ShortRead Lab: Working with Aligned Sequences

Martin Morgan

29 January 2010

This lab takes a quick tour through the ShortRead package.

Exercise 1
Load the ShortRead package.

The data we are intersted in is the output from the Bowtie alignment software. It is located at

```
> bowtieFile <- system.file("extdata", "BYe9.head.map",
+     package = "day3")
```

Use the `readAligned` *function in ShortRead to input the data. This function can take two arguments, the file name, and an argument* `type` *specifying the type of the file to be input. Use* `type="Bowtie"`*. What class is this object? How would you find help about it? How many reads are in this file?*

Exercise 2
Let's explore the data a little.

The command `strand` *can be used to extract the strand information from the AlignedRead object. However, we do not want to display all the strand information to the screen. The function* `table` *tallys the number of times an element occurs in a vector. Thus*

```
> table(c("a", "a", "b", "a", "b", "c"))
```

```
a b c
3 2 1
```

Use `strand` *and* `table` *to summarize which strands reads align to.*

The read sequences and quality scores are available using the functions `sread` *and* `strand`*. Take a look at this information. What class is used to represent the reads? What can you do with those reads?*

Exercise 3
The `alphabetByCycle` *function takes a DNAStringSet object and returns a matrix summarizing the number of times each letter in the DNA alphabet (the IUPAC alphabet, which includes letters to represent redundancies) occurs at each cycle. Use* `alphabetFrequency` *on the result of* `sread` *to summarize how often each nucleotide occurs. Subset the result to include only those letters corresponding to the nucleotides A, C, G, and T.*

Exercise 4

The `matplot` *function takes a matrix and plots each column, using the row index as the x-coordinate and the column as the y-coordinate. We want to see how nucleotide use changes with cycle. To do this we need to transpose the matrix returned by* `alphabetByCycle`*, and then use* `matplot`*. Set the* `type` *equal to* `"l"`*.*

Exercise 5

Fastq quality scores are an encoded representation of how confidently the baes are called. ShortRead *provides a* coercion *function to convert the quality scores to a numerical matrix representation; for the quality scores that we have, it makes sense to substract 33 from all elements of the matrix. Thus*

```
> m <- as(quality(aln), "matrix") - 33
```

gives a matrix of quality scores. What are the dimensions of `m`*? Can you guess at what the rows and columns represent? Use* `colMeans` *to calculate the average of the column means. What does this represent? Use* `plot` *with the results of the* `colMeans` *to display your result.*

Exercise 6

The chromosomes information is presented differently from how it is represented in other resources we will use later in the course. Use `table` *and the accessor* `chromosome` *to summarize how many reads align to each chromosome.*

```
> table(chromosome(aln))
```

```
   chrI   chrII  chrIII   chrIV    chrV   chrVI
   7372   29844    9641   41084   18322    7571
 chrVII chrVIII   chrIX    chrX   chrXI  chrXII
  30037   16682    8141   18056   17239  562222
chrXIII  chrXIV   chrXV  chrXVI
  26811   24655   27330   26763
```

We do not want to include the mitochondrial chromosome in subsequent analyses. Create a logical vector `chromosome(aln) != "chrmt_S288C"` *and use this to subset the* `AlignedRead` *object.*

```
> aln <- aln[chromosome(aln) != "chrmt_S288C"]
```

Now some tricky stuff! The chromosome infomration is stored as a factor, and the levels look like `chr04_S288C`*. We want the levels to be like* `chrIV`*. The first task is to extract the 'numbers' 04 from the original levels. We'll use the* `sub` *function. The first argument to* `sub` *is a* regular expression*. The regular expression we'll use is* `"chr(.+)_S288C"`*. The* `chr` *says 'start matching when the exact character sequence chr occurs'. Skipping the parentheses for a second, The* `.+` *says 'continue matching any character (the* `.`*) one or more times (the* `+`*)'. Then we are required to match* `_S288C`*. So* `chr04_S288C` *is matched in three steps:* `chr` *in the regular expression matching 'chr',* `.+` *matching '04', and* `_S288C` *matching '_S288C'. The parentheses* `(.+)` *say to remember the string matching the pattern inside the parentheses, i.e., remember '04'. The lines*

```
> chrom <- factor(chromosome(aln))
> levels(chrom)

 [1] "chrI"    "chrII"   "chrIII"  "chrIV"   "chrV"
 [6] "chrVI"   "chrVII"  "chrVIII" "chrIX"   "chrX"
[11] "chrXI"   "chrXII"  "chrXIII" "chrXIV"  "chrXV"
[16] "chrXVI"

> i <- sub("chr(.+)_S288C", "\\1", levels(chrom))
> i

 [1] "chrI"    "chrII"   "chrIII"  "chrIV"   "chrV"
 [6] "chrVI"   "chrVII"  "chrVIII" "chrIX"   "chrX"
[11] "chrXI"   "chrXII"  "chrXIII" "chrXIV"  "chrXV"
[16] "chrXVI"
```

extract the chromosome from the `AlignedRead`*, and then for each level, substitutes any string matching* `"chr(.+)_S288C"` *with the second argument,* `"1"`*. In the second argument,*
`1` *represents the first remembered (i.e., enclosed in parentheses) argument. Surprisingly, R has a function* `as.roman` *which converts things that look like integers into roman numerals. We can paste these together with* `chr` *to get the levels we want, then update the levels on* `chromosome`*:*

```
> lvls <- paste("chr", as.roman(i), sep = "")
> lvls

 [1] "chrNA" "chrNA" "chrNA" "chrNA" "chrNA" "chrNA"
 [7] "chrNA" "chrNA" "chrNA" "chrNA" "chrNA" "chrNA"
[13] "chrNA" "chrNA" "chrNA" "chrNA"

> levels(chrom) <- lvls
```

ShortRead allows the AlignedRead object to be updated by a call like

```
> aln <- initialize(aln, chromosome = chrom)
```

Put all these pieces together to recode the chromosome levels and update the `AlignedRead` *object*

Exercise 7
ShortRead can produce a quality assessment report from a collectio n of files. The report is produced in two stages. The first phase visits each file and accumulates statistics. This is a long and memory-intensive phase, and we will not do this in class. The basic commands are.

```
> bowtieDir <- "/path/to/alignments"
> qa <- qa(bowtieDir, ".*map$", type = "Bowtie")
```

The result of this operation for the four Bowtie files representing the complete experiment is availabe in the day3 *package, using.*

```
> data("qa_caudy_28_jan_2009")
```

The second phase takes the accumulated statistics and produces an html report. Do this with

```
> rpt <- report(qa)
> browseURL(rpt)
```