Bioinformatics

Lecture 2

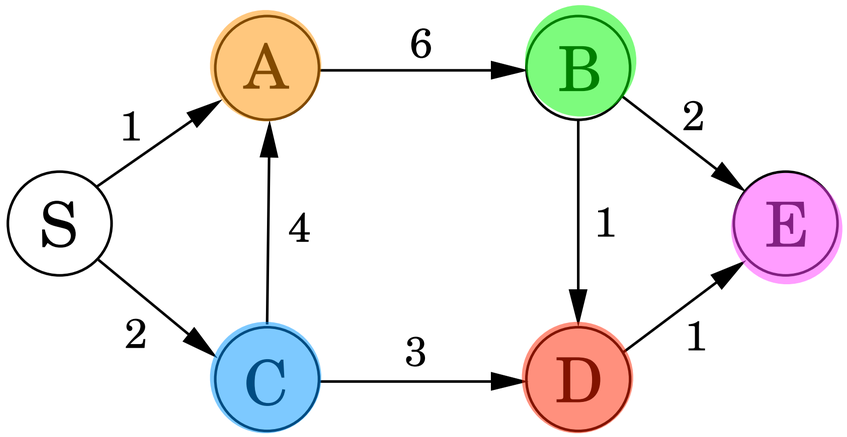
Weidong Tian, School of Life Sciences, Fudan University

Sept 12, 2024

Outline

- Construction of substitution matrices
- Multiple sequence alignments
- Database search

Dynamic Programming



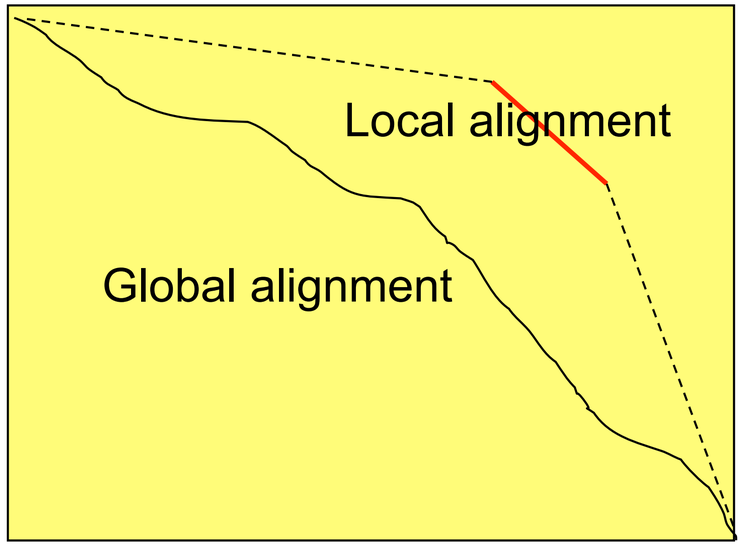
so to find the shortest path to D, we need only compare these two routes:

$$\texttt{dist}(D) = \min\{\texttt{dist}(B) + 1, \texttt{dist}(C) + 3\}.$$

A similar relation can be written for every node.

```
initialize all dist(·) values to ∞
dist(s) = 0
for each v ∈ V\{s}, in linearized order:
    dist(v) = min_{(u,v)∈E}{dist(u) + l(u, v)}
```

pairwise sequence alignment



$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \\ s_{i-1,j-1} + \delta(v_i, w_j) \end{cases}$$

$$s_{i,j} = \max \begin{cases} 0 \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \\ s_{i-1,j-1} + \delta(v_i, w_j) \end{cases}$$

- Appropriate gap penalties
- Appropriate substitution matrices
- Complexity in computation

Given a scoring system, dynamic programming guarantees to produce the optimal global or local sequence alignment.

Nucleotide substitution scoring matrices

The default nucleotide substitution scoring matrix for BLASTN search.

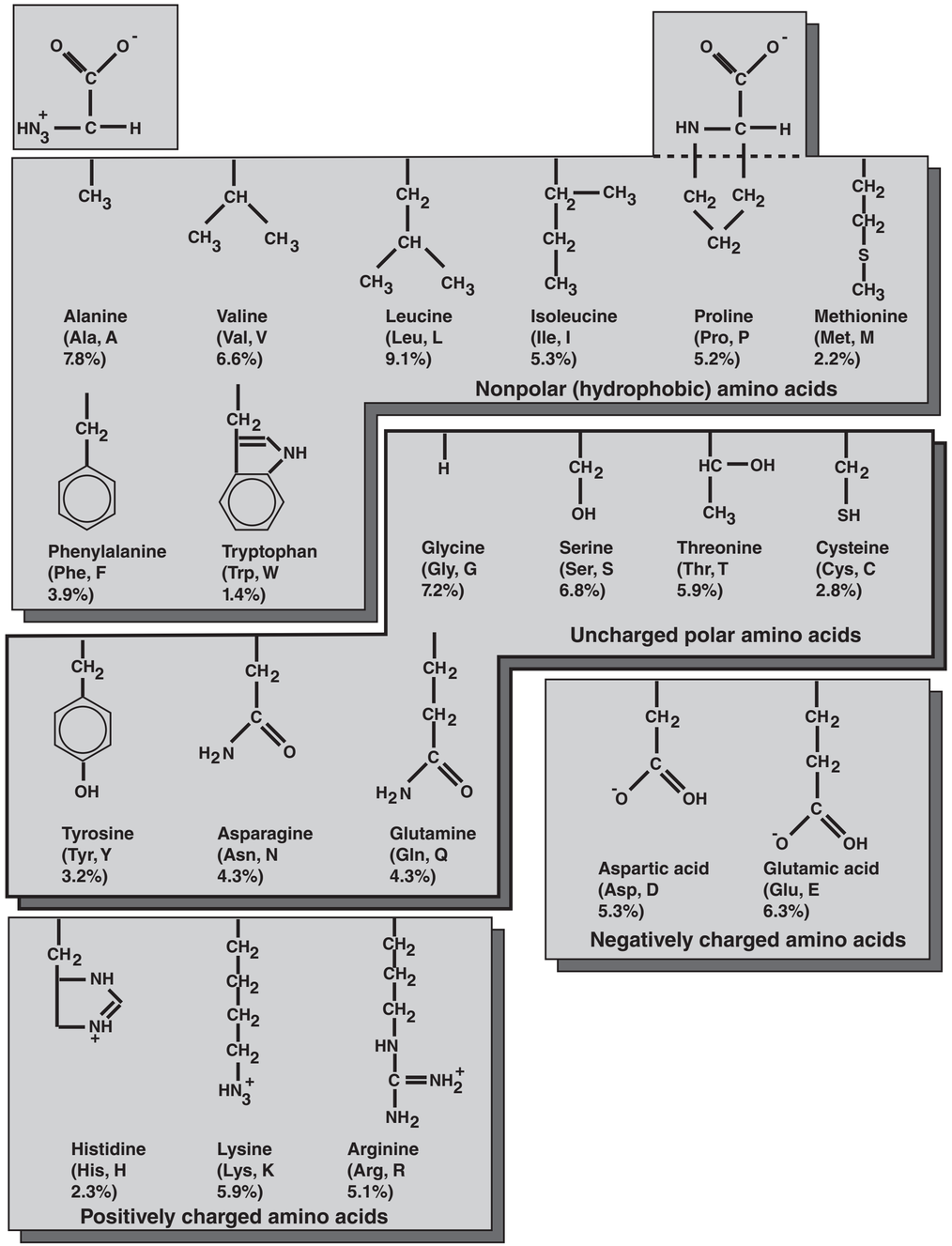
	A	C	G	T
A	$+2$	-3	-3	-3
C	-3	$+2$	-3	-3
G	-3	-3	$+2$	-3
T	-3	-3	-3	$+2$

Amino acids substitution scoring matrices

The default amino acids substitution scoring matrix for BLASTP search.

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

BOX 3.2 STRUCTURES AND ONE- AND THREE-LETTER ABBREVIATIONS OF 20 COMMON AMINO ACIDS



It is very helpful to memorize these abbreviations and to become familiar with the physical properties of the amino acids. The percent

Substitution scoring matrices

- A substitution matrix is a collection of scores for aligning nucleotides or amino acids with one another.

- These scores generally represent the relative ease with which one nucleotide or amino acid may mutate into or substitute for another, and they are used to measure similarity in sequence alignments.

    - **Positive scores**: Represent substitutions that are favorable or likely, indicating a high probability that these substitutions could occur through evolutionary processes without significantly disrupting function.

    - **Negative scores**: Represent less favorable or unlikely substitutions, where changes could impair function or are less probable from an evolutionary perspective.

- There are numerous amino acid or nucleotides substitution scoring matrices, each developed to address different evolutionary distances, sequence types, and alignment needs.

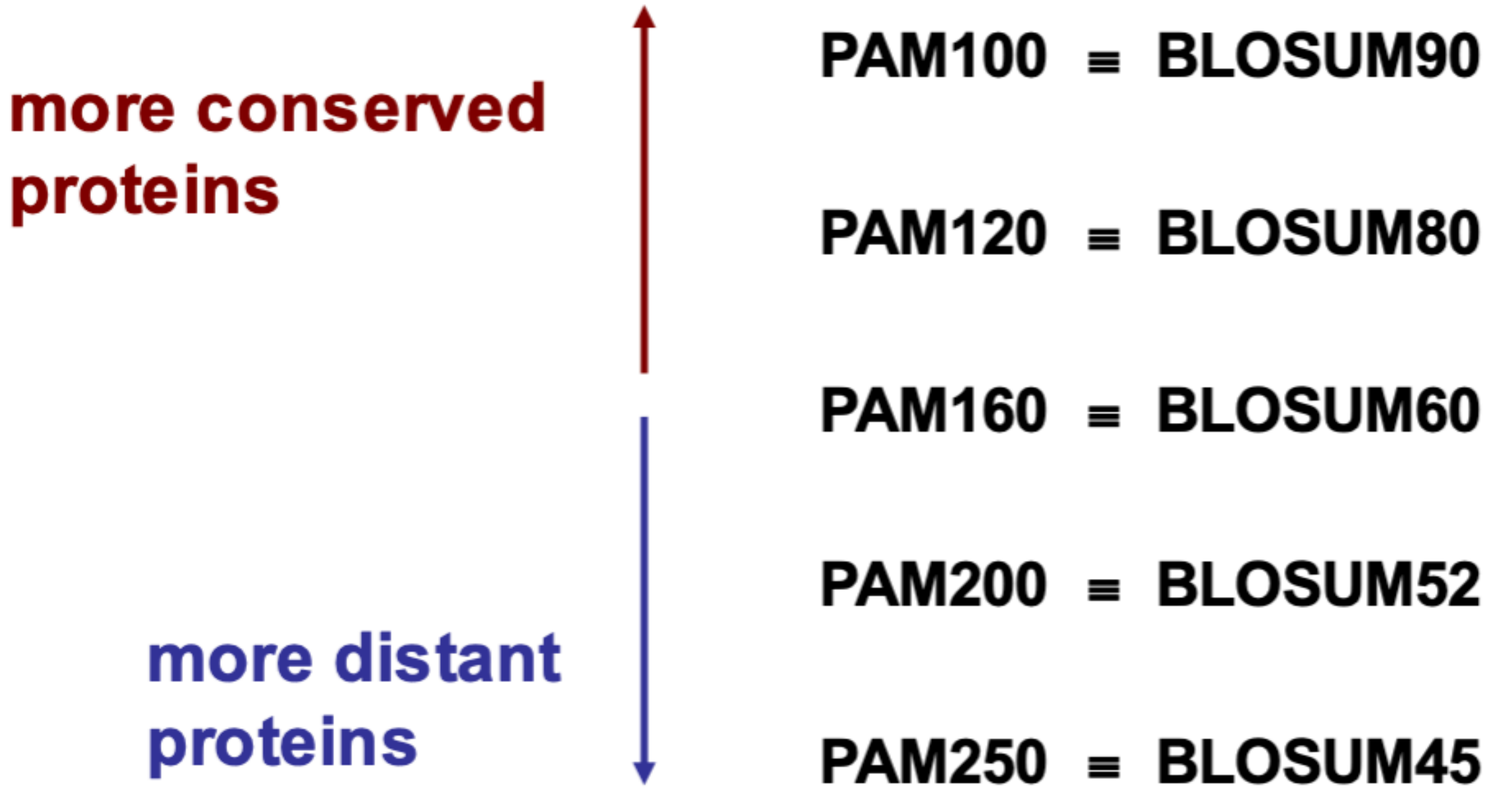
- Substitution scoring matrices are vital because they provide a structured, quantitative way to compare biological sequences, offering insights into evolution, function, and structure.

How to construct and choose the appropriate substitution matrices?

Two commonly used amino acids substitution scoring matrix families

- PAM (Point Accepted Mutation) Matrices: A series of matrices developed based on evolutionary models.
    - The PAM matrices are numbered according to evolutionary distance, with **PAM1** representing very closely related sequences and higher numbers (e.g., **PAM250**) representing more distantly related sequences.
    - A **PAM unit** is based on the idea of **one point accepted mutation per 100 amino acids**, meaning that after 1 PAM unit of evolutionary time, 1% of the amino acids in a sequence will have changed due to accepted mutations.
    - **Point Accepted Mutation** refers to an amino acid substitution that has been accepted by natural selection during evolution.
- BLOSUM (Blocks Substitution Matrix) Matrices: Derived from observed substitutions in conserved regions of protein families.
    - BLOSUM matrices are numbered according to the degree of sequence similarity within the block used to generate the matrix, with **BLOSUM62** being the most widely used.

PAM vs BLOSUM matrices



PAM250 scoring matrix

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	2	−2	0	0	−2	0	0	1	−1	−1	−2	−1	−1	−3	1	1	1	−6	−3	0
R	−2	6	0	−1	−4	1	−1	−3	2	−2	−3	3	0	−4	0	−1	−1	2	−4	−2
N	0	0	2	2	−4	1	1	0	2	−2	−3	1	−2	−3	0	1	0	−4	−2	−2
D	0	−1	2	4	−5	2	3	1	1	−2	−4	0	−3	−5	−1	0	0	−7	−4	−2
C	−2	−4	−4	−5	12	−5	−5	−3	−3	−2	−6	−5	−5	−4	−3	0	−2	−8	0	−2
Q	0	1	1	2	−5	4	2	−1	3	−2	−2	1	−1	−5	0	−1	−1	−5	−4	−2
E	0	−1	1	3	−5	2	4	0	1	−2	−3	0	−2	−5	−1	0	−1	−7	−4	−2
G	1	−3	0	1	−3	−1	0	5	−2	−3	−4	−2	−3	−5	0	1	0	−7	−5	−1
H	−1	2	2	1	−3	3	1	−2	6	−2	−2	0	−2	−2	0	−1	−1	−3	0	−2
I	−1	−2	−2	−2	−2	−2	−2	−3	−2	5	2	−2	2	1	−2	−1	0	−5	−1	4
L	−2	−3	−3	−4	−6	−2	−3	−4	−2	2	6	−3	4	2	−3	−3	−2	−2	−1	2
K	−1	3	1	0	−5	1	0	−2	0	−2	−3	5	0	−5	0	0	0	−3	−4	−2
M	−1	0	−2	−3	−5	−1	−2	−3	−2	2	4	0	6	0	−2	−2	−1	−4	−2	2
F	−3	−4	−3	−5	−4	−5	−5	−5	−2	1	2	−5	0	9	−5	−3	−2	0	7	−1
P	1	0	0	−1	−3	0	−1	0	0	−2	−3	0	−2	−5	6	1	0	−6	−5	−1
S	1	−1	1	0	0	−1	0	1	−1	−1	−3	0	−2	−3	1	2	1	−2	−3	−1
T	1	−1	0	0	−2	−1	−1	0	−1	0	−2	0	−1	−2	0	1	3	−5	−3	0
W	−6	2	−4	−7	−8	−5	−7	−7	−3	−5	−2	−3	−4	0	−6	−2	−5	17	0	−6
Y	−3	−4	−2	−4	0	−4	−4	−5	0	−1	−1	−4	−2	7	−5	−3	−3	0	10	−1
V	0	−2	−2	−2	−2	−2	−2	−1	−2	4	2	−2	2	−1	−1	−1	0	−6	−1	4

BLOSUM45 scoring matrix

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	5	−2	−1	−2	−1	−1	−1	0	−2	−1	−2	−1	−1	−3	−1	1	0	−2	−3	0
R	−2	7	−1	−2	−4	1	−1	−3	0	−3	−3	3	−2	−3	−2	−1	−1	−3	−1	−3
N	−1	−1	7	2	−2	0	0	−1	1	−3	−4	0	−2	−4	−2	1	0	−4	−2	−3
D	−2	−2	2	8	−4	0	2	−1	−1	−4	−5	−1	−4	−5	−1	0	−1	−5	−4	−3
C	−1	−4	−2	−4	13	−3	−4	−3	−4	−2	−2	−3	−2	−3	−4	−1	−1	−4	−3	−1
Q	−1	1	0	0	−3	7	2	−2	1	−3	−2	1	0	−4	−1	−1	−1	−2	−3	−2
E	−1	−1	0	2	−4	2	6	−2	0	−3	−4	0	−2	−4	−1	0	−1	−3	−3	−3
G	0	−3	−1	−1	−3	−2	−2	8	−3	−4	−4	−2	−3	−4	−3	0	−2	−3	−4	−4
H	−2	0	1	−1	−4	1	0	−3	10	−4	−3	−1	−2	−2	−2	−1	−2	−3	2	−4
I	−1	−3	−3	−4	−2	−3	−3	−4	−4	6	2	−3	2	1	−3	−2	0	−4	−1	3
L	−2	−3	−4	−5	−2	−2	−4	−4	−3	2	6	−3	3	1	−3	−3	−2	−2	−1	1
K	−1	3	0	−1	−3	1	0	−2	−1	−3	−3	6	−2	−4	−1	0	−1	−3	−2	−3
M	−1	−2	−2	−4	−2	0	−2	−3	−2	2	3	−2	7	0	−2	−2	−1	−2	−1	1
F	−3	−3	−4	−5	−3	−4	−4	−4	−2	1	1	−4	0	8	−4	−3	−2	1	4	−1
P	−1	−2	−2	−1	−4	−1	−1	−3	−2	−3	−3	−1	−2	−4	10	−1	−1	−4	−4	−3
S	1	−1	1	0	−1	−1	0	0	−1	−2	−3	0	−2	−3	−1	5	2	−4	−3	−2
T	0	−1	0	−1	−1	−1	−1	−2	−2	0	−2	−1	−1	−2	−1	2	6	−4	−2	0
W	−2	−3	−4	−5	−4	−2	−3	−3	−3	−4	−2	−3	−2	1	−4	−4	−4	15	2	−3
Y	−3	−1	−2	−4	−3	−3	−3	−4	2	−1	−1	−2	−1	4	−4	−3	−2	2	8	−1
V	0	−3	−3	−3	−1	−2	−3	−4	−4	3	1	−3	1	−1	−3	−2	0	−3	−1	6

PAM250

	A	R
A	2	−2
R	−2	6
N	0	0
D	0	−1
C	−2	−4
Q	0	1
E	0	−1
G	1	−3
H	−1	2
I	−1	−2
L	−2	−3
K	−1	3
M	−1	0
F	−3	−4
P	1	0
S	1	−1
T	1	−1
W	−6	2
Y	−3	−4
V	0	−2

BLOSUM45

	A	R
A	5	−2
R	−2	7
N	−1	−1
D	−2	−2
C	−1	−4
Q	−1	1
E	−1	−1
G	0	−3
H	−2	0
I	−1	−3
L	−2	−3
K	−1	3
M	−1	−2
F	−3	−3
P	−1	−2
S	1	−1
T	0	−1
W	−2	−3
Y	−3	−1
V	0	−3

Scoring an alignment from a probabilistic viewpoint

```
Protein x:   VLSPAD-KTNVKAAW
Protein y:   VLSPADAKTNVRAAW
```

$$\frac{\Pr(x, y \mid M)}{\Pr(x, y \mid R)}$$

← sequences have common ancestor

← sequences are aligned by chance

- Several assumptions about the evolutionary model of the sequence alignment:
  - Mutations are point changes that occur independently.
  - The rate of evolutionary change is constant over time.
  - Substitutions occur according to known probabilities based on observed data.
  - The sequences share a common ancestor, and gaps represent rare insertions or deletions.

$$\Pr(x, y \mid M) = \prod_i p_{x_i y_i}$$

*i: i*th position in the alignment, positions are independent to each other

$$\Pr(x, y \mid R) = \prod_i q_{x_i} \prod_i q_{y_i}$$

q_a: the frequency of amino acid a

Scoring an alignment from a probabilistic viewpoint

$$\Pr(x, y \mid M) = \prod_i p_{x_i y_i}$$

i: *i*th position in the alignment, positions are independent to each other

$$\Pr(x, y \mid R) = \prod_i q_{x_i} \prod_i q_{y_i}$$

q_a: the frequency of amino acid a

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

$$\log \frac{\Pr(x, y \mid M)}{\Pr(x, y \mid R)} = \sum_i \log \left(\frac{p_{x_i y_i}}{q_{x_i} q_{y_i}} \right)$$

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

log-odds score for amino acid *a*, *b*

$$Alignment\ Score = \sum_i S(a, b)$$

Calculation of log-odds score for two amino acids

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

log-odds score
for amino acid *a*, *b*

$$P_{a,b} = P(a) \cdot P(b \mid a)$$

P(b | a): the mutation probability of amino acid a -> b

- *Let* $M^t(i, j)$ represent the mutation probability of amino acid j to amino acid i in a PAM_n mutation probability matrix, where t refers to a specific evolutionary distance defined by PAM.;
- Let *f(i)* and *f(j)* represent the frequency of amino acid i and j in the background;

$$s(a,b) = \log\left(\frac{P_{ab}}{q_a q_b}\right) = \log\left(\frac{f(j)M^t(i,j)}{f(i)f(j)}\right) = \log\left(\frac{M^t(i,j)}{f(i)}\right)$$

Breaking the mutational probability into two components

$$M^t(i,j) = P(C_i|C_j,t) = P(C_i, i \neq j|C_j,t)$$

The probability of an amino acid C_j mutating into a different amino acid C_i after a time *t*.

The probability that the mutation of C_j results in C_i specifically after a time *t.*

The probability of C_j mutating into any other amino acid (not just C_i) after a time *t.*

$$P(C_i, i \neq j|C_j,t) = P(C_i|C_j, i \neq j, t)P(i \neq j|C_j,t)$$

$t = 1PAM$

PAM1=only 1% amino acids changed.

$$P(C_i, i \neq j|C_j) = P(C_i|C_j, i \neq j)P(i \neq j|C_j)$$

Empirical calculation of the mutational probability —Dayhoff's approach

- **Dayhoff and her colleagues analyzed 71 groups of protein sequences from 34 superfamilies.** These groups consisted of homologous sequences with over 85% sequence identity (with no gaps), indicating a recent common ancestor.

- **Dayhoff manually constructed multiple sequence alignments for each group and inferred ancestral sequences using phylogenetic methods like maximum parsimony.**

- **By comparing pairs of sequences (including ancestral sequences), she identified 1572 accepted point mutations.** These were amino acid substitutions that occurred at specific positions in the alignments.

- **Based on these accepted point mutations, Dayhoff estimated the probability of various amino acid substitutions.** This data formed the basis for the PAM1 matrix, a fundamental tool in evolutionary biology and bioinformatics

What are accepted point mutations?

```
Protein x:  VLSPADAKTNVKAAW
Protein y:  VLSPADAKTNVRAAW
```

- For each aligned position, if there was a difference in the amino acid between two sequences at the same position, Dayhoff and her team assumed that a single point mutation had occurred. This mutation was called an **accepted point mutation** (APM) because it had been accepted by natural selection and persisted in the extant (living) sequences.

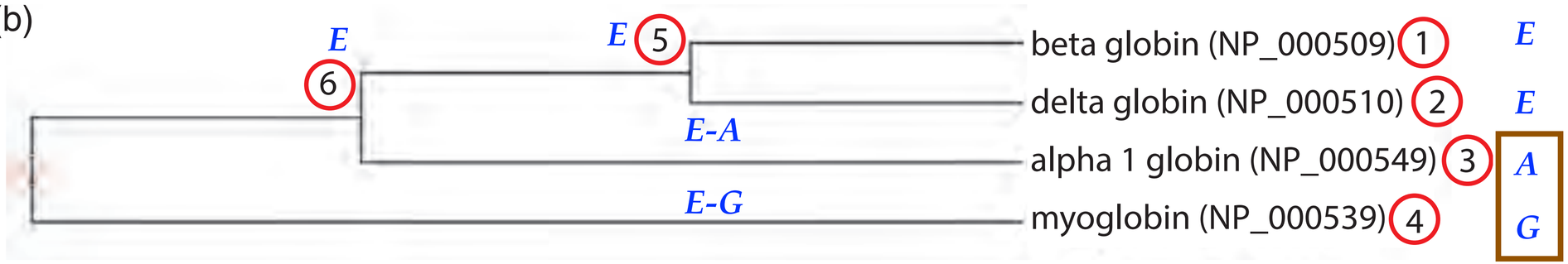
- Because the sequences selected by Dayhoff were closely related, it was assumed that only **one mutation** had occurred at each position over evolutionary time. This assumption is part of the **infinite-sites model**, which assumes that for closely related sequences, the chance of multiple mutations occurring at the same site is extremely low.

Identify accepted point mutations using ancestral sequences reconstructed from phylogenetic trees

(a)

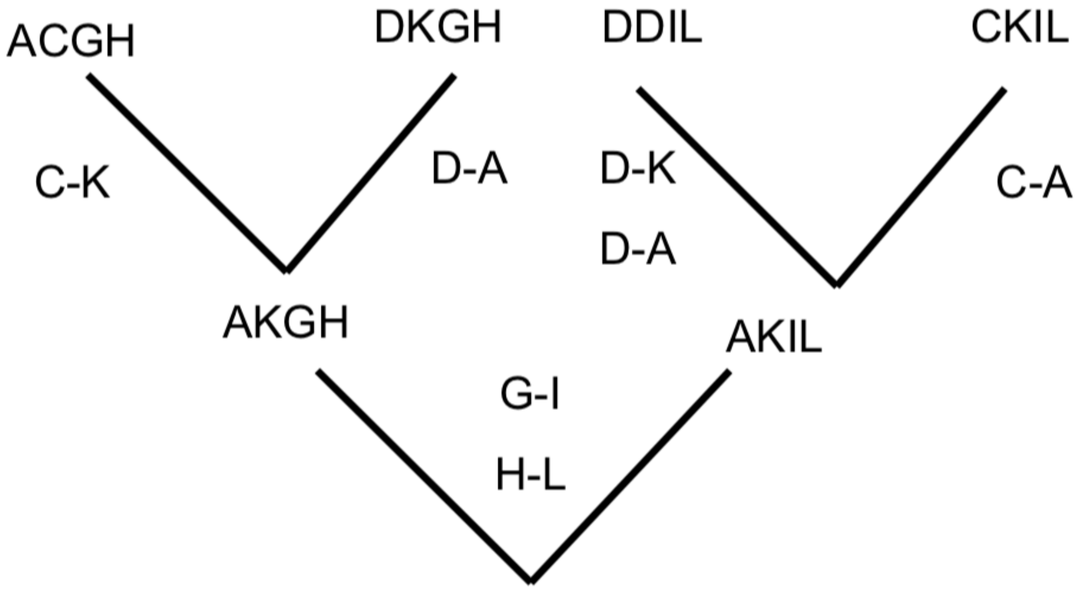
```
beta globin        MVHLTPEEKSAVTALWGKV
delta globin       MVHLTPEEKTAVNALWGKV
alpha 1 globin     MV.LSPADKTNVKAAWGKV
myoglobin          .MGLSDGEWQLVLNVWGKV
5                  MVHLSPEEKTAVNALWGKV
6                  MVHLTPEEKTAVNALWGKV
```

(b)



Dayhoff's approach to determining amino acid substitutions. At each of the columns labeled in red, there was no direct interchange of two amino acids between alpha 1 globin and myoglobin. Instead, an ancestral residue diverged. For example, the arrow in (a) indicates an ancestral glutamate that evolved to become alanine or glycine, but it would not be correct to suggest that alanine had been converted directly to glycine.

Identify accepted point mutations using ancestral sequences reconstructed from phylogenetic trees



	A	C	D	G	H	I	K	L
A		1	2					
C	1						1	
D	2						1	
G						1		
H								1
I				1				
K		1	1					
L					1			

The process assumed that the mutation was reversible, meaning that the probability of a mutation from amino acid A to amino acid B was the same as the probability of a mutation from B to A.

(1) Identify accepted point mutations from 71 phylogenetic trees

	A Ala	R Arg	N Asn	D Asp	C Cys	Q Gln	E Glu	G Gly	H His	I Ile	L Leu	K Lys	M Met	F Phe	P Pro	S Ser	T Thr	W Trp	Y Tyr	V Val
A																				
R	30																			
N	109	17																		
D	154	0	532																	
C	33	10	0	0																
Q	93	120	50	76	0															
E	266	0	94	831	0	422														
G	579	10	156	162	10	30	112													
H	21	103	226	43	10	243	23	10												
I	66	30	36	13	17	8	35	0	3											
L	95	17	37	0	y	75	15	17	40	253										
K	57	477	322	85	0	147	104	60	23	43	39									
M	29	17	0	0	0	20	7	7	0	57	207	90								
F	20	7	7	0	0	0	0	17	20	90	167	0	17							
P	345	67	27	10	10	93	40	49	50	7	43	43	4	7						
S	772	137	432	98	117	47	86	450	26	20	32	168	20	40	269					
T	590	20	169	57	10	37	31	50	14	129	52	200	28	10	73	696				
W	0	27	3	0	0	0	0	0	3	0	13	0	0	10	0	17	0			
Y	20	3	36	0	30	0	10	0	40	13	23	10	0	260	0	22	23	6		
V	365	20	13	17	33	27	37	97	30	661	303	17	77	10	50	43	186	0	17	
	A Ala	R Arg	N Asn	D Asp	C Cys	Q Gln	E Glu	G Gly	H His	I Ile	L Leu	K Lys	M Met	F Phe	P Pro	S Ser	T Thr	W Trp	Y Tyr	V Val

FIGURE 3.8 Numbers of accepted point mutations, multiplied by 10, in 1572 cases of amino acid substitutions from closely related protein sequences. Amino acids are presented alphabetically according to the three-letter code. Notice that some substitutions (green shaded boxes) are very commonly accepted (such as V and I or S and T). Other amino acids, such as C and W, are rarely substituted by any other residue (orange shaded boxes).

Source: Dayhoff (1972). Reproduced with permission from National Biomedical Research Foundation.

(2) Calculate the normalized amino acid frequency

TABLE 3.1 Normalized frequencies of amino acid. These values sum to 1. If the 20 amino acids were equally represented in proteins, these values would all be 0.05 (i.e., 5%); instead, amino acids vary in their frequency of occurrence.

Gly	0.089	Arg	0.041
Ala	0.087	Asn	0.040
Leu	0.085	Phe	0.040
Lys	0.081	Gln	0.038
Ser	0.070	Ile	0.037
Val	0.065	His	0.034
Thr	0.058	Cys	0.033
Pro	0.051	Tyr	0.030
Glu	0.050	Met	0.015
Asp	0.047	Trp	0.010

Source: Dayhoff (1972). Reproduced with permission from National Biomedical Research Foundation.

(3) Calculate relative mutability for each amino acid

Total number of point mutations involving C_j

$$m_j = \frac{\sum_{i \neq j} A_{ij}}{\text{Total occurrences of } C_j}$$

Total number of C_j in 71 groups of sequences

$$m_j \approx P(i \neq j | C_j)$$

- m_j represents the **rate** at which amino acid C_j is involved in mutations compared to its overall frequency.

- m_j is proportional to the mutation probability of C_j into any other amino acid in a given evolutionary time.

Relative mutability of amino acids

TABLE 3.2 Relative mutabilities of amino acids. The value of alanine is arbitrarily set to 100.

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

Source: Dayhoff (1972). Reproduced with permission from National Biomedical Research Foundation.
Dayhoff (1972). Reproduced with permission from National Biomedical Research Foundation.

(4) Calculate the mutational probability of an amino acid mutating into another amino acid

$$P(C_i, i \neq j | C_j) = P(C_i | C_j, i \neq j) P(i \neq j | C_j)$$

$$P(C_i | C_j, i \neq j) = \frac{A_{ij}}{\sum_{k \neq j} A_{kj}}$$

$$P(i \neq j | C_j, t) \approx m_j = \lambda m_j$$

$$P(C_i, i \neq j | C_j) = \lambda m_j \cdot \frac{A_{ij}}{\sum_{k \neq j} A_{kj}}$$

How to estimate λ?

Probability of an amino acid remaining unchanged

$$P(C_j|C_j) = 1 - P(i \neq j|C_j) = 1 - \lambda m(j)$$

Proportion of amino acids remaining unchanged

$$\frac{\sum_{j=1}^{20} n_j P(C_j|C_j)}{N} = 0.99$$

PAM1=only 1% amino acids changed, i.e., 99% unchanged.

$$\frac{\sum_{j=1}^{20} n_j (1 - \lambda m(j))}{N} = \frac{\sum_{j=1}^{20} n(j) - \lambda \sum_{j=1}^{20} n(j)m(j)}{N} = 1 - \lambda \sum_{j=1}^{20} f(j)m(j) = 0.99$$

λ can be caclulated.

PMA1 mutation probability matrix

Mutational probability matrix derived by Dayhoff for the 20 amino acids

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	9867	2	9	10	3	8	17	21	2	6	4	2	6	2	22	35	32	0	2	18
R	1	9913	1	0	1	10	0	0	10	3	1	19	4	1	4	6	1	8	0	1
N	4	1	9822	36	0	4	6	6	21	3	1	13	0	1	2	20	9	1	4	1
D	6	0	42	9859	0	6	53	6	4	1	0	3	0	0	1	5	3	0	0	1
C	1	1	0	0	9973	0	0	0	1	1	0	0	0	0	1	5	1	0	3	2
Q	3	9	4	5	0	9876	27	1	23	1	3	6	4	0	6	2	2	0	0	1
E	10	0	7	56	0	35	9865	4	2	3	1	4	1	0	3	4	2	0	1	2
G	21	1	12	11	1	3	7	9935	1	0	1	2	1	1	3	21	3	0	0	5
H	1	8	18	3	1	20	1	0	9912	0	1	1	0	2	3	1	1	1	4	1
I	2	2	3	1	2	1	2	0	0	9872	9	2	12	7	0	1	7	0	1	33
L	3	1	3	0	0	6	1	1	4	22	9947	2	45	13	3	1	3	4	2	15
K	2	37	25	6	0	12	7	2	2	4	1	9926	20	0	3	8	11	0	1	1
M	1	1	0	0	0	2	0	0	0	5	8	4	9874	1	0	1	2	0	0	4
F	1	1	1	0	0	0	0	1	2	8	6	0	4	9946	0	2	1	3	28	0
P	13	5	2	1	1	8	3	2	5	1	2	2	1	1	9926	12	4	0	0	2
S	28	11	34	7	11	4	6	16	2	2	1	7	4	3	17	9840	38	5	2	2
T	22	2	13	4	1	3	2	2	1	11	2	8	6	1	5	32	9871	0	2	9
W	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	9976	1	0
Y	1	0	3	0	3	0	1	0	4	1	1	0	0	21	0	1	1	2	9945	1
V	13	2	1	1	3	2	2	3	3	57	11	1	17	1	3	2	10	0	2	9901

For clarity, the values have been multiplied by 10000

This matrix corresponds to an evolution time period giving 1 mutation/100 amino acids, and is refered to as the **PAM1 matrix.**

$$P(C_i, i \neq j | C_j) = \lambda m_j \cdot \frac{A_{ij}}{\sum_{k \neq j} A_{kj}}$$

How to estimate the mutation probability matrix at PAM2?

Source: Dayhoff, 1978

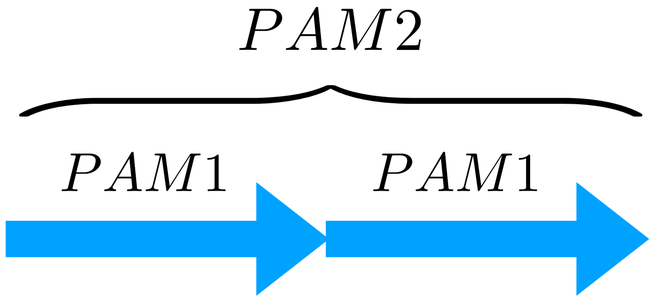
Markovian Assumption in Evolution

- **A Markov process is a stochastic process where the future state depends only on the present state and not on the past states.** In other words, the "memoryless" property holds: the past history of the process is irrelevant for predicting its future, given the current state.

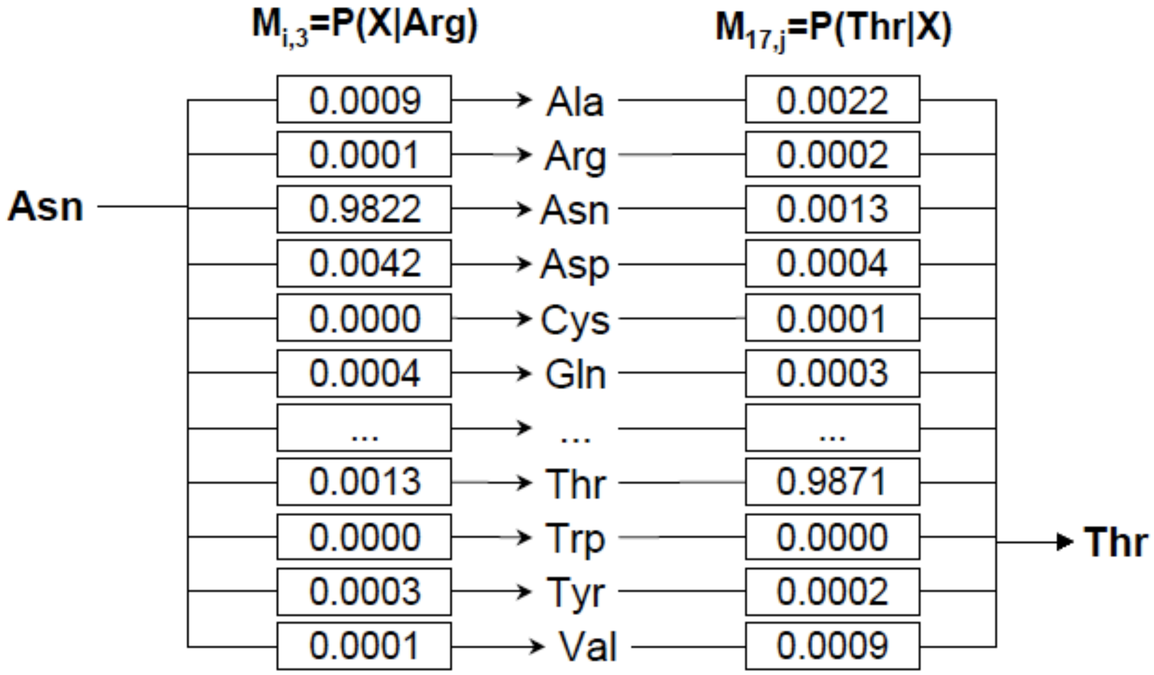
  - **Weather Patterns:** Weather patterns can often be modeled as Markov processes. The weather tomorrow depends primarily on today's weather conditions and not on the weather from previous days.

- The inference that **evolution** is **Markovian** assumes that base changes (or amino acid changes) occur at a constant rate and depend only on the *identity of the current base (or amino acid).*

Markovian Assumption in Evolution



From PAM1 to PAM2



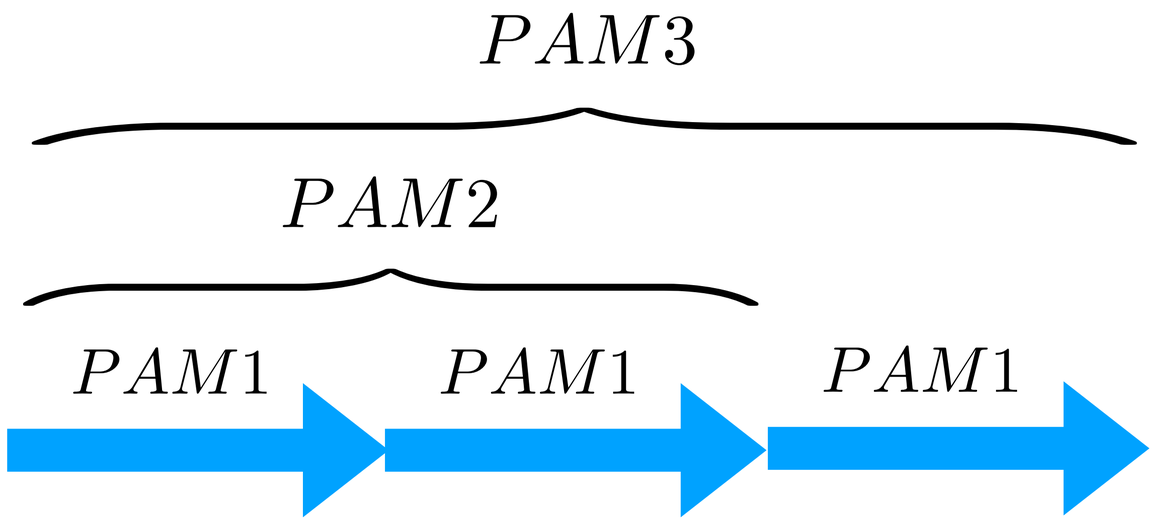
P(Asn -> Thr)= P(Asn -> Ala -> Thr) + P(Asn -> Arg -> Thr) + ... + P(Asn -> Val -> Thr)
= (0.0009)(0.0001) + (0.0001)(0.0002) + ... + (0.0001)(0.009)

line 3 of PAM1 column 17 of PAM1

=> Matrix product: PAM2 = PAM1 x PAM1

Source: J. van Helden

Markovian Assumption in Evolution



$$PAM3 = PAM1 \times PAM1 \times PAM1 = PAM1^3$$

PAM250 mutation probability matrix

$$PAM250 = PAM1^{250}$$

Replacement amino acid	Original amino acid																			
		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
	A	13	6	9	9	5	8	9	12	6	8	6	7	7	4	11	11	11	2	4	9
	R	3	17	4	3	2	5	3	2	6	3	2	9	4	1	4	4	3	7	2	2
	N	4	4	6	7	2	5	6	4	6	3	2	5	3	2	4	5	4	2	3	3
	D	5	4	8	11	1	7	10	5	6	3	2	5	3	1	4	5	5	1	2	3
	C	2	1	1	1	52	1	1	2	2	2	1	1	1	1	2	3	2	1	4	2
	Q	3	5	5	6	1	10	7	3	7	2	3	5	3	1	4	3	3	1	2	3
	E	5	4	7	11	1	9	12	5	6	3	2	5	3	1	4	5	5	1	2	3
	G	12	5	10	10	4	7	9	27	5	5	4	6	5	3	8	11	9	2	3	7
	H	2	5	5	4	2	7	4	2	15	2	2	3	2	2	3	3	2	2	3	2
	I	3	2	2	2	2	2	2	2	2	10	6	2	6	5	2	3	4	1	3	9
	L	6	4	4	3	2	6	4	3	5	15	34	4	20	13	5	4	6	6	7	13
	K	6	18	10	8	2	10	8	5	8	5	4	24	9	2	6	8	8	4	3	5
	M	1	1	1	1	0	1	1	1	1	2	3	2	6	2	1	1	1	1	1	2
	F	2	1	2	1	1	1	1	1	3	5	6	1	4	32	1	2	2	4	20	3
	P	7	5	5	4	3	5	4	5	5	3	3	4	3	2	20	6	5	1	2	4
	S	9	6	8	7	7	6	7	9	6	5	4	7	5	3	9	10	9	4	4	6
	T	8	5	6	6	4	5	5	6	4	6	4	6	5	3	6	8	11	2	3	6
	W	0	2	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	55	1	0
	Y	1	1	2	1	3	1	1	1	3	2	2	1	2	15	1	2	2	3	31	2
	V	7	4	4	4	4	4	4	5	4	15	10	4	10	5	5	5	7	2	4	17

FIGURE 3.13 The PAM250 mutation probability matrix. At this evolutionary distance, only one in five amino acid residues remains unchanged from an original amino acid sequence (columns) to a replacement amino acid (rows). Note that the scale has changed relative to **Figure 3.11**, and the columns sum to 100.

PAM0 -> PAM∞

original amino acid

PAM0	A	R	N	D	C	Q	E	G
A	100	0	0	0	0	0	0	0
R	0	100	0	0	0	0	0	0
N	0	0	100	0	0	0	0	0
D	0	0	0	100	0	0	0	0
C	0	0	0	0	100	0	0	0
Q	0	0	0	0	0	100	0	0
E	0	0	0	0	0	0	100	0
G	0	0	0	0	0	0	0	100

replacement amino acid replacement amino acid

original amino acid

PAM∞	A	R	N	D	C	Q	E	G
A	8.7	8.7	8.7	8.7	8.7	8.7	8.7	8.7
R	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1
N	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0
D	4.7	4.7	4.7	4.7	4.7	4.7	4.7	4.7
C	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3
Q	3.8	3.8	3.8	3.8	3.8	3.8	3.8	3.8
E	5.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0
G	8.9	8.9	8.9	8.9	8.9	8.9	8.9	8.9

FIGURE 3.12 Portion of the matrices for a zero PAM value (PAM0; upper panel) or for an infinite PAM∞ value (lower panel). At PAM∞ (i.e., if the PAM1 matrix is multiplied by itself an infinite number of times), all the entries in each row converge on the normalized frequency of the replacement amino acid (see **Table 3.1**). A PAM2000 matrix has similar values that tend to converge on these same limits. In a PAM2000 matrix, the proteins being compared are at an extreme of unrelatedness. In constrast, at PAM0 no mutations are tolerated and the residues of the proteins are perfectly conserved.

The PAM250 log-odds scoring matrix

$$s(a,b) = \log\left(\frac{p_{ab}}{q_a q_b}\right) = log\frac{f(j)M^n(i,j)}{f(i)f(j)} = log\frac{M^n(i,j)}{f(i)}$$

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	2																			
R	-2	6																		
N	0	0	2																	
D	0	-1	2	4																
C	-2	-4	-4	-5	12															
Q	0	1	1	2	-5	4														
E	0	-1	1	3	-5	2	4													
G	1	-3	0	1	-3	-1	0	5												
H	-1	2	2	1	-3	3	1	-2	6											
I	-1	-2	-2	-2	-2	-2	-2	-3	-2	5										
L	-2	-3	-3	-4	-6	-2	-3	-4	-2	-2	6									
K	-1	3	1	0	-5	1	0	-2	0	-2	-3	5								
M	-1	0	-2	-3	-5	-1	-2	-3	-2	2	4	0	6							
F	-3	-4	-3	-6	-4	-5	-5	-5	-2	1	2	-5	0	9						
P	1	0	0	-1	-3	0	-1	0	0	-2	-3	-1	-2	-5	6					
S	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	2				
T	1	-1	0	0	-2	-1	0	0	-1	0	-2	0	-1	-3	0	1	3			
W	-6	2	-4	-7	-8	-5	-7	-7	-3	-5	-2	-3	-4	0	-6	-2	-5	17		
Y	-3	-4	-2	-4	0	-4	-4	-5	0	-1	-1	-4	-2	7	-5	-3	-3	0	10	
V	0	-2	-2	-2	-2	-2	-2	-1	-2	4	2	-2	2	-1	-1	-1	0	-6	-2	4

FIGURE 3.14 Log-odds matrix for PAM250. High PAM values (e.g., PAM250) are useful for aligning very divergent sequences. A variety of algorithms for pairwise alignment, multiple sequence alignment, and database searching (e.g., BLAST) allow you to select an assortment of PAM matrices such as PAM250, PAM70, and PAM30. Adapted from NCBI, ftp://ftp.ncbi.nlm.nih.gov/blast/matrices/.

Scoring an alignment using PAM250

```
T  A  H  G  K
Y  S  D  G  D
```

$S_{alignment}$ = s(T,Y) + s(A,S) + s(H,D) + s(G,G) + s(K,D)

= -3 + 1 + 1 + 5 + 0

= 4

How to choose PAM?

Altschul SF(1991) Amino acid substitution matrices from an information theoretic perspective. *J Mol Biol.* 219:555-65.

- PAM120 matrix is the most appropriate for database searches
- PAM200 matrix is the most appropriate for comparing two specific proteins with suspected homology

Remark:

In the PAM matrices, the **index** indicates the percentage of substitution per position.

Higher indexes are more appropriate for **more distant** proteins (PAM250 better than PAM100 for distant proteins).

Limitation of PAM matrices

- **Assumption of small evolutionary distances** and assumption of linearity in mutations over time. PAM matrices are therefore less accurate for highly diverged sequences.
- **Small and biased dataset**: Derived from a limited set of closely related proteins.
- **Reversibility assumption**: Mutations are assumed to occur equally in both directions, which isn't always true.
- **Constant substitution rates**: In reality, substitution rates can vary due to factors like evolutionary pressures, functional constraints on the protein, or changes in mutation rates across different time periods or environments.
- **Ignores structural/functional constraints**: Treats all amino acid sites equally, overlooking important functional roles.
- **Single-nucleotide substitution bias**: Does not consider codon bias or transition/transversion differences.

BLOSUM matrices

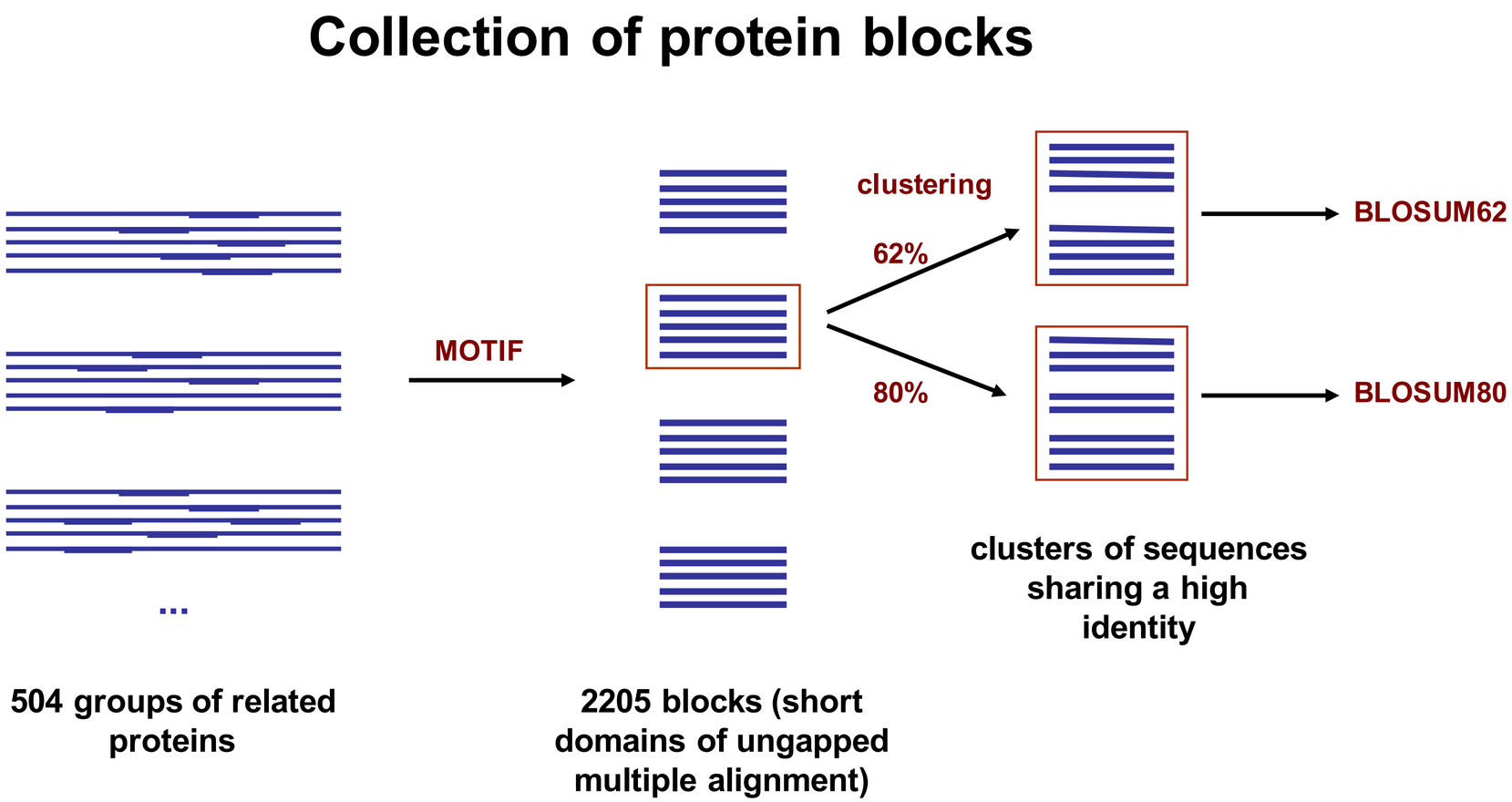
- BLOSUM matrices were developed by analyzing groups of similar (but not identical) protein sequences. Researchers looked at regions where the sequences were highly conserved (called "blocks") and counted how often each amino acid was substituted for another in those regions. By grouping sequences that were a certain percentage identical (e.g., 62% for BLOSUM62), they could focus on sequences with specific evolutionary distances.

- **Wide evolutionary distances**: BLOSUM matrices are based on blocks of conserved sequences, allowing them to be effective for both closely and distantly related sequences. **Larger and more diverse dataset**: BLOSUM matrices use a broader set of protein sequences, capturing more realistic evolutionary changes.

- **Variable substitution rates**: By using different thresholds (e.g., BLOSUM62, BLOSUM80), BLOSUM matrices allow for analysis at various levels of evolutionary divergence.

- **No reversibility assumption**: BLOSUM matrices do not assume that mutations are equally likely in both directions, providing more realistic substitution probabilities.

BLOSUM matrices



- Collect Protein Sequences
- Identify Conserved Regions (Blocks)
- Cluster Sequences by Similarity
- Count Amino Acid Substitutions
- Calculate Log-Odds Scores
- Generate the BLOSUM Matrix

BLOSUM62 matrix

		Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val	B	Z	X	*
		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V				
Ala	A	4																							
Arg	R	-1	5																						
Asn	N	-2	0	6																					
Asp	D	-2	-2	1	6																				
Cys	C	0	-3	-3	-3	9																			
Gln	Q	-1	1	0	0	-3	5																		
Glu	E	-1	0	0	2	-4	2	5																	
Gly	G	0	-2	0	-1	-3	-2	-2	6																
His	H	-2	0	1	-1	-3	0	0	-2	8															
Ile	I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4														
Leu	L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4													
Lys	K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5												
Met	M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5											
Phe	F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6										
Pro	P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7									
Ser	S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4								
Thr	T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5							
Trp	W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11						
Tyr	Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7					
Val	V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4				
	B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4			
	Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	1	4		
	X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	0	0	-2	-1	-1	-1	-1	-1	
	*	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1

Here is the BLOSUM matrix obtained by Henikoff & Henikoff on the basis of 2205 block of proteins. In each group, the sequences have been clustered together if they were 62% identical. This matrix is referred to as **BLOSUM62**.

PAM vs. BLOSUM matrices

- PAM matrices are derived **global** alignments of closely related sequences (full-length sequences), assuming that all amino acids have an equal probability of mutating. BLOSUM matrices are derived from **local** alignments of conserved regions within protein sequences (blocks).

- PAM matrices are more suitable for comparing closely related sequences, while BLOSUM matrices can handle both closely and distantly related sequences. BLOSUM matrices are generally less sensitive to sequence composition than PAM matrices.

Multiple sequence alignments (MSA)

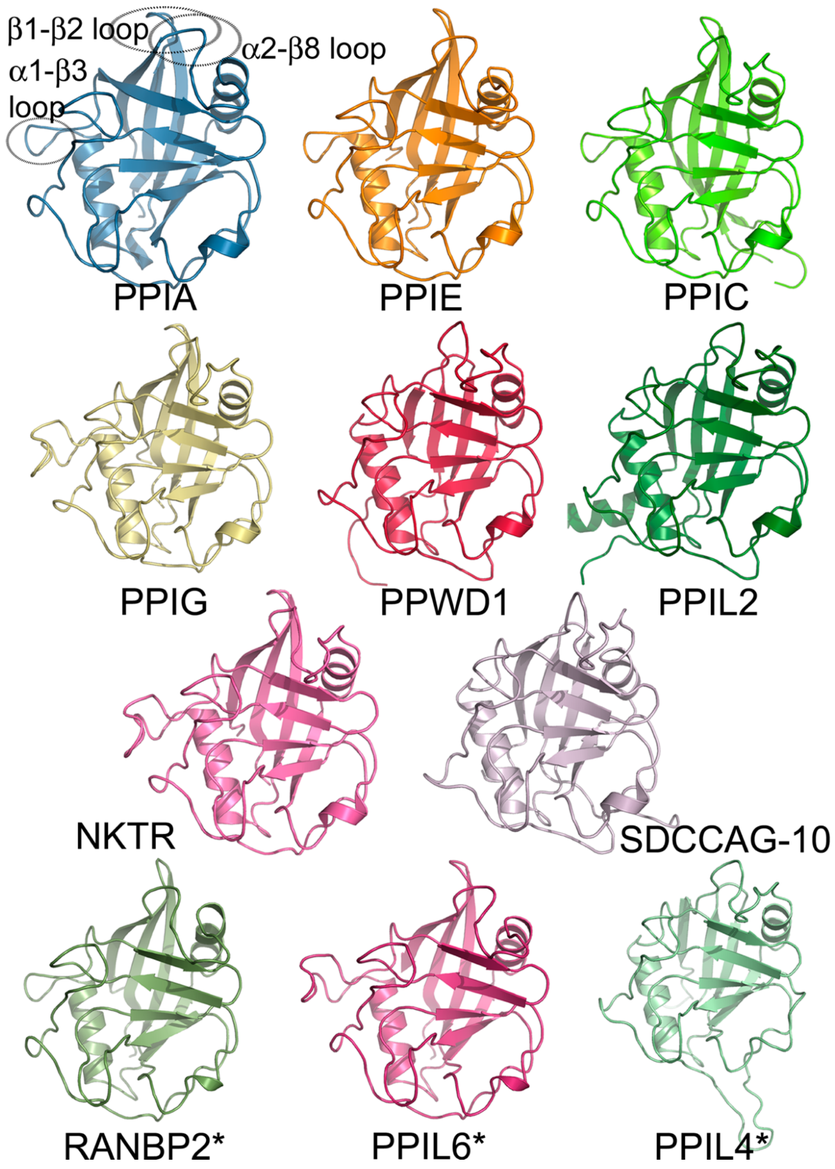
An example of multiple sequence alignment

```
Protein A: MTQKAIWLTAALQD
Protein B: MTQKGIWLTAAGQD
Protein C: MSQKAIWLTAALQE
```

- Multiple sequence alignment (MSA) of protein sequences is used to align three or more protein sequences in a way that maximizes their similarity, revealing conserved regions that may have functional or structural significance.

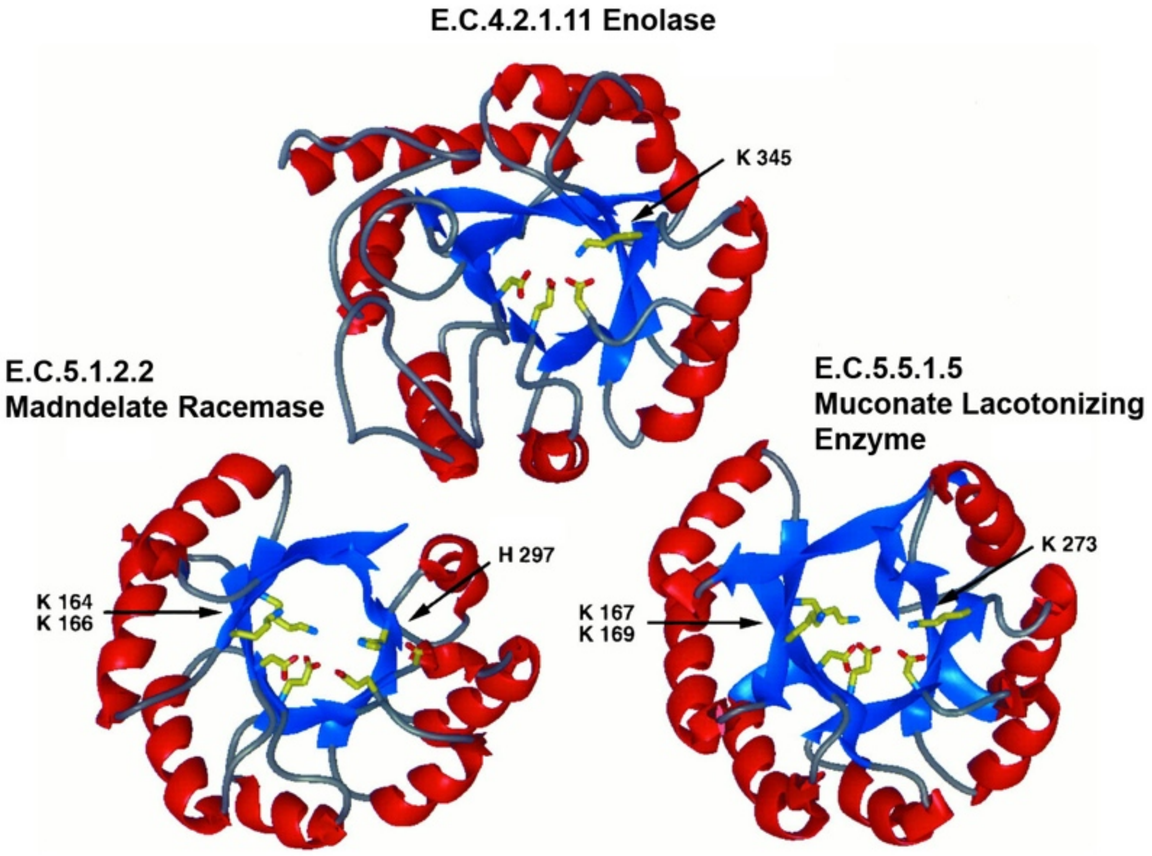
- The positions with the same or similar amino acids across all sequences are often biologically significant.

The evolution of protein family



- The isomerase domains of sequences in the human cyclophilin family have highly conserved structure.
- These domains are functionally versatile, involved in protein folding, immune regulation, and viral replication.

The evolution of protein functions



- During evolution, protein sequences can diverge significantly. However, residues that are crucial for maintaining the protein's function or structure are often conserved to ensure the protein remains functional.

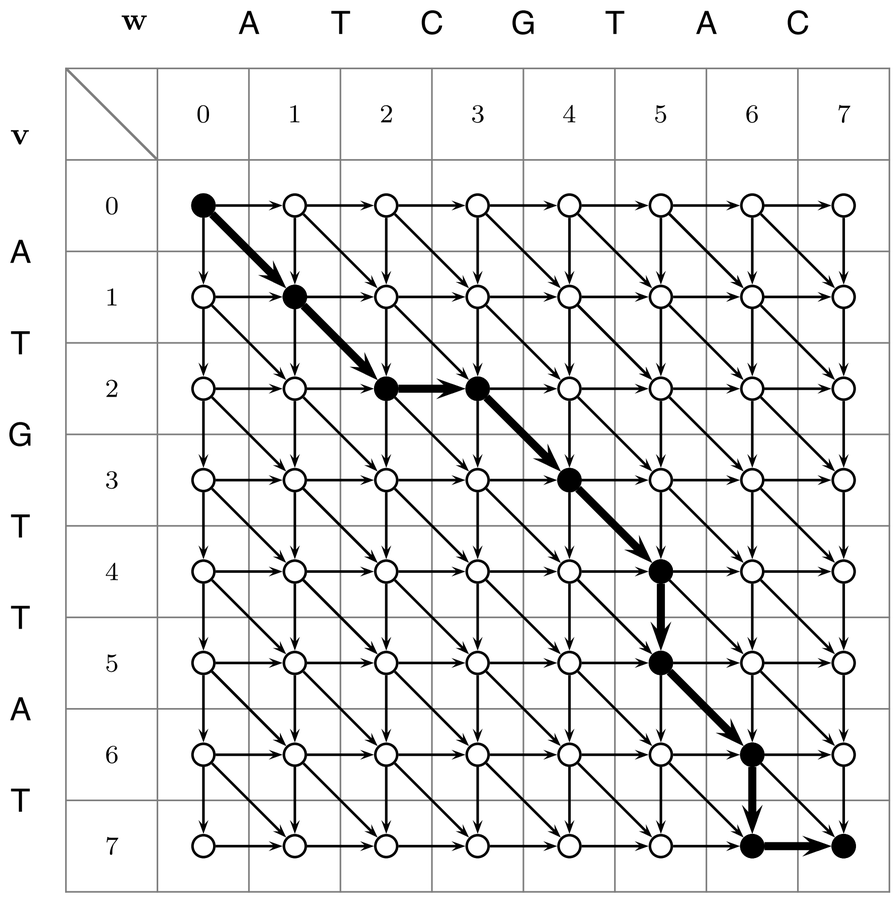
- By superimposing the structures of proteins, we can derive multiple sequence alignments (MSAs) for a protein family, which helps in identifying residues that are functionally or structurally important.

- Most protein sequences lack experimentally determined structures, so we must use the sequences themselves to construct multiple sequence alignments (MSAs). How is this done, and how accurate are these alignments?

Multiple sequence alignments

- Applications :
  - phylogeny
  - domain organization
  - functional residue identification
  - 2D/3D structure prediction
  - transmembrane prediction
  - …

Pairwise alignment can be done by dynamic programming

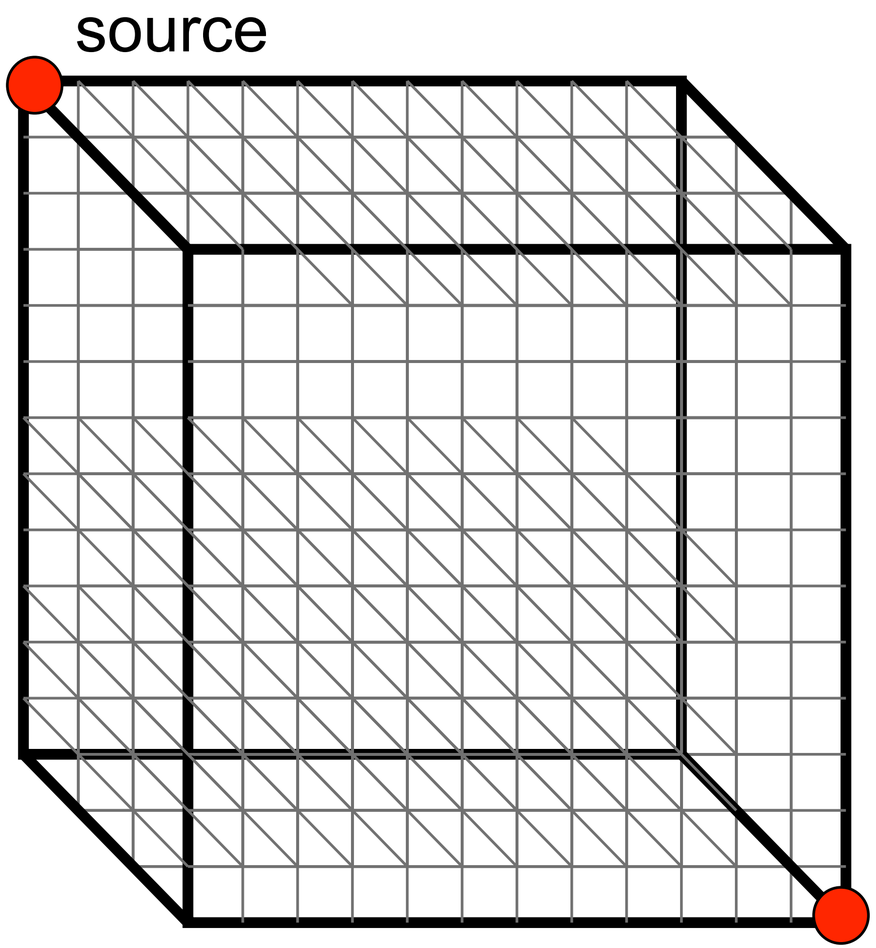


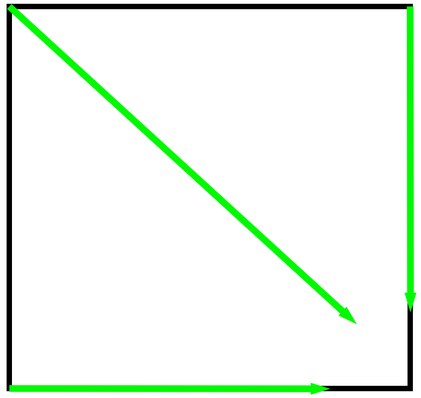
↘ ↘ → ↘ ↘ ↓ ↘ ↓ →
A T - G T T A T -
A T C G T - A - C

$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \\ s_{i-1,j-1} + \delta(v_i, w_j) \end{cases}$$

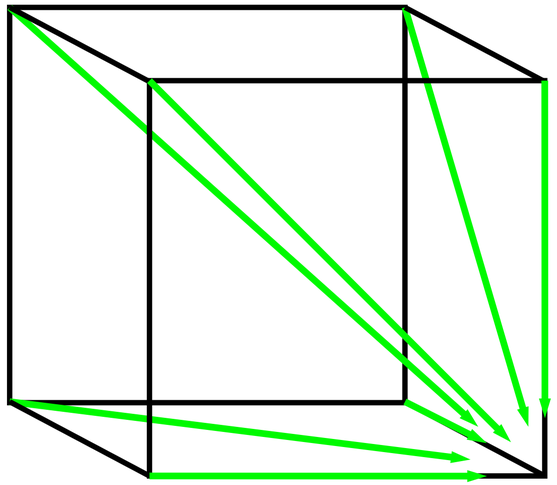
$$s_{i,j} = \max \begin{cases} 0 \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \\ s_{i-1,j-1} + \delta(v_i, w_j) \end{cases}$$

Use a 3-D “Manhattan Cube” to align three sequences by dynamic programming



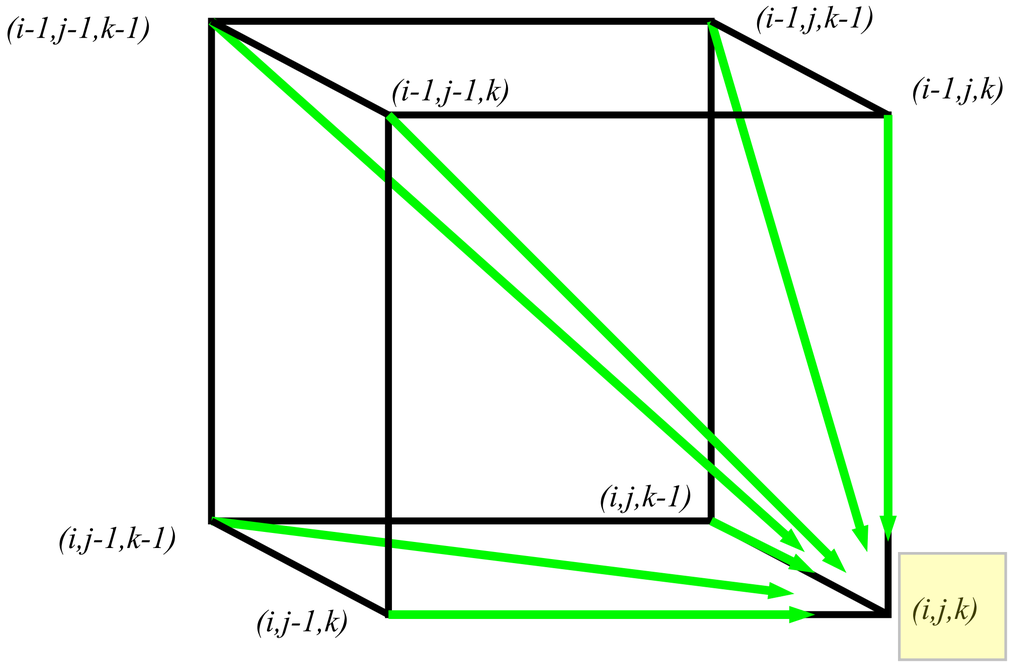


In **2-D**, 3 edges in each unit square



In **3-D**, 7 edges in each unit cube

Use a 3-D “Manhattan Cube” to align three sequences by dynamic programming



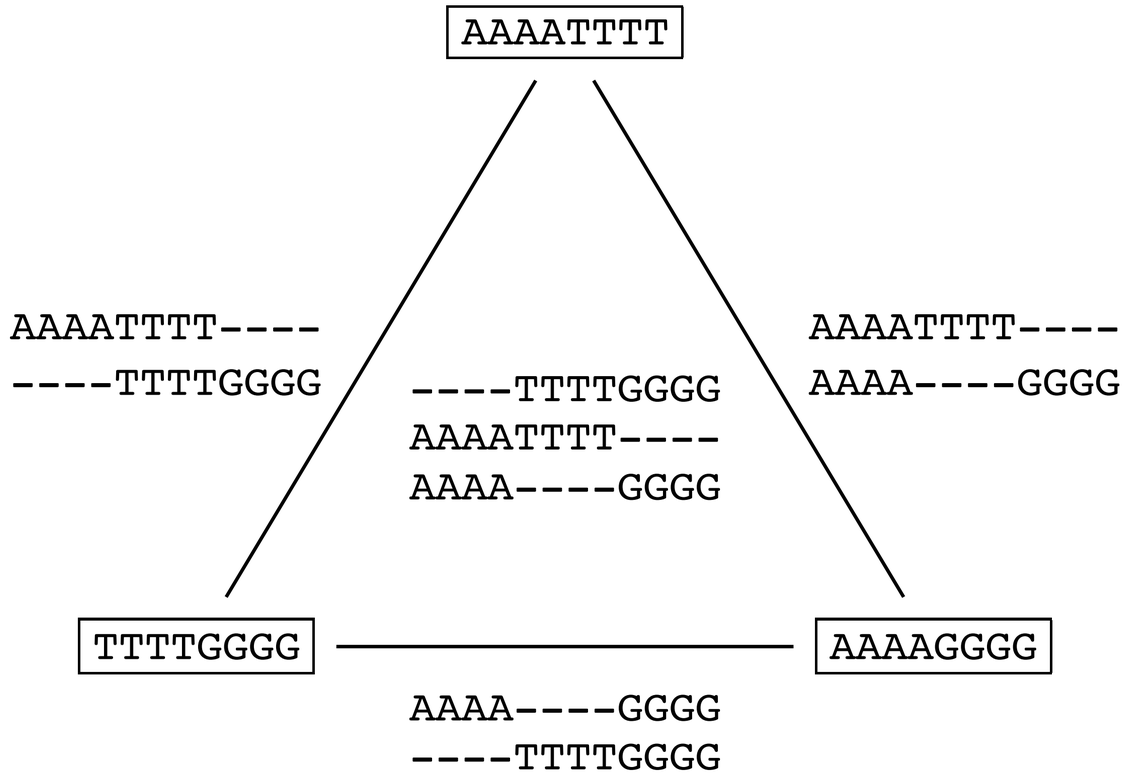
$$s_{i,j,k} = \max \begin{cases} s_{i-1,j-1,k-1} + \delta(v_i, w_j, u_k) & \text{cube diagonal: no indels} \\ s_{i-1,j-1,k} + \delta(v_i, w_j, \_) & \\ s_{i-1,j,k-1} + \delta(v_i, \_, u_k) & \text{face diagonal: one indel} \\ s_{i,j-1,k-1} + \delta(\_, w_j, u_k) & \\ s_{i-1,j,k} + \delta(v_i, \_, \_) & \\ s_{i,j-1,k} + \delta(\_, w_j, \_) & \text{edge diagonal: two indels} \\ s_{i,j,k-1} + \delta(\_, \_, u_k) & \end{cases}$$

$\delta(x, y, z)$ is an entry in the 3-D scoring matrix

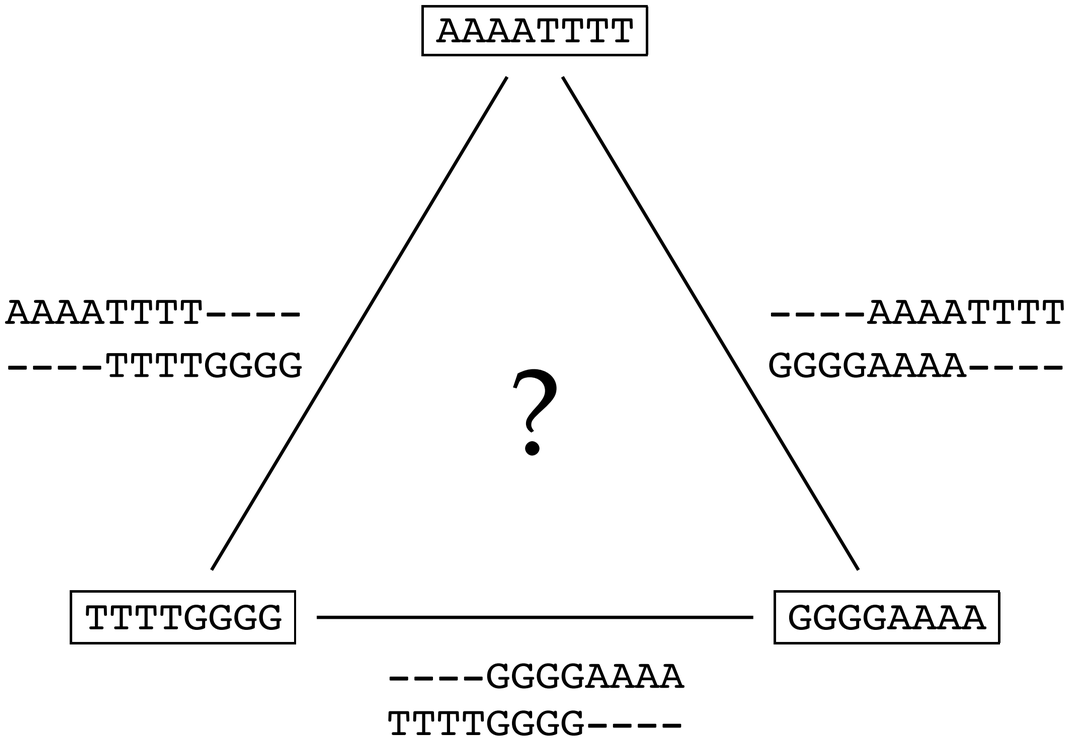
Computation complexity of dynamic programming-based multiple sequence alignment

- For 3 sequences of length n, the run time is $7n^3$; $O(n^3)$
- For k sequences, build a k-dimensional Manhattan, with run time $(2^k-1)(n^k)$; $O(2^k n^k)$
- Dynamic programming approach for alignment between two sequences is easily extended to k sequences, but is impractical due to exponential running time.

Direct inference of MSA based on pairwise alignment?

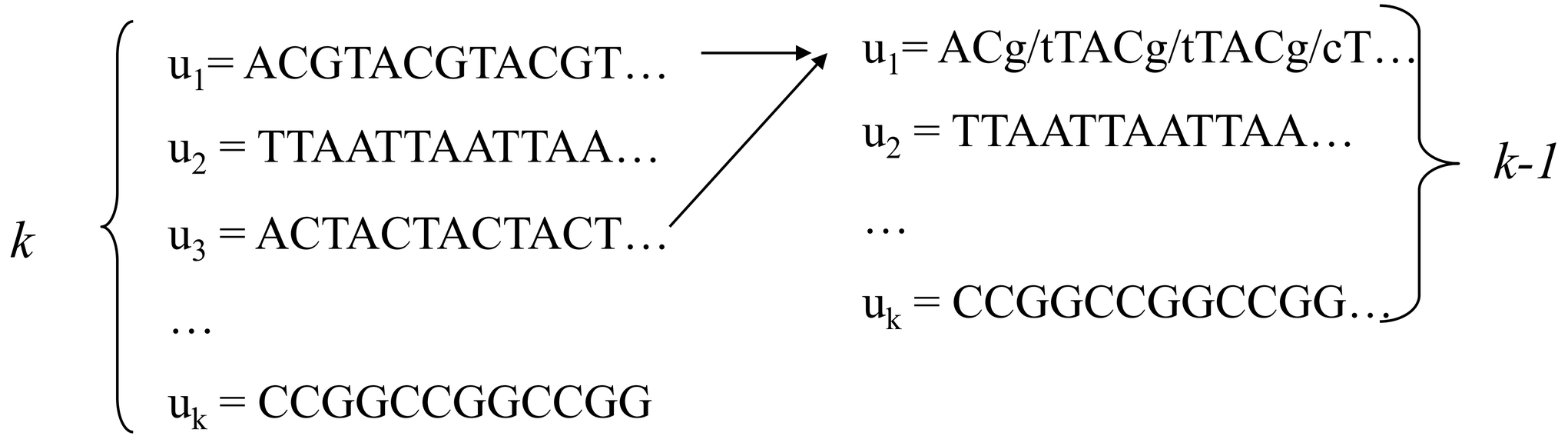


(a) Compatible pairwise alignments



(b) Incompatible pairwise alignments

A heuristic approach to construct MSA



- To align multiple sequences, we can start by finding the two most similar sequences and aligning them. Then, we can create a profile from this alignment and align the remaining sequences to this profile. We can repeat this process until all sequences are aligned.

Align a sequence to a profile using dynamic programming

	A	2/3	0	0	1/3	0
	C	0	1	1/3	0	0
	G	1/3	0	0	2/3	0
	T	0	0	0	0	1
	-	0	0	2/3	0	0
	0	-3	-6	-9	-12	-15
A	-3	5/3	-4/3	-5/3	-14/3	-23/3
C	-6	-4/3	11/3	10/3	1/3	-8/3
G	-9	-13/3	2/3	3	5	2

match=3
mismatch=-1
gap=-3

UP = -3 + G = -3 – 3 = -6
LEFT = -3 + G = -3 -3 = -6
DIAG = 0 + 2/3 × M_{Nuc} + 1/3 × MM_s =5/3

DIAG = -3-1 =-4
UP = -6 -3 = -9
LEFT = 5/3 - 3 = -4/3

A typical progressive alignment

- Progressive alignment is a variation of greedy algorithm with a somewhat more intelligent strategy for choosing the order of alignments.

- Three-step process

  1.) Construct pairwise alignments

  2.) Build Guide Tree

  3.) Progressive Alignment guided by the tree

Step 1: pairwise sequence alignment

- Aligns each sequence against each other
- Calculate a similarity matrix between all sequences. Similarity = exact matches / sequence length (percent identity)

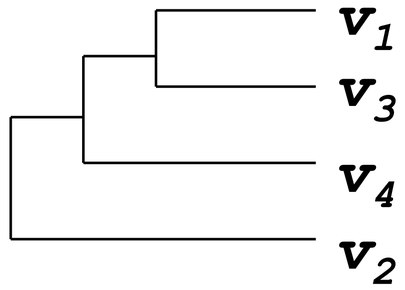
	v_1	v_2	v_3	v_4
v_1	-			
v_2	.17	-		
v_3	.87	.28	-	
v_4	.59	.33	.62	-

(.17 means 17 % identical)

Step 2: guide tree

- Create a Guide Tree using the similarity matrix, which roughly reflects evolutionary relations.

	v_1	v_2	v_3	v_4
v_1	-			
v_2	.17	-		
v_3	.87	.28	-	
v_4	.59	.33	.62	-



Calculate:

$v_{1,3}$ = alignment (v_1, v_3)

$v_{1,3,4}$ = alignment$((v_{1,3}),v_4)$

$v_{1,2,3,4}$ = alignment$((v_{1,3,4}),v_2)$

Step 3: progressive alignment

- Start by aligning the two most similar sequences.
- Following the guide tree, add in the next sequences, aligning to the existing alignment.
- Insert gaps as necessary.

```
FOS_RAT          PEEMSVTS-LDLTGGLPEATTPESEEAFTLPLLNDPEPK-PSLEPVKNISNMELKAEPFD
FOS_MOUSE        PEEMSVAS-LDLTGGLPEASTPESEEAFTLPLLNDPEPK-PSLEPVKSISNVELKAEPFD
FOS_CHICK        SEELAAATALDLG----APSPAAAEEAFALPLMTEAPPAVPPKEPSG--SGLELKAEPFD
FOSB_MOUSE       PGPGPLAEVRDLPG-----STSAKEDGFGWLLPPPPPPP-----------------LPFQ
FOSB_HUMAN       PGPGPLAEVRDLPG-----SAPAKEDGFSWLLPPPPPPP-----------------LPFQ
                 .   . :   ** .     :..  *:.*   *   . *                   **:
```

Dots and stars show how well-conserved a column is.

The first widely used MSA tool— clustalW

- Major improvements on the alignment parameter problem: **clustalW dynamically vary the gap penalties in a position- and residue-specific manner:**

- The observed relative frequencies of gaps adjacent to each of the 20 amino acids (12) are used to locally adjust the gap opening penalty after each residue. The locations of the gaps found in the early alignments are also given reduced gap opening penalties.

- Different weight matrices are chosen as the alignment proceeds, depending on the estimated divergence of the sequences to be aligned at each stage.

Gene weighting by clustalW

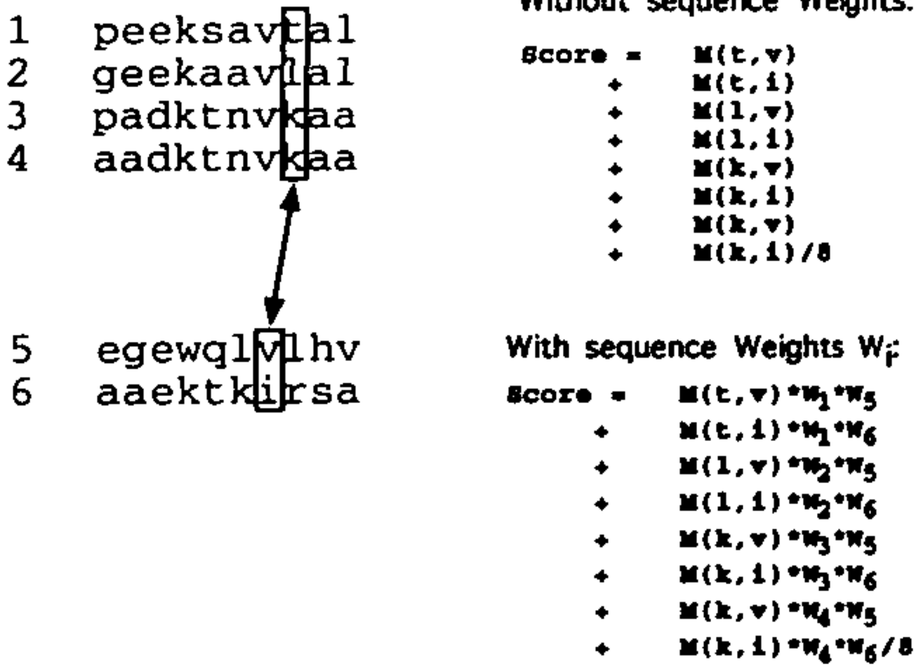


Figure 2. The scoring scheme for comparing two positions from two alignments. Two sections of alignment with 4 and 2 sequences respectively are shown. The score of the position with amino acids T,L,K,K versus the position with amino acids V and I is given with and without sequence weights. $M(X,Y)$ is the weight matrix entry for amino acid X versus amino acid Y. W_n is the weight for sequence n.

Residue and position-specific gap opening penalties in clustalW

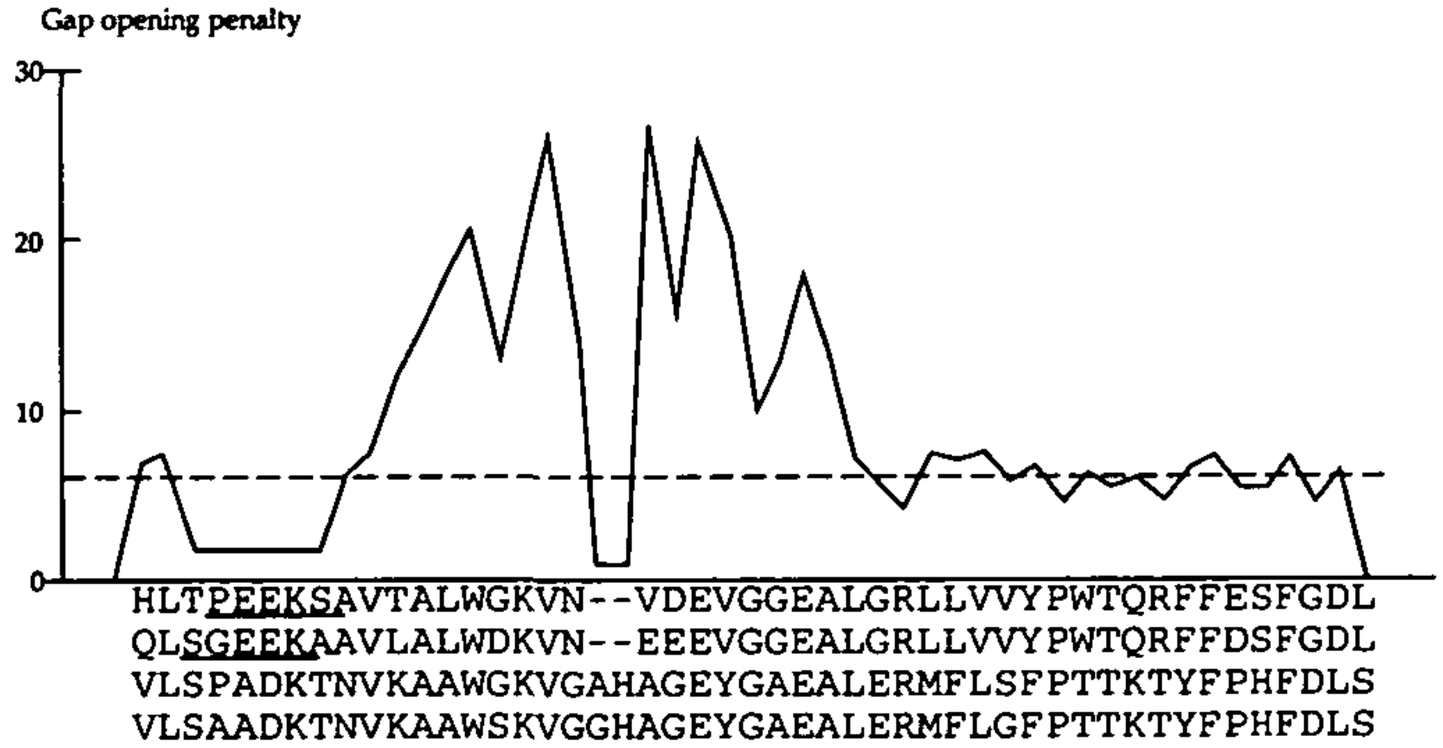
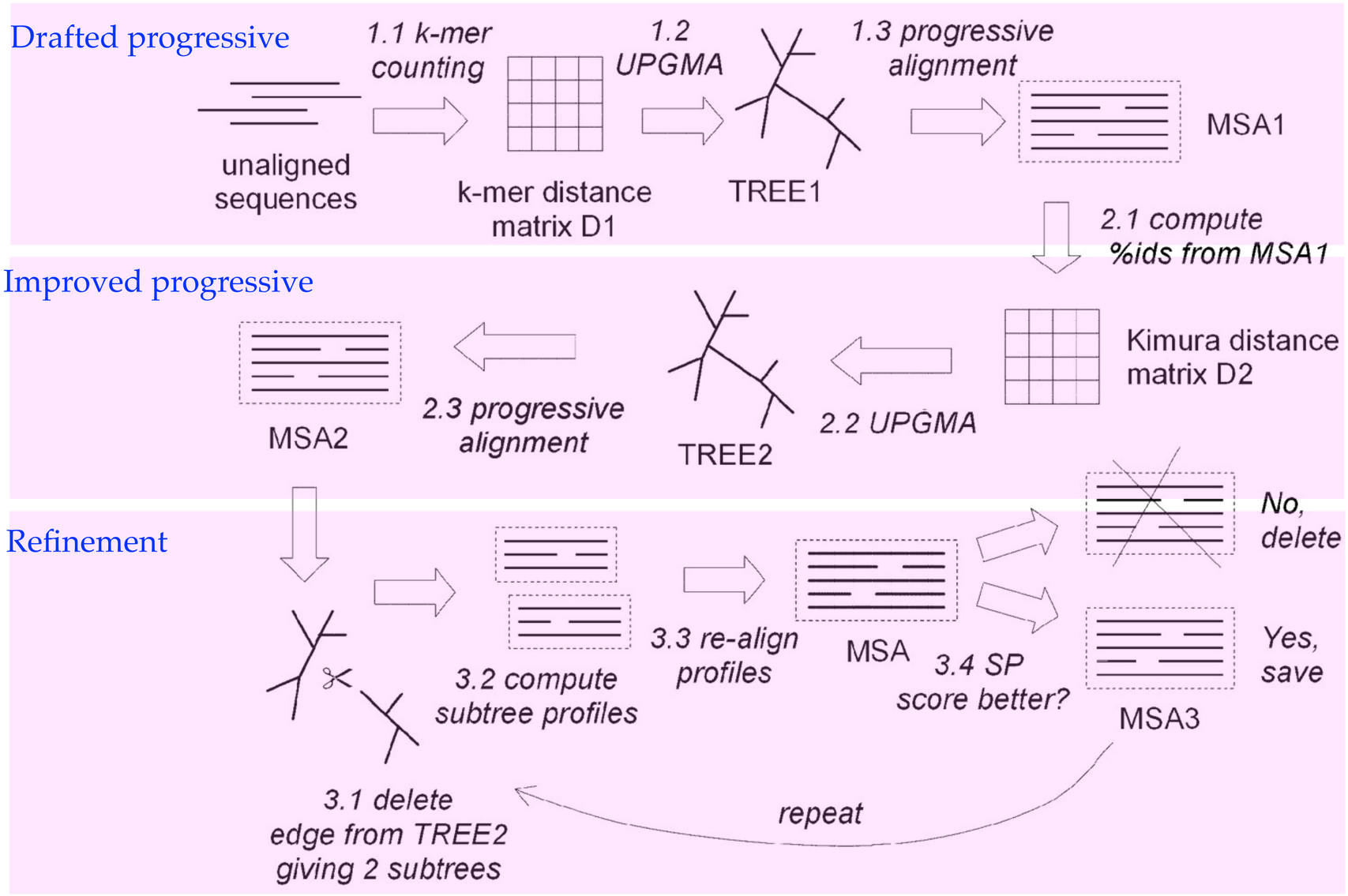


Figure 3. The variation in local gap opening penalty is plotted for a section of alignment. The inital gap opening penalty is indicated by a dotted line. Two hydrophilic stretches are underlined. The lowest penalties correspond to the ends of the alignment, the hydrophilic stretches and the two positions with gaps. The highest values are within 8 residues of the two gap positions. The rest of the variation is caused by the residue specific gap penalties (12).

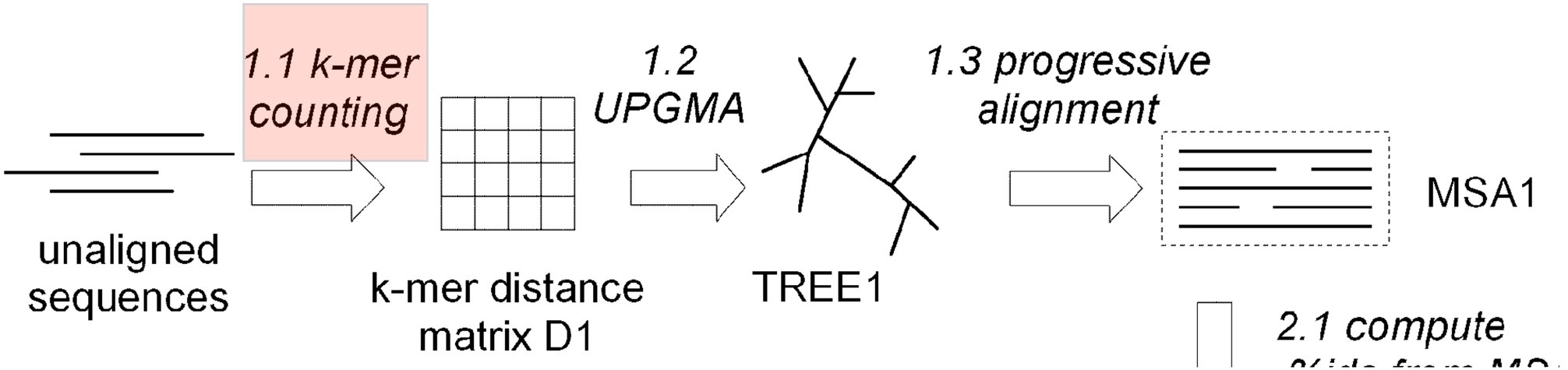
Limitations of clustalW

- Quality deteriorates when more distantly related sequences are included.Develop familiarity with key bioinformatics algorithms
- Pairwise alignment using dynamic programming is conducted first, which will be time consuming to align large number of sequences.

Current widely used MSA tool—MUSCLE



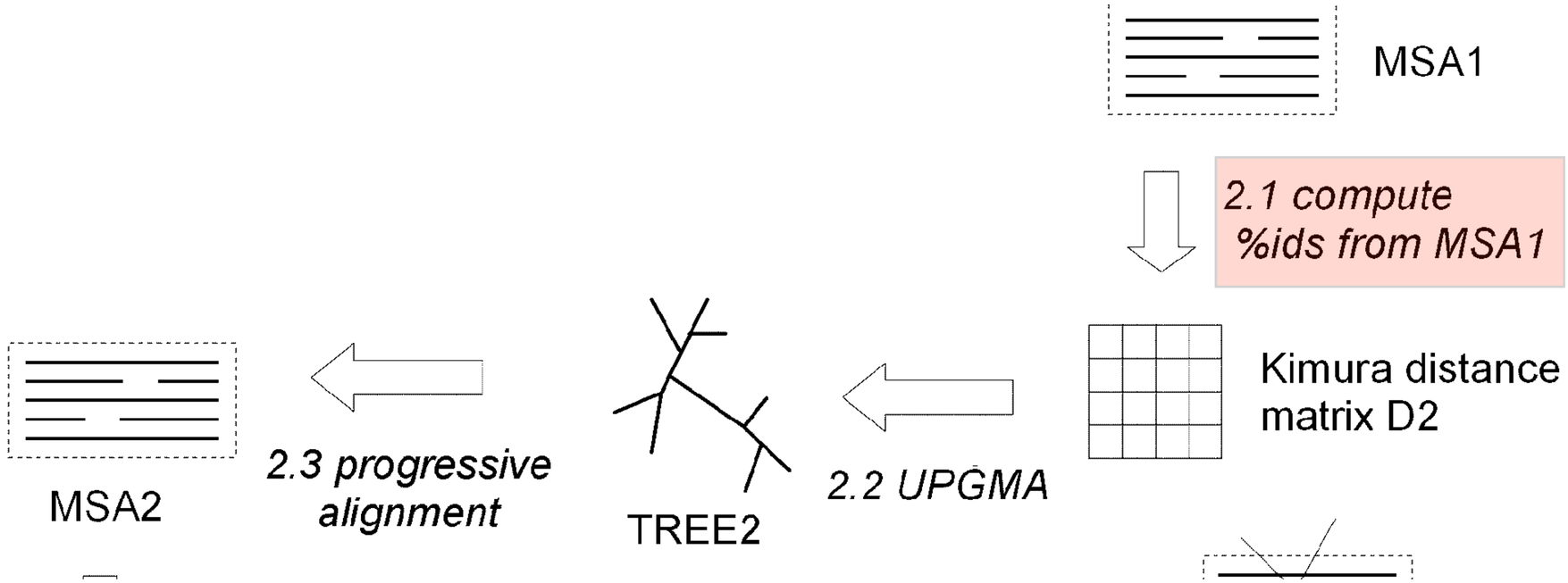
Stage 1: Draft progressive



Avoid time-consuming pairwise alignment

- MUSCLE uses a much faster, but somewhat more approximate, method to compute distances: it counts the number of short sub-sequences (known as *k*- mers, *k*-tuples or words) that two sequences have in common, without constructing an alignment. This is typically around 3,000 times faster that CLUSTALW's method, but the trees will generally be less accurate.

Stage 2: Improved progressive

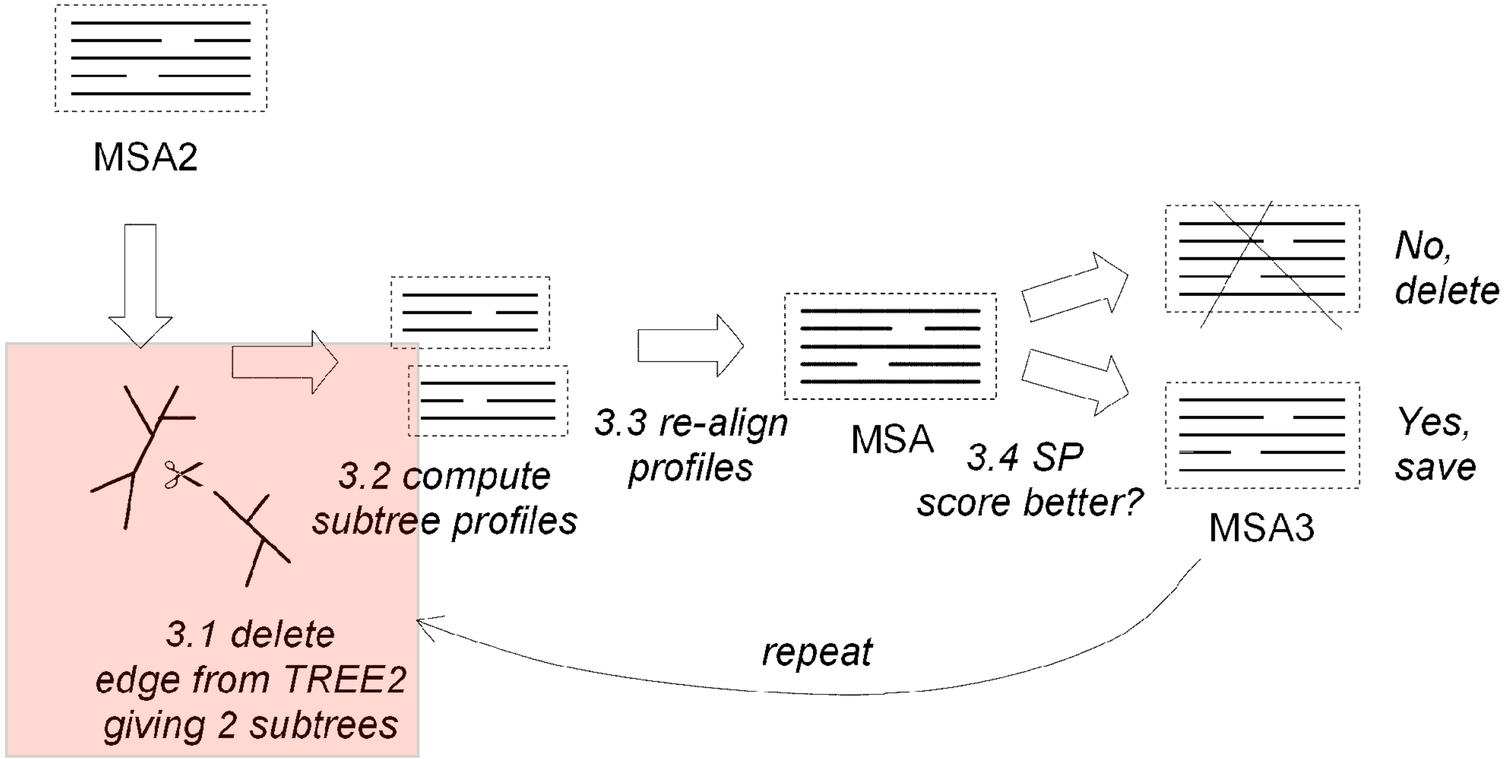


- The MSA from the “draft progressive” alignment is used to compute the pair-wise identities of each pair of sequences, and then to produce a new distance matrix, and a new tree.

- The new tree is compared to the old tree. Subgroups are re-aligned where needed to produce a progressive multiple alignment from the new tree.

- If the two trees are identical, there is nothing to do; if there are no subtrees that agree (very unusual), then the whole progressive alignment procedure must be repeated from scratch.

Stage 3: Refinement



- Two subsets are obtained by *bipartitioning* the tree based on MSA2.
- Profile-profile alignment is conducted using the same pair-wise alignment algorithm as used in the progressive stage.
- Sum-of-pair Score is computed to control the iteration.

Advantages of MUSCLE

- **Speed and Scalability**: MUSCLE is much faster than many traditional MSA tools, especially for large datasets. It uses a progressive alignment method combined with iterative refinement, which allows it to scale efficiently even with large numbers of sequences. K-mer counting

- **High Accuracy**: MUSCLE produces highly accurate alignments, often comparable or superior to other tools like ClustalW. It achieves this by refining the alignment iteratively, improving the accuracy of the initial alignment.Tree refinement and MSA refinement.

- **Fast Convergence**: MUSCLE's algorithm converges quickly, meaning that it reaches an optimal alignment solution more rapidly than many other tools. This makes it efficient for both small and large datasets.

MAFFT

Nucleic Acids Research, 2002, Vol. 30 No. 14 3059–3066

MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform

Kazutaka Katoh, Kazuharu Misawa[1], Kei-ichi Kuma and Takashi Miyata*

Department of Biophysics, Graduate School of Science, Kyoto University, Kyoto 606-8502, Japan and
[1]Institute of Molecular Evolutionary Genetics, Pennsylvania State University, University Park, PA 16802, USA

Received April 8, 2002; Revised and Accepted May 24, 2002

ABSTRACT

A multiple sequence alignment program, MAFFT, has been developed. The CPU time is drastically reduced as compared with existing methods. MAFFT includes two novel techniques. (i) Homologous regions are rapidly identified by the fast Fourier transform (FFT), in which an amino acid sequence is converted to a sequence composed of volume and polarity values of each amino acid residue. (ii) We propose a simplified scoring system that performs well for reducing CPU time and increasing the accuracy of alignments even for sequences having large insertions or extensions as well as distantly related sequences of similar length. Two different heuristics, the progressive method (FFT-NS-2) and the iterative refinement method (FFT-NS-i), are implemented in MAFFT. The performances of FFT-NS-2 and FFT-NS-i were compared with other methods by computer simulations and benchmark tests; the CPU time of FFT-NS-2 is drastically reduced as compared with CLUSTALW with comparable accuracy. FFT-NS-i is over 100 times faster than T-COFFEE, when the number of input sequences exceeds 60, without sacrificing the accuracy.

difficulty, various heuristic methods, including progressive methods (3) and iterative refinement methods (4–6), have been proposed to date. They are mostly based on various combinations of successive two-dimensional DP, which takes CPU time proportional to N^2.

Even if these heuristic methods successfully provide the optimal alignments, there remains the problem of whether the optimal alignment really corresponds to the biologically correct one. The accuracy of resulting alignments is greatly affected by the scoring system. Thompson *et al.* (7) developed a complicated scoring system in their program CLUSTALW, in which gap penalties and other parameters are carefully adjusted according to the features of input sequences, such as sequence divergence, length, local hydropathy and so on. Nevertheless, no existing scoring system is able to process correctly global alignments for various types of problems including large terminal extension of internal insertion (8). Considerable improvements in the accuracy have recently been made in CLUSTALW (7) version 1.8, the most popular alignment program with excellent portability and operativity, and T-COFFEE (9), which provides alignments of the highest accuracy among known methods to date.

On the other hand, few improvements have been made successfully to reduce the CPU time, since the proposal of the progressive method by Feng and Doolittle (3). A high-speed computer program applicable to large-scale problems is becoming more important with the rapid increase in the number of protein and DNA sequences. In order to improve

- MAFFT incorporates a **Fast Fourier Transform (FFT)** algorithm, which speeds up the alignment process by converting the sequences into frequency space. In this space, sequence similarities are computed more quickly. This step dramatically reduces the computational time required for large-scale alignments, particularly when comparing thousands of sequences.

- MAFFT uses a p**rogressive alignment strategy with iterative refinement**, leading to better handling of complex sequences with gaps, insertions, and deletions.

- MAFFT excels at aligning a set of sequences (profile) with another existing alignment, a feature that allows the efficient merging of large datasets. This capability is crucial for incremental or hierarchical alignment tasks that require combining several smaller alignments into one comprehensive alignment.

Clustal omega

Molecular Systems Biology 7; Article number 539; doi:10.1038/msb.2011.75
Citation: ***Molecular Systems Biology*** **7: 539**

www.molecularsystemsbiology.com

molecular systems biology

REPORT

Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega

Fabian Sievers[1,8], Andreas Wilm[2,8], David Dineen[1], Toby J Gibson[3], Kevin Karplus[4], Weizhong Li[5], Rodrigo Lopez[5], Hamish McWilliam[5], Michael Remmert[6], Johannes Söding[6], Julie D Thompson[7] and Desmond G Higgins[1,*]

[1] School of Medicine and Medical Science, UCD Conway Institute of Biomolecular and Biomedical Research, University College Dublin, Dublin, Ireland, [2] Computational and Systems Biology, Genome Institute of Singapore, Singapore, [3] Structural and Computational Biology Unit, European Molecular Biology Laboratory, Heidelberg, Germany, [4] Department of Biomolecular Engineering, University of California, Santa Cruz, CA, USA, [5] EMBL Outstation—European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge, UK, [6] Gene Center Munich, University of Munich (LMU), Muenchen, Germany and [7] Département de Biologie Structurale et Génomique, IGBMC (Institut de Génétique et de Biologie Moléculaire et Cellulaire), CNRS/INSERM/Université de Strasbourg, Illkirch, France
[8] These authors contributed equally to this work
* Corresponding author. UCD Conway Institute of Biomolecular and Biomedical Research, University College Dublin, Belfield, Dublin 4, Ireland. Tel.: +353 1 716 6833; Fax: +353 1 716 6713; E-mail: des.higgins@ucd.ie

Received 23.7.11; accepted 6.9.11

Multiple sequence alignments are fundamental to many sequence analysis methods. Most alignments are computed using the progressive alignment heuristic. These methods are starting to become a bottleneck in some analysis pipelines when faced with data sets of the size of many thousands of sequences. Some methods allow computation of larger data sets while sacrificing quality, and others produce high-quality alignments, but scale badly with the number of sequences. In this paper, we describe a new program called Clustal Omega, which can align virtually any number of protein sequences quickly and that delivers accurate alignments. The accuracy of the package on smaller test cases is similar to that of the high-quality aligners. On larger data sets, Clustal Omega outperforms other packages in terms of execution time and quality. Clustal Omega also has powerful features for adding sequences to and exploiting information in existing alignments, making use of the vast amount of precomputed information in public databases like Pfam.

Molecular Systems Biology **7**: 539; published online 11 October 2011; doi:10.1038/msb.2011.75
Subject Categories: bioinformatics
Keywords: bioinformatics; hidden Markov models; multiple sequence alignment

- Clustal Omega is accurate but also **allows alignments of almost any size** to be produced. It can generate alignments of over 190 000 sequences on a single processor in a few hours.

- Clustal Omega uses a modified version of **mBed** (Blackshields et al, 2010), which has complexity of O(N log N), and which produces guide trees that are just as accurate as those from conventional methods. **mBed works by 'emBedding' each sequence in a space of n dimensions where n is proportional to log N. Each sequence is then replaced by an n element vector, where each element is simply the distance to one of n 'reference sequences.'** These vectors can then be clustered extremely quickly by standard methods such as K-means or UPGMA. In Clustal Omega, the alignments are then computed using the very accurate HHalign package (So¨ding, 2005), which aligns two profile hidden Markov models (Eddy, 1998).

T-coffe

doi:10.1006/jmbi.2000.4042 available online at http://www.idealibrary.com on IDEAL® *J. Mol. Biol.* (2000) **302**, 205–217

JMB

T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment

Cédric Notredame[1,2,3]*, Desmond G. Higgins[4] and Jaap Heringa[1]

[1]*National Institute for Medical Research, The Ridgeway, Mill Hill, London NW7 1AA, UK*

[2]*ISREC, 155, Ch. des Boveresses, CH, 1066 Epalinges/s Lausanne Switzerland*

[3]*Information Genetique et Structurale, CNRS-UMR 1889 31 Ch. Joseph Aiguier 13402 Marseille, France*

[4]*Department of Biochemistry University College, Cork Ireland*

**Corresponding author*

We describe a new method (T-Coffee) for multiple sequence alignment that provides a dramatic improvement in accuracy with a modest sacrifice in speed as compared to the most commonly used alternatives. The method is broadly based on the popular progressive approach to multiple alignment but avoids the most serious pitfalls caused by the greedy nature of this algorithm. With T-Coffee we pre-process a data set of all pair-wise alignments between the sequences. This provides us with a library of alignment information that can be used to guide the progressive alignment. Intermediate alignments are then based not only on the sequences to be aligned next but also on how all of the sequences align with each other. This alignment information can be derived from heterogeneous sources such as a mixture of alignment programs and/or structure superposition. Here, we illustrate the power of the approach by using a combination of local and global pair-wise alignments to generate the library. The resulting alignments are significantly more reliable, as determined by comparison with a set of 141 test cases, than any of the popular alternatives that we tried. The improvement, especially clear with the more difficult test cases, is always visible, regardless of the phylogenetic spread of the sequences in the tests.

Keywords: pair-wise alignment; progressive alignment; local alignment; global alignment; multiple sequence alignment

- One of the main reasons T-Coffee is highly accurate is its **consistency-based scoring method**. Unlike traditional MSA algorithms, which align sequences directly, T-Coffee checks the "consistency" of the alignments with respect to multiple pairwise comparisons.

- T-Coffee builds a **library of pairwise alignments** (including global and local alignments), which serves as a guide during the multiple sequence alignment process. Each alignment is checked against this library to ensure that each alignment step agrees with the majority of pairwise alignments in the set. This consistency improves the reliability of the final alignment, reducing the impact of errors that might arise from aligning distant sequences directly.

T-coffee's consistency checking

b)Primary Library

```
SeqA GARFIELD THE LAST FAT CAT
SeqB GARFIELD THE FAST CAT ---
```
Prim. Weight = 88

```
SeqA GARFIELD THE LAST FA-T CAT
SeqC GARFIELD THE VERY FAST CAT
```
Prim. Weight = 77

```
SeqA GARFIELD THE LAST FAT CAT
SeqD -------  THE ---- FAT CAT
```
Prim. Weight =100

```
SeqB GARFIELD THE ---- FAST CAT
SeqC GARFIELD THE VERY FAST CAT
```
Prim Weight = 100

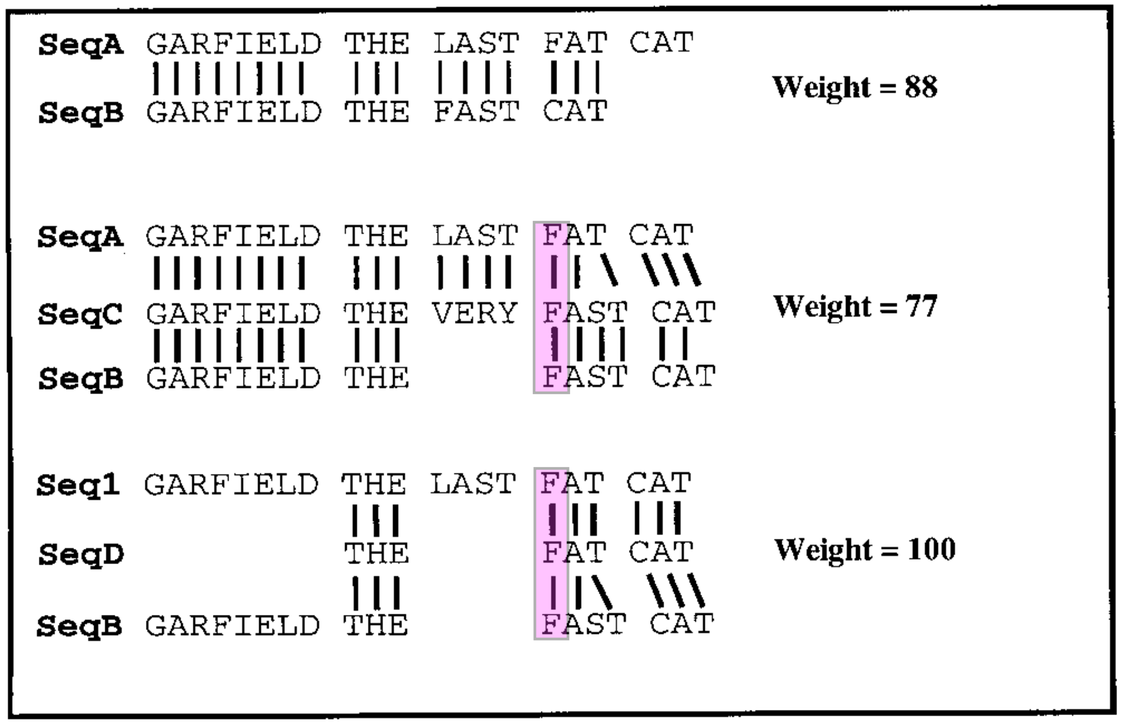
```
SeqB GARFIELD THE FAST CAT
SeqD -------  THE FA-T CAT
```
Prim. Weight = 100

```
SeqC GARFIELD THE VERY FAST CAT
SeqD -------  THE ---- FA-T CAT
```
Prim. Weight = 100

- Primary weight between a pair of sequences is the sequence identity between them.
- Sequence identity = Matched residues/aligned residues
- Each pair of residues will be assigned the Primary weight.

T-coffee's consistency checking

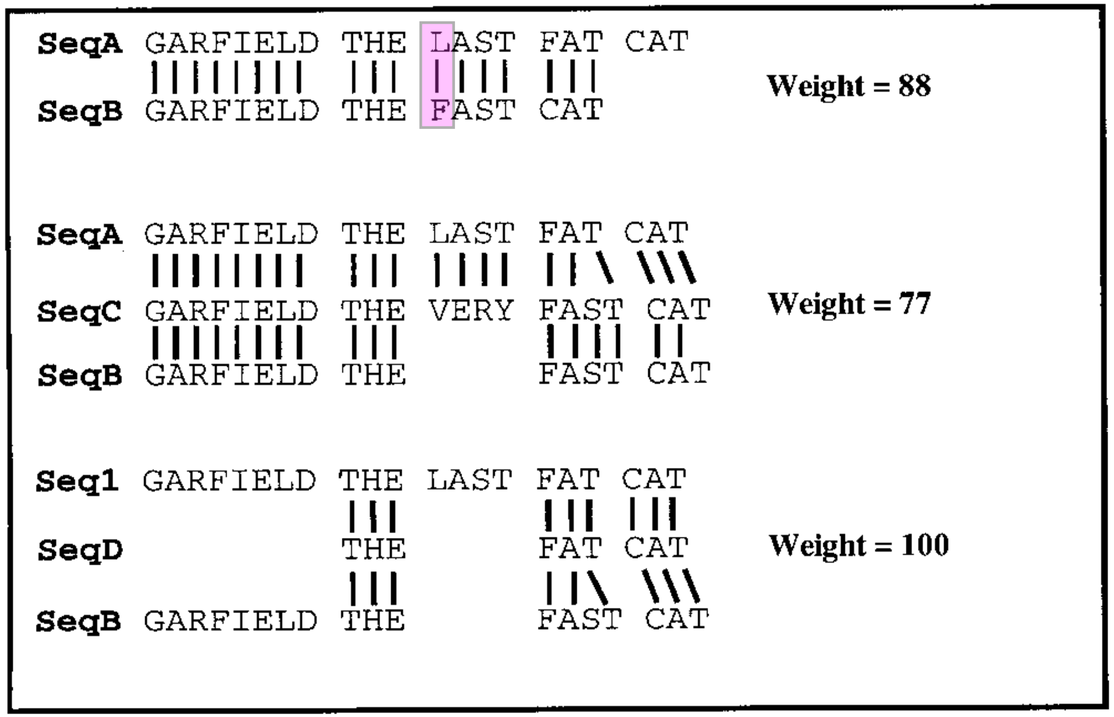
c)Extended Library for seq1 and seq2



- For A-B: F(AT):F(AST)
- In A-B: W(F:F) = 0
- In A-C-B: W(F:F) = min(77(w(A,C), 100(w(C,B))) = 77
- In A-D-B: W(F:F) = min(100(w(A,D), 100(w(D,B))) = 100
- Final A-B: F(AT):F(AST) = 77 + 100 = 177

T-coffee's consistency checking

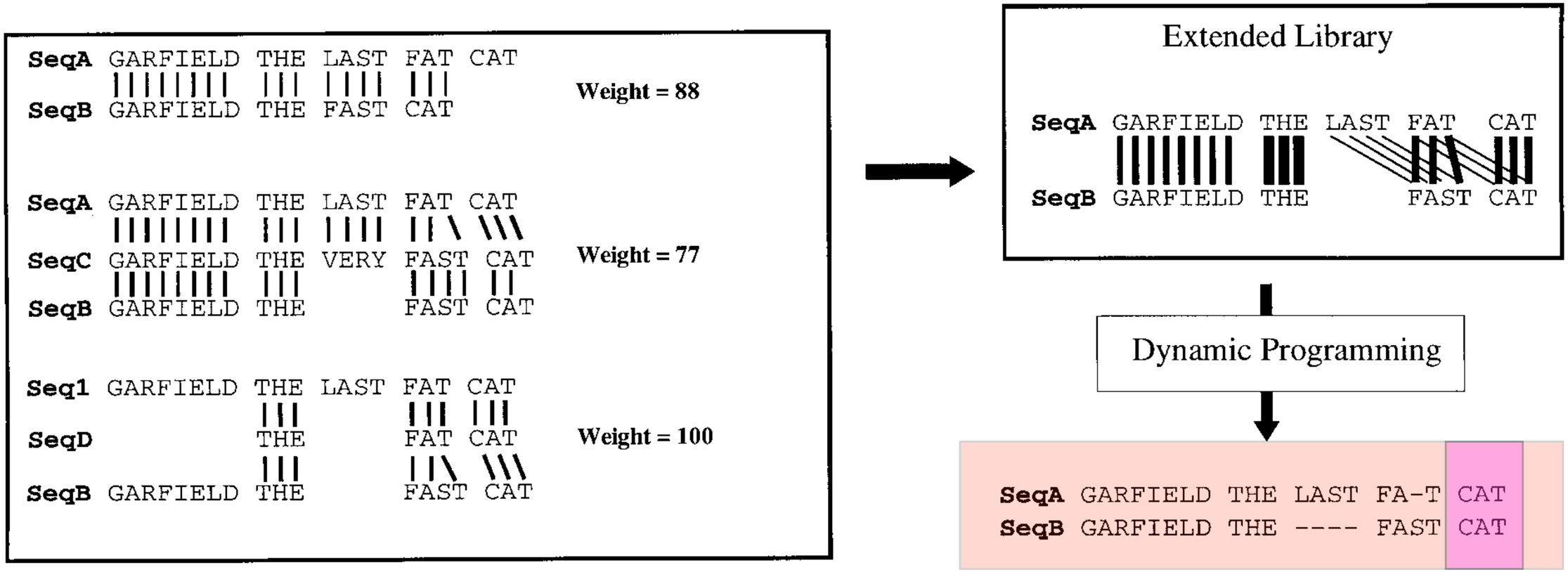
c)Extended Library for seq1 and seq2



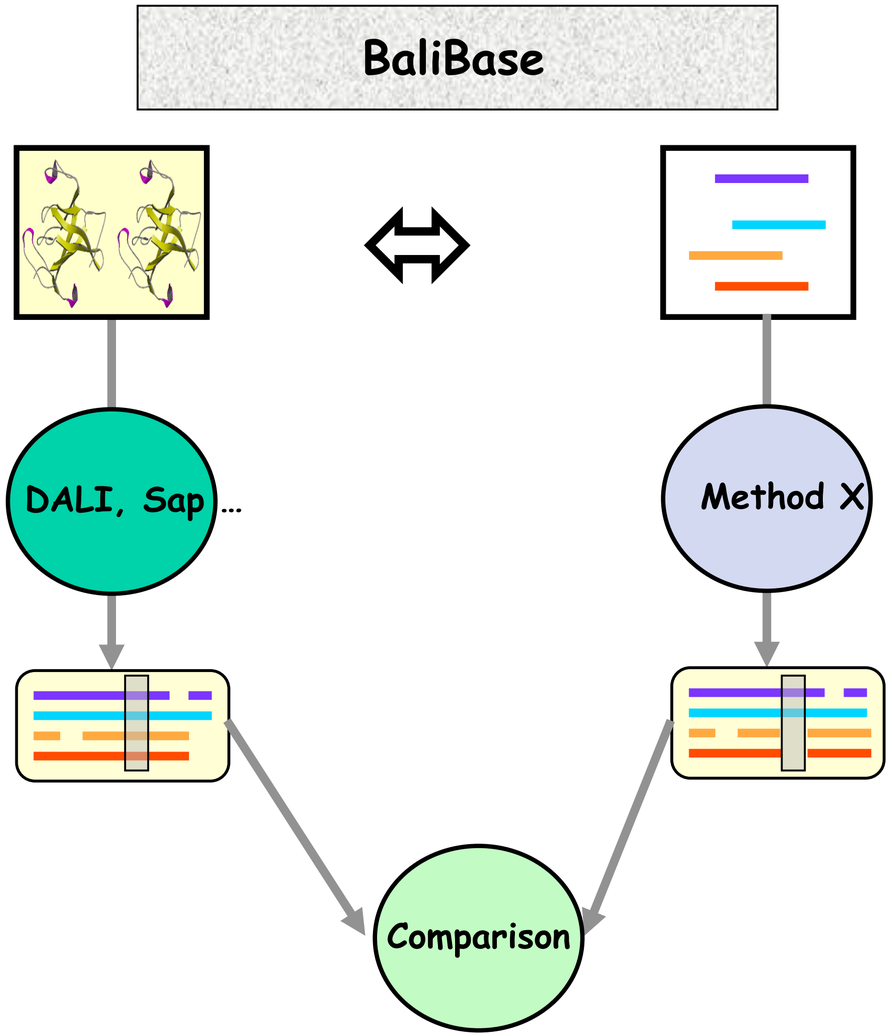
- For A-B: L(AST):F(AST)
- In A-B: W(L:F) = 88
- In A-C-B: W(L:F) = min(0,0) = 0
- In A-D-B: W(L:F) = min(0,0) = 0
- Final A-B: L(AST):F(AST) = 88 + 0 = 88

T-coffee's consistency checking

c)Extended Library for seq1 and seq2



Direct assessment of the quality of an MSA



Structure alignment method

Sequence alignment method

Indirect assessment of the quality of a MSA

- Quality measures of MSA
  - Number of matches (multiple longest common subsequence score)
  - Entropy score
  - Sum of pairs (SP-Score)

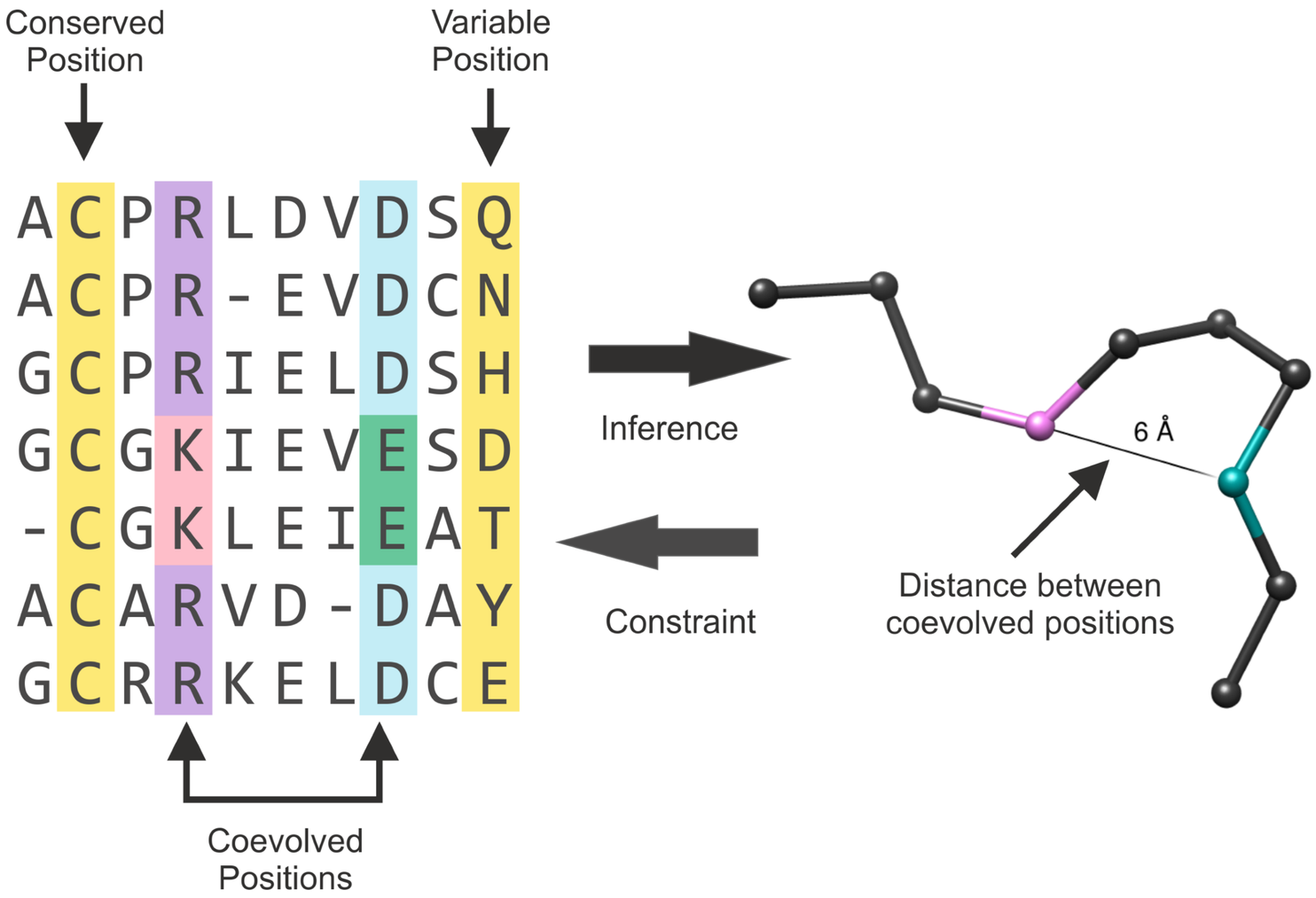
Entropy score

The entropy $\mathcal{H}(P)$ of a probability distribution $P = (p_1, \ldots, p_n)$ is defined by

$$\mathcal{H}(P) = \mathbf{E}(-\log P) = -\sum_{i=1}^{n} p_i \log p_i. \tag{B.1}$$

The units used to measure entropy depend on the base used for the logarithms. When the base is 2, the entropy is measured in bits. The entropy measures the prior uncertainty in the outcome of a random experiment described by P, or the information gained when the outcome is observed. It is

Entropy score

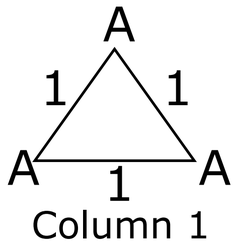


Sum of Pairs Score (SP-Score)

a_1 ATG–C–AAT
· A–G–CATAT
a_k ATCCCATTT

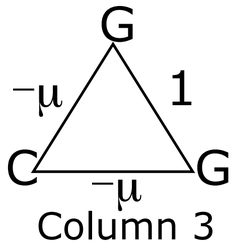
To calculate each column:

$$s'(a_1 \ldots a_k) = \sum_{i,j} s^*(a_i, a_j) \longleftarrow \binom{n}{2} \text{ Pairs of Sequences}$$



Score=3

Column 1



Column 3

Comparison of different MSA tools

Tool	Speed	Accuracy	Best for	Limitation
MAFFT	Very fast	High	Large datasets, complex alignments	Slightly less accurate for divergent sequences
MUSCLE	Fast	High	Medium-sized datasets, general-purpose	Slower than MAFFT, less scalable
Clustal Omega	Very fast	Moderate to High	Large datasets, fast scalability	Less accurate for highly divergent sequences
T-coffee	Slow	Very High	Highly accurate alignments, complex regions	Slower, not suitable for large datasets
ProbCons	Very slow	Very high	High accuracy, consistency	Very slow, impractical for large datasets

Summary for MSA tools

- Multiple Sequence Alignment (MSA) tools have evolved from early methods like ClustalW to modern algorithms such as MAFFT, MUSCLE, and Clustal Omega, which prioritize speed, accuracy, and scalability for large datasets.

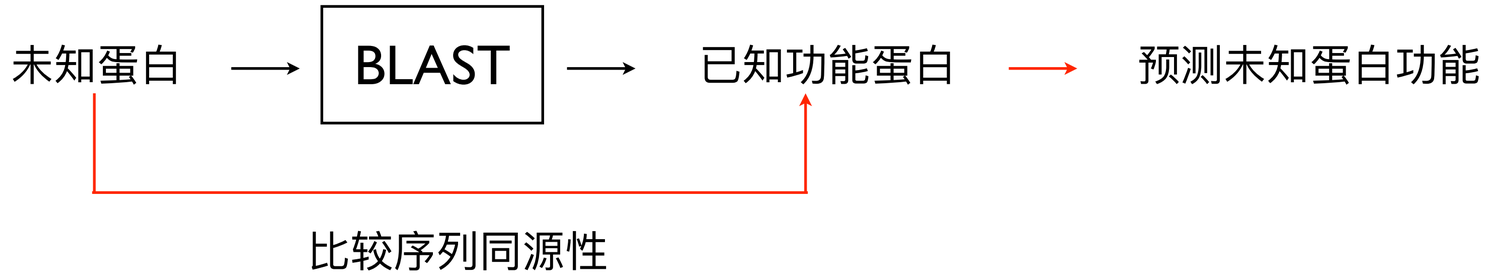
- These tools are widely applied in phylogenetics, functional genomics, protein structure prediction, and variant analysis.

- Future developments will likely focus on improving algorithm efficiency, integrating machine learning, and creating cloud-based platforms for real-time analysis.

Sequence database search

Sequence database search

- Given a query sequence
- Goal: identify all sequences in the database that are homologous to the query sequence, and if the homologous sequence's function is known, infer the function of the query sequence.

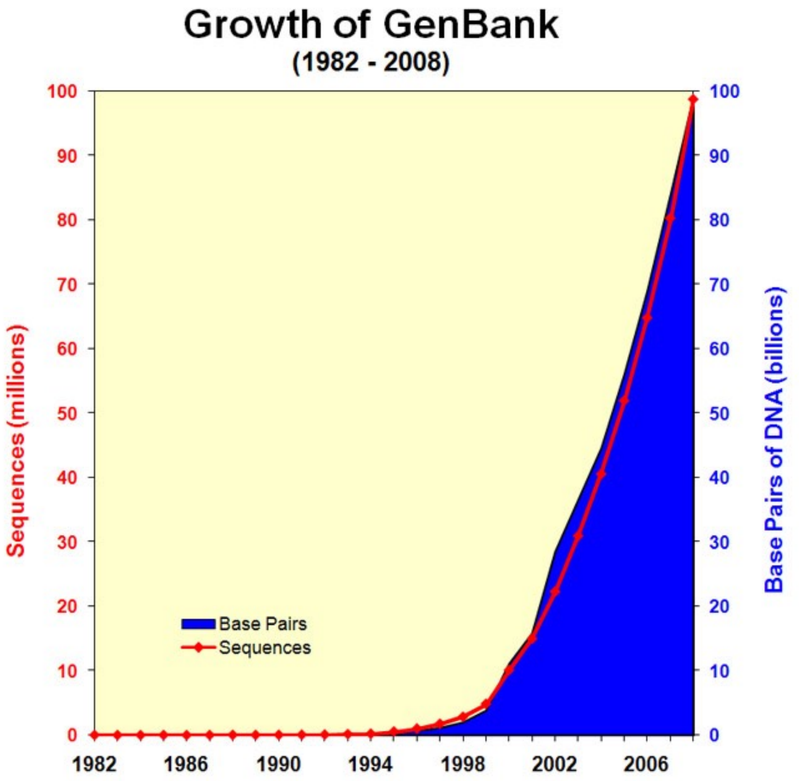


Sequence database search

- Approach: compare the query sequence with each of the sequence in the database by pairwise sequence comparison, and determine the homologous sequences whose alignment scores are above a certain threshold.

- Problem: The time complexity of dynamic programming is *O(nm)* for a pair of sequences. Given a total number of *k* sequences in the database, the complexity would be *O(knm).* However, k is increasing exponentially over the years.

- Dynamic programming for all pairs of sequences is computationally impossible.



GenBank has grown exponentially and currently doubles in size approximately every 2 years

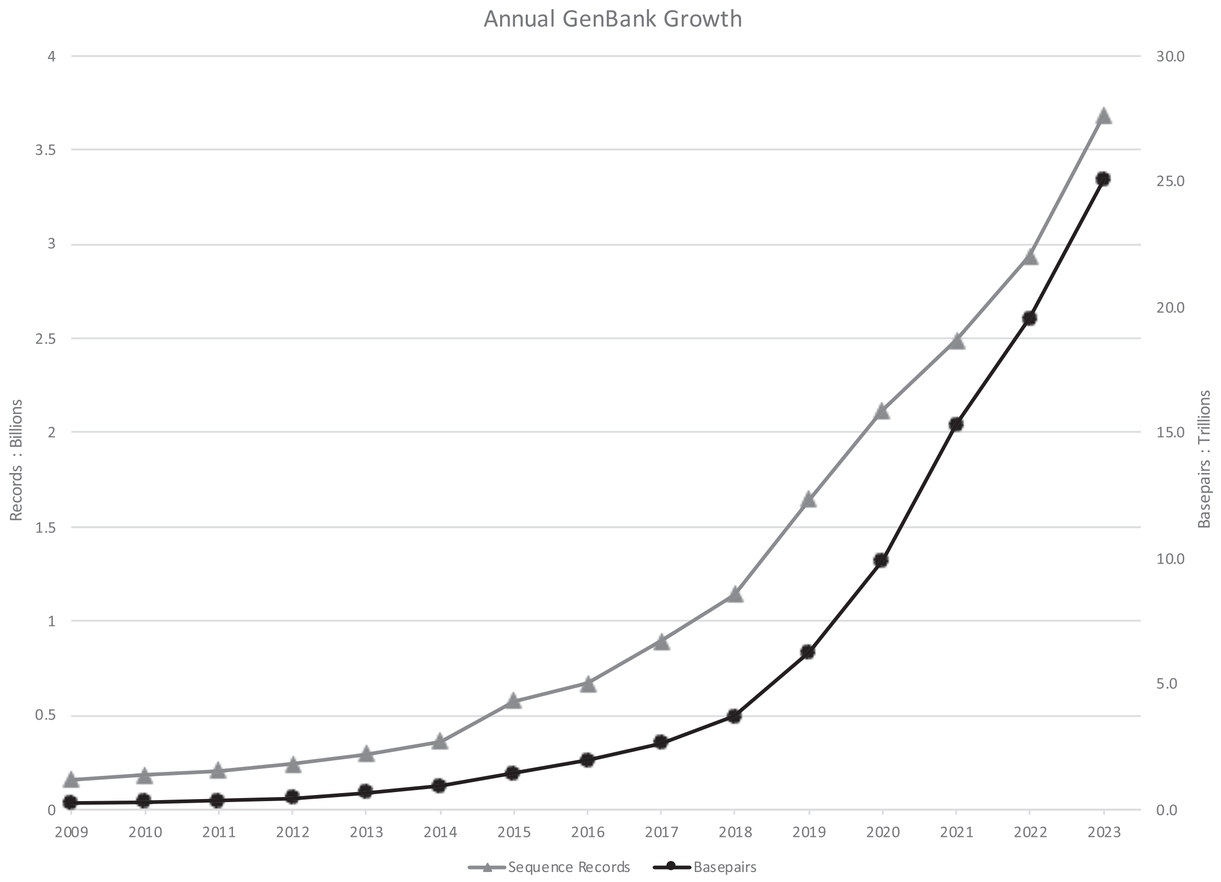
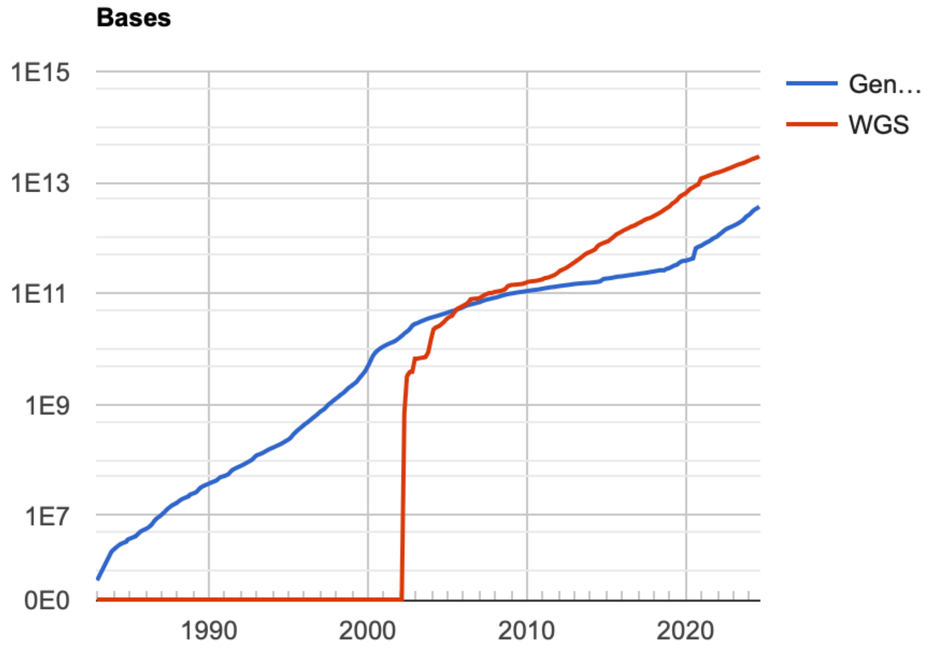
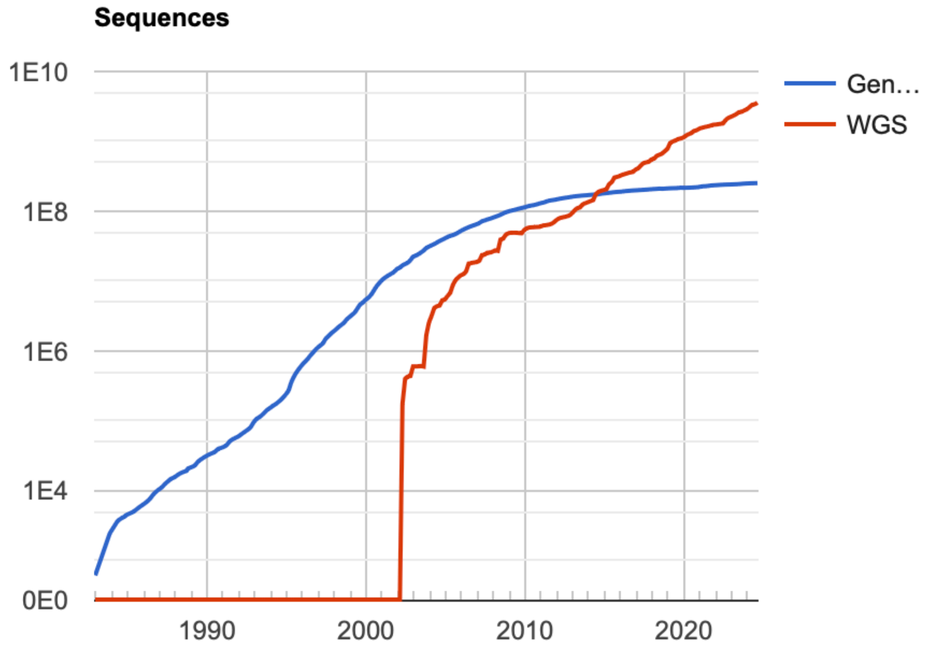


Figure 1. Growth of GenBank recorded in both base pairs (triangles) and the number of sequence records (circles). Each point represents the GenBank release in August of each year, starting with release 173 (August 2009).

GenBank and WGS Statistics





The heuristic approach to solve the database search problem

- A heuristic technique, often called simply a heuristic, is any approach to problem solving or self-discovery that employs a practical method, not guaranteed to be optimal, perfect, or rational, but instead sufficient for reaching an immediate goal. Where finding an optimal solution is impossible or impractical, heuristic methods can be used to speed up the process of finding a satisfactory solution.

- Basic assumption for sequence database search: most sequences in the database are not homologous to the query sequence.

- The heuristic idea: avoiding comparison with most sequences.

The heuristic approach to solve the database search problem

- Knowledge about pairwise sequence comparison between homologous sequences: two homologous sequences must have some common subsequences of absolute identity (short lengths of exact matches).

- No need to compare with most sequences

  - First, identify very short exact matches.

  - Next, the best hits from the first step are extended to longer regions of similarity.

  - Finally, the best hits are optimized.

BLAST

- BLAST, the Basic Local Alignment Search Tool (Altschul et al., 1990), is perhaps the most widely used bioinformatics tool ever written. It is an alignment heuristic that determines “local alignments” between a *query* and a *database*. It uses an approximation of the Smith-Waterman algorithm.

- BLAST consists of two components: a search algorithm and computation of the statistical significance of the alignments.

BLAST

Step 1.A: Compile a List of Words

§ Given a word length ***w***...

- Word length is analogous to window length

$$w = 3$$

VHR = first word

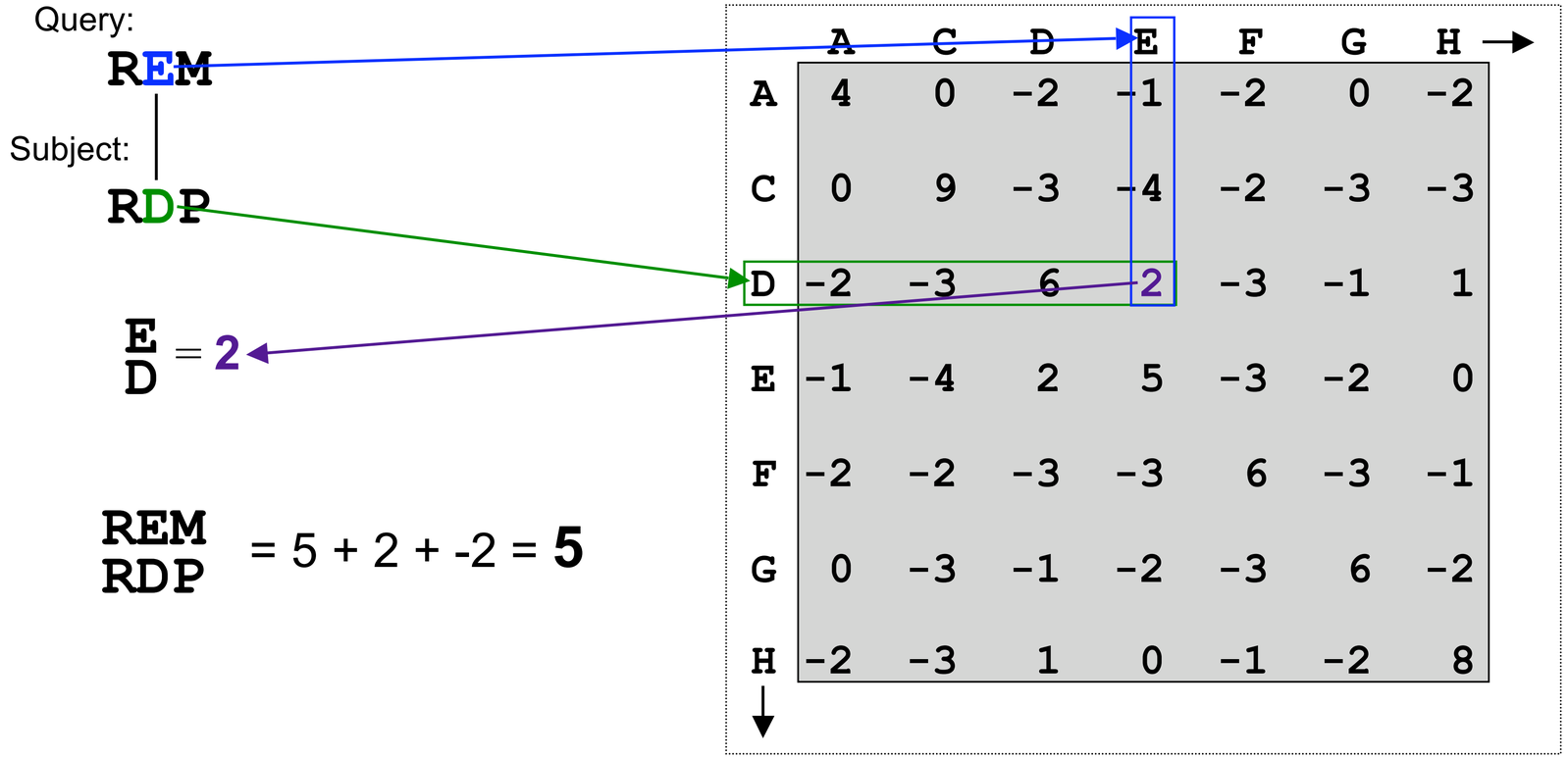
HRE = second word

REM = third word

BLAST

Step 1.B: Choose a Scoring Matrix to Calculate Word Scores

§ Choose an appropriate substitution matrix to calculate word scores



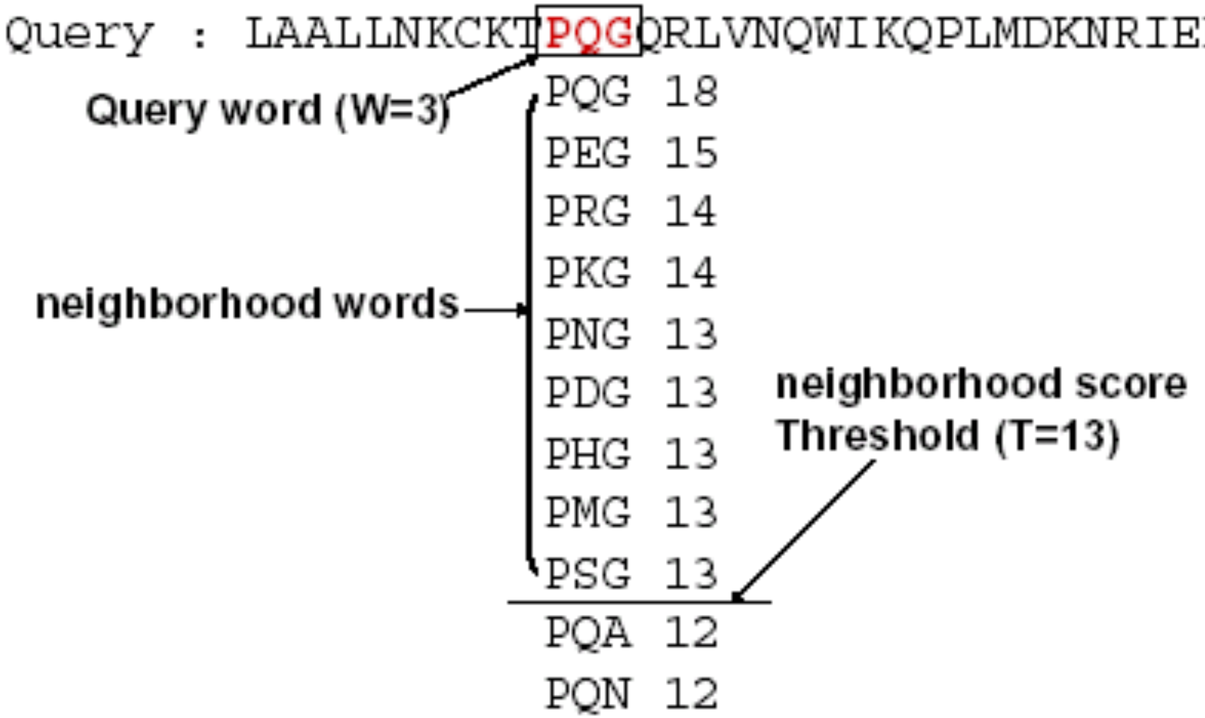
BLAST

Step 1.C: Compile List of High-Scoring Words

§ Calculate score of all possible variations for each word

§ Select word variations that have score higher than ***T***

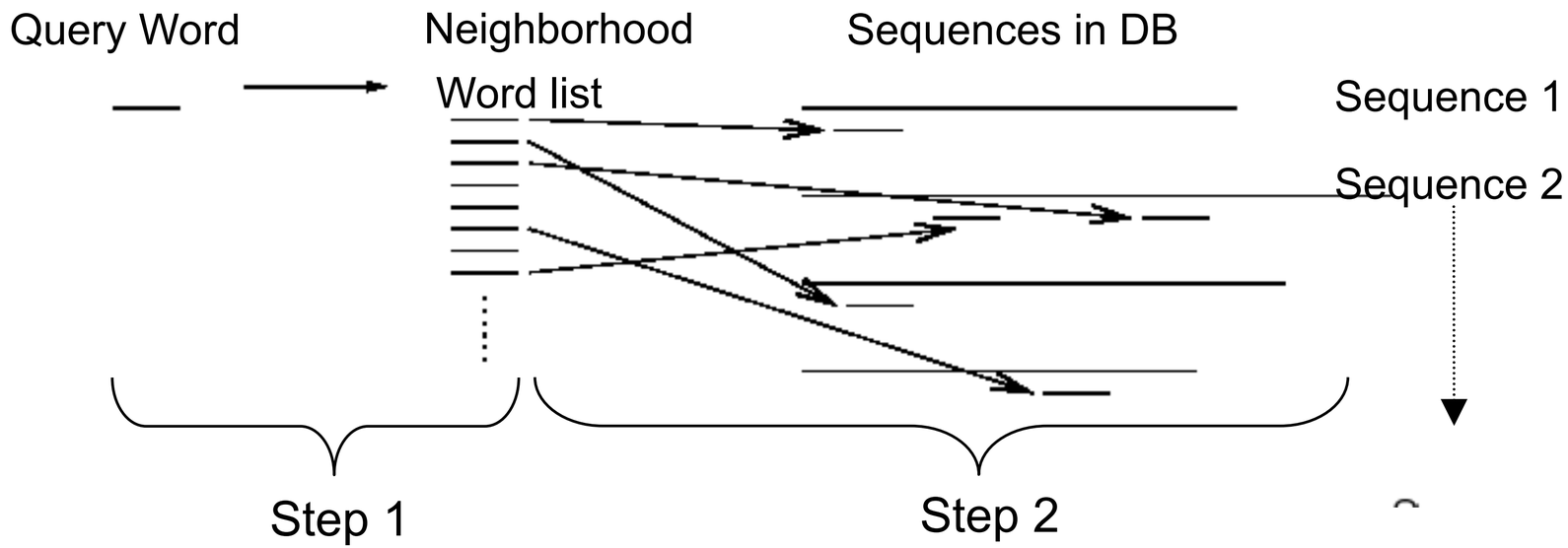
- ***T*** is chosen empirically



BLAST

- Step 2: Scanning DB

  For each words list, identify all exact matches with DB sequences



The purpose of Step 1 and 2 is as same as FASTA

BLAST

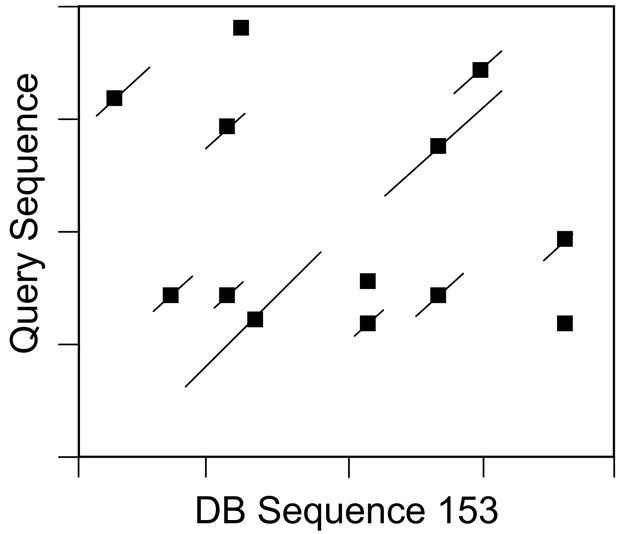
Step 3: Extend Hits to Differentiate Random Hits from Meaningful Hits

§ Extend hits by summing residue pairs from both sides of the word boundary

§ Extension stops when score drops below a threshold ($\mathbf{X_U}$) of the best score yet observed

§ Original BLAST extended each hit to find ungapped segments

§ All extended hits above the minimum score ***S*** are reported



Starting Score = 15

Query:	V	H	R	E	M	H	A	R	T	S	P	L	R	P	L
	-2	-2	2				1	1	0	0	7	2	-3	-4	-4
DB 153:	K	W	K	D	M	H	S	Q	A	D	P	I	I	F	G

BLAST

Step 3.B: Gapped BLAST -- Triggering Extensions

For Gapped BLAST:

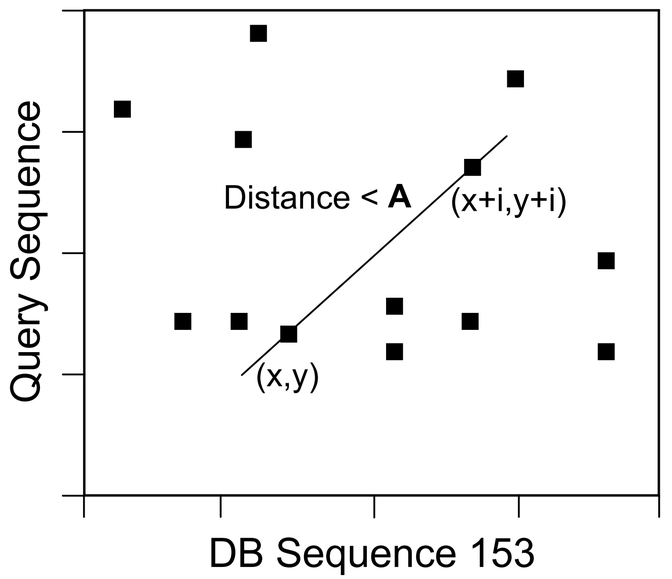
§ Extensions triggered only by two or more hits on the same diagonal

§ Hits must also be less than distance ***A*** from each other to trigger extensions

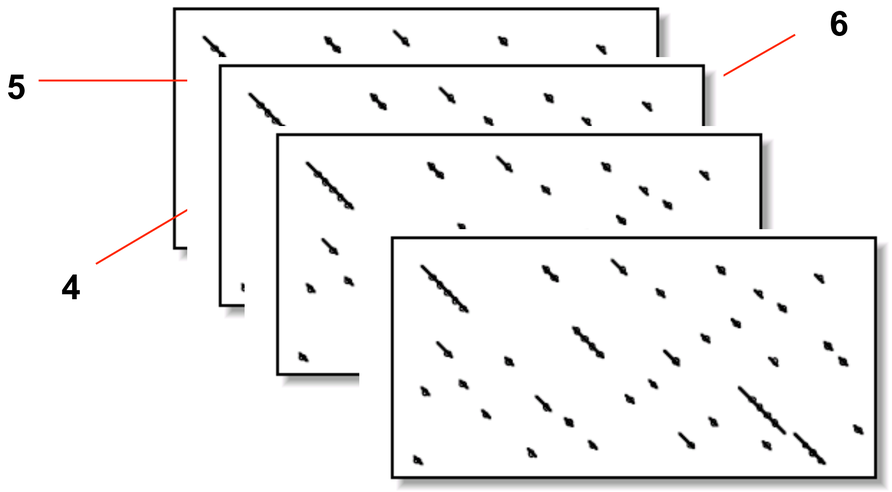
§ Serves to reduce the number of extensions

§ Typically the Dynamic Programming sequence alignment method is used to find gapped alignment

§ Gapped BLAST is more sensitive and selective than the original, ungapped BLAST



The BLAST statistics

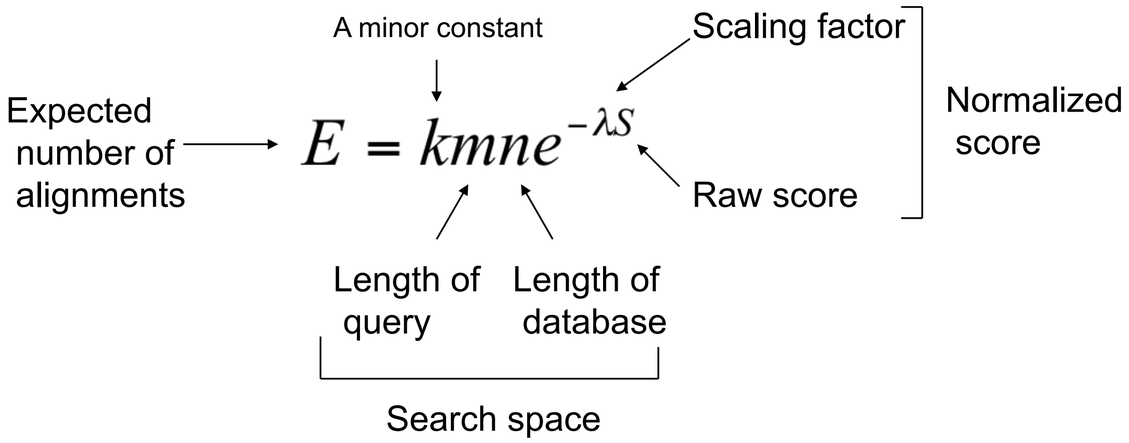


- How different is the result from a random match, or a match between unrelated sequences? Given a set of sequences *not related* to the query (or a set of random sequences), what is the probability of finding a match with the same alignment score by chance?

- Given an alignment of score X, how many sequences do we expect to find by chance?

The BLAST statistics

The Karlin-Altschul Equation



- The e-value indicates the number of alignments one expects to find with a score equal or greater to the given one in a search against a random database.
- Large e-value (5 or 10) indicates that the alignment is probably by chance.
- E-values of 0.1 or 0.05 are typical cutoff values for database search.

Raw score vs. bit score

$$S' = \frac{\lambda S - \ln K}{\ln 2}$$

$$E = mn\,2^{-S'}$$

- Raw scores: the sum of the scores of the maximal-scoring segment pairs (MSPs) that makes up the alignment. Because of differences between scoring matrices raw scores are not directly comparableLarge e-value (5 or 10) indicates that the alignment is probably by chance.

- Bit scores: these are raw scores that have been converted from the log base of the scoring matrix that creates the alignment to log base 2. This rescaling allows bit scores to be comparable.

E-value vs. P-value

$$P = 1 - e^{-E}$$

$$E = -\ln(1 - P)$$

Note that E ~= P if either value < $1e^{-5}$

- E = 5, E = 10 versus P = 0.993 and P = 0.99995
- When E < 0.01 P-values and E-values are nearly identical

- **E-value** is used in BLAST because it directly accounts for the size of the database being searched. As the database size increases, the likelihood of finding a match purely by chance increases, and the E-value reflects this.
- **P-value**, on the other hand, gives a measure of significance without adjusting for database size. In sequence searches, the size of the database significantly affects the probability of random matches, so the E-value is preferred.
- The **E-value** is conceptually similar to the P-value but more practical for large-scale database searches since it estimates the number of expected hits by chance, allowing users to filter results accordingly.

Next lecture

- Database searches
- Phylogenetic tree construction