

## Multiple alignment

The natural generalization of a pairwise alignment

A multiple alignment of  $S_1, \dots, S_k \in \Sigma^*$  is a

$k \times l$ -matrix where

-  $\mathcal{M}_{ij} \in \Sigma \cup \{-\}$

$$\mathcal{M} = \begin{bmatrix} - & R_1 & - \\ & \vdots & \\ - & R_k & - \end{bmatrix}$$

- row  $i$ ,  $R_i$ , with "-"s removed, spells  $S_i$

A multiple alignment can be used to

- emphasize seq. commonalities
- highlight weak seq. similarities
- describe evolutionary relationships ....

## Computing a good multiple alignment

Given a SCORE-function, find  $\mathcal{M}^*$  where

$$\mathcal{M}^* = \underset{\mathcal{M}}{\operatorname{argmax}} \operatorname{SCORE}(\mathcal{M})$$

we need a "good" score-function ...

## Sum-of-Pairs score

The SP-score  $SP(\mathcal{M})$  is the sum of the scores of all induced pairwise alignments ...

$$\mathcal{M} = \left[ \begin{array}{c} \text{--- } R_1 \text{ ---} \\ \vdots \\ \text{--- } R_n \text{ ---} \end{array} \right] \quad SP(\mathcal{M}) = \sum_{1 \leq i < j \leq n} \text{PAIRSCORE}(R_i, R_j)$$

If PAIRSCORE is column-based (e.g. global alignment with linear gap cost) then ...

$$\begin{aligned} SP(\mathcal{M}) &= \sum_{1 \leq i < j \leq n} \text{PAIRSCORE}(R_i, R_j) \\ &= \sum_{1 \leq i < j \leq n} \sum_{1 \leq c \leq L} d(R_i[c], R_j[c]) \\ &= \sum_{1 \leq c \leq L} \sum_{1 \leq i < j \leq n} d(\mathcal{M}_{ic}, \mathcal{M}_{jc}) \\ &= \sum_{1 \leq c \leq L} SP \left( \begin{array}{c} \mathcal{M}_{ic} \\ \vdots \\ \mathcal{M}_{nc} \end{array} \right) \end{aligned}$$

↑  
column  $c$  in  $\mathcal{M}$

... SP-score is column-based

## Computing an optimal mult. align with a col.-based score func.

\* Let  $D(i_1, \dots, i_k)$  be the cost of an optimal mult. align of  $S_1[1..i_1], \dots, S_k[1..i_k]$

\* Compute  $D(i_1, \dots, i_k)$  by optimizing over all possible last columns of such an alignment.

$$\begin{bmatrix} S_1[i_1] \\ S_2[i_2] \\ \vdots \\ S_k[i_k] \end{bmatrix}, \begin{bmatrix} - \\ S_2[i_2] \\ \vdots \\ S_k[i_k] \end{bmatrix}, \dots, \begin{bmatrix} S_1[i_1] \\ S_2[i_2] \\ \vdots \\ S_k[i_k] \end{bmatrix}, \dots$$

\* possible last columns =  $2^k - 1$

$$D(i_1, \dots, i_k) = \min_{\text{last.col}} [ D(i_1', \dots, i_k') + \text{cost}(\text{last.col}) ]$$

• Time and space?

$$\text{Time: } O(t \cdot 2^k \cdot n^k)$$

↑  
time to compute cost of col.

$$\text{Space: } O(n^k)$$

• An optimal alignment can be found by backtracking ...

$$\begin{bmatrix}
 A & A & G & A & A & - & A \\
 A & T & - & A & A & T & G \\
 C & T & G & - & G & - & G \\
 A & T & G & A & A & - & G
 \end{bmatrix}$$

$$\textcircled{1} \begin{bmatrix}
 A & A & G & A & A & - & A \\
 A & T & - & A & A & T & G
 \end{bmatrix}$$

$$\textcircled{2} \begin{bmatrix}
 A & T & - & A & A & T & G \\
 C & T & G & - & G & - & G
 \end{bmatrix}$$

$$\textcircled{3} \begin{bmatrix}
 A & A & G & A & A & - & A \\
 C & T & G & - & G & - & G
 \end{bmatrix}$$