Searching a DNA sequence using the \texttt{matchPattern} method (work in progress)

Hervé Pagès

December 1, 2007

Contents

1 Load a genome 1
2 Find patterns in a DNA sequence 2
3 A note on performance 4

1 Load a genome

Load the \textit{Caenorhabditis elegans} genome:

\begin{verbatim}
> library(BSgenome.Celegans.UCSC.ce2)
> ls("package:BSgenome.Celegans.UCSC.ce2")

[1] "Celegans"

> Celegans

C. elegans genome:
- organism: Caenorhabditis elegans
- provider: UCSC
- provider version: ce2
- release date: Mar. 2004
- release name: WormBase v. WS120
- single sequences (DNAString objects, see '?seqnames'):
  chrI  chrII  chrIII  chrIV  chrV  chrX  chrM
- multiple sequences (BStringViews objects, see '?mseqnames'):
  upstream1000  upstream2000  upstream5000
  (use the '$' or '
  [' operator to access a given sequence)

> comment(Celegans$chrI)

[1] "Caenorhabditis elegans - chromosome I (generated from FASTA file chrI.fa)"
Display chromosome I:

> Celegans$chrI

15080483-letter "DNAString" instance
Value: GCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTA...AGGCTTAGGCTTAGGCTTAGGTTTAGGCTTAGGC

The number of letters in this sequence can be retrieved with:

> cI <- Celegans$chrI
> length(cI)

[1] 15080483

Some basic stats:

> af <- alphabetFrequency(cI)
> af

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>M</th>
<th>R</th>
<th>W</th>
<th>S</th>
<th>Y</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4838561</td>
<td>2697177</td>
<td>2693544</td>
<td>4851201</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>V</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

> sum(af) == length(cI)

[1] TRUE

2 Find patterns in a DNA sequence

To find all exact matches of pattern "ACCCAGGGC":

> p <- "ACCCAGGGC"
> countPattern(p, cI)

[1] 0

> countPattern(p, cI, mismatch = 1)

[1] 235

The matches can be stored in a BStringViews object by using the matchPattern method:

> m <- matchPattern(p, cI, mismatch = 1)
> m[4:6]
Views on a 15080483-letter DNAString subject
Subject: GCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCT...GGCTTAGGCTTAGGCTTAGGTTAGGCTTAGGC
Views:

<table>
<thead>
<tr>
<th>start</th>
<th>end</th>
<th>width</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>187350</td>
<td>9</td>
<td>[ACCCAAGGC]</td>
</tr>
<tr>
<td>2</td>
<td>213236</td>
<td>9</td>
<td>[ACCCAGGGG]</td>
</tr>
<tr>
<td>3</td>
<td>424133</td>
<td>9</td>
<td>[ACCCAGGAC]</td>
</tr>
</tbody>
</table>

> mismatch(p, m[4:6])

[[1]]
[1] 6

[[2]]
[1] 9

[[3]]
[1] 8

The mismatch method (new in Biostrings 2) returns the positions of the mismatching letters.

Note: The mismatch method is in fact a particular case of a (vectorized) alignment function where only “replacements” are allowed. Current implementation is slow but this will change.

It may happen that a match is out of limits like here:

> p2 <- DNAString("AAGCCTAAGCCTAAGCCTAA")
> m2 <- matchPattern(p2, cI, mismatch = 2)
> m2[1:4]

Views on a 15080483-letter DNAString subject
Subject: GCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCT...GGCTTAGGCTTAGGCTTAGGTTAGGCTTAGGC
Views:

<table>
<thead>
<tr>
<th>start</th>
<th>end</th>
<th>width</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>18</td>
<td>20</td>
<td>[GCCTAAGCCTAAGCCTAA]</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>20</td>
<td>[AAGCCTAAGCCTAAGCCTAA]</td>
</tr>
<tr>
<td>11</td>
<td>30</td>
<td>20</td>
<td>[AAGCCTAAGCCTAAGCCTAA]</td>
</tr>
<tr>
<td>17</td>
<td>36</td>
<td>20</td>
<td>[AAGCCTAAGCCTAAGCCTAA]</td>
</tr>
</tbody>
</table>

> p2 == m2[1:4]

[1] FALSE TRUE TRUE TRUE

> mismatch(p2, m2[1:4])

[[1]]
[1] 1 2
The list of exact matches and the list of inexact matches can both be obtained with:

```r
> m2[p2 == m2]
> m2[p2 != m2]
```

Note that the length of `m2[p2 == m2]` should be equal to `countPattern(p2, cI, mismatch=0)`.

### 3 A note on performance

If needed, the `matchPattern` and `countPattern` methods convert their first argument (the pattern) to an object of the same class than their second argument (the subject) before they pass it to the function that actually implements the fast search algorithm.

So if you need to reuse the same pattern a high number of times, it’s a good idea to convert it before to pass it to the `matchPattern` or `countPattern` method. This way the conversion is done only once:

```r
> library(hgu95av2probe)
> tmpseq <- BStringViews(hgu95av2probe$sequence, "DNAString")
> someStats <- function(v) {
+   GC <- DNAString("GC")
+   CG <- DNAString("CG")
+   sapply(1:length(v), function(i) {
+     y <- v[i]
+     c(alphabetFrequency(y)[1:4], GC = countPattern(GC, y),
+        CG = countPattern(CG, y))
+   })
+ }
> someStats(tmpseq[1:10])
```

```
A  1  5  6  4  4  2  4  5  9  2
C 10  5  4  7  5  7 10  8  7 10
G  6  5  3  8  8  6  4  5  4  4
T  8 10 12  6  8 10  7  7  5  9
GC  2  1  1  4  3  2  2  2  1  1
CG  0  0  0  2  1  1  0  0  0  0
```