

Programação Dinâmica para alinhamento local

Ricardo Manhães Savii

11 de Junho de 2016

Exercício

Atividade: construa a matriz de Programação dinâmica alinhando o para de sequências abaixo:

seq1: GAATTCAGTTA

seq2: GGATCGA

Resultado esperado (submiter):

- A) Programa de alinhamento local (Programação dinâmica)
- B) A matriz de alinhamento local (Programação dinâmica)
- C) O alinhamento e o escore total das duas sequencias como a Fig. 1 do link abaixo.

<http://vlab.amrita.edu/?sub=3&brch=274&sim=1433&cnt=1>

G	A	A	T	T	C	A		G	A	A	T	T	-	C
 	 	 	 	 		 		 	 	 	 	 		
G	A	C	T	T	-	A		G	A	C	T	T	A	C
+	+	-	+	+	-	+		+	+	-	+	+	-	+
5	5	3	5	5	4	5		5	5	3	5	5	4	5

Figure 1: Alinhamento de duas sequências quaisquer.

- D) Alinhamento de duas sequencias quaisquer

Introdução

O algoritmo de Smith-Waterman realiza o alinhamento local e utiliza programação dinâmica para tal função. Para isso ele precisa de uma matriz de score, à qual nos foi dada pelo enunciado do exercício como:

- match = +5
- mismatch = -3
- gap = -4

Logo nossa matriz terá o seguinte fomato para sequências de DNA:

```
matriz_score <- matrix(data = c(5, -3, -3, -3, -4,
                                -3, 5, -3, -3, -4,
                                -3, -3, 5, -3, -4,
                                -3, -3, -3, 5, -4,
                                -4, -4, -4, -4, -4), nrow = 5, ncol = 5, byrow = TRUE )
aa <- c('A', 'G', 'T', 'C', '-')
rownames(matriz_score) <- aa
colnames(matriz_score) <- aa
matriz_score
```

```
##      A  G  T  C  -
## A   5 -3 -3 -3 -4
## G  -3  5 -3 -3 -4
## T  -3 -3  5 -3 -4
## C  -3 -3 -3  5 -4
## -  -4 -4 -4 -4 -4
```

Salvaremos agora as duas sequências de teste em duas variáveis no programa. Salvei previamente as sequências do enunciado em um arquivo `seq_teste.fasta` e farei a leitura do arquivo utilizando a função `read.fasta` do pacote `seqinr`.

```
sequencias <- seqinr::read.fasta("seq_teste.fasta", seqtype = "AA")
seq1 <- sequencias[[1]]
seq2 <- sequencias[[2]]
seq3 <- sequencias[[3]]
seq4 <- sequencias[[4]]
seq1
```

```
## [1] "G" "A" "A" "T" "T" "C" "A" "G" "T" "T" "A"
## attr(,"name")
## [1] ""
## attr(,"Annot")
## [1] "> seq1:"
## attr(,"class")
## [1] "SeqFastaAA"
```

```
seq2
```

```
## [1] "G" "G" "A" "T" "C" "G" "A"
## attr(,"name")
## [1] ""
## attr(,"Annot")
## [1] "> seq2:"
## attr(,"class")
## [1] "SeqFastaAA"
```

```
seq3
```

```
## [1] "G" "A" "C" "T" "T" "A" "C"
## attr(,"name")
## [1] ""
```

```
## attr("Annot")
## [1] "> seq_teste1:"
## attr("class")
## [1] "SeqFastaAA"
```

```
seq4
```

```
## [1] "C" "G" "T" "G" "A" "A" "T" "T" "C" "A" "T"
## attr("name")
## [1] ""
## attr("Annot")
## [1] "> seq_teste2:"
## attr("class")
## [1] "SeqFastaAA"
```

Com as variáveis de matriz de score e sequências vamos iniciar a implementar a função comparadora e realizar testes na próxima seção.

Desenvolvimento do código e testes

Pensando na função, seus parâmetros precisam ser as duas sequências à serem alinhadas e a matriz de escore. Como resultado devemos obter o escore final da comparação local entre as duas sequências, a matriz de comparação entre as duas sequências e a matriz de direções das decisões tomadas para adquirir o melhor alinhamento. Tendo a matriz conseguiremos gerar o alinhamento e o escore total das duas sequências conforme a figura do enunciado.

```
compara <- function(seq1, seq2, matriz_score) {
  seq1 <- c('-',seq1)
  seq2 <- c('-',seq2)

  matriz <- matrix(nrow = length(seq1), ncol = length(seq2))
  direcoes <- matrix(nrow = length(seq1), ncol = length(seq2))

  rownames(matriz) <- seq1
  colnames(matriz) <- seq2
  rownames(direcoes) <- seq1
  colnames(direcoes) <- seq2

  matriz[1,] <- matriz[,1] <- 0
  for(i in 2:length(seq1)) {
    for(j in 2:length(seq2)) {
      matriz[i,j] <- max(matriz[i-1,j-1] + matriz_score[seq1[i],seq2[j]],
                        matriz[i ,j-1] + matriz_score['-',seq2[j]],
                        matriz[i-1,j ] + matriz_score[seq1[i],'-'],
                        0)
      if(matriz[i-1,j-1] + matriz_score[seq1[i],seq2[j]] >=
         matriz[i ,j-1] + matriz_score['-',seq2[j]] &&
         matriz[i-1,j-1] + matriz_score[seq1[i],seq2[j]] >=
         matriz[i-1,j ] + matriz_score[seq1[i],'-']) {
        direcoes[i,j] <- '/'
      } else {
```

```

        if(matriz[i ,j-1] + matriz_score['-',seq2[j]] >=
            matriz[i-1,j-1] + matriz_score[seq1[i],seq2[j]] &&
            matriz[i ,j-1] + matriz_score['-',seq2[j]] >=
            matriz[i-1,j ] + matriz_score[seq1[i],'-']) {
            direcoes[i,j] <- '-'
        } else {
            if (matriz[i-1,j ] + matriz_score[seq1[i],'-'] >
                matriz[i-1,j-1] + matriz_score[seq1[i],seq2[j]] &&
                matriz[i-1,j ] + matriz_score[seq1[i],'-'] >
                matriz[i ,j-1] + matriz_score['-',seq2[j]]) {
                direcoes[i,j] <- '|'
            }
        }
    }
}
}
}
}

list(matriz,direcoes)
}

```

Primeiro vamos comparar as sequências que foram dadas a matriz resposta, assim verificaremos se nossa função está funcionando corretamente:

```

resultado_teste <- compara(seq3, seq4, matriz_score)
resultado_teste[[1]]

```

```

##  - C G T G  A A  T  T  C  A  T
## - 0 0 0 0 0  0 0  0  0  0  0  0
## G 0 0 5 1 5  1 0  0  0  0  0  0
## A 0 0 1 2 1 10 6  2  0  0  5  1
## C 0 5 1 0 0  6 7  3  0  5  1  2
## T 0 1 2 6 2  2 3 12 8  4  2  6
## T 0 0 0 7 3  0 0  8 17 13 9  7
## A 0 0 0 3 4  8 5  4 13 14 18 14
## C 0 5 1 0 0  4 5  2  9 18 14 15

```

Vemos que temos o mesmo resultado ao apresentado na Figura 2 fornecida pelo professor:

Logo elaboramos corretamente a função. Com isso podemos obter agora a matriz para as sequências **seq1** e **seq2** solicitadas no enunciado.

```

resultado <- compara(seq2, seq1, matriz_score)
resultado[[1]]

```

```

##  - G  A A  T T  C  A  G  T T  A
## - 0 0  0 0  0 0  0  0  0  0  0  0
## G 0 5  1 0  0 0  0  0  5  1  0  0
## G 0 5  2 0  0 0  0  0  5  2  0  0
## A 0 1 10 7  3 0  0  5  1  2  0  5
## T 0 0  6 7 12 8  4  1  2  6  7  3
## C 0 0  2 3  8 9 13  9  5  2  3  4
## G 0 5  1 0  4 5  9 10 14 10 6  2
## A 0 1 10 6  2 1  5 14 10 11 7 11

```

	-	C	G	T	G	A	A	T	T	C	A	T
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	5	1	5	1	0	0	0	0	0	0
A	0	0	1	2	1	10	6	2	0	0	5	1
C	0	5	1	0	0	6	7	3	0	5	1	2
T	0	1	2	6	2	2	3	12	8	4	2	6
T	0	0	0	7	3	0	0	8	17	13	9	7
A	0	0	0	3	4	8	5	4	13	14	18	14
C	0	5	1	0	0	4	5	2	9	18	14	15

Figure 2: Matriz preenchida e os back pointers.

Agora precisamos de uma segunda função que faça a busca retro-ativa na matriz para encontrar os maiores scores e nos mostre os melhores alinhamentos entre as duas sequências. Para representar os alinhamentos vou pensar na imagem como uma matriz e alimentar as duas primeiras linhas com os alinhamentos gerados e a terceira linha com os valores de escore. Logo podemos gerar uma nova função:

```
alinhar <- function(matriz) {
  res <- list()
  maximos <- which(matriz[[1]] == max(matriz[[1]]), arr.ind = TRUE)

  for (i in 1:dim(maximos)[1]) {
    linha = maximos[i,1]
    coluna = maximos[i,2]
    seq1 <- c()
    seq2 <- c()
    score <- c()

    while(TRUE) {
      if (matriz[[1]][linha, coluna] == 0) break
      if (matriz[[2]][linha, coluna] == '/') {
        seq1 = c( rownames(matriz[[1]])[linha], seq1 )
        seq2 = c( colnames(matriz[[1]])[coluna], seq2 )
        score = c( matriz[[1]][linha,coluna] - matriz[[1]][linha-1,coluna-1], score )

        linha = linha-1
        coluna = coluna-1

      } else if (matriz[[2]][linha, coluna] == '-') {
        seq1 = c( '-', seq1 )
        seq2 = c( colnames(matriz[[1]])[coluna], seq2 )
        score = c( matriz[[1]][linha,coluna] - matriz[[1]][linha,coluna-1], score )

        coluna = coluna-1

      } else if (matriz[[2]][linha, coluna] == '|') {
        seq1 = c( rownames(matriz[[1]])[linha], seq1 )
        seq2 = c( '-', seq2 )
        score = c( matriz[[1]][linha,coluna] - matriz[[1]][linha-1,coluna], score )
      }
    }
  }
}
```

```

        linha = linha-1
    }
}
res[[i]] <- rbind(seq1, seq2, score)
}
res
}

```

Com a função definida acima conseguimos agora verificar se conseguiremos com as matrizes `escore` e `direções` em `resultado_teste` o mesmo resultado visto na **Figura 1**.

```

teste1 <- alinhar(resultado_teste)
teste1

```

```

## [[1]]
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## seq1 "G"  "A"  "C"  "T"  "T"  "A"  "C"
## seq2 "G"  "A"  "A"  "T"  "T"  "-"  "C"
## score "5"  "5"  "-3" "5"  "5"  "-4" "5"
##
## [[2]]
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## seq1 "G"  "A"  "C"  "T"  "T"  "-"  "A"
## seq2 "G"  "A"  "A"  "T"  "T"  "C"  "A"
## score "5"  "5"  "-3" "5"  "5"  "-4" "5"

```

Confirmado, os alinhamentos obtidos são iguais aos da **Figura 1**. Com isso podemos aplicar nossa nova função as matrizes `escore` e `direções` de `resultado` para obtermos os alinhamentos das sequências `seq1` e `seq2`.

```

aux <- alinhar(resultado)
aux

```

```

## [[1]]
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## seq1 "G"  "G"  "A"  "-"  "T"  "C"  "G"  "A"
## seq2 "G"  "A"  "A"  "T"  "T"  "C"  "-"  "A"
## score "5"  "-3" "5"  "-4" "5"  "5"  "-4" "5"
##
## [[2]]
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## seq1 "G"  "G"  "A"  "-"  "T"  "C"  "-"  "G"
## seq2 "G"  "A"  "A"  "T"  "T"  "C"  "A"  "G"
## score "5"  "-3" "5"  "-4" "5"  "5"  "-4" "5"

```

Com estes alinhamentos concluímos o solicitado no enunciado até o item C). Na próxima seção irei alinhar duas sequências quaisquer.

Comparações de sequências e Análises

Para duas novas sequências criei um novo arquivo com nome `seq.fasta`, escolhi duas sequências da tarefa 5 (identificação de ORFs). Farei a leitura conforme feito anteriormente e aplicarei as funções criadas na seção anterior.

```
sequencias <- seqinr::read.fasta("seq.fasta", seqtype = "AA")
seq1 <- toupper(sequencias[[1]])
seq2 <- toupper(sequencias[[2]])
seq1
```

```
## [1] "A" "T" "G" "C" "C" "T" "T" "A" "C" "C" "A" "A" "A" "G" "G" "A" "T"
## [18] "G" "A" "C" "A" "T" "T" "A" "T" "T" "A" "A" "T" "G" "T" "A" "G"
## attr(,"name")
## [1] ""
## attr(,"Annot")
## [1] "> seq1:"
## attr(,"class")
## [1] "SeqFastaAA"
```

```
seq2
```

```
## [1] "A" "T" "G" "A" "A" "C" "A" "A" "G" "A" "T" "G" "T" "C" "T" "T" "A"
## [18] "G"
## attr(,"name")
## [1] ""
## attr(,"Annot")
## [1] "> seq2:"
## attr(,"class")
## [1] "SeqFastaAA"
```

Utilizaremos a mesma **matriz_score** definida anteriormente para realizar a comparação:

```
resultado <- compara(seq2, seq1, matriz_score)
max(resultado[[1]])
```

```
## [1] 44
```

```
which(resultado[[1]] == max(resultado[[1]]), arr.ind = TRUE)
```

```
## row col
## A 18 25
```

Vemos a matriz de resultado, o valor máximo é 44 e só ocorre uma vez na linha 18 e coluna 25. Agora iremos gerar esse melhor alinhamento das sequências:

```
aux <- alinhar(resultado)
```

```
print(aux)
```

```
## [[1]]
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## seq1  "T"  "G"  "A"  "A"  "C"  "-"  "A"  "A"  "-"  "G"  "A"  "T"  "G"
## seq2  "T"  "T"  "A"  "C"  "C"  "A"  "A"  "A"  "G"  "G"  "A"  "T"  "G"
## score "5"  "-3" "5"  "-3" "5"  "-4" "5"  "5"  "-4" "5"  "5"  "5"  "5"
##      [,14] [,15] [,16] [,17] [,18] [,19]
## seq1  "T"   "C"   "-"   "T"   "T"   "A"
## seq2  "A"   "C"   "A"   "T"   "T"   "A"
## score "-3"  "5"  "-4"  "5"  "5"  "5"
```

Obtivemos então o melhor alinhamento entre as duas sequências selecionadas.

Conclusões

Acredito ter cumprido os objetivos da atividade, implementei as funções com programação dinâmica para obter os alinhamentos com maior score e também encontrar os melhores alinhamentos à partir da matriz de escore.

Referências

- [1] CHARIF, Delphine; LOBRY, Jean R. “SeqinR 1.0-2: a contributed package to the R project for statistical computing devoted to biological sequences retrieval and analysis.” In: Structural approaches to sequence evolution. Springer Berlin Heidelberg, 2007. p. 207-232.
- [2] XIE, Yihui. “Dynamic Documents with R and knitr”. CRC Press, 2015.
- [3] RStudio Team (2015). “RStudio: Integrated Development for R. RStudio”, Inc., Boston, MA URL <http://www.rstudio.com/>.
- [4] Pearson, William R. “Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms.” Genomics 11.3 (1991): 635-650.
- [5] Mott, Richard. “Maximum-likelihood estimation of the statistical distribution of Smith-Waterman local sequence similarity scores.” Bulletin of Mathematical Biology 54.1 (1992): 59-75.