segmentSeq: methods for detecting methylation loci and differential methylation

Thomas J. Hardcastle

April 24, 2017

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")
```

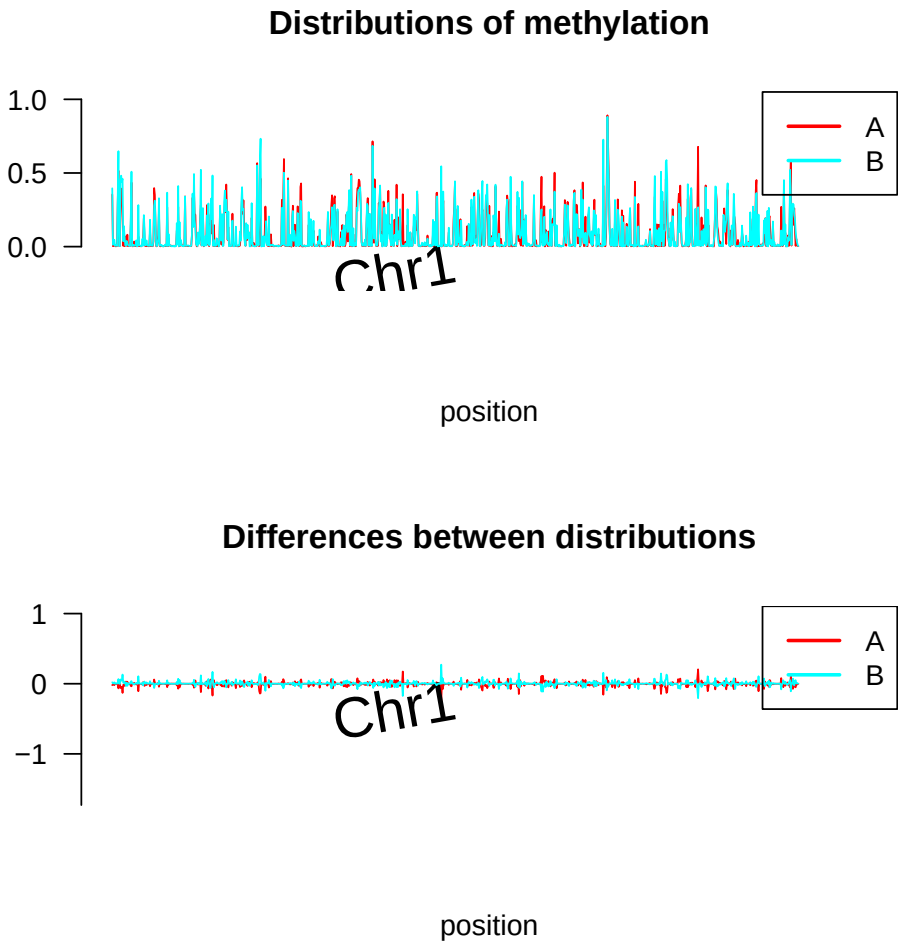

Figure 1: Distributions of methylation on the genome (first two million bases of chromosome 1.

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 may be determined by examining the data using the 'thresholdFinder' function, or supplied manually.

```
> thresh <- thresholdFinder("beta", mD, cl = cl, bootstrap = 1)
> hS <- heuristicSeg(sD = sD, aD = mD, prop = thresh, cl = cl, gap = 100, getLikes = FALSE)
> hS
GRanges object with 4298 ranges and 0 metadata columns:
         seqnames             ranges strand
            <Rle>          <IRanges>  <Rle>
     [1]     Chr1         [108, 115]      *
     [2]     Chr1         [161, 500]      *
     [3]     Chr1         [511, 511]      *
     [4]     Chr1         [642, 650]      *
     [5]     Chr1         [809, 809]      *
     ...      ...                ...    ...
  [4294]     Chr1 [1988245, 1988245]      *
  [4295]     Chr1 [1990160, 1990160]      *
  [4296]     Chr1 [1990186, 1990298]      *
  [4297]     Chr1 [1993149, 1993149]      *
  [4298]     Chr1 [1993335, 1993335]      *
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
An object of class "lociData"
4298 rows and 4 columns

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

Slot "data":
     [,1]  [,2]  [,3]  [,4]
[1,] 64:12 14:6  31:9  1:1
[2,] 49:13 47:21 40:15 8:3
[3,] 10:0  9:2   8:2   5:0
[4,] 55:18 75:31 36:9  10:2
[5,] 0:0   0:3   5:0   0:0
4293 more rows...

Slot "annotation":
data frame with 0 columns and 4298 rows

Slot "locLikelihoods" (stored on log scale):
Matrix with  4298  rows.
       A   B
1      1   1
2      1   1
3      1   1
4      1   1
5      0   1
...  ... ...
4294   1   0
4295   0   1
4296   0   1
4297   1   1
4298   1   0

Expected number of loci in each replicate group
   A    B
3199 3083
```

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.

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

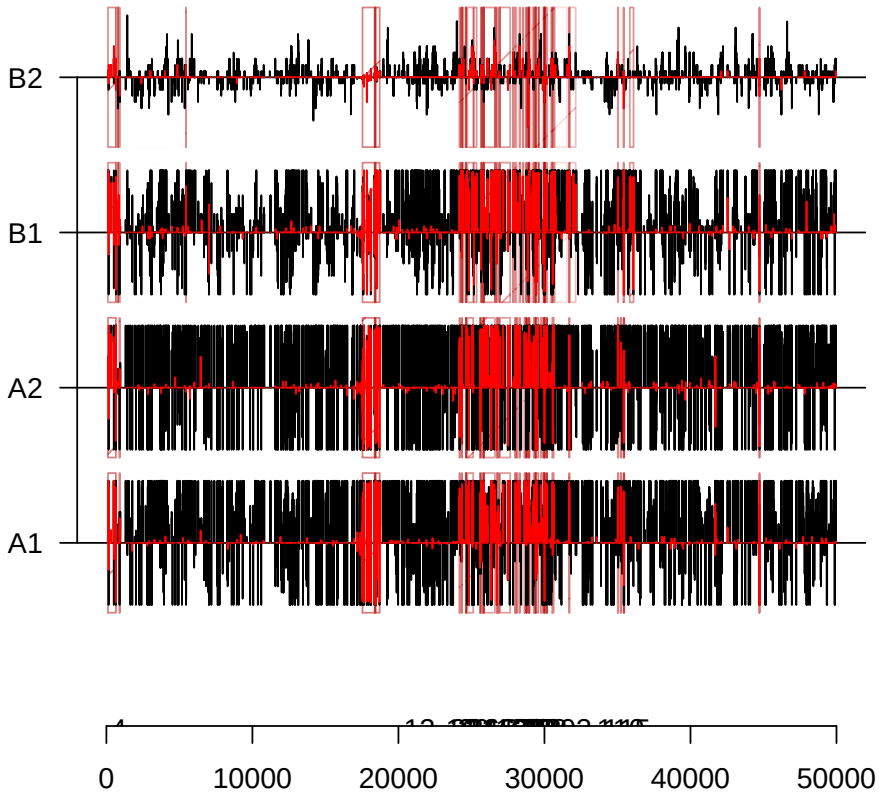

Figure 2: Methylation and identified loci on the first ten thousand bases of chromosome 1.

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	Likelihood	ordering	FDR.DE	FWER.DE
1	Chr1	1439804	1440002	199	*	1:239	0:119	102:56	11:5	0.9996908	B>A	0.0003091902	0.0003091902
2	Chr1	774658	774937	280	*	0:172	0:90	38:16	14:7	0.9996887	B>A	0.0003102264	0.0006203566
3	Chr1	1192725	1192730	6	*	37:4	15:2	0:55	0:8	0.9996146	A>B	0.0003352703	0.0010054756
4	Chr1	888560	888590	31	*	423:90	217:56	1:37	0:7	0.9995701	A>B	0.0003589179	0.0014349042
5	Chr1	1733868	1733987	120	*	0:104	0:150	76:62	11:7	0.9995680	B>A	0.0003735373	0.0018662989
6	Chr1	25580	25583	4	*	20:2	49:3	0:58	0:3	0.9995674	A>B	0.0003833844	0.0022981114
7	Chr1	297002	297016	15	*	60:23	48:19	0:48	0:9	0.9995402	A>B	0.0003943065	0.0027568939
8	Chr1	889432	889508	77	*	2:204	0:131	75:24	9:3	0.9995291	B>A	0.0004038827	0.0032265120
9	Chr1	526399	526607	209	*	113:148	150:192	0:300	0:15	0.9995000	A>B	0.0004145570	0.0037248502
10	Chr1	1686726	1686728	3	*	27:2	24:1	0:30	0:5	0.9994945	A>B	0.0004236465	0.0042284193

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

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

Session Info

```
> sessionInfo()
```

```
R version 3.4.0 (2017-04-21)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: OS X El Capitan 10.11.6

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib

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

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

other attached packages:
 [1] segmentSeq_2.10.0          ShortRead_1.34.0           GenomicAlignments_1.12.0
 [4] SummarizedExperiment_1.6.0 DelayedArray_0.2.0         matrixStats_0.52.2
 [7] Biobase_2.36.0             Rsamtools_1.28.0           Biostrings_2.44.0
[10] XVector_0.16.0             BiocParallel_1.10.0        baySeq_2.10.0
[13] abind_1.4-5                GenomicRanges_1.28.0       GenomeInfoDb_1.12.0
[16] IRanges_2.10.0             S4Vectors_0.14.0           BiocGenerics_0.22.0

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.10            RColorBrewer_1.1-2      compiler_3.4.0
 [4] bitops_1.0-6            tools_3.4.0             zlibbioc_1.22.0
 [7] digest_0.6.12           evaluate_0.10           lattice_0.20-35
[10] Matrix_1.2-9            yaml_2.1.14             GenomeInfoDbData_0.99.0
[13] stringr_1.2.0           hwriter_1.3.2           knitr_1.15.1
[16] locfit_1.5-9.1          rprojroot_1.2           grid_3.4.0
[19] rmarkdown_1.4           limma_3.32.0            latticeExtra_0.6-28
[22] edgeR_3.18.0            magrittr_1.5            backports_1.0.5
[25] htmltools_0.3.5         BiocStyle_2.4.0         stringi_1.1.5
[28] RCurl_1.95-4.8
```

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).