Vectorizing the `DNAString` function (work in progress)

Hervé Pagès

August 9, 2008

Contents

1 Introduction

This is a short tour on the `DNAString` function vectorization feature.

Feel free to add your own comments.

2 `DNAString` vs `XStringViews`

The `Biostrings2Classes` vignette presents a proposal for 2 new classes (*XString* and *XStringViews*) as a replacement for the *BioString* class currently defined in the *Biostrings* 1 (*Biostrings* v 1.4.x) package.

It also shows how to use the `DNAString` function to create a *DNAString* object (a *DNAString* object is just a particular case of an *XString* object):

```
> d <- DNAString("TTGAAAA-CTC-N")
> is(d, "XString")
```

```
[1] TRUE
```

However this function is NOT vectorized: it always returns a *DNAString* object (which can only represent a *single* string).

In *Biostrings* 1, the `DNAString` function IS vectorized. Its vectorized form does the following: (1) concats the elements of its `src` argument into a single big string, (2) stores the offsets of all these elements in the `offsets` slot.

This behaviour is not immediatly obvious to the user, until he looks at the `offsets` slot.

It always returns a *BioString* object (with has as many values as the number of elements passed in the `src` argument).

3 The `XStringViews` generic function

The feature described in the previous section (provided by the vectorized form of the `DNAString` function in *Biostrings* 1) is provided in *Biostrings* 2 via the `XStringViews` generic function:

```
> v <- XStringViews(c("TTGAAAA-C", "TC-N"), "DNAString")
> v

  Views on a 13-letter DNAString subject
subject: TTGAAAA-CTC-N
views:
    start end width
[1]     1   9     9 [TTGAAAA-C]
[2]    10  13     4 [TC-N]
```

4 Performance

The following example was provided by Wolfgang:

```
> library(hgu95av2probe)

> system.time(z <- XStringViews(hgu95av2probe$sequence, "DNAString"))

   user  system elapsed
  2.424   0.020   2.444

> z

  Views on a 5045000-letter DNAString subject
subject: TGGCTCCTGCTGAGGTCCCCTTTCCGGCTGTGAA...CAAGCCCTCGTGCTCCTTGTCAACAGCGCACCCA
views:
          start     end width
     [1]      1      25    25 [TGGCTCCTGCTGAGGTCCCCTTTCC]
     [2]     26      50    25 [GGCTGTGAATTCCTGTACATATTTC]
     [3]     51      75    25 [GCTTCAATTCCATTATGTTTTAATG]
     [4]     76     100    25 [GCCGTTTGACAGAGCATGCTCTGCG]
     [5]    101     125    25 [TGACAGAGCATGCTCTGCGTTGTTG]
     [6]    126     150    25 [CTCTGCGTTGTTGGTTTCACCAGCT]
     [7]    151     175    25 [GGTTTCACCAGCTTCTGCCCTCACA]
```

```
     [8]      176     200    25 [TTCTGCCCTCACATGCACAGGGATT]
     [9]      201     225    25 [CCTCACATGCACAGGGATTTAACAA]
     ...      ...     ...   ... ...
[201792] 5044776 5044800    25 [GAGTGCCAATTCGATGATGAGTCAG]
[201793] 5044801 5044825    25 [ACACTGACACTTGTGCTCCTTGTCA]
[201794] 5044826 5044850    25 [CAATTCGATGATGAGTCAGCAACTG]
[201795] 5044851 5044875    25 [GACTTTCTGAGGAGATGGATAGCCT]
[201796] 5044876 5044900    25 [AGATGGATAGCCTTCTGTCAAAGCA]
[201797] 5044901 5044925    25 [ATAGCCTTCTGTCAAAGCATCATCT]
[201798] 5044926 5044950    25 [TTCTGTCAAAGCATCATCTCAACAA]
[201799] 5044951 5044975    25 [CAAAGCATCATCTCAACAAGCCCTC]
[201800] 5044976 5045000    25 [GTGCTCCTTGTCAACAGCGCACCCA]
```

With *Biostrings* 1, the call to `DNAString(hgu95av2probe$sequence)` takes about 20 minutes... (the implementation of the vectorization feature is quadratic in time, as reported by Wolfgang).

5 Loading a FASTA file into an *XStringViews* object

The `read.XStringViews` function can be used to load a FASTA file in an *XStringViews* object:

```
> file <- system.file("extdata", "someORF.fa", package = "Biostrings")
> orf <- read.XStringViews(file, "fasta", "DNAString")
> orf

  Views on a 26339-letter DNAString subject
subject: ACTTGTAAATATATCTTTTATTTTCCGAGAGGAA...ATATACATAGGGCTAAGGAAGAAAAAAAAAATCAC
views:
    start   end width
[1]     1  5573  5573 [ACTTGTAAATATATCTTTTATTTTCCG...ACGCTTATCGACCTTATTGTTGATAT]
[2]  5574 11398  5825 [TTCCAAGGCCGATGAATTCGACTCTTT...CAGAGTAAATTTTTTTCTATTCTCTT]
[3] 11399 14385  2987 [CTTCATGTCAGCCTGCACTTCTGGGTC...CGATGGTACTCATGTAGCTGCCTCAT]
[4] 14386 18314  3929 [CACTCATATCGGGGGTCTTACTTCCCA...ACGTGTCCCGAAACACGAAAAAGTAC]
[5] 18315 20962  2648 [AGAGAAAGAGTTTCACTTCTTGATTAT...AAAATATAATTTATGTGTGAACATAG]
[6] 20963 23559  2597 [GTGTCCGGGCCTCGCAGGCGTTCTACT...TTCAAGTTTTGGCAGAATGTACTTTT]
[7] 23560 26339  2780 [CAAGATAATGTCAAAGTTAGTGGTCGT...AGGGCTAAGGAAGAAAAAAAAAATCAC]

> names(orf)

[1] "YAL001C TFC3 SGDID:S0000001, Chr I from 152168-146596, reverse complement, Verified ORF"
[2] "YAL002W VPS8 SGDID:S0000002, Chr I from 142709-148533, Verified ORF"
[3] "YAL003W EFB1 SGDID:S0000003, Chr I from 141176-144162, Verified ORF"
[4] "YAL005C SSA1 SGDID:S0000004, Chr I from 142433-138505, reverse complement, Verified ORF"
[5] "YAL007C ERP2 SGDID:S0000005, Chr I from 139347-136700, reverse complement, Verified ORF"
[6] "YAL008W FUN14 SGDID:S0000006, Chr I from 135916-138512, Verified ORF"
[7] "YAL009W SPO7 SGDID:S0000007, Chr I from 134856-137635, Verified ORF"
```

6 Switching between DNA and RNA views

The `XStringViews` function can also be used to switch between “DNA” and “RNA” views on the same string:

```
> orf2 <- XStringViews(orf, "RNAString")
```

These conversions are very fast because no string data needs to be copied:

```
> orf[[0]]@xdata
```

```
26339-byte XRaw object (data starting at memory address 0x45630d8)
```

```
> orf2[[0]]@xdata
```

```
26339-byte XRaw object (data starting at memory address 0x45630d8)
```