# 3 PAIRWISE ALIGNMENTS

3 Pairwise alignments ..... 18
3.1. Aim of alignments ..... 19
3.2. Introduction ..... 19
3.3. Definitions ..... 20
3.4. Types of algorithms ..... 20
3.5. The scoring model ..... 21
3.5.1 Substitution matrices ..... 21
3.5.2 Gap penalties ..... 23
3.6. Alignment algorithms ..... 24
3.6.1 Aligning based on visual inspection ..... 24
3.6.2 Dynamic programming ..... 25
3.6.3 Parametric sequence alignments (niet kennen) ..... 33
3.7. Deriving score parameters from alignment data ..... 34
3.7.1 BLOcks SUbstitution Matrices (BLOSUM) ..... 34
3.7.2 PAM substitution matrices ..... 38
3.7.3 Comparison BLOSUM and PAM matrices ..... 41
3.7.4 Substitution (evolution) models for nucleic acid sequences ..... 42
3.8. Heuristic pairwise alignment algorithms ..... 43
3.8.1 Fasta ..... 43
3.8.2 Blast ..... 45
3.9. Statistical methods to score the significance of pairwise alignments ..... 47
3.9.1 Bayesian score (niet kennen) ..... 48
3.9.2 Extreme value distribution (niet kennen) ..... 49
3.10. Recent developments ..... 50

## 3.1. Aim of alignments

Sequence alignment is the procedure of comparing two (pairwise alignment) or more (multiple sequence alignment) sequences (DNA or protein) by searching for a series of individual characters or character patterns that are in the same order in the sequences. Two sequences are aligned by writing them across a page in two rows. Identical or similar characters are placed in the same column, and nonidentical characters can either be placed in the same column as a mismatch or opposite a gap in the other sequence.

```
sequence1 . A G C T T G - - C C T C G C A ...
sequence2 ..A G - T T G T T C C T G G C A ...
```

In an optimal alignment, nonidentical characters and gaps are placed to bring as many identical or similar characters as possible into vertical register. Sequences that can be readily aligned in this manner are said to be similar.

Sequence alignment is useful for discovering functional, structural and evolutionary information in biological sequences. Sequences that are very much alike ("i.e. similar") probably have the same function (either biochemical (for proteins) or regulatory function (for DNA molecules) or are important for the structural properties of the protein (DNA). Indeed if 2 sequences of different organisms are similar (in this case called homologous sequences) they might have been derived from a common ancestor sequence. The alignment indicates the sequence changes that have been occurred between the 2 homologous sequences and the common ancestor. Residues that are conserved during evolution are likely to be functionally or structurally relevant. Close inspection of a sequence alignment allows 1) to derive either useful structural and/or functional information on the biological function of the protein. 2) to reconstruct the phylogeny of an organism (see further).

## 3.2. Introduction

The most basic task in sequence analysis is to ask whether two sequences are similar? Answering this question is done by 1) aligning the two sequences and 2) subsequently deciding whether the alignment is biologically relevant (two sequences are related) or whether the alignment occurred by chance.

Key issues are:

1) what sorts of alignment should be considered (local versus global)
2) the scoring system to rank the alignments
3) algorithms to find alignments (versus heuristic)
4) statistical methods used to evaluate the significance of the alignment

Fig shows an example of pairwise alignments (identical positions indicated by the matching letter, similar positions indicated by a +). A distinction is made between identical residues and functional conservative residues (+ e.g. both negatively charged amino acids).

The first alignment is a biologically meaningful alignment, i.e. both sequences are evolutionarily related. Fig 2 is a spurious alignment (alignment of 2 completely unrelated sequences having a completely different structure and function).

How to distinguish biological meaningful alignments from spurious ones is the challenge for pairwise alignment methods? This is why the choice of the scoring scheme is so important. Even with the best methods it will not always be possible to make the distinction. As a rule of thumb, when aligning sequences one should try to maximize similarity between sequences by introducing as few gaps as possible.

## 3.3. Definitions

Global alignment: an attempt is made to align the entire sequences. Sequences that are quite similar and have approximately the same length are good candidates for global alignments (e.g. orthologous or paralogous sequences).

Local alignments: stretches of sequence with the highest density of matches are aligned generating islands of matches i.e. subalignments in the sequence. Local alignments are used for sequences with different length or sequences that contain only conserved domains but exhibit an overall low degree of conservation.

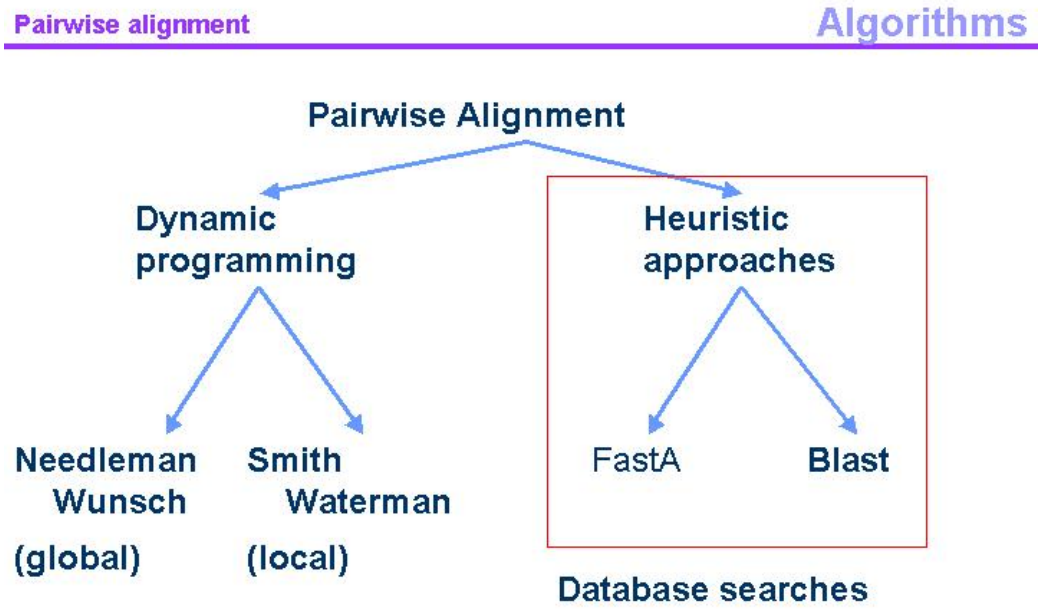
Homologous genes: genes exhibiting a considerable sequence similarity

Orthologous genes: homologous genes from distinct organisms.

Paralagous genes: homologous genes within the same organism, paralogous genes usually originate from gene duplication events

Remark that 2 proteins with the same catalytic domain (similar region) do not necessarily have been originated from a common ancestor. Nature has invented sometimes twice the same solution to a problem, this process is called convergent evolution. If sequences contain locally highly conserved domains that have been produced by convergent evolution they are called analogous.

## 3.4. Types of algorithms



A distinction is made between the algorithms that are based on dynamic programming and the heuristic algorithms. The ones based on dynamic programming allow to find the most optimal alignment between two sequences but they are slow. Therefore they are less suitable for homology searches against databases. To this end blast and fastA are used. But all algorithms make use of the same scoring system.

## 3.5. The scoring model

When sequences are compared, we are looking for evidence that they have diverged from a common ancestor by a process of natural selection. The basic mutational processes are

1) substitutions: change residues in a sequence,
2) insertions: adding residues and
3) deletions: removing residues.

In the context of an alignment insertions and deletions introduce gaps.




The total score of an alignment will be the sum of terms

For each aligned pair

Plus terms for gaps

In probabilistic terms: this corresponds to the log of the relative likelihood that the sequences are related compared to being unrelated.

The additive scoring schemes that is used in the following makes the assumption that mutations at different sites occur independently, an assumption that is reasonable for DNA and protein sequences but eg does not hold for RNA sequences (baseparings introduces long range dependencies).

### 3.5.1 Substitution matrices

Given a pair of aligned sequences, a score is assigned to each alignment that assesses the relative likelihood that the sequences are related (generated by a match model) as opposed to being unrelated (generated by the random model).

Consider 2 sequences x and y of lengths n and m respectively. Let xi denote the ith symbol of x and yj the jth symbol of y. These symbols can either be 4 nucleotides for DNA or 20 amino acids for proteins.

For instance consider an ungapped global pairwise alignment

Random model: assumes that each letter x occurs with some frequency $q_x$, and the probability of two sequences is just the product of the probabilities of each amino acid

$$p(x,y|R)=\prod_i q_{x_i}\prod_j q_{y_j}$$

Match model: aligned pairs of residues occur with a joint probability $p_{xy}$. Accordingly the probability of the whole alignment is

$$p(x,y|M)=\prod_i p_{x_i y_i}$$

The ratio of these two likelihoods, known as the odds ratio:

$$\frac{p(x,y|M)}{p(x,y|R)}=\prod_i \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}$$

To come to an additive score, the logarithm is taken: log-odds ratio

$$S=\sum_i s(x_i,y_i)$$

$$s(a,b)=\log\left(\frac{p_{ab}}{q_a q_b}\right)$$

s(a,b) is the log likelihood of the residue pair (a,b) occurring as an aligned pair as opposed to an unaligned pair. The score of the complete alignment is S, the sum of the separate terms. These s(a,b) scores are usually arranged in a matrix (20 X 20 matrix for amino acids) known as the score or the substitution matrix (e.g. BLOSUM PAM matrices, see below). The probability of changing amino acid A to B is always assumed to be identical to the reverse probability of changing B to A.

| | C | S | T | P | A | G | N | D | E | Q | H | R | K | M | I | L | V | F | Y | W | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | 9 | | | | | | | | | | | | | | | | | | | | C |
| S | -1 | 4 | | | | | | | | | | | | | | | | | | | S |
| T | -1 | 1 | 5 | | | | | | | | | | | | | | | | | | T |
| P | -3 | -1 | -1 | 7 | | | | | | | | | | | | | | | | | P |
| A | 0 | 1 | 0 | -1 | 4 | | | | | | | | | | | | | | | | A |
| G | -3 | 0 | -2 | -2 | 0 | 6 | | | | | | | | | | | | | | | G |
| N | -3 | 1 | 0 | -2 | -2 | 0 | 6 | | | | | | | | | | | | | | N |
| D | -3 | 0 | -1 | -1 | -2 | -1 | 1 | 6 | | | | | | | | | | | | | D |
| E | -4 | 0 | -1 | -1 | -1 | -2 | 0 | 2 | 5 | | | | | | | | | | | | E |
| Q | -3 | 0 | -1 | -1 | -1 | -2 | 0 | 0 | 2 | 5 | | | | | | | | | | | Q |
| H | -3 | -1 | -2 | -2 | -2 | -2 | 1 | -1 | 0 | 0 | 8 | | | | | | | | | | H |
| R | -3 | -1 | -1 | -2 | -1 | -2 | 0 | -2 | 0 | 1 | 0 | 5 | | | | | | | | | R |
| K | -3 | 0 | -1 | -1 | -1 | -2 | 0 | -1 | 1 | 1 | -1 | 2 | 5 | | | | | | | | K |
| M | -1 | -1 | -1 | -2 | -1 | -3 | -2 | -3 | -2 | 0 | -2 | -1 | -1 | 5 | | | | | | | M |
| I | -1 | -2 | -1 | -3 | -1 | -4 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | 1 | 4 | | | | | | I |
| L | -1 | -2 | -1 | -3 | -1 | -4 | -3 | -4 | -3 | -2 | -3 | -2 | -2 | 2 | 2 | 4 | | | | | L |
| V | -1 | -2 | 0 | -2 | 0 | -3 | -3 | -3 | -2 | -2 | -3 | -3 | -2 | 1 | 3 | 1 | 4 | | | | V |
| F | -2 | -2 | -2 | -4 | -2 | -3 | -3 | -3 | -3 | -3 | -1 | -3 | -3 | 0 | 0 | 0 | -1 | 6 | | | F |
| Y | -2 | -2 | -2 | -3 | -2 | -3 | -2 | -3 | -2 | -1 | 2 | -2 | -2 | -1 | -1 | -1 | -1 | 3 | 7 | | Y |
| W | -2 | -3 | -2 | -4 | -3 | -2 | -4 | -4 | -3 | -2 | -2 | -3 | -3 | -1 | -3 | -2 | -3 | 1 | 2 | 11 | W |
| | C | S | T | P | A | G | N | D | E | Q | H | R | K | M | I | L | V | F | Y | W | |

BLOSUM 50 matrix

Identities (eg C-C) and conservative substitutions (L-I) are expected to be more likely in alignments than we expect by chance, so they contribute to positive scores. In real alignment non conservative replacements are expected to be observed less frequently than we expect by chance so these contribute to negative score terms (eg aromatic amino acid (red) with positively charged amino acid (pink)).

The log odds scores in this Table lie within the range of -8 to +17. A value of 0 indicates that the frequency of the substitution between a matched pair of amino acids in related proteins is as expected by chance: a value less than 0 or greater than 0 indicates that the frequency is less than or greater than that expected by chance, respectively. Using such a matrix, a high positive score between two amino acids means that the pair is more likely to be found aligned in sequences that are derived from a common ancestor, i.e. homologous, than in unrelated or nonhomologous sequences. The highest-scoring replacements are for amino acids whose side chains are chemically similar, as might be expected if the amino acid substitution is not to impede function.

### 3.5.2 Gap penalties

Gap penalties are used to assign a negative score to the introduction of gaps (insertions, deletions).

```
IGALx        IGx--
LGy--        LGVLy
```

Two types of gap scores have been defined: a linear score and an affine score:

$\gamma(g) = -gd$

$\gamma(g) = -d - (g-1)e$

with g gap length
with d gap open penalty
with e gap extension penalty

In the affine gap penalty, usually the gap extension penalty is set less than the gap open penalty. As such long gaps (long deletions and insertions) are less penalized than when using the linear gap penalty.
In contrast to the substitution matrices there is no time dependent gap model. In practice people choose gap costs empirically once they have chosen their substitution scores. The range of the values of gap penalties have to be chosen to balance the scores in the scoring matrix that is used e.g. if too high the range of the gap penalty as compared to the range of the scores in the score matrix gaps will never appear in the alignment. Fortunately most programs will suggest gap penalties that are appropriate for a given scoring matrix.

## 3.6. Alignment algorithms

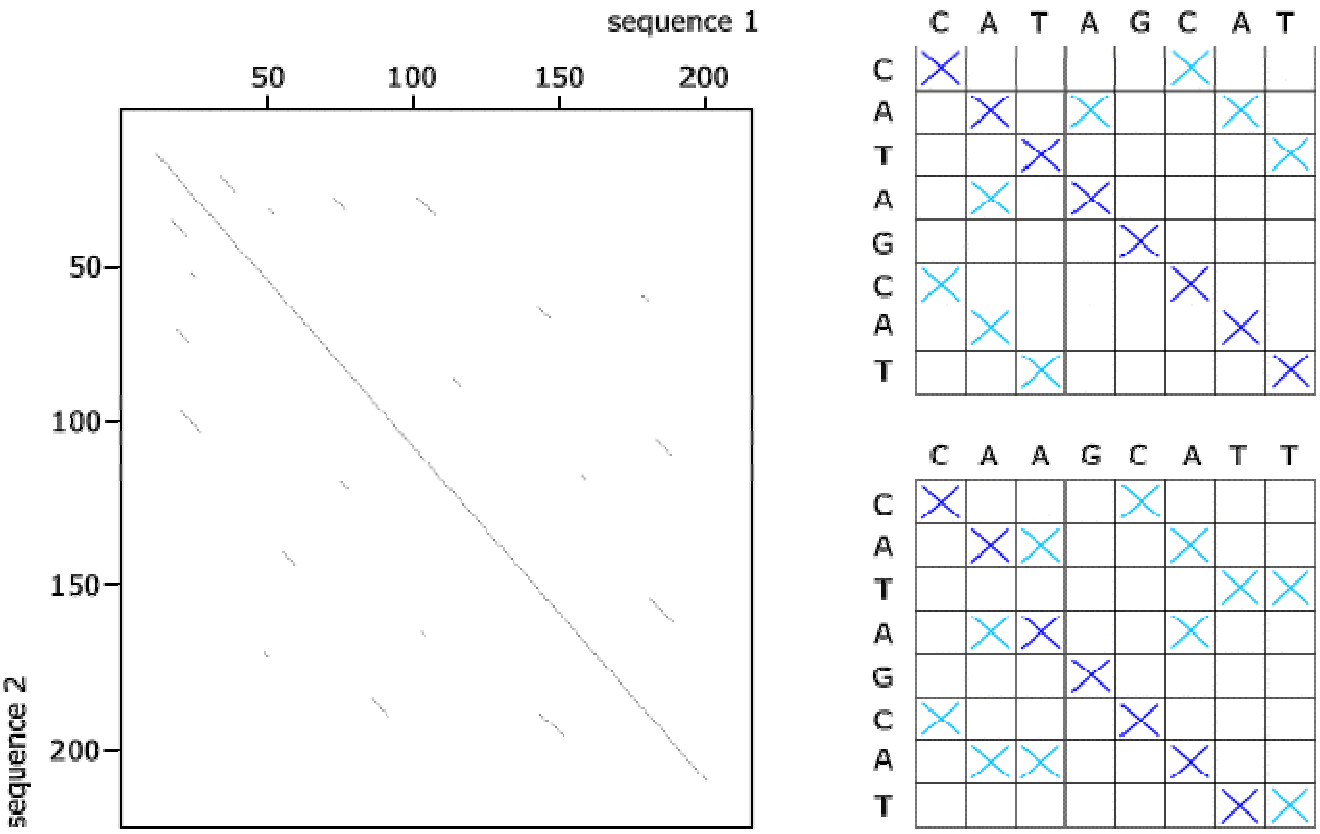
Besides a scoring system, we need an algorithm for finding an optimal alignment for a pair of sequences. Suppose there are 2 sequences of length n that need to be aligned Then there are

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2} \approx \frac{2^{2n}}{\sqrt{2\pi n}}$$

possible alignments between the 2 sequences. It is not computationally feasible to enumerate them all, even for moderate values of n.(formule hoef je niet te kennen of te snappen)

### 3.6.1 Aligning based on visual inspection

The simplest way to compare and align two sequences is a visual way, through the construction of a dot plot. In a dot plot, the sequences are written along the borders of a 2-dimensional table (see below), and a dot is placed on the crossing of rows and columns where the same 'character' (for example, A, C, G, or T in nucleic acids) is found. A diagonal line then shows regions of identity between the two sequences. Isolated dots not on the diagonal represent random matches that are probably not related to any significant alignment. Detection of matching regions is improved by filtering out random matches in a dot matrix. Filtering is achieved by using a sliding window to compare the two sequences. Instead of comparing single sequence positions a window of adjacent positions In the two sequences is compared at the same time, and a dot is printed only is a certain minimal number of matches occur. PAM and BLOSUM matrices are sometimes used to measure the degree of similarity in a window.




Dotplots are also still used to compare chromosomes and genomes in order to find homologous or paralogous regions of genes (syntheny).

### 3.6.2 Dynamic programming

offers a solution to this problem. A dynamic programming algorithm solves a problem by combining solutions to subproblems and is typically used when a problem has many possible solutions and an optimal one needs to be found (e.g. traveling salesman problem). Dynamic programming in aligning sequences is very important because it has been proven mathematically to




produce the optimal alignment between sequences under a given set of conditions.

Besides the algorithms that allow detection of the optimal solution, heuristic approaches have been developed. These heuristic searches make additional assumptions on the problem that very much shorten the computational time. The drawback is that finding the optimal solution is not guaranteed (see further) but they are useful when a sequences needs to be aligned against each single entry in a complete database of sequences (Blast, FastA).

Because a scoring scheme of log odds ratio was introduced, better alignments will have higher scores. Finding the optimal alignment implies maximizing the score.

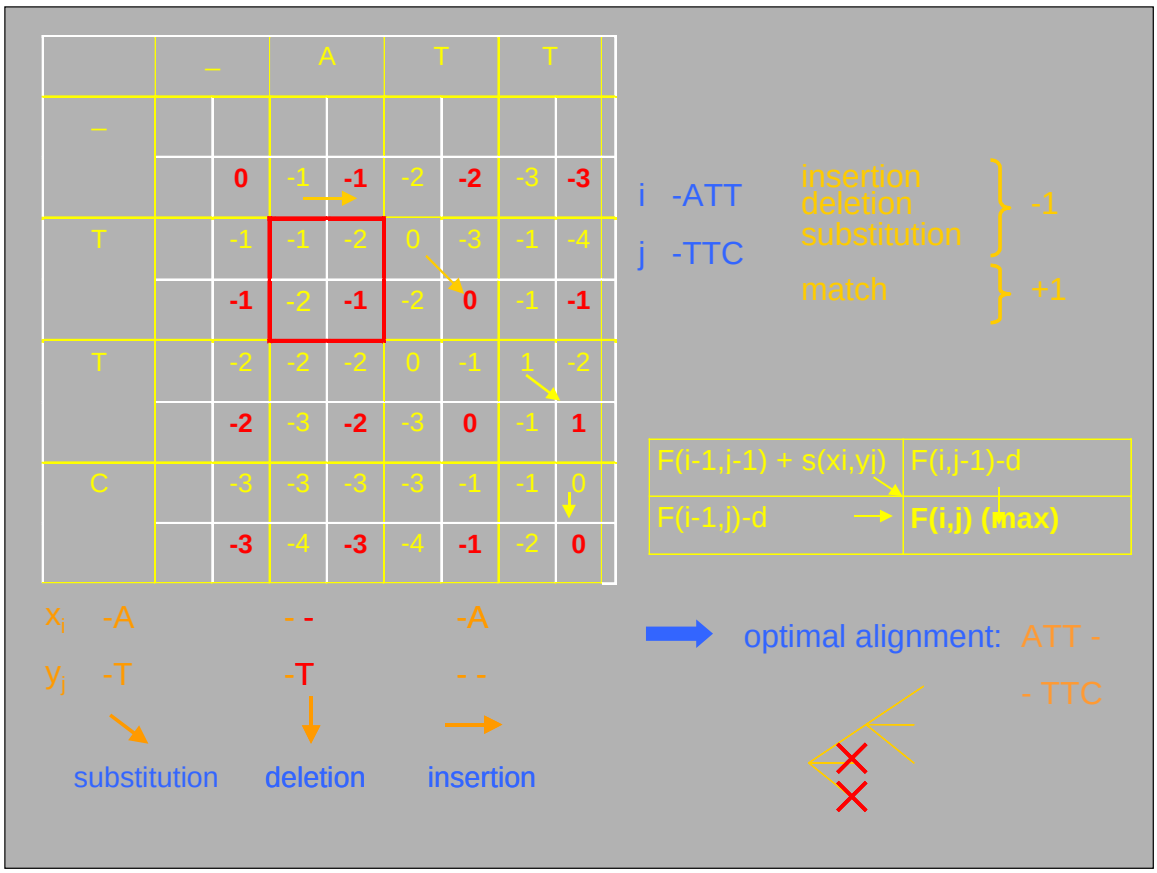
To illustrate the backtracking algorithm a simplified example on DNA was used because the scoring system is very simple: matches have score 1 while mismatches get score –1. Gaps (insertions and deletions are also penalized) by a score of –1.
Suppose 2 very short DNA sequences need to be aligned seq i ATT and seq j TTC.

1) Construct matrix F, indexed by i and j. ; place sequence 1 ATT across the top of the matrix and sequence 2 TTC down the left side. Leave an extra row and an extra column before each sequence labeled GAP - to allow for gaps.
2) F(i, j) is the score of the best alignment between the initial segment xi…1 of x up to xi and the initial segment y1…j up to yj. Build F(i, j) recursively: start at F(0, 0) = 0 and proceed to fill the matrix from top left to bottom right. Horizontal and vertical movements in the matrix D refer to insertions or deletions, while diagonal moves refer to matches or mismatches and their respective cost (taken from eg. PAM250 matrix). A certain cell (i, j) in the matrix can only be reached through its predecessors (i-1, j-1), (i-1, j) or (i, j-1). As a result, D[i, j] depends only on the values of these predecessors and the new value is computed by taken the minimum of D[i-1, j-1]+s($a_i$, $b_i$), D[i-1, j]+d and D[i, j-1]+d. Now the values of all cells can be computed from left to right and from top to bottom. For each matrix position 3 distinct trial values are calculated from all moves into that position allowed by the algorithm.

Starting from F(0,0) three different trials can be made:

- Align A with T: this corresponds to a move on the diagonal and corresponds to a substitution (score –1),
- Align A with gap: this corresponds to a move on the horizontal and corresponds to an insertion in the first sequence (score –1), subsequently move vertical this corresponds to a gap in the first sequence and an insertion of a T in the second sequence
- Align a gap with T: this corresponds to a move on the vertical and corresponds to a deletion in the first sequence (score –1) adnd a T in the second sequence, subsequently move horizontal this corresponds to a gap in the second sequence and an insertion of a A in the first sequence
-

From all trials the one with the highest score
3) Keep a pointer in each cell back to the cell from which it was derived
4) Value of the final cell is the best score for the alignment: The element D[m, n] will then give the ultimate distance (cost) between both sequences. The alignment can be written down by tracing back in the matrix to find back the path of choices that led to the final alignment score. The alignment is built by starting from the last cell, following the pointers that were stored when building the matrix. At each step in the traceback process we move back from the current cell (i,j) to the one of the cells (i-1,j-1), (i-1,j), (i,j-1) from which the value F(i,j) was derived). At the same time we add a pair of symbols onto the front of the current alignment. $x_i$ and $y_j$ if the step was to (i-1, j-1), $x_i$ and a gap if the step was to (i-1,j) or $y_j$ and a gap if the step was to (i, j-1).




One of the first and most applied algorithms, based on dynamic programming, to globally align two sequences is **the algorithm of Needleman & Wunsch** (1970). A related algorithm to align 2 sequences locally is that of **Smith & Waterman** (1981). Additional modifications have been implemented to align repeated matches, overlapping matches and to introduce affine gapscores. The algorithm described above needs to store (n+1)X(m+1) numbers and each number requires a constant number of calculations (3 sums and a max). The algorithm is said to take O(nm) time and O(nm) memory i.e. the computational time and the memory scale with the product of the sequence lengths. Because usually n and m are comparable the algorithm is said to be O(n2). Such algorithms are a little slow but feasible on current computers.

In the following additional examples for protein sequences are given. In the first example the algorithm for a global alignment using a more complex scoring model is given ((PAM250) and affine gap score) . In the second example the modifications that allow a local alignment are illustrated. More information can also be found in the book of David W. Mount (2001), of which also the following examples were taken.

**Example 1:** This example illustrates a global alignment of two hypothetical sequences, sequence 1 = MNAL**SDRT** and sequence 2 = MG**SDRT**TET. Notice the subsequence SDRT in the two sequences which one might expect to be aligned if the sequences are aligned properly.

The first step is to prepare a 8 x 9 matrix (lengths of the sequences) and place sequence 1 across the top of the matrix and sequence 2 down the left side, as in a dotplot. Leave an extra row and an extra column before each sequence labeled GAP to allow for gaps at the end of alignment. Fill in the extra row and column with the penalties for gaps of length zero to 8. The gap penalty used here is GAP = - 12 - 4 (x - 1), where x is the length of the gap. -12 is the penalty for opening the gap in the

alignment, and -4 is the penalty for each additional sequence character in the gap (this is usually referred to as the **affine gap cost**).

| | GAP | M | N | A | L | S | D | R | T |
|---|---|---|---|---|---|---|---|---|---|
| GAP | 0 | -12 | -16 | -20 | -24 | -28 | -32 | -36 | -40 |
| M | -12 | | | | | | | | |
| G | -16 | | | | | | | | |
| S | -20 | | | | | | | | |
| D | -24 | | | | | | | | |
| R | -28 | | | | | | | | |
| T | -32 | | | | | | | | |
| T | -36 | | | | | | | | |
| E | -40 | | | | | | | | |
| T | -44 | | | | | | | | |

Shown in parentheses are examples for the four possible matches between the first two amino acids. These scores are taken from the log odds form of the Dayhoff scoring matrix at 250 PAMs described above.

In a second step we have to fill in the score for each amino acid pair in the matrix. All possible paths start in O = (0,0) and travel through the matrix until T = (m,n). Horizontal and vertical movements in the matrix D refer to insertions or deletions, while diagonal moves refer to matches or mismatches and their respective cost (taken from eg. PAM250 matrix). A certain cell (i, j) in the matrix can only be reached through its predecessors (i-1, j-1), (i-1, j) or (i, j-1). As a result, D[i, j] depends only on the values of these predecessors and the new value is computed by taken the minimum of D[i-1, j-1]+s($a_i$, $b_i$), D[i-1, j]+d and D[i, j-1]+d. Now the values of all cells can be computed from left to right and from top to bottom. The element D[m, n] will then give the ultimate distance (cost) between both sequences.

| | GAP | M | N | A | L | S | D | R | T |
|---|---|---|---|---|---|---|---|---|---|
| GAP | 0 | -12 | -16 | -20 | -24 | -28 | -32 | -36 | -40 |
| M | -12 | (6) | (-2) | | | | | | |
| G | -16 | (-3) | (0) | | | | | | |
| S | -20 | | | | | | | | |
| D | -24 | | | | | | | | |
| R | -28 | | | | | | | | |
| T | -32 | | | | | | | | |
| T | -36 | | | | | | | | |
| E | -40 | | | | | | | | |
| T | -44 | | | | | | | | |

| | | C | S | T | P | A | G | N | D | E | Q | H | R | K | M | I | L | V | F | Y | W |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cys | C | 12 | | | | | | | | | | | | | | | | | | | |
| Ser | S | 0 | 2 | | | | | | | | | | | | | | | | | | |
| Thr | T | -2 | 1 | 3 | | | | | | | | | | | | | | | | | |
| Pro | P | -3 | 1 | 0 | 6 | | | | | | | | | | | | | | | | |
| Ala | A | -2 | 1 | 1 | 1 | 2 | | | | | | | | | | | | | | | |
| Gly | G | -3 | 1 | 0 | -1 | 1 | 5 | | | | | | | | | | | | | | |
| Asn | N | -4 | 1 | 0 | -1 | 0 | 0 | 2 | | | | | | | | | | | | | |
| Asp | D | -5 | 0 | 0 | -1 | 0 | 1 | 2 | 4 | | | | | | | | | | | | |
| Glu | E | -5 | 0 | 0 | -1 | 0 | 0 | 1 | 3 | 4 | | | | | | | | | | | |
| Gln | Q | -5 | -1 | -1 | 0 | 0 | -1 | 1 | 2 | 2 | 4 | | | | | | | | | | |
| His | H | -3 | -1 | -1 | 0 | -1 | -2 | 2 | 1 | 1 | 3 | 6 | | | | | | | | | |
| Arg | R | -4 | 0 | -1 | 0 | -2 | -3 | 0 | -1 | -1 | 1 | 2 | 8 | | | | | | | | |
| Lys | K | -5 | 0 | 0 | -1 | -1 | -2 | 1 | 0 | 0 | 1 | 0 | 3 | 5 | | | | | | | |
| Met | M | -5 | -2 | -1 | -2 | -1 | -3 | -2 | -3 | -2 | -1 | -2 | 0 | 0 | 6 | | | | | | |
| Ile | I | -2 | -1 | 0 | -2 | -1 | -3 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | 2 | 5 | | | | | |
| Leu | L | -6 | -3 | -2 | -3 | -2 | -4 | -3 | -4 | -3 | -2 | -2 | -3 | -3 | 4 | 2 | 6 | | | | |
| Val | V | -2 | -1 | 0 | -1 | 0 | -1 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | 2 | 4 | 2 | 4 | | | |
| Phe | F | -4 | -3 | -3 | -5 | -4 | -5 | -4 | -6 | -5 | -5 | -2 | -4 | -5 | 0 | 1 | 2 | -1 | 9 | | |
| Tyr | Y | 0 | -3 | -3 | -5 | -3 | -5 | -2 | -4 | -4 | -4 | 0 | -4 | -4 | -2 | -1 | -1 | -2 | 7 | 10 | |
| Trp | W | -8 | -2 | -5 | -6 | -6 | -7 | -4 | -7 | -7 | -5 | -3 | 2 | -3 | -4 | -5 | -2 | -6 | 0 | 0 | 17 |

So we calculate the score in each of the above positions. The maximum score of the M/M position is the GAP/GAP score of 0 plus 6 for an M/M match, or 6. The arrow indicates the previous matrix position that was used to obtain a score of 6; i.e., the box labeled with a score of 0. Similarly, the maximum possible score in the N/M position is 6 - 12 (one gap penalty) = -6, that of the M/G position is 6 - 12 = -6, and that of the N/G position is 6 + 0 = 6 (no gap penalty). Note that each sequential row and column must be completed before moving to a lower row or more rightward column. The next step is to complete the matrix by choosing at each position the maximum possible score while keeping track of all moves made to reach a maximum score at each position.

| | GAP | M | N | A | L | S | D | R | T |
|---|---|---|---|---|---|---|---|---|---|
| GAP | 0 | -12 | -16 | -20 | -24 | -28 | -32 | -36 | -40 |
| M | -12 | 6 (6) | -6 (-2) | | | | | | |
| G | -16 | -6 (-3) | 6 (0) | | | | | | |
| S | -20 | | | | | | | | |
| D | -24 | | | | | | | | |
| R | -28 | | | | | | | | |
| T | -32 | | | | | | | | |
| T | -36 | | | | | | | | |
| E | -40 | | | | | | | | |
| T | -44 | | | | | | | | |

| | GAP | M | N | A | L | S | D | R | T |
|---|---|---|---|---|---|---|---|---|---|
| GAP | 0 | -12 | -16 | -20 | -24 | -28 | -32 | -36 | -40 |
| M | -12 | 6 (6) | -6 (-2) | -10 | -14 | -18 | -22 | -26 | -30 |
| G | -16 | -6 (-3) | 6 (0) | -5 | -10 | -13 | -17 | -22 | -26 |
| S | -20 | -10 | -5 | 7 | -5 | -8 | -13 | -17 | -21 |
| D | -24 | -14 | -8 | -5 | 3 | -5 | -4 | -14 | -17 |
| R | -28 | -18 | -14 | -10 | -8 | 3 | -6 | 2 | -10 |
| T | -32 | -22 | -18 | -13 | -12 | -7 | 3 | -7 | 5 |
| T | -36 | -26 | -22 | -17 | -15 | -11 | -7 | 2 | -4 |
| E | -40 | -30 | -25 | -22 | -20 | -15 | -8 | -8 | 0 |
| T | -44 | -34 | -30 | -24 | -24 | -21 | -15 | -9 | -5 |

The **scoring matrix** is completed to find the highest score at each matrix position. The process has been continued to fill in the entire matrix with the maximum possible score at each matrix position. The right column and lowest row are then examined for the highest possible score because the alignment is a global one, meaning that the alignment will end only when the end of one of the sequences has been reached. Any remaining unmatched sequence will be opposite gaps. The highest-scoring box in the right-hand column and lowest row is a 5 in row 7. If end gaps were not being penalized, this would be the end of the search for the best score. However, if the alignment were to end here, there are three unmatched positions left in sequence 2, and each will be opposite a gap. Thus, an additional penalty score for three gaps (-20) corresponding to the heavy dotted line will have to be subtracted from 5, leaving an alignment score of 5 - 20 = -15. By subtracting any remaining end gap penalties from all positions in the last column and bottom row (not shown), one

finds that the best score is actually -5 in the right-hand, lowest corner of the matrix obtained by a diagonal move to this position, giving a score of -8+3 = -5.
The task then is to find a path back through the matrix using the moves made to get to that highest-scoring position and stopping at the beginning of one of the sequences. When the path turns from a previous diagonal move, a gap is placed opposite the next character in sequence 1 if the path turns upward and sequence 2 if it turns to the left. Two paths, shown as darker lines, are possible. These correspond to the alignments 1 and 2 shown below. Note that if end gap penalties were not used, the path shown by the light dotted line would be the correct alignment. This alignment, alignment 3, is also shown below. Note that wherever there are two paths leading to a matrix position, i.e., two possible ways of achieving that score, two alternative alignments will branch from that position.

**Alignment 1.**

```
sequence    1      M     -     N     A     L     S     D     R     T
sequence    2      M     G     S     D     R     T     T     E     T
score       6 -12  1  0 -3  1  0 -1  3  =  -5
```

Although this alignment has a low and insignificant score of -5, it is the best-scoring alignment that can be made between these two short sequences with the **Needleman-Wunsch algorithm** with end gaps penalized. Normally, it only makes sense to use a global alignment method for producing an alignment between sequences that are about the same length and that are expected to align along their entire lengths. The end gap penalty forces the ends to align. For sequences that are quite similar along their lengths, using end gap penalties will not have the dramatic effect that it does in this hypothetical example.

**Alignment 2.**

```
sequence    1      M     N     A     -     L     S     D     R     T
sequence    2      M     G     S     D     R     T     T     E     T
score       6  0   1 -12 -3  1  0 -1  3  =  -5
```

This second alignment is found by the trace-back procedure because there were two possible paths at one location in the matrix. This result illustrates that the dynamic programming alignment method may find more than one alignment having the same.

**Alignment 3.** (no end gap penalty included)

```
sequence    1      M     N     A     L     S     D     R     T     -     -     -
sequence    2      -     -     M     G     S     D     R     T     T     E     T
score       0  0 -1 -4  2  4  6  3  0  0  0  =  10
```

On initial observation, this alignment has a great deal more appeal than the above two and has a much higher score. However, all of the gaps needed to make the alignment have been put on the ends, where they do not count. There is a positive effect of helping to identify the region of similarity SDRT without scoring end gaps, but using the Needleman-Wunsch algorithm, this region could still be missed if this amino acid pattern was broken by higher-scoring neighboring alignments.

Detailed matrix:

| GAP | | M | | N | | A | | L | | S | | D | | R | | T | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | -12 | -12 | -16 | -16 | -20 | -20 | -24 | -24 | -28 | -28 | -32 | -32 | -36 | -36 | -40 | -40 |
| M | -12 | 6 | -24 | -14 | -28 | -19 | -32 | -16 | -36 | -26 | -40 | -31 | -44 | -29 | -48 | -37 | -52 |
| | -12 | -24 | 6 | -6 | -6 | -10 | -10 | -14 | -14 | -18 | -18 | -22 | -22 | -26 | -26 | -30 | -30 |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G | -16 | | -6 | 6 | -18 | -5 | -22 | -14 | -26 | -13 | -30 | -19 | -34 | -25 | -32 | -26 | -42 |
| | -16 | -28 | -6 | -18 | 6 | -6 | -5 | -17 | -14 | -26 | -13 | -17 | -17 | -21 | -21 | -25 | -25 |
| S | -20 | -18 | -10 | -5 | -6 | 7 | -17 | -8 | -26 | -12 | -25 | -13 | -29 | -17 | -33 | -20 | -37 |
| | -20 | -32 | -10 | -22 | -5 | -17 | 7 | -5 | -5 | -9 | -12 | -24 | -13 | -25 | -17 | -29 | -20 |
| D | -24 | -23 | -14 | -8 | -17 | -5 | -5 | 3 | -17 | -5 | -24 | -8 | -25 | -14 | -29 | -17 | -32 |
| | -24 | -36 | -14 | -36 | -8 | -20 | -5 | -17 | 3 | -9 | -5 | -17 | -8 | -20 | -14 | -26 | -17 |
| R | -28 | -24 | -18 | -14 | -20 | -10 | -17 | -8 | -9 | 3 | -17 | -6 | -20 | 0 | -26 | -15 | -29 |
| | -28 | -40 | -18 | -30 | -14 | -26 | -10 | -22 | -8 | -20 | 3 | -9 | -6 | -18 | 0 | -12 | -15 |
| T | -32 | -29 | -22 | -18 | -26 | -13 | -22 | -12 | -20 | -7 | -9 | 3 | -18 | -7 | -12 | 3 | -27 |
| | -32 | -44 | -22 | -24 | -18 | -30 | -13 | -25 | -12 | -24 | -7 | -19 | 3 | -9 | -7 | -19 | 3 |

PS matrix bevat nog fouten

**Example 2 (ter illustratie, niet kennen):** The **Smith-Waterman local alignment algorithm** discussed below is specifically designed to find such conserved patterns of local alignment. Below is scoring matrix shown for a Smith-Waterman alignment of sequence 1, MNALSDRT, and sequence 2, MGSDRTTET. The same sequences as well as the PAM250 scoring matrix and gap penalty scores (-12 and -4 for gap opening and gap extension penalties, respectively) for internal and end gaps, were used. The major difference between this scoring matrix and the Needleman-Wunsch matrix is that there are no negative scores in the Smith-Waterman scoring matrix. The effect of this change is that an alignment can begin anywhere without receiving a negative penalty from a previously low-scoring alignment. Once an alignment has been built, it stops when negative alignment scores or the introduction of gaps reduces the following alignment scores to 0. Thus, only a portion of each sequence that was in this high-scoring region will be reported. Note that in this example the initial end gap penalty does not have any effect because all first row and column scores are 0, the minimum allowed by the Smith-Waterman algorithm. Because a gap penalty at the end of the alignment produces a score of zero, the end gap penalty similarly has no effect. To find the optimal local alignment, the highest-scoring position in the scoring matrix is located (15), and the trace-back from this position is followed up to a zero in the matrix. The resulting sequence alignment is shown below. As opposed to the complex moves in the Needleman-Wunsch matrix, which are designed to test many combinations of matches, mismatches, and gaps, **only simple diagonal moves were made in the Smith-Waterman matrix**. Thus, there is only one alignment starting from the highest position. However, many other lower-scoring alignments are apparent, such as the second highest-scoring alignment of MNA with MGS starting at the position that scores 7. Note that this second alignment does not include any of the same amino acids, and not even any of the same aligned amino acids, that were used in the first alignment. It is possible to have multiple local alignments that do use the same aligned amino acid pairs, as there was in the global alignment example given above, but there are no examples in this matrix.

| | GAP | M | N | A | L | S | D | R | T |
|---|---|---|---|---|---|---|---|---|---|
| GAP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M | 0 | 6 | 0 | 0 | 4 | 0 | 0 | 0 | 0 |
| G | 0 | 0 | 6 | 1 | 0 | 5 | 1 | 0 | 0 |
| S | 0 | 0 | 1 | 7 | 0 | 2 | 5 | 1 | 1 |
| D | 0 | 0 | 2 | 1 | 3 | 0 | 6 | 4 | 1 |
| R | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 12 | 3 |
| T | 0 | 0 | 0 | 1 | 0 | 1 | 3 | 0 | 15 |
| T | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 2 | 3 |
| E | 0 | 0 | 1 | 0 | 0 | 0 | 4 | 0 | 2 |
| T | 0 | 0 | 0 | 2 | 0 | 1 | 0 | 3 | 3 |

**Alignment:**

```
sequence        1       S       D       R       T
sequence        2       S       D       R       T
score       2 4 6 3 = 15
```

Sequence alignment determined by the above procedure. Note that the score of the alignment is the same as that shown by the highest-scoring position in the scoring matrix. The inclusion of any additional sequence would reduce the score below 15.
An online application to perform both Needleman-Wunsch or Smith-Waterman sequence alignment is available **here**

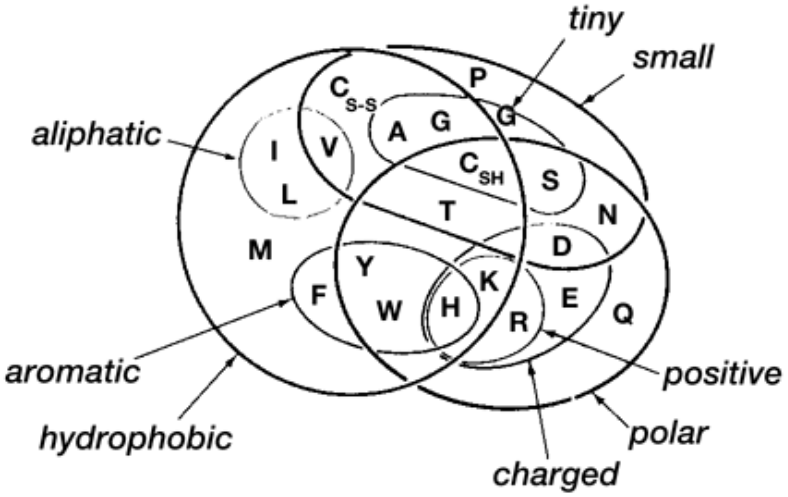
### 3.6.3 Parametric sequence alignments (niet kennen)

The problem with previous methods is that the alignment is dependent to a great extent on the choice of the parameter settings (one combination of parameters gives rise to one alignment). When 2 sequences are very homologous this does not pose a problem. If the degree of homology is not so pronounced and if several blocks are separated by gaps, the local alignment procedures as implemented above start to fail. This is a problem that often occurs when aligning intergenic sequences (see phylogenetic footprinting). Several novel alignment methods have been developed to that purpose:
Computer methods that find a range of possible alignments in response to varying the scoring system (different scoring matrices, gaps) are called parametric sequence comparisons. An example is the Bayes block aligner. Instead of having to try out different PAM matrices to estimate the best alignment and optimise the alignment score manually, the Bayes block aligner does this automatically. The output is not one alignment but a family of high scoring alignments. (nog verder).
Another class of alignments methods do not longer introduce “gaps”. Only homologous blocks are aligned, non homologous blocks are left unaligned (Dalign, DNA block aligner). (VISTA, Pipmaker)

## 3.7. Deriving score parameters from alignment data

In the first section we described how to score pairwise alignments by using substitution matrices. The left open issue is how to estimate these probabilities of substitution (i.e. how to construct the scoring matrices such as BLOSUM and PAM).

For protein sequences, some substitutions are much more likely than others and the performance of any alignment algorithm is improved when substitution matrices used to score the alignment accounts for this difference. There are two frequently used approaches to finding substitution matrices. One leads to the **PAM** (Accepted Point Mutation) family of matrices, the other to the **BLOSUM** (BLocks SUbstitution Matrices) family. In this section we will discuss how these substitution matrices are derived.




Any attempt to create a scoring matrix for amino acids substitutions must start from a set of data that can be trusted i.e. a good random sample of confirmed alignments. These 'trusted' data are then used to determine substitutions probabilities by simply counting the frequencies of the aligned residue pairs in the confirmed alignments and setting the probabilities to the normalized frequencies. This approach implies two major difficulties: 1) at first it is not straightforward to obtain a set random sample of confirmed alignments and 2) alignments are not independent from each other because they come in families. For two closely related sequences the probability should be small and hence s(a,b) should be strongly negative unless a = b. When a long time passed since two sequences diverged we expect $p_{ab}$ to be close to the background sequences $q_a q_b$ so s(a,b) should be close to 0 for all a,b. this implies that we should use scores that matched to the expected divergence of the sequences we wish to compare.

Although historically the PAM matrices were developed first (Dayhoff et al., 1978), the derivation of BLOSUM matrices is somewhat simpler than that for PAM matrices, and therefore we will start by considering BLOSUM matrices.

### 3.7.1 BLOcks SUbstitution Matrices (BLOSUM)

The BLOSUM approach was introduced by Henikoff and Henikoff (1992). They started with a set of protein sequences from public databases that had been grouped into related families. From these sequences they obtained 'blocks' of aligned sequences. A block is the 'ungapped' alignment of a relatively highly conserved region of a family of proteins. Methods for producing such alignments are discussed below.

These alignments provide the basic data for the BLOSUM approaches to constructing substitution matrices. An example of such an alignment leading to four blocks is given here:

```
WWYIR   CASILRKIYIYGPV   GVSRLRTAYGGRK   NRG
WFYVR   CASILRHLYHRSPA   GVGSITKIYGGRK   RNG
WYYVR   AAAVARHIYLRKTV   GVGRLRKVHGSTK   NRG
WYFIR   AASICRHLYIRSPA   GIGSFEKIYGGRR   RRG
WYYTR   AASIARKIYLRQGI   GVHHFQKIYGGRQ   RNG
WFYKR   AASVARHIYMRKQV   GVGKLNKLYGGAK   SRG
WFYKR   AASVARHIYMRKQV   GVGKLNKLYGGSK   RRG
 WYYVR   TASVARRLYIRSPT  GVGALRRVYGGNK   RRG
WFYTR   AASTARHLYLRGGA   GVGSMTKIYGGRQ   RNG
WWYVR   AAALLRRVYIDGPV   GVNSLRTHYGGKK   DRG
```

Since the algorithms used to construct the aligned blocks employ substitution matrices, there is a circularity involved in the procedure if the aligned blocks are subsequently used to find substitution matrices. Henikoff and Henikoff broke this circularity as follows. They started by using a simple 'unitary' substitution matrix where the score is 1 for a match and 0 for a mismatch. Then, using data from suitable groups of proteins, they constructed only those blocks that they could obtain with this simple matrix. This procedure has the effect of generating a conservative set of blocks, omitting blocks with low sequence identity. While this restricted the number of blocks derived, the blocks obtained were trustworthy and were not biased toward any specific scoring scheme.
Using the blocks so constructed, Henikoff and Henikoff then counted the number of occurrences of each amino acid and the number of occurrences of each pair of amino acids aligned in the same column. Consider a very simplified example, with only three amino acids, A, B, and C, and only one block:

```
B A B A
A A A C
A A C C
A A B A
A A C C
A A B C
```

In this block there are 24 amino acids observed, of which 14 are A, 4 are B, and 6 are C. The **observed** proportions are thus:

q(A) 14/24
q(B) 4/24
q(C) 6/24

There are **N(N-1)/2 * 4 aligned pairs** of amino acids in this block. These 60 pairs (N=6) occur with proportions as given in the following table:

p(A to A) 26/60
p(A to B) 8/60
p(A to C) 10/60
p(B to B) 3/60
p(B to C) 6/60
p(C to C) 7/60

We now compare these observed proportions to the **expected** proportion of times that each amino acid pair is aligned under a random assortment of the amino acids observed, given the observed amino acid frequencies. In other words, if we choose two sequences of the same length at random with the frequencies observed, and put them into alignment, then **the expected proportion of pairs in which A is aligned with A** is 14/24 * 14/24 (the observed frequency of A is 14/24 i.e. $q_{a*}$ $q_a$). The mean proportion of pairs in which A is aligned with B is 14/24 * 4/24 * 2 i.e. $q_{a*}$ $q_b$, and so on. The factor of 2 in the second calculation allows for the two cases where A is in the first sequence and B in the second, and that where B is in the first sequence and A in the second.

These fractions are now used to calculate 'estimated likelihood ratios' as shown in the following table:

| aligned pair | proportion observed | proportion expected | 2 $\log_2$(proportion observed/proportion expected) |
|---|---|---|---|
| A to A | 26/60 | 196/576 | 0.70 |
| A to B | 8/60 | 112/576 | -1.09 |
| A to C | 10/60 | 168/576 | -1.61 |
| B to B | 3/60 | 16/576 | 1.70 |
| B to C | 6/60 | 48/576 | 0.53 |
| C to C | 7/60 | 36/576 | 1.80 |

For each row in this table the ratio of the entries in the second and third columns is an estimate, from the data, of the ratio of the proportion of times that each amino acid combination occurs in any column to the proportion expected under random allocation of amino acids into columns. With one important qualification, described later, the respective elements in the BLOSUM substitution matrix are now found by calculating twice the logarithm of this ratio (as shown in the last column of the above table), and then rounding the result to the nearest integer. In this simplified example, the **substitution matrix** would thus be:

| | A | B | C |
|---|---|---|---|
| A | 1 | -1 | -2 |
| B | -1 | 2 | 1 |
| C | -2 | 1 | 2 |

$$s(a,b) = \log\left(\frac{p_{ab}}{q_a q_b}\right)$$

$$q_a = \frac{\sum_b A_{ab}}{\sum_{cd} A_{cd}}$$

$$s(a,b) = \log_2\left(\frac{p_{ab}}{q_a q_b}\right)$$

In general, the procedure is thus as follows. For each pair of amino acids *a* and *b*, they estimated $q_a$ i.e. the fraction of aligned pairs that include an a and $p_{ab}$ i.e. the fraction of pairings between *a* and *b* out of all observed pairs. From this they derived the scoring matrix entries using the standard equation s(a,b). This quantity is the ratio of the loglikelihood that *a* and *b* are actually observed aligned in the same column in the blocks to the probability that they are aligned by chance, given their frequencies of occurrence in the blocks,. The resulting logg odds values are scaled and rounded to the nearest integer value. **In this way, pairs that are more likely than chance will have positive scores, and those less likely will have negative scores**.

While this approach is still rudimentary, it does yield a more useful scoring scheme than the original one that merely scores 1 for a match and 0 for a mismatch. Its main shortcoming is that it overlooks an important factor that can bias the results. The substitution matrix derived will depend significantly on which sequences of each family happen to be in the database used to create the blocks. In particular, if there are many very closely related proteins in one block, and only a few others that are less closely related, then the contribution of that block will be biased toward closely related proteins. For example, suppose the data in one block are as follows:

```
A  B  A  A
A  B  A  A
A  B  A  A
A  B  A  A
A  A  B  D
A  C  B  A
D  A  B  A
```

The first four sequences possibly derive from closely related species and the last three from three more distant species. Since A occurs with high frequency in the first four sequences, the observed number of pairings of A with A will be higher than is appropriate if we are comparing more distantly related sequences. Ultimately, we would prefer in each block to have sequences such that any pair have roughly the same amount of 'evolutionary distance' between them. The solution to this problem used by Henikoff and Henikoff is to group, or cluster, those sequences in each block that are 'sufficiently close' to each other and, in effect, use the resulting cluster as a single sequence.
This step requires a definition of 'sufficiently close' and this is done by specifying a cut-off proportion, say 85%, and then grouping the sequences in each block into clusters in such a way that each sequence in any cluster has 85% or higher sequence identity to at least one other sequence in the cluster in that block.
Next is described how the counting is done in this case. Consider a simple example with two blocks.

```
B      A      B      A
B      A      B      C
A      A      C      C

and

C      B      B
C      B      B
A      B      C
A      A      C
```

Suppose the identity for clustering is taken to be 0.75. Thus we cluster the first two sequences in each block together. The A's are counted as follows. The first column of the first block has one A, the second column contributes two A's, since the first two sequences are clustered it has 1 + 1/2 + 1/2 = 2 A's. The fourth column contributes 1/2 A. In the second block there are three A's, since each occurrence occurs in a cluster of size one. So, in total there are 6,5 A's. Now to get the proportion of A's, we must divide by 17, since each column of the first block contributes 2 to the counts of the symbols, and each column of the second block contributes 3 to the counts. So the proportion of A's is 6,5/17. We record the proportions for all symbols:

```
A      6,5/17
B      5/17
C      5,5/17
```

To count the A-B pairs, each occurrence in the first column of the first block contributes 1/2, and in the second column of the second block the contribution is 1/2 + 1/2 + 1. So the total A-B count is 3. There are a total of 13 pairs in the block, four in the first block (each column contributes one pair, or more precisely, two half pairs) and three in the second block (9+4). Thus the proportion of A-B pairs is 3/13. We record the proportions for all pairs of symbols in the following table:

```
A     to    A                2/13
A     to    B                3/13
A     to    C     5/26       (2.5/13)
B     to    B                1/13
B     to    C                3/13
C     to    C     3/26       (1.5/13)
```

The procedure is then carried out as before.
A further refinement was taken by Henikoff and Henikoff (1992). After obtaining a BLOSUM substitution matrix as just described, the matrix obtained is then used instead of the conservative 'unitary' matrix to construct a second, less conservative, set of blocks. A new substitution matrix is then obtained from these blocks. Then the process is repeated a third time. From this third set of blocks, Henikoff and Henikoff derive the final family of BLOSUM matrices, and it is these whose use is suggested.
If the 0.85 similarity score criterion is adopted, the final matrix is called a BLOSUM85 matrix. In general, if clusters with X% identity are used, then the resulting matrix is called BLOSUMX. The BLOSUM matrices currently available on the BLAST web page at NCBI are BLOSUM45, BLOSUM62, and BLOSUM80. Note that the larger-numbered matrices correspond to more recent divergence, and the smaller-numbered matrices correspond to more distantly related sequences. One often has prior knowledge about the evolutionary distance between the sequences of interest that helps one choose which BLOSUM matrix to use. With no information, BLOSUM62 is often used.

### 3.7.2 PAM substitution matrices

In this section we outline the Dayhoff et al. (1978) approach for deriving the so-called **PAM substitution matrices**. An 'accepted point mutation' is a substitution of one amino acid of a protein by another that is 'accepted' by evolution, in the sense that within some given species, the mutation has not only arisen but has, over time, spread to essentially the entire species. A **PAM1** transition **matrix is a matrix applying for a time period over which we expect 1% of the amino acids to undergo accepted point mutations within the species of interest.**
The construction of PAM matrices starts with ungapped multiple alignments of proteins into blocks for which all pairs of sequences in any block are, as in the BLOSUM procedure, 'sufficiently close' to each other. In the original construction of Dayhoff et al. (1978), the requirement was that each sequence in any block be no more than 15% different from any other sequence. This requirement resulted, for their data, in 71 blocks of aligned proteins. Imposing the requirement of close within-block similarity is aimed at minimizing the number of substitutions in the data that may have resulted from two or more consecutive substitutions at the same site. This is important because the initial goal is to create a transition matrix for a short enough time period so that multiple mutations are very unlikely to happen during this period.
The phylogenetic reconstruction method adopted for the data within each block in the database is the method of maximum parsimony. This algorithm constructs trees such as that the total number of substitutions across the tree is minimal. Such a tree is called a most parsimonious tree. In a maximum parsimony tree, the number of changes can be counted. Since, in general there will be more than one block in the data set, and if so, the counts (mutations from one amino acid to another one) from the different blocks will simply be added to obtain an overall count matrix. This is the original **PAM1 mutation matrix** ('original point substitution matrix for the evolutionary time of 1 PAM'), using the approach described above, as developed by Dayhoff (1979): Afleiding PAM niet kennen

| | Ala | Arg | Asn | Asp | Cys | Gln | Glu | Gly | His | Ile | Leu | Lys | Met | Phe | Pro | Ser | Thr | Trp |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ala | 0 | 30 | 109 | 154 | 33 | 93 | 266 | 579 | 21 | 66 | 95 | 57 | 29 | 20 | 345 | 772 | 590 | 0 |
| Arg | 30 | 0 | 17 | 0 | 10 | 120 | 0 | 10 | 103 | 30 | 17 | 477 | 17 | 7 | 67 | 137 | 20 | 27 |
| Asn | 109 | 17 | 0 | 532 | 0 | 50 | 94 | 156 | 226 | 36 | 37 | 322 | 0 | 7 | 27 | 432 | 169 | 3 |
| Asp | 154 | 0 | 532 | 0 | 0 | 76 | 831 | 162 | 43 | 13 | 0 | 85 | 0 | 0 | 10 | 98 | 57 | 0 |
| Cys | 33 | 10 | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 17 | 0 | 0 | 0 | 0 | 10 | 117 | 10 | 0 |
| Gln | 93 | 120 | 50 | 76 | 0 | 0 | 422 | 30 | 243 | 8 | 75 | 147 | 20 | 0 | 93 | 47 | 37 | 0 |
| Glu | 266 | 0 | 94 | 831 | 0 | 422 | 0 | 112 | 23 | 35 | 15 | 104 | 7 | 0 | 40 | 86 | 31 | 0 |
| Gly | 579 | 10 | 156 | 162 | 10 | 30 | 112 | 0 | 10 | 0 | 17 | 60 | 7 | 17 | 43 | 450 | 50 | 0 |
| His | 21 | 103 | 226 | 43 | 10 | 243 | 23 | 10 | 0 | 3 | 40 | 23 | 0 | 20 | 50 | 26 | 14 | 3 |
| Ile | 66 | 30 | 36 | 13 | 17 | 8 | 35 | 0 | 3 | 0 | 253 | 43 | 57 | 90 | 7 | 20 | 129 | 0 |
| Leu | 95 | 17 | 37 | 0 | 0 | 75 | 15 | 17 | 40 | 253 | 0 | 39 | 207 | 167 | 43 | 32 | 52 | 13 |
| Lys | 57 | 477 | 322 | 85 | 0 | 147 | 104 | 60 | 23 | 43 | 39 | 0 | 90 | 0 | 43 | 168 | 200 | 0 |
| Met | 29 | 17 | 0 | 0 | 0 | 20 | 7 | 7 | 0 | 57 | 207 | 90 | 0 | 17 | 4 | 20 | 28 | 0 |
| Phe | 20 | 7 | 7 | 0 | 0 | 0 | 0 | 17 | 20 | 90 | 167 | 0 | 17 | 0 | 7 | 40 | 10 | 10 |
| Pro | 345 | 67 | 27 | 10 | 10 | 93 | 40 | 43 | 50 | 7 | 43 | 43 | 4 | 7 | 0 | 269 | 73 | 0 |
| Ser | 772 | 137 | 432 | 98 | 117 | 47 | 86 | 450 | 26 | 20 | 32 | 168 | 20 | 40 | 169 | 0 | 696 | 17 |
| Thr | 590 | 20 | 169 | 57 | 10 | 37 | 31 | 50 | 14 | 129 | 52 | 200 | 28 | 10 | 73 | 696 | 0 | 0 |
| Trp | 0 | 27 | 3 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 13 | 0 | 0 | 10 | 0 | 17 | 0 | 0 |
| Tyr | 20 | 3 | 36 | 0 | 30 | 0 | 10 | 0 | 40 | 13 | 23 | 10 | 0 | 260 | 0 | 22 | 23 | 6 |
| Val | 365 | 20 | 13 | 17 | 33 | 27 | 37 | 97 | 30 | 661 | 303 | 17 | 77 | 10 | 50 | 43 | 186 | 0 |

The upper row shows the original amino acid, the left column shows the amino acid by which the original amino acid is replaced. This matrix of empirical observations will be transformed into a probability matrix by normalizing each entry.

On the basis of this substitution matrix, one can compute a so-called **mutation probability matrix**, considering the **mutability of amino acids** (computed by summing the columns and dividing by the frequency, then normalizing against Ala). This mutability takes into account the frequency of amino acids in the data set. In the dataset of Dayhoff, the frequencies of amino acid were:
In the following is shown how the expected frequency of each amino acid is calculated **($q_j$)**.

| C | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ala | Arg | Asn | Asp | Cys | Gln | Glu | Gly | His | Ile | Leu | Lys | Met | Phe | Pro | Ser | Thr | Trp | Tyr | Val |
| 8.7 | 4.1 | 4 | 4.7 | 3.3 | 3.9 | 5 | 8.9 | 3.4 | 3.7 | 8.5 | 8.1 | 1.5 | 4 | 5.1 | 7 | 5.8 | 1 | 3 | 6.5 |

From this is derived the mutability of the amino acids

| Amino acid mutabilities relative to Alanine | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ala | Arg | Asn | Asp | Cys | Gln | Glu | Gly | His | Ile | Leu | Lys | Met | Phe | Pro | Ser | Thr | Trp | Tyr | Val |
| 100 | 65 | 134 | 106 | 20 | 93 | 102 | 49 | 66 | 96 | 40 | 56 | 94 | 41 | 56 | 120 | 97 | 18 | 41 | 74 |

Based on these values and the original substitution matrix, one can compute a **mutation probability matrix** as follows: (with $c = \lambda m_j$)

$$M_{ij} = \frac{\lambda m_j A_{ij}}{\sum_i A_{ij}}$$ for the non-diagonal elements and

$M_{ij} = 1 - \lambda m_j$, for the diagonal elements,

and where $A_{ij}$ is an element of the substitution matrix M, $m_j$ is the mutability of amino acid j, and lambda is a constant (in this case, lambda equals 0.0133, when mutabilities are divided by 100). The resulting **mutation probability matrix** is:

| | Ala A | Arg R | Asn N | Asp D | Cys C | Gln Q | Glu E | Gly G | His H | Ile I | Leu L | Lys K | Met M | Phe F | Pro P | Ser S | Thr T | Trp W | Tyr Y | Val V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ala A | 9867 | 2 | 9 | 10 | 3 | 8 | 17 | 21 | 2 | 6 | 4 | 2 | 6 | 2 | 22 | 35 | 32 | 0 | 2 | 18 |
| Arg R | 1 | 9913 | 1 | 0 | 1 | 10 | 0 | 0 | 10 | 3 | 1 | 19 | 4 | 1 | 4 | 6 | 1 | 8 | 0 | 1 |
| Asn N | 4 | 1 | 9822 | 36 | 0 | 4 | 6 | 6 | 21 | 3 | 1 | 13 | 0 | 1 | 2 | 20 | 9 | 1 | 4 | 1 |
| Asp D | 6 | 0 | 42 | 9859 | 0 | 6 | 53 | 6 | 4 | 1 | 0 | 3 | 0 | 0 | 1 | 5 | 3 | 0 | 0 | 1 |
| Cys C | 1 | 1 | 0 | 0 | 9973 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 5 | 1 | 0 | 3 | 2 |
| Gln Q | 3 | 9 | 4 | 5 | 0 | 9876 | 27 | 1 | 23 | 1 | 3 | 6 | 4 | 0 | 6 | 2 | 2 | 0 | 0 | 1 |
| Glu E | 10 | 0 | 7 | 56 | 0 | 35 | 9865 | 4 | 2 | 3 | 1 | 4 | 1 | 0 | 3 | 4 | 2 | 0 | 1 | 2 |
| Gly G | 21 | 1 | 12 | 11 | 1 | 3 | 7 | 9935 | 1 | 0 | 1 | 2 | 1 | 1 | 3 | 21 | 3 | 0 | 0 | 5 |
| His H | 1 | 8 | 18 | 3 | 1 | 20 | 1 | 0 | 9912 | 0 | 1 | 1 | 0 | 2 | 3 | 1 | 1 | 1 | 4 | 1 |
| Ile I | 2 | 2 | 3 | 1 | 2 | 1 | 2 | 0 | 0 | 9872 | 9 | 2 | 12 | 7 | 0 | 1 | 7 | 0 | 1 | 33 |
| Leu L | 3 | 1 | 3 | 0 | 0 | 6 | 1 | 1 | 4 | 22 | 9947 | 2 | 45 | 13 | 3 | 1 | 3 | 4 | 2 | 15 |
| Lys K | 2 | 37 | 25 | 6 | 0 | 12 | 7 | 2 | 2 | 4 | 1 | 9926 | 20 | 0 | 3 | 8 | 11 | 0 | 1 | 1 |
| Met M | 1 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 5 | 8 | 4 | 9874 | 1 | 0 | 1 | 2 | 0 | 0 | 4 |
| Phe F | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 8 | 6 | 0 | 4 | 9946 | 0 | 2 | 1 | 3 | 28 | 0 |
| Pro P | 13 | 5 | 2 | 1 | 1 | 8 | 3 | 2 | 5 | 1 | 2 | 2 | 1 | 1 | 9926 | 12 | 4 | 0 | 0 | 2 |
| Ser S | 28 | 11 | 34 | 7 | 11 | 4 | 6 | 16 | 2 | 2 | 1 | 7 | 4 | 3 | 17 | 9840 | 38 | 5 | 2 | 2 |
| Thr T | 22 | 2 | 13 | 4 | 1 | 3 | 2 | 2 | 1 | 11 | 2 | 8 | 6 | 1 | 5 | 32 | 9871 | 0 | 2 | 9 |
| Trp W | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 9976 | 1 | 0 |
| Tyr Y | 1 | 0 | 3 | 0 | 3 | 0 | 1 | 0 | 4 | 1 | 1 | 0 | 0 | 21 | 0 | 1 | 1 | 2 | 9945 | 1 |

This matrix is thus a **mutation probability matrix** for an evolutionary distance of 1 PAM (which means sequences characterized by one amino acid replacement per 100 residues). One element in this matrix, [Mij], denotes the chance that an amino acid in column *j* will be replaced by an amino acid in row *i*, when these sequences have diverged over a 1 PAM distance. For example, the probability that *Asp* is replaced by *Glu* is 0.56% (both differ by only 1 $CH_2$ group). Note that all values have been multiplied by 10.000. The sum of each column has to equal 1. Tryptophane (W) is the biggest aromatic amino acid and is therefore difficult to replace without changing the function of the proteins, as can be clearly observed from the matrix.
In principle, we can derive such matrices for every PAM distance (by matrix multiplication: see book by Mount, page 82). The mutation probability matrix below for example is the very well known PAM250 matrix obtained by matrix multiplication of the PAM1 matrix.

| | Ala A | Arg R | Asn N | Asp D | Cys C | Gln Q | Glu E | Gly G | His H | Ile I | Leu L | Lys K | Met M | Phe F | Pro P | Ser S | Thr T | Trp W | Tyr Y | Val V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ala A | 13 | 6 | 9 | 9 | 5 | 8 | 9 | 12 | 6 | 8 | 6 | 7 | 7 | 4 | 11 | 11 | 11 | 2 | 4 | 9 |
| Arg R | 3 | 17 | 4 | 3 | 2 | 5 | 3 | 2 | 6 | 3 | 2 | 9 | 4 | 1 | 4 | 4 | 3 | 7 | 2 | 2 |
| Asn N | 4 | 4 | 6 | 7 | 2 | 5 | 6 | 4 | 6 | 3 | 2 | 5 | 3 | 2 | 4 | 5 | 4 | 2 | 3 | 3 |
| Asp D | 5 | 4 | 8 | 11 | 1 | 7 | 10 | 5 | 6 | 3 | 2 | 5 | 3 | 1 | 4 | 5 | 5 | 1 | 2 | 3 |
| Cys C | 2 | 1 | 1 | 1 | 52 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 2 | 3 | 2 | 1 | 4 | 2 |
| Gln Q | 3 | 5 | 5 | 6 | 1 | 10 | 7 | 3 | 7 | 2 | 3 | 5 | 3 | 1 | 4 | 3 | 3 | 1 | 2 | 3 |
| Glu E | 5 | 4 | 7 | 11 | 1 | 9 | 12 | 5 | 6 | 3 | 2 | 5 | 3 | 1 | 4 | 5 | 5 | 1 | 2 | 3 |
| Gly G | 12 | 5 | 10 | 10 | 4 | 7 | 9 | 27 | 5 | 5 | 4 | 6 | 5 | 3 | 8 | 11 | 9 | 2 | 3 | 7 |
| His H | 2 | 5 | 5 | 4 | 2 | 7 | 4 | 2 | 15 | 2 | 2 | 3 | 2 | 2 | 3 | 3 | 2 | 2 | 3 | 2 |
| Ile I | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 10 | 6 | 2 | 6 | 5 | 2 | 3 | 4 | 1 | 3 | 9 |
| Leu L | 6 | 4 | 4 | 3 | 2 | 6 | 4 | 3 | 5 | 15 | 34 | 4 | 20 | 13 | 5 | 4 | 6 | 6 | 7 | 13 |
| Lys K | 6 | 18 | 10 | 8 | 2 | 10 | 8 | 5 | 8 | 5 | 4 | 24 | 9 | 2 | 6 | 8 | 8 | 4 | 3 | 5 |
| Met M | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 2 | 3 | 2 | 6 | 2 | 1 | 1 | 1 | 1 | 1 | 2 |
| Phe F | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 3 | 5 | 6 | 1 | 4 | 32 | 1 | 2 | 2 | 4 | 20 | 3 |
| Pro P | 7 | 5 | 5 | 4 | 3 | 5 | 4 | 5 | 5 | 3 | 3 | 4 | 3 | 2 | 20 | 6 | 5 | 1 | 2 | 4 |
| Ser S | 9 | 6 | 8 | 7 | 7 | 6 | 7 | 9 | 6 | 5 | 4 | 7 | 5 | 3 | 9 | 10 | 9 | 4 | 4 | 6 |
| Thr T | 8 | 5 | 6 | 6 | 4 | 5 | 5 | 6 | 4 | 6 | 4 | 6 | 5 | 3 | 6 | 8 | 11 | 2 | 3 | 6 |
| Trp W | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 55 | 1 | 0 |
| Tyr Y | 1 | 1 | 2 | 1 | 3 | 1 | 1 | 1 | 3 | 2 | 2 | 1 | 2 | 15 | 1 | 2 | 2 | 3 | 31 | 2 |
| Val V | 7 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | 4 | 15 | 10 | 4 | 10 | 5 | 5 | 5 | 72 | 4 | 17 |

When one compares two amino acid sequences over an evolutionary distance of 250 PAM, the probability that both sequences contain an Alanine at the same site is 13% (all values are multiplied by 100). The probability that an Alanine has been replaced by an Arginine is 3%, and so on. For the alignment of sequences, one usually uses a **log odds matrix**. The odds score represents the likelihood that the two amino acids will be aligned in alignments of similar proteins divided by the likelihood that they will be aligned by chance in an alignment. In a series of individual matches in an alignment, these odds scores are multiplied to give an overall odds score for the alignment itself. For convenience, odds scores are converted to log odds scores so that the values for amino acid

pairs in an alignment may be summed (instead of multiplied) to obtain the log odds score of the alignment.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cys | C | 12 | | | | | | | | | | | | | | | | | |
| Ser | S | 0 | 2 | | | | | | | | | | | | | | | | |
| Thr | T | -2 | 1 | 3 | | | | | | | | | | | | | | | |
| Pro | P | -3 | 1 | 0 | 6 | | | | | | | | | | | | | | |
| Ala | A | -2 | 1 | 1 | 1 | 2 | | | | | | | | | | | | | |
| Gly | G | -3 | 1 | 0 | -1 | 1 | 5 | | | | | | | | | | | | |
| Asn | N | -4 | 1 | 0 | -1 | 0 | 0 | 2 | | | | | | | | | | | |
| Asp | D | -5 | 0 | 0 | -1 | 0 | 1 | 2 | 4 | | | | | | | | | | |
| Glu | E | -5 | 0 | 0 | -1 | 0 | 0 | 1 | 3 | 4 | | | | | | | | | |
| Gln | Q | -5 | -1 | -1 | 0 | 0 | -1 | 1 | 2 | 2 | 4 | | | | | | | | |
| His | H | -3 | -1 | -1 | 0 | -1 | -2 | 2 | 1 | 1 | 3 | 6 | | | | | | | |
| Arg | R | -4 | 0 | -1 | 0 | -2 | -3 | 0 | -1 | -1 | 1 | 2 | 8 | | | | | | |
| Lys | K | -5 | 0 | 0 | -1 | -1 | -2 | 1 | 0 | 0 | 1 | 0 | 3 | 5 | | | | | |
| Met | M | -5 | -2 | -1 | -2 | -1 | -3 | -2 | -3 | -2 | -1 | -2 | 0 | 0 | 6 | | | | |
| Ile | I | -2 | -1 | 0 | -2 | -1 | -3 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | 2 | 5 | | | |
| Leu | L | -6 | -3 | -2 | -3 | -2 | -4 | -3 | -4 | -3 | -2 | -2 | -3 | -3 | 4 | 2 | 6 | | |
| Val | V | -2 | -1 | 0 | -1 | 0 | -1 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | 2 | 4 | 2 | 4 | |
| Phe | F | -4 | -3 | -3 | -5 | -4 | -5 | -4 | -6 | -5 | -5 | -2 | -4 | -5 | 0 | 1 | 2 | -1 | 9 |
| Tyr | Y | 0 | -3 | -3 | -5 | -3 | -5 | -2 | -4 | -4 | -4 | 0 | -4 | -4 | -2 | -1 | -1 | -2 | 7 |
| Trp | W | -8 | -2 | -5 | -6 | -6 | -7 | -4 | -7 | -7 | -5 | -3 | 2 | -3 | -4 | -5 | -2 | -6 | 0 |
| | | C | S | T | P | A | G | N | D | E | Q | H | R | K | M | I | L | V | F |
| | | Cys | Ser | Thr | Pro | Ala | Gly | Asn | Asp | Glu | Gln | His | Arg | Lys | Met | Ile | Leu | Val | Phe |

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cys | C | 12 | | | | | | | | | | | | | | | | | |
| Ser | S | 0 | 2 | | | | | | | | | | | | | | | | |
| Thr | T | -2 | 1 | 3 | | | | | | | | | | | | | | | |
| Pro | P | -3 | 1 | 0 | 6 | | | | | | | | | | | | | | |
| Ala | A | -2 | 1 | 1 | 1 | 2 | | | | | | | | | | | | | |
| Gly | G | -3 | 1 | 0 | -1 | 1 | 5 | | | | | | | | | | | | |
| Asn | N | -4 | 1 | 0 | -1 | 0 | 0 | 2 | | | | | | | | | | | |
| Asp | D | -5 | 0 | 0 | -1 | 0 | 1 | 2 | 4 | | | | | | | | | | |
| Glu | E | -5 | 0 | 0 | -1 | 0 | 0 | 1 | 3 | 4 | | | | | | | | | |
| Gln | Q | -5 | -1 | -1 | 0 | 0 | -1 | 1 | 2 | 2 | 4 | | | | | | | | |
| His | H | -3 | -1 | -1 | 0 | -1 | -2 | 2 | 1 | 1 | 3 | 6 | | | | | | | |
| Arg | R | -4 | 0 | -1 | 0 | -2 | -3 | 0 | -1 | -1 | 1 | 2 | 8 | | | | | | |
| Lys | K | -5 | 0 | 0 | -1 | -1 | -2 | 1 | 0 | 0 | 1 | 0 | 3 | 5 | | | | | |
| Met | M | -5 | -2 | -1 | -2 | -1 | -3 | -2 | -3 | -2 | -1 | -2 | 0 | 0 | 6 | | | | |
| Ile | I | -2 | -1 | 0 | -2 | -1 | -3 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | 2 | 5 | | | |
| Leu | L | -8 | -3 | -2 | -3 | -2 | -4 | -3 | -4 | -3 | -2 | -2 | -3 | -3 | 4 | 2 | 8 | | |
| Val | V | -2 | -1 | 0 | -1 | 0 | -1 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | 2 | 4 | 2 | 4 | |
| Phe | F | -4 | -3 | -3 | -5 | -4 | -5 | -4 | -6 | -5 | -5 | -2 | -4 | -5 | 0 | 1 | 2 | -1 | 9 |
| Tyr | Y | 0 | -3 | -3 | -5 | -3 | -5 | -2 | -4 | -4 | -4 | 0 | -4 | -4 | -2 | -1 | -1 | -2 | 7 |
| Trp | W | -8 | -2 | -5 | -6 | -6 | -7 | -4 | -7 | -7 | -5 | -3 | 2 | -3 | -4 | -5 | -2 | -6 | 0 |
| | | C | S | T | P | A | G | N | D | E | Q | H | R | K | M | I | L | V | F |
| | | Cys | Ser | Thr | Pro | Ala | Gly | Asn | Asp | Glu | Gln | His | Arg | Lys | Met | Ile | Leu | Val | Phe |

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cys | C | 12 | | | | | | | | | | | | | | | | | |
| Ser | S | 0 | 2 | | | | | | | | | | | | | | | | |
| Thr | T | -2 | 1 | 3 | | | | | | | | | | | | | | | |
| Pro | P | -3 | 1 | 0 | 6 | | | | | | | | | | | | | | |
| Ala | A | -2 | 1 | 1 | 1 | 2 | | | | | | | | | | | | | |
| Gly | G | -3 | 1 | 0 | -1 | 1 | 5 | | | | | | | | | | | | |
| Asn | N | -4 | 1 | 0 | -1 | 0 | 0 | 2 | | | | | | | | | | | |
| Asp | D | -5 | 0 | 0 | -1 | 0 | 1 | 2 | 4 | | | | | | | | | | |
| Glu | E | -5 | 0 | 0 | -1 | 0 | 0 | 1 | 3 | 4 | | | | | | | | | |
| Gln | Q | -5 | -1 | -1 | 0 | 0 | -1 | 1 | 2 | 2 | 4 | | | | | | | | |
| His | H | -3 | -1 | -1 | 0 | -1 | -2 | 2 | 1 | 1 | 3 | 6 | | | | | | | |
| Arg | R | -4 | 0 | -1 | 0 | -2 | -3 | 0 | -1 | -1 | 1 | 2 | 8 | | | | | | |
| Lys | K | -5 | 0 | 0 | -1 | -1 | -2 | 1 | 0 | 0 | 1 | 0 | 3 | 5 | | | | | |
| Met | M | -5 | -2 | -1 | -2 | -1 | -3 | -2 | -3 | -2 | -1 | -2 | 0 | 0 | 6 | | | | |
| Ile | I | -2 | -1 | 0 | -2 | -1 | -3 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | 2 | 5 | | | |
| Leu | L | -8 | -3 | -2 | -3 | -2 | -4 | -3 | -4 | -3 | -2 | -2 | -3 | -3 | 4 | 2 | 8 | | |
| Val | V | -2 | -1 | 0 | -1 | 0 | -1 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | 2 | 4 | 2 | 4 | |
| Phe | F | -4 | -3 | -3 | -5 | -4 | -5 | -4 | -6 | -5 | -5 | -2 | -4 | -5 | 0 | 1 | 2 | -1 | 9 |
| Tyr | Y | 0 | -3 | -3 | -5 | -3 | -5 | -2 | -4 | -4 | -4 | 0 | -4 | -4 | -2 | -1 | -1 | -2 | 7 |
| Trp | W | -8 | -2 | -5 | -6 | -6 | -7 | -4 | -7 | -7 | -5 | -3 | 2 | -3 | -4 | -5 | -2 | -6 | 0 |
| | | C | S | T | P | A | G | N | D | E | Q | H | R | K | M | I | L | V | F |
| | | Cys | Ser | Thr | Pro | Ala | Gly | Asn | Asp | Glu | Gln | His | Arg | Lys | Met | Ile | Leu | Val | Phe |

For example, the log odds value between Phe-Tyr (7) is computed as follows. The Phe-Tyr score in the 250 PAM matrix, 0.15 (or 15 if multiplied by 100), was divided by the frequency of *Phe* in the sequence data, 0.04 (or 4 when multiplied by 100), to give the relative frequency of change. This ratio, 0.15/0.04 = 3.75, was converted to a logarithm to the base 10 (log10 of 3.75 = 0.57) and multiplied by 10 to remove fractional values. Similarly, the Tyr to Phe score is 0.20/0.03 = 6.7, and the logarithm of this number is log10 6.7 = 0.83, and multiplied by 10. The average of 5.7 and 8.3 is 7, the number entered in the log odds table for changes between Phe en Tyr at 250 PAMs of evolutionary distance.

### 3.7.3 Comparison BLOSUM and PAM matrices

In contrast to the BLOSUM matrices the construction of a PAM matrix starts from an underlying model of evolution which is basically a Markov process: each amino acid site in a protein can change at any time to any other amino acid with probabilities given in the PAM matrix and the changes at one particular site occur independent of the amino acids found at other sites (each amino acid change is independent of the other changes and of a previous change at the same site). These assumptions have been challenged. Not all amino acids in a protein necessarily exhibit the same degree of mutability (presence of mutagenesis hot spots). Another criticism on PAM matrices is that they extrapolate the degree of mutagenesis as observed in a selected set of conserved proteins to longer evolution times by simply multiplying the PAM1 matrices (extrapolation of short term changes to long term changes). This assumes that the rate of evolution and mutagenesis remains constant over time. For the construction of a PAM matrix the complete sequence is used.
BLOSUM matrices are based on a much larger family of proteins than the original PAM matrices and for their construction only blocks of sequences that are conserved are considered. For construction of a BLOSUM matrix no explicit evolution model is used, all blocks are assumed to be derived from the same ancestor but all other relationships (more distant closer) are ignored.
BLOSUM matrices are designed to detect conserved domains while PAM matrices are designed to track evolutionary origins of proteins.

**Note: How to choose the best mutation probability matrix?**

In sequence alignment programs and homology search programs, often the possibility exists to choose between different PAM or BLOSUM matrices. Which one to choose is not always obvious. Furthermore, many more substitution probability matrices have been developed recently or being developed as we speak. Therefore, comparison of amino acid scoring matrices poses several problems for the reader to sort out. First, many extremely useful scoring matrices are not based on an explicit model of protein evolution, as are the Dayhoff matrices. Instead, they are designed to locate sequence similarity as a basis for some other common feature between proteins, such as structure, biochemical function, or a family relationship, because they were generated by examining similarity in sequences that share such features in common. For example, Jones et al. (1994) have prepared a scoring matrix specifically for transmembrane proteins. Other matrices include the Gonnet92, and JO93 (Johnson and Overington, 1993) matrices. Recent research (Henikoff and Henikoff 1993; Pearson 1995; 1996; 1998) has shown that, for the majority of sequence analysis projects, these matrices are the most likely to identify a relationship or to find related sequences in a database search, provided that appropriate gap penalties are chosen. At the same time, these matrices are not as useful for discovering evolutionary relationships, because an explicit model of protein evolution was not used in their generation (Altschul et al. 1994). If evolutionary relationships are to be emphasized, the PAM and JTT scoring matrices offer the best analysis, provided that the matrix that corresponds to the evolutionary distance between the sequences and appropriate gap penalties for that matrix are used.

Amino acid scoring matrices are used for two types of analyses. One is to produce alignments and score similarity between two or more protein sequences (see further). A second purpose is to find sequences similar to a test sequence in a database search such as by the BLAST or FASTA programs. For producing sequence alignments, there is no best scoring matrix for every purpose, and the best choice for the investigator is to try several. For the Dayhoff matrices that are based on a particular model of protein evolution, the best alignment is theoretically expected when the correct matrix is used. This matrix should be based on the same evolutionary distance as is apparent between the sequences. The Dayhoff matrix at 250 PAMs of evolutionary distance corresponds to proteins that should be about 20% similar. This level of similarity is within the "twilight zone" of alignment in which most alignments are difficult to prove or to believe. Perhaps in this case, using a Smith-Waterman program will reveal a local area of similarity that is more convincing than a Needleman-Wunsch program. Sequences that are 60% similar are separated by approximately 60 PAMs, 50% similar by 80 PAMs, 40% similar by 120 PAMs, and 30% similar by 160 PAMs of distance. These matrices are widely available and more can be generated by the PAM program (distributed with the BLAST package of programs) to accommodate any distance. As a more appropriate matrix is used, the alignment should improve and be reflected in an increase in the score of the alignment, provided that other factors such as the gap penalty are also adjusted in a reasonable way.

### 3.7.4 Substitution (evolution) models for nucleic acid sequences

Substitution models or models of evolution have also been described for nucleic acid sequences. However, much more than for aligning sequences, these are being used in the construction of phylogenetic trees. Substitution models used for this purpose are discussed in the corresponding chapter.

## 3.8. Heuristic pairwise alignment algorithms

Heuristic pairwise alignments algorithm have been developed to search databases for homologous sequences. They are used for homology searches i.e. instead of aligning 2 sequences a sequence is aligned with each possible sequences in a database. Algorithms that are used for such purpose need to be much faster than the pairwise alignment algorithms based on dynamic programming. Blast and fastA are customarily used homology search algorithms.
When database searches were first attempted, machine size and speed were limiting factors that prevented use of a full alignment program, such as the dynamic programming algorithm (see <a href="Sequence_Alignment.html">here</a>). The number of searches that are presently being performed created a need for faster procedures.; Two methods that are at least 50 to 100 times faster than dynamic programming techniques have been developed.; These methods follow a heuristic approach that works to find related sequences in a database, but that does not have the underlying guarantee of an optimal solution like the dynamic programming algorithm (see e.g. Pertsemlidis and Fondon, 2001).; The first rapid search method that was developed was FASTA, which found short common patterns in the query and database sequences and joined these into an alignment (Wilbur and Lipman, 1983; Lipman and Pearson, 1985; Pearson and Lipman, 1988).; The second method was BLAST (Altschul et al., 1990) and is similar to FASTA but gained a further increase in speed by searching for rarer, but more significantly patterns in nucleic acid and protein sequences; BLAST is extremely popular due to its nice user interface and the NCBI server receives tens of thousands of requests a day.; Both the FASTA and BLAST programs have undergone evolution to recent versions that provide very powerful tools for the moleular biologist and are freely available to run on many computer platforms.

### 3.8.1 Fasta

Rather than comparing individual residues in two sequences, FASTA (Fast Alignment) searches for matching sequence patterns or words, or k-tuples. (Wilbur and Lipman, 1983; Lipman and Pearson, 1985; Pearson and Lipman, 1988). These patterns comprise k consecutive matches in both sequences. The program then attempts to build a local alignment based on these word matches. FASTA uses a lookup table as a rapid way (called hashing) to find common letters and words in the same order and of approximately the same separation in two sequences. For example, assume the following 2 sequences:

sequence 1: ACNGTSCHQE
sequence 2: GCHCLSAGQD

The following lookup table for single amino acids (in this case, k-tuple is 1) gives the common offsets for each shared amino acid in sequence 1 and sequence 2. Note that when an amino acid appears multiple times, all combinations must be considered.

| Amino Acid | Position in Seq 1 | Position in Seq 2 | Offset Value |
|---|---|---|---|
| A | 1 | 7 | 6 |
| C | 2 | 2 | 0 |
| | 2 | 4 | 2 |
| | 7 | 2 | -5 |
| | 7 | 4 | -3 |

| | | | |
|---|---|---|---|
| D | - | 10 | - |
| E | 10 | - | - |
| G | 4 | 1 | -3 |
| | 4 | 8 | -4 |
| H | 8 | 3 | -5 |
| L | - | 5 | - |
| N | 3 | - | - |
| Q | 9 | 9 | 0 |
| S | 6 | 6 | 0 |
| T | 5 | - | - |

The lookup table indicates three common offsets with values 0, 3, and 5. Each common offset indicates a potential alignment, shown below:

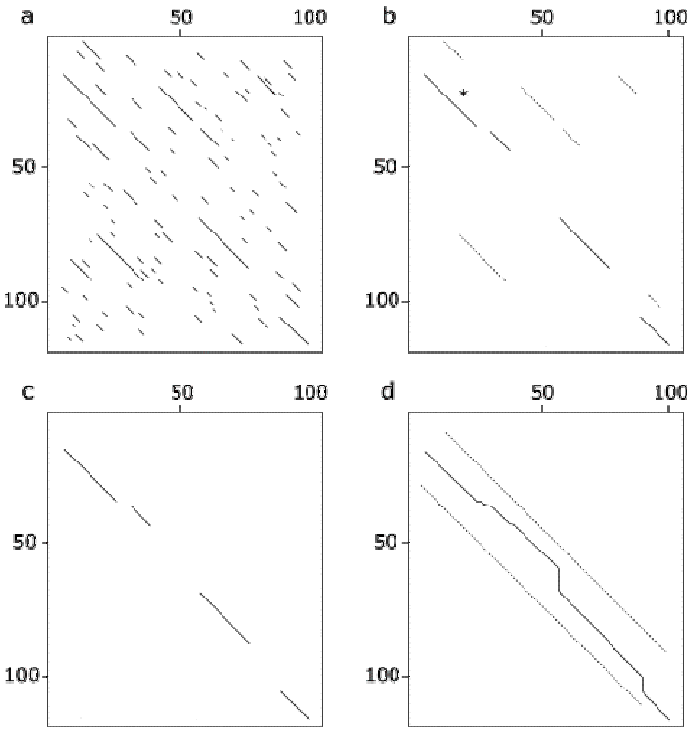
```
sequence 1: ACNGTSCHQE
             C   S  Q       << offset = 0
sequence 2: GCHCLSAGQD

sequence 1: ACNGTSCHQE---
               G  C         << offset = -3
sequence 2: ---GCHCLSAGQD

sequence 1: ACNGTSCHQE-----
                 CH        << offset = -5
sequence 2: -----GCHCLSAGQD
```

The next step in the FASTA algorithm is to join those matches within a certain distance of each other (for proteins, 32 for k=1, and 16 for k=2), along with the region between them (see a, figure below). The regions with the highest density are identified and assigned a score using a scoring matrix chosen by the user (e.g., BLOSUM62 or PAM250).
In conclusion, the FASTA algorithm proceeds as follows (see Figure):

- First, all sets of k consecutive matches are detected (see dot plot). For DNA sequences, k is usually 4-6 and for protein sequences, 1-2.
- Second, the 10 best-matching regions between the query sequence and the sequence in the database are identified. For protein sequences, these 10 highest density regions are identified and evaluated using an amino acid substitution matrix such as PAM or BLOSUM. A corresponding matrix may also be used for DNA sequences.
- Next, an optimal subset of regions is identified that can be combined into one initial, non-overlapping alignment. In a final step, a full local alignment is performed using the Smith-Waterman dynamic programming algorithm.




FASTA thus compares an input DNA or protein sequence to all of the sequences in a target database and reports the best-matched sequences and local alignments of these matched sequences with the input sequence. Hits will be represented by a so-called expect score E. Simply said, the expect score E of a database match is the number of times that an unrelated database sequence would obtain a score higher than the so-called z score (amongst other things dependent on the substitution probability matrix used) just by chance (see box below for more information). For a match to be significant, E should be < 0.01-0.05. If the search has correctly identified homologous sequences, the corresponding E values should be much less than 0.01, whereas scores between unrelated sequences should be much greater than this value; e.g. at least 0.5. If there are no E scores less than 0.1, the search has not found any sequences with significant similarity to the query sequence.

### 3.8.2 Blast

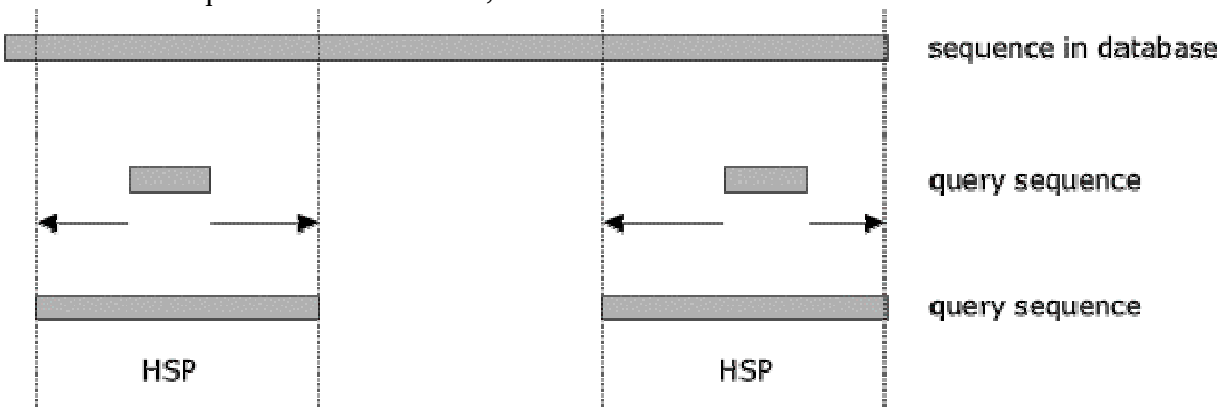
Like FASTA, The BLAST algorithm increases the speed of sequence alignment by searching first for common words or k-tuples in the query sequence and each database sequence. However, whereas FASTA searches for all possible words of the same length, BLAST confines the search to the words that are most significant. For proteins, significance is determined by evaluating these word matches using log odds scores in e.g. the BLOSUM62 amino acid substitution matrix. For the BLAST algorithm, the word length is fixed at 3 for proteins and 11 for nucleic acids. This length is the minimum needed to achieve a word score that is high enough to be significant but not so long as to miss short but significant patterns.

When short similar words (hits) are found, BLAST attempts to extend an alignment from the matching words in each direction along the sequences, continuing for as long as the score continued to increase.

```
L P   P Q G   L L   query sequence
M P   P E G   L L   database sequence
<word>        three letter word found initially
      7 2 6         BLOSUM62 scores, word score = 15
<--           -->   extension to left and right
```

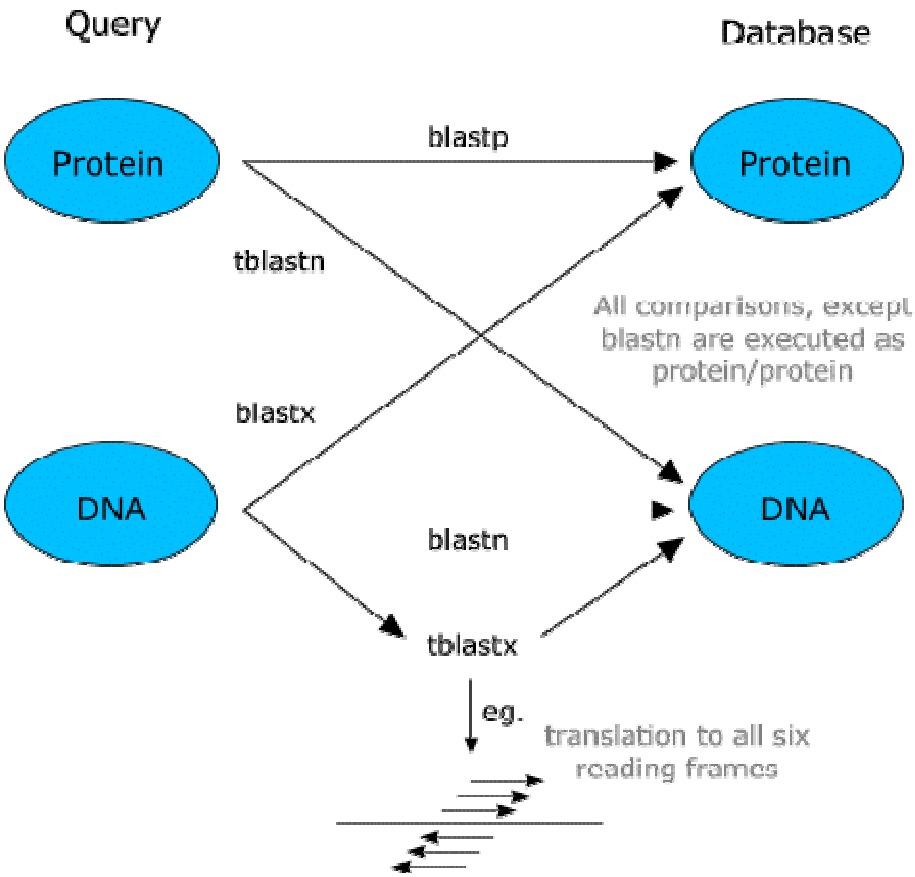
```
2 7   7 2 6   4 4
<      HSP      >   HSP score = 9 + 15 + 8 = 32
```

n this way, one becomes a collection of High-Scoring Segment Pairs (HSPs). A HSP thus consists of 2 sequence fragments of a certain length; one fragment being from the query sequence and the other from a sequence in the database, and a score.




A drawback of the original BLAST program was that gaps were not considered. However, in the improved BLAST2 or gapped BLAST algorithm, 2 'word hits' within a certain distance from one another (thus within a so-called window of x residues) needs to be found, in order to be considered as a HSP. The advantage is that many single random hits can be ignored, resulting in a faster execution of the program. Smith-Waterman local alignments are shown for the query sequence with each of the matched sequences in the database.
Different BLAST programs are available (see Figure below):

blastp: compares an protein (query) sequence with sequences in protein databases (other protein sequences)
blastn: compares a nucleic acid sequence with sequences in nucleic acid databases (other nucleic acid sequences)
blastx: compares all possible translations (according to the six reading frames) of a nucleic acid sequence with sequences in protein databases
tblastn: compares a protein sequence with all possible translations (according to the six reading frames) of sequences in nucleic acid databases
tblastx: compares all possible translations of a nucleic acid sequence (according to the six reading frames) with all possible six reading frame translations of sequences in nucleic acid databases




## 3.9. Statistical methods to score the significance of pairwise alignments

Assessing the statistical relevance of an alignment is of extreme importance when comparing two sequences that show only a restricted degree of similarity or when searching similarities between a sequence and large database in order to distinguish statistically significant alignments from spurious hits.

Originally the significance of sequence alignment scores was evaluated on the basis of the assumptions that alignment scores followed a normal statistical distribution but later more careful analysis showed that this was not true but in fact sequence scores follow an extreme value distribution. The statistical analysis of alignment scores is much better understood for local than for global alignments.

The score system used to produce a local alignment is designed to reveal regions of closely matching sequence with a positive alignment score. In random sequences or unrelated sequences these regions are rarely found. Hence their presence in real sequence alignments is significant and the probability of their occurring by chance alignment of unrelated sequences can be readily calculated.

The significance is represented by the p-value. The p-value expresses how likely it would be to observe a certain score assuming that the H) (zero hypothesis) is valid. The zero hypothesis in this context states that 2 sequences are random i.e. unrelated. A high alignment score thus will be very unlikely to observe under the H0 and will have therefore a low p-value.

The problem is to estimate H0. Usually this is done by shuffling the sequences to randomise them, by randomly generating them… as such it was detected that alignment scores in random sequences are distributed according to the extreme value distribution. The alignment score and the extreme value distribution is thus used to calculate a p-value.

When searching a database, however, the significance does not only depend on how well two sequences align but also on the size of the database that was searched. The larger the database the more chance a spurious hit will occur. The e-value compensates for this.

**E(score > S) = database size * p-value**

Assessing the statistical relevance of a global alignment as obtained by a Needle Man Wunsh algorithm is not trivial. Because the average match of an amino acid or nucleotide is assessed by a positive score, the score of any alignment (also a random alignment) grows proportionally to the length of the sequences. Moreover there are different possible global alignments and small changes in the parameter settings can produce a very different alignment. There is no theory to assess significance of global alignments similar to the one for local alignments. Therefore usually a global alignment is used if sequences are already known to be related. The alignment is subsequently used for eg phylogenetic inference.

### 3.9.1 Bayesian score (niet kennen)

$$P(M|x,y)$$

–Posterior: probability that sequences are related as opposed to being unrelated

–Likelihood calculated:

$$P(x,y|M)$$

–Prior probability:

$$P(M)$$

It can be shown that

The posterior probability that the match model is correct and that the sequences are related is the logistic function of S'. (S = logg odds score of the alignment and p(M)/p(R) the ratio of the prior probabilities.

$$P(M|x,y) = \frac{P(x,y|M)P(M)}{P(x,y)} \quad \frac{P(x,y|M)P(M)}{P(x,y|M)P(M)+P(x,y|R)P(R)}$$

$$P(M|x,y) = \sigma(S')$$

$$S' = S + \log\left(\frac{P(M)}{P(R)}\right)$$

$$S = \log\left(\frac{P(x,y|M)}{P(x,y|R)}\right)$$

$$\sigma(x) = \frac{e^x}{1+e^x}$$

### 3.9.2 Extreme value distribution (niet kennen)

More classical statistical framework to score an alignment. Use of the bit scores helps to assess the statistical relevance of the alignment.

## 3.10. Recent developments

With the recent sequencing of complete genomes, there are substantial practical problems to deal with. The size of the genomes, which range from 100 Mb to 3 Gb, means that the basic algorithmical approach (comparing every base in one genome to every base in another using a rigorous mathematical framework) is impractical.
In the past 5 years, an increasing number of programs have been produced that aim to align large genomic sequences from different organisms. Each of these attempts, in different ways, to address the problems of speed and accuracy for whole-genome alignments. Because this field is rapidly evolving and productive, the explosion in the availability of tools might confuse end-user biologists and make it difficult for them to decide which tool is most suitable for their research. Indeed, to interpret these resources, biologists must be aware of the differences between them and between their underlying assumptions.

The naive approach for calculating an alignment of two sequences is to create a matrix using a recursive calculation called dynamic programming (DP). For short DNA sequences and protein sequences this is feasible and relies on two well-known algorithms: Needleman and Wunsch (NW)27 and Smith-Waterman (SW)26, both of which have two main limitations. First, they do not properly handle alignments of rearranged sequences (such as inversions, duplications and translocations). Second, and most important, if the two sequences are >10,000 bp, calculating alignments by these methods becomes extremely expensive both in terms of time and memory usage, and is not feasible. To gain speed, all fast alignment programs rely on some sort of heuristic (or seeding) strategy to reduce the computational complexity. A seeding strategy is based on the idea that correct alignments will have significant stretches of ungapped matches; by first finding a set of significant ungapped matches, the algorithm can avoid large amounts of computation. The loss of sensitivity of such heuristics is tolerated, as it gives a tremendous gain in speed and a reduction in memory usage.

Generally, all algorithms go through three main steps, although how these steps are executed differs depending on the algorithm. In step one, SEEDS are found using the seeding strategy that is particular to a given algorithm. In step two, the seeds can be used as nucleation points for extension or as anchor points to partition the alignment problem. In step three, a DP process is used to produce gapped alignments. Here, we concentrate on one such whole-genome alignment approach: AVID and LAGAN. Results of AVID and LAGAN can be visualized by VISTA.

AVID is a global aligner, which assumes that the two sequences that are to be aligned are strictly homologous and have not undergone changes such as gene duplication, inversion or translocation, during evolution — in other words, it assumes that the order of the functionally conserved region has been preserved. This assumption is incorrect when applied directly to a whole genome; so, to achieve a whole-genome alignment, AVID needs a pre-processing step to identify the putative homologous unrearranged sequences or SYNTENIC REGIONS.

Like AVID, LAGAN is a global aligner that uses the same three-step strategy, with recursion of steps one and two. The advantages of LAGAN over AVID are that it can align larger sequences (>2 Mb), owing to the considerably lower memory requirements, and that it uses a different matching algorithm in step one — AVID looks for exact matches, limiting the comparisons to closely related organisms, whereas LAGAN was designed to align both distantly and closely related organisms.

For individual genomic regions, 'adventurous' biologists might do their own comparisons and interpretation. This is generally advisable, because the large-scale genome comparisons use algorithms that are necessarily less sensitive than those that search for the optimal alignment and, perhaps more importantly, because it is only by engaging with the data and the computational methods that it is possible to have a feel for them.

AVID....http://www-gsd.lbl.gov/vista/details_avid.htm

LAGAN/MLAGAN....http://lagan.stanford.edu