segmentSeq: methods for detecting methylation loci and differential methylation

Thomas J. Hardcastle

October 26, 2021

1 Introduction

This vignette introduces analysis methods for data from high-throughput sequencing of bisulphite treated DNA to detect cytosine methylation. The `segmentSeq` package was originally designed to detect siRNA loci [1] and many of the methods developed for this can be used to detect loci of cytosine methylation from replicated (or unreplicated) sequencing data.

2 Preparation

Preparation of the segmentSeq package proceeds as in siRNA analysis. We begin by loading the `segmentSeq` package.

```
>   library(segmentSeq)
```

Note that because the experiments that `segmentSeq` is designed to analyse are usually massive, we should use (if possible) parallel processing as implemented by the `parallel` package. If using this approach, we need to begin by define a *cluster*. The following command will use eight processors on a single machine; see the help page for 'makeCluster' for more information. If we don't want to parallelise, we can proceed anyway with a `NULL` cluster. Results may be slightly different depending on whether or not a cluster is used owing to the non-deterministic elements of the method.

```
> if(require("parallel"))
+ {
+     numCores <- min(8, detectCores())
+     cl <- makeCluster(numCores)
+ } else {
+     cl <- NULL
+ }
```

The `segmentSeq` package is designed to read in output from the YAMA (Yet Another Methylome Aligner) program. This is a perl-based package using either bowtie or bowtie2 to align bisulphite treated reads (in an unbiased manner) to a reference and identify the number of times each cytosine is identified as methylated or unmethylated. Unlike most other aligners, YAMA does not require that reads that map to more than one location are discarded, instead it reports the number of alternate matches to the reference for each cytosine. This is then used by `segmentSeq` to weight the observed number of methylated/un-methylated cytosines at a location. The files used here have been compressed to save space.

```
> datadir <- system.file("extdata", package = "segmentSeq")
> files <- c("short_18B_C24_C24_trim.fastq_CG_methCalls.gz",
+ "short_Sample_17A_trimmed.fastq_CG_methCalls.gz",
+ "short_13_C24_col_trim.fastq_CG_methCalls.gz",
+ "short_Sample_28_trimmed.fastq_CG_methCalls.gz")
> mD <- readMeths(files = files, dir = datadir,
+ libnames = c("A1", "A2", "B1", "B2"), replicates = c("A","A","B","B"),
+ nonconversion = c(0.004777, 0.005903, 0.016514, 0.006134))
```

We can begin by plotting the distribution of methylation for these samples. The distribution can be plotted for each sample individually, or as an average across multiple samples. We can also subtract one distribution from another to visualise patterns of differential methylation on the genome.

```
> par(mfrow = c(2,1))
> dists <- plotMethDistribution(mD, main = "Distributions of methylation", chr = "Chr1")
> plotMethDistribution(mD,
+                      subtract = rowMeans(sapply(dists, function(x) x[,2])),
+                      main = "Differences between distributions", chr = "Chr1")
```

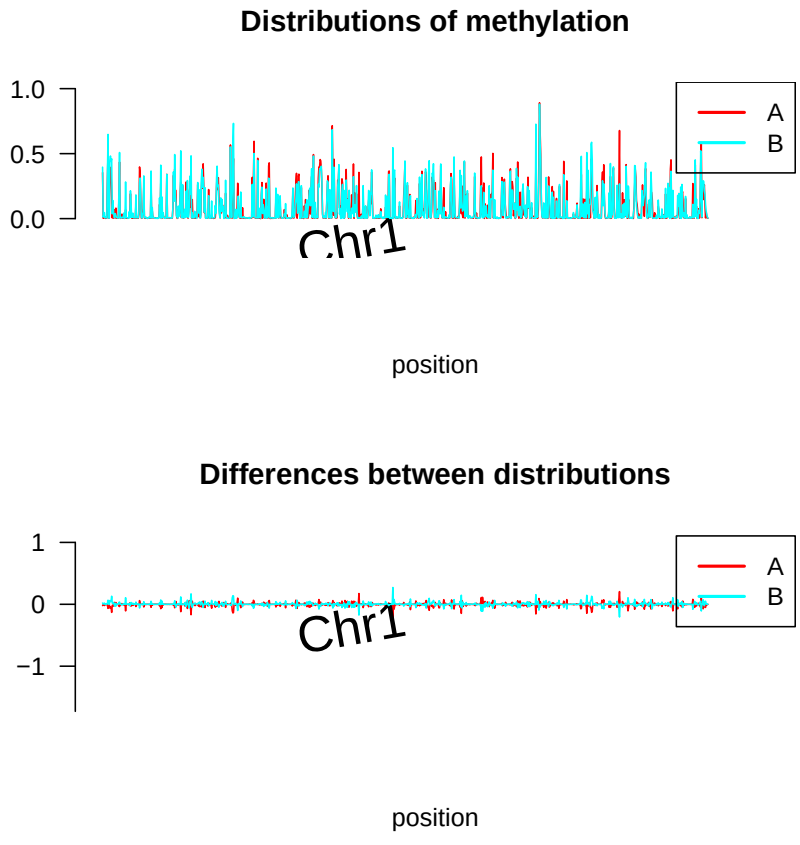

Figure 1: Distributions of methylation on the genome (first two million bases of chromosome 1

Next, we process this `alignmentData` object to produce a `segData` object. This `segData` object contains a set of potential segments on the genome defined by the start and end points of regions of overlapping alignments in the `alignmentData` object. It then evaluates the number of tags that hit in each of these segments.

```
> sD <- processAD(mD, cl = cl)
```

We can now construct a segment map from these potential segments.

Segmentation by heuristic Bayesian methods

A fast method of segmentation can be achieved by assuming a binomial distribution on the data with an uninformative beta prior, and identifying those potential segments which have a sufficiently large posterior likelihood that the proportion of methylation exceeds some critical value. This value can be determined by examining the data using the 'thresholdFinder' function, but expert knowledge is likely to provide a better guess as to where methylation becomes biologically significant.

```
> thresh = 0.2
> hS <- heuristicSeg(sD = sD, aD = mD, prop = thresh, cl = cl, gap = 100, getLikes = FALSE)
> hS
```

```
GRanges object with 2955 ranges and 0 metadata columns:
         seqnames          ranges strand
            <Rle>       <IRanges>  <Rle>
     [1]     Chr1         108-948      *
     [2]     Chr1       5449-5465      *
     [3]     Chr1            6452      *
     [4]     Chr1            6520      *
     [5]     Chr1       7020-7034      *
     ...      ...             ...    ...
  [2951]     Chr1 1993118-1993128      *
  [2952]     Chr1         1993149      *
  [2953]     Chr1         1993335      *
  [2954]     Chr1         1994611      *
  [2955]     Chr1 1994857-1994886      *
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
An object of class "lociData"
2955 rows and 4 columns

Slot "replicates"
A A B B
Slot "groups":
list()

Slot "data":
     [,1]    [,2]    [,3]    [,4]
[1,] 247:169 175:133 165:149 28:31
[2,] 0:49    1:86    23:36   2:2
[3,] 2:0     5:3     0:3     0:1
[4,] 0:3     0:9     2:1     0:0
```

```
[5,] 0:119   2:78    20:37   0:2
2950 more rows...

Slot "annotation":
data frame with 0 columns and 2955 rows

Slot "locLikelihoods" (stored on log scale):
Matrix with  2955  rows.
       A   B
1      1   1
2      0   1
3      1   0
4      0   1
5      0   1
...  ... ...
2951   1   1
2952   1   1
2953   1   0
2954   1   0
2955   0   1

Expected number of loci in each replicate group
   A    B
1968 2168
```

Within a methylation locus, it is not uncommon to find completely unmethylated cytosines. If the coverage of these cytosines is too high, it is possible that these will cause the locus to be split into two or more fragments. The `mergeMethSegs` function can be used to overcome this splitting by merging loci with identical patterns of expression that are not separated by too great a gap. Merging in this manner is optional, but recommended.

```
> hS <- mergeMethSegs(hS, mD, gap = 5000, cl = cl)
```

We can then estimate posterior likelihoods on the defined loci by applying empirical Bayesian methods. These will not change the locus definition, but will assign likelihoods that the identified loci represent a true methylation locus in each replicate group.

```
> hSL <- lociLikelihoods(hS, mD, cl = cl)
```

Visualising loci

By one of these methods, we finally acquire an annotated `methData` object, with the annotations describing the co-ordinates of each segment.

We can use this `methData` object, in combination with the `alignmentMeth` object, to plot the segmented genome.

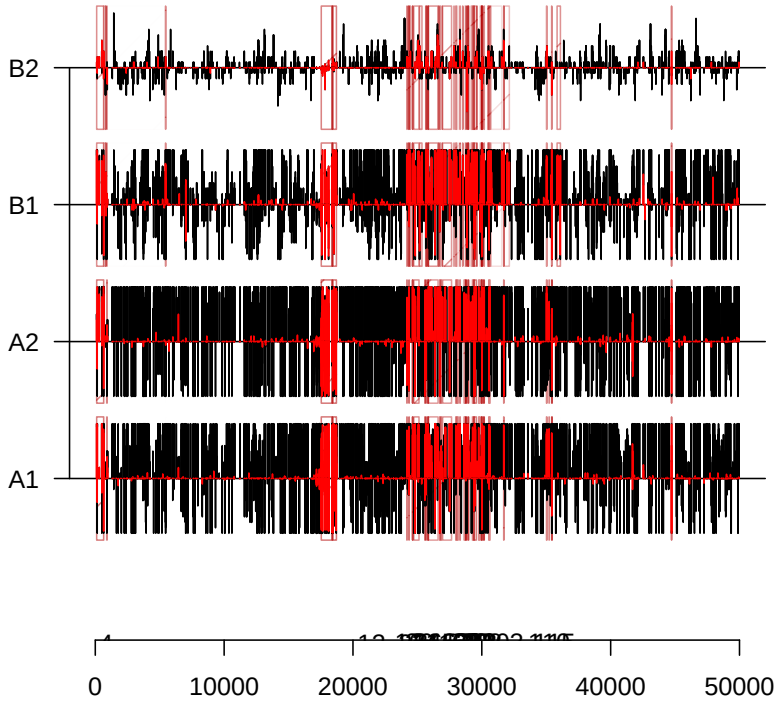

Figure 2: Methylation and identified loci on the first ten thousand bases of chromosome 1

```
> plotMeth(mD, hSL, chr = "Chr1", limits = c(1, 50000), cap = 10)
```

Differential Methylation analysis

We can also examine the `methData` object for differentially methylated regions using the beta-binomial methods [2] implemented in baySeq. We first define a group structure on the data.

```
> groups(hSL) <- list(NDE = c(1,1,1,1), DE = c("A", "A", "B", "B"))
```

The methObservables function pre-calculates a set of data to improve the speed of prior and posterior estimation (at some minor memory cost).

```
> hSL <- methObservables(hSL)
```

The density function used here is a composite of the beta-binomial and a binomial distribution that accounts for the reported non-conversion rates.

```
> densityFunction(hSL) <- bbNCDist
```

We can then determine a prior distribution on the parameters of the model for the data.

```
> hSL <- getPriors(hSL, cl = cl)
```

We can then find the posterior likelihoods of the models defined in the groups structure.

```
> hSL <- getLikelihoods(hSL, cl = cl)

.
```

We can then retrieve the data for the top differentially methylated regions.

```
> topCounts(hSL, "DE")

        seqnames   start     end width strand     A.1     A.2    B.1  B.2
NA.3932     Chr1 1439804 1440002   199      *   1:239   0:119 102:56 11:5
NA.3064     Chr1  774658  774937   280      *   0:172    0:90  38:16 14:7
NA.3651     Chr1 1192725 1192730     6      *    37:4    15:2   0:55  0:8
NA.3225     Chr1  888560  888590    31      *  423:90  217:56   1:37  0:7
NA.4221     Chr1 1733868 1733987   120      *   0:104   0:150  76:62 11:7
NA.2257     Chr1   25580   25583     4      *    20:2    49:3   0:58  0:3
NA.2517     Chr1  297002  297016    15      *   60:23   48:19   0:48  0:9
NA.3229     Chr1  889432  889508    77      *   2:204   0:131  75:24  9:3
NA.2768     Chr1  526399  526607   209      * 113:148 150:192  0:300 0:15
NA.4179     Chr1 1686726 1686728     3      *    27:2    24:1   0:30  0:5
            likes  DE       FDR.DE      FWER.DE
NA.3932 0.9996912 B>A 0.0003088359 0.0003088359
NA.3064 0.9996892 B>A 0.0003098112 0.0006195264
NA.3651 0.9996150 A>B 0.0003348622 0.0010042520
NA.3225 0.9995704 A>B 0.0003585405 0.0014333962
NA.4221 0.9995684 B>A 0.0003731480 0.0018643556
NA.2257 0.9995678 A>B 0.0003829942 0.0022957746
NA.2517 0.9995406 A>B 0.0003939133 0.0027541480
NA.3229 0.9995296 B>A 0.0004034744 0.0032232545
NA.2768 0.9995003 A>B 0.0004141697 0.0037213755
NA.4179 0.9994953 A>B 0.0004232257 0.0042242269
```

Finally, to be a good citizen, we stop the cluster we started earlier:

```
> if(!is.null(cl))
+     stopCluster(cl)
```

Session Info

```
> sessionInfo()

R version 4.1.1 (2021-08-10)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 20.04.3 LTS
```

```
Matrix products: default
BLAS:   /home/biocbuild/bbs-3.14-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.14-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

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

other attached packages:
 [1] segmentSeq_2.28.0           ShortRead_1.52.0
 [3] GenomicAlignments_1.30.0    SummarizedExperiment_1.24.0
 [5] Biobase_2.54.0              MatrixGenerics_1.6.0
 [7] matrixStats_0.61.0          Rsamtools_2.10.0
 [9] Biostrings_2.62.0           XVector_0.34.0
[11] BiocParallel_1.28.0         baySeq_2.28.0
[13] abind_1.4-5                 GenomicRanges_1.46.0
[15] GenomeInfoDb_1.30.0         IRanges_2.28.0
[17] S4Vectors_0.32.0            BiocGenerics_0.40.0

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.7             RColorBrewer_1.1-2     compiler_4.1.1
 [4] BiocManager_1.30.16    bitops_1.0-7           tools_4.1.1
 [7] zlibbioc_1.40.0        digest_0.6.28          evaluate_0.14
[10] lattice_0.20-45        png_0.1-7              rlang_0.4.12
[13] Matrix_1.3-4           DelayedArray_0.20.0    rstudioapi_0.13
[16] yaml_2.2.1             xfun_0.27              fastmap_1.1.0
[19] GenomeInfoDbData_1.2.7 hwriter_1.3.2          knitr_1.36
[22] locfit_1.5-9.4         grid_4.1.1             jpeg_0.1-9
[25] rmarkdown_2.11         latticeExtra_0.6-29    limma_3.50.0
[28] edgeR_3.36.0           htmltools_0.5.2        BiocStyle_2.22.0
[31] RCurl_1.98-1.5         crayon_1.4.1
```

References

[1] Thomas J. Hardcastle and Krystyna A. Kelly and David C. Baulcombe. *Identifying small RNA loci from high-throughput sequencing data.* Bioinformatics (2012).

[2] Thomas J. Hardcastle and Krystyna A. Kelly. *Empirical Bayesian analysis of paired high-throughput sequencing data with a beta-binomial distribution.* BMC Bioinformatics (2013).