Short template switch events explain mutation clusters in the human genome

Ari Löytynoja[1] and Nick Goldman[2]

[1] *Institute of Biotechnology, University of Helsinki, Helsinki, Finland;*

[2] *European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Genome Campus, Hinxton, UK*

Supplemental Algorithm S1: Dynamic programming algorithm to find optimal template-switch solution under the four-point model

The algorithm defines the dynamic programming recursions to find the optimal alignment solution under the four-point mutation model. See also Supplemental Fig. S2.

Let sequences x and y consist of bases $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_m$. Function $\mathsf{c}(x_i)$ gives the complement of base x_i, function $\mathsf{s}(x_i, y_j)$ gives the score of matching bases x_i and y_j, and d is the indel penalty. In our implementation, the match score is 1 for two identical bases, otherwise -1, and the indel penalty is -2.

The algorithm for the template switch alignment differs from the standard linear-cost alignment of Sankoff[1] in two aspects. First, the algorithm consists of Recursions 1–3 that define three matrices, V_1, V_2 and V_3, and the alignment path must go via these matrices in order, starting with sequence fragment F1 in V_1 and ending with F3 in V_3. Second, in matrix V_2 the alignment proceeds backwards in respect to sequence y (generation of fragment F2 is in reverse direction; see Supplemental Fig. S2). Similarly to standard alignment, pointer matrices recording the move chosen at each cell (i, j), including the jumps between the matrices, need to be stored. Once the recursions have been completed, $V_3(n, m)$ holds the optimal score; the actual alignment solution is traced back using the pointer matrices.

Two variants of the algorithm allow for computing control statistics: with minor changes in Recursion 2, one can replace the search of a reverse-complement match in V_2 with a search for a reverse match (denoted variant 2r) or a complement match (2c):

2r: **reverse:** within the '$\max$' term for $V_2(i, j)$, $\mathsf{s}(x_i, \mathsf{c}(y_j))$ is replaced by $\mathsf{s}(x_i, y_j)$

2c: **complement:** within the second loop, $j = m, ..., 1$ is replaced by $j = 1, ..., m$, and $V_2(i-1, j+1)$ is replaced by $V_2(i-1, j-1)$

In our analyses, we used variant 2r to compute control statistics.

We note that some event types can be reverted. While the direction of mutation could be determined using an outgroup, our approach considers just two sequences and finds only solutions consistent with the defined reference sequence.

[1] D Sankoff. Matching sequences under deletion-insertion constraints. *Proc. Natl. Acad. Sci. U. S. A.*, 69:4–6, 1972

Supplemental Algorithm S1 Optimal template-switch solution under the four-point model

```
Initialisation:
    $V_1(0,0)$ is set to 0                              ▷ Alignment path must start from $V_1(0,0)$
    $V_2(\bullet, m+1)$ and $V_2(0,\bullet)$ are set to $-\infty$     ▷ Optimal path in $V_2$ must come from $V_1$
    $V_3(\bullet, 0)$ and $V_3(0,\bullet)$ are set to $-\infty$       ▷ Optimal path in $V_3$ must come from $V_2$

Recursion 1:                                            ▷ Find optimal path for $V_1$
  for $i = 0, ..., n$ do                                ▷ Loop over sequence $x$
    for $j = 0, ..., m$ do                              ▷ Loop over sequence $y$
      $V_1(i,j) = \max \begin{cases} V_1(i-1,j-1) + \mathsf{s}(x_i, y_j) \\ V_1(i-1,j) + \mathsf{d} \\ V_1(i,j-1) + \mathsf{d} \end{cases}$    ▷ Select best move (match or indel).
    end for
  end for

Recursion 2:                                            ▷ Find optimal path for $V_2$
  for $i = 1, ..., n$ do                                ▷ Loop over sequence $x$
    $c = \max(V_1(i-1,\bullet))$;                       ▷ Best score for a jump from $V_1$
    for $j = m, ..., 1$ do                              ▷ Backwards loop over sequence $y$
      $V_2(i,j) = \max \begin{cases} c + \mathsf{s}(x_i, \mathsf{c}(y_j)) \\ V_2(i-1,j+1) + \mathsf{s}(x_i, \mathsf{c}(y_j)) \end{cases}$    ▷ Select best move (jump or match)
    end for
  end for

Recursion 3:                                            ▷ Find optimal path for $V_3$
  for $i = 1, ..., n$ do                                ▷ Loop over sequence $x$
    $c = \max(V_2(i-1,\bullet))$;                       ▷ Best score for a jump from $V_2$
    for $j = 1, ..., m$ do                              ▷ Loop over sequence $y$
      $V_3(i,j) = \max \begin{cases} c + \mathsf{s}(x_i, y_j) \\ V_3(i-1,j-1) + \mathsf{s}(x_i, y_j) \\ V_3(i-1,j) + \mathsf{d} \\ V_3(i,j-1) + \mathsf{d} \end{cases}$    ▷ Select best move (jump, match or indel)
    end for
  end for
```