Pairwise Alignment

- Given sequences $X = (X_1, \ldots, X_m)$ and $Y = (Y_1, \ldots, Y_n)$ constructed from a common alphabet (usually of size 4 or 20), the *gapped global alignment* of $X$ and $Y$ is defined by

  (i) an intercalation of $X$ with some number of gaps
  (ii) an intercalation of $Y$ with some number of gaps

  such that

  (a) both intercalations have the same length
  (b) at any given position in the intercalation, one of the two sequences does not have a gap.

For example, if the alphabet is $\{A, T, G, C\}$, and we denote a gap using $-$, then the following are gapped global alignments of $X = (A, G, G, A, T)$ and $Y = (T, G, A, T, T)$:

\[
\begin{align*}
\text{AGGAT} & \quad \text{AGGAT-} & \quad \text{AGGAT-} & \quad \text{AGGAT-} & \quad --\text{AGGAT} \\
\text{TGATT} & \quad \text{TG-ATT} & \quad \text{TG-ATT} & \quad -\text{TGATT} & \quad \text{TGAT--T}
\end{align*}
\]

- To compare different alignments, we need a scoring procedure. Suppose that $P(x, y)$ denotes the probability of letter $x$ being substituted for letter $y$ in the process of evolution. Let $P(x)$ and $P(y)$ denote the overall frequencies of letter $x$ and letter $y$. The log likelihood ratio $s(x, y) = \log P(x, y) - \log P(x) - \log P(y)$ measures the fit for a given pair of letters in one position of an alignment. We also need to specify the score for alignment against a gap. The simplest procedure is to take $s(x, -) = s(-, y) = -d$ for some $d > 0$. 

The score for the entire global alignment is the sum of the scores for each position in the alignment.

If we score \( s(x, x) = 2 \), \( s(x, y) = -1 \) for \( x \neq y \), and \( s(x, -) = s(-, y) = -2 \), then the previous examples receive scores 1, 1, 1, 1, -5.

- There is a one-to-many function from intercalations of \( X \) and \( Y \) to global alignments. For example, when \( m = n = 3 \), the intercalation \( AXYBCZ \) we would produce (among others)

\[
\begin{align*}
A--BC & \quad A-BC & \quad A--BC & \quad A--BC \\
XY--Z & \quad XY-Z & \quad -XY-Z & \quad -XY-Z \\
\end{align*}
\]

- An optimal gapped global alignment is a gapped global alignment that has a score at least as high as any other gapped global alignment.

The number of intercalations of a sequence of length \( m \) with a sequence of length \( n \) is \( \binom{m+n}{m} = \binom{m+n}{n} \). Thus the number of of alignments that would need to be considered in an exhaustive search for the optimal alignment would be prohibitive.

- The fact that the score is additive over distinct positions in the alignment allows the alignment to be computed much more efficiently.

- The key observation is that truncating an optimal alignment produces an optimal alignment of subsequences. Let \( G(a, b) \) be the optimal alignment of \( X_1, \ldots, X_a \) with \( Y_1, \ldots, Y_b \). There are three possibilities:

1. If \( X_a/- \) is the final column of the alignment, then \( G(a - 1, b) \) is the optimal alignment score for \( X_1, \ldots, X_{a-1} \) with \( Y_1, \ldots, Y_b \).
2. If $−/Y_b$ is the final column of the alignment, then $G(a, b−1)$ is the optimal alignment score for $X_1, \ldots, X_a$ with $Y_1, \ldots, Y_{b−1}$.
3. If $X_a/Y_b$ is the final column of the alignment, then $G(a−1, b−1)$ is the optimal alignment score for $X_1, \ldots, X_{a−1}$ with $Y_1, \ldots, Y_{b−1}$.

In each of the three cases, the optimal alignment itself is obtained by truncating the optimal alignment of the full sequences.

• The previous point describes the truncation of optimal alignments. Now we turn to the extension of optimal alignments. If we have the optimal alignment score $G(a, b)$, then just as truncating produces one of $G(a−1, b), G(a, b−1)$, or $G(a−1, b−1)$, then if we have knowledge of these three scores, we can produce $G(a, b)$ as

$$G(a, b) = \max \begin{cases} 
G(a−1, b) − d \\
G(a−1, b−1) + s(X_a, Y_b) \\
G(a, b−1) − d.
\end{cases}$$

When we compute the three quantities on the right to determine which is greatest, we retain a traceback pointer to whichever truncated alignment was extended.

• Following the above procedure, we can produce a table of all $G(a, b), a \leq m, b \leq n$, along with the traceback pointers. It is useful to add an extra column along the left edge, and an extra row along the top edge (corresponding to gaps).

To begin with, place $−kd$ for $k = 1, \ldots, m$ in the leftmost column, and $−kd$ for $k = 1, \ldots, n$ along the top row.
Suppose we are aligning \( X = GAA \) against \( Y = GATA \). The table of scores and the table of traceback pointers are:

<table>
<thead>
<tr>
<th></th>
<th>-</th>
<th>G</th>
<th>A</th>
<th>T</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
</tr>
<tr>
<td>G</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
</tr>
<tr>
<td>A</td>
<td>-4</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>-6</td>
<td>-2</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

In this case the optimal alignment is \( \text{GA-A/}GATA \) (the score can easily be verified to be 4).

- The **complexity** of finding the optimal global alignment is the number of elementary computations that must be performed (up to the order of magnitude). Since there are \( mn \) positions in the table of optimal scores, and the construction of each value in the table requires 3 additions and 3 comparisons, the operation count for constructing the table is on the order \( mn \). The complexity of constructing the optimal alignment itself once the table of scores is available is on the order of \( m + n \), so is negligible compared to the construction of the table.

- A **local alignment** is the alignment of a subsequence of \( X \) against a subsequence of \( Y \). The construction of the optimal gapped local alignment can be constructed recursively by filling in an \( m \times n \) table of alignment scores.
The key difference is that we start a new candidate local alignment whenever the score becomes negative. This is reflected in the fact that we change the update formula to:

\[ G(a, b) = \max \begin{cases} 
0 \\
G(a - 1, b) - d \\
G(a - 1, b - 1) + s(X_a, Y_b) \\
G(a, b - 1) - d.
\end{cases} \]

We also fill in the top-most and left-most rows with zero rather than \(-kd\).

- Once the alignment table is complete, we locate the highest score in the entire table (not just the lower-right corner), and trace back from this position until we reach zero. This path determines the optimal local alignment.

- The scores determine whether short local alignments tend to be locally optimal or whether long local alignments (or even global alignments) tend to be locally optimal.

Note that in the global alignment problem, if we add a constant \(C\) to all scores and to the gap penalty, the optimal alignment score is shifted by \(C\), and the optimal alignment itself is unaffected. That is not true in the case of local alignment. For example, if we add a large positive number to all the scores and to the gap so that all values are positive, the optimal local alignment will be the optimal global alignment (it can only improve the score to extend the alignment to the full sequence).

The size of the local alignment is controlled by the expected score for a random pairing of non-gaps. This is given by
\[
\sum_{x,y} s(x,y)p(x)p(y) = \sum_{x,y} \log(p(x,y)/p(x)p(y)) \cdot p(x)p(y) \\
\leq \log \sum_{x,y} p(x,y)/p(x)p(y) \cdot p(x)p(y) \\
= 0,
\]

where the interchange of the log and the sum follows from \textit{Jensen’s inequality}. This demonstrates that using the natural (log likelihood ratio-derived) scores, one can not obtain high scoring local alignments simply by taking alignments of long subsequences that don’t match well. In general, larger values of \(d\) will tend to produce shorter optimal local alignments.

- \textit{Repeated alignment} is another type of alignment that is of interest. In repeated alignment, one of the sequences is typically short relative to the other sequence, and the shorter sequence is aligned several times to the longer sequence.

We set a threshold \(T\) such that each repeated alignment of the shorter sequence against a subsequence of the longer sequence must score at least \(T\) in order to be considered.

- Fill in the first column of the table with zeros. Then use the following two updating rules to fill in the table:

\[
G(0, b) = \max \left\{ \begin{array}{l}
G(0, b - 1) \\
G(1, b - 1) - T \\
G(2, b - 1) - T \\
\vdots
\end{array} \right.
\]
\[ G(a, b) = \max \left\{ \begin{array}{l}
G(0, b) \\
G(a - 1, b - 1) + s(X_a, Y_b) \\
G(a - 1, b) - d \\
G(a, b - 1) - d
\end{array} \right. \]

When the \((m + 1) \times (n + 1)\) table is filled in, add one more cell, \(G(0, m + 1)\), using the first set of rules given above. The optimal repeated alignment is the traceback from this cell. Note that the optimal alignment score will have \(T\) times the number of passes through the shorter sequence subtracted from it.

A restart occurs when one of options 2, 3, \ldots in the first set of rules gives the maximum, and the top cell in the column lies on the optimal path. Since the first row must be non-negative this ensures that a restart only occurs when the previous alignment score exceeds \(T\).

When option 1 in the first block of rules is taken, the longer sequence is aligned against gaps (i.e. such a point is not in the middle of one of the repeated alignments of the shorter sequence).

Options 2, 3, 4 in the second block correspond to the usual lengthening of the alignment as in the global and local alignment algorithms.

A restart can begin before the shorter sequence has been completely aligned, but only if (i) the score of the aligned portion exceeds \(T\), and (ii) all extensions of the shorter sequence are downhill.

- **Tandem repeats:** If two or more copies of the shorter sequence are aligned against the longer sequence in immediate succession
(no gaps between repeats), we have a tandem repeat. In this case, it may be desirable to require the set of tandem repeats to exceed $T$, rather than requiring each separate repeat to exceed $T$. To achieve this, replace the second updating rule when $a = 1$ with

$$G(1, b) = \max \left\{ \begin{array}{l}
G(0, b - 1) + s(X_a, Y_b) \\
G(m, b - 1) + s(X_a, Y_b) \\
G(1, b - 1) - d \\
G(0, b) - d
\end{array} \right\}$$

- **Memory issues**: The optimal gapped global alignment requires order $mn$ operations to compute. If also requires storage (memory) of order $mn$ bytes. There is no generally-applicable method for improving the operation count, but the memory demand can be reduced substantially by implementing the algorithm recursively.

- First note that $G(m, n)$ can be computed using memory of order $m + n$, since at any given point in building up the table we only need to save the computed elements that border the elements yet to be computed (i.e. at most one row and one column).

- Let $m' = \lfloor m/2 \rfloor$, and let $C(i, j)$ be the column such that $(m', C(i, j))$ lies on the optimal path through the traceback pointers from $G(i, j)$ to $G(0, 0)$. The $C(i, j)$ can be computed as follows. First note that $C(i, j)$ is undefined if $i < m$. If $i = m$ then $C(i, j) = j$. If $i > m$ and the traceback pointer from $(i, j)$ leads to $(i', j')$, then $C(i, j) = C(i', j')$.

- Once we have computed all $C(m, n)$, we know that the optimal alignment of $X$ to $Y$ is equal to the concatenation of the optimal
alignment of $X_1, \ldots, X_{m'}$ to $Y_1, \ldots, Y_{C(m,n)}$ with the optimal alignment of $X_{m'+1}, \ldots, X_m$ to $Y_{C(m,n)+1} + 1, \ldots, Y_n$.

The recursion can continue until adequate memory is available to apply the direct algorithm.