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

Thomas J. Hardcastle

April 14, 2011

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 based on the Poisson-Gamma conjugacy and 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 (2010) [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 `snow` package. We therefore need to load the `snow` package (if it exists) and define a *cluster*.

```
> library(snow)
> cl <- makeCluster(7, "MPI")
```

If `snow` is not present, 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.

```
> cl <- NULL
```

There is a convenience function, `processTags` which 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, 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'.

```
> chrlens <- c(1e+06, 5e+05)
> datadir <- system.file("extdata", package = "segmentSeq")
> libfiles <- c("SL9.txt", "SL10.txt", "SL26.txt", "SL32.txt")
> libnames <- c("SL9", "SL10", "SL26", "SL32")
> replicates <- c(1, 1, 2, 2)
> aD <- processTags(libfiles, dir = datadir, replicates, libnames,
+     chrlens, chrs = c(">Chr1", ">Chr2"), header = TRUE, gap = 200)
> aD

An object of class "alignmentData"
22717 rows and 4 columns

Slot "alignments":
        chr start end                    tag matches chunk chunkDup
9233  >Chr1     1  22 GTTTAGGGTTTAGGGTTTAGGG       2     1     TRUE
26069 >Chr1     3  21    CTAAACCCTAAACCCTAAA       2     1     TRUE
9231  >Chr1     5  19        AAACCCTAAACCCTA       2     1     TRUE
9232  >Chr1     5  20       AAACCCTAAACCCTAA       2     1     TRUE
9234  >Chr1     5  23    AAACCCTAAACCCTAAACC       2     1     TRUE
22712 more rows...

Slot "data":
     [,1] [,2] [,3] [,4]
[1,]    0    1    0    0
[2,]    0    0    0    8
[3,]    0    1    0    0
[4,]    0    1    0    0
[5,]    0    2    0    0
 more rows...
```

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

Slot "libsizes":
[1] 20627 35908 36864 30038

Slot "replicates":
[1] 1 1 2 2

Slot "chrs":
    chr     len
1 >Chr1 1000000
2 >Chr2  500000
```

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, cl = cl)
> sD
```

```
An object of class "segData"
91425 rows and 4 columns

Slot "data":
  SL9 SL10 SL26 SL32
1  27   17    0   16
2  28   17    0   16
3  31   18    0   16
4  32   18    0   16
5  32   19    0   18
91420 more rows...

Slot "libsizes":
[1] 20627 35908 36864 30038

Slot "replicates":
[1] 1 1 2 2

Slot "segInfo":
    chr start end chunk leftSpace rightSpace
1 >Chr1     1  63     1         0          1
2 >Chr1     1  88     1         0          1
3 >Chr1     1 113     1         0        151
4 >Chr1     1 284     1         0        120
5 >Chr1     1 427     1         0        172
91420 more rows...
```

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

Segmentation by Clustering

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.

```
> clustSegs <- heuristicSeg(sD = sD, aD = aD, bimodality = TRUE,
+     getLikes = TRUE, cl = cl)

..

> clustSegs

An object of class "postSeg"
704 rows and 4 columns

Slot "data":
   SL9 SL10 SL26 SL32
1   27   17    0   16
    12    4    0    2
50  23   26   51   83
    11   11   14    0
55  13    7    0    0
     2   18    0    2
66   0   38  126    4
     0    2    0    0
70  76   83  147  545
     2    1    0    0
694 more rows...

Slot "libsizes":
[1] 20627 35908 36864 30038

Slot "groups":
[[1]]
[1] 1 1 2 2


Slot "annotation":
      chr start   end
352 >Chr1     1    63
1   >Chr1    64   761
353 >Chr1   762   830
2   >Chr1   831   940
354 >Chr1   941   967
3   >Chr1   968 12997
355 >Chr1 12998 13014
4   >Chr1 13015 17054
```

```
356 >Chr1 17055 17111
5   >Chr1 17112 17274
694 more rows...
Slot "posteriors":
             [,1]          [,2]
 [1,] -0.26892787 -4.994135e-01
 [2,] -2.20559720 -3.334568e+00
 [3,] -0.25369109 -8.474732e-03
 [4,] -2.66407373 -9.705260e-01
 [5,] -0.22965050 -3.573336e+00
 [6,] -3.81301246 -4.572728e+00
 [7,] -0.05055371 -6.579487e-04
 [8,] -4.48568530 -4.628260e+00
 [9,] -0.00142193 -2.953120e-06
[10,] -3.34904640 -4.129325e+00
694 more rows...
```

Segmentation by Classification

A more refined approach to the problem uses an existing segment map (or, if not provided, a segment map defined by the `clustSegs` 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 identifying all potential segments in the with a posterior likelihood of being a segment greater than some value 'locsens' and containing no subregion with a posterior likelihood of being a null greater than 'nulsens'. 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 = clustSegs, subRegion = NULL,
+     getLikes = TRUE, cl = cl, lociCutoff = 0.9, nullCutoff = 0.9)

..

> classSegs

An object of class "postSeg"
344 rows and 4 columns

Slot "data":
    SL9 SL10 SL26 SL32
51   34   37   65   83
     15   25    0    2
66    0   38  126    4
      0    2    0    0
71   76   84  147  545
121  78   84  147  545
     26   23    0   15
259  17   13    0    0
      5    3    0   12
```

```
302 557  498 1259  555
334 more rows...

Slot "libsizes":
[1] 20627 35908 36864 30038

Slot "groups":
[[1]]
[1] 1 1 2 2


Slot "annotation":
      chr start   end
169 >Chr1   762   916
1   >Chr1   917 12997
170 >Chr1 12998 13014
2   >Chr1 13015 17054
171 >Chr1 17055 17185
172 >Chr1 17186 17394
3   >Chr1 17395 17696
173 >Chr1 17697 17724
4   >Chr1 17725 17803
174 >Chr1 17804 18701
334 more rows...
Slot "posteriors":
             [,1]         [,2]
 [1,] -0.86259909 -0.030655216
 [2,] -5.74609123 -5.153592815
 [3,] -0.23259889 -0.006139093
 [4,] -4.91241739 -5.143861125
 [5,] -0.01499119 -0.006159920
 [6,] -0.05320428 -0.014979822
 [7,] -2.10159811 -1.428036114
 [8,] -0.02951962 -4.268691286
 [9,] -2.08723171 -1.054451620
[10,] -0.02780860 -0.014202757
334 more rows...
```

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

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

```
> par(mfrow = c(2, 1), mar = c(2, 2, 2, 2))
> plotGenome(aD, clustSegs, chr = ">Chr1", limits = c(1, 3e+05))
> plotGenome(aD, classSegs, chr = ">Chr1", limits = c(1, 3e+05))
```

This `countData` object can now be examined for differential expression with the `baySeq` package.

References

[1] Thomas J. Hardcastle and Krystyna A. Kelly. *Empirical Bayesian Methods For Identifying Patterns of Differential Expression in Count Data*. In submission. 2010.

[2] Thomas J. Hardcastle and Krystyna A. Kelly. *Genome Segmentation From High-Throughput Sequencing Data.*. In preparation. 2010.

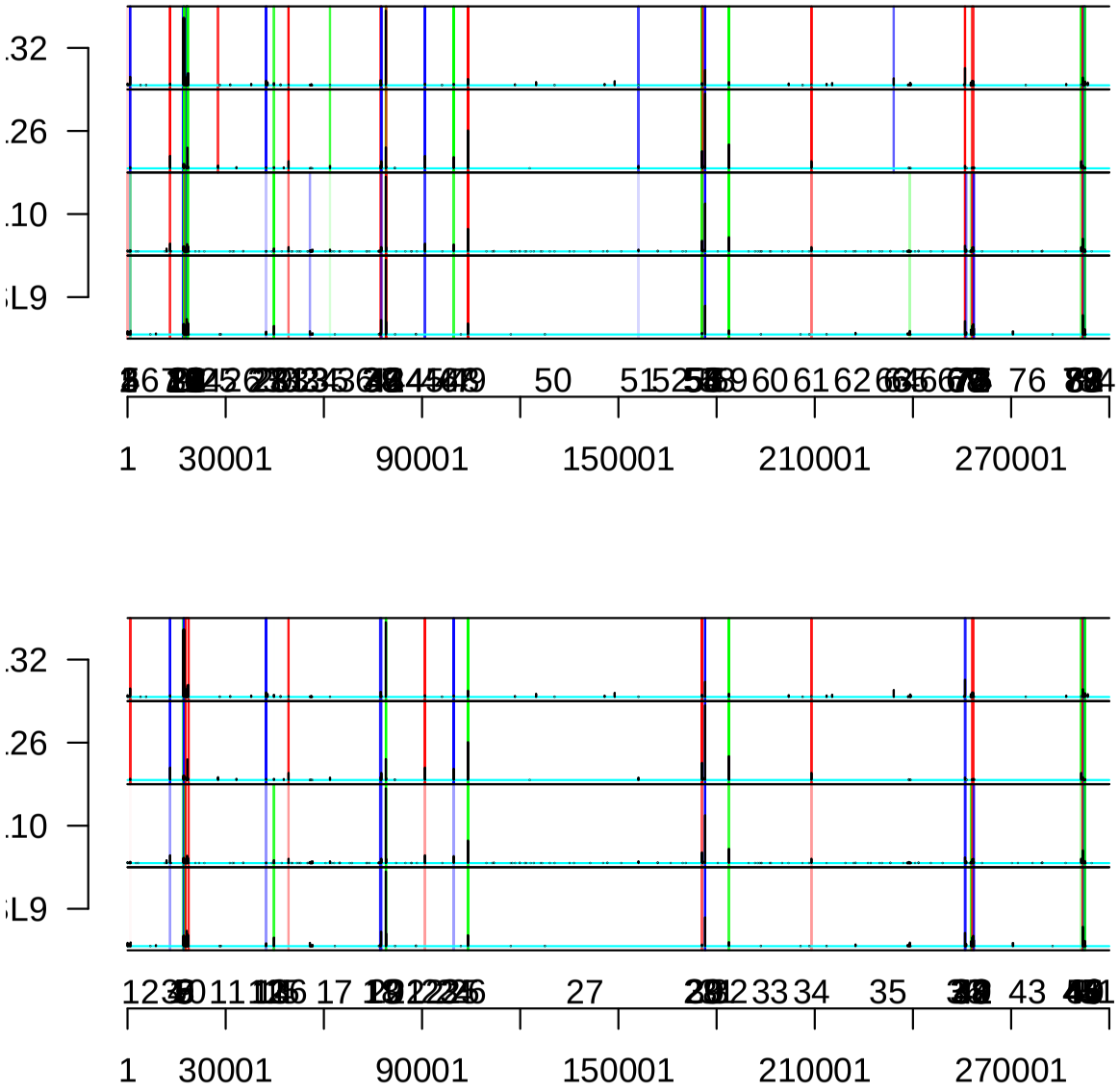

Figure 1: The segmented genome (first 10^5 bases of chromosome 1.