Lab: ShortRead

Martin Morgan

29 July, 2008

1 Input and quality assessment

This part of the lab explores navigation, input, and quality assessment of Solexa reads.

Exercise 1
This exercise suggests ways of effectively exploring files produces by the Solexa pipeline.

Load the ShortRead library, and use `SolexaPath` *to point to the mini-experiment included in the lab. You'll want the first argument to end at the directory called* `Solexa` *in the* `extdata` *folder. Display the object and summarize the type of files found in each directory. Use* `readLines` *and* `strsplit` *to look at the first line of common file types. The following function might be useful to look at file content:*

```
> peek <- function(dirPath, file, lines = 1,
+     sep = "\t") {
+     filepath <- list.files(dirPath, file,
+         full = TRUE)
+     unlist(strsplit(readLines(filepath, lines),
+         sep))
+ }
```

The specific path, below, depends on where you copied files.

```
> library(ShortRead)
> sp <- SolexaPath("../extdata/Solexa")
> sp
```

```
class: SolexaPath
experimentPath: ../extdata/Solexa
dataPath: Data
scanPath: NA
imageAnalysisPath: C1-35Firecrest
baseCallPath: Bustard
analysisPath: GERALD
```

```
> head(peek(imageAnalysisPath(sp), "s_1_0001_int.txt"))

[1] "1"
[2] "1"
[3] "109"
[4] "548"
[5] " 409.0  504.5  475.0 11120.8"
[6] " 880.8 3231.2  464.8 7933.4"

> peek(baseCallPath(sp), "s_1_0001_seq.txt")

[1] "1"
[2] "1"
[3] "109"
[4] "548"
[5] "TTGTTTTCATGTGATTTTAAAAATGTATTTGTTTGT"

> head(peek(baseCallPath(sp), "s_1_0001_prb.txt"))

[1] " -40  -40  -40   40" " -40  -40  -40   40"
[3] " -40  -40   40  -40" " -40  -40  -40   40"
[5] " -40  -40  -40   40" " -40  -40  -40   40"

> peek(analysisPath(sp), "s_1_sequence.txt",
+     8)

[1] "@HWI-EAS88_1:1:1:109:548"
[2] "TTGTTTTCATGTGATTTTAAAAATGTATTTGTTTG"
[3] "+HWI-EAS88_1:1:1:109:548"
[4] "YYYYYYYYYYYYTYYYYYYYYYYYYYYYYHVVVH"
[5] "@HWI-EAS88_1:1:1:105:517"
[6] "TCCAAACTGGTAGACAATACAAACATTCTCAAATC"
[7] "+HWI-EAS88_1:1:1:105:517"
[8] "YYYYYYYYYYYYYYYYYYYYYYYYYYYYYVVVVV"

> peek(analysisPath(sp), "s_1_export.txt")

 [1] "HWI-EAS88"
 [2] "1"
 [3] "1"
 [4] "1"
 [5] "98"
 [6] "349"
 [7] ""
 [8] ""
 [9] "TTTCAGTTTTCTCGCCTTATTCCATGTCCTACAGT"
[10] "YYYYYYYYYYYYYYVYYYYYYYYYYYUYYYYVVVKV"
[11] "chrUn_random.fa"
```

```
[12] ""
[13] "5251632"
[14] "F"
[15] "16A18"
[16] "3"
[17] ""
[18] ""
[19] ""
[20] ""
[21] ""
[22] "Y"
```

Consult Solexa documentation for file content details. We focus mostly on `export, sequence, seq, prb` in the lab.

Exercise 2

This exercise shows how to access and manipulate the data structures returned by `readAligned`.

Use `readAligned` *to read the* `s_1_export.txt` *file in to R. Use* `strand` *and* `table` *to summarize the number of reads aligned to each strand, and not aligned at all.*

Use `alignData` *to create the subset of reads passing Solexa filtering. How many reads pass filtering? How many of these map to each strand?*

Use `sread` *and* `quality` *to extract the short reads and their qualities from the reads passing filtering. Interpret, perhaps with your neighbor, the meaning of the quality scores. Which bases are 'good', which bad?*

Use `as` *to convert the fastq quality scores to a matrix.*

```
> aln <- readAligned(sp, "s_1_export.txt")
> table(strand(aln))

          F     R
50705 24661 24634

> filtered <- aln[alignData(aln)[["filtering"]] ==
+     "Y"]
> filtered

class: AlignedRead
length: 59975 reads; width: 35 cycles
chromosome: chrUn_random.fa chr1.fa ... NM 255:255:255
position: 5251632 163068613 ... NA NA
strand: F R ...
alignQuality: NumericQuality
alignData varLabels: run lane ... y filtering
```

```
> table(strand(filtered))

          F     R
18866 20513 20596

> sread(filtered)

  A DNAStringSet instance of length 59975
        width seq
    [1]    35 TTTCAGTTTTCTCGCCTTATTCCATGTCCTACAGT
    [2]    35 TAGACTGCTGCCTAGCAAGCCTTAAGGATTCTTCT
    [3]    35 AAAATGAGAAACATCCACTTGACTCCTTGAAAAAT
    [4]    35 TGGGCTGACGTCATGCCTGAGCTGTCACGAGCAGA
    [5]    35 GCGAGGAAAACTGAAAAAGGTGGAAAATTTAGAAA
    [6]    35 GATGAACAAGAGTTTACCAAAAGGTCAAAATGAAA
    [7]    35 GAAAAATGAGAAATGCACACTGTAGGACCTGGAAT
    [8]    35 GTTTTAGGAACTCTCCTTATAGAAGAAACAACTCG
    [9]    35 ACCACCATCATCGATGTGCCCGGTTTACAGGTCTT
    ...   ... ...
[59967]    35 ACTCACTCCTAGTCACTGCCCATTCTCGGGTCGTA
[59968]    35 CAAGAGAAAGCCACAGTACTCCACCAGTGTCTCAT
[59969]    35 TCGCCTATGCTCCGAGCTCCGCTCAGGGTGGTCTC
[59970]    35 CATGCGCGACGACGCGGTCGTGACCATCCGCCAAC
[59971]    35 CGGGATTGGGGGCGGGGTGCCTTCGGGAAGCCCGA
[59972]    35 TCAGGGGGCCTTGCGCCCACCTTTGGCTGTGAGCG
[59973]    35 AGACTGCAGCTCTTCCCACCAAGGGCGAGAGTTAT
[59974]    35 TAACTCAGCCAGCCACAAAGAGAAACAACCAAACC
[59975]    35 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

> quality(filtered)

class: SFastqQuality
quality:
  A BStringSet instance of length 59975
        width seq
    [1]    35 YYYYYYYYYYYYYVYYYYYYYYYYYUYYYYVVVKV
    [2]    35 YYYYYYYYYYYYYYYYYYYYYYYYYYYYYVVVVV
    [3]    35 YYYYYYYYYYYYYYYYYYYYYYYYYYYYYVVVVV
    [4]    35 YYYYYYYYYYYYYYYYYYYYYYYYYYYYYVVVVV
    [5]    35 YYYYYYYYYYYYYYYYYYYYYYYYYYYYYVVVVV
    [6]    35 YYYYYYYYYYYYMYYYYYYYYYYYYYYYSSVUU
    [7]    35 YYYYYYYYYYYYYYYYYYYYYYYYYYYYVVQUV
    [8]    35 YYYYYYYYYYYYYYYYYYYYYYYYYYYYUVVVQ
    [9]    35 YYYYYYYYYYYYYYYYYYYYYYYYYYYYSVVVV
    ...   ... ...
[59967]    35 YYYOYYYYYYYYYYYYYYYYYYYYYOYGNPNN
[59968]    35 YYYYPYYYYYYYYYYDPYJYPKDYCYCYYPTGTR
```

```
[59969]    35 YYYYYYYYYYYYYYYYYYYYYYYYYYYYVSVVV
[59970]    35 YYYYYYYYYYYYYYYYYYYYYYYYYYYRQSHQQ
[59971]    35 YYYYYYYYYYYYYYYYYYYYYYYYYYFYYQQQQD
[59972]    35 YYYYYYYYYYYYYYYYYYYYYYYYYYYYQQQSQ
[59973]    35 YYYYYYYYYYYYYYYYYYYYYYYYYYYYVVJSV
[59974]    35 YYYYYOYYYYYYYYYYFFYOYYYDYYFYOYPCNPP
[59975]    35 YYYYYYYYYYYYYYYYYYYYYYYYYYYYYUVUVU

> m <- as(quality(filtered), "matrix")
> dim(m)

[1] 59975    35

> m[1:3, 1:15]

     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]   25   25   25   25   25   25   25   25   25
[2,]   25   25   25   25   25   25   25   25   25
[3,]   25   25   25   25   25   25   25   25   25
     [,10] [,11] [,12] [,13] [,14] [,15]
[1,]    25    25    25    25    22    25
[2,]    25    25    25    25    25    25
[3,]    25    25    25    25    25    25
```

Exercise 3

This exercise introduces functions for calculating quality assurance measures, and for generating a pdf report from the measures.

Use `qa` *to perform quality assessment on the Solexa path. Access the* `readCounts` *element of the result to see how many reads are in the* `s_1_export.txt` *file.*

Either run the `report` *function on the result of* `qa` *or, better, open the file* `qa_080623_080728.pdf` *using a pdf viewer. Discuss with your neighbors various aspects of the report. Are the numbers (e.g., of reads, of uncalled nucleotides) typical? What's the interpretation of figure in the section titled 'Read distribution: occurrences'? Any ideas on what the* `frequentSequences` *represent?*

What additional information would make the qa reports more informative?

```
> assess <- qa(sp)
> assess

class: SolexaExportQA(9)
QA elements (access with qa[["elt"]]):
  readCounts: data.frame(1 3)
  baseCalls: data.frame(1 5)
```

```
  readQualityScore: data.frame(1536 4)
  baseQuality: data.frame(94 3)
  alignQuality: data.frame(74 3)
  frequentSequences: data.frame(150 4)
  sequenceDistribution: data.frame(69 4)
  perCycle: list(2)
    baseCall: data.frame(175 4)
    quality: data.frame(589 5)
  perTile: list(2)
    readCounts: data.frame(900 4)
    medianReadQualityScore: data.frame(900 4)
```

```
> assess[["readCounts"]]
```

```
                 read filtered aligned
s_1_export.txt 100000    59975   49295
```

The remainder of the questions are really for discussion. The numbers are fairly typical of recent runs that I have seen.

The 'occurrences' figure shows that many reads are represented just once; these are likely to be sequencing or other errors in Solexa. Some reads are represented a great many times. These are likely to be artifacts of sample preparation or sequencing. The 'interesting' reads are typically those with intermediate representation.

Many of the `frequentSequences` are (near to) the Solexa adapter or primer sequences, or poly-A or other reads that likely represent. These present significant opportunities for data shaping before further analysis, and upstream quality control.

2 Manipulation

This part of the lab introduces essential tools for housekeeping and other exploratory investigation of short reads in R.

Exercise 4
This exercise illustrates how to obtain a high-level summary of the sequences present in objects defined by ShortRead.

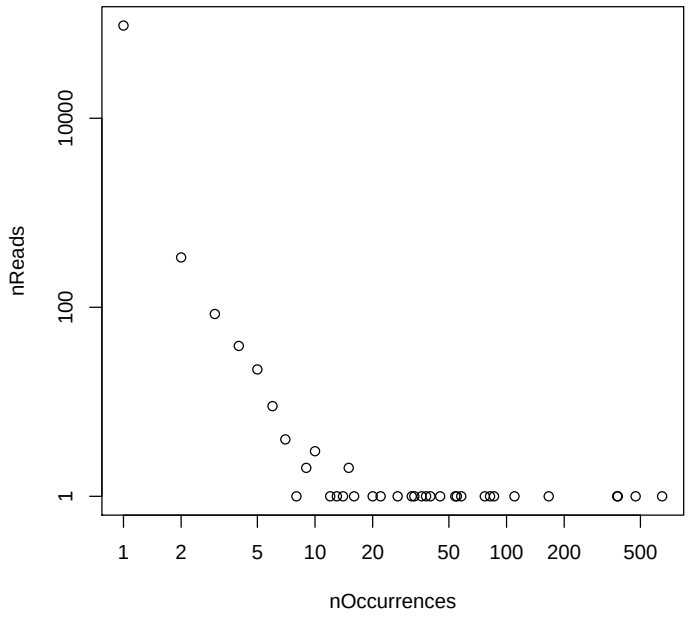
Use `tables` *and* `sread` *on the result of* `readAligned` *(from earlier in the lab) to create a list of common reads, and a table of occurrences of all reads.*

Display the name and count of the most common reads (i.e., the `top` *element of the result of* `tables`*). From the* `distribution` *element, plot the number of reads as a function of number of occurrences, using* `plot` *with option* `log="xy"`*.*

```
> tbls <- tables(sread(aln))
> head(tbls[["top"]])
```

```
GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGAA
                                   649
GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTAGA
                                   472
ANNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
                                   381
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
                                   378
GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGGA
                                   166
GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTTGA
                                   110

> with(tbls$distribution, plot(nReads ~ nOccurrences,
+     log = "xy"))
```



Exercise 5

This exercise shows how columns of DNA strings can be read into R objects useful for further analysis.

Use `readLines` *to read the first two lines of* `s_5_0001_seq.txt` *in the* `baseCallPath` *of the* SolexaPath *object.*

Use `readXStringColumns` *with argument* colClasses *equal to* `c(rep(list(NULL), 4), "DNAString")` *to read the reads into R.* `readXStringColumns` *returns a list, and you will want the first element of the list.*

```
> filepath <- file.path(baseCallPath(sp), "s_5_0001_seq.txt")
> strsplit(readLines(filepath, 2), "\t")
```

```
[[1]]
[1] "5"
[2] "1"
[3] "112"
[4] "532"
[5] "TTTGTCATTGTGAGCATTTTCATCCCGAAGTTGCGG"

[[2]]
[1] "5"
[2] "1"
[3] "101"
[4] "541"
[5] "TACCATGAACAAAATGTGACTCATATCTAAACCAGT"
```

```
> colClasses <- c(rep(list(NULL), 4), "DNAString")
> ln5 <- readXStringColumns(baseCallPath(sp),
+     "s_5_0001_seq.txt", colClasses = colClasses)[[1]]
```

Exercise 6

This exercise illustrates important tools for identifying and removing reads with particular sequences or other characteristics.

Use `tables` *and* `srdistance` *to identify reads that are an edit distance of less than or equal to 4 to the most common sequence. How many of these reads are there?*

Use `alphabetFrequency` *with it's* `baseOnly=TRUE` *argument to identify reads with more than 30 A nucleotides. How many reads are there satisfying this criterion? What about other nucleotides?*

Remove all of these reads (adapater-like, poly-A, C, G, T) and reads with one or more uncalled bases. Run `tables` *again and look at the most common reads.*

Some analyses may warrant removal of duplicated reads, e.g., because many of these might represent artifacts of the short read methodology rather than biological signal. Use `srduplicated` *to select a single copy of all reads.*

```
> tbls <- tables(ln5)
> topSeq <- names(tbls$top)[[1]]
> dist <- srdistance(ln5, topSeq)[[1]]
> sum(dist <= 4)
```

```
[1] 186

> ln5.1 <- ln5[dist > 4]
> alf <- alphabetFrequency(ln5.1, baseOnly = TRUE)
> poly <- alf[, 1:4] > 30
> colSums(poly)

A C G T
2 5 0 0

> noN <- alf[, "other"] == 0
> sum(noN)

[1] 29000

> ln5.2 <- ln5.1[rowSums(poly) == 0 & noN]
> head(tables(ln5.2)[["top"]])

CGGTAGGTTTTCTGCTTAGGAGTTTAATCATGTTTC
                                   9
GGACTGTGTGACTATTGACGTCCTTCCTCGTACGCC
                                   9
TTCTTCGGCACCTGTTTTACAGACACCTAAAGCTAC
                                   9
AACGTTATATTTTGATAGTTTGACGGTTAATGCTGG
                                   8
ATTGTATGTTTTCATGCCTCCAAATCTTGGAGGCTT
                                   8
CCGAGGGTCGCAAGGCTAATGATTCACACGCCGACT
                                   8

> ln5.3 <- ln5.2[!srduplicated(ln5.2)]
> length(ln5)

[1] 30302

> length(ln5.1)

[1] 30116

> length(ln5.2)

[1] 28993

> length(ln5.3)

[1] 20771
```

Exercise 7

This exercise shows how cycle-specific effects can be investigated; approaches like those outlined below can also be used to summarize cycle-specific quality scores.

Use `alphabetByCycle` *to determine the frequency of each nucleotide, at each cycle.* `alphabetByCycle` *returns a matrix with rows representing (IUPAC) nucleotides and columns representing cycles.*

Uncalled bases are recorded in this file as `-`*. Plot the frequency of uncalled bases as a function of cycle number.*

Use standard matrix manipulation to Create a compact version of the `alphabetByCycle` *matrix containing only the nucleotides A, C, G, and T. Let's call this matrix* `abc1`

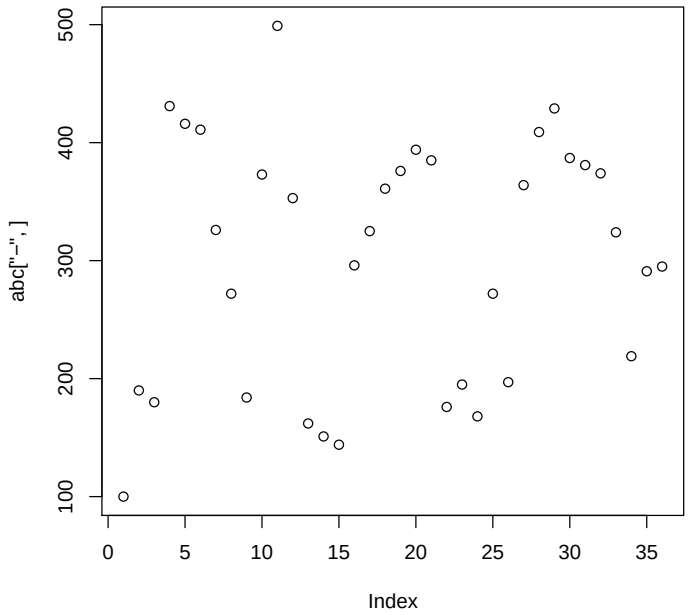
Many exploratory Solexa analyses can be plotted effectively using the *lattice*. *lattice works best when presented with a data frame with data frame columns indicating the level of the corresponding data frame row. Make a data frame from the result of* `alphabetFrequency`*. The columns of the data frame might be:*

1. *The count of each nucleotide,* `as.vector(abc1)`.
2. *The corresponding nucleotide,* `c("A", "C", "G", "T")`.
3. *The cycle,* `as.vector(col(abc1))`

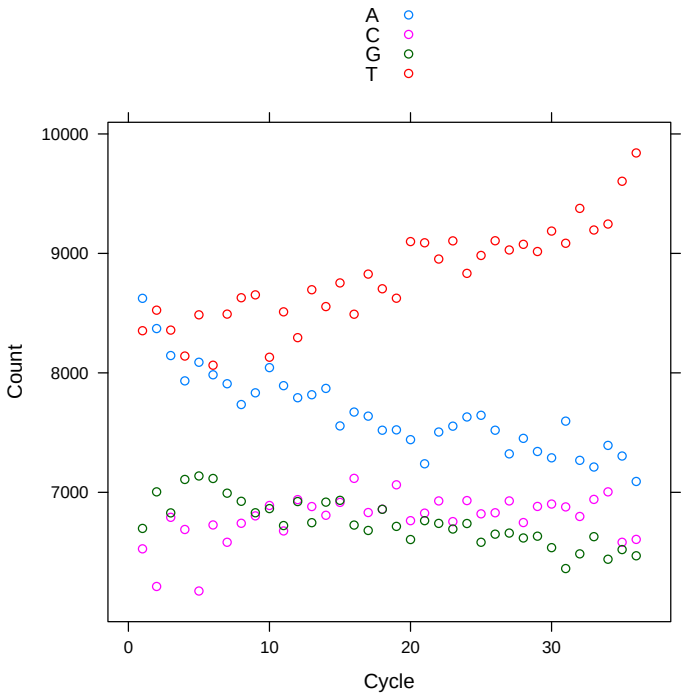
Use the data frame to plot nucleotide frequency as a function of cycle. The incantation might involve the formula `count ~ cycle` *and the argument* `group=nucleotide`.

```
> abc <- alphabetByCycle(ln5)

> plot(abc["-", ])
```




```
> abc1 <- abc[1:4, ]
> abcDf <- data.frame(Count = as.vector(abc1),
+     Nucleotide = c("A", "C", "G", "T"), Cycle = as.vector(col(abc1)))
> print(xyplot(Count ~ Cycle, group = Nucleotide,
+     abcDf, auto.key = TRUE))
```


A
C
G
T
10000
9000
8000
7000
Count
0
10
20
30
Cycle