Métodos Estadísticos y Analíticos de Datos Genómicos: ShortRead, Biostrings y Genominator

Alejandro Reyes
areyes@lcg.unam.mx
Licenciatura en Ciencias Genómicas

Universidad Nacional Autónoma de México

January 20, 2010
ShortRead and Biostrings

Introduction to biostrings

Exploring data with Shortread package

Aligned shortreads

Genominator
Libraries

- Packages we are going to use in this section
  
  ```
  > source("http://bioconductor.org/biocLite.R")
  > biocLite(c("ShortRead", "Genominator"))
  > library(ShortRead)
  ```
What is Biostrings?

- It provides containers for representing large biological sequences
- Provides utilities for basic computations on sequences (alphabet frequency, translate, reverseComplement)
- Tools for matching and pairwise alignments
Alignment tools

- matchPDict is fast, find all occurrences with a given number of mismatches, supports masked regions but does not support indels.
- vmatchPattern is similar to matchPDict, but it supports indels and uses edit distance penalty scheme.
- pairwiseAlignment is not useful for large sequences, returns only the best score, cannot handle masked genomes but includes a quality-based scoring.
Little example

▶ We want to find the "TATAAT" -10 boxes in a region of the Ecoli genome

> library(BSgenome.Ecoli.NCBI.20080805)
> Ecoli

E. coli genome

| organism: Escherichia coli (E. coli)
| provider: NCBI
| provider version: 2008/08/05
| release date: NA
| release name: NA

| sequences (see '?seqnames'):
| NC_008253  NC_008563  NC_010468
Little example

| NC_004431 | NC_009801 | NC_009800 |
| NC_002655 | NC_002695 | NC_010498 |
| NC_007946 | NC_010473 | NC_000913 |
| AC_000091 |

(use the '$' or '[[ operator to
access a given sequence)

> matchPattern("GAAC", Ecoli[["NC_008253"]])

Views on a 4938920-letter DNAString subject
subject: AGCTTTTTTATTCTG...AGTAAGTGTATTTTC
views:

<table>
<thead>
<tr>
<th>start</th>
<th>end</th>
<th>width</th>
<th>[GAAC]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>75</td>
<td>78</td>
<td>4</td>
</tr>
<tr>
<td>[2]</td>
<td>378</td>
<td>381</td>
<td>4</td>
</tr>
</tbody>
</table>
**Little example**

<table>
<thead>
<tr>
<th></th>
<th>Start</th>
<th>End</th>
<th>Score</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>[3]</td>
<td>537</td>
<td>540</td>
<td>4</td>
<td>[GAAC]</td>
</tr>
<tr>
<td>[4]</td>
<td>552</td>
<td>555</td>
<td>4</td>
<td>[GAAC]</td>
</tr>
<tr>
<td>[5]</td>
<td>1446</td>
<td>1449</td>
<td>4</td>
<td>[GAAC]</td>
</tr>
<tr>
<td>[6]</td>
<td>1476</td>
<td>1479</td>
<td>4</td>
<td>[GAAC]</td>
</tr>
<tr>
<td>[7]</td>
<td>1515</td>
<td>1518</td>
<td>4</td>
<td>[GAAC]</td>
</tr>
<tr>
<td>[8]</td>
<td>1641</td>
<td>1644</td>
<td>4</td>
<td>[GAAC]</td>
</tr>
<tr>
<td>[9]</td>
<td>1905</td>
<td>1908</td>
<td>4</td>
<td>[GAAC]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>[18985]</td>
<td>4936662</td>
<td>4936665</td>
<td>4</td>
<td>[GAAC]</td>
</tr>
<tr>
<td>[18986]</td>
<td>4937077</td>
<td>4937080</td>
<td>4</td>
<td>[GAAC]</td>
</tr>
<tr>
<td>[18987]</td>
<td>4937126</td>
<td>4937129</td>
<td>4</td>
<td>[GAAC]</td>
</tr>
<tr>
<td>[18988]</td>
<td>4937158</td>
<td>4937161</td>
<td>4</td>
<td>[GAAC]</td>
</tr>
<tr>
<td>[18989]</td>
<td>4937260</td>
<td>4937263</td>
<td>4</td>
<td>[GAAC]</td>
</tr>
<tr>
<td>[18990]</td>
<td>4937338</td>
<td>4937341</td>
<td>4</td>
<td>[GAAC]</td>
</tr>
<tr>
<td>[18991]</td>
<td>4938297</td>
<td>4938300</td>
<td>4</td>
<td>[GAAC]</td>
</tr>
</tbody>
</table>
Little example

[18992] 4938486 4938489 4 [GAAC]
[18993] 4938744 4938747 4 [GAAC]
What is ShortRead?

- It was developed by Martin Morgan
- "The ShortRead package aims to provide key functionality for input, quality assurance, and basic manipulation of short read DNA sequences such as those produced by Solexa, 454, Helicos, SOLiD, and related technologies"
Length of the reads

- Why is it important to consider alphabet frequency per cycle in solexa reads?

- According to solexa pipeline, in sequencing a genome we should find this frequencies similar to the GC content of the organism

```r
> reads <- readFastq("..", pattern = "typhi")
> abc <- alphabetByCycle(sread(reads),
+    alphabet = c("A", "T", "G",
+                "C", "N"))
> abc <- abc/colSums(abc)
> dataabc <- as.data.frame(abc)
```
Alphabet frequency per cycle
Finding overrepresented sequences

- With very simple and fast code we can find overrepresented sequences!

```r
> seq <- tables(reads, n = 15)
> topReads <- data.frame(read = names(seq[["top"]]),
                         +   count = unname(seq[["top"]]))
> topReads
     read
1 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
2 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGAAAAAAAAAAAAAAAAAAAAA
3 AGATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGAAAAAAAAAAAAAAATAA
4 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACA
5 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAA
6 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAA
7 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAA
8 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAA
9 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAA
10 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAA
11 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGAAAAAAAAAAAAAAATAA
12 AGATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGAAAAAAAAAAAAAAAGA
13 GATCGGAAGAGCTCGTATGCCGTCTTCTGCTTGAAAAAAAAAAAAAAATA
```
## Finding overrepresented sequences

<table>
<thead>
<tr>
<th>count</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2891</td>
</tr>
<tr>
<td>2</td>
<td>533</td>
</tr>
<tr>
<td>3</td>
<td>429</td>
</tr>
<tr>
<td>4</td>
<td>109</td>
</tr>
<tr>
<td>5</td>
<td>95</td>
</tr>
<tr>
<td>6</td>
<td>71</td>
</tr>
<tr>
<td>7</td>
<td>63</td>
</tr>
<tr>
<td>8</td>
<td>58</td>
</tr>
<tr>
<td>9</td>
<td>56</td>
</tr>
<tr>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>11</td>
<td>36</td>
</tr>
<tr>
<td>12</td>
<td>35</td>
</tr>
<tr>
<td>13</td>
<td>34</td>
</tr>
<tr>
<td>14</td>
<td>32</td>
</tr>
<tr>
<td>15</td>
<td>32</td>
</tr>
</tbody>
</table>
Working with the sequencing qualities

- ShortRead also allows you to work with the qualities given by Solexa reads.
- This is a very important thing to consider, and you can filter your reads to have just what you need.
Qualities

- We can also plot the qualities by cycle in order to cut the sequences when quality falls down.

```r
> qualitymatrix <- as(quality(reads),
+   "matrix")
> head(qualitymatrix[, 1:7])

[1,] 40 40 40 40 40 40 40
[2,] 40 40 40 40 40 40 40
[3,] 40 40 40 40 40 40 40
[4,] 40 40 40 40 40 40 40
[5,] 40 40 40 40 40 40 40
[6,] 40 40 40 40 40 40 40

> meanquality <- apply(qualitymatrix,
+   2, mean)
```

16 / 37
Quality per cycle
Cutting sequences

- The last plot indicates that we need to cut the sequences in order to have shorter but with a better quality sequence.
- The function `narrow` allows us to do this easily.

```r
> reads

class: ShortReadQ
length: 1045208 reads; width: 51 cycles

> shortreads <- narrow(reads, start = 1, +   end = 25)
> shortreads

class: ShortReadQ
length: 1045208 reads; width: 25 cycles
```
Ok, now we have just the first 25 cycles!

In solexa reads we have 2 sequences that are very common

1. AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

2. GATCGGAAGAGCTCGTATGCCGTCT

The function `srdistance` calculates the distance between two sequences, therefore it is useful to eliminate this sequences.

```
> distance1 <- srdistance(shortreads, 
+  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA")[[1]]
> distance2 <- srdistance(shortreads, 
+  "GATCGGAAGAGCTCGTATGCCGTCT")[[1]]
```
Distance

Distribution of distances to AAAAAAAAAAAAAAAAAAAAAAAAAAAA
Distance

Distribution of distances to GATCGGAAGAGCTCGTATGCCGTCT

Percent of Total

Percent of Total

Distance2
Clean sequences

- We have two vectors containing the distance to a respective sequence, how can we remove this sequences from our reads??

```r
> length(shortreads)
[1] 1045208

> cleanreads <- reads[distance1 >
+ 5 & distance2 > 5]
> length(cleanreads)
[1] 1036040
```

- Then, we can write our clean sequences into a fastq file!

```r
> writeFastq(cleanreads, "cleanthypi.fastq")
```
Create our own filters

- We can also create our own filters with `srFilters`

```r
> filter <- srFilter(function(x) {
+   apply(as(quality(x), "matrix"),
+         1, sum) > 1000
+ }, name = "GoodQualityBases")
> reads[filter(reads)]

class: ShortReadQ
length: 1030888 reads; width: 51 cycles
```
Aligned shortreads

- ShortRead also contains function to work with aligned reads
- It is focused on aligned reads produced by Solexa Genome Analyzer ELAND Software, but there are also ways to read alignments from MAQ or Bowtie. The name of the funcion is `readAligned`.

```r
> exptPath <- system.file("extdata", 
+        package = "ShortRead")
> sp <- SolexaPath(exptPath)
```

- Note that there al NA values in strand and position. This means that those sequences could not be aligned by the software.
Functions

It is focused on aligned reads produced by Solexa Genome Analyzer ELAND Software, but there are also ways to read alignments from MAQ or Bowtie. The name of the function is `readAligned`.

```r
> aln <- readAligned(sp, "s_2_export.txt")
> aln

class: AlignedRead
length: 1000 reads; width: 35 cycles
chromosome: NM NM ... chr5.fa 29:255:255
position: NA NA ... 71805980 NA
strand: NA NA ... + NA
alignQuality: NumericQuality
alignData varLabels: run lane ... filtering contig
```
Functions

There are some functions to analyze the data such as `position`, `strand`.

```r
> head(position(aln), 3)
[1] NA NA NA
```

```r
> table(strand(aln))
   -  +  *
203 203  0
```
Qualities, again!

- We can also play with the qualities of both the sequences and of the alignment!
- The qualities are string-coded by Solexa establishment, the letter A corresponds to the log10 of 1.
- The qualities of the alignment are a little bit different, being 0 a failure in the alignment.

```r
> head(quality(alignQuality(aln)))
[1] 0 0 0 0 0 0
```
Filter data

And with this qualities we can filter our data!!!!

Lets suppose we want just the sequences that are aligned and filtered by Solexa.

```r
> filtered <- alignData(aln)["filtering"] == "Y"
> mapped <- !is.na(position(aln))
> filteredmapped <- aln[filtered & mapped]
> filteredmapped

class: AlignedRead
length: 364 reads; width: 35 cycles
chromosome: chr17.fa chr18.fa ... chr8.fa chr5.fa
position: 69345321 54982866 ... 19708804 71805980
strand: - + ... - +
alignQuality: NumericQuality
alignData varLabels: run lane ... filtering contig
And in case you are a little more biologist than bioinformatician...

▶ If you want a default analysis or you do not know how to do graphs (hope is not your case)... ShortRead can do this for you!

```r
> qual <- qa(sp)
> rpt <- report(qual, dest = ".")
```
The Genominator package provides an interface to storing and retrieving genomic data, together with some additional functionality aimed at high-throughput sequence data.

There are 3 broad classes of functions within Genominator: functions that import and transform data, functions that retrieve and summarize data and finally functions that operate on retrieved data (focused on analysis of next generation sequencing data).

This package is very new and is still under development.
Import

- It uses SQLite!
  ```r
  > library(Genominator)
  > aln <- readAligned("..", pattern = "andale.txt",
  +     type = "Bowtie")
  > aln2 <- readAligned("..", pattern = "andale.txt",
  +     type = "Bowtie")
  > chrMap <- levels(chromosome(aln))
  > lista <- NULL
  > lista <- list(Ecoli = aln, otro = aln2)
  > eData <- importFromAlignedReads(lista,
  +     chrMap = chrMap, dbFilename = "my.db",
  +     tablename = "raw", overwrite = TRUE,
  +     deleteIntermediates = FALSE)
  ```
Import

Writing table: 0.42 sec
Creating index: 0.688 sec
Creating table: __tmp_7567: 0.005 sec
inserting: 0.109 sec
dropping original table: 0.019 sec
renaming table: 0.005 sec
creating index: 0.043 sec
Writing table: 0.408 sec
Creating index: 0.685 sec
Creating table: __tmp_8926: 0.004 sec
inserting: 0.109 sec
dropping original table: 0.019 sec
renaming table: 0.005 sec
creating index: 0.043 sec

> head(eData)
Import

<table>
<thead>
<tr>
<th>chr</th>
<th>location</th>
<th>strand</th>
<th>Ecoli</th>
<th>otro</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>148</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1175</td>
<td>-1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2580</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2650</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>3646</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>8103</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>
Import

```r
> getRegion(eData, chr = 1, strand = 0,
+     start = 10000, end = 12000)

     chr location strand Ecoli otro
    1         1    12000     -1     1     1

> laneCounts <- summarizeExpData(eData)
> laneCounts

  Ecoli otro
 92885 92885
```
References

- http://www.lcg.unam.mx/compu2/cei/
Session info

> sessionInfo()

R version 2.11.0 Under development (unstable) (2009-10-31 r50269)
i686-pc-linux-gnu

locale:
[1] LC_CTYPE=en_US.UTF-8
[2] LC_NUMERIC=C
[3] LC_TIME=en_US.UTF-8
[4] LC_COLLATE=en_US.UTF-8
[5] LC_MONETARY=C
[6] LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=en_US.UTF-8
[8] LC_NAME=C
[9] LC_ADDRESS=C
[10] LC_TELEPHONE=C
[12] LC_IDENTIFICATION=C

attached base packages:
[1] stats graphics grDevices
Session info

[4] utils    datasets  methods
[7] base

other attached packages:
[1] Genominator_1.1.3
[2] RSQLite_0.7-3
[3] DBI_0.2-5
[4] BSgenome.Ecoli.NCBI.20080805_1.3.16
[5] ShortRead_1.5.10
[6] lattice_0.17-26
[7] BSgenome_1.15.2
[8] Biostrings_2.15.2
[9] IRanges_1.5.12

loaded via a namespace (and not attached):
[1] Biobase_2.7.0  grid_2.11.0
[3] hwriter_1.1