Analysing CpG comethylation via bisulfite sequencing

Peter Hickey

November 15, 2011

Contents

Extend to non-neighbouring CpGs and non-CpG methylation

1 Statistical framework

I consider each **pair** of **neighbouring** CpG dinucleotides in the reference genome. Denote a pair of neighbouring CpGs by $(\mathrm{X}_{c,p_1}, \mathrm{X}_{c,p_2})$, where the subscript c denotes the chromosome and subscripts p_1 and p_2 denote the position of the cytosine in each CpG; the pairs are ordered so that $p_1 < p_2$. In what follows I consider a single pair of CpGs and simplify the notation to $X = (X_1, X_{1+d})$ where $d > 1$ is the distance between the cytosine positions in each CpG.

We are interested in studying the *comethylation* of neighbouring CpGs. For each DNA fragment the methylation status at a CpG is a binary outcome (M = methylated or U = unmethylated). Statistically this means we are interested in studying correlated binary variates. We are also interested in studying how this correlation varies with d and how the correlation changes according to the region of the genome (e.g. CpG Islands, promoter regions of genes, etc.).

For each pair X, we tabulate the number of reads that report methylation at each CpG. If we were to only count reads that overlapped **both** CpGs then we would get something like Table 1. If we instead count all reads that overlap at least one of the CpGs in the pair then we would get a table like Table 2[1]. Table 2 will give better estimates of the marginal probabilities, i.e. the probability of X_1 or X_{1+d} being methylated. Table 1 is a sub-table of Table 2.

Table 2 includes information we probably don't care about, such as the number of reads in the sample that don't overlap either CpG_1 or CpG_2; this value (entry n_{33}) will be overwhelmingly large for every table.

[1] Actually this table considers all reads regardless of any overlap. This is discussed later.

Table 1: A 2×2 contingency table of the reads overlapping a pair of neighbouring CpGs. M = methylated, and U = unmethylated. This table only includes reads that overlap both CpGs in the pair.

		CpG_2		
		M	U	Total
CpG_1	M	η_{11}	η_{12}	η_{10}
	U	η_{21}	η_{22}	η_{20}
	Total	η_{01}	η_{02}	η

Table 2: A 3×3 contingency table of all reads and whether each read overlaps the CpG pair. M = methylated, U = unmethylated, and $-$ = read does not overlap the CpG.

		CpG_2			
		M	U	$-$	Total
CpG_1	M	n_{11}	n_{12}	n_{13}	n_{10}
	U	n_{21}	n_{22}	n_{23}	n_{20}
	$-$	n_{31}	n_{32}	n_{33}	n_{30}
	Total	n_{01}	n_{02}	n_{03}	n

2 Programming strategy

Create CpG_fw.bed and CpG_rev.bed?

Constructing the CpG tables is not a straightforward programming task. It involves the following steps:

1. Define X
2. Identify all reads overlapping at least one of the CpGs in X.
   (a) Reads aligned to the positive strand must overlap the cytosine in the CpG.
   (b) Reads aligned to the negative strand must overlap the guanine in the CpG.
3. Tabulate the methylation status of each read.
   (a) For reads aligned to the forward strand, $C \Rightarrow M$ and $T \Rightarrow U$ at a CpG.
   (b) For reads aligned to the reverse strand, $G \Rightarrow M$ and $A \Rightarrow U$ at a CpG.

The Illumina data have been aligned with `Bismark` and then converted to the bam format using `bismark2SAM_v5_xm.pl`[2] and `SAMtools`. The methylation string is stored in the `XM` tag of the bam file. I wrote a function to read the bam file into R. It requires the bam file to be indexed and assumes that the reads are aligned to the hg19 reference genome, though this requirement is easily modified. The output if a `GRanges` object.

```
> library(Rsamtools)
> library(BSgenome)
> library(BSgenome.Hsapiens.UCSC.hg19)
> readBismarkBam <- function(x) {
+     bam.param <- ScanBamParam(what = c("rname", "strand", "pos",
+         "qwidth"), tag = "XM")
+     bam <- scanBam(file = x, index = x, param = bam.param)
+     bam[[1]]$rname <- as(bam[[1]]$rname, "character")
+     bam[[1]]$strand <- as(bam[[1]]$strand, "character")
+     gr <- GRanges(seqnames = bam[[1]]$rname, strand = bam[[1]]$strand,
+         ranges = IRanges(start = bam[[1]]$pos, width = bam[[1]]$qwidth),
+         XM = bam[[1]]$tag$XM)
+     seqlevels(gr) <- seqlevels(Hsapiens)
+     seqlengths(gr) <- seqlengths(Hsapiens)
+     gr
+ }
> readBismarkBam("~/Desktop/Comethylation/sandpit/SRR206931_bismark_sorted.bam")

GRanges with 2888270 ranges and 1 elementMetadata value
          seqnames               ranges strand   |
             <Rle>            <IRanges>  <Rle>   |
```

[2] Available from `http://www.bioinformatics.bbsrc.ac.uk/projects/download.html#bismark`.

```
       [1]     chr1     [545158, 545237]      +   |
       [2]     chr1     [545945, 546024]      +   |
       [3]     chr1     [546012, 546091]      +   |
       [4]     chr1     [546012, 546091]      +   |
       [5]     chr1     [546013, 546092]      +   |
       [6]     chr1     [546013, 546092]      +   |
       [7]     chr1     [546013, 546092]      +   |
       [8]     chr1     [546013, 546092]      +   |
       [9]     chr1     [546013, 546085]      +   |
       ...      ...                  ...    ... ...
[2888262]     chrY [59018435, 59018514]      +   |
[2888263]     chrY [59018435, 59018514]      +   |
[2888264]     chrY [59018435, 59018514]      +   |
[2888265]     chrY [59018435, 59018514]      +   |
[2888266]     chrY [59018435, 59018514]      +   |
[2888267]     chrY [59018435, 59018514]      +   |
[2888268]     chrY [59028112, 59028191]      +   |
[2888269]     chrY [59028113, 59028192]      +   |
[2888270]     chrY [59241190, 59241269]      -   |
                                                                                XM
                                                                       <character>
       [1] .....x.h.....Z.Z.h.Z..Zx..Z...xh...xhh.............x....Z..Z...x..x......x.h....
       [2] hx...hx...Z..xZ..Z...z.Z.....h.x......x..x...........x..................hx.....
       [3] hhx...hx...Z..xZ..Z.h.Z.Z.....h.x.....hx......Z..Z.x..x.............hhx.........
       [4] hhx...hx...Z..xZ..Z.h.Z.Z.....h.x.....hx..x......Z....x.............hx....x....
       [5] hx...hx...Z..xZ..Z...Z.Z.....h.x.....hx......Z.......................hx..........
       [6] hx...hx...Z..xZ..Z...z.z.....h.x.....hx..x...Z.....................hhx....x.....
       [7] hx...hx...Z..xZ..Z...Z.z.....h.x.....hx..x...Z.....................hhx....x...Z.
       [8] hx...hx...Z..xZ..Z...Z.Z.....h.x......x.............................hx....x.....
       [9]        hx...hx...Z..xZ..Z.h.Z.z.....h.x......x..x...Z.......x.............hhx...
       ...                                                                             ...
[2888262] ............hh.................hxz........h.........h.h........h...Z.h..........
[2888263] ............hh.................hxz........h.........h.h........h...Z.h..........
[2888264] ............hh.................hxZ........h.........h..........h...Z.h..........
[2888265] ............hh.................hxz........h.........h.h........h...Z.h..........
[2888266] ............hh.................hxZ........h.........h.h........h...Z.h..........
[2888267] ............hH.................hxZ........h.........h.h........h...Z.h..........
[2888268] ...x..h..x......x.....h.h...........hh.............................x...........Z.x.
[2888269] ..x..h..x......x.....h.h...........hh.............................x...........Z.x..
[2888270] .......h.h.hhh.h.hhh.h.hhhhh.hhhhh.....h.h.hhhh.z...h.hhh.h.hhh.h.h...hh.z.....h

seqlengths
      chr1       chr2 ... chrUn_gl000249
 249250621  243199373 ...          38502
```

`readBismarkBam()` is slightly worse than linear in speed vs. size of bam file — a 182MB bam took an average 11 seconds to process and a 912MB bam took on average 79 seconds to process, with $n = 5$ runs per file (NB: the first run was the slowest by 30 seconds for the 912MB bam file or 10× longer than the first run of the 182MB file).

Todo list