Alignments

VJC

August 9, 2008

Contents

1 Introduction

In this document we illustrate import of FASTA files, simple alignments, and consensus matrix calculation.

2 Data import

The *Biostrings* package includes FASTA files encoding the DNA sequence for Yeast chromosome I, along with some ORFs. Here we load the `someORF.fa` file into a *DNAStringSet* object by using the `read.DNAStringSet` function:

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

  A DNAStringSet instance of length 7
    width seq                                               names
[1]  5573 ACTTGTAAATATATCTTTTATTT...CTTATCGACCTTATTGTTGATAT YAL001C TFC3 SGDI...
[2]  5825 TTCCAAGGCCGATGAATTCGACT...AGTAAATTTTTTTCTATTCTCTT YAL002W VPS8 SGDI...
[3]  2987 CTTCATGTCAGCCTGCACTTCTG...TGGTACTCATGTAGCTGCCTCAT YAL003W EFB1 SGDI...
```

```
[4]   3929 CACTCATATCGGGGGTCTTACTT...TGTCCCGAAACACGAAAAAGTAC YAL005C SSA1 SGDI...
[5]   2648 AGAGAAAGAGTTTCACTTCTTGA...ATATAATTTATGTGTGAACATAG YAL007C ERP2 SGDI...
[6]   2597 GTGTCCGGGCCTCGCAGGCGTTC...AAGTTTTGGCAGAATGTACTTTT YAL008W FUN14 SGD...
[7]   2780 CAAGATAATGTCAAAGTTAGTGG...GCTAAGGAAGAAAAAAAAATCAC YAL009W SPO7 SGDI...
```

Note that some of the ORF sequences are represented in reverse complement form.

3 Optimal Pairwise Alignments

The function `pairwiseAlignment` solves the (Needleman-Wunsch) global, the (Smith-Waterman) local, and the overlap optimal pairwise alignment problems. The solution to each of these problems is dependent on the specified substitution scores and the gap penalties:

- Substitution Scores: The substitution scores can either be fixed for each pairing of letters within the two strings or be dependent on the qualities associated with those letters. When the scores are fixed by pairing, the `substituionMatrix` argument takes a matrix with the appropriate alphabets as dimension names. When the scores are quality-based, the `patternQuality` and `subjectQuality` arguments accept the equivalent of [0-99] numeric quality values for the respective strings.
- Gap Penalties: Gaps have the potential to incur a cost when they are introduced and when they are extended in an optimal pairwise alignment. The former is regulated by the `gapOpening` argument and the latter by the `gapExtension` argument.

The `pairwiseAlignment` function uses memory and computation time proportional to the product of the two string lengths.

The BLOSUM50 matrix is available in this package as a matrix:

```
> data(BLOSUM50)
> BLOSUM50[1:4, 1:4]
```

```
   A  R  N  D
A  5 -2 -1 -2
R -2  7 -1 -2
N -1 -1  7  2
D -2 -2  2  8
```

A comparison of the three optimal pairwise alignments problems from Haubold and Wiehe:

```
> s1 <- DNAString("ACTTCACCAGCTCCCTGGCGGTAAGTTGATCAAAGGAAACGCAAAGTTTTCAAG")
> s2 <- DNAString("GTTTCACTACTTCCTTTCGGGTAAGTAAATATATAAATATATAAAAATATAATTTTCATC")
> mat <- matrix(-3, nrow = 4, ncol = 4)
> diag(mat) <- 1
> rownames(mat) <- colnames(mat) <- DNA_ALPHABET[1:4]
> globalAlign <- pairwiseAlignment(s1, s2, substitutionMatrix = mat,
+     gapOpening = -5, gapExtension = -2)
> localAlign <- pairwiseAlignment(s1, s2, type = "local", substitutionMatrix = mat,
+     gapOpening = -5, gapExtension = -2)
> overlapAlign <- pairwiseAlignment(s1, s2, type = "overlap", substitutionMatrix = ma
+     gapOpening = -5, gapExtension = -2)
```

A canonical global optimal pairwise alignment example from Durbin, Eddy et al:

```
> nwdemo <- pairwiseAlignment(AAString("PAWHEAE"), AAString("HEAGAWGHEE"),
+     substitutionMatrix = BLOSUM50, gapOpening = 0, gapExtension = -8)
> nwdemo
```

```
Global Pairwise Alignment
1:  --P-AW-HEAE
2:  HEAGAWGHE-E
Score:  1
```

The score (final element of dynamic programming score matrix) can be accessed:

```
> score(nwdemo)
```

```
[1] 1
```

4 Computing alignment consensus matrices

The consmat function is provided for computing a consensus matrix for a set of equal-length strings assumed to be aligned. To illustrate, the following application assumes the ORF data to be aligned for the first 10 positions (patently false):

```
> orf10 <- DNAStringSet(orf, end = 10)
> consmat(orf10)
```

```
      pos
letter         1         2         3         4         5         6         7
     A 0.2857143 0.2857143 0.2857143 0.0000000 0.5714286 0.4285714 0.4285714
     C 0.4285714 0.1428571 0.2857143 0.2857143 0.2857143 0.1428571 0.0000000
     G 0.1428571 0.1428571 0.1428571 0.2857143 0.1428571 0.0000000 0.4285714
     T 0.1428571 0.4285714 0.2857143 0.4285714 0.0000000 0.4285714 0.1428571
```

```
      pos
letter         8         9        10
     A 0.4285714 0.2857143 0.1428571
     C 0.0000000 0.2857143 0.4285714
     G 0.4285714 0.1428571 0.2857143
     T 0.1428571 0.2857143 0.1428571
```

The information content as defined by Hertz and Stormo 1995 is computed as follows:

```
> infContent <- function(Lmers) {
+     zlog <- function(x) ifelse(x == 0, 0, log(x))
+     co <- consmat(Lmers, freq = TRUE)
+     lets <- rownames(co)
+     fr <- alphabetFrequency(Lmers)[lets]
+     sum(co * zlog(co/fr))
+ }
> infContent(orf10)
```

```
[1] NA
```