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

Thomas J. Hardcastle

October 31, 2024

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(FALSE) { # set to FALSE if you don't want parallelisation
+     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.

```
> 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,
+                   polyLength = 10, header = TRUE, gap = 200)
> aD

An object of class "alignmentData"
3149 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 3149 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      - | AAGGAGTAAGAATGACAATA..          1
     [3]    >Chr1       406-420      - |        AAGAATGACAATAAA          1
     [4]    >Chr1       600-623      + | AAGGATTGGTGGTTTGAAGA..          1
     [5]    >Chr1       665-688      + | ATCCTTGTAGCACACATTTT..          1
     ...      ...           ...    ... .                    ...        ...
  [3145]    >Chr1 991569-991589      - |  CCGATAAACGCATACTTCCCT          1
  [3146]    >Chr1 992039-992054      - |       AAGGAAATTAGAAAAT          1
  [3147]    >Chr1 995357-995372      + |       AGAGACATGGGCGACA          1
  [3148]    >Chr1 995493-995507      + |        AAACTCGTGAAGAAG          1
  [3149]    >Chr1 995817-995840      - | AGAGATCAAGTATATAGAAT..          1
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths

Slot "data":
Matrix with  3149  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
...  ...  ...  ...  ...
3145   1    0    0    0
3146   0    1    0    0
3147   0    1    0    0
3148   0    1    0    0
3149   1    0    0    0

Slot "libsizes":
[1] 1193 1598 1818 1417
```

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

GRanges object with 1452 ranges and 1 metadata column:
         seqnames        ranges strand | chunk
            <Rle>     <IRanges>  <Rle> | <Rle>
     [1]    >Chr1       265-284      * |     1
     [2]    >Chr1       265-420      * |     1
     [3]    >Chr1       265-623      * |     1
     [4]    >Chr1       265-688      * |     1
     [5]    >Chr1       265-830      * |     1
     ...      ...           ...    ... .   ...
  [1448]    >Chr1 992039-992054      * |   171
  [1449]    >Chr1 995357-995372      * |   172
  [1450]    >Chr1 995357-995507      * |   172
  [1451]    >Chr1 995493-995507      * |   172
  [1452]    >Chr1 995817-995840      * |   173
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
An object of class "lociData"
1452 rows and 4 columns
```

```
Slot "replicates"
AGO6 AGO6 AGO4 AGO4
Slot "groups":
list()

Slot "data":
     SL9 SL10 SL26 SL32

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

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

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.

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

..........

> cS

GRanges object with 65 ranges and 0 metadata columns:
       seqnames        ranges strand
          <Rle>     <IRanges>  <Rle>
   [1]    >Chr1         1-264      *
   [2]    >Chr1       265-839      *
   [3]    >Chr1       840-967      *
   [4]    >Chr1     968-17054      *
   [5]    >Chr1   17055-17738      *
   ...      ...           ...    ...
  [61]    >Chr1 789508-789548      *
  [62]    >Chr1 789549-944194      *
  [63]    >Chr1 944195-944222      *
  [64]    >Chr1 944223-958610      *
  [65]    >Chr1 958611-959152      *
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
An object of class "lociData"
65 rows and 4 columns

Slot "replicates"
AGO6 AGO6 AGO4 AGO4
Slot "groups":
list()

Slot "data":
     AGO6.1 AGO6.2 AGO4.1 AGO4.2
[1,]      0      0      0      0
[2,]     31     29     51     85
[3,]     24     18     14      0
[4,]      2      3      0      0
[5,]    121    120    147    560
60 more rows...

Slot "annotation":
data frame with 0 columns and 65 rows

Slot "locLikelihoods" (stored on log scale):
Matrix with  65  rows.
         AGO4     AGO6
1    0.031548 0.034761
2      0.8338   0.9883
3     0.61923  0.85384
4    0.014835 0.019089
5     0.95559  0.84837
...        ...      ...
61    0.66692  0.93945
62  0.0076827 0.024958
63    0.98159  0.86769
64    0.12483 0.018825
65    0.32536  0.99319

Expected number of loci in each replicate group
    AGO4     AGO6
29.40546 36.18992
```

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, cS, chr = ">Chr1", limits = c(1, 1e5),
+             showNumber = FALSE, cap = 50)
```

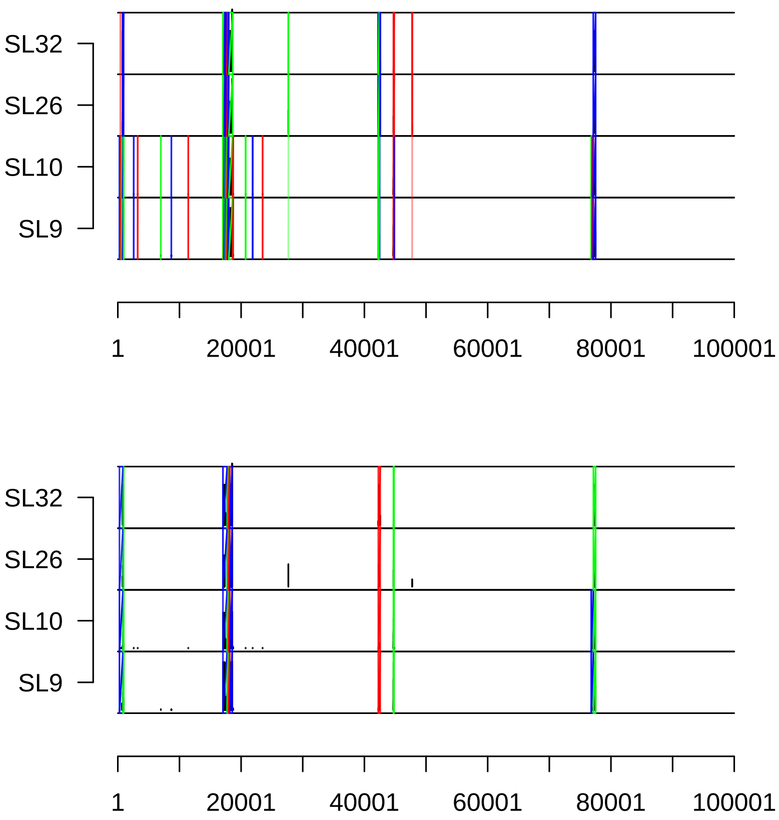

Figure 1: The segmented genome (first 10^5 bases of chromosome 1.

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

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

GRanges object with 34 ranges and 0 metadata columns:
       seqnames        ranges strand
          <Rle>     <IRanges>  <Rle>
   [1]    >Chr1       265-839      *
   [2]    >Chr1       840-967      *
   [3]    >Chr1   17739-17909      *
   [4]    >Chr1   17910-18157      *
   [5]    >Chr1   18158-18553      *
   ...      ...           ...    ...
  [30]    >Chr1 758302-758848      *
  [31]    >Chr1 758849-759196      *
  [32]    >Chr1 789508-789548      *
  [33]    >Chr1 944195-944222      *
  [34]    >Chr1 958611-959152      *
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
An object of class "lociData"
34 rows and 4 columns

Slot "replicates"
AGO6 AGO6 AGO4 AGO4
Slot "groups":
list()

Slot "data":
     AGO6.1 AGO6.2 AGO4.1 AGO4.2
[1,]     31     29     51     85
[2,]     24     18     14      0
[3,]     16     21      0     17
[4,]    239    173    162    177
[5,]    301    302   1096    349
29 more rows...

Slot "annotation":
data frame with 0 columns and 34 rows

Slot "locLikelihoods" (stored on log scale):
Matrix with  34  rows.
       AGO4    AGO6
1    0.8338  0.9883
2   0.61923 0.85384
3    0.6483 0.88199
4   0.98546 0.99732
5   0.98467 0.99843
...     ...     ...
30  0.98027 0.96982
31  0.56102 0.98788
32  0.66692 0.93945
33  0.98159 0.86769
34  0.32536 0.99319

Expected number of loci in each replicate group
    AGO4     AGO6
25.33357 32.75701
```

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(cS) <- 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.

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

Then we get the posterior likelihoods of the data conforming to each model. Since the 'cS' 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.

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

.
```

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

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

      seqnames  start    end  width strand AGO6.1 AGO6.2 AGO4.1 AGO4.2
NA.61    >Chr1 789549 944194 154646      *     13     37     29     29
NA.28    >Chr1 372735 423138  50404      *      0      0      0      0
NA.45    >Chr1 552693 587497  34805      *      0      0      0      0
          likes       FDR.      FWER.
NA.61 0.9781566 0.02184336 0.02184336
NA.28 0.9754303 0.02320653 0.04587638
NA.45 0.9753956 0.02367247 0.06935198
```

The highest likelihood expressed but non-differentially expressed regions

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

      seqnames  start    end width strand AGO6.1 AGO6.2 AGO4.1 AGO4.2     likes
NA.14    >Chr1  77151  77519   369      *    176    147    275    131 0.9307905
NA.56    >Chr1 758302 758848   547      *    167    166    269    134 0.8751151
NA.42    >Chr1 550077 550655   579      *      5     11      0     19 0.8663408
         FDR.NDE   FWER.NDE
NA.14 0.06920947 0.06920947
NA.56 0.09704717 0.18545113
NA.42 0.10925117 0.29432304
```

And the highest likelihood differentially expressed regions

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

      seqnames  start    end width strand AGO6.1 AGO6.2 AGO4.1 AGO4.2     likes
NA.48    >Chr1 634297 634350    54      *     65     90     12     17 0.9950597
NA.16    >Chr1 238359 238417    59      *      9      9      0      0 0.9914359
NA.13    >Chr1  76799  77150   352      *      6     21      0      0 0.9784148
             DE      FDR.DE    FWER.DE
NA.48 AGO6>AGO4 0.004940330 0.00494033
NA.16 AGO6>AGO4 0.006752209 0.01346211
NA.13 AGO6>AGO4 0.011696524 0.03475668
```

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

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

Session Info

```
> sessionInfo()

R version 4.4.1 (2024-06-14)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 24.04.1 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3 
LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so;  LAPACK version 3.12.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C              
 [3] LC_TIME=en_US.UTF-8        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       

time zone: Etc/UTC
tzcode source: system (glibc)

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

other attached packages:
 [1] segmentSeq_2.41.0           ShortRead_1.64.0           
 [3] GenomicAlignments_1.43.0    SummarizedExperiment_1.36.0
 [5] Biobase_2.67.0              MatrixGenerics_1.19.0      
 [7] matrixStats_1.4.1           Rsamtools_2.22.0           
 [9] Biostrings_2.75.0           XVector_0.46.0             
[11] BiocParallel_1.41.0         GenomicRanges_1.59.0       
[13] GenomeInfoDb_1.43.0         IRanges_2.41.0             
[15] S4Vectors_0.44.0            BiocGenerics_0.53.1        
[17] generics_0.1.3              baySeq_2.41.0              

loaded via a namespace (and not attached):
 [1] SparseArray_1.6.0       bitops_1.0-9            jpeg_0.1-10            
 [4] lattice_0.22-6          digest_0.6.37           RColorBrewer_1.1-3     
 [7] evaluate_1.0.1          grid_4.4.1              fastmap_1.2.0          
```

```
[10] jsonlite_1.8.9          Matrix_1.7-1            limma_3.63.0
[13] BiocManager_1.30.25     httr_1.4.7              UCSC.utils_1.2.0
[16] codetools_0.2-20        abind_1.4-8             cli_3.6.3
[19] rlang_1.1.4             crayon_1.5.3            BiocStyle_2.35.0
[22] DelayedArray_0.33.1     yaml_2.3.10             S4Arrays_1.6.0
[25] tools_4.4.1             deldir_2.0-4            interp_1.1-6
[28] locfit_1.5-9.10         GenomeInfoDbData_1.2.13 hwriter_1.3.2.1
[31] png_0.1-8               buildtools_1.0.0        R6_2.5.1
[34] zlibbioc_1.52.0         pwalign_1.3.0           edgeR_4.4.0
[37] Rcpp_1.0.13             statmod_1.5.0           xfun_0.48
[40] sys_3.4.3               knitr_1.48              latticeExtra_0.6-30
[43] htmltools_0.5.8.1       rmarkdown_2.28          maketools_1.3.1
[46] compiler_4.4.1
```

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).