segmentSeq: methods for identifying small RNA loci from high-throughput sequencing data

Thomas J. Hardcastle

October 13, 2015

1 Introduction

High-throughput sequencing technologies allow the production of large volumes of short sequences, which can be aligned to the genome to create a set of *matches* to the genome. By looking for regions of the genome which to which there are high densities of matches, we can infer a segmentation of the genome into regions of biological significance. The methods we propose allows the simultaneous segmentation of data from multiple samples, taking into account replicate data, in order to create a consensus segmentation. This has obvious applications in a number of classes of sequencing experiments, particularly in the discovery of small RNA loci and novel mRNA transcriptome discovery.

We approach the problem by considering a large set of potential *segments* upon the genome and counting the number of tags that match to that segment in multiple sequencing experiments (that may or may not contain replication). We then adapt the empirical Bayesian methods implemented in the `baySeq` package [1] to establish, for a given segment, the likelihood that the count data in that segment is similar to background levels, or that it is similar to the regions to the left or right of that segment. We then rank all the potential segments in order of increasing likelihood of similarity and reject those segments for which there is a high likelihood of similarity with the background or the regions to the left or right of the segment. This gives us a large list of overlapping segments. We reduce this list to identify non-overlapping loci by choosing, for a set of overlapping segments, the segment which has the lowest likelihood of similarity with either background or the regions to the left or right of that segment and rejecting all other segments that overlap with this segment. For fuller details of the method, see Hardcastle *et al.* [2].

2 Preparation

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.

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

The `readGeneric` function is able to read in tab-delimited files which have appropriate column names, and create an `alignmentData` object. Alternatively, if the appropriate column names are not present, we can specify which columns to use for the data. In either case, to use this function we pass a character vector of files, together with information on which data are to be treated as replicates to the function. We also need to define the lengths of

the chromosome and specifiy the chromosome names as a character. The data here, drawn from text files in the 'data' directory of the segmentSeq package are taken from the first million bases of an alignment to chromosome 1 and the first five hundred thousand bases of an alignment to chromosome 2 of *Arabidopsis thaliana* in a sequencing experiment where libraries 'SL9' and 'SL10' are replicates, as are 'SL26' and 'SL32'. Libraries 'SL9' and 'SL10' are sequenced from an Argonaute 6 IP, while 'SL26' and 'SL32' are an Argonaute 4 IP.

A similar function, readBAM performs the same operation on files in the BAM format. Please consult the help page for further details.

```
> chrlens <- c(1e6, 2e5)
> datadir <- system.file("extdata", package = "segmentSeq")
> libfiles <- c("SL9.txt", "SL10.txt", "SL26.txt", "SL32.txt")
> libnames <- c("SL9", "SL10", "SL26", "SL32")
> replicates <- c("AGO6", "AGO6", "AGO4", "AGO4")
> aD <- readGeneric(files = libfiles, dir = datadir,
+                   replicates = replicates, libnames = libnames,
+                   chrs = c(">Chr1", ">Chr2"), chrlens = chrlens,
+                   polyLength = 10, header = TRUE, gap = 200)
> aD
```

```
An object of class "alignmentData"
13765 rows and 4 columns

Slot "libnames":
[1] "SL9"  "SL10" "SL26" "SL32"

Slot "replicates":
[1] AGO6 AGO6 AGO4 AGO4
Levels: AGO4 AGO6

Slot "alignments":
GRanges object with 13765 ranges and 2 metadata columns:
          seqnames           ranges strand   |                     tag multireads
             <Rle>        <IRanges>  <Rle>   |             <character>  <numeric>
      [1]    >Chr1       [265, 284]      -   |     AAATGAAGATAAACCATCCA          1
      [2]    >Chr1       [405, 427]      -   |  AAGGAGTAAGAATGACAATAAAT          1
      [3]    >Chr1       [406, 420]      -   |           AAGAATGACAATAAA          1
      [4]    >Chr1       [600, 623]      +   | AAGGATTGGTGGTTTGAAGACACA          1
      [5]    >Chr1       [665, 688]      +   | ATCCTTGTAGCACACATTTTGGCA          1
      ...      ...              ...    ... ...                      ...        ...
  [13761]    >Chr2 [179972, 179993]      +   |   ATGAATGGCTCTCTCTAGCGGA          1
  [13762]    >Chr2 [179978, 180000]      -   |  GAGATTCTCCGCTAGAGAGAGCC          1
  [13763]    >Chr2 [179999, 180022]      -   | ATTAATATTAATTCATCGGGAAGA          1
  [13764]    >Chr2 [180002, 180022]      -   |    ATTAATATTAATTCATCGGGA          1
  [13765]    >Chr2 [180014, 180037]      +   | AATATTAATGGTATTTGTGGAAAA          1
  -------
  seqinfo: 2 sequences from an unspecified genome

Slot "data":
Matrix with  13765  rows.
      SL9 SL10 SL26 SL32
1       1    0    0    0
2       0    0    0    2
3       0    1    0    0
4       0    1    0    0
5       7    1    0    0
...   ...  ...  ...  ...
13761   2    7    0    0
13762   0    1    0    0
13763   0    1    0    0
```

```
13764   0    1    0    0
13765   1    0    0    0

Slot "libsizes":
[1] 4447 6531 9666 6675
```

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(aD, gap = 100, cl = cl)
> sD
```

```
An object of class "segData"
14444 rows and 4 columns

Slot "replicates":
[1] AGO6 AGO6 AGO4 AGO4
Levels: AGO4 AGO6

Slot "coordinates":
GRanges object with 14444 ranges and 0 metadata columns:
          seqnames            ranges strand
             <Rle>         <IRanges>  <Rle>
      [1]    >Chr1       [265, 284]      *
      [2]    >Chr1       [405, 427]      *
      [3]    >Chr1       [600, 623]      *
      [4]    >Chr1       [600, 688]      *
      [5]    >Chr1       [600, 830]      *
      ...      ...              ...    ...
  [14440]    >Chr2 [179708, 179872]      *
  [14441]    >Chr2 [179708, 180037]      *
  [14442]    >Chr2 [179738, 179872]      *
  [14443]    >Chr2 [179738, 180037]      *
  [14444]    >Chr2 [179923, 180037]      *
  -------
  seqinfo: 2 sequences from an unspecified genome

Slot "locLikelihoods" (stored on log scale):
Matrix with  0  rows.
<0 x 0 matrix>

Slot "data":
Matrix with  0  rows.Matrix with  0  rows.
     SL9 SL10 SL26 SL32

Slot "libsizes":
[1] 4447 6531 9666 6675
```

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

Segmentation by heuristic methods

A fast method of segmentation can be achieved by exploiting the bimodality of the densities of small RNAs in the potential segments. In this approach, we assign each potential segment to one of two clusters for each replicate group, either as a segment or a null based on the density of sequence tags within that segment. We then combine these clusterings for each replicate group to gain a consensus segmentation map.

```
> hS <- heuristicSeg(sD = sD, aD = aD, RKPM = 1000, largeness = 1e8, getLikes = TRUE, cl = cl)
```

```
......
```

Segmentation by empirical Bayesian methods

A more refined approach to the problem uses an existing segment map (or, if not provided, a segment map defined by the hS function) to acquire empirical distributions on the density of sequence tags within a segment. We can then estimate posterior likelihoods for each potential segment as being either a true segment or a null. We then identify all potential segments in the with a posterior likelihood of being a segment greater than some value 'lociCutoff' and containing no subregion with a posterior likelihood of being a null greater than 'nullCutoff'. We then greedily select the longest segments satisfying these criteria that do not overlap with any other such segments in defining our segmentation map.

```
> classSegs <- classifySeg(sD = sD, aD = aD, cD = hS, cl = cl)
```

```
............
```

```
> classSegs
```

```
GRanges object with 260 ranges and 0 metadata columns:
        seqnames            ranges strand
           <Rle>         <IRanges>  <Rle>
    [1]    >Chr1    [    1,   599]      *
    [2]    >Chr1    [  600,   938]      *
    [3]    >Chr1    [  939,   967]      *
    [4]    >Chr1    [  968, 17054]      *
    [5]    >Chr1    [17055, 18728]      *
    ...      ...               ...    ...
  [256]    >Chr2 [169231, 178343]      *
  [257]    >Chr2 [178344, 178636]      *
  [258]    >Chr2 [178637, 179707]      *
  [259]    >Chr2 [179708, 180037]      *
  [260]    >Chr2 [180038, 200000]      *
  -------
  seqinfo: 2 sequences from an unspecified genome
An object of class "lociData"
260 rows and 4 columns

Slot "replicates"
AGO6 AGO6 AGO4 AGO4
Slot "groups":
[[1]]
[1] AGO6 AGO6 AGO4 AGO4
Levels: AGO4 AGO6


Slot "data":
     AGO6.1 AGO6.2 AGO4.1 AGO4.2
[1,]      1      1      0      2
[2,]     41     39     65     83
[3,]     13      7      0      0
[4,]      2      3      0      0
[5,]    682    621   1405   1103
255 more rows...

Slot "annotation":
data frame with 0 columns and 260 rows

Slot "locLikelihoods" (stored on log scale):
Matrix with  260  rows.
```

```
        AGO4     AGO6
1     0.2714  0.10747
2    0.95111  0.96155
3    0.13302   0.9882
4   0.034435 0.033831
5    0.98355  0.99454
...      ...      ...
256  0.32674 0.073065
257  0.92274  0.97614
258 0.052708 0.063101
259   0.9597  0.96776
260 0.033827 0.025357

Expected number of loci in each replicate group
    AGO4     AGO6
115.9893 138.8527
```

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

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

```
> par(mfrow = c(2,1), mar = c(2,6,2,2))
> plotGenome(aD, hS, chr = ">Chr1", limits = c(1, 1e5),
+            showNumber = FALSE, cap = 50)
> plotGenome(aD, classSegs, chr = ">Chr1", limits = c(1, 1e5),
+            showNumber = FALSE, cap = 50)
```

Given the calculated likelihoods, we can filter the segmented genome by controlling on likelihood, false discovery rate, or familywise error rate

```
> loci <- selectLoci(classSegs, FDR = 0.05)
> loci
```

```
GRanges object with 92 ranges and 0 metadata columns:
       seqnames           ranges strand
          <Rle>        <IRanges>  <Rle>
   [1]    >Chr1   [  600,   938]      *
   [2]    >Chr1   [  939,   967]      *
   [3]    >Chr1   [17055, 18728]      *
   [4]    >Chr1   [44710, 44811]      *
   [5]    >Chr1   [76799, 76890]      *
   ...      ...              ...    ...
  [88]    >Chr2 [152150, 152173]      *
  [89]    >Chr2 [152768, 152829]      *
  [90]    >Chr2 [169196, 169230]      *
  [91]    >Chr2 [178344, 178636]      *
  [92]    >Chr2 [179708, 180037]      *
  -------
  seqinfo: 2 sequences from an unspecified genome
An object of class "lociData"
92 rows and 4 columns

Slot "replicates"
AGO6 AGO6 AGO4 AGO4
Slot "groups":
[[1]]
[1] AGO6 AGO6 AGO4 AGO4
Levels: AGO4 AGO6
```

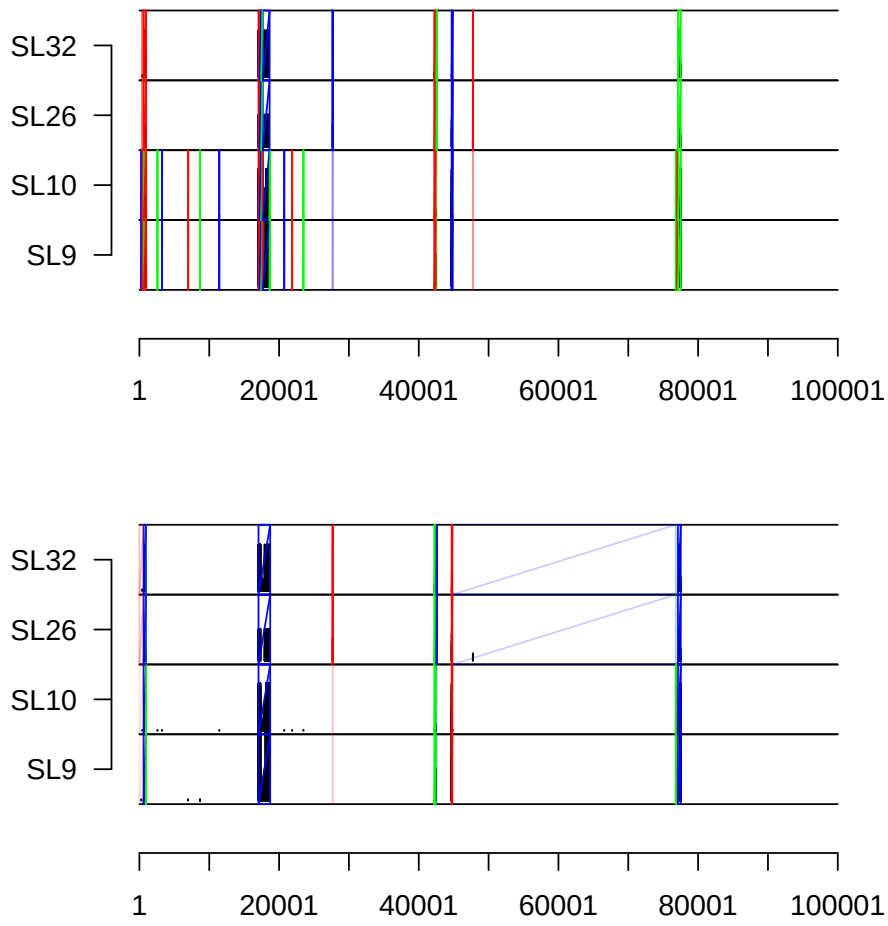

Figure 1: The segmented genome (first 10^5 bases of chromosome 1.

```
Slot "data":
     AGO6.1 AGO6.2 AGO4.1 AGO4.2
[1,]     41     39     65     83
[2,]     13      7      0      0
[3,]    682    621   1405   1103
[4,]     73     57     47     21
[5,]      6     19      0      0
87 more rows...

Slot "annotation":
data frame with 0 columns and 92 rows

Slot "locLikelihoods" (stored on log scale):
Matrix with  92  rows.
       AGO4    AGO6
1   0.95111 0.96155
2   0.13302  0.9882
3   0.98355 0.99454
4   0.96349  0.9965
5   0.10008 0.96181
...     ...     ...
88  0.13952  0.9601
```

```
89  0.96398 0.94447
90  0.12695 0.98545
91  0.92274 0.97614
92   0.9597 0.96776

Expected number of loci in each replicate group
    AGO4     AGO6
65.12488 89.41338
```

The `lociData` objects can now be examined for differential expression with the `baySeq` package.

First we define the possible models of differential expression on the data. In this case, the models are of non-differential expression and pairwise differential expression.

```
> groups(classSegs) <- list(NDE = c(1,1,1,1), DE = c("AGO6", "AGO6", "AGO4", "AGO4"))
```

Then we get empirical distributions on the parameter space of the data.

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

Then we get the posterior likelihoods of the data conforming to each model. Since the 'classSegs' object contains null regions as well as true loci, we will use the 'nullData = TRUE' option to distinguish between non-differentially expressed loci and non-expressed regions. By default, the loci likelihoods calculated earlier will be used to weight the initial parameter fit in order to detect null data.

```
> classSegs <- getLikelihoods(classSegs, nullData = TRUE, cl = cl)
```

.

We can examine the highest likelihood non-expressed ('null') regions

```
> topCounts(classSegs, NULL, number = 3)
```

```
  seqnames  start    end width strand AGO6.1 AGO6.2 AGO4.1 AGO4.2 Likelihood       FDR.
1    >Chr1 754198 758593  4396      *      1      0      1      0  0.9316901 0.06830994
2    >Chr1 309348 365852 56505      *      1      0      0      3  0.9284444 0.06993277
3    >Chr1 950796 958752  7957      *      2      4      3      2  0.9271848 0.07089357
       FWER.
1 0.06830994
2 0.13497758
3 0.19796435
```

The highest likelihood expressed but non-differentially expressed regions

```
> topCounts(classSegs, "NDE", number = 3)
```

```
  seqnames  start    end width strand AGO6.1 AGO6.2 AGO4.1 AGO4.2 Likelihood    FDR.NDE
1    >Chr2   1554  14423 12870      *   7632  18470  20242  13836  0.9784592 0.02154085
2    >Chr1 446325 447437  1113      *    789    536   1291   1184  0.9378767 0.04183210
3    >Chr1  17055  18728  1674      *    682    621   1405   1103  0.9026889 0.06032510
    FWER.NDE
1 0.02154085
2 0.08232600
3 0.17162588
```

And the highest likelihood differentially expressed regions

```
> topCounts(classSegs, "DE", number = 3)
```

```
  seqnames start   end width strand AGO6.1 AGO6.2 AGO4.1 AGO4.2 Likelihood  ordering
1    >Chr2 52652 53314   663      *     84     86    982    696  0.9990025 AGO4>AGO6
2    >Chr2 58131 59084   954      *     81     83    965    677  0.9986530 AGO4>AGO6
3    >Chr2 49137 50333  1197      *    101    116    992    719  0.9984819 AGO4>AGO6
        FDR.DE      FWER.DE
1 0.0009974804 0.0009974804
2 0.0011722264 0.0023431091
3 0.0012875303 0.0038576901
```

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

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

Session Info

```
> sessionInfo()
```

```
R version 3.2.2 (2015-08-14)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 14.04.3 LTS

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=C               LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

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

other attached packages:
 [1] segmentSeq_2.4.0           ShortRead_1.28.0           GenomicAlignments_1.6.0
 [4] SummarizedExperiment_1.0.0 Biobase_2.30.0             Rsamtools_1.22.0
 [7] Biostrings_2.38.0          XVector_0.10.0             BiocParallel_1.4.0
[10] baySeq_2.4.0               perm_1.0-0.0               abind_1.4-3
[13] GenomicRanges_1.22.0       GenomeInfoDb_1.6.0         IRanges_2.4.0
[16] S4Vectors_0.8.0            BiocGenerics_0.16.0

loaded via a namespace (and not attached):
 [1] zlibbioc_1.16.0      lattice_0.20-33      hwriter_1.3.2        tools_3.2.2
 [5] grid_3.2.2           latticeExtra_0.6-26  lambda.r_1.1.7       futile.logger_1.4.1
 [9] RColorBrewer_1.1-2   futile.options_1.0.0 bitops_1.0-6         BiocStyle_1.8.0
```

References

[1] Thomas J. Hardcastle and Krystyna A. Kelly. *baySeq: Empirical Bayesian Methods For Identifying Differential Expression In Sequence Count Data.* BMC Bioinformatics (2010).

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