Bioconductor tools to engage short reads

Martin Morgan (mtmorgan@fhcrc.org)
Fred Hutchinson Cancer Research Center
Seattle, WA, USA

October 5, 2009

Abstract

Bioconductor offers a diverse, flexible, and growing set of tools for analysis of short read sequence data. This tutorial provides a very quick tour of facilities for input, exploration, quality assessment, analysis, annotation, and export of short read data, using a subset of the CAMDA 2009 test data as a running example. Participants will benefit most if they come with a working, current version of R, with the ShortRead package installed following instructions at http://bioconductor.org/install

1 Introduction

R is an open-source, open development programming environment particularly suited to statistical analysis; it is available for all major operating systems. R is extensible through user-contributed packages, of which there are close to 2000 available. R has excellent graphics capabilities, and has facilities for accessing diverse resources (e.g., SQL data bases, XML parsing, graph rendering). Bioconductor is a project based on R, but focusing on analysis of high-throughput genomic data.

A typical R or Bioconductor session involves the command line, with variables defined and commands evaluated at a command prompt.

```
mtmorgan@gladstone:~> R

> x <- 1:10
> mean(x)

[1] 5.5

> library(IRanges)
```

Users often have a front end text editor that is capable of sending chunks of script to the R evaluator. Find help on functions or packages with

```
> ?mean
> library(help=IRanges)
```

A number of Bioconductor packages are available for analysis of short reads. Some of the packages contributed by the core group at the Fred Hutchinson Cancer Research Center include:

IRanges Representing and manipulating data as ranges (e.g., regions of the genome covered by aligned reads).

Biostrings sequence-based pattern matching and manipulation.

ShortRead I/O and quality assessment of short reads.

BSgenome Representing genome-scale data, e.g., BSgenome.Hsapiens.UCSC.hg18.db

rtracklayer UCSC and other genome browser interactions, including I/O of common track formats (bed, wig, ...).

There are a number of additional packages, and the next Bioconductor release, scheduled for the end of October, contains additional sequence-related packages. Some highlights include

HilbertViz and HilbertVizGUI Novel visualization of large genomic regions.

Rolexa Solexa / Illumina base caller

genomeIntervals Alternative representation of ranged data.

chipseq Analysis of ChIP-seq experiments

GenomicFeatures 'Experimental' representation of transcript-based information for *M. musculus* and *H. sapiens*.

These packages interact with established R and Bioconductor packages, such as the gene-centric annotation packages (AnnotationDbi; e.g., org.Hs.eg.db, GO.db) and facilities for querying web-based resources (e.g., biomaRt).

2 In action

2.1 Experimental design

Here we will walk through a very superficial preliminary analysis of the CAMDA contest data. Set the R directory to the location of the distributed data

```
> setwd("c:/path/to/folder")
> stopifnot("extdata" %in% list.files())
```

and read in the sample information

```
> fl <- file.path("extdata", "SampleInfo_camda2009.txt")
> samples <- read.table(fl, header = TRUE, sep = "\t")
```

`samples` is a *data.frame* (like a spread sheet) that we can subset. Let's focus on the polII experiment, selecting those samples that do *not* have "IFNg" in their sample type label.

```
> isPolII <- !grepl("IFNg", samples[["SampleType"]])
> polIIdf <- samples[isPolII, ]
```

`polIIdf` is also a data frame; the first two columns are file names; here we display the column names and a subset of `polIIdf`, dropping columns 1 to 2:

```
> names(polIIdf)
```

```
[1] "SequenceFileName" "MappedFileName"
[3] "SampleType"       "Replicate"
[5] "Lane"             "FlowCell"
[7] "Date"
```

```
> polIIdf[, -(1:2)]
```

```
        SampleType Replicate  Lane FlowCell     Date
1        Input DNA      rep1 laneA   FC6144    82907
2        Input DNA      rep1 laneB  FC12033    91907
3        Input DNA      rep1 laneC  FC12033    91907
4        Input DNA      rep1 laneD  FC12033    91907
5        Input DNA      rep1 laneE  FC12033    91907
6        Input DNA      rep1 laneF  FC12044    91407
7        Input DNA      rep1 laneG  FC12060    90907
8        Input DNA      rep1 laneH  FC12170   102207
9        Input DNA      rep1 laneI  FC12170   102207
10       Input DNA      rep1 laneJ  FC12170   102207
11       Input DNA      rep1 laneK  FC12170   102207
12       Input DNA      rep1 laneL  FC12187   102707
13       Input DNA      rep1 laneM  FC12187   102707
14 Pol II ChIP-seq      rep1 laneA FC201WVA 20080307
15 Pol II ChIP-seq      rep1 laneB FC201WVA 20080307
16 Pol II ChIP-seq      rep2 laneA   FC5817       NA
17 Pol II ChIP-seq      rep2 laneB   FC5817       NA
18 Pol II ChIP-seq      rep2 laneC   FC5817       NA
19 Pol II ChIP-seq      rep2 laneD FC201WVA 20080307
20 Pol II ChIP-seq      rep2 laneE FC201WVA 20080307
21 Pol II ChIP-seq      rep3 laneA   FC4390    70107
22 Pol II ChIP-seq      rep3 laneB FC201WVA 20080307
23 Pol II ChIP-seq      rep3 laneC FC20B5RA 20080331
24 Pol II ChIP-seq      rep3 laneD FC20B5RA 20080331
```

Are there possible statistical issues with the experimental design?

2.2 Input

Now read in a lane (actually, a random sample of 500,000 reads) of data. We do this using the ShortRead package. The data we will look at has been aligned, and is in the Solexa 'Result' format.

```
> library(ShortRead)
> polII <- readAligned("extdata", "FC20B5RA_20080331_s_1_eland_result.shuf.txt",
+     type = "SolexaResult")
```

Aligned read objects can be *large*, and in real work flows we will often arrange to work with just one lane in memory at a time; it will sometimes be necessary to use a large memory, non-Windows computer. Discover the class of these objects, and explore the associated help page, with

```
> class(polII)
> `?`("AlignedRead-class")
```

2.3 Exploration

The data in `polII` or `input` is readily available for exploration. For instance, we can extract the reads with

```
> sread(polII)
```

```
  A DNAStringSet instance of length 500000
         width seq
     [1]    28 TTAGTAGCTAGTGATTTTCATCTGTTCA
     [2]    28 CTGTCTGGTCTCCTAGGCTGCTGTCGCG
     [3]    28 GNNNNNNNNNNNNNNNNNNNNNNNNNNN
     [4]    28 TTTCATGGCGCCCCGGCCACGGCGAGCC
     [5]    28 CTGCCGCACCACACCTCATCTAACAACA
     [6]    28 GTGGCCCGCCTCCCCACATCCGGTCGCC
     [7]    28 ACACAAAAAAACAAAAAAAAATAAACAC
     [8]    28 TCATCCCAACACCACCTCACACCACCCT
     [9]    28 CGGAGCGCGGCGACTCCAGGGGACCCCC
     ...   ... ...
[499992]    28 AGGTGCTGTGGTCTCCTCTTCGTGCCCC
[499993]    28 AACCTTCCCGAATCTTGACTGTAGTAAA
[499994]    28 CCGGGAGGCCGGGCGGGCACAATAGGCC
[499995]    28 CNNNNNNNNNNNNNNNNGGNGCTGAGG
[499996]    28 TTTCCTGGCTTCAGAGCCCGGGGTGGGA
[499997]    28 AAGGAAAATAAAGAAAGTTTTATATAAT
[499998]    28 CNNNNNNNNNNNNNNNNNNNNNNNNNN
[499999]    28 ACCAAACTGGAGGCAGAGCTATGACACA
[500000]    28 CTTTTCCCTGAAGCTCAAACTGTGATGG
```

and summarize the strand to which reads aligned using the `strand` accessor and R's `table` function.

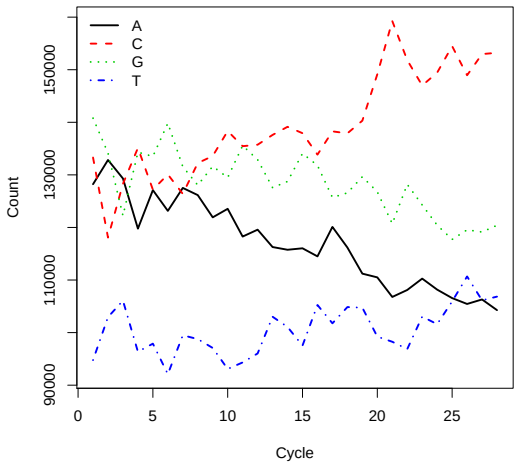

Figure 1: Nucleotide count per cycle.

```
> table(strand(polII), useNA = "always")

     -      +      *   <NA>
126206 126073      0 247721
```

This shows that about $1/2$ of the reads did not align to any strand, and that of the remainder about equal numbers aligned to the plus and minus strands. The following slightly more complicated example determines how often each letter in the IUPAC DNA alphabet is used in each cycle of the reads, and then creates a graph of this data. The results are shown in Figure **??**.

```
> abc <- alphabetByCycle(sread(polII))[1:4,
+     ]
> matplot(t(abc), type = "l", xlab = "Cycle",
+     ylab = "Count", lwd = 2)
> legend("topleft", rownames(abc), bty = "n",
+     lty = 1:4, lwd = 2, col = 1:4)
```

There are several technological artifacts shown in the figure. For instance, the nucleotide frequencies are not constant across cycle as might be expected if sonication where random. This particular artifacts are not present in more recent generations of the Illumina hardware and analysis software, but it is likely that significant technology biases still exist. Artefacts abound in archived data.

There are many additional ways of manipulating reads, e.g., filtering for low-complexity (`dustyFilter`), subsetting (e.g., `polII[chromosome(polII) == "chr21.fa"]` to select just those reads aligned to chromosome 21), narrowing and trimming reads, etc.

2.4 Quality assessment

Interactively exploring data is one way to get an understanding of the data. ShortRead also contains a quality assessment report generator. This requires all the data, and so it is not feasible to do in this lab. The commands are broken into two phases, a relatively slow and resource-intensive collection of QA data from each aligned read, and processing of the collated statistics into a report. Here are the commands:

```
> qa <- qa("exdata", ".*stat1.*", type = "SolexaExport")
> browseURL(report(qa, dest = file.path("inst",
+     "qa", "qa_stat1_report")))
```

The arguments to qa can be discovered on the help page ?qa. The actual report for the STAT1 data is in the folder inst/qa/qa_stat1_report.

2.5 A simplistic ChIP-seq work flow

WARNING: this work flow is not rigorous, it is meant to provide a feeling for how operations can be performed in R. We start by loading an object containing all reads from one lane, aligning to chromosome 21.

```
> load(file.path("data", "chr21.rda"))
```

This loads two objects into the environment, inputchr21 and polIIchr21, e.g.,

```
> polIIchr21
```

```
class: AlignedRead
length: 42523 reads; width: 28 cycles
chromosome: chr21.fa chr21.fa ... chr21.fa chr21.fa
position: 16575376 43686130 ... 14417225 26375394
strand: + + ... + +
alignQuality: NumericQuality
alignData varLabels: matchCode nExactMatch ... mismatchDetailOne mismatchDetailTwo
```

```
> length(inputchr21)
```

```
[1] 27236
```

Let's take a very quick route to peak-finding:

1. Down-sample polIIchr21 so that each lane has the same number of reads

   ```
   > idx <- sample(length(polIIchr21), length(inputchr21))
   > polIIchr21 <- polIIchr21[idx]
   ```

2. Extend each read by a constant number of nucleotides, and summarize coverage over the length of chromosome 21

```
> inputcvg <- coverage(inputchr21, start=1, end=46944323,
+                      extend=150L)[["chr21.fa"]]
> polIIcvg <- coverage(polIIchr21, start=1, end=46944323,
+                      extend=150L)[["chr21.fa"]]
```

3. Combine coverage from each source, and call a 'peak' any contiguous region with coverage greater than 8.

```
> cvg <- inputcvg + polIIcvg
> isles <- slice(cvg, 8)
```

Notice that these operations are fast, and the resulting object size (e.g., `object.size(cvg)` is small.

4. Use the peaks found in the pooled data to determine the number of each type of read found in the two lanes separately, storing the result as a *RangedData* object

```
> counts <- RangedData(isles, input = aggregate(inputcvg,
+     isles, sum), polII = aggregate(polIIcvg,
+     isles, sum), space = "chr21")
```

To visualize our results, we use the lattice graphics package to plot (transformed) counts in the polII lane against those in the input lane (Figure **??**).

```
> print(xyplot(asinh(1 + polII) ~ asinh(1 +
+     input), as.data.frame(counts)))
```

```
pdf
  2
```

As indicated, this is a very rough work flow, illustrating the flexiblity of Bioconductor for working with sequence data rather than best practices for ChIP-seq analysis; there are two packages for more rigorous analysis in the forthcoming Bioconductor release.

2.6 Annotation and export

As a final activity, lets focus on a particular region of interest, say a 300k startign a nucleotide 33.5 million of chromosome 21. We specify the region of interest as a *RangesList* object; this object could contain multiple ranges on a chromosome, and multiple chromosomes.

```
> roi <- RangesList(chr21 = IRanges(33500000,
+     33800000))
```

We then find where our peaks overlap with our region(s) of interest.

```
> olap <- overlap(ranges(counts), roi)
```

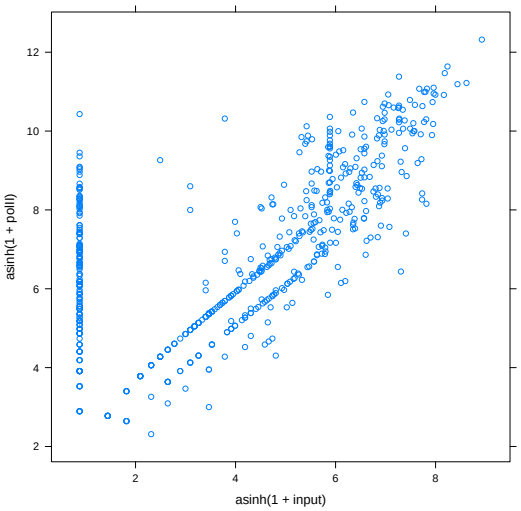

Figure 2: Per-island nucleotide count in a 'polII' and 'input' lane.

Once we have the overlap, we extract the `matchMatrix` desribing the matches for `chr21` and then obtain the indicies of the matches in the 'subject' (i.e., the indicies in the `counts` object).

```
> idx <- matchMatrix(olap[["chr21"]])[, "subject"]
```

We can use the rtracklayer package to export the data to an approriate format for reading in to a genome browser. The rtracklayer package expects a column named `score` to contain the information to be exported, so we create that column and populate it with the polII count data. Here we print the output to the console; normally the second argument would be a file name. There is a bug in the released version of rtracklayer that requires the name of the space to be revised before export.

```
> library(rtracklayer)
> counts[["score"]] <- counts[["polII"]]
> names(counts) <- "21"
> export(counts[idx, ], stdout(), "wig")

track name="R Track" type=wiggle_0
chr21        33524104        33524326        2908
chr21        33560523        33560836        4059
chr21        33561144        33561204        423
chr21        33618496        33619638        21399
chr21        33674781        33674968        1792
chr21        33697989        33698282        3514
chr21        33773852        33773867        105
```

```
chr21       33773873        33774207        7968
chr21       33774285        33774660        7093
```

Track data can also be read in to R, using rtracklayer's `import` function. A useful facility is to import tracks directly from the UCSC browser, e.g., all SNPs in our region of interest

```
> session <- browserSession()
> snp130 <- track(session, "snp130", roi)
```

3 Directions

This has been a lightning-fast tour of sequencing capabilitites, touching on a few of the available packages. For additional information:

- Visit the 'Software' link on the web site, and view vignettes of the packages described here.
- Visit the 'Workshops' tab on the Bioconductor home page for other recent presentations (e.g., during the BioC2009 conference).
- Participate in the `bioc-sig-seq` mailing list by following the 'Mailing lists' link on the Bioconductor home page.