome,
    sample.name = sample.name, ranges.filter = FALSE, chr.filter = FALSE)
head(data)
## GRanges object with 6 ranges and 7 metadata columns:
##               seqnames              ranges strand |          Sample
##                  <Rle>           <IRanges>  <Rle> |     <character>
##    rs60216355     chr1 [11046517, 11046558]      + | example.dataset
##    rs58092391     chr1 [11046539, 11046580]      + | example.dataset
##   rs113663169     chr1 [11046544, 11046585]      + | example.dataset
##   rs112732333     chr1 [11046576, 11046617]      + | example.dataset
##    rs72868197     chr1 [11046634, 11046675]      + | example.dataset
##    rs60216355     chr1 [11046517, 11046558]      - | example.dataset
##                       SNP       Allel            REF             ALT
##               <character> <character> <DNAStringSet> <DNAStringSet>
##    rs60216355  rs60216355         *|*              T               C
##    rs58092391  rs58092391         *|*              A               G
##   rs113663169 rs113663169         *|*              T               C
##   rs112732333 rs112732333         *|*              G               A
##    rs72868197  rs72868197         *|*              T               A
##    rs60216355  rs60216355         *|*              T               C
##                          REF.sequence           ALT.sequence
##                         <DNAStringSet>         <DNAStringSet>
##    rs60216355 CGTGTTAGCC...CCTCGTGATC CGTGTTAGCC...CCTCGTGATC
##    rs58092391 ATCTCCTGAC...CCTCCCAAAG ATCTCCTGAC...CCTCCCAAAG
```

```
##   rs113663169 CTGACCTCGT...CAAAGTGCTG CTGACCTCGT...CAAAGTGCTG
##   rs112732333 AAAGTGCTGG...CGCCCGGTCA AAAGTGCTGG...CGCCCGGTCA
##    rs72868197 ATAGTTGGAA...AGCCCCAGCA ATAGTTGGAA...AGCCCCAGCA
##    rs60216355 GCACAATCGG...GGAGCACTAG GCACAATCGG...GGAGCACTAG
##   -------
##   seqinfo: 93 sequences (1 circular) from hg19 genome
```

3.2 Step 2| Motif selection

In order to analyse the variant dataset we have to select motifs to compare our dataset to. A usefull library is MotifDb, which consists of several motif databases. Here we have selected only human motifs provided by the JASPARCORE database. This database consists of 66 well known Transcription factor motifs.

```
library(MotifDb)
JASPARCORE <- query(MotifDb, "JASPAR_CORE")
JASPARCORE <- query(JASPARCORE, "hsapiens")
JASPARCORE
## MotifDb object of length 66
## | Created from downloaded public sources: 2013-Aug-30
## | 66 position frequency matrices from 1 source:
## |        JASPAR_CORE:   66
## | 1 organism/s
## |           Hsapiens:   66
## Hsapiens-JASPAR_CORE-TFAP2A-MA0003.1
## Hsapiens-JASPAR_CORE-NR2F1-MA0017.1
## Hsapiens-JASPAR_CORE-E2F1-MA0024.1
## Hsapiens-JASPAR_CORE-NFIL3-MA0025.1
## Hsapiens-JASPAR_CORE-ELK1-MA0028.1
## ...
## Hsapiens-JASPAR_CORE-SPI1-MA0080.2
## Hsapiens-JASPAR_CORE-AP1-MA0099.2
## Hsapiens-JASPAR_CORE-SP1-MA0079.2
## Hsapiens-JASPAR_CORE-ESR2-MA0258.1
## Hsapiens-JASPAR_CORE-HIF1A::ARNT-MA0259.1
```

Additionally in our library we provide the hocomoco core position count matrix in text format. This is an example, on how to provide your own motif database, which is currently only compatible in .txt file.

```
motifs <- system.file("extdata", "hocomoco.core.txt", package = "TFBSfindR")
motifs <- read.motif.database(motifs)
motifs
## List of length 401
## names(401): >AHR_HUMAN.H11MO.0.B ... >ZSC31_HUMAN.H11MO.0.C
```

3.3 Step 3| Motif analysis

The motifs from JASPARCORE are compared to our variants provided earlier. In the JASPARCORE dataset, motifs are provided as a Position Probability Matrix (PPM). The initial step of TFBS.findR is transforming the motif matrix provided to a Position Weight Matrix. A Position Frequency Matrix (PFM), Position Probability Matrix and ofcourse Position Weight Matrix (PWM) can be provided. the PFM is a frequency matrix with the nucleotide identified on set position on the TF read. Here for all we visualize the motif for TF GATA3, derived from Jaspar.

$$PFM = \begin{matrix} A \\ C \\ G \\ T \end{matrix} \begin{bmatrix} 15 & 4 & 41 & 36 & 7 & 19 & 3 \\ 11 & 35 & 1 & 2 & 29 & 14 & 22 \\ 10 & 2 & 1 & 4 & 6 & 7 & 15 \\ 7 & 2 & 0 & 1 & 1 & 3 & 3 \end{bmatrix}$$

A PPM is a probability matrix rather than a frequency matrix as follows:

$$PPM = \begin{matrix} A \\ C \\ G \\ T \end{matrix} \begin{bmatrix} 0.35 & 0.09 & 0.95 & 0.84 & 0.16 & 0.44 & 0.07 \\ 0.26 & 0.81 & 0.02 & 0.05 & 0.67 & 0.33 & 0.51 \\ 0.23 & 0.05 & 0.02 & 0.09 & 0.14 & 0.16 & 0.35 \\ 0.16 & 0.05 & 0.00 & 0.02 & 0.02 & 0.07 & 0.07 \end{bmatrix}$$

The position weight matrix is calculated by using the following equation:

$$PWM = \frac{log((PFM_{i,j} + pseudocount_{i,j} * background)}{((sequenceCount_{i,j} + pseudocount_{i,j}) * background))}$$

In which sequenceCount is the number of reads used for the identification of our motif: $sequenceCount_{i,j} = \sum_{i=1}^{n}$. The pseudocount is added to eliminate the possibility of infinite values in the equation. In general, smaller pseudocounts have been shown to improve PWM analysis as shown by Nishida et al., 2009, suggesting a pseudocount 0.8 (which is our default). Here we calculate the pseudocount, as done by Papatsenko et al., 2003, using: $pseudocount_{i,j} = log(sequenceCount_{i,j})$. The background is a matrix with the probability for each nucleotide, such as $A = 0.25, C = 0.25, G = 0.25, T = 0.25$.

$$PWM = \begin{matrix} A \\ C \\ G \\ T \end{matrix} \begin{bmatrix} 0.31 & -0.86 & 1.28 & 1.15 & -0.39 & 0.53 & -1.09 \\ 0.02 & 1.12 & -1.78 & -1.38 & 0.94 & 0.25 & 0.67 \\ -0.07 & -1.38 & -1.80 & -0.86 & -0.52 & -0.39 & 0.31 \\ -0.39 & -1.38 & -2.52 & -1.80 & -1.80 & -1.09 & -1.09 \end{bmatrix}$$

Next, the score for a sequence is calculated using the sum of this log probability matrix. This is simply done by comparing set positions to the motif both for the variant and reference using an shifting window approach, calculating a PWM score for every position it will cover the variant in the 41 nucleotide sequence.

$$PWM.score = \sum_{i=1}^{m} M(i, u_i)$$

For both the variant and the reference, motif alterations for all posible positions are calculated.

Depending on the method selected, Kumusaka score and/or Pvalue can be calculated. Providing both as we have done here, will logically provided both these values for the variant.

The Kumasaka score is a posterior probability of a transcription factor binding. The following equation is used for calculating this score:

$$p(TFbinding|sequence) = \frac{\pi PWM_1}{(1-\pi)PWM_0 + \pi PWM_1}$$

In which PWM_1 is the score for the motif given in the sequence for either the variant or reference. PWM_0 is the PWM score with the background probability (i.e. 0.25 for all nucleotides). The prior is set to the False discovery rate, in our case 0.1.

The pvalue is currently computed using an external package TFMPvalue. The pvalue calculation is achieved by first calculating the score distribution. first the probability that the background model can achieve a score equal to α for a specific matrix (M) is defined as $Q(M, \alpha)$. If s is not an accessible score then $Q(M, s) = 0$. Q is calculated using ||WORK IN PROGRESS|| Using this equation, a pvalue can be obtained with the relation:

$$Pvalue(M, \alpha) = \sum_{s \geq \alpha} Q(M, s)$$

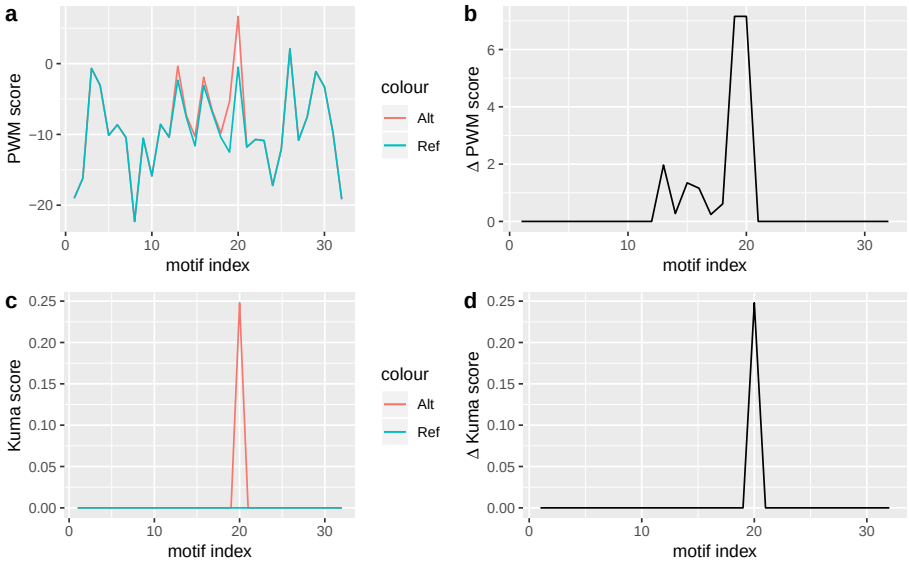
Lastly, in TFBS.findR bbparam can be assessed to determine the number of workers (cores) to use for this process. Note that currently, the script is not optomized for memory usage, spliting on multiple cores will create double required memory capacity.

```
data <- TFBS.findR(data, motiflist = JASPARCORE, motif.type = "PPM",
    method = "both", background = c(A = 0.25, C = 0.25, G = 0.25,
        T = 0.25), pseudocount = "log.of.reads", prior = 0.1,
    BPPARAM = bpparam)
```

4 Data analysis

Here we can visualize a specific variant for the scores on the shifting window. Note that Kuma score is excluded for negative values, as calculations are impossible if PWM < 0. Delta PWM scores are depicted on the right for both the Kuma and PWM score.

```
snp <- data[data$SNP %in% "rs113663169"]
snp.plot(snp, method = "both", motif = "TFAP2A", strand = "+")
```



5 Readable dataframe

Lastly, the GRangesobject provides a score for each position of the sequence and is difficult to interpert. Here, we provide a cleaner way to create a dataframe output, which select the highest position score without further filtering. Note that it is important to provide the same method as provided with TFBS.findR.

```
data <- data.update(data, pseudocount = 0.01, method = "both",
    BPPARAM = bpparam)
head(data)
##   seqnames    start      end width strand          Sample         SNP Allel
## 1     chr1 11046559 11046567     9      + example.dataset  rs58092391   *|*
## 2     chr1 11046563 11046571     9      + example.dataset rs113663169   *|*
## 3     chr1 11046593 11046604    12      - example.dataset rs112732333   *|*
## 4     chr1 11046585 11046596    12      + example.dataset rs112732333   *|*
## 5     chr1 11046536 11046541     6      - example.dataset  rs60216355   *|*
## 6     chr1 11046593 11046597     5      - example.dataset rs112732333   *|*
##   REF ALT Snp.loc     Sequence     MotifDB provider  Motif  Ref.score
## 1   A   G       1    ACCCGTCTC JASPAR_CORE MA0003.1 TFAP2A -3.0793595
## 2   T   C       2    GTCTCAGCC JASPAR_CORE MA0003.1 TFAP2A -0.4810847
## 3   G   A       4 CCGCACTCGGTG JASPAR_CORE MA0137.2  STAT1 -1.2648751
## 4   G   A      12 GGATTACAGGCG JASPAR_CORE MA0137.2  STAT1 -2.1773726
## 5   T   C       2       AACTAG JASPAR_CORE MA0037.1  GATA3 -4.5693202
## 6   G   A       4        CCGCA JASPAR_CORE MA0036.1  GATA2 -3.5403642
##   Alt.score Delta.score Kuma.ref.score Kuma.alt.score Kuma.delta.score
## 1  4.078016    7.157375              0     0.16762631         4.150773
## 2  6.676290    7.157375              0     0.24794691         4.689002
## 3  5.149911    6.414786              0     0.16018430         4.089026
## 4  4.172422    6.349795              0     0.13384979         3.846491
## 5  1.358056    5.927376              0     0.09140201         3.342014
## 6  2.225080    5.765444              0     0.16512554         4.130318
```

```
##   Pvalue.ref.score Pvalue.alt.score
## 1       0.08277130      0.007461548
## 2       0.03458023      0.002155304
## 3       0.01099390      0.001096249
## 4       0.01448137      0.001625180
## 5       0.24584961      0.036621094
## 6       0.20898438      0.022460938
```

6 Further filtering | Preliminary from here

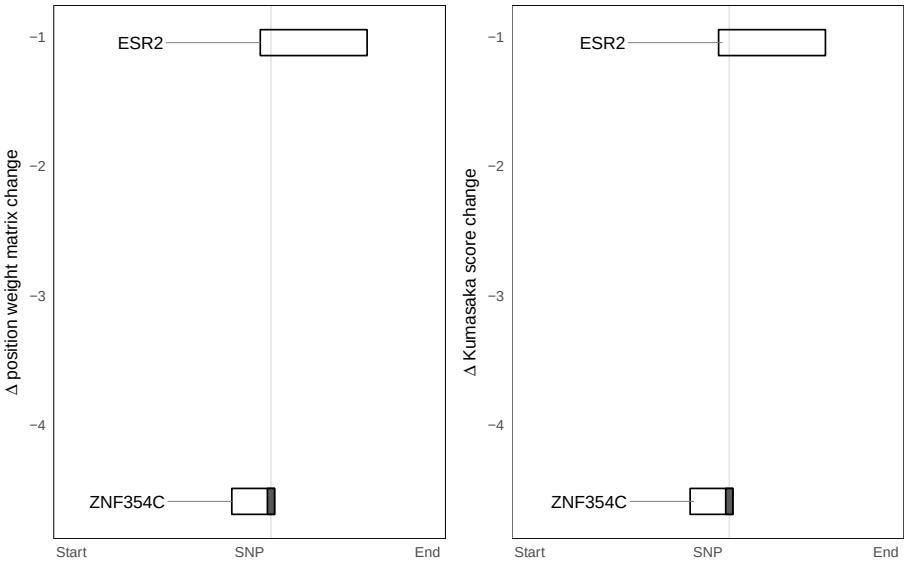
Currently filtering is set to pvalue 0.001, still have to discuss what the best method is for Kuma

```
thres <- 0.001

data <- data[(data$Pvalue.alt.score <= thres & data$Pvalue.ref.score >
    thres) | (data$Pvalue.alt.score > thres & data$Pvalue.ref.score <=
    thres), ]
head(data)
##     seqnames    start      end width strand          Sample         SNP
## 33      chr1 11046650 11046655     6      - example.dataset  rs72868197
## 48      chr1 11046530 11046547    18      - example.dataset  rs60216355
## 85      chr1 11046559 11046578    20      + example.dataset rs113663169
## 87      chr1 11046557 11046576    20      + example.dataset rs113663169
## 148     chr1 11046559 11046573    15      - example.dataset  rs58092391
## 159     chr1 11046537 11046551    15      - example.dataset  rs60216355
##     Allel REF ALT Snp.loc             Sequence     MotifDB provider Motif
## 33    *|*   T   A       5               TCTTAA JASPAR_CORE MA0095.1   YY1
## 48    *|*   T   C       8   TACCAGAACTAGAGGACT JASPAR_CORE MA0051.1  IRF2
## 85    *|*   T   C       6 ACCCGTCTCAGCCTCCCAAA JASPAR_CORE MA0073.1 RREB1
## 87    *|*   T   C       8 CCACCCGTCTCAGCCTCCCA JASPAR_CORE MA0073.1 RREB1
## 148   *|*   A   G       1      TGGGCAGAGTCGGAG JASPAR_CORE MA0258.1  ESR2
## 159   *|*   T   C       1      ACTAGAGGACTGGAG JASPAR_CORE MA0112.2  ESR1
##     Ref.score Alt.score Delta.score Kuma.ref.score Kuma.alt.score
## 33   2.835734  7.479649    4.643915     0.17359086      0.3565193
## 48   1.367128  5.711716    4.344589     0.03265396      0.1235989
## 85   3.483928  5.600379    2.116451     0.07185738      0.1106786
## 87   3.895866  5.940619    2.044754     0.07967679      0.1166185
## 148  6.241427  5.193688   -1.047739     0.15606912      0.1333640
## 159  6.658066  5.176402   -1.481663     0.16477071      0.1329792
##     Kuma.delta.score Pvalue.ref.score Pvalue.alt.score
## 33         0.9973948     0.0166015625     0.0007324219
## 48         1.6471561     0.0065060399     0.0009574574
## 85         0.5599854     0.0025761025     0.0009712388
## 87         0.4976818     0.0021480038     0.0008229893
## 148       -0.2121007     0.0008890601     0.0014314856
## 159       -0.2896580     0.0007016901     0.0014773831
```

7 Variant plot

```
snp <- data[data$SNP == "rs58092391", ]
variant.plot(snp, JASPARCORE)
```



8 SessionInfo

```
sessionInfo()
## R version 3.4.4 (2018-03-15)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.6 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblasp-r0.2.18.so
##
## 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] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
```

```
##  [1] MotifDb_1.20.0                    BSgenome.Hsapiens.UCSC.hg19_1.4.0
##  [3] BSgenome_1.46.0                   rtracklayer_1.38.3
##  [5] Biostrings_2.46.0                 XVector_0.18.0
##  [7] GenomicRanges_1.30.3              GenomeInfoDb_1.14.0
##  [9] IRanges_2.12.0                    S4Vectors_0.16.0
## [11] BiocGenerics_0.24.0               TFBSfindR_0.1.0
## [13] BiocParallel_1.12.0               knitr_1.22
## [15] BiocStyle_2.6.1
##
## loaded via a namespace (and not attached):
##  [1] Biobase_2.38.0             httr_1.4.0
##  [3] RMySQL_0.10.16             bit64_0.9-7
##  [5] assertthat_0.2.1           blob_1.1.1
##  [7] GenomeInfoDbData_1.0.0     Rsamtools_1.30.0
##  [9] yaml_2.2.0                 progress_1.2.0
## [11] ggrepel_0.8.0              pillar_1.3.1
## [13] RSQLite_2.1.1              lattice_0.20-38
## [15] glue_1.3.1                 digest_0.6.18
## [17] colorspace_1.4-1           cowplot_0.9.4
## [19] htmltools_0.3.6            Matrix_1.2-15
## [21] plyr_1.8.4                 XML_3.98-1.17
## [23] pkgconfig_2.0.2            biomaRt_2.34.2
## [25] bookdown_0.9               zlibbioc_1.24.0
## [27] purrr_0.3.2                scales_1.0.0
## [29] tibble_2.1.1               ggplot2_3.1.1
## [31] SummarizedExperiment_1.8.1 GenomicFeatures_1.30.3
## [33] TFMPvalue_0.0.8            lazyeval_0.2.2
## [35] splitstackshape_1.4.6      magrittr_1.5
## [37] crayon_1.3.4               memoise_1.1.0
## [39] evaluate_0.13              data.table_1.12.0
## [41] tools_3.4.4                prettyunits_1.0.2
## [43] hms_0.4.2                  formatR_1.5
## [45] matrixStats_0.54.0         stringr_1.4.0
## [47] munsell_0.5.0              DelayedArray_0.4.1
## [49] AnnotationDbi_1.40.0       compiler_3.4.4
## [51] rlang_0.3.4                grid_3.4.4
## [53] RCurl_1.95-4.11            rstudioapi_0.9.0
## [55] VariantAnnotation_1.24.5   labeling_0.3
## [57] bitops_1.0-6               rmarkdown_1.11
## [59] codetools_0.2-16           gtable_0.3.0
## [61] DBI_1.0.0                  R6_2.4.0
## [63] GenomicAlignments_1.14.2   gridExtra_2.3
## [65] dplyr_0.8.0.1              bit_1.1-14
## [67] stringi_1.4.3              Rcpp_1.0.1
## [69] tidyselect_0.2.5           xfun_0.6
```