EMBL course on Short Read analysis with Bioconductor: An exercise with coverage vectors

Simon Anders

European Bioinformatics Institute,
Hinxton, Cambridge, UK

sanders@fs.tum.de

23 June 2009

In the ChIP-Seq tutorial, we used the *IRanges* and *ShortRead* packages to get coverage vectors from aligned Solexa reads. The function `coverage` gave us a *GenomeData* object that contained for each chromosome an *Rle* vector of coverage scores.

In this exercise, we explore an interesting analysis that is of use especially in ChIP-Seq experiments studying histone modifications or other characteristics of chromatin structure. In one of the pioneering works in this field, Barski et al. [1] have performed ChIP-Seq for various histone modifications. Among other questions they investigated whether the positioning of histone methylation marks correlates with transcription start sites (TSSs). They could confirm that (as was already knwon before) the mark H3K4me3 peaks sharply at TSSs with a distinctive peak shape.

Here, we will re-do the final step of this analysis.

In order to save time, we have already performed these steps for you:

- Download Barski et al.'s raw data from the NCBI Short Read Archive.
- Use Maq to align the data from the lanes with H3K4me1 and H3K4me3 samples to the human reference genome.
- Use *ShortRead*'s *ReadAligned* function to read in the output of Maq.
- Call the `coverage` method to get the coverage vectors.

The following commands were used for this:

```
> library( ShortRead )
> maps.me1 <- sapply( list.files( "ShortReadExampleData/H3K4me1", "run.*lane.\\.map" ),
+    function(filename)
+       readAligned( "ShortReadExampleData/H3K4me1", filename, type="MAQMapShort" ) )
> maps.me3 <- sapply( list.files( "ShortReadExampleData/H3K4me3", "run.*lane.\\.map" ),
+    function(filename)
+       readAligned( "ShortReadExampleData/H3K4me3", filename, type="MAQMapShort" ) )
> get_coverage <- function( lanes, chrom, chromLength ) {
+    res <- Rle( 0, chromLength )
```

```
+    for ( lane in lanes ) {
+       lc <- lane[ chromosome(lane) == chrom ]
+       cvg <- coverage( lc, start = 1, end = chromLength,
+          extend = 185L - width(lc) )
+       res <- res + cvg[[ chrom ]]
+    }
+    res
+ }
> lengthChr10 <- 135374737
> coverage.chr10.me1 <- get_coverage( maps.me1, "10", lengthChr10 )
> coverage.chr10.me3 <- get_coverage( maps.me3, "10", lengthChr10 )
> save( coverage.chr10.me1, coverage.chr10.me3, file="coverage.H3K4meX.rda")
```

In case you want to do this yourself, you can find the raw data at `http://www.ebi.ac.uk/~anders/ShortReadExampleData`

To save space, we have taken only the vector for chromosome 10 from this data. Load the *ShortRead* package and the R data file `coverage.H3K4meX.rda`.

```
> library("ShortRead")
> load(file.path("..", "data", "coverage.H3K4meX.rda"))
```

We have these two *Rle* vectors:

```
> coverage.chr10.me1

  'numeric' Rle instance of length 135374737 with 1144002 runs
  Lengths:  49865 185 336 185 632 185 1985 50 135 12 ...
  Values :  0 1 0 1 0 1 0 1 2 1 ...

> coverage.chr10.me3

  'numeric' Rle instance of length 135374737 with 1857096 runs
  Lengths:  50124 185 67 185 74 54 131 54 23 185 ...
  Values :  0 1 0 1 0 1 2 1 0 1 ...
```

If you would like to study these tracks in a genome browser, you can save them as Wiggle files by using the *rtracklayer* package. *rtracklayer* expects to get the data as *RangedData*, hence we have to use a coercion method first.

```
> library("rtracklayer")
> coverage.chr10.me1.rd <- as( coverage.chr10.me1, "RangedData" )
> coverage.chr10.me3.rd <- as( coverage.chr10.me3, "RangedData" )
> names( coverage.chr10.me1.rd ) <- "chr10"
> names( coverage.chr10.me3.rd ) <- "chr10"
```

The coercion method cannot know that the data is from "chr10", because this information is not part of the *Rle* vector. (The original call to `coverage` returned a *GenomeData* object, which contaions the chromosome names. However, we have only one of its *Rle* elements here.) Hence, we have to add the chromosome name manually.

Now, we can export the data as wiggle files:

```
> export( coverage.chr10.me1.rd, "chr10_me1.wig", format="wig" )
> export( coverage.chr10.me3.rd, "chr10_me3.wig", format="wig" )
```

Inspect the wiggle files with a text editor. You may find that the chromosome names are wrong, they read "chrchr10". This is a bug in *rtracklayer* that will be corrected in the next few days: The `export` function prefixes an extra "chr" to the chromosome names. To resolve the issue, just omit "chr" in your chromosome name:

```
> names( coverage.chr10.me1.rd ) <- "10"
> names( coverage.chr10.me3.rd ) <- "10"
> export( coverage.chr10.me1.rd, "chr10_me1.wig", format="wig" )
> export( coverage.chr10.me3.rd, "chr10_me3.wig", format="wig" )
```

Now, use your favorite genome browser to display the wiggle files.

You can also use Hilbert curve visualisation. Either, use the stand-alone tool and load the wiggle files, or type

```
> library( HilbertVisGUI )
> hilbertDisplay( as.integer( coverage.chr10.me1 ) )
```

Using HilbertVis from R unfortunately requires us to convert from an *Rle* vector to an ordinary one, and this will require a lot of RAM. (I'll change this soon.) So, it might not work well on a 2 GB machine.

If you have enough RAM, you may also compare the two data vectors:

```
> hilbertDisplay(
+    as.integer( coverage.chr10.me1 ),
+    as.integer( coverage.chr10.me3 ) )
```

In order to see how the methylation marks correlate around the TSSs, we place windows around them and collect the coverage from 200 bp before to 200 bp after the TSS. This gives us a vector of length 401 from each TSS, and by adding them all up, we see whether the histone marks tend to be at a non-random distance to TSSs.

Assume that a TSS is at position 123456. Then, we can get the coverage of the 101-bp vector around it with the `subseq` method:

```
> as.vector( subseq( coverage.chr10.me1, 123456-50, 123456+50 ) )

  [1]  5  5  7  7  7  7  7  7  7  7  7  7  7  7  6  7  8  8  8  8  8  8  8  8  8
 [26]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
 [51]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  9  9  9  9
 [76]  9 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
[101] 10
```

Note that `subseq` returns an *Rle* object. However, our result is a short vector, and for these sizes, it is easy enough to use ordinary vectors. Hence, the call to `as.vector`.

If we have a vector of TSS positions, we can now easily add all these intervals up.

Exercise 1
Write a function to do that. It should take an Rle coverage vector, an integer vector of TSS positions and a boolean vector indicating whether the TSS and its gene are on the reverse strand.

Solution: You might write something like this:

```
> deltaConvolve <- function( coverage, winCentres, revStrand,
+                            left, right)
+ {
+    result <- rep( 0, right - left + 1 )
+    for( i in seq(along=winCentres) ) {
+       v <- as.vector( subseq( coverage, winCentres[i] + left,
+                                         winCentres[i] + right ) )
+       if( revStrand[i] )
+          v <- rev( v )
+       result <- result + v
+    }
+    result
+ }
```

If you like it more elegant, try using `sapply` instead of `for`. This might actually be slower, however.

Exercise 2
Now, we need a list of the transcription start sites of all genes on human chromosome 10. If you are familiar with the biomaRt package, try to get this information from Ensembl.

Solution: The following Biomart query does the job.

```
> library("biomaRt")
> mart <- useMart( "ensembl", dataset = "hsapiens_gene_ensembl" )
> martReply <- getBM( attributes=c( "transcript_start", "transcript_end", "strand" ),
+   filters="chromosome_name", values="10", mart=mart )
```

We now have

```
> head(martReply)
```

```
  transcript_start transcript_end strand
1        104275728      104275989      1
2         21893406       21893512     -1
3        122104417      122104685     -1
4        121517754      121518044      1
5        120810487      120810613     -1
6        120535391      120535465     -1
```

Note that `transcription_start` is always smaller than `transcription_end`, even when the transcript is on the "−" strand. Hence, we have to use either the start or the end coordinate of the transcript, depending on the strand, to get the actual transcription start sites, i.e., the 5' ends of the transcripts:[1]

```
> tss <- ifelse( martReply$strand == 1,
+    martReply$transcript_start,
+    martReply$transcript_end )
```

This allows us to call our function:

[1]The version of this document that we distributed at the course neglected this point. Many thanks to Ludo Pagie who spotted this mistake.

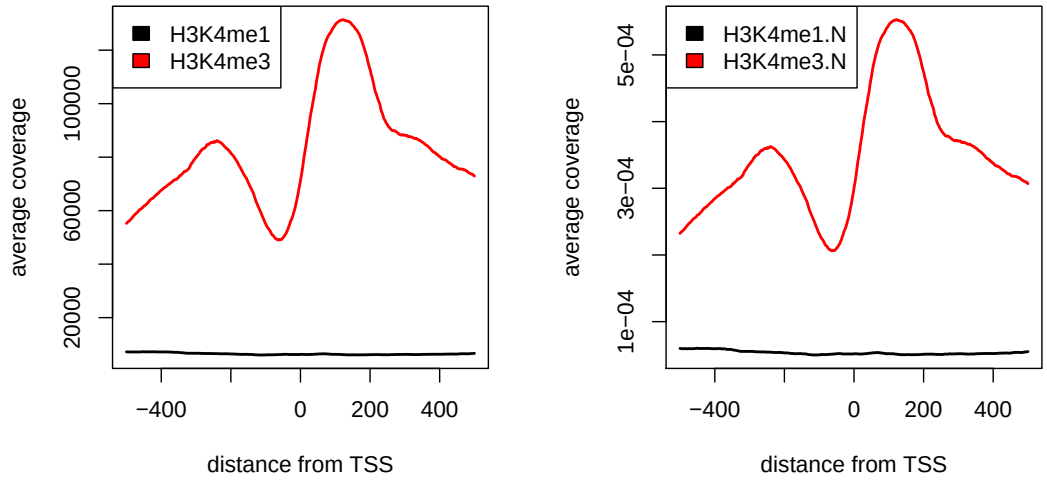

Figure 1: Histone modification signals, averaged over transcripts originating from chromosome 10 and aligned by TSS. Compare to Fig. 2 B and D in [1].

```
> win = c(-500, 500)
> H3K4me1 <- deltaConvolve( coverage.chr10.me1, tss, martReply$strand == -1,
+    left=win[1], right=win[2] )
> H3K4me3 <- deltaConvolve( coverage.chr10.me3, tss, martReply$strand == -1,
+    left=win[1], right=win[2] )
```

We can plot this and get two curves similar to those in Barski et al.'s paper.

```
> plotfun = function( win, ... ) {
+   y = cbind( ... )
+   matplot( win[1]:win[2], y, type='l', lty=1, lwd=2,
+         ylab="average coverage", xlab="distance from TSS" )
+   legend( "topleft", colnames(y), fill=seq_len(ncol(y)) )
+ }

> plotfun( win, H3K4me1, H3K4me3 )
```

The result is shown in the left panel of Figure 1. As we used data from only one chromosome, ours is less smooth than theirs.

It might be more appropriate to normalize the two curves by the respective library sizes:

```
> H3K4me1.N = H3K4me1 / sum(coverage.chr10.me1)
> H3K4me3.N = H3K4me3 / sum(coverage.chr10.me3)

> plotfun( win, H3K4me1.N, H3K4me3.N )
```

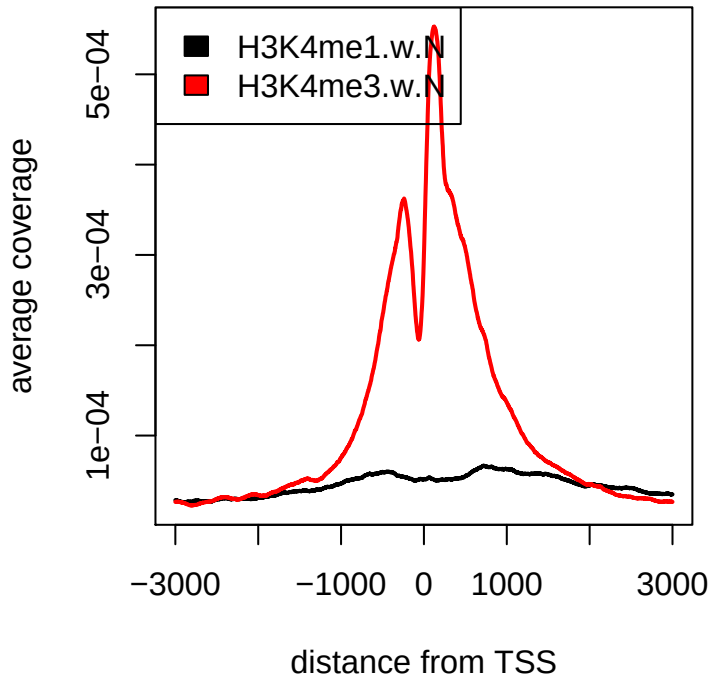

Figure 2: The same plot as in Fig. 1b, but with a wider window around the TSSs.

The result is shown in the right panel of Figure 1.

A wider range of the window might be helpful, too. hence, the same again with a window of width 6,000:

```
> win.w = c(-3000, 3000)
> H3K4me1.w <- deltaConvolve(coverage.chr10.me1, tss, martReply$strand == -1,
+    left=win.w[1], right=win.w[2])
> H3K4me3.w <- deltaConvolve(coverage.chr10.me3, tss, martReply$strand == -1,
+    left=win.w[1], right=win.w[2])
> H3K4me1.w.N = H3K4me1.w / sum(coverage.chr10.me1)
> H3K4me3.w.N = H3K4me3.w / sum(coverage.chr10.me3)
> plotfun( win.w, H3K4me1.w.N, H3K4me3.w.N )
```

The output is shown in Fig. 2.

References

[1] A. Barski, S. Cuddapah, K. Cui, T.-Y. Roh, D. E. Schones, Z. Wang, G. Wei, I. Chepelev, K. Zhao. High-resolution proling of histone methylations in the human genome. *Cell* 129:823–837 (2007)