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

Thomas J. Hardcastle

May 2, 2014

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.

```
> cl <- makeCluster(8)
```

If we don't want to parallelise, we can proceed anyway with a `NULL` cluster.

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 with 13765 ranges and 2 metadata columns:
          seqnames           ranges strand   |                      tag
             <Rle>        <IRanges>  <Rle>   |              <character>
      [1]    >Chr1       [265, 284]      -   |     AAATGAAGATAAACCATCCA
      [2]    >Chr1       [405, 427]      -   |  AAGGAGTAAGAATGACAATAAAT
      [3]    >Chr1       [406, 420]      -   |           AAGAATGACAATAAA
      [4]    >Chr1       [600, 623]      +   | AAGGATTGGTGGTTTGAAGACACA
```

```
      [5]    >Chr1       [665, 688]      +   | ATCCTTGTAGCACACATTTTGGCA
      ...      ...              ...    ... ...                       ...
  [13761]    >Chr2 [179972, 179993]      +   |   ATGAATGGCTCTCTCTAGCGGA
  [13762]    >Chr2 [179978, 180000]      -   |  GAGATTCTCCGCTAGAGAGAGCC
  [13763]    >Chr2 [179999, 180022]      -   | ATTAATATTAATTCATCGGGAAGA
  [13764]    >Chr2 [180002, 180022]      -   |    ATTAATATTAATTCATCGGGA
  [13765]    >Chr2 [180014, 180037]      +   | AATATTAATGGTATTTGTGGAAAA
         multireads
          <numeric>
      [1]          1
      [2]          1
      [3]          1
      [4]          1
      [5]          1
      ...        ...
  [13761]          1
  [13762]          1
  [13763]          1
  [13764]          1
  [13765]          1
  ---
  seqlengths:
     >Chr1   >Chr2
   1000000  200000

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 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]      *
  ---
  seqlengths:
     >Chr1   >Chr2
    1000000  200000

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

Slot "data":
Matrix with  14444  rows.Matrix with  14444  rows.
      SL9 SL10 SL26 SL32
1       1    0    0    0
2       0    1    0    2
3       0    1    0    0
4       7    2    0    0
5      30   28   51   83
...   ...  ...  ...  ...
14440  32   30   76   88
14441  39   43   85   95
14442  31   30   76   88
14443  38   43   85   95
14444   7   13    9    7

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.

```
> clustSegs <- heuristicSeg(sD = sD, aD = aD, RKPM = 1000, largeness = 1e8, getLikes = TRU
```

```
NULL
......
```

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 `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 identify 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,
+                          lociCutoff = 0.9, nullCutoff = 0.9, cl = cl)
```

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

```
> classSegs
```

```
GRanges with 669 ranges and 0 metadata columns:
        seqnames           ranges strand
           <Rle>        <IRanges>  <Rle>
    [1]    >Chr1       [  1, 264]      *
    [2]    >Chr1       [265, 284]      *
    [3]    >Chr1       [285, 404]      *
    [4]    >Chr1       [405, 427]      *
    [5]    >Chr1       [428, 599]      *
    ...      ...              ...    ...
  [665]    >Chr2 [178637, 179096]      *
  [666]    >Chr2 [179097, 179111]      *
  [667]    >Chr2 [179112, 179707]      *
  [668]    >Chr2 [179708, 180037]      *
  [669]    >Chr2 [180038, 200000]      *
  ---
  seqlengths:
     >Chr1   >Chr2
```

```
   1000000  200000
An object of class "lociData"
669 rows and 4 columns

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

Slot "libsizes"
 SL9 SL10 SL26 SL32
4447 6531 9666 6675

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


Slot "data":
     SL9 SL10 SL26 SL32
[1,]   0    0    0    0
[2,]   1    0    0    0
[3,]   0    0    0    0
[4,]   0    1    0    2
[5,]   0    0    0    0
664 more rows...

Slot "annotation":
data frame with 0 columns and 669 rows

Slot "locLikelihoods" (stored on log scale):
Matrix with  669  rows.
         AGO4     AGO6
1    0.012485 0.023231
2    0.051274  0.83836
3    0.017957 0.034743
4     0.32246  0.83668
5    0.015026 0.028159
...       ...      ...
665  0.010208 0.019194
666  0.059147  0.85023
667 0.0094383 0.017896
668   0.99559  0.98478
669 0.0063219 0.012659

Expected number of loci in each replicate group
    AGO4     AGO6
120.1922 278.1393
```

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, clustSegs, chr = ">Chr1", limits = c(1, 1e5),
+            showNumber = FALSE, cap = 50)
> plotGenome(aD, classSegs, chr = ">Chr1", limits = c(1, 1e5),
+            showNumber = FALSE, cap = 50)
```

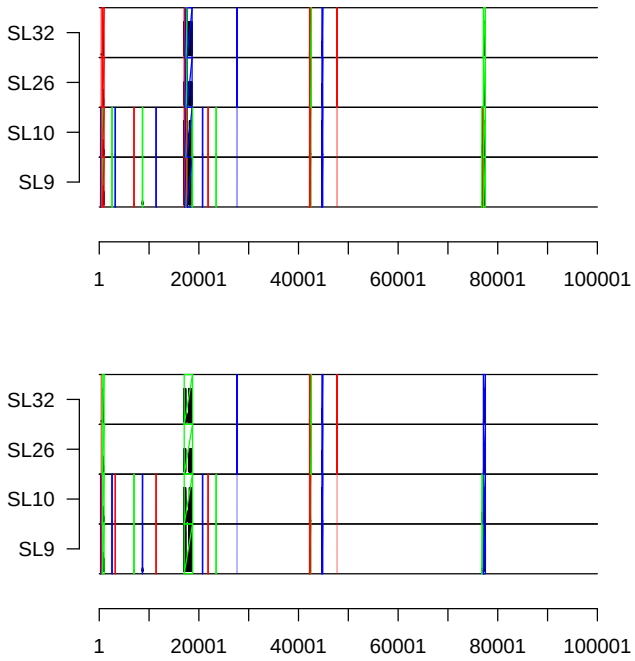

Figure 1: The segmented genome (first 10^5 bases of chromosome 1.

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 with 128 ranges and 0 metadata columns:
        seqnames           ranges strand
           <Rle>        <IRanges>  <Rle>
    [1]    >Chr1   [  600,   967]      *
    [2]    >Chr1   [17055, 18728]      *
    [3]    >Chr1   [27657, 27677]      *
```

```
    [4]    >Chr1   [42217, 42435]      *
    [5]    >Chr1   [42547, 42570]      *
    ...      ...              ...     ...
  [124]    >Chr2 [152150, 152173]      *
  [125]    >Chr2 [152768, 152877]      *
  [126]    >Chr2 [169196, 169230]      *
  [127]    >Chr2 [178344, 178636]      *
  [128]    >Chr2 [179708, 180037]      *
  ---
  seqlengths:
     >Chr1   >Chr2
   1000000  200000
An object of class "lociData"
128 rows and 4 columns

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

Slot "libsizes"
 SL9 SL10 SL26 SL32
4447 6531 9666 6675

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


Slot "data":
     SL9 SL10 SL26 SL32
[1,]  54   46   65   83
[2,] 758  705 1552 1648
[3,]   0    0   36    0
[4,]  31   11   48   56
[5,]   0    0    0   12
123 more rows...

Slot "annotation":
data frame with 0 columns and 128 rows

Slot "locLikelihoods" (stored on log scale):
Matrix with  128  rows.
        AGO4    AGO6
1    0.98596 0.98574
2    0.99997 0.99979
3    0.99965 0.18887
4    0.99334 0.93925
5     0.9938  0.1706
...      ...     ...
```

```
124 0.046565 0.96262
125   0.9647 0.87363
126 0.037569 0.99743
127  0.99995 0.99542
128  0.99559 0.98478

Expected number of loci in each replicate group
    AGO4     AGO6
 95.1946 107.8749
```

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

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

```
> stopCluster(cl)
```

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).