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

Thomas J. Hardcastle

October 18, 2010

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(4, "SOCK")
```

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("data", 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)
> aD

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

Slot "alignments":
         chr start end                    tag duplicated
9233   >Chr1     1  22 GTTTAGGGTTTAGGGTTTAGGG       TRUE
206662 >Chr1     3  21    CTAAACCCTAAACCCTAAA       TRUE
9231   >Chr1     5  19        AAACCCTAAACCCTA       TRUE
9232   >Chr1     5  20       AAACCCTAAACCCTAA       TRUE
9234   >Chr1     5  23    AAACCCTAAACCCTAAACC       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":
[1] ">Chr1" ">Chr2"

Slot "chrlens":
[1] 1000000  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, maxgaplen = 500, cl = cl)
> sD
```

```
An object of class "segData"
101866 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
101861 more rows...

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

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

Slot "segInfo":
    chr start end leftSpace rightSpace
1 >Chr1     1  63         0          1
2 >Chr1     1  88         0          1
3 >Chr1     1 113         0        151
4 >Chr1     1 284         0        120
5 >Chr1     1 427         0        172
101861 more rows...
```

We then try and estimate prior parameters on the data with the `getPriors` function. This function constructs a random sample of non-overlapping segments from the segData object `sD` and uses the prior estimation functions from the package `baySeq` to construct a set of prior parameters on the data. At present only the Poisson-Gamma method of prior estimation implemented by `baySeq` is supported as alternative methods require excessive computational time. Additional options can be given to the `getPriors` function to be passed on to the `getPriors.Pois` method of the `baySeq` library.

```
> sDP <- getPriors(sD, type = "Pois", samplesize = 100, perSE = 0.5,
+     maxit = 1000, cl = cl)
```

Having found estimates of the prior parameters of the data, we can then segment the genome based on the information in the `segData` object. The function compares, for each replicate group, the likelihood that the number of alignments matching in a segment is similar to background, or that it is similar to the regions to the left and right of the segment. It then evaluates the likelihood that this similarity occurs in all replicate groups, and ranks all the potential segments in order of decreasing likelihood of similarity. Any segment with a likelihood of similarity greater than the `pcut` argument is discarded at this point. The function then filters the potential segments to form a non-overlapping set of segments by choosing the segment with the lowest likelihood of similarity and discarding any segments which overlap with this. By iteratively discarding overlapping segments, we form a non-overlapping set of segments which have a low likelihood of being similar to background (and are thus regions corresponding to a high density of aligned sequences).

```
> simSegs <- similaritySeg(sDP, pcut = 0.1, estimatePriors = FALSE,
+     verbose = TRUE, cl = cl)
```

We finally acquire an annotated `countData` object, with the annotations describing the co-ordinates of each segment.

```
> simSegs
```

```
An object of class "countData"
326 rows and 4 columns

Slot "data":
    SL9 SL10 SL26 SL32
10   86   65   65  101
63    0   15    0    0
70    0   40  126    4
97  759  706 1553 1672
378   1    0   36    3
393   0    0   16    0
401  31   11   48   74
421  74   59   47   21
431   0    0   12    0
433   0   21   68    2
316 more rows...
```

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

Slot "groups":
list()

Slot "annotation":
      chr start   end         PSame
10  >Chr1     1   967 1.053127e-92
63  >Chr1 11427 11927 2.962147e-05
70  >Chr1 12998 13315 1.008540e-48
97  >Chr1 17055 18728 0.000000e+00
378 >Chr1 27657 28200 1.844603e-12
393 >Chr1 33277 33298 2.278137e-04
401 >Chr1 42217 42806 8.540898e-47
421 >Chr1 44668 44870 5.638646e-58
431 >Chr1 47742 47761 3.557223e-03
433 >Chr1 49056 49222 4.273948e-25
316 more rows...
```

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

```
> plotGenome(aD, simSegs, chr = ">Chr1", limits = c(1, 1e+05))
```

This `countData` object can also 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 submission. 2010.

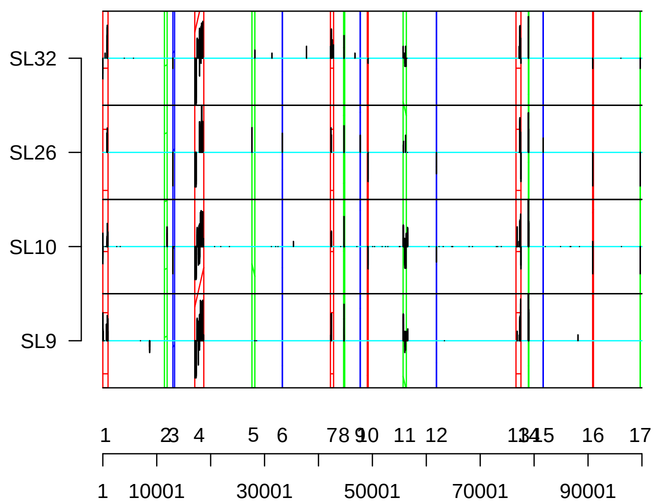

Figure 1: The segmented genome (first 10^5 bases of chromosome 1.