DMRs identification with mCSEA package

true true

1 November 2022

Abstract

mCSEA (methylathed CpGs Set Enrichment Analysis) searches Differentially Methylated Regions (DMRs) between conditions using methylation data from Illumina's 450k or EPIC microarrays. The evaluated DMRs are predefined regions (promoters, gene bodies, CpG Islands and user-defined regions). This package contains functions to rank the CpG probes, to apply a GSEA analysis for DMRs identification, to plot the results and to integrate them with expression data.

Previous steps

The input of mCSEA is tipically a matrix with the processed β-values for each probe and sample. If you start from the raw methylation files (like .idat files) you should first preprocess the data with any of the available packages for that purpose (e.g. *minfi* or *ChAMP*). Minfi includes functions to get a matrix of β-values (*getBeta()*) or M-values (*getM()*). ChAMP output class depends on the type of analysis performed. For instance, *champ.norm()* function returns a matrix, while *champ.load()* returns a list of results, and one of them is a β-values matrix. So mCSEA is totally compatible with minfi and ChAMP outputs as long as a matrix with the methylation values is obtained.

Reading .idat files

Here we provide a minimal example to read .idat files with *minfi* package. A samplesheet must be provided in order to read the raw files and generate a *RGChannelSet* object. For more information about this step, read minfi's vignette.

```r
library(minfi)
minfiDataDir <- system.file("extdata", package = "minfiData")
targets <- read.metharray.sheet(minfiDataDir, verbose = FALSE)
RGset <- read.metharray.exp(targets = targets)
```

Cell type heterogeneity correction

Different cell types proportions across samples are one of the major sources of variability in methylation data from tissues like blood or saliva (McGregor et al. (2016)). There are a lot of packages which can be used to estimate cell types proportions in each sample in order to correct for this bias (reviewed in McGregor et al. (2016)). Here we supply an example where blood reference data is used to estimate cell types proportions of each blood sample.

```
library(FlowSorted.Blood.450k)
library(mCSEA)
data(mcseadata)

cellCounts = estimateCellCounts(RGset)
```

```
## Warning in DataFrame(sampleNames = c(colnames(rgSet),
## colnames(referenceRGset)), : 'stringsAsFactors' is ignored
```

```
print(cellCounts)
```

```
##                         CD8T      CD4T        NK     Bcell      Mono       Gran
## 5723646052_R02C02 0.12132355 0.2489442 0.1525943 0.3177123 0.1922554 0.13958941
## 5723646052_R04C01 0.07437663 0.2927509 0.1592739 0.3203660 0.1682549 0.15928208
## 5723646052_R05C02 0.15451979 0.2578973 0.1600397 0.2438732 0.1983064 0.15169740
## 5723646053_R04C02 0.17332656 0.2522868 0.1167347 0.2657515 0.2335970 0.11646712
## 5723646053_R05C02 0.08811529 0.2725395 0.1752246 0.2837298 0.2011335 0.13415227
## 5723646053_R06C02 0.09190789 0.3202139 0.2006286 0.2350504 0.2052394 0.09815978
```

These proportions could be introduced as covariates in the linear models.

Step 1: Ranking CpGs probes

To run a mCSEA analysis, you must rank all the evaluated CpGs probes with some metric (e.g. t-statistic, Fold-Change…). You can use *rankProbes()* function for that aim, or prepare a ranked list with the same structure as the *rankProbes()* output.

We load sample data to show how *rankProbes()* works:

```
library(mCSEA)
data(mcseadata)
```

We loaded to our R environment **betaTest** and **phenoTest** objects, in addition to **exprTest**, annotation objects and association objects (we will talk about these after). **betaTest** is a matrix with the β-values of 10000 EPIC probes for 20 samples. **phenoTest** is a dataframe with the explanatory variable and covariates associated to the samples. When you load your own data, the structure of your objects should be similar.

```
head(betaTest, 3)
```

```
##                    1         2         3         4         5         6
## cg18478105 0.6845279 0.6917252 0.8622046 0.6966168 0.1204777 0.7670960
## cg10605442 0.1370685 0.8450987 0.5480076 0.8671236 0.8300113 0.1667405
## cg27657131 0.1333706 0.6745949 0.8702664 0.9338893 0.8788454 0.1853554
##                     7          8          9        10         11         12
## cg18478105 0.93804510 0.88166619 0.90385504 0.9287976 0.04052779 0.10765614
## cg10605442 0.08727434 0.10568040 0.11896201 0.1764874 0.73534148 0.05741730
## cg27657131 0.10463463 0.05660229 0.06469281 0.2235293 0.92030432 0.04618165
##                   13        14        15         16        17        18
## cg18478105 0.1459481 0.8334884 0.1209040 0.07747453 0.7001099 0.7528026
## cg10605442 0.8213965 0.8208602 0.1671381 0.10157830 0.8874912 0.1723724
## cg27657131 0.1374107 0.8432675 0.9642680 0.14536637 0.9372422 0.9315385
##                    19         20
## cg18478105 0.86687272 0.85999403
## cg10605442 0.88836050 0.06521765
## cg27657131 0.06357636 0.50609450
```

```
print(phenoTest)
```

```
##      expla cov1
## 1     Case    1
## 2     Case    2
## 3     Case    1
## 4     Case    1
## 5     Case    3
## 6     Case    3
## 7     Case    2
## 8     Case    2
## 9     Case    2
## 10    Case    1
## 11 Control    1
## 12 Control    2
## 13 Control    1
## 14 Control    1
## 15 Control    3
## 16 Control    3
## 17 Control    2
## 18 Control    2
## 19 Control    2
## 20 Control    1
```

rankProbes() function uses these two objects as input and apply a linear model with *limma* package. By default, *rankProbes()* considers the first column of the phenotypes table as the explanatory variable in the model (e.g. cases and controls) and does not take into account any covariate to adjust the models. You can change this behaviour modifying **explanatory** and **covariates** options.

By default, *rankProbes()* assumes that the methylation data object contains β-values and transform them to M-values before calculating the linear models. If your methylation data object contains M-values, you must specify it (typeInput = "M"). You can also use β-values for models calculation (typeAnalysis = "beta"), although we do not recommend it due to it has been proven that M-values better accomplish the statistical assumptions of limma analysis (Du et al. (2010)).

```
myRank <- rankProbes(betaTest, phenoTest, refGroup = "Control")
```

```
## Transforming beta-values to M-values

## Calculating linear model...

##  Explanatory variable: expla

##  Case group: Case

##  Reference group: Control

##  Total samples: 20

##  Covariates: None

##  Categorical variables: expla cov1

##  Continuous variables: None
```

myRank is a named vector with the t-values for each CpG probe.

```
head(myRank)
```

```
## cg18478105 cg10605442 cg27657131 cg08514185 cg13587582 cg25802399
##  2.2586016 -0.4230906 -0.8578285 -0.6890975 -3.0001263  0.7390646
```

You can also supply *rankProbes()* function with a SummarizedExperiment object. In that case, if you don't specify a **pheno** object, phenotypes will be extracted from the SummarizedExperiment object with *colData()* function.

Paired analysis

It is possible to take into account paired samples in *rankProbes()* analysis. For that aim, you should use paired = TRUE parameter and specify the column in pheno containing pairing information (pairColumn parameter).

Step 2: Searching DMRs in predefined regions

Once you calculated a score for each CpG, you can perform the mCSEA analysis. For that purpose, you should use *mCSEATest()* function. This function takes as input the vector generated in the previous step, the methylation data and the phenotype information. By default, it searches for differentially methylated promoters, gene bodies and CpG Islands. You can specify the regions you want to test with *regionsTypes* option. *minCpGs* option specifies the minimum amount of CpGs in a region to be considered in the analysis (5 by default). You can increase the number of processors to use with *nproc* option (recommended if you have enough computational resources). Finally, you should specify if the array platform is 450k or EPIC with the *platform* option.

```
myResults <- mCSEATest(myRank, betaTest, phenoTest,
                      regionsTypes = "promoters", platform = "EPIC")
```

```
## Associating CpG sites to promoters

## Analysing promoters

##   |                                                                      |

## 38 DMRs found (padj < 0.05)
```

mCSEATest() returns a list with the GSEA results and the association objects for each region type analyzed, in addition to the input data (methylation, phenotype and platform).

```
ls(myResults)
```

```
## [1] "methData"                "pheno"                   "platform"
## [4] "promoters"               "promoters_association"
```

promoters is a data frame with the following columns (partially extracted from *fgsea* help):

- *pval:* Estimated P-value.
- *padj:* P-value adjusted by BH method.
- *log2err:* Expected error for the standard deviation of the P-value logarithm.
- *ES:* Enrichment score.
- *NES:* Normalized enrichment score by number of CpGs associated to the feature.
- *size:* Number of CpGs associated to the feature.
- *leadingEdge:* Leading edge CpGs which drive the enrichment.

```
head(myResults[["promoters"]][,-7])
```

```
##                pval         padj   log2err         ES       NES size
## DMD    1.983028e-41 9.835820e-39 1.6781779 -0.9649723 -2.271420   65
## BANP   2.610929e-38 6.475105e-36 1.6088796 -0.9668041 -2.242631   59
## KTN1   1.079396e-16 6.692254e-15 1.0574636 -0.9605936 -1.960773   27
## XIAP   5.657210e-16 3.117752e-14 1.0276699 -0.9626065 -1.952226   25
## SEMA3B 2.335239e-15 1.052981e-13 1.0073180 -0.9603008 -1.947550   25
## GOLGB1 1.142063e-13 4.357409e-12 0.9436322 -0.9635271 -1.893988   20
```

On the other hand, **promoters_association** is a list with the CpG probes associated to each feature:

```
head(myResults[["promoters_association"]], 3)
```

```
## $YTHDF1
##  [1] "cg18478105" "cg10605442" "cg27657131" "cg08514185" "cg13587582"
##  [6] "cg25802399" "cg22485414" "cg03501095" "cg24092253" "cg12589387"
##
## $EIF2S3
## [1] "cg09835024" "cg06127902" "cg12275687" "cg00914804" "cg27345735"
## [6] "cg12590845" "cg25034591" "cg16712639" "cg07622257"
##
## $PKN3
## [1] "cg14361672" "cg06550760" "cg14204415" "cg11056832" "cg14036226"
## [6] "cg22365023" "cg20593100"
```

You can also provide a custom association object between CpG probes and regions (*customAnnotation* option). This object should be a list with a structure similar to this:

```
head(assocGenes450k, 3)
```

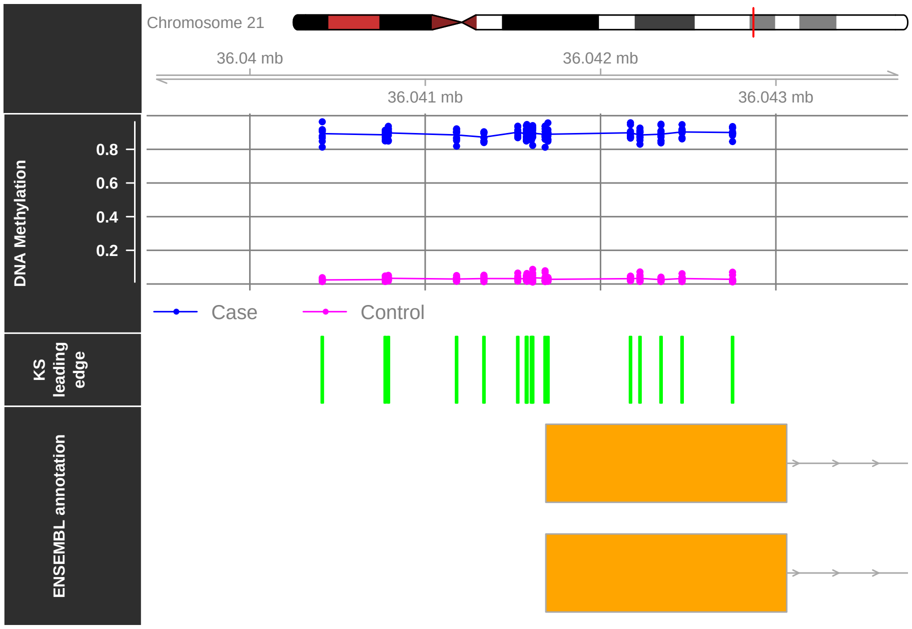
```
## $TSPY4
##  [1] "cg00050873" "cg03443143" "cg04016144" "cg05544622" "cg09350919"
##  [6] "cg15810474" "cg15935877" "cg17834650" "cg17837162" "cg25705492"
## [11] "cg00543493" "cg00903245" "cg01523029" "cg02606988" "cg02802508"
## [16] "cg03535417" "cg04958669" "cg08258654" "cg08635406" "cg10239257"
## [21] "cg13861458" "cg14005657" "cg25538674" "cg26475999"
##
## $TTTY14
## [1] "cg03244189" "cg05230942" "cg10811597" "cg13765957" "cg13845521"
## [6] "cg15281205" "cg26251715"
##
## $NLGN4Y
##  [1] "cg03706273" "cg25518695" "cg01073572" "cg01498999" "cg02340092"
##  [6] "cg03278611" "cg04419680" "cg05939513" "cg07795413" "cg08816194"
## [11] "cg09300505" "cg09748856" "cg09804407" "cg10990737" "cg18113731"
## [16] "cg19244032" "cg27214488" "cg27265812" "cg27443332"
```

Step 3: Plotting the results

Once you found some DMRs, you can make a plot with the genomic context of the interesting ones. For that, you must provide *mCSEAPlot()* function with the *mCSEATest()* results, and you must specify which type of region you want to plot and the name of the DMR to be plotted (e.g. gene name). There are some graphical parameters you can adjust (see *mCSEAPlot()* help). Take into account that this function connects to some online servers in order to get genomic information. For that reason, this function could take some minutes to finish the plot, specially the first time it is executed.

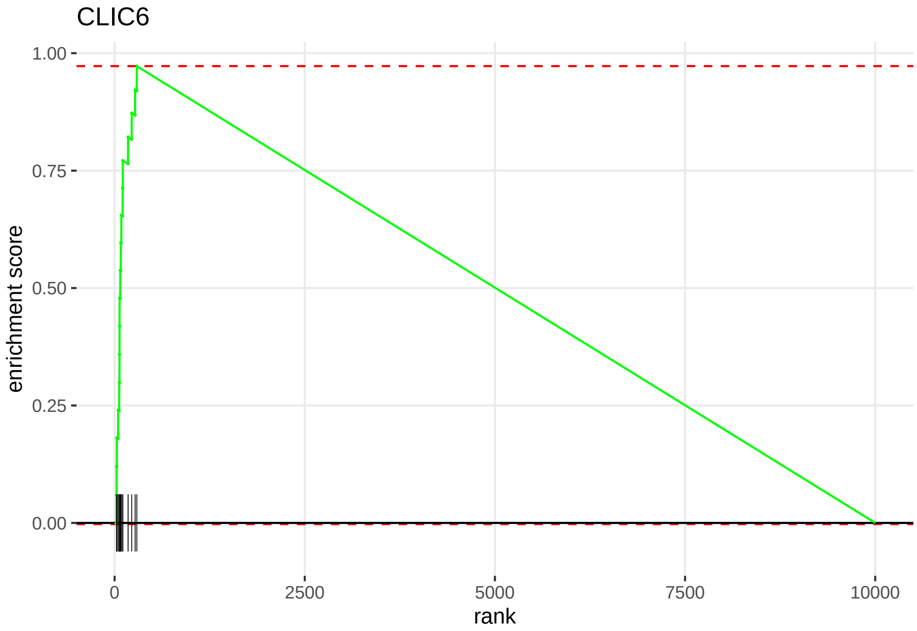
```
mCSEAPlot(myResults, regionType = "promoters",
          dmrName = "CLIC6",
          transcriptAnnotation = "symbol", makePDF = FALSE)
```

```
## Warning: Ensembl will soon enforce the use of https.
## Ensure the 'host' argument includes "https://"
```



You can also plot the GSEA results for a DMR with *mCSEAPlotGSEA()* function.

```
mCSEAPlotGSEA(myRank, myResults, regionType = "promoters", dmrName = "CLIC6")
```



Integrating methylation and expression data

If you have both methylation and expression data for the same samples, you can integrate them in order to discover significant associations between methylation changes in a DMR and an expression alterations in a close gene. *mCSEAIntegrate()* considers the DMRs identified by *mCSEATest()* passing a P-value threshold (0.05 by default). It calculates the mean methylation for each condition using the leading edge CpGs and performs a correlation test between this mean DMR methylation and the expression of close genes. This function automatically finds the genes located within a determined distance (1.5 kb) from the DMR. Only correlations passing thresholds (0.5 for correlation value and 0.05 por P-value by default) are returned. For promoters, only negative correlations are returned due to this kind of relationship between promoters methylation and gene expression has been largely observed (Jones (2012)). On the contrary, only positive correlations between gene bodies methylation and gene expression are returned, due to this is a common relationship observed (Jones (2012)). For CpG islands and custom regions, both positive and negative correlations are returned, due to they can be located in both promoters and gene bodies.

To test this function, we extracted a subset of 100 genes expression from bone marrows of 10 healthy and 10 leukemia patients (**exprTest**). Data was extracted from *leukemiasEset* package.

```
# Explore expression data
head(exprTest, 3)
```

```
##                        1        2        3        4        5        6        7
## ENSG00000179023 4.145748 4.388779 4.265583 4.374576 4.463465 4.078678 4.335878
## ENSG00000179029 4.485414 5.044662 5.411474 5.590093 5.365381 4.951236 6.626413
## ENSG00000179041 6.618769 6.443408 7.642324 7.989362 7.133374 7.224613 5.853054
##                        8        9       10       11       12       13       14
## ENSG00000179023 4.121601 4.163271 4.219654 4.340421 3.917131 4.284802 4.161627
## ENSG00000179029 5.070305 5.582466 5.688895 5.675448 5.053258 5.708689 5.170988
## ENSG00000179041 8.198245 6.847891 6.598557 6.546835 7.211352 7.190893 6.825418
##                       15       16       17       18       19       20
## ENSG00000179023 4.308718 4.074333 4.171878 4.083548 4.549825 4.199466
## ENSG00000179029 5.480265 5.118550 5.657001 5.257061 5.677323 5.171198
## ENSG00000179041 7.342032 7.309422 6.831020 7.728485 7.214401 6.781880
```

```
# Run mCSEAIntegrate function
resultsInt <- mCSEAIntegrate(myResults, exprTest, "promoters", "ENSEMBL")
```

```
## 0 genes removed from exprData due to Standar Deviation = 0
```

```
## Integrating promoters methylation with gene expression
```

```
resultsInt
```

```
##   Feature regionType            Gene Correlation      PValue   adjPValue
## 1   GATA2  promoters ENSG00000179348  -0.8908771 1.39373e-07 1.39373e-07
```

It is very important to specify the correct gene identifiers used in the expression data (*geneIDs* parameter). *mCSEAIntegrate()* automatically generates correlation plots for the significant results and save them in the directory specified by *folder* parameter (current directory by default).

Session info

```
## R version 4.2.1 (2022-06-23)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.5 LTS
## 
## Matrix products: default
## BLAS:   /home/biocbuild/bbs-3.16-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.16-bioc/R/lib/libRlapack.so
## 
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C              
##  [3] LC_TIME=en_GB              LC_COLLATE=C              
##  [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       
## 
```

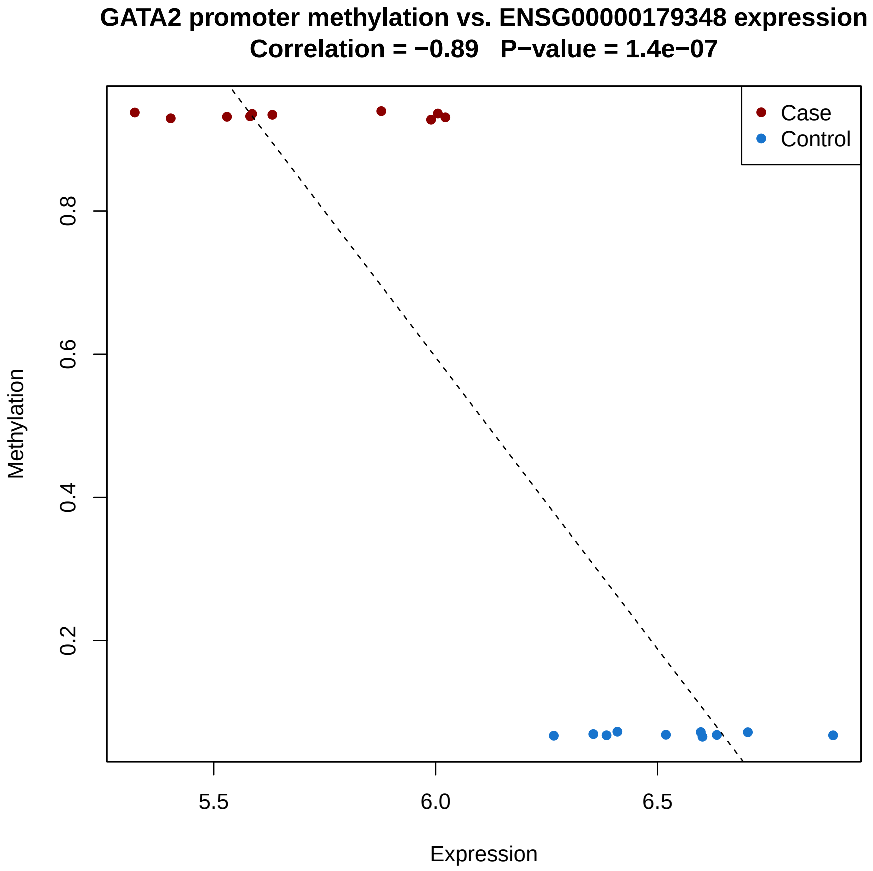


Figure 1: Integration plot for GATA2 promoter methylation and ENSG00000179348 expression. Note that, actually, both names refers to the same gene, but SYMBOL was used to analyze promoters methylation and ENSEMBL ID was used as gene identifiers in the expression data.

```
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] IlluminaHumanMethylation450kanno.ilmn12.hg19_0.6.1
##  [2] IlluminaHumanMethylation450kmanifest_0.4.0
##  [3] mCSEA_1.18.0
##  [4] Homo.sapiens_1.3.1
##  [5] TxDb.Hsapiens.UCSC.hg19.knownGene_3.2.2
##  [6] org.Hs.eg.db_3.16.0
##  [7] GO.db_3.16.0
##  [8] OrganismDbi_1.40.0
##  [9] GenomicFeatures_1.50.0
## [10] AnnotationDbi_1.60.0
## [11] mCSEAdata_1.17.0
## [12] FlowSorted.Blood.450k_1.35.0
## [13] minfi_1.44.0
## [14] bumphunter_1.40.0
## [15] locfit_1.5-9.6
## [16] iterators_1.0.14
## [17] foreach_1.5.2
## [18] Biostrings_2.66.0
## [19] XVector_0.38.0
## [20] SummarizedExperiment_1.28.0
## [21] Biobase_2.58.0
## [22] MatrixGenerics_1.10.0
## [23] matrixStats_0.62.0
## [24] GenomicRanges_1.50.0
## [25] GenomeInfoDb_1.34.0
## [26] IRanges_2.32.0
## [27] S4Vectors_0.36.0
## [28] BiocGenerics_0.44.0
##
## loaded via a namespace (and not attached):
##   [1] utf8_1.2.2                tidyselect_1.2.0
##   [3] RSQLite_2.2.18            htmlwidgets_1.5.4
##   [5] grid_4.2.1                BiocParallel_1.32.0
##   [7] munsell_0.5.0             codetools_0.2-18
##   [9] preprocessCore_1.60.0     interp_1.1-3
##  [11] withr_2.5.0               colorspace_2.0-3
##  [13] filelock_1.0.2            highr_0.9
##  [15] knitr_1.40                rstudioapi_0.14
##  [17] labeling_0.4.2            GenomeInfoDbData_1.2.9
##  [19] farver_2.1.1              bit64_4.0.5
##  [21] rhdf5_2.42.0              vctrs_0.5.0
##  [23] generics_0.1.3           xfun_0.34
##  [25] biovizBase_1.46.0        BiocFileCache_2.6.0
##  [27] R6_2.5.1                 illuminaio_0.40.0
##  [29] AnnotationFilter_1.22.0  bitops_1.0-7
##  [31] rhdf5filters_1.10.0      cachem_1.0.6
##  [33] reshape_0.8.9            fgsea_1.24.0
##  [35] DelayedArray_0.24.0      assertthat_0.2.1
##  [37] BiocIO_1.8.0             scales_1.2.1
##  [39] nnet_7.3-18              gtable_0.3.1
##  [41] ensembldb_2.22.0         rlang_1.0.6
##  [43] genefilter_1.80.0        splines_4.2.1
##  [45] rtracklayer_1.58.0       lazyeval_0.2.2
##  [47] GEOquery_2.66.0          dichromat_2.0-0.1
##  [49] checkmate_2.1.0          BiocManager_1.30.19
##  [51] yaml_2.3.6               backports_1.4.1
##  [53] Hmisc_4.7-1              RBGL_1.74.0
##  [55] tools_4.2.1              nor1mix_1.3-0
##  [57] ggplot2_3.3.6            ellipsis_0.3.2
##  [59] RColorBrewer_1.1-3       siggenes_1.72.0
##  [61] Rcpp_1.0.9               plyr_1.8.7
##  [63] base64enc_0.1-3          sparseMatrixStats_1.10.0
##  [65] progress_1.2.2           zlibbioc_1.44.0
##  [67] purrr_0.3.5              RCurl_1.98-1.9
##  [69] prettyunits_1.1.1        rpart_4.1.19
##  [71] openssl_2.0.4            deldir_1.0-6
##  [73] cowplot_1.1.1            cluster_2.1.4
##  [75] magrittr_2.0.3           data.table_1.14.4
##  [77] ProtGenerics_1.30.0      hms_1.1.2
##  [79] evaluate_0.17            xtable_1.8-4
##  [81] XML_3.99-0.12            jpeg_0.1-9
##  [83] mclust_6.0.0             gridExtra_2.3
##  [85] compiler_4.2.1           biomaRt_2.54.0
##  [87] tibble_3.1.8             crayon_1.5.2
##  [89] htmltools_0.5.3          tzdb_0.3.0
##  [91] Formula_1.2-4            tidyr_1.2.1
##  [93] DBI_1.1.3                dbplyr_2.2.1
##  [95] MASS_7.3-58.1            rappdirs_0.3.3
##  [97] BiocStyle_2.26.0         Matrix_1.5-1
##  [99] readr_2.1.3              cli_3.4.1
## [101] quadprog_1.5-8           Gviz_1.42.0
## [103] pkgconfig_2.0.3          GenomicAlignments_1.34.0
## [105] foreign_0.8-83           xml2_1.3.3
## [107] annotate_1.76.0          rngtools_1.5.2
## [109] multtest_2.54.0          beanplot_1.3.1
## [111] doRNG_1.8.2              scrime_1.3.5
## [113] VariantAnnotation_1.44.0 stringr_1.4.1
## [115] digest_0.6.30              graph_1.76.0
## [117] rmarkdown_2.17             base64_2.0.1
## [119] fastmatch_1.1-3            htmlTable_2.4.1
## [121] DelayedMatrixStats_1.20.0 restfulr_0.0.15
## [123] curl_4.3.3                 Rsamtools_2.14.0
## [125] rjson_0.2.21               lifecycle_1.0.3
## [127] nlme_3.1-160               Rhdf5lib_1.20.0
## [129] askpass_1.1                limma_3.54.0
## [131] BSgenome_1.66.0            fansi_1.0.3
## [133] pillar_1.8.1               lattice_0.20-45
## [135] KEGGREST_1.38.0            fastmap_1.1.0
## [137] httr_1.4.4                 survival_3.4-0
## [139] glue_1.6.2                 png_0.1-7
## [141] bit_4.0.4                  stringi_1.7.8
## [143] HDF5Array_1.26.0           blob_1.2.3
## [145] latticeExtra_0.6-30        memoise_2.0.1
## [147] dplyr_1.0.10
```

References

Du, P, X Zhang, C Huang, N Jafari, WA Kibbe, L Hou, and SM Lin. 2010. "Comparison of Beta-Value and M-Value Methods for Quantifying Methylation Levels by Microarray Analysis." *BMC Bioinformatics*.

Jones, Peter A. 2012. "Functions of DNA Methylation: Islands, Start Sites, Gene Bodies and Beyond." *Nature Reviews Genetics*.

McGregor, K, S Bernatsky, I Colmegna, T Pastinen, A Labbe, and CMT Greenwood. 2016. "An Evaluation of Methods Correcting for Cell-Type Heterogeneity in DNA Methylation Studies." *Genome Biology*.