Caution: Two protein sequences being k PAM-units apart do not necessarily differ at k percent of their positions as multiple mutations may occur at the same position.

k-PAM-Matrix: Scoring matrix suitable to compare protein sequences being k PAM-units apart.

Question: How can such matrices be determined?

Problem: The PAM-distance can not be measured exactly in practice.

Ideal case:

- We know many pairs of homologous (two protein sequences are called homologous if the corresponding proteins have the same function, e.g. in different organisms) protein sequences,
- which we know to be k PAM-units apart.

We furthermore assume that we know the positions of gaps in the optimal alignment of each pair.
We call the set of these alignments A and the multiset of columns in A with no gap symbol $\mathcal{C}(A)$.

$$freq(a_i, a_j) := \frac{\#\text{ occurences of } (a_i, a_j) \text{ and } (a_j, a_i) \text{ in } \mathcal{C}(A)}{2 \cdot |\mathcal{C}(A)|}$$

Intuition: Relative frequency of a_i pairing with a_j in a column in A ignoring order.

$$freq(a_i) := \frac{\#\text{ occurences of } a_i \text{ in all alignments}}{\text{total length of sequences}},$$

$freq(a_i)$ thus is the relative frequency of a_i in all alignments.

Now we set entry (i, j) of the k-PAM-matrix to

$$PAM_{i,j}^{(k)} := \log\left(\frac{freq(a_i, a_j)}{freq(a_i) \cdot freq(a_j)}\right).$$

Note: Matrix is symmetric.

Intuition: Entry (i,j) of the matrix describes the relation of the probability of replacing symbol a_i with symbol a_j by accepted mutations to the probability of the pair appearing by chance.

Logarithm: Use sums instead of products.

Practice: multiply by 10 and round to nearest integer.

Problem: There is no data available in practice that allows to count the number of substitutions.

Idea: Choose a set of very similar sequences with common ancestor for which the assumption of being only one PAM-unit apart is reasonable.
For these sequences determine the positions of gaps in the alignments (here we are looking for the *evolutionary truth* that can't be determined algorithmically; these alignments are done by hand by experts, which is possible due to the great similarity of the sequences) and determine 1-PAM-matrix as described for the idealized case.

To determine k-PAM-matrices for $k > 1$ we assume F to be the 20×20 matrix, whose entry (i,j) describes the probability of a_i to mutate into a_j in one PAM-unit (independent of the occurrence frequency of a_i).
The entries of this matrix can too be read off the pairwise alignments of the protein sequences. F^k then describes the probability of substitution in k PAM-units for each pair of symbols. Thus we set

$$PAM_{i,j}^{(k)} := \log\left(\frac{freq(a_i) \cdot F_{i,j}^k}{freq(a_i) \cdot freq(a_j)}\right) = \log\left(\frac{F_{i,j}^k}{freq(a_j)}\right).$$

Application: As the PAM-distance of two given protein sequences is unknown several standard values (e.g. $k = 40$, $k = 100$ and $k = 250$) are considered.

Disadvantage of PAM-matrices: Base of computation are sequences closely related.
$\Rightarrow$ Extrapolation by F^k for k-PAM-matrices does not work for k too large, thus examination of proteins further distant is impossible.

Alternative: BLOSUM
Here relative frequencies are read off a database with informations on similar regions in amino acid sequeneces of related proteins. Evolutionary far distant sequences are used too to avoid the PAM-matrices' problem.

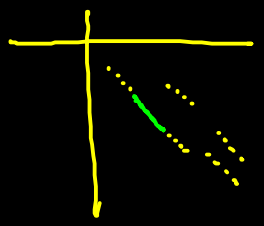
No new insights from a computer science point of view.

Heuristic methods

In practice databases of DNA and protein sequences are used that are so large that even the aforementioned polynomial time algorithms are too slow for convenient use.
For this reason faster heuristic methodas are used that do however not guarantee to find an optimal alignment.

FASTA (Abbr. for *fast-all*)
FASTA compares the searched pattern P with all sequences T stored in the database one after another. This happens in 4 steps:
1.) For a chosen Parameter k all exact matches of substrings of length k from P and equally long substrings of T are determined. These matches are called *hot spots*. Usually $k = 6$ for DNA and $k = 2$ for protein sequences.
As k is chosen so small instead of using a string matching algorithm all substrings of P and T of length k are hashed into a common hash table. A collision between a substring of P and one of T then is very likely a match.

2.) In this step it is tried to group several hot spots together.
To do this a matrix D similar to the alignment matrix M is created. Its rows correspond to symbols of P, its columns to symbols of T.
$D_{i,j}$ is set to 1 if $P_i = T_j$, 0 otherwise. Thus each hot spot corresponds to a diagonal run of ones in the matrix. Afterwards scores for all parts of diagonals beginning and ending with a hot spot are computed. So to say, it is tried to *fusion* hot spots. Each hot spot gets a positive rating, gaps are rated negative. The longer a gap is, the more it is penelised.
In the end the ten best rated diagonals are selected. They correspond to ten differend alignments of substrings of P and T containing matches and substitutions but no gaps.
Now for all these pairs of substrings an optimal local alignment is determined, usually using one of the scoring matrices mentioned above. The partial alignment with the best score is called $init1$.



3.) Here we consider those partial alignments from step 2 which are scored higher than a previously fixed threshold.
It is tried to connect these alignments to a longer and better alignment.
This is done by creating a digraph with vertices corresponding to one alignment each and labelled with the score of the corresponding alignment.
Let u such a vertex with the corresponding alignment ending at position (i, j) in D and v a vertex with the corresponding alignment starting at (i', j'). An edge from u to v is added if and only if $i < i'$ and $j < j'$ hold as this means the alignments do **not** overlap.
The edge is labeled by a negative number which depends on the distance of (i, j) and (i', j') in D (thereby penalising gaps).
An optimal alignment made up of the partial alignmments then corresponds to a path with maximal sum of vertex and edge scores.
The alignment determined thusly is one output of the heuristic.

4.) In this step an alternative solution is computed. Starting with $init1$ an exact solution for an optimal alignment is computed with matrix M restricted to a not too wide strip around $init1$.

These 4 steps are done for all words T in the database. Afterwards statistical methods are used to judge the significance of the solutions (more details on this follow later). The goal is to separate random matches from those not random.

BLAST

At the moment BLAST is the dominating heuristic. Like FASTA BLAST exploits the fact that best alignments often contain almost exactly matching substrings. These substrings are used as *seed* for an alignment.
BLAST however does not use hashing to find exact matching substrings but local alignments without gaps.
The algorithm takes the following steps:

1.) For each subword α of P of length w (typically $w = 11$ for DNA and $w = 3$ for protein sequences) determine all words α' of length w such that a gapless local alignment of α and α' has scoring at least t (threshold) $\rightsquigarrow A$. Then, by exact algorithms (e.g. Aho Corasick), search all occurrences of $\alpha' \in A$ as a substring of $T \rightsquigarrow hits$

2.) In this step all pairs of hits no more than d characters apart are searched. Hits that are not part of a pair are discarded.
d is chosen dependent of w. In case of protein sequences $d = 16$ is a typical value.

3.) Now it is tried to elongate pairs of hits (i.e. parts of diagonals from an alignment matrix) by adding additional alignment columns on both sides until the score no longer increases. Newer versions of BLAST allow for the addition of gaps in this phase.
If the alignments created this way score higher than a threshold S they are called *high scoring pair*. The set of these pairs ordered by score is the output of BLAST.

Additionally BLAST determines the *bit score* for the results which is independent of the scoring matrix (details follow). This allows for the comparison of results gained with different scoring matrices.
As in FASTA statistic significance of the results is estimated.

Significance of local alignments

We assume two DNA or protein sequences $X \in \Sigma^n$ and $Y \in \Sigma^m$ to be aligned using a given scoring matrix with the resulting local alignment having a score of b. Is this result of any biological significance or just by chance?
We furthermore assume that (as in the first version of BLAST) only gapless alignments are feasible and we follow the classic principle of statistics:
We try to show that assuming both sequences are independent scores of b or higher are extremely unlikely.

Model: We assume the symbols of Σ to be numbered 1 to $|\Sigma|$. Let p_i denote the probability of $a_i \in \Sigma$. Furthermore we assume all positions in $Z \in \Sigma^n$ to be stochastically independent so the probability of $Z = a_{i_1} a_{i_2} \cdots a_{i_n}$ is given by $p_{i_1} \cdot p_{i_2} \cdots p_{i_n}$.

Null hypothesis: The null hypothesis is: The sequences $X \in \Sigma^n$ and $Y \in \Sigma^m$ are independent, meaning

$$\Pr[X_k = a_i, Y_l = a_j] = p_i \cdot p_j, \text{for all } k \leq n,\ l \leq m,\ a_i, a_j \in \Sigma.$$

Our target is to discard the hypothesis.

Alternative: There is an alignment of X and Y reflecting evolution. For respective pairs of positions (k, l)

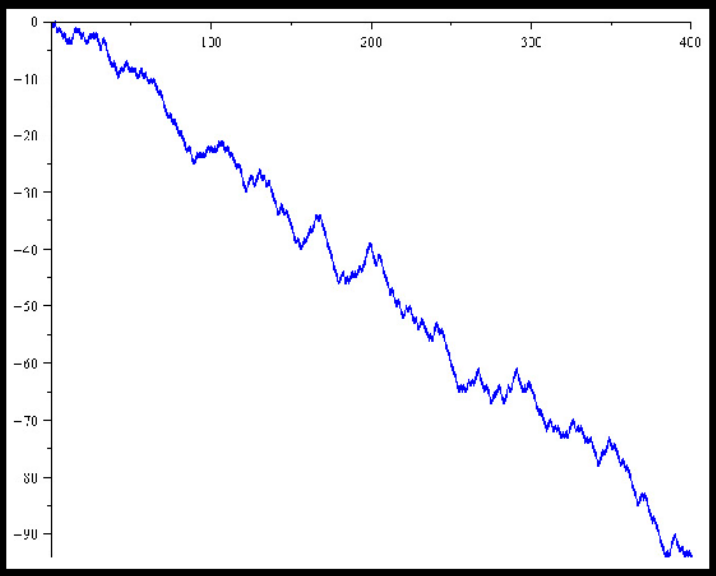
$$\Pr[X_k = a_i, Y_l = a_j] = q(i,j) \neq p_i \cdot p_j$$

holds.
If we manage to discard the null hypothesis we want to believe in this alternative.

Measure of significance: e-values being (roughly speaking) the expected number of partial alignments with a scoring $\geq b$.

Starting to align gapless (along a diagonal) at a position (i, j) of matrix M randomly picked the probability of pair $\begin{smallmatrix} a_k \\ a_l \end{smallmatrix}$ being next and contributing $\delta(a_k, a_l)$ to the total score of the alignment is $p_k \cdot p_l$.
Thus all increments of the score are stochastically independent and identically distributed. Such a process with independent identically distributed increases is called a random walk.

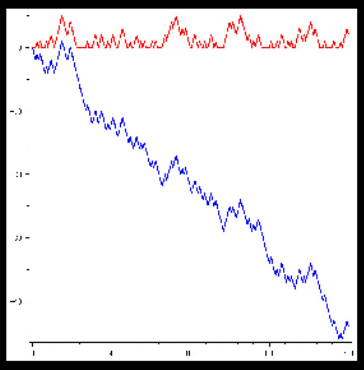
Example: We only distinguish match and substitution, ignoring the symbols involved. Matches score 1, substitutions score -1. Then in each step the random walk goes up 1 with probability $p := \sum_{a_i \in \Sigma} p_i^2$ and down 1 with probability $1 - p$.



Ladder points: All the places where the value is lower than ever before.

Excursion: Part between two ladder points (Often two ladder points will be immediate neighbors. Then the corresponding excursion has length 0.).

Transformation of the random walk: On each ladder point reset the process to zero. (Thus the name *excursion*).
$\Rightarrow$ Excursion are transformed into paths above the x-axis always returning to their starting point (level wrt. the y-axis).



Correspondence to alignment:

Random walk: Describes the change of rating of a gapless alignment along a diagonal of M.

Excursion: (modified random walk) Here each point gives the highest rating possible for a gapless local alignment along the diagonal.
If we reach a point which would get a negative rating (ladder point) an elongation of the alignment would be pointless as the prefix gives a negativ contribution
$\Rightarrow$ start with new substring and reset rating to zero.

$\Rightarrow$ Parts starting in a ladder point and ending in a maximum of the respective excursion correspond to local alignments that can not be elongated in any direction without lowering the score.

$\Rightarrow$ high scoring pair from BLAST.

Thus the probability of a high scoring pair reaching score b is equal to the probability of an excursion reaching height b.

We consider the simple case of $\delta(x,x) = 1$, $\delta(x,y) = -1$ for $x \neq y$: Let ... We denote prob. that random walk starting at level k reaches level b before reaching level a.

$\rightsquigarrow W_b = 1, \quad W_a = 0$

$$W_k = p \cdot W_{k+1} + (1-p) \cdot W_{k-1}, \quad a < k < b$$

$$\Longrightarrow W_{k+1} - W_k = \frac{1-p}{p}(W_k - W_{k-1})$$

Observation: W_k changes by a constant factor in each step $\rightsquigarrow$ exponential behaviour.

Ansatz: $W_k = K \cdot e^{\lambda k} + c$

$$\Longrightarrow K \cdot e^{\lambda n} + c = p \cdot \left(K \cdot e^{\lambda(n+1)} + c\right) + (1-p)\left(K \cdot e^{\lambda(n-1)} + c\right)$$

$$\curvearrowright \quad 1 = p \cdot e^{\lambda} + (1-p) \cdot e^{-\lambda}$$

Setting $x := e^{\lambda}$ we find

$$0 = p \cdot x^2 - x - p + 1$$

Solutions: 1) $x = 1 \quad \hat{=} \quad \lambda = 0$

2) $x = \frac{1-p}{p} \quad \hat{=} \quad \lambda = \ln\left(\frac{1-p}{p}\right)$

If a homogeneous difference-equation has two different solutions then we have to consider their linear combination:

$$W_n = c_1 \cdot 1 + c_2 \cdot e^{\lambda \cdot n}, \quad \underline{\lambda = \ln\left(\frac{1-p}{p}\right)}$$

Determine c_1 and c_2 from initial cond.

$$\left.\begin{aligned} W_a = 0 &= c_1 + c_2 \cdot e^{\lambda a} \\ W_b = 1 &= c_1 + c_2 \cdot e^{\lambda b} \end{aligned}\right\} \quad \begin{aligned} c_2 &= e^{\lambda a}/(e^{\lambda a} - e^{\lambda b}) \\ c_2 &= 1/(e^{\lambda b} - e^{\lambda a}) \end{aligned}$$

$$\Longrightarrow \text{Solution: } W_n = \frac{e^{\lambda n} - e^{\lambda a}}{e^{\lambda b} - e^{\lambda a}}$$

Now consider $n = 0$, $a = -1$

$$\rightsquigarrow W_0 = \frac{e^0 - e^{-\lambda}}{e^{\lambda b} - e^{-\lambda}} \sim (1 - e^{-\lambda}) e^{-\lambda b}, \quad b \to \infty$$

Consequence: P.d. for a hsp with score $\geq b$ to start at position (i,j) is given by

$$W_0 \cdot P_1[(i,j) \text{ is ladder point}]$$

$\rightsquigarrow$ solution of pattern $C \cdot e^{-\lambda b}$

E-value: $\mathbb{E}$ number of hsp with score $\geq b$

$\rightsquigarrow \sum$ over all positions (i,j)

Sequences of length m and n resp.

$\rightsquigarrow$ $m \cdot n \cdot C \cdot e^{-\lambda b}$

Remark: Nearly the same result holds for any scoring schema of practical importance; only C and λ must be adapted (Dembo, Karlin and Zeitouni 1994).

Consequences: The smaller the e-value for a local alignment the larger its significance (e-value tells us how likely score b found for our input results from aligning two random sequences).

Question: How to compare alignments computed with different scoring schemes?

Use the normalized scoring B' with

$$B' := \frac{\lambda B - \ln(C)}{\ln(2)}, \text{ i.e. } B = \frac{B' \cdot \ln(2) + \ln(C)}{\lambda}.$$

Then the e-value with respect to B' (called *bit-scores*) becomes

$$E_{B' \geq b} = E_{B \geq \frac{b \cdot \ln(2) + \ln(C)}{\lambda}} \sim m \cdot n \cdot 2^{-b},$$

i.e. asymptotically independent of C and λ.

$\rightsquigarrow$ bit-scores much more appropriate to compare different alignments.

BLAST

1. $A := \{\alpha' \mid \alpha' \approx_t \alpha \wedge \alpha \text{ subword of } P \wedge |\alpha| = w\}$
   Search for all elements of A in T (Aho Corasick) $\rightsquigarrow$ hits.
2. Determine all pairs of hits no more than d characters apart:

$$\left.\begin{array}{l}\text{first hit} = \text{string similar to subword of } P \\ \text{second hit} = \text{subword of } T\end{array}\right\} = \begin{array}{l}\text{local alignment} \\ \text{of } P \text{ and } T\end{array}$$

3. Pairs of hits are elongated until score increases no longer.
   Score $> S \rightsquigarrow$ high scoring pair (hsp).

Multiple alignments

Motivation: Alignment of two words only of interest if random events can be ruled out as source of similarities. (Inference of structural or functional relation).
Similarity of two sequences often too small to be detected.
Hope: Common functional parts of multiple sequences get visible by amplifying effects of a multiple alignment.

Definition

Given k words $S_i \in \Sigma^\star$, $1 \leq i \leq k$, gap symbol $- \notin \Sigma$ and $\Sigma' := \Sigma \cup \{-\}$. Furthermore $h : (\Sigma')^\star \to \Sigma^\star$ the homomorphism induced by $h(a) = a$ for all $a \in \Sigma$ and $h(-) = \varepsilon$.
A multiple alignment of $S_1, \ldots S_k$ is a tuple $(S^{(1)}, \ldots, S^{(k)})$ of words over $(\Sigma')^l$, $l \geq \max\{|S_i| \mid 1 \leq i \leq k\}$, satisfying:

1. $h(S^{(i)}) = S_i$, $1 \leq i \leq k$;
2. $(\nexists j \in [1 : l])(\forall i \in [1 : k] : S^{(i)}_j = -)$.

We call l the length of the multiple alignment.

How to rate such an alignment?

First possibility: *consensus*

Definition

Given a multiple alignment $\mathcal{A} = (S^{(1)}, \ldots, S^{(k)})$ *of words* $S_1, \ldots,$ $S_k \in \Sigma^\star$ *and length* l.
A word $\mathcal{C} \in \Sigma^l$ *is called* consensus *of* $\mathcal{A}$, *if*

$$\mathcal{C}_j = \operatorname{argmax}_{a \in \Sigma} |\{S_j^{(i)} = a \mid 1 \leq i \leq k\}| \text{ for all } 1 \leq j \leq l$$

holds. The distance $dist(\mathcal{C}, \mathcal{A})$ from $\mathcal{A}$ to $\mathcal{C}$ *is defined by*

$$dist(\mathcal{C}, \mathcal{A}) := \sum_{1 \leq j \leq l} |\{S_j^{(i)} \mid 1 \leq i \leq k \wedge S_j^{(i)} \neq \mathcal{C}_j\}|.$$

Note 1: This definition implicitly uses edit distance. A symbol in column j contributes $\delta(S_j^{(i)}, \mathcal{C}_j) = 1$, if the symbols differ, 0 if they are the same.

Consensus: Consists of symbols $\mathcal{C}_j$, minimising $\sum_{1 \leq i \leq k} \delta(\mathcal{C}_j, S_j^{(i)})$.

More general: Allow *arbitrary* δ.