msgbsR: an R package to analyse methylation sensitive genotyping by sequencing (MS-GBS) data

Benjamin Mayne

November 8, 2024

Contents

1 Introduction

Current data analysis tools do not fulfil all experimental designs. For example, GBS experiments using methylation sensitive restriction enzymes (REs), which is also known as methylation sensitive genotyping by sequencing (MS-GBS), is an effective method to identify differentially methylated sites that may not be accessible in other technologies such as microarrays and methyl capture sequencing. However, current data analysis tools do not satisfy the requirements for these types of experimental designs.

Here we present msgbsR, an R package for data analysis of MS-GBS experiments. Read counts and cut sites from a MS-GBS experiment can be read directly into the R environment from a sorted and indexed BAM file(s).

2 Reading data into R

The analysis with the msgbsR pipeline begins with a directory which contains sorted and indexed BAM file(s). msgbsR contains an example data set containing 6 samples from a MS-GBS experiment using the restriction enzyme MspI. In this example the 6 samples are from the prostate of a rat and have been truncated for chromosome 20. 3 of the samples were fed a control diet and the other 3 were fed an experimental high fat diet.

To read in the data directly into the R environment can be done using the rawCounts() function, which requires the directory path to where the sorted and indexed files are located and the desired number of threads to be run (Default = 1).

```
> library(msgbsR)
> library(GenomicRanges)
> library(SummarizedExperiment)
> my_path <- system.file("extdata", package = "msgbsR")
> se <- rawCounts(bamFilepath = my_path)
> dim(assay(se))
```

```
[1] 16047     6
```

The result is an RangedSummarizedExperiment object containing the read counts. The columns are samples and the rows contain the location of each unique cut sites. Each cut site has been given a unique ID (chr:position-position:strand). The cut site IDs can be turned into a GRanges object. Information regarding the samples such as treatment or other groups can be added into the return object as shown below

```
> colData(se) <- DataFrame(Group = c(rep("Control", 3), rep("Experimental", 3)),
+                          row.names = colnames(assay(se)))
```

3 Confirmation of correct cut sites

After the data has been generated into the R environment, the next step is to confirm that the cut sites were the correctly generated sites. In this example, the methylated sensitive restriction enzyme that has been used is MspI which recognizes a 4bp sequence (C/CGG). MspI cuts between the two cytosines when the outside cytosine is methylated.

The first step is to extract the location of the cut sites from se and adjust the cut sites such that the region will cover the recognition sequence of MspI. It is important to note that in this example the user must adjust the region over the cut sites specifically for each strand. In other words although the enzyme cuts at C/CGG on the minus strand this would appear as CCG/G. The code below shows how to adjust the postioining of the cut sites to cover the recginition site on each strand.

```
> cutSites <- rowRanges(se)
> # # Adjust the cut sites to overlap recognition site on each strand
> start(cutSites) <- ifelse(test = strand(cutSites) == '+',
+                           yes = start(cutSites) - 1, no = start(cutSites) - 2)
> end(cutSites) <- ifelse(test = strand(cutSites) == '+',
+                           yes = end(cutSites) + 2, no = end(cutSites) + 1)
```

The object cutSites is a GRanges object that contains the start and end position of the MspI sequence length around the cut sites. These cut sites can now be checked if the sequence matches the MspI sequence.

msgbsR offer two approaches to checking the cut sites. The first approach is to use a BSgenome which can be obtained from Bioconductor. In this example, BSgenome.Rnorvegicus.UCSC.rn6 will be used.

```
> library(BSgenome.Rnorvegicus.UCSC.rn6)
> correctCuts <- checkCuts(cutSites = cutSites, genome = "rn6", seq = "CCGG")
```

If a BSgenome is unavailable for a species of interest, another option to checking the cut sites is to use a fasta file which can be used throught the checkCuts() function.

The correctCuts data object is in the format of a GRanges object and contains the correct sites that contained the recognition sequence. These sites can be kept within se by using the subsetByOverlaps function.

The incorrect MspI cut sites can be filtered out of datCounts:

```
> se <- subsetByOverlaps(se, correctCuts)
> dim(assay(se))
```

```
[1] 13983     6
```

se now contains the correct cut sites and can now be used in downstream analyses.

4 Visualization of read counts

Before any further downstream analyses with the data, the user may want to filter out samples that did not generate a sufficient number of read counts or cut sites. The msgbsR package contains a function which plots the total number of read counts against the total number of cut sites produced per sample. The user can also use the function to visulaise if different categories or groups produced varying amount of cut sites or total amount of reads.

To visualize the total number of read counts against the total number of cut sites produced per sample:

```
> plotCounts(se = se, cateogory = "Group")
```

This function generates a plot (Figure 1) where the x axis and y axis represents the total number of reads and the total number of cut sites produced for each sample respectively.

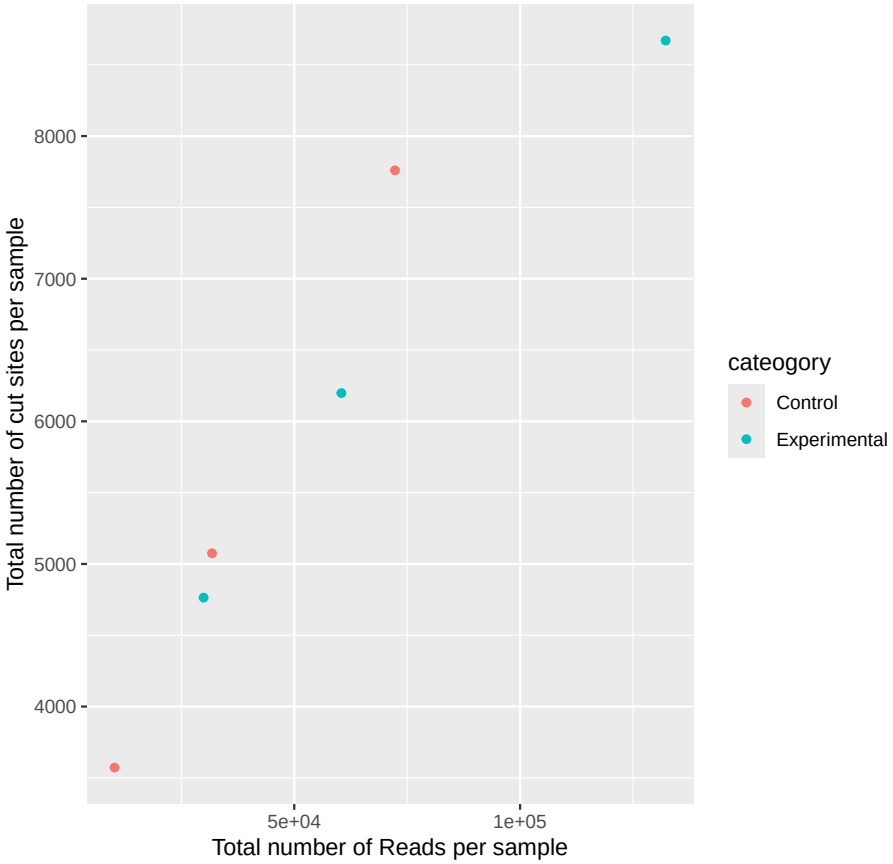

Figure 1: The distribution of the total number of reads and cut sites produced by each sample.

5 Differential methylation analysis

msgbsR utilizes edgeR in order to determine which cut sites are differentially methylated between groups. Since MS-GBS experiments can have multiple groups or conditions msgbsR offers a wrapper function of edgeR (Zhou et al., 2014) tools to automate differential methylation analyses.

To determine which cut sites are differentiallly methylated between groups:

```
> top <- diffMeth(se = se, cateogory = "Group",
+                 condition1 = "Control", condition2 = "Experimental",
+                 cpmThreshold = 1, thresholdSamples = 1)
```

The top object now contains a data frame of the cut sites that had a CPM $>$ 1 in at least 1 sample and which cut sites are differentially methylated between the two groups.

6 Visualization of cut site locations

The msgbsR package contains a function to allow visualization of the location of the cut sites. Given the lengths of the chromosomes the cut sites can be visualized in a circos plot (Figure 2).

Firstly, define the length of the chromosome.

```
> ratChr <- seqlengths(BSgenome.Rnorvegicus.UCSC.rn6)["chr20"]
```

Extract the differentially methylated cut sites.

```
> my_cuts <- GRanges(top$site[which(top$FDR < 0.05)])
```

To generate a circos plot:

```
> plotCircos(cutSites = my_cuts, seqlengths = ratChr,
+            cutSite.colour = "red", seqlengths.colour = "blue")
```

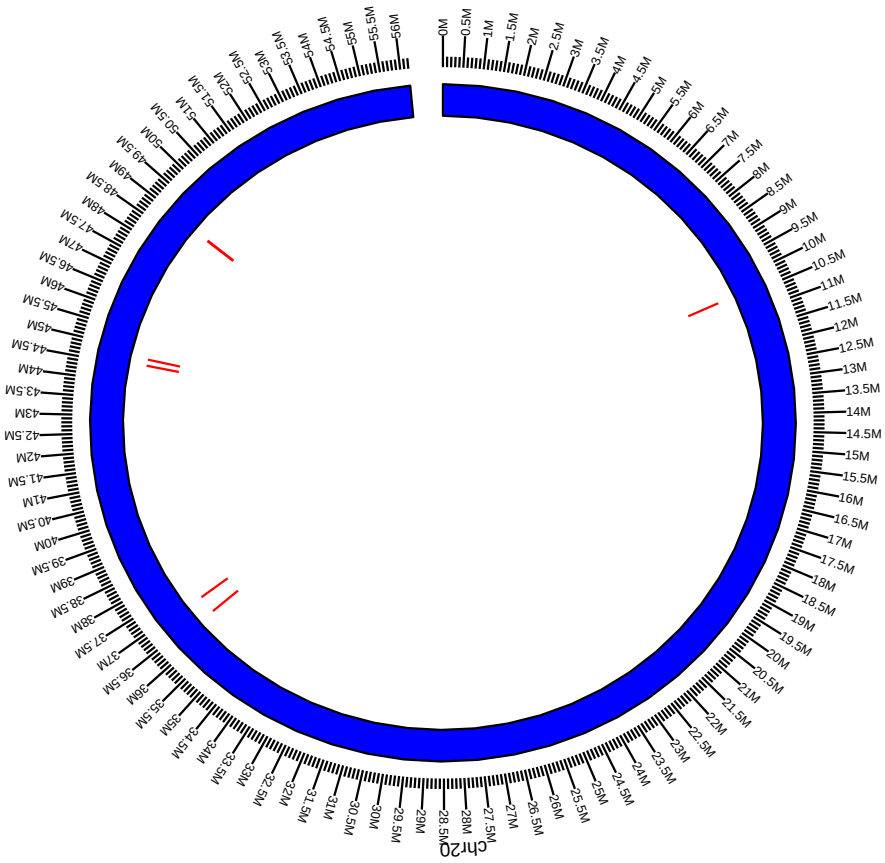

Figure 2: A circos plot of chromosome 20 representing cut sites defined by the user.

7 Session Information

This analysis was conducted on:

```
> sessionInfo()

R version 4.4.1 (2024-06-14)
Platform: aarch64-apple-darwin20
Running under: macOS Ventura 13.6.7

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/New_York
tzcode source: internal

attached base packages:
[1] stats4    stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] BSgenome.Rnorvegicus.UCSC.rn6_1.4.1 BSgenome_1.74.0
 [3] rtracklayer_1.66.0                  BiocIO_1.16.0
 [5] Biostrings_2.74.0                   XVector_0.46.0
 [7] SummarizedExperiment_1.36.0         Biobase_2.66.0
 [9] MatrixGenerics_1.18.0               matrixStats_1.4.1
[11] msgbsR_1.30.0                       GenomicRanges_1.58.0
[13] GenomeInfoDb_1.42.0                 IRanges_2.40.0
[15] S4Vectors_0.44.0                    BiocGenerics_0.52.0

loaded via a namespace (and not attached):
  [1] bitops_1.0-9             filelock_1.0.3           tibble_3.2.1
  [4] R.oo_1.27.0              graph_1.84.0             XML_3.99-0.17
  [7] rpart_4.1.23             lifecycle_1.0.4          httr2_1.0.6
 [10] pwalign_1.2.0            edgeR_4.4.0              lattice_0.22-6
 [13] ensembldb_2.30.0         OrganismDbi_1.48.0       backports_1.5.0
 [16] magrittr_2.0.3           limma_3.62.1             Hmisc_5.2-0
 [19] rmarkdown_2.29           yaml_2.3.10              ggbio_1.54.0
 [22] DBI_1.2.3                RColorBrewer_1.1-3       abind_1.4-8
 [25] ShortRead_1.64.0         zlibbioc_1.52.0          purrr_1.0.2
 [28] R.utils_2.12.3           AnnotationFilter_1.30.0  biovizBase_1.54.0
 [31] RCurl_1.98-1.16          nnet_7.3-19              VariantAnnotation_1.52.0
 [34] rappdirs_0.3.3           GenomeInfoDbData_1.2.13  codetools_0.2-20
```

```
 [37] DelayedArray_0.32.0      xml2_1.3.6               tidyselect_1.2.1
 [40] UCSC.utils_1.2.0         farver_2.1.2             BiocFileCache_2.14.0
 [43] base64enc_0.1-3          GenomicAlignments_1.42.0 jsonlite_1.8.9
 [46] Formula_1.2-5            tools_4.4.1              progress_1.2.3
 [49] Rcpp_1.0.13-1            glue_1.8.0               gridExtra_2.3
 [52] SparseArray_1.6.0        xfun_0.49                dplyr_1.1.4
 [55] genomeIntervals_1.62.0   withr_3.0.2              BiocManager_1.30.25
 [58] fastmap_1.2.0            GGally_2.2.1             latticeExtra_0.6-30
 [61] fansi_1.0.6              digest_0.6.37            R6_2.5.1
 [64] colorspace_2.1-1         jpeg_0.1-10              dichromat_2.0-0.1
 [67] biomaRt_2.62.0           RSQLite_2.3.7            LSD_4.1-0
 [70] R.methodsS3_1.8.2        utf8_1.2.4               tidyr_1.3.1
 [73] generics_0.1.3           intervals_0.15.5         data.table_1.16.2
 [76] prettyunits_1.2.0        httr_1.4.7               htmlwidgets_1.6.4
 [79] S4Arrays_1.6.0           ggstats_0.7.0            pkgconfig_2.0.3
 [82] gtable_0.3.6             blob_1.2.4               hwriter_1.3.2.1
 [85] htmltools_0.5.8.1        RBGL_1.82.0              ProtGenerics_1.38.0
 [88] scales_1.3.0             png_0.1-8                knitr_1.49
 [91] rstudioapi_0.17.1        reshape2_1.4.4           rjson_0.2.23
 [94] checkmate_2.3.2          curl_6.0.0               cachem_1.1.0
 [97] stringr_1.5.1            parallel_4.4.1           foreign_0.8-87
[100] AnnotationDbi_1.68.0     restfulr_0.0.15          pillar_1.9.0
[103] grid_4.4.1               vctrs_0.6.5              easyRNASeq_2.42.0
[106] dbplyr_2.5.0             cluster_2.1.6            htmlTable_2.4.3
[109] evaluate_1.0.1           GenomicFeatures_1.58.0   cli_3.6.3
[112] locfit_1.5-9.10          compiler_4.4.1           Rsamtools_2.22.0
[115] rlang_1.1.4              crayon_1.5.3             labeling_0.4.3
[118] interp_1.1-6             plyr_1.8.9               stringi_1.8.4
[121] deldir_2.0-4             BiocParallel_1.40.0      txdbmaker_1.2.0
[124] munsell_0.5.1            lazyeval_0.2.2           Matrix_1.7-1
[127] hms_1.1.3                bit64_4.5.2              ggplot2_3.5.1
[130] KEGGREST_1.46.0          statmod_1.5.0            memoise_2.0.1
[133] bit_4.5.0
```

8 References

Zhou X, Lindsay H, Robinson MD (2014). Robustly detecting differential expression in RNA sequencing data using observation weights. Nucleic Acids Research, 42(11), e91.