Searching a DNA sequence using the `matchPattern` method
(work in progress)

Hervé Pagès
November 7, 2006

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 *Caenorhabditis elegans* genome:

```r
> library(BSgenome.Celegans.UCSC.ce2)
> ls(2)
[1] "Celegans"
> Celegans
Caenorhabditis elegans genome:
[1] "chrI" "chrII" "chrIII" "chrIV" "chrV"
[6] "chrM" "chrX" "upstream1000" "upstream2000" "upstream5000"
(use the '$' or '[[ operator to access a given chromosome)

> comment(Celegans$chrI)
[1] "Caenorhabditis elegans - chromosome I (generated from FASTA file chrI.fa)"

Display chromosome I:

```r
> Celegans$chrI
Views on a 15080483-letter DNAString subject
Subject: GCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCT...GGCTTAGGCTTAGGCTTAGGTTAGGCTTAGGC
Views:
  first  last  width
[1]  1 15080483 15080483 [GCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCT...TAGGCTTAGGCTTAGGCTTAGGC]
```
Note that Celegans$chrI is a BStringViews object with a single view (length(Celegans$chrI) is 1). The number of letters in this sequence can be retrieved with:

```r
> cI <- Celegans$chrI[[1]]
> length(cI)
[1] 15080483
```

Try with an upstream object:

```r
> length(Celegans$upstream2000)
[1] 992
> length(Celegans$upstream2000[[1]])
[1] 2000
```

Some basic stats:

```r
> af <- alphabetFrequency(cI)
> af

A  C  G  T  -  other
4838561  2697177  2693544  4851201  0   0

> sum(af) == length(cI)
[1] TRUE
```

2 Find patterns in a DNA sequence

To find all exact matches of pattern "ACCCAGGGC":

```r
> 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:

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

Subject: GCCTAAGGCTAAGCCTAAGCCTAAGC...GGCTTAGGCTTAGGCTTAGGTTAGGCTTAGGC

Views:

<table>
<thead>
<tr>
<th>first</th>
<th>last</th>
<th>width</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>187350</td>
<td>187358</td>
<td>9</td>
<td>[ACCCAAGGC]</td>
</tr>
<tr>
<td>213236</td>
<td>213244</td>
<td>9</td>
<td>[ACCCAGGGG]</td>
</tr>
<tr>
<td>424133</td>
<td>424141</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 (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: GCCTAAGGCTAAGCCTAAGCCTAAGC...GGCTTAGGCTTAGGCTTAGGTTAGGCTTAGGC

Views:

<table>
<thead>
<tr>
<th>first</th>
<th>last</th>
<th>width</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>18</td>
<td>20</td>
<td>GCCTAAGGCTAAGCCTAA</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  2  6  5  5  5  4  6  8  5  6
C  9  9  9  8  3  9  7  5  8  7
G  5  7  7  6 10  8  4  6  5  8
T  9  3  4  6  7  4  8  6  7  4
GC  3  4  4  2  1  3  0  0  2  2
CG  0  0  0  0  0  0  0  0  0  0
```