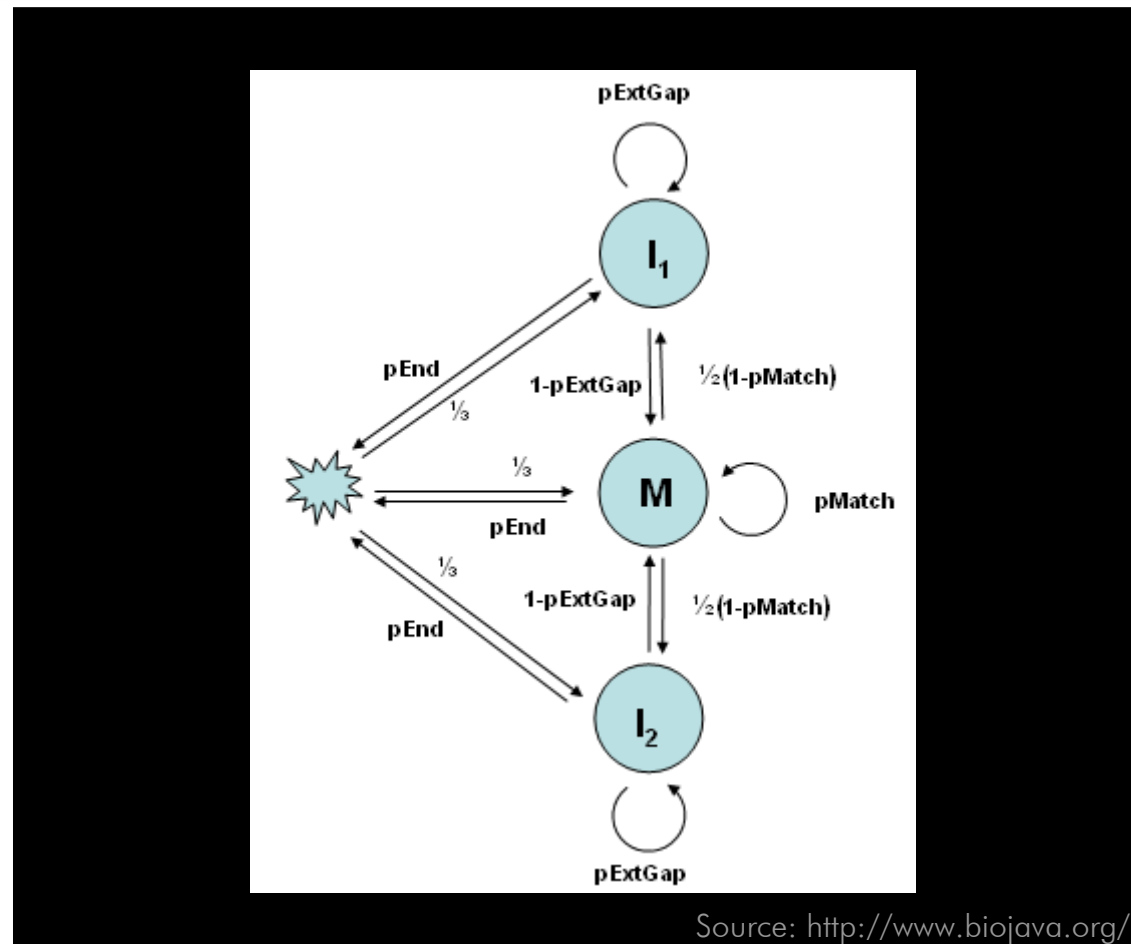


Pairwise Alignment



Sequence Comparison

1. To observe patterns of *conservation* (or *variability*)
2. To find the *common motifs* present in both sequences
3. To assess whether it is likely that two sequences *evolved* from the same sequence
4. To find out which sequences from the database are *similar* to the sequence at hand

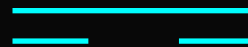
Sequence Alignment(s)

There are three different types of sequence alignment:

1. *Global* alignment



2. *Local* alignment



3. *Multiple* sequence alignment



Sequence Alignment Framework

- ✓ Sequence alignment is the establishment of **residue-to-residue correspondence** between two or more sequences such that the **order** of residues in each sequence **is preserved**.
- ✓ A gap, which indicates a residue-to-nothing match, may be introduced in either sequence.
- ✓ A **gap-to-gap** match is meaningless and is **not allowed**.

Scoring Scheme

1. Given two sequences we need a number to associate with each possible alignment (“*goodness*” of alignment).
2. The *scoring scheme* is a set of *rules* which assigns the alignment score to any given alignment of two sequences:
 1. The scoring scheme is *residue-based*: it consists of residue *substitution scores* (i.e. score for each possible residue alignment), plus *penalties for gaps*.
 2. The alignment *score* is the *sum* of substitution scores and gap penalties.

(DNA) Substitution Matrix *... a naïve approach*

A concise way to express the residue substitution costs can be achieved with a $N \times N$ matrix (N is 4 for DNA and 20 for proteins).

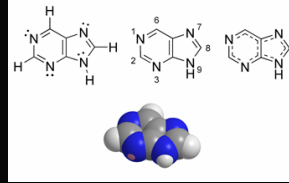
The substitution matrix for the simple scoring scheme:

	C	T	A	G
C	1	-1	-1	-1
T	-1	1	-1	-1
A	-1	-1	1	-1
G	-1	-1	-1	1

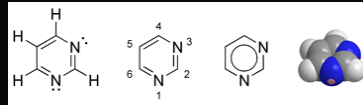
(DNA) Substitution Matrix

... a more sophisticated approach

A, G are *purines*



T, C are *pyrimidines*



From evolutionary standpoint *purine/pyrimidine* mutations are less likely to occur compared to *purine/purine* (or *pyrimidine/pyrimidine*) mutations.

A substitution matrix could be:

	C	T	A	G
C	2	1	-1	-1
T	1	2	-1	-1
A	-1	-1	2	1
G	-1	-1	1	2

Protein Substitution Matrices

- ✓ Protein substitution matrices are *more complex* than DNA scoring matrices.
- ✓ Proteins are composed of 20 amino acids, and *physico-chemical* properties of individual amino acids vary considerably.
- ✓ A protein substitution matrix can be based on any property of amino acids: *size, polarity, charge, hydrophobicity*.
- ✓ In practice the most important are *evolutionary* substitution matrices.

Evolutionary substitution matrices

- ✓ **PAM** ("point accepted mutation") family: PAM250, PAM120, etc.
- ✓ **BLOSUM** ("Blocks substitution matrix") family: BLOSUM62, BLOSUM50, etc.
- ✓ The substitution scores of both PAM and BLOSUM matrices are derived from ... the analysis of *known alignments* of closely related proteins.
- ✓ The BLOSUM matrices are *newer* and considered more realistic.

Substitution matrices

... a probabilistic approach

1. Consider a pair of sequences, x and y , of lengths n and m , respectively. Let x_i be the i th symbol in x and y_j the j th symbol of y .

Substitution matrices

... a probabilistic approach

1. Consider a pair of sequences, x and y , of lengths n and m , respectively. Let x_i be the i th symbol in x and y_j the j th symbol of y .
2. Given a pair of aligned sequences we want to assign a score to the alignment that gives a measure of the relative likelihood that the sequences are related as opposed to being unrelated.

Substitution matrices ***... a probabilistic approach***

1. Consider a pair of **sequences**, x and y , of **lengths** n and m , respectively. Let x_i be the i th symbol in x and y_j the j th **symbol** of y .
2. Given a **pair of aligned sequences** we want to assign a **score** to the alignment that gives a measure of the **relative likelihood** that the sequences are **related** as opposed to being **unrelated**.
3. The **unrelated** or **random** model R assumes that letter a occurs **independently** with some frequency q_a and hence the **probability of the two sequences** is just the product of the probabilities of each amino acid:

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

Substitution matrices

... a probabilistic approach

1. Consider a pair of sequences, x and y , of lengths n and m , respectively. Let x_i be the i th symbol in x and y_j the j th symbol of y .
2. Given a pair of aligned sequences we want to assign a score to the alignment that gives a measure of the relative likelihood that the sequences are related as opposed to being unrelated.
3. The unrelated or random model R assumes that letter a occurs independently with some frequency q_a and hence the probability of the two sequences is just the product of the probabilities of each amino acid:

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

4. In the alternative match model M aligned pairs of residues occur with a joint probability p_{ab} . This value p_{ab} can be thought of as the probability that the residues a and b have each been derived from some unknown original residue c in their common ancestor. So the probability of the whole alignment is:

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

Substitution matrices *... a probabilistic approach*

5. The ratio of those two likelihoods is known as the *odds ratio*:

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

Substitution matrices ... a probabilistic approach

5. The ratio of those two likelihoods is known as the *odds ratio*:

$$\frac{P(x, y | M)}{P(x, y | R)} = \frac{\prod_i P_{x_i y_i}}{\prod_i q_{x_i} \prod_j q_{y_j}} = \prod_i \frac{P_{x_i y_i}}{q_{x_i} q_{y_i}}$$

6. In order to arrive at an additive scoring system, we take the logarithm of this ratio known as the *log-odds ratio*:

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

Substitution matrices ... a probabilistic approach

5. The ratio of those two likelihoods is known as the *odds ratio*:

$$\frac{P(x, y | M)}{P(x, y | R)} = \frac{\prod_i P_{x_i y_i}}{\prod_i q_{x_i} \prod_j q_{y_j}} = \prod_i \frac{P_{x_i y_i}}{q_{x_i} q_{y_i}}$$

6. In order to arrive at an additive scoring system, we take the logarithm of this ratio known as the *log-odds ratio*:

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

where

$$s(a, b) = \log \left(\frac{p_{ab}}{q_a q_b} \right) \quad (2.3)$$

is the *log likelihood ratio* of the residue pair (a, b) occurring as an *aligned* pair as opposed to an *unaligned* pair. The $s(a, b)$ scores can be arranged in a *matrix*, known as *score matrix* or *substitution matrix*; for proteins such a matrix is a 20 x 20 matrix.

Deriving score parameters from alignment data

How do we determine the **components** of the **scoring model**, the substitution and gap scores?

✓ A **simple** and obvious approach would be to **count** the frequencies of aligned residue pairs and of gaps in confirmed alignments, and **set** the **probabilities** to the **normalised frequencies (MLE)**.

Deriving score parameters from alignment data

How do we determine the **components** of the **scoring model**, the substitution and gap scores?

✓ A **simple** and obvious approach would be to **count** the frequencies of aligned residue pairs and of gaps in confirmed alignments, and **set** the **probabilities** to the **normalised frequencies (MLE)**.


✓ However different pairs of sequences have **diverged by different amounts**. When two sequences have diverged from a common ancestor **very recently** we expect **many** of their residues to be **identical**. The probability p_{ab} for $a \neq b$ should be small and hence $s(a,b)$ should be strongly negative unless $a=b$.

Deriving score parameters from alignment data

How do we determine the **components** of the **scoring model**, the substitution and gap scores?

✓ A **simple** and obvious approach would be to **count** the frequencies of aligned residue pairs and of gaps in confirmed alignments, and **set** the **probabilities** to the **normalised frequencies (MLE)**.

✓ However different pairs of sequences have **diverged by different amounts**. When two sequences have diverged from a common ancestor **very recently** we expect **many** of their residues to be **identical**. The probability p_{ab} for $a \neq b$ should be small and hence $s(a,b)$ should be strongly negative unless $a=b$.


$$s(a,b) = \log \left(\frac{p_{ab}}{q_a q_b} \right) \quad (2.3)$$

Deriving score parameters from alignment data

How do we determine the components of the scoring model, the substitution and gap scores?

✓ A simple and obvious approach would be to count the frequencies of aligned residue pairs and of gaps in confirmed alignments, and set the probabilities to the normalised frequencies (MLE).

✓ However different pairs of sequences have diverged by different amounts. When two sequences have diverged from a common ancestor very recently we expect many of their residues to be identical. The probability p_{ab} for $a \neq b$ should be small and hence $s(a,b)$ should be strongly negative unless $a=b$.

✓ At the other extreme when long time has passed since two sequences diverged we expect p_{ab} to tend to the background frequency $q_a q_b$ so $s(a,b)$ should be close to zero for all a, b .

Deriving score parameters from alignment data

How do we determine the **components** of the **scoring model**, the substitution and gap scores?

✓ A **simple** and obvious approach would be to **count** the frequencies of aligned residue pairs and of gaps in confirmed alignments, and **set** the **probabilities** to the **normalised frequencies (MLE)**.

✓ However different pairs of sequences have **diverged by different amounts**. When two sequences have diverged from a common ancestor **very recently** we expect **many** of their residues to be **identical**. The probability p_{ab} for $a \neq b$ should be small and hence $s(a,b)$ should be strongly negative unless $a=b$.

✓ At the other extreme **when long time** has passed since two sequences diverged we expect p_{ab} to tend to the background frequency $q_a q_b$ so $s(a,b)$ should be close to zero for all a, b .

✓ We should therefore use **scores** that are matched to the **expected divergence** of the sequences we want to compare.

Sequences are not what we see ...

A
A
T
T
C
G
A
G
A
C

Sequences are not what we see ...

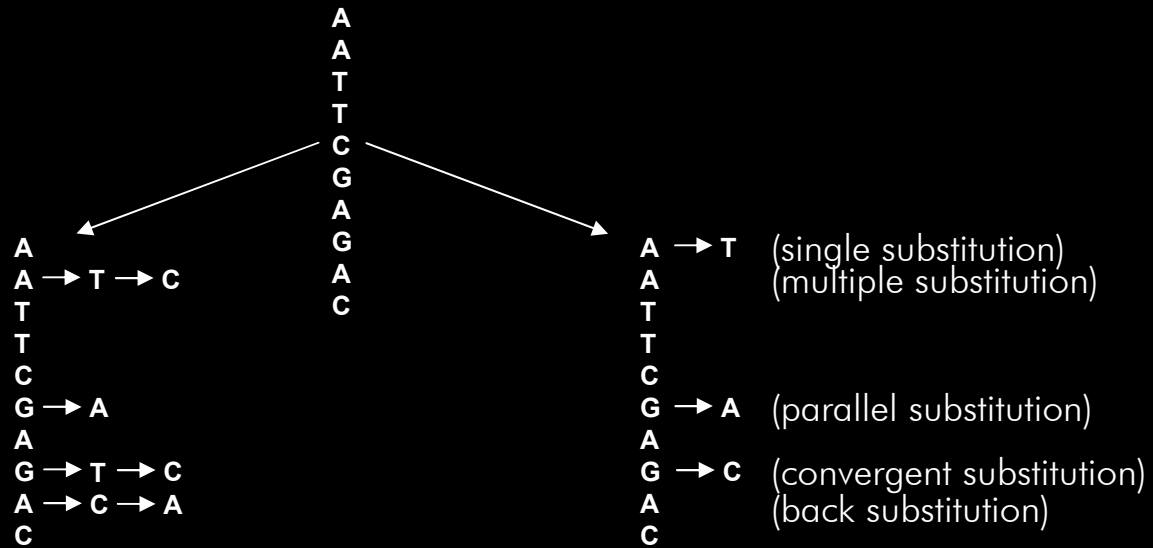


A
A
T
T
C
G
A
G
A
C

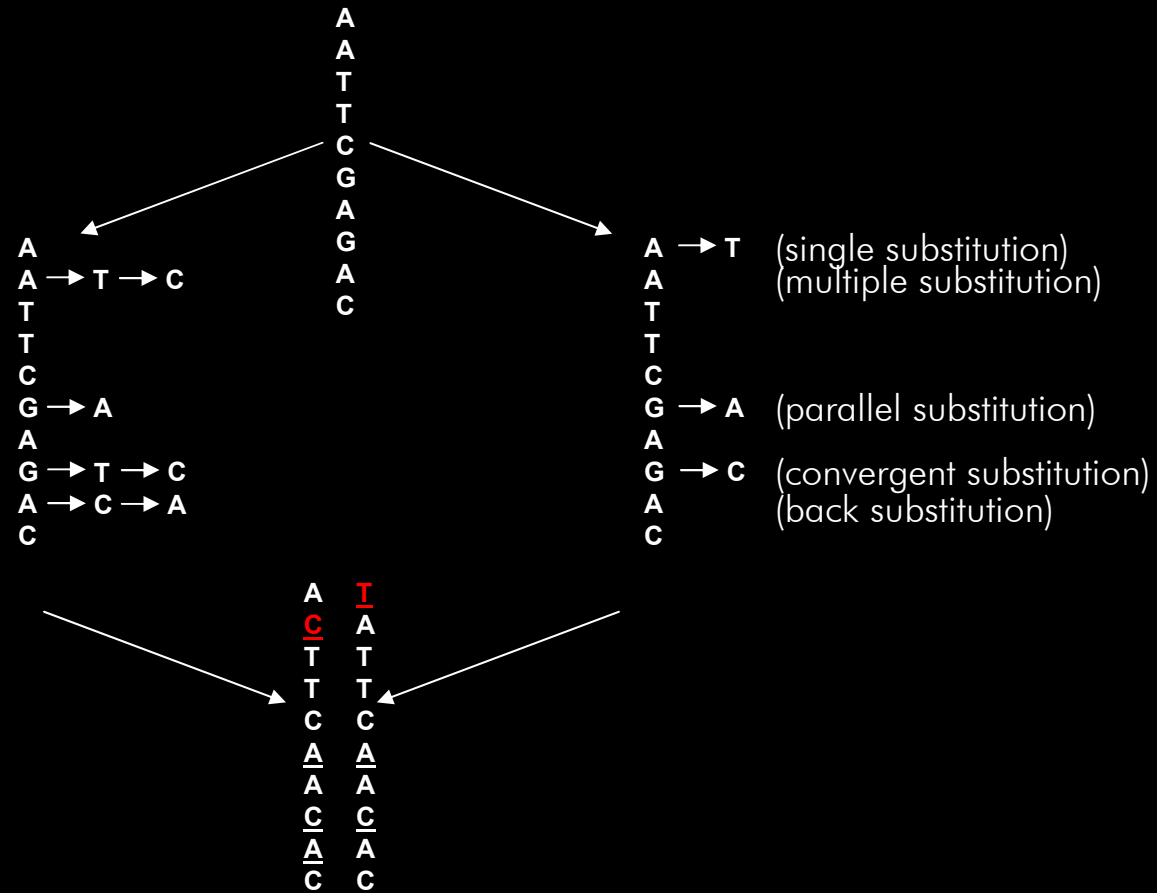
A
A
T
T
C
G
A
G
A
C

A
A
T
T
C
G
A
G
A
C

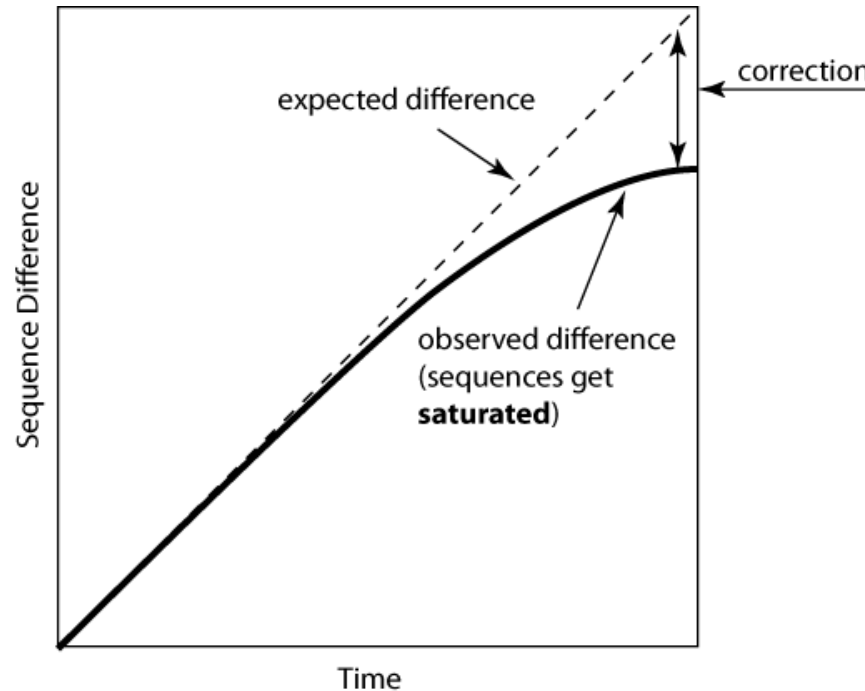
Sequences are not what we see ...



Sequences are not what we see ...

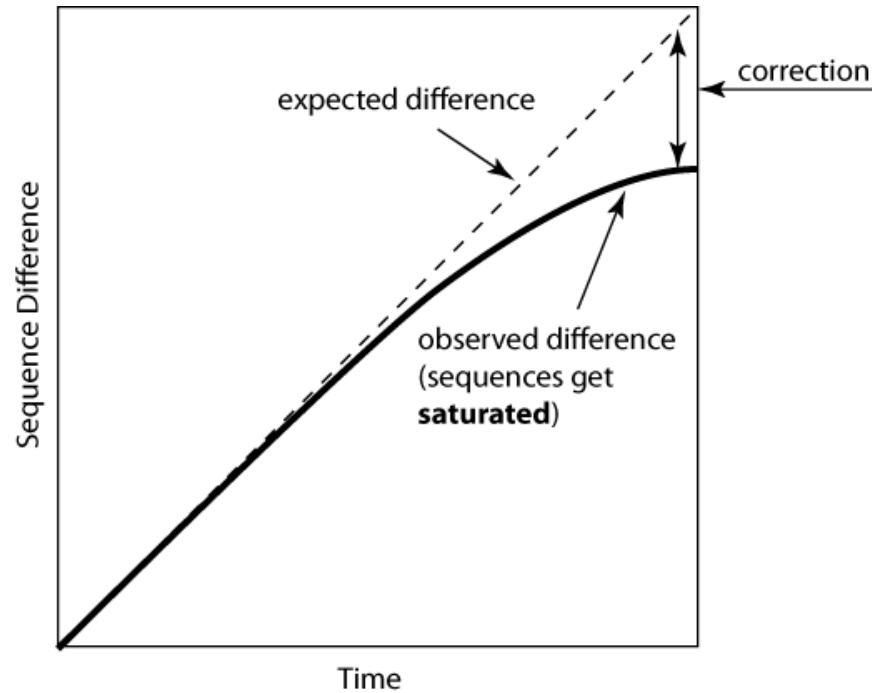


Observed vs Expected



Observed vs. **Expected** number of DNA substitutions. As time since divergence increases, multiple substitutions start to occur, **making number of visible substitutions smaller** than the number of **actual** ones. Eventually, after long-long time there will be substitutions at every site. Two random sequences with equal frequencies of base pairs will differ on average in 3/4 of sites. Correction is required to compensate for the difference in observed and expected number of substitutions.

Observed vs Expected



Observed vs. Expected number of DNA substitutions. As time since divergence increases, multiple substitutions start to occur, making number of visible substitutions smaller than the number of actual ones. Eventually, after long-long time there will be substitutions at every site. Two random sequences with equal frequencies of base pairs will differ on average in $3/4$ of sites. Correction is required to compensate for the difference in observed and expected number of substitutions.

Dayhoff PAM matrices

1. Dayhoff, Schwartz & Orcutt (1978) took these difficulties into consideration when defining their PAM matrices. The **basis** of their approach is to **obtain** substitution data **from alignments between very similar proteins** allowing for the evolutionary relationships of the proteins in families and then **extrapolate** this information to **longer** evolutionary distances.

Dayhoff PAM matrices

1. Dayhoff, Schwartz & Orcutt (1978) took these difficulties into consideration when defining their PAM matrices. The **basis** of their approach is to **obtain** substitution data **from alignments between very similar proteins** allowing for the evolutionary relationships of the proteins in families and then **extrapolate** this information to **longer** evolutionary distances.
2. They started by **constructing** hypothetical **phylogenetic** trees relating the sequences in **71 families** where each pair of sequences **differed by no more than 15% of their residues**.

Dayhoff PAM matrices

1. Dayhoff, Schwartz & Orcutt (1978) took these difficulties into consideration when defining their PAM matrices. The **basis** of their approach is to **obtain** substitution data **from alignments between very similar proteins** allowing for the evolutionary relationships of the proteins in families and then **extrapolate** this information to **longer** evolutionary distances.
2. They started by **constructing** hypothetical **phylogenetic** trees relating the sequences in **71 families** where each pair of sequences **differed by no more than 15% of their residues**.
3. To build the trees they used the **parsimony** method (see phylogenetic trees lecture) which provides a list of the **residues** that are **most likely** to have occurred at each position in each **ancestral** sequence.

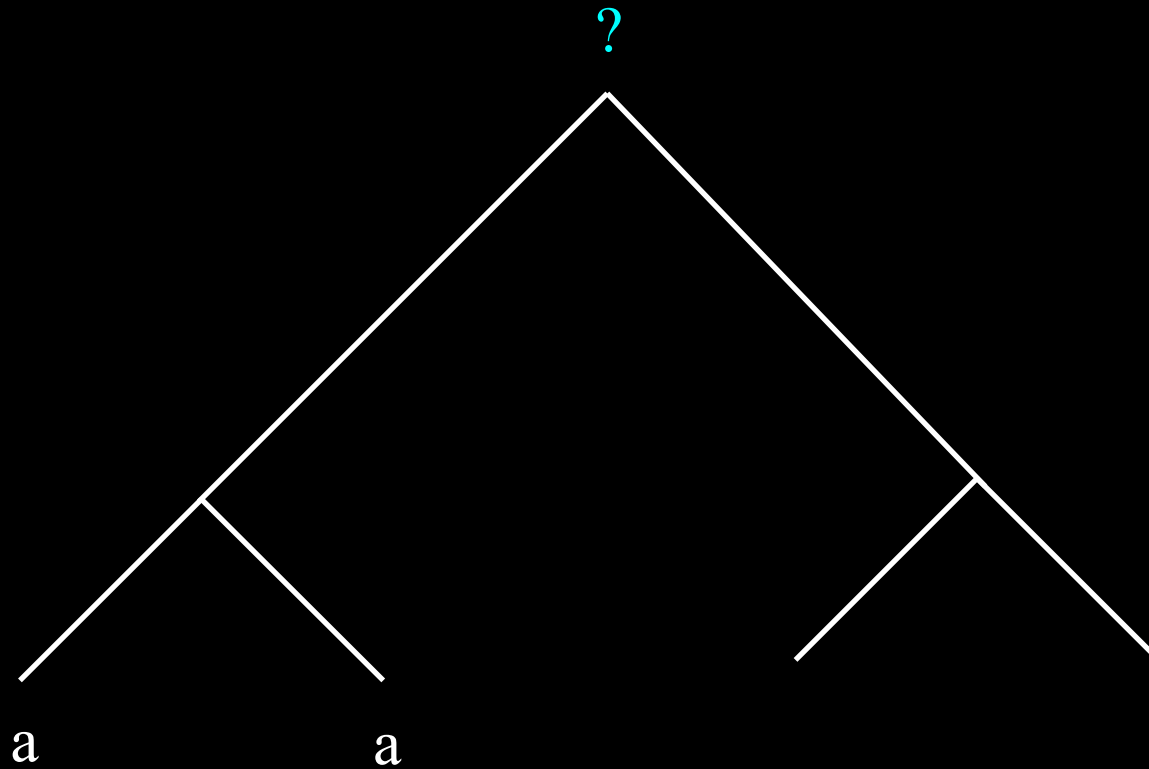
Dayhoff PAM matrices

1. Dayhoff, Schwartz & Orcutt (1978) took these difficulties into consideration when defining their PAM matrices. The **basis** of their approach is to **obtain** substitution data **from alignments between very similar proteins** allowing for the evolutionary relationships of the proteins in families and then **extrapolate** this information to **longer** evolutionary distances.
2. They started by **constructing** hypothetical **phylogenetic** trees relating the sequences in **71 families** where each pair of sequences **differed by no more than 15% of their residues**.
3. To build the trees they used the **parsimony** method (see phylogenetic trees lecture) which provides a list of the **residues** that are **most likely** to have occurred at each position in each **ancestral** sequence.
4. From this they could accumulate an array **A_{ab}** containing the **frequencies of all pairings** of residues **a** and **b** between sequences and their **immediate ancestors** in the tree.

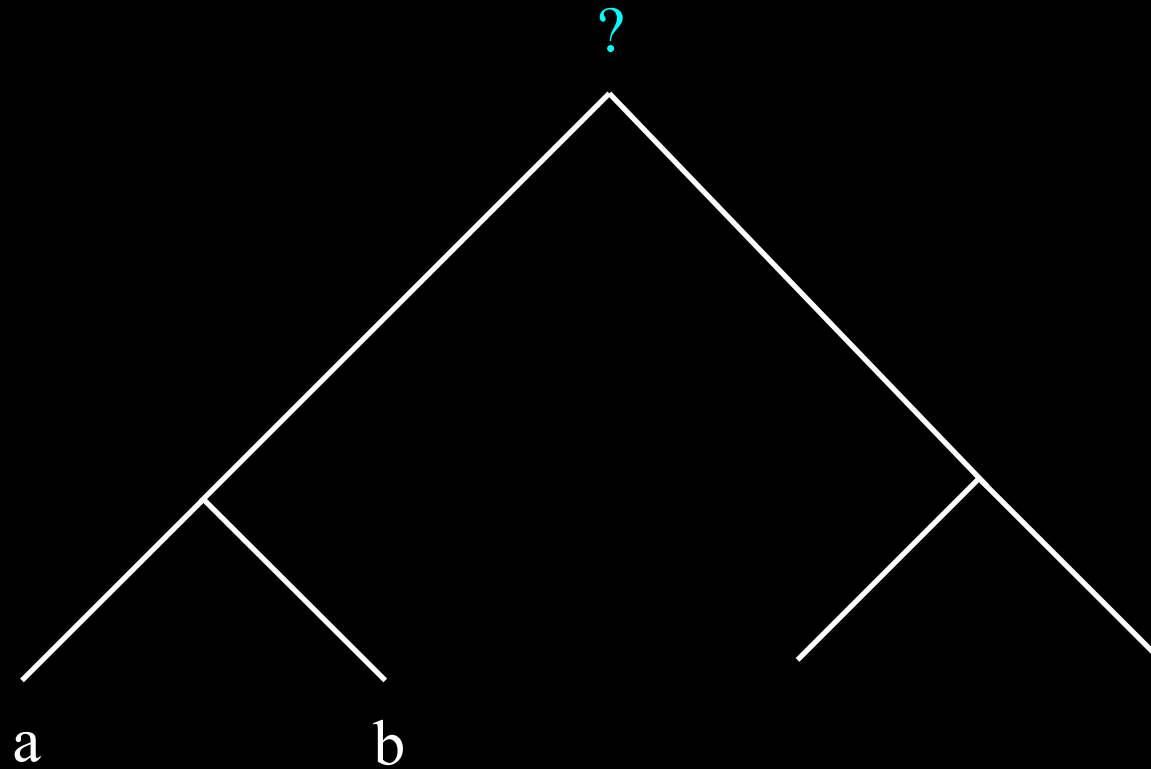
Dayhoff PAM matrices

1. Dayhoff, Schwartz & Orcutt (1978) took these difficulties into consideration when defining their PAM matrices. The **basis** of their approach is to **obtain** substitution data **from alignments between very similar proteins** allowing for the evolutionary relationships of the proteins in families and then **extrapolate** this information to **longer** evolutionary distances.
2. They started by **constructing** hypothetical **phylogenetic** trees relating the sequences in **71 families** where each pair of sequences **differed by no more than 15% of their residues**.
3. To build the trees they used the **parsimony** method (see phylogenetic trees lecture) which provides a list of the **residues** that are **most likely** to have occurred at each position in each **ancestral** sequence.
4. From this they could accumulate an array **A_{ab}** containing the **frequencies of all pairings** of residues **a** and **b** between sequences and their **immediate ancestors** in the tree.
5. The evolutionary **direction** of this pairing was **ignored** both **A_{ab}** and **A_{ba}** being incremented each time either an **a** in the ancestral sequence was replaced by a **b** in the descendant or vice versa. Basing the counts on the tree **avoided over-counting** substitutions because of evolutionary relatedness.

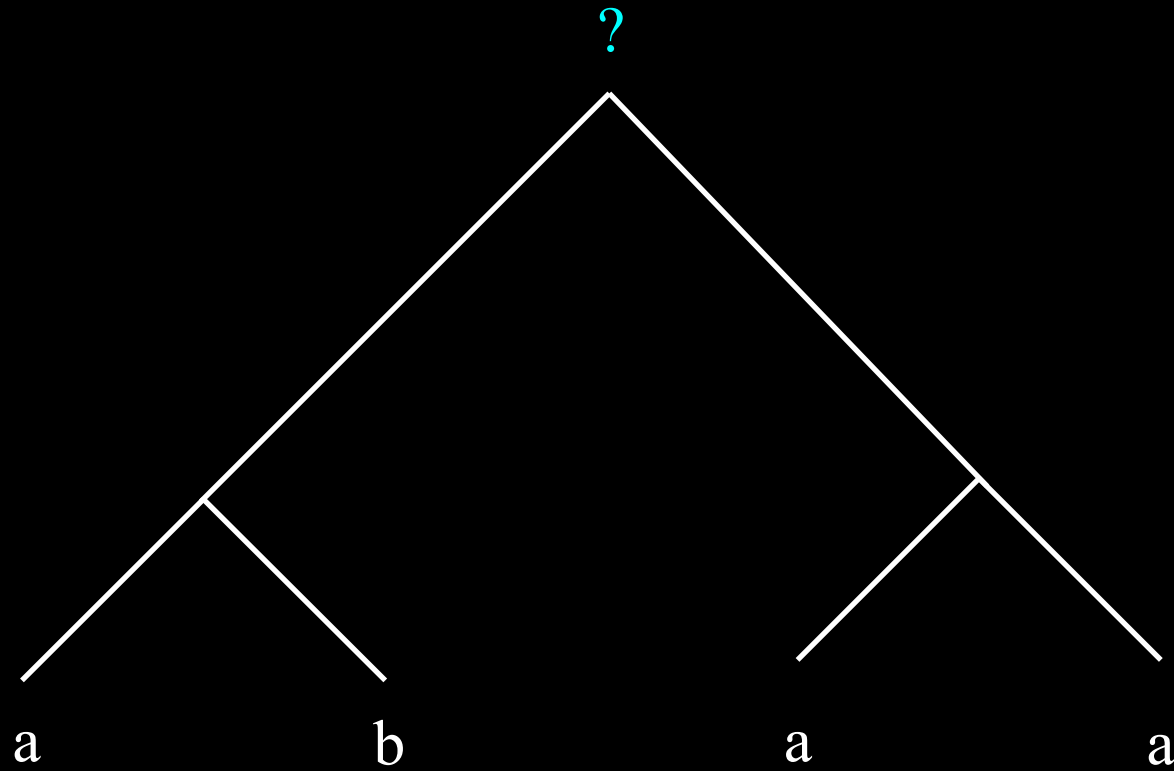
Parsimony
... a quick taste



***Parsimony
... a quick taste***



Parsimony
... a quick taste



Dayhoff PAM matrices

6. Because they wanted to **extrapolate to longer times**, the primary value that they needed to estimate was **not** the **joint** probability P_{ab} of seeing a aligned to b , but instead the **conditional** probability $P(b | a, t)$ that residue a is substituted by b in time t .

Dayhoff PAM matrices

6. Because they wanted to *extrapolate to longer times*, the primary value that they needed to estimate was *not* the *joint* probability P_{ab} of seeing a aligned to b , but instead the *conditional* probability $P(b|a,t)$ that residue a is substituted by b in time t .
7. $P(b|a,t) = P_{ab}(t)/q_a$. We can calculate conditional probabilities for a long time interval by multiplying those for a short interval. These conditional probabilities are known as *substitution probabilities*. The short time interval estimates for $P(b|a)$ can be derived from the A_{ab} matrix by setting $P(b|a) = B_{a,b} = A_{ab}/\sum_c(A_{ac})$.

Dayhoff PAM matrices

6. Because they wanted to **extrapolate to longer times**, the primary value that they needed to estimate was **not** the **joint** probability P_{ab} of seeing a aligned to b , but instead the **conditional** probability $P(b|a,t)$ that residue a is substituted by b in time t .
7. $P(b|a,t) = P_{ab}(t)/q_a$. We can calculate conditional probabilities for a long time interval by multiplying those for a short interval. These conditional probabilities are known as **substitution probabilities**. The short time interval estimates for $P(b|a)$ can be derived from the A_{ab} matrix by setting $P(b|a) = B_{a,b} = A_{ab}/\sum_c(A_{ac})$.
8. These values must next be adjusted to **correct** for divergence **time t** . The expected number of substitutions in a "typical" protein where the residue a occurs at the frequency q_a , is $\sum_{a,b}(q_a q_b B_{ab})$. Dayhoff et al. defined a substitution matrix to be a **1 PAM (point accepted mutation)** matrix if the **expected** number of substitutions was 1%, i.e. $\sum_{a,b}(q_a q_b B_{ab}) = 0.01$.

Dayhoff PAM matrices

9. To turn their B matrix into a 1 PAM matrix of substitution probabilities, they scaled the off-diagonal terms by a factor σ and adjusted the diagonal terms to keep the sum of a row equal to 1. More precisely, they defined $C_{ab} = \sigma B_{ab}$ for $a \neq b$ and $C_{aa} = \sigma B_{aa} + (1 - \sigma)$, with σ chosen to make C into a 1 PAM matrix.

Dayhoff PAM matrices

9. To turn their B matrix into a 1 PAM matrix of substitution probabilities, they scaled the off-diagonal terms by a factor σ and adjusted the diagonal terms to keep the sum of a row equal to 1. More precisely, they defined $C_{ab} = \sigma B_{ab}$ for $a \neq b$ and $C_{aa} = \sigma B_{aa} + (1 - \sigma)$, with σ chosen to make C into a 1 PAM matrix.
10. We will denote this 1 PAM C by $S(1)$. Its entries can be regarded as the probability of substituting a with b in unit time, $P(b | a, t=1)$.

Dayhoff PAM matrices

9. To turn their B matrix into a 1 PAM matrix of substitution probabilities, they scaled the off-diagonal terms by a factor σ and adjusted the diagonal terms to keep the sum of a row equal to 1. More precisely, they defined $C_{ab} = \sigma B_{ab}$ for $a \neq b$ and $C_{aa} = \sigma B_{aa} + (1 - \sigma)$, with σ chosen to make C into a 1 PAM matrix.
10. We will denote this 1 PAM C by $S(1)$. Its entries can be regarded as the probability of substituting a with b in unit time, $P(b | a, t=1)$.
11. To generate substitution matrices appropriate to longer times, $S(1)$ is raised to a power n , giving $S(n) = S(1)^n$. For instance $S(2)$, the matrix product of $S(1)$ with itself, has entries $P(a | b, t=2) = \sum_c P(a | c, t=1) P(c | b, t=1)$, which are the probabilities of the substitution of b by a occurring via some intermediate c .

Dayhoff PAM matrices

9. To turn their B matrix into a 1 PAM matrix of substitution probabilities, they scaled the off-diagonal terms by a factor σ and adjusted the diagonal terms to keep the sum of a row equal to 1. More precisely, they defined $C_{ab} = \sigma B_{ab}$ for $a \neq b$ and $C_{aa} = \sigma B_{aa} + (1 - \sigma)$, with σ chosen to make C into a 1 PAM matrix.
10. We will denote this 1 PAM C by $S(1)$. Its entries can be regarded as the probability of substituting a with b in unit time, $P(b | a, t=1)$.
11. To generate substitution matrices appropriate to longer times, $S(1)$ is raised to a power n , giving $S(n) = S(1)^n$. For instance $S(2)$, the matrix product of $S(1)$ with itself, has entries $P(a | b, t=2) = \sum_c P(a | c, t=1) P(c | b, t=1)$, which are the probabilities of the substitution of b by a occurring via some intermediate c .
12. For small n , the off-diagonal entries increase approximately linearly with n . Another way to view this is that the matrix $S(n)$ represents the results of n steps of a Markov chain with 20 states corresponding to the 20 amino acids, each step having transition probabilities given by $S(1)$.

Dayhoff PAM matrices

13. Finally, a matrix of scores is obtained from $S(t)$. Since $P(b | a) = p_{ab}/q_a$, the entries of the score matrix for time t are given by

$$s(a, b | t) = \log \frac{P(b | a, t)}{q_b}$$

The values are scaled and rounded to the nearest integer by computational convergence. The most widely used matrix is PAM250, which is scaled by $3/\log 2$ to give scores in third-bits.

BLOSUM matrices

1. The Dayhoff matrices have been one the mainstays of the sequence comparison techniques, but they do have their limitations. The entries in $S(1)$ arise mostly from short time interval substitutions and raising $S(1)$ to a higher power, to give e.g. PAM250, does not capture the true difference between short and long time substitutions. The former are dominated by aminoacid substitutions that arise from single base changes in codon triplets, for example $L \leftrightarrow I$, $L \leftrightarrow V$, or $Y \leftrightarrow F$, whereas the latter show all types of codon changes.

BLOSUM matrices

1. The Dayhoff matrices have been one of the mainstays of the sequence comparison techniques, but they do have their limitations. The entries in $S(1)$ arise mostly from short time interval substitutions and raising $S(1)$ to a higher power, to give e.g. PAM250, does not capture the true difference between short and long time substitutions. The former are dominated by amino acid substitutions that arise from single base changes in codon triplets, for example $L \leftrightarrow I$, $L \leftrightarrow V$, or $Y \leftrightarrow F$, whereas the latter show all types of codon changes.
2. Since the PAM matrices were made, databases have been formed containing multiple alignments of more distantly related proteins. A more successful matrix, exploits these advancements and is named BLOSUM (Henikoff & Henikoff 1992). The sequences from each block were clustered putting two sequences into the same cluster whenever their percentage of identical residues exceeded some level $L\%$. Henikoff & Henikoff then calculated the frequencies A_{ab} of observing residue a in one cluster aligned against residue b in another cluster, correcting for the sizes of the clusters by weighting each occurrence by $1/(n_1 n_2)$, where n_1 and n_2 are the respective cluster sizes.

BLOSUM matrices

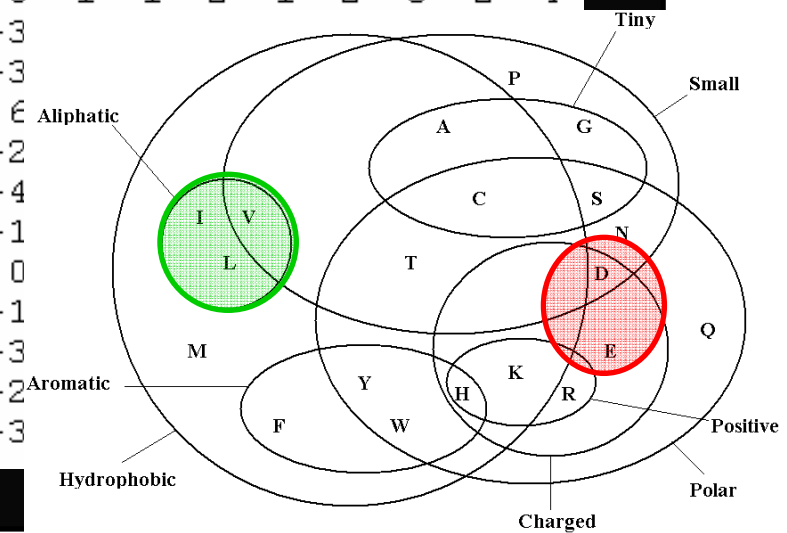
1. The Dayhoff matrices have been one of the mainstays of the sequence comparison techniques, but they do have their limitations. The entries in $S(1)$ arise mostly from short time interval substitutions and raising $S(1)$ to a higher power, to give e.g. PAM250, does not capture the true difference between short and long time substitutions. The former are dominated by amino acid substitutions that arise from single base changes in codon triplets, for example $L \leftrightarrow I$, $L \leftrightarrow V$, or $Y \leftrightarrow F$, whereas the latter show all types of codon changes.
2. Since the PAM matrices were made, databases have been formed containing multiple alignments of more distantly related proteins. A more successful matrix, exploits these advancements and is named BLOSUM (Henikoff & Henikoff 1992). The sequences from each block were clustered putting two sequences into the same cluster whenever their percentage of identical exceeded some level $L\%$. Henikoff & Henikoff then calculated the frequencies A_{ab} of observing residue a in one cluster aligned against residue b in another cluster, correcting for the sizes of the clusters by weighting each occurrence by $1/(n_1 n_2)$, where n_1 and n_2 are the respective cluster sizes.
3. From the A_{ab} , they estimated q_a and p_{ab} by $q_a = \sum_b A_{ab} / \sum_{cd} A_{cd}$, i.e. the fraction of pairings between a and b out of all observed pairings. From these they derived the score matrix entries using the standard equations $s(a,b) = \log p_{ab} / q_a q_b$ (2.3). Again the resulting log-odds score matrices were scaled and rounded to the nearest integer value. The matrices for $L=62$ and $L=50$ are widely used for pairwise alignment and database searching, with **blosum62** being standard for **ungapped** matching and **blosum50** for **gapped** alignments. Note that **lower L values** correspond to **longer evolutionary time**, and are applicable for more distant searches.

BLOSUM 50 Substitution Matrix

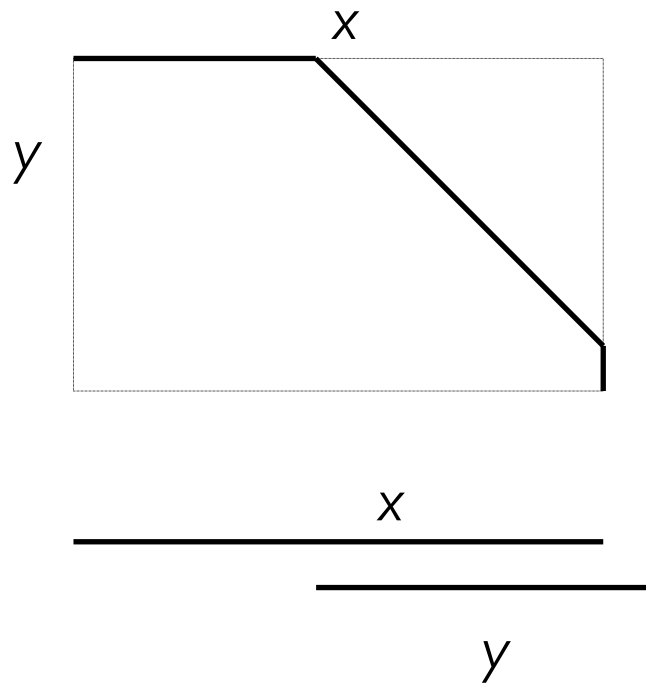
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3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

BLOSUM 50 Substitution Matrix

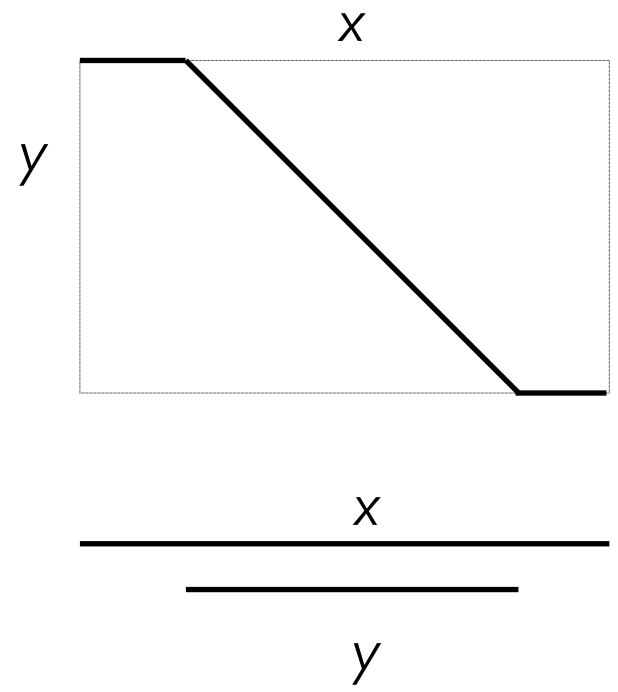
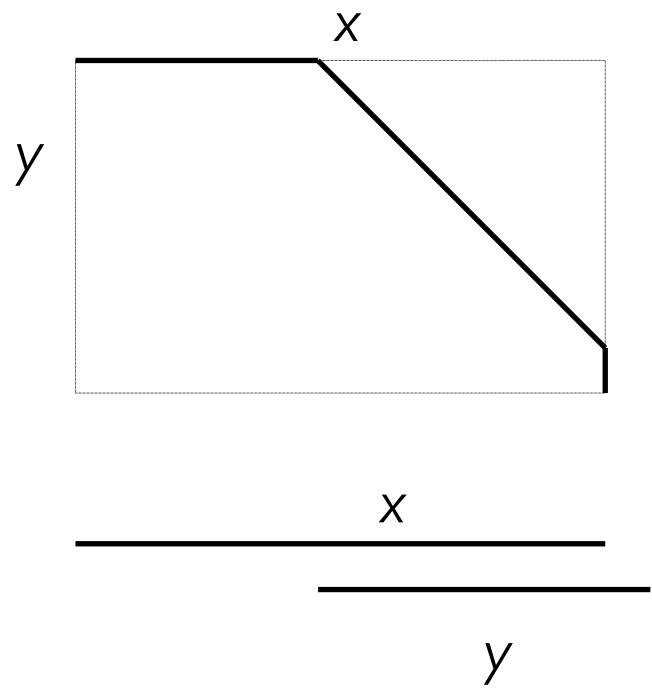
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3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



Sequence Alignment ... a 2D viewpoint

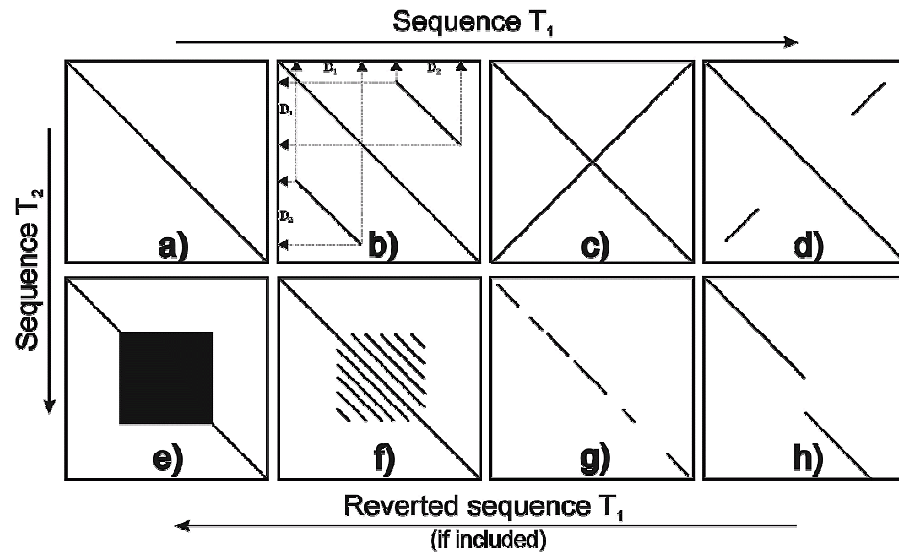


Sequence Alignment ... a 2D viewpoint



Dot Plot Patterns

- a) Perfect match
- b) Duplication
- c) Palindromic sequences (e.g. ATGCGTA)
- d) Partial Palindromic sequences
- e) Bold blocks on the main diagonal: repetition of the same symbol in both sequences, e.g. (G)50, so called microsatellite repeats
- f) Parallel lines indicate tandem repeats of a larger motif in both sequences, e.g. (AGCTCTGAC)20, so called minisatellite patterns.
- g) Mismatch
- h) Partial deletion in sequence 1 or insertion in sequence 2, so called 'indel'.





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

For two sequences of length $n=100$, there are 10^{60} different global alignments

Global Alignment

The Needleman-Wunsch algorithm

- ✓ A elegant way to *reduce* the massive number of possibilities that need to be considered, yet still *guarantees* that the best solution will be found (Saul Needleman and Christian Wunsch, 1970).
- ✓ The basic idea is to build up the best alignment by using optimal alignments of *smaller sub-sequences*.
- ✓ The Needleman-Wunsch algorithm is an example of *dynamic programming*, a concept first fuelled by by Richard Bellman in 1940.

Dynamic Programming ***...the basics***

A *divide-and-conquer* strategy:

1. Break the problem into smaller sub-problems.
2. Solve the smaller problems optimally.
3. Use the sub-problem solutions to construct an optimal solution for the original problem.

Dynamic programming can be applied only to problems exhibiting the properties of *overlapping sub-problems*.

Examples include:

1. Travelling salesman problem.
2. Finding the best chess move.

Dynamic Programming

...the basics

A *divide-and-conquer* strategy:

1. Break the problem into smaller sub-problems.
2. Solve the smaller problems optimally.
3. Use the sub-problem solutions to construct an optimal solution for the original problem.

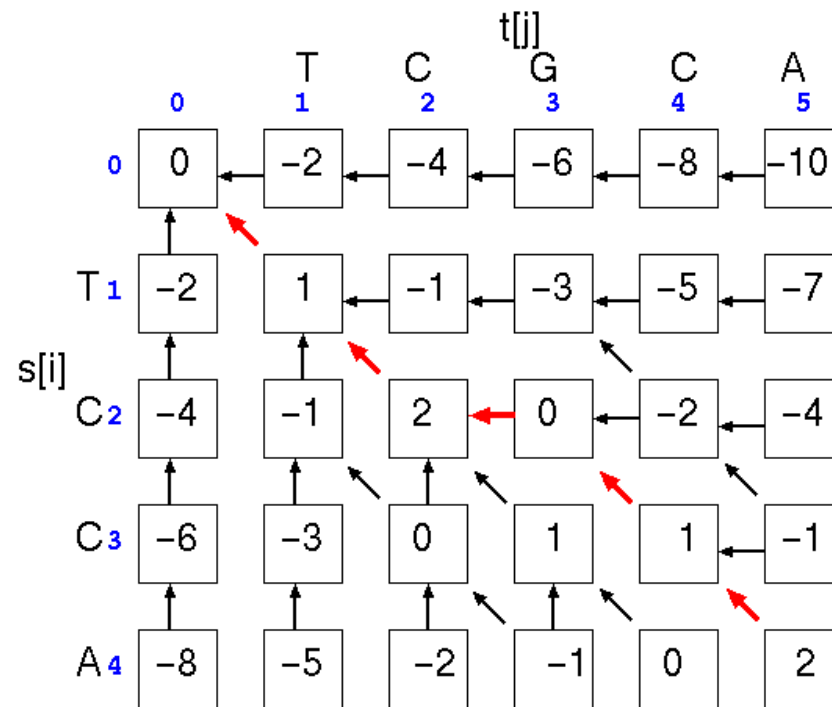
Dynamic programming can be applied only to problems exhibiting the properties of *overlapping sub-problems*.

Examples include:

1. Travelling salesman problem.
2. Finding the best chess move.

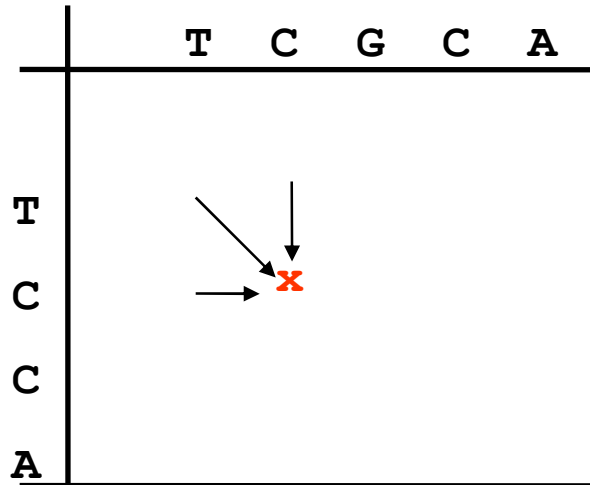
Global Alignment

The Needleman-Wunsch algorithm



Global Alignment

The Needleman-Wunsch algorithm

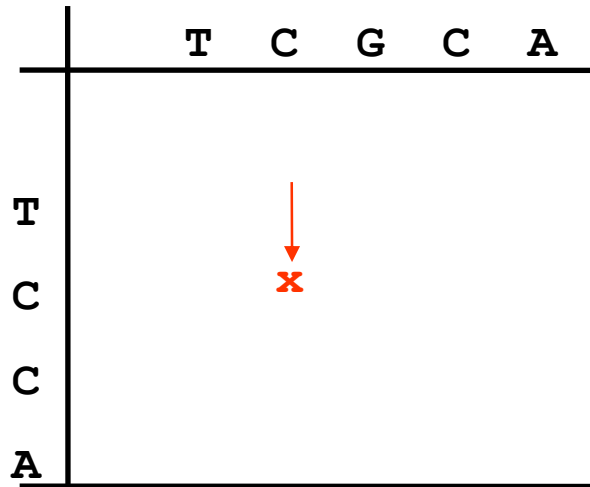


Any given point in matrix can only be reached from *three* possible previous positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the *highest scoring* of the three possibilities.

Global Alignment

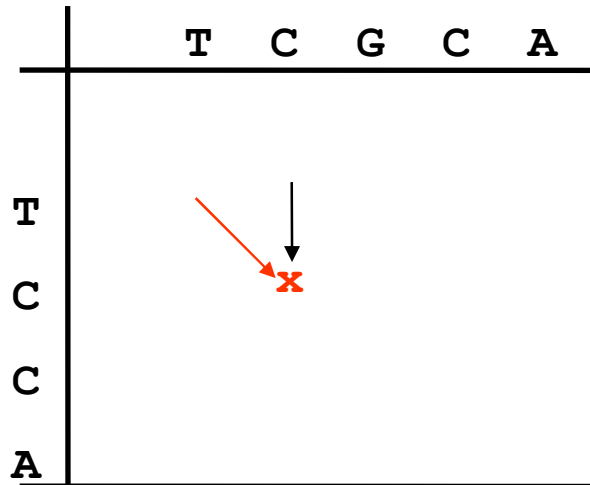
The Needleman-Wunsch algorithm



$$\text{score}(x,y) = \max \left\{ \begin{array}{l} \text{score}(x,y-1) - \text{Gap}_{\text{penalty}} \\ \text{score}(x-1,y) + \text{match/mismatch} \\ \text{score}(x-1,y-1) + \text{match/mismatch} \end{array} \right.$$

Global Alignment

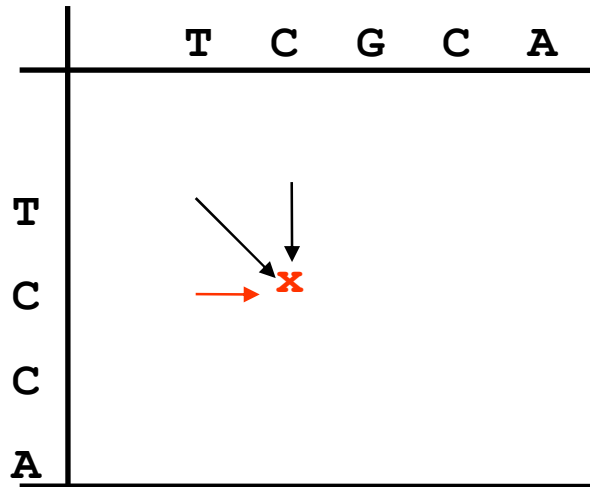
The Needleman-Wunsch algorithm



$$\text{score}(x,y) = \max \begin{cases} \text{score}(x,y-1) - \text{Gap}_{\text{penalty}} \\ \text{score}(x-1,y-1) + \text{substitution}_{\text{score}(x,y)} \end{cases}$$

Global Alignment

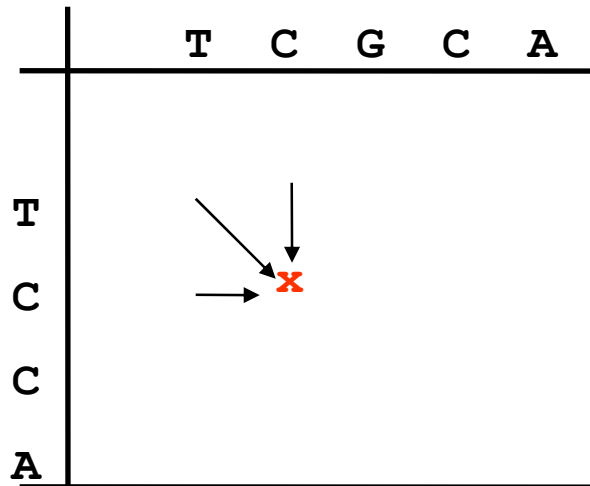
The Needleman-Wunsch algorithm



$$\text{score}(x,y) = \max \begin{cases} \text{score}(x,y-1) - \text{Gap}_{\text{penalty}} \\ \text{score}(x-1,y-1) + \text{substitution}_{\text{score}(x,y)} \\ \text{score}(x-1,y) - \text{Gap}_{\text{penalty}} \end{cases}$$

Global Alignment

The Needleman-Wunsch algorithm



Each new score is found by choosing the *maximum* of three possibilities.

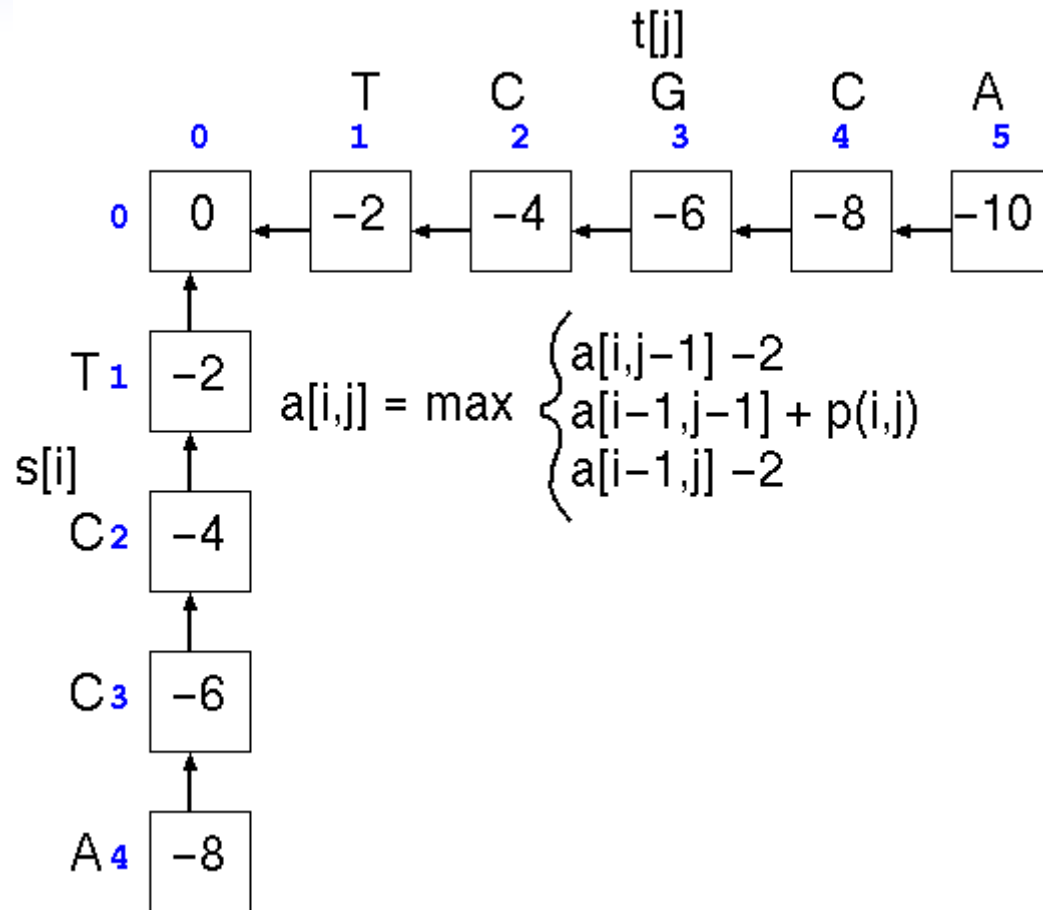
For each square in matrix: *keep track* of where best score came from.

Fill in scores one row at a time, *starting in upper left* corner of matrix, *ending in lower right corner*.

$$\text{score}(x,y) = \max \begin{cases} \text{score}(x,y-1) - \text{Gap}_{\text{penalty}} \\ \text{score}(x-1,y-1) + \text{substitution}_{\text{score}(x,y)} \\ \text{score}(x-1,y) - \text{Gap}_{\text{penalty}} \end{cases}$$

Global Alignment

The Needleman-Wunsch algorithm

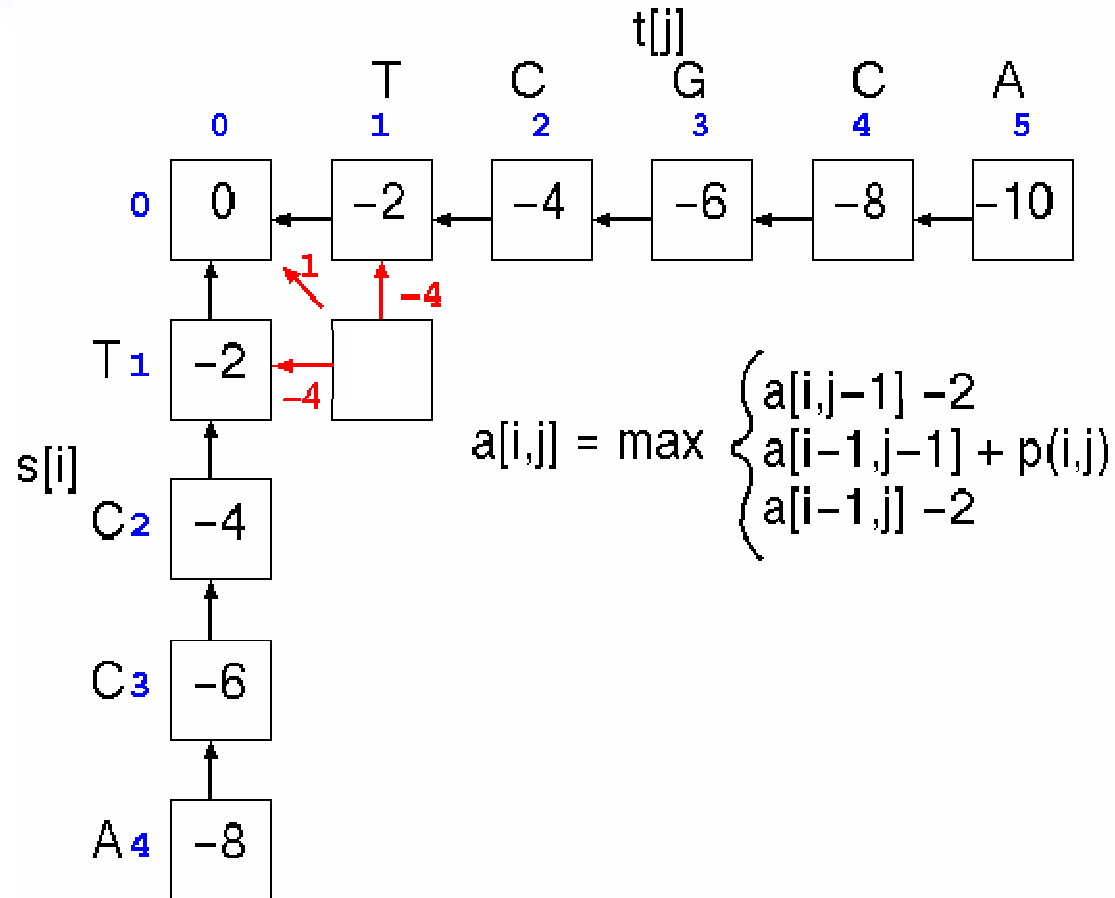


	A	C	G	T
A	1	-1	-1	-1
C	-1	1	-1	-1
G	-1	-1	1	-1
T	-1	-1	-1	1

Gaps: -2

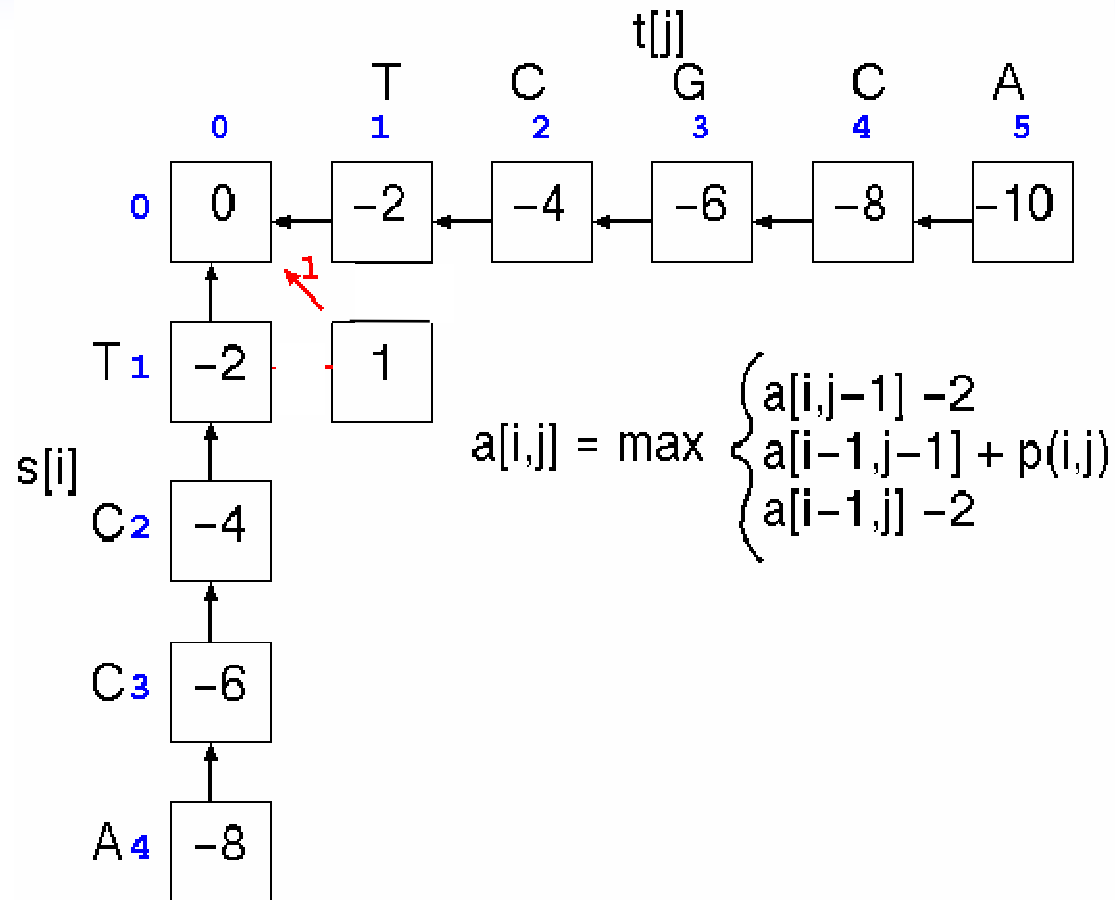
Global Alignment

The Needleman-Wunsch algorithm



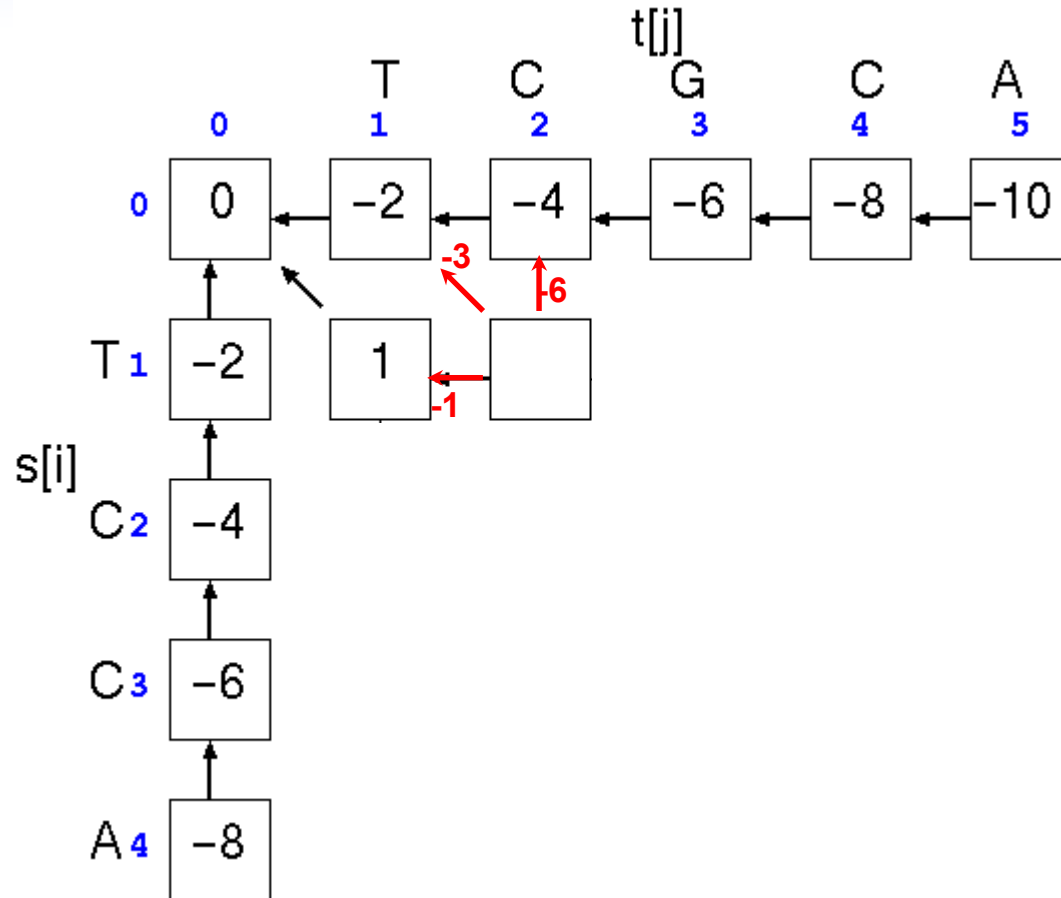
Global Alignment

The Needleman-Wunsch algorithm



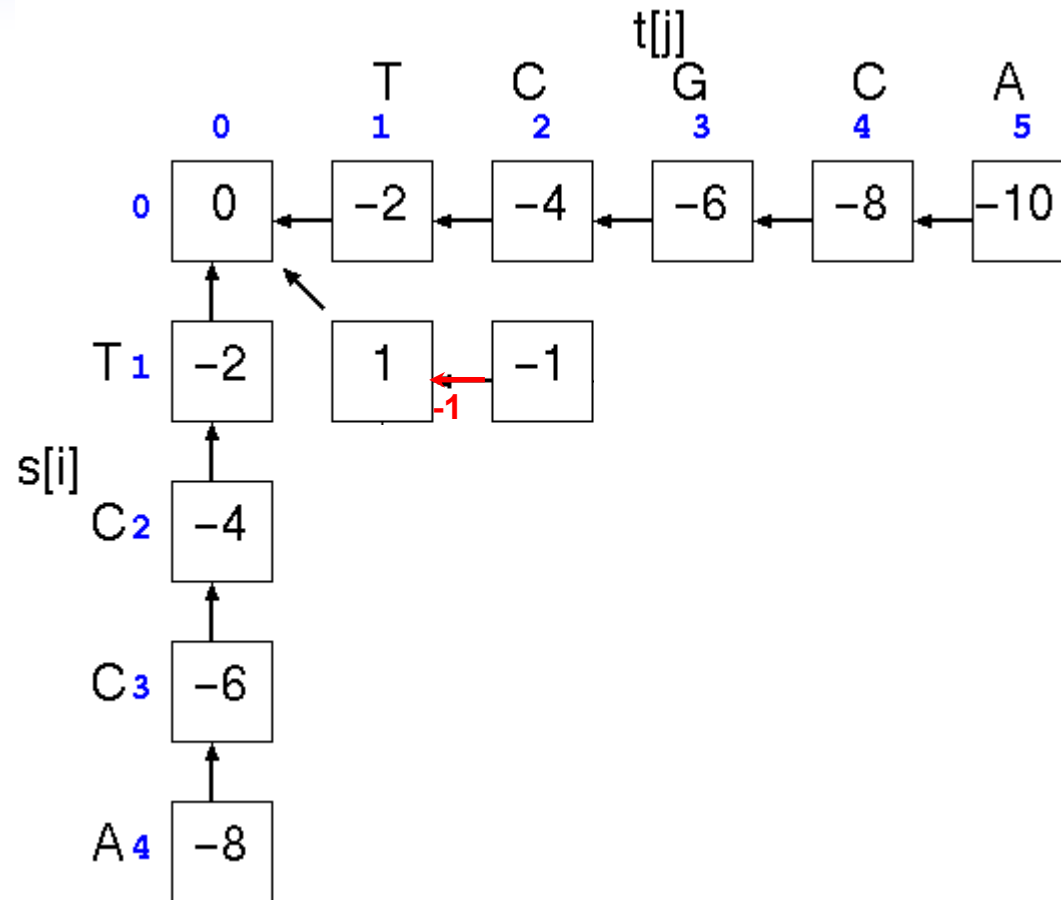
Global Alignment

The Needleman-Wunsch algorithm



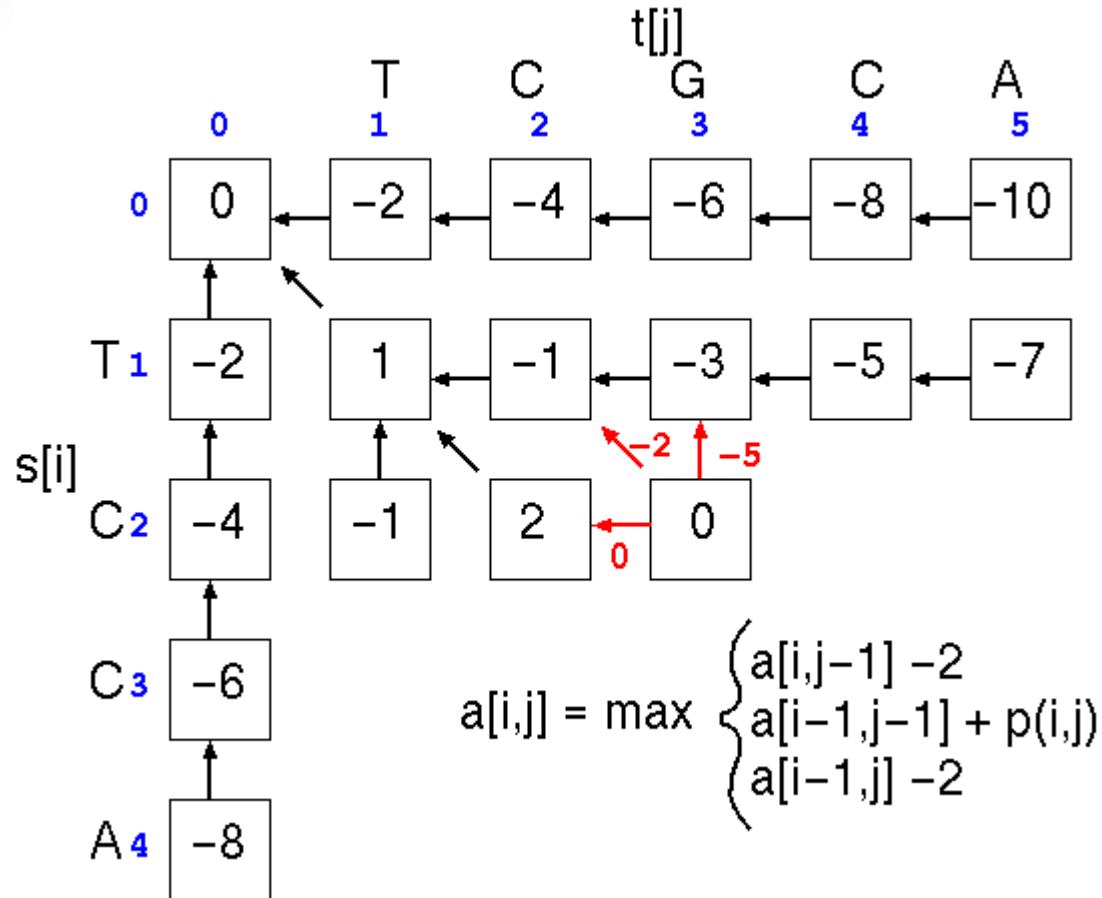
Global Alignment

The Needleman-Wunsch algorithm



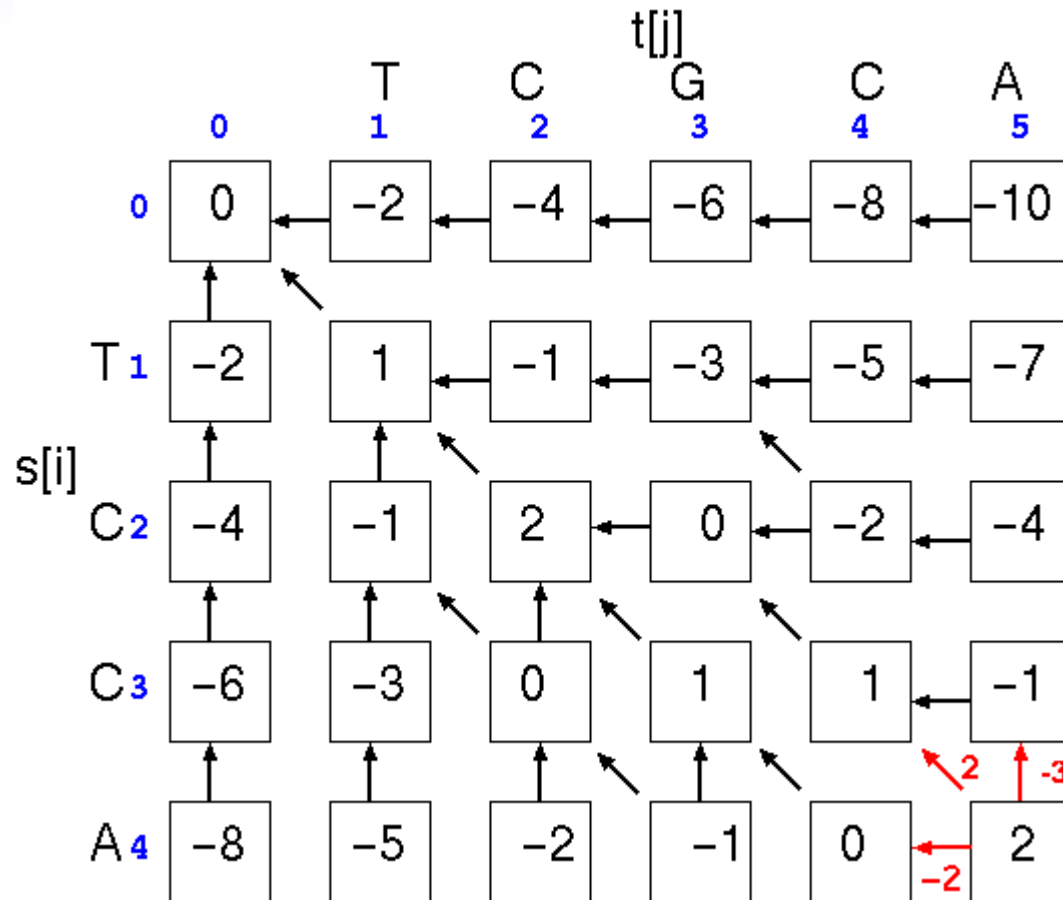
Global Alignment

The Needleman-Wunsch algorithm



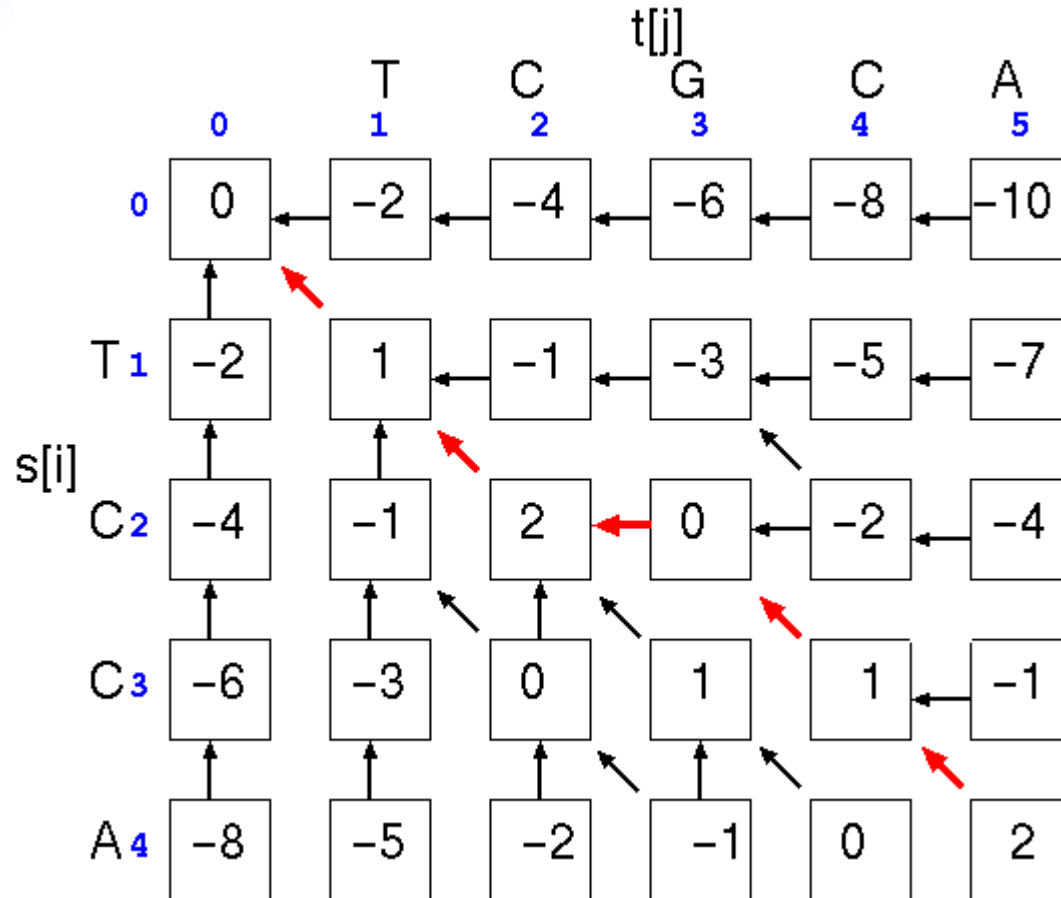
Global Alignment

The Needleman-Wunsch algorithm



Global Alignment

The Needleman-Wunsch algorithm



$$\begin{array}{cccccc}
 \text{T} & \text{C} & \text{G} & \text{C} & \text{A} & \\
 : & : & & : & : & \\
 \text{T} & \text{C} & - & \text{C} & \text{A} & \\
 \hline
 1 & + & 1 & - & 2 & + & 1 & + & 1 & = & \underline{2}
 \end{array}$$

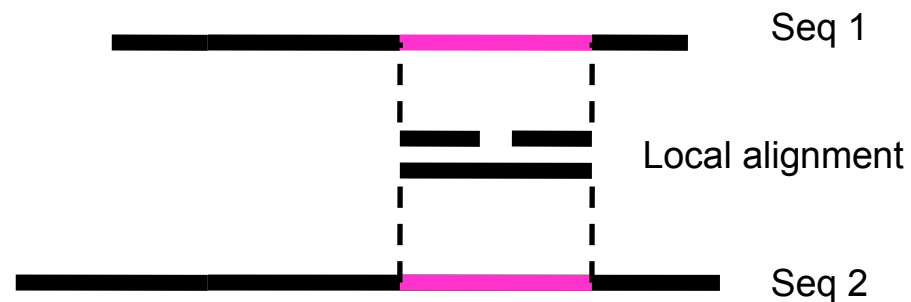
Local Alignment

The Smith-Waterman algorithm

Global alignment: align full length of both sequences. (The “Needleman-Wunsch” algorithm).



Local alignment: find best partial alignment of two sequences (the “Smith-Waterman” algorithm).



Local Alignment

The Smith-Waterman algorithm

Modifications relative to the Needleman-Wunsch algorithm:

1. The recursive formula is changed by adding a fourth possibility: *zero*.
2. This means local alignment scores are *never negative*.

$$\text{score}(x,y) = \max \left\{ \begin{array}{l} \text{score}(x,y-1) - \text{gap-penalty} \\ \text{score}(x-1,y-1) + \text{substitution-score}(x,y) \\ \text{score}(x-1,y) - \text{gap-penalty} \\ 0 \end{array} \right.$$

3. Trace-back is *started* at the *highest* value *rather than in lower right corner*.
4. Trace-back is *stopped* as soon as a *zero* is encountered.

Local Alignment ... example

		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	5	0	5	0	0	0	0	0
W	0	0	0	0	2	0	20	12	4	0	0
H	0	10	2	0	0	0	12	18	22	14	6
E	0	2	16	8	0	0	4	10	18	28	20
A	0	0	8	21	13	5	0	4	10	20	27
E	0	0	6	13	18	12	4	0	4	16	26

AWGHE
AW-HE

Global alignment

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_i) \\ F(i-1, j) - d \\ F(i, j-1) - d \end{cases}$$

Hybrid match

$$F(i, j) = \max \begin{cases} F(i-1, 0) + s(x_i, y_i) \\ F(i-1, n) + s(x_i, y_i) \\ F(i-1, 1) - d \\ F(i, 0) - d \end{cases}$$

Local alignment

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_i) \\ F(i-1, j) - d \\ F(i, j-1) - d \\ 0 \end{cases}$$

Alignment with complex gaps

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_i) \\ F(k, j) + \gamma(i-k) & k = 0, \dots, i-1 \\ F(i, k) + \gamma(j-k) & k = 0, \dots, j-1 \end{cases}$$

Repeated matches

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_i) \\ F(i-1, j) - d \\ F(i, j-1) - d \\ F(i, 0) \end{cases}$$

$$F(i, 0) = \max \begin{cases} F(i-1, 0) \\ F(i-1, j) - T \quad j = 1, \dots, m \end{cases}$$

Overlap matches

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_i) \\ F(i-1, j) - d \\ F(i, j-1) - d \\ F(i, 0) \end{cases}$$

$$F(i, 0) = \max \begin{cases} F(i-1, 0) \\ F(i-1, m) - T \end{cases}$$

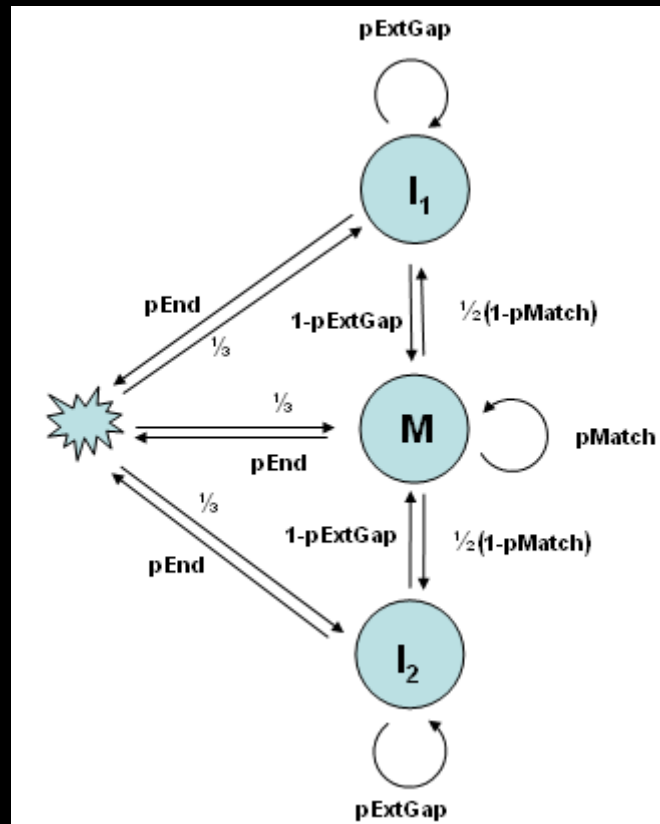
Alignment with affine gaps

$$M(i, j) = \max \begin{cases} M(i-1, j-1) + s(x_i, y_i) \\ I_x(i-1, j-1) + s(x_i, y_i) \\ I_y(i-1, j-1) + s(x_i, y_i) \end{cases}$$

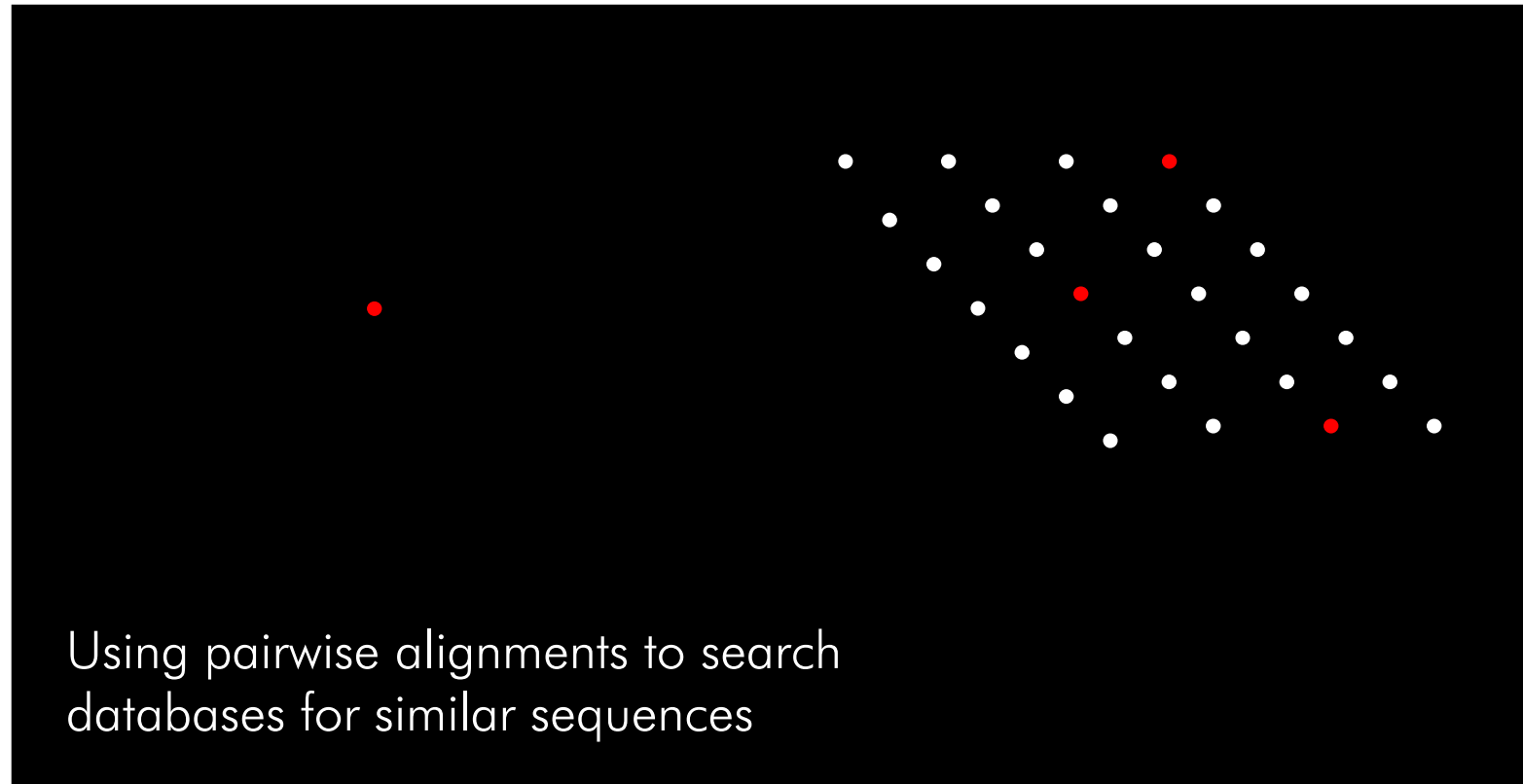
$$I_x(i, j) = \max \begin{cases} M(i-1, j) - d \\ I_x(i-1, j) - e \end{cases}$$

$$I_y(i, j) = \max \begin{cases} M(i, j-1) - d \\ I_y(i, j-1) - e \end{cases}$$

Finite State Automaton (FSA)



Database Searching



I am exhausted !

... searching with exhaustive algorithms

- ✓ Alignment algorithms described so far *guarantees* to find the *optimal score* given a specified scoring scheme.
- ✓ They have *time complexity* of the order of $O(n, m)$, i.e. the product of the sequence lengths.
- ✓ For a sequence of *1,000* residues and a protein database of *100,000,000* residues we would need to evaluate $\sim 10^{11}$ *matrix cells*.
- ✓ At 10,000,000 cells per second (single workstation) this would take *10⁴ seconds*.
- ✓ Solution: *heuristic* algorithms to search as *small a fraction* as possible of the cells in the dynamic programming matrix, while still looking at all the *high scoring alignments*.

Heuristic Algorithms

FASTA (Pearson 1995)

1. Uses heuristics to avoid calculating the full dynamic programming matrix
2. Speed up searches by an order of magnitude compared to full Smith-Waterman
3. The statistical side of FASTA is still stronger than BLAST

BLAST (Altschul 1990, 1997)

1. Uses rapid word lookup methods to completely skip most of the database entries
2. Extremely fast; One order of magnitude faster than FASTA
3. Two orders of magnitude faster than Smith-Waterman
4. Almost as sensitive as FASTA

<http://blast.ncbi.nlm.nih.gov/Blast.cgi>

Blast-ies

BLASTN

- Nucleotide query sequence
- Nucleotide database

BLASTP

- Protein query sequence
- Protein database

BLASTX

- Nucleotide query sequence
- Protein database
- Compares all six reading frames with the database

TBLASTN

- Protein query sequence
- Nucleotide database
- "On the fly" six frame translation of database

TBLASTX

- Nucleotide query sequence
- Nucleotide database
- Compares all reading frames of query with all reading frames of the database

Significance of scores

- ✓ Now that we know how to find an optimal alignment, how can we assess the *significance of its score*?
- ✓ How do we decide if it is a *biologically meaningful* alignment giving evidence of a *homology* or just the best alignment between two entirely *unrelated sequences*?
- ✓ Calculate the *chance* of a match score *greater* than the *observed* value, assuming a null model, i.e. that the underlying sequences were unrelated.

Significance of scores ... the Bayesian Approach

✓What we are really after is the **probability** that the sequences are related as opposed to being unrelated $P(M | x,y)$ rather than the **likelihood** $P(x,y | M)$. We can calculate $P(M | x,y)$ using Bayes' rule.

Significance of scores ... the Bayesian Approach

- ✓ What we are really after is the **probability** that the sequences are related as opposed to being unrelated $P(M | x,y)$ rather than the **likelihood** $P(x,y | M)$. We can calculate $P(M | x,y)$ using Bayes' rule.
- ✓ Assumptions: the prior probability that the sequences are related: $P(M)$, i.e. the match model is correct and the prior probability that the random model is correct: $P(R) = 1 - P(M)$.

Significance of scores ... the Bayesian Approach

- ✓ What we are really after is the **probability** that the sequences are related as opposed to being unrelated $P(M | x,y)$ rather than the **likelihood** $P(x,y | M)$. We can calculate $P(M | x,y)$ using Bayes' rule.
- ✓ Assumptions: the prior probability that the sequences are related: $P(M)$, i.e. the match model is correct and the prior probability that the random model is correct: $P(R) = 1 - P(M)$.
- ✓ Once we have seen the data, the posterior probability that the match model is correct, is:

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

Significance of scores ... the Bayesian Approach

- ✓ What we are really after is the **probability** that the sequences are related as opposed to being unrelated $P(M | x,y)$ rather than the **likelihood** $P(x,y | M)$. We can calculate $P(M | x,y)$ using Bayes' rule.
- ✓ Assumptions: the prior probability that the sequences are related: $P(M)$, i.e. the match model is correct and the prior probability that the random model is correct: $P(R) = 1 - P(M)$.
- ✓ Once we have seen the data, the posterior probability that the match model is correct, is:

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

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

Significance of scores ... the Bayesian Approach

- ✓ What we are really after is the **probability** that the sequences are related as opposed to being unrelated $P(M | x,y)$ rather than the **likelihood** $P(x,y | M)$. We can calculate $P(M | x,y)$ using Bayes' rule.
- ✓ Assumptions: the prior probability that the sequences are related: $P(M)$, i.e. the match model is correct and the prior probability that the random model is correct: $P(R) = 1 - P(M)$.
- ✓ Once we have seen the data, the posterior probability that the match model is correct, is:

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

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

$$= \frac{P(x, y | M)P(M) / P(x, y | R)P(R)}{1 + P(x, y | M)P(M) / P(x, y | R)P(R)}$$

Significance of scores ... the Bayesian Approach

Let

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

where

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

is the log-odds score of the alignment. Then

Significance of scores ... the Bayesian Approach

Let

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

where

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

is the log-odds score of the alignment. Then

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

where

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

Significance of scores ... the Bayesian Approach

Let

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

where

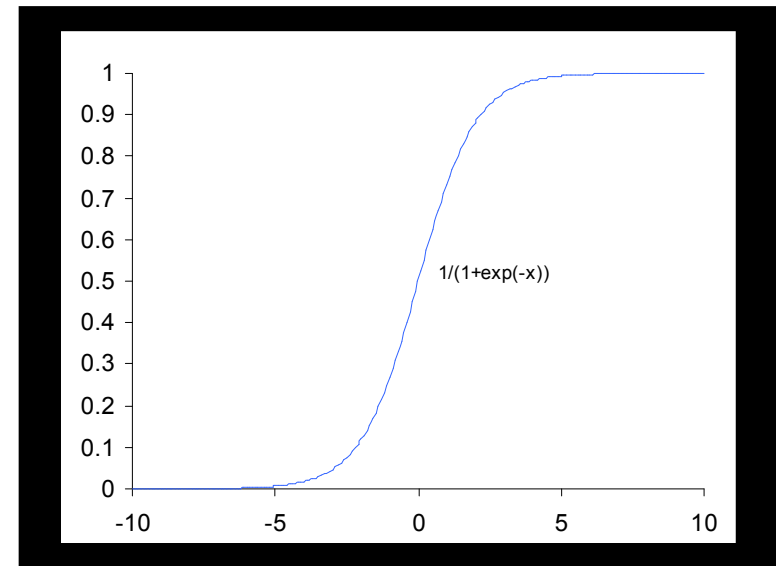
$$S = \log \left(\frac{P(x, y | M)}{P(x, y | R)} \right)$$

is the log-odds score of the alignment. Then

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

where

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



The **logistic function** normalizes the output of model and can be **considered** as an **estimate** of the **probability** that a given structure is true, given the model.

Significance of scores ... the extreme value distribution

✓ There is an alternative way to consider significance using a more classical statistical framework. We can look at the **distribution** of the **maximum** of N match scores of **independent random** sequences.

Significance of scores ... the extreme value distribution

- ✓ There is an alternative way to consider significance using a more classical statistical framework. We can look at the **distribution** of the **maximum** of N match scores of **independent random** sequences.
- ✓ If the probability of this maximum being greater than the observed best score is small, then the observation is considered significant.

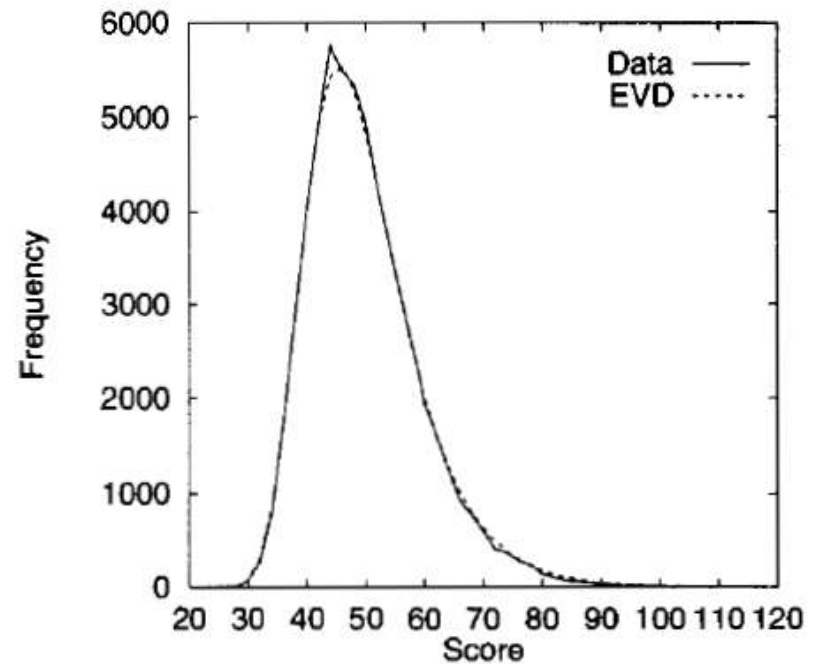
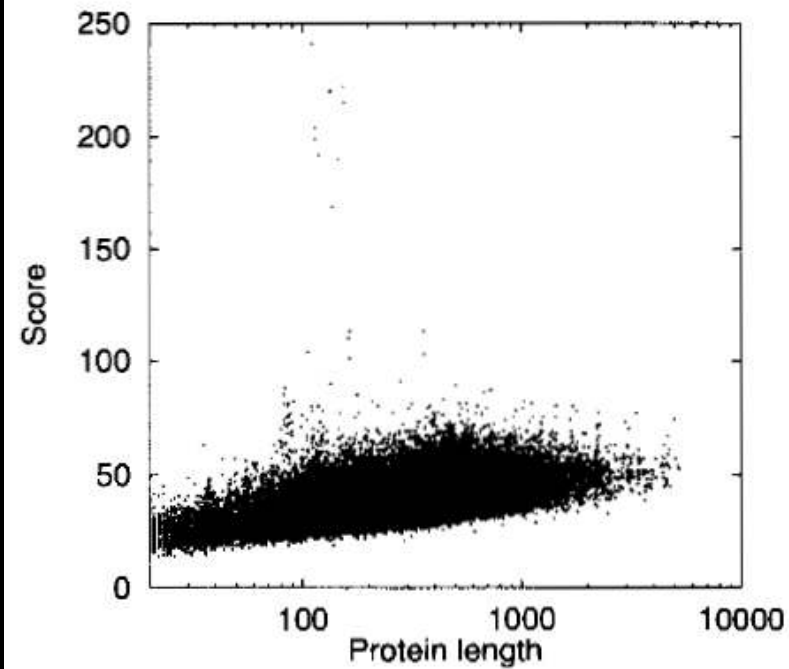
Significance of scores ... the extreme value distribution

- ✓ There is an alternative way to consider significance using a more classical statistical framework. We can look at the **distribution** of the **maximum** of N match scores of **independent random** sequences.
- ✓ If the probability of this maximum being greater than the observed best score is small, then the observation is considered significant.
- ✓ In the simple case of a fixed ungapped alignment **the score of a match** to a random sequence is the sum of many similar random variables and can thus be very well **approximated** by a **normal distribution**. The asymptotic distribution of the maximum M_N of a series of N **independent normal random variables** is known and has the form:

$$P(M_N \leq x) \approx \exp(-KNe^{\lambda(x-\mu)}) \quad (2.18)$$

For some constants K, λ . This form of distribution is known as the **extreme value distribution** or **EVD**.

Significance of scores *... the extreme value distribution*



Significance of scores ... the extreme value distribution

For local ungapped alignments, Karlin and Altschul (1990) derived the approximate **EVD distribution analytically**.

The **number of unrelated matches with score greater than S** is approximately **Poisson distributed**, with mean:

$$E(S) = Kmne^{-\lambda S} \quad (2.19)$$

where λ is the positive root of

$$\sum_{a,b} q_a q_b e^{\lambda s(a,b)} = 1 \quad (2.20)$$

Significance of scores ... the extreme value distribution

and K is a constant given by a geometrically convergent series also dependent on the q_a and $s(a,b)$.

Significance of scores ... the extreme value distribution

and K is a constant given by a geometrically convergent series also dependent on the q_a and $s(a,b)$.

This K corrects for the non-independence of possible starting points for matches. The λ is a scale parameter to convert $s(a,b)$ into natural scale.

If $s(a,b)$ were initially derived as log likelihood quantities then $\lambda = 1$, since $e^{\lambda s(a,b)} = p_{ab}/q_a q_b$

Significance of scores ... the extreme value distribution

and K is a constant given by a geometrically convergent series also dependent on the q_a and $s(a,b)$.

This K corrects for the non-independence of possible starting points for matches. The λ is a scale parameter to convert $s(a,b)$ into natural scale.

If $s(a,b)$ were initially derived as log likelihood quantities then $\lambda = 1$, since $e^{\lambda s(a,b)} = p_{ab}/q_a q_b$

The probability that there is a match of score greater than S is then

$$P(x > S) = 1 - e^{-E(S)} \quad (2.21)$$

Combining equations 2.19 and 2.21 gives a distribution of the same EVD form as 2.18, without the μ .

Significance of scores ... the extreme value distribution

In fact we **do not need** to calculate a probability, but just to use a requirement that $E(S)$ is significantly less than 1, or to put it simple we are after the requirement that:

$$S > T + \frac{\log mn}{\lambda}$$

for some fixed constant T .

Significance of scores ... the extreme value distribution

In fact we **do not need** to calculate a probability, but just to use a requirement that $E(S)$ is significantly less than 1, or to put it simple we are after the requirement that:

$$S > T + \frac{\log mn}{\lambda}$$

for some fixed constant T .



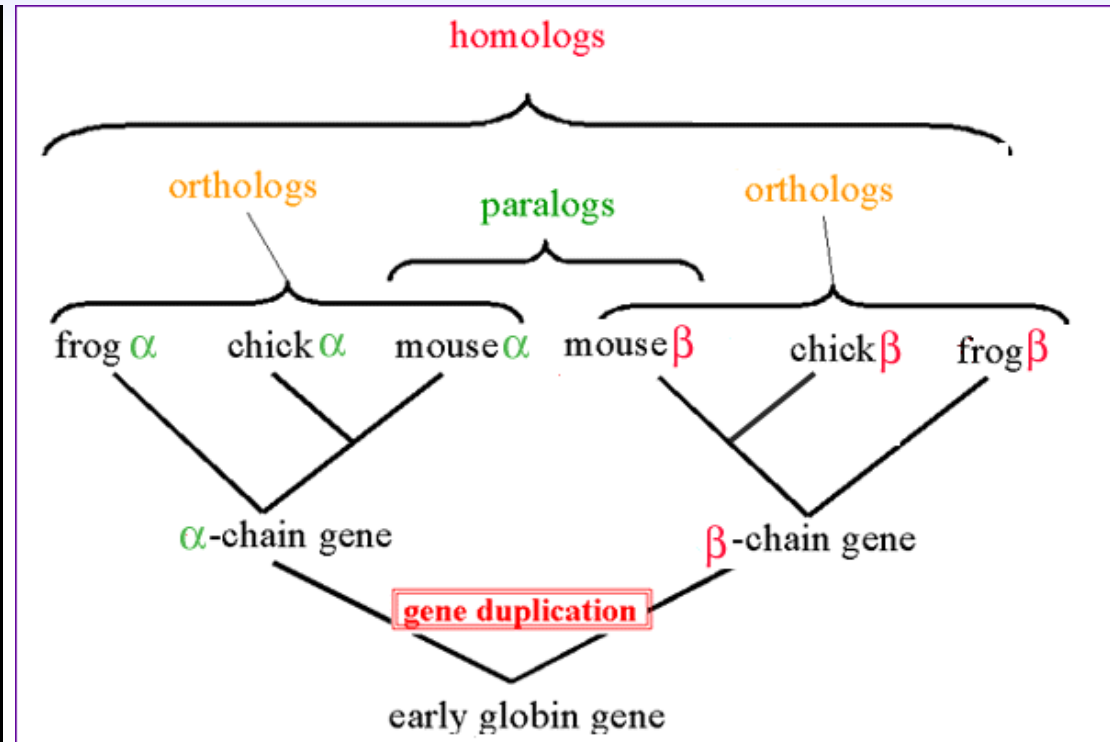
A simple **estimate** of the **number of start points** of local matches is the product of the lengths of the sequences, nm .

However this is **true if** all matches had **constant length** and all start points gave independent matches. This is **not true**, so we need a small **correction** (T) that is independent of nm and depends only on the **scoring function** s .

Biological relationships ...

Speciation

Speciation is the origin of a new species capable of making a living in a new way from the species from which it arose. As part of this process it has also acquired some barrier to genetic exchange with the parent species.



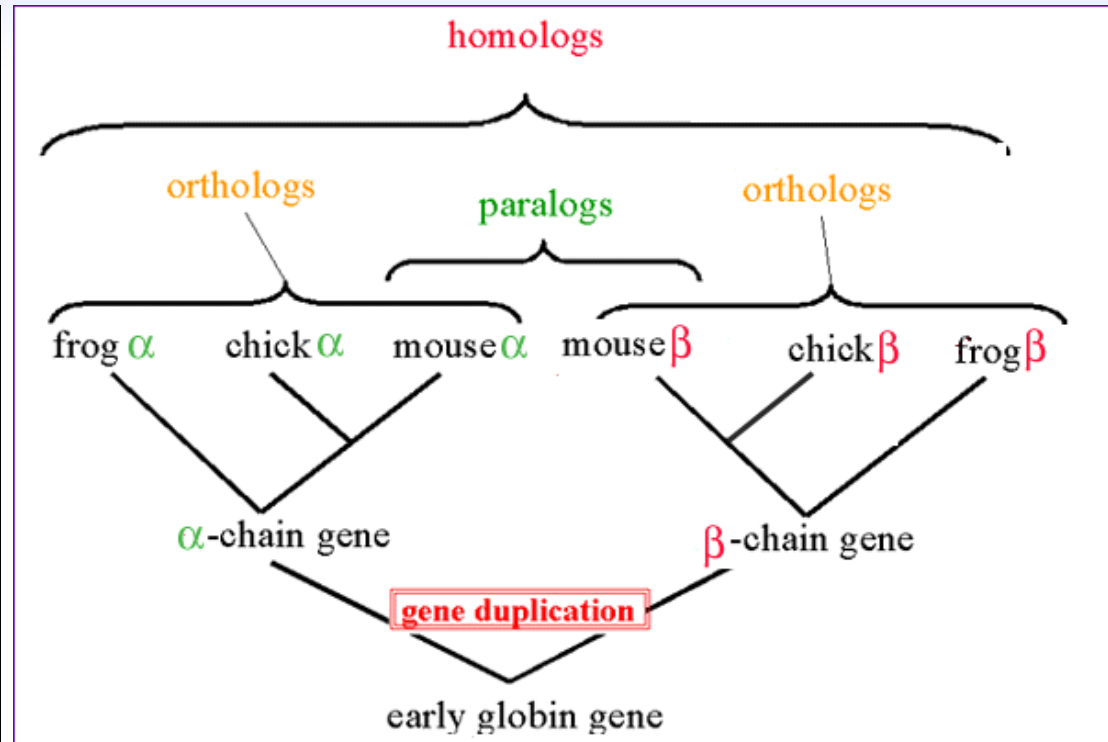
Biological relationships ...

Speciation

Speciation is the origin of a new species capable of making a living in a new way from the species from which it arose. As part of this process it has also acquired some barrier to genetic exchange with the parent species.

Homolog

A gene related to a second gene by descent from a common ancestral DNA sequence. The term, homolog, may apply to the relationship between genes separated by the event of speciation (see ortholog) or to the relationship between genes separated by the event of genetic duplication (see paralog).



Biological relationships ...

Speciation

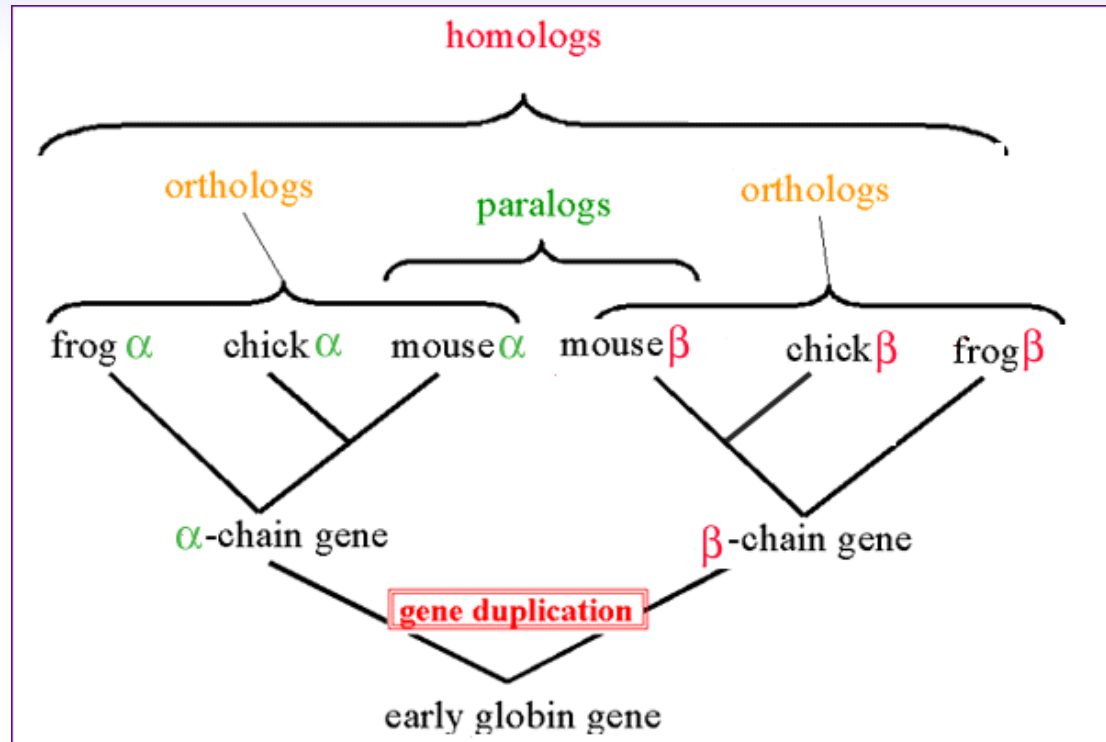
Speciation is the origin of a new species capable of making a living in a new way from the species from which it arose. As part of this process it has also acquired some barrier to genetic exchange with the parent species.

Homolog

A gene related to a second gene by descent from a common ancestral DNA sequence. The term, homolog, may apply to the relationship between genes separated by the event of speciation (see ortholog) or to the relationship between genes separated by the event of genetic duplication (see paralog).

Ortholog

Orthologs are genes in different species that evolved from a common ancestral gene by speciation. Normally, orthologs retain the same function in the course of evolution. Identification of orthologs is critical for reliable prediction of gene function in newly sequenced genomes. (See also Paralogs.)



Biological relationships ...

Speciation

Speciation is the origin of a new species capable of making a living in a new way from the species from which it arose. As part of this process it has also acquired some barrier to genetic exchange with the parent species.

Homolog

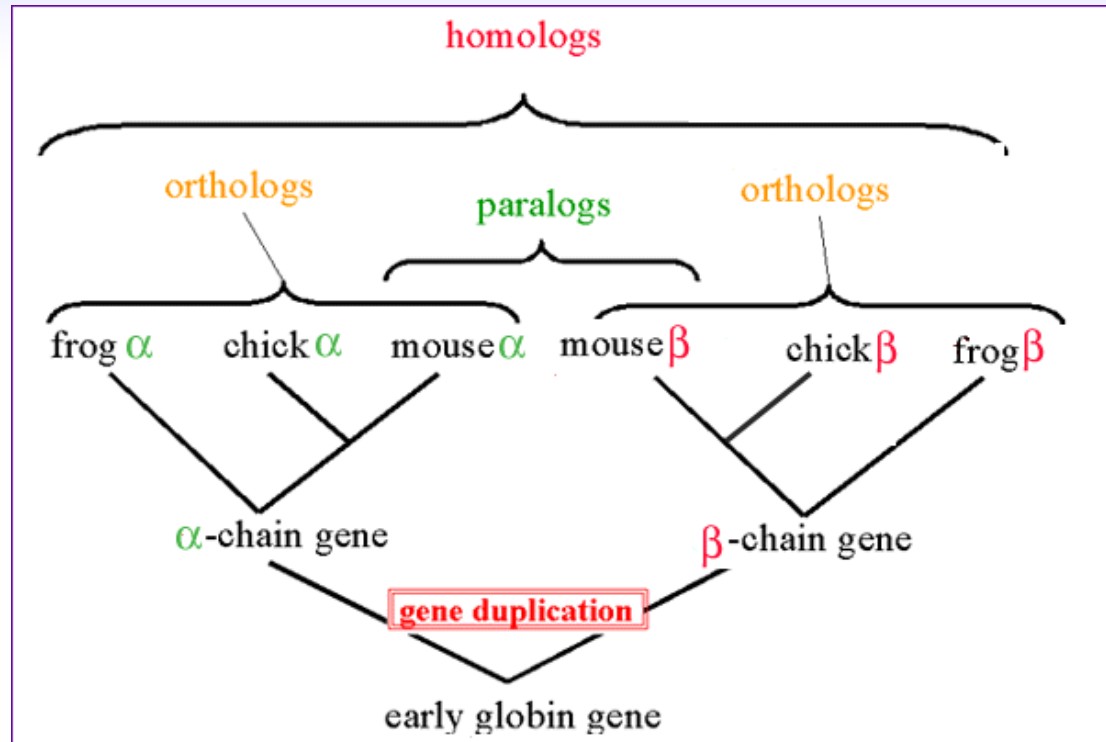
A gene related to a second gene by descent from a common ancestral DNA sequence. The term, homolog, may apply to the relationship between genes separated by the event of speciation (see ortholog) or to the relationship between genes separated by the event of genetic duplication (see paralog).

Ortholog

Orthologs are genes in different species that evolved from a common ancestral gene by speciation. Normally, orthologs retain the same function in the course of evolution. Identification of orthologs is critical for reliable prediction of gene function in newly sequenced genomes. (See also Paralogs.)

Paralog

Paralogs are genes related by duplication within a genome. Orthologs retain the same function in the course of evolution, whereas paralogs evolve new functions, even if these are related to the original one.



Significance of scores

- ✓ Searching a database of unrelated sequences result in scores following an *extreme value distribution*.
- ✓ The exact *shape* and *location* of the distribution depends on the exact nature of the *database* and the *query* sequence

Steps

1. *Align query* sequence to all sequences in *database*, note scores
2. Determine *shape* of background *distribution* (which is an extreme value distribution) from distribution of all scores
3. Use *fitted extreme-value* distribution to *predict* how many random hits to expect for any given score (the "E-value")

Values and scores

E-value

Expect value. The E-value is a parameter that describes the number of hits one can “expect” to see by chance when searching a database of a particular size. It decreases exponentially with the score (S) that is assigned to a match between two sequences. For example, an E-value of 1 assigned to a hit can be interpreted as meaning that in a database of the current size, one might expect to see one match with a similar score simply by chance. This means that the lower the E-value, or the closer it is to “0”, the higher is the “significance” of the match.

Values and scores

E-value

Expect value. The E-value is a parameter that describes the number of hits one can “expect” to see by chance when searching a database of a particular size. It decreases exponentially with the score (S) that is assigned to a match between two sequences. For example, an E-value of 1 assigned to a hit can be interpreted as meaning that in a database of the current size, one might expect to see one match with a similar score simply by chance. This means that the lower the E-value, or the closer it is to “0”, the higher is the “significance” of the match.

bit score

The value S' is derived from the raw alignment score S in which the statistical properties of the scoring system used have been taken into account. By normalizing a raw score, a “bit score” S' is attained, which has a standard set of units. Because bit scores have been normalized with respect to the scoring system, they can be used to compare alignment scores from different searches.

Values and scores

E-value

Expect value. The E-value is a parameter that describes the number of hits one can “expect” to see by chance when searching a database of a particular size. It decreases exponentially with the score (S) that is assigned to a match between two sequences. For example, an E-value of 1 assigned to a hit can be interpreted as meaning that in a database of the current size, one might expect to see one match with a similar score simply by chance. This means that the lower the E-value, or the closer it is to “0”, the higher is the “significance” of the match.

bit score

The value S' is derived from the raw alignment score S in which the statistical properties of the scoring system used have been taken into account. By normalizing a raw score, a “bit score” S' is attained, which has a standard set of units. Because bit scores have been normalized with respect to the scoring system, they can be used to compare alignment scores from different searches.

P-value

Is the probability of obtaining a test statistic at least as extreme as the one that was actually observed, assuming that the null hypothesis is true.

E-value is the average number of times in multiple testing that one expects to obtain a test statistic at least as extreme as the one that was actually observed, assuming that the null hypothesis is true. The E-value is the product of the number of tests and the p-value.

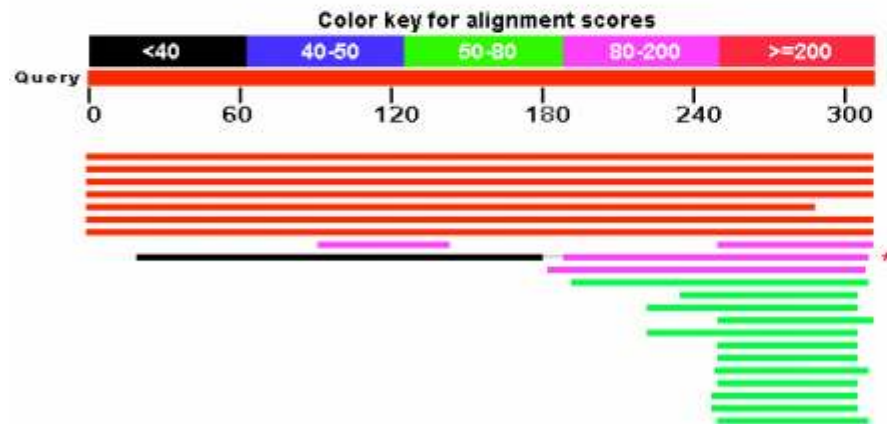
Values and scores

Example: BLAST - Pho4p (*S. cerevisiae*)

```
>gi|259146228|emb|CAY79487.1| Pho4p [Saccharomyces cerevisiae EC1118]  
MGRTTSEGIHGFVDDLEPKSSILDKVGFITVNTKRHDGREDFNEQNDELNSQEHHNSENENENEQD  
SLALDDLDRAFELVEGMDMDWMMPSHAHHSPATTATIKPRLLYSPLIHTQSAVPVTISPVLVATATSTTS  
ANKVTKNKSNSSPYLNKRRGKPGPDSATSLFELPDSVIPTPKPKPKPKQYPKVILPSNSTRRISPVTAKT  
SSSAEGVVVASESPVIAPHGSSHSRSLSKRRSSGALVDDDKRESHKHAEQARRNRLAVALHELASLIPAE  
WKQQNVSAAPSKATTVEAACRYIRHLQQNVST
```

Query (input) sequence
(Pho4p from *S. cerevisiae*)

BLAST (default parameters) ↓



Results (output) of BLAST

- The top segment displays the color key and the query based scale.
- The colored bars represent the actual HSPs. The position of each bar indicates the region of the query the HSP covers.
- The thin line (see *) indicates that the two HSPs are from the same sequence.
- Small vertical lines (not obtained here) indicate breaks, i.e., segments which are not connected in the actual alignment.

Explanation of Output of a BLAST Search:

http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/new_view.html

Values and scores

Example: BLAST - Pho4p (*S. cerevisiae*)

Results (output) of BLAST

Sequences producing significant alignments:

Accession	Description	Max score	Total score	Query coverage	E value
CAY79487.1	Pho4p [Saccharomyces cerevisiae EC1118]	640	640	100%	0.0
EDV09876.1	myc-family transcription factor [Saccharomyces cerevisiae RM1	637	637	100%	0.0
NP_116692.1	Basic helix-loop-helix (bHLH) transcription factor of the myc-fam	637	637	100%	0.0
EEU04180.1	Pho4p [Saccharomyces cerevisiae JAY291]	629	629	100%	2e-178
CAA27345.1	unnamed protein product [Saccharomyces cerevisiae]	584	584	92%	4e-165
CAA36809.1	unnamed protein product [Saccharomyces cerevisiae]	562	562	100%	2e-158
CAA36810.1	unnamed protein product [Saccharomyces cerevisiae]	561	561	100%	3e-158
1A0A_A	Chain A, Phosphate System Positive Regulatory Protein Pho4DN	126	126	19%	4e-27
ED272385.1	YFR034Cp-like protein [Saccharomyces cerevisiae AWRI1631]	102	102	16%	4e-20
XP_002553686.1	KLTH0E04664p [Lachancea thermotolerans] >emb CAR23249.1	84.0	120 *	90%	2e-14
NP_983973.2	ADL123Cp [Ashbya gossypii ATCC 10895] >gb AAS51797.2 AD	83.6	83.6	40%	3e-14
XP_002489917.1	hypothetical protein [Pichia pastoris GS115] >emb CAY67636.1	72.4	72.4	37%	6e-11
XP_445634.1	unnamed protein product [Candida glabrata] >emb CAG58545.1	68.6	68.6	22%	1e-09

Max score = highest alignment score (bit-score) between the query sequence and the database sequence segment .

Total score = sum of alignment scores of all segments from the same database sequence that match the query sequence (calculated over all segments). This score is different from the max score if several parts of the database sequence match different parts of the query sequence (see " * " in the example).

Query coverage = percent of the query length that is included in the aligned segments. This coverage is calculated over all segments (cf. total score).

E-value = number of alignments expected by chance with a particular score or better. The expect value is the default sorting metric and normally gives the same sorting order as Max score.

Values and scores

Example: BLAST - Pho4p (*S. cerevisiae*)

Results (output) of BLAST

The diagram illustrates the BLAST output with callouts for key metrics:

- Bit-score**: Points to the value 83.6 bits (205).
- E-value**: Points to the value 3e-14.
- Identity (%)**: Points to the value 61/136 (44%).
- Similarity (%)**: Points to the value 73/136 (53%).
- Gaps (%)**: Points to the value 18/136 (13%).

The BLAST output text is as follows:

```
Score = 83.6 bits (205), Expect = 3e-14, Method: Compositional matrix adjust.
Identities = 61/136 (44%), Positives = 73/136 (53%), Gaps = 18/136 (13%)

Query 184 KPKPKQYPKVLPSNSTRRISPVTAKTSSSAEGVVVASESPVIAPHGSSHSRSLSKRRSS 243
          KP P P+ ILPSN+ +R P S V+ AS+SPVI P+ + RS
Sbjct 269 KPAPG-LPRFILPSNNPQRQLPPPPSDS-----VIHASQSPVIKPNYAGKPPGFVSARSV 322

Query 244 GALVDDD-----KRESHKHAEQARRNRLAVALHELASLIPAEWKQQNVSAAPSKATT 295
          L D K+E HK AEQ RRNRL AL EL L+P E K+ + PSKATT
Sbjct 323 RTLSSGGDANTGDEFIKKEVHKVAEQGRRNRLNNAELNDLLPPELKES--AQVPSKATT 380

Query 296 VEAACRYIRHL--QQN 309
          VE AC+YIR L QQN
Sbjct 381 VELACKYIRQLTGQQN 396
```

Exercise



Find a second equal-scoring optimal alignment in the dynamic programming matrix in Figure 2.5

	H	E	A	G	A	W	G	H	E	E	
P	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80
A	-8	-2	-9	-17	-25	-33	-42	-49	-57	-65	-73
W	-16	-10	-3	-4	-12	-20	-28	-36	-44	-52	-60
H	-24	-18	-11	-6	-7	-15	-5	-13	-21	-29	-37
E	-32	-14	-18	-13	-8	-9	-13	-7	-3	-11	-19
A	-40	-22	-8	-16	-16	-9	-12	-15	-7	3	-5
E	-48	-30	-16	-3	-11	-11	-12	-12	-15	-5	2
E	-56	-38	-24	-11	-6	-12	-14	-15	-12	-9	1

HEAGAWGHE-E

--P-AW-HEAE

	H	E	A	G	A	W	G	H	E	E
P	-2	-1	-1	-2	-1	-4	-2	-2	-1	-1
A	-2	-1	5	0	5	-3	0	-2	-1	-1
W	-3	-3	-3	-3	-3	15	-3	-3	-3	-3
H	10	0	-2	-2	-2	-3	-2	10	0	0
E	0	6	-1	-3	-1	-3	-3	0	6	6
A	-2	-1	5	0	5	-3	0	-2	-1	-1
E	0	6	-1	-3	-1	-3	-3	0	6	6

JAVA Applet: Dynamic Programming Demo

BABA: Needleman & Wunsch Dynamic Programming

AntiAliasing: On Zoom: Normal Info: Calculating DP Table. Step: 17

D(i,j)		P	A	W	H	E	A	E
	0	-6	-12	-18	-24	-30	-36	-42
H	-6	0	-6	-12	-17	-23	-29	-35
E	-12	-6	0	-6	-12	-16	-22	-28
A	-18	-12	-5	0	-6	0	1	0
G	-24	0	0	0	0	0	0	0
A	-30	0	1	0	0	0	1	0
W	-36	0	0	1	0	0	0	0
G	-42	0	0	0	0	0	0	0
H	-48	0	0	0	1	0	0	0
E	-54	0	0	0	0	1	0	1
E	-60	0	0	0	0	1	0	1

String S1: HEAGAWGHEE
S1 Score Gap Penalty Array: {-6, -6, -6, -6, -6, -6, -6, -6, -6}

String S2: PAWHEAE
S2 Score Gap Penalty Array: {-6, -6, -6, -6, -6, -6}

D(3, 4) = Max {
 $D(3, 3) + sc. = 0 + -6 = -6$
 $D(2, 4) + sc. = -12 + -6 = -18$
 $D(2, 3) + sc. = -6 + 0 = -6$

s1:
s2:

<http://baba.sourceforge.net/>

Dynamic programming (Java) implementation

Cell Class

```
public class Cell {  
    private Cell prevCell;           L.1  
    private int score;  
    private int row;  
    private int col;  
}
```

Source: <http://www.ibm.com/developerworks/java/library/j-seqalign/index.html>

Dynamic Programming initialization code

```
protected void initializeScores() {  
    for (int i = 0; i < scoreTable.length; i++) {  
        for (int j = 0; j < scoreTable[i].length; j++) {  
            scoreTable[i][j].setScore(getInitialScore(i, j));  
        }  
    }  
}  
L.2  
  
protected void initializePointers() {  
    for (int i = 0; i < scoreTable.length; i++) {  
        for (int j = 0; j < scoreTable[i].length; j++) {  
            scoreTable[i][j].setPrevCell(getInitialPointer(i, j));  
        }  
    }  
}  
  
protected void initialize() {  
    for (int i = 0; i < scoreTable.length; i++) {  
        for (int j = 0; j < scoreTable[i].length; j++) {  
            scoreTable[i][j] = new Cell(i, j);  
        }  
    }  
    initializeScores();  
    initializePointers();  
  
    isInitialized = true;  
}  
  
protected abstract Cell getInitialPointer(int row, int col);  
  
protected abstract int getInitialScore(int row, int col);
```

Dynamic Programming code for filling in the table

```
protected abstract void fillInCell(Cell currentCell, Cell cell
Above,
    Cell cellToLeft, Cell cellAboveLeft);           L.3

protected void fillIn() {
    for (int row = 1; row < scoreTable.length; row++) {
        for (int col = 1; col < scoreTable[row].length; col+
+) {
            Cell currentCell = scoreTable[row][col];
            Cell cellAbove = scoreTable[row - 1][col];
            Cell cellToLeft = scoreTable[row][col - 1];
            Cell cellAboveLeft = scoreTable[row - 1][col - 1];
            fillInCell(currentCell, cellAbove, cellToLeft, cellAbov
eLeft);
        }
    }

    tableIsFilledIn = true;
}
```

DynamicProgramming.getTraceback()
method

```
abstract protected Object getTraceback();   L.4
```

LCS initialization method

```
protected int getInitialScore(int row, int col) {
    return 0;
}                                           L.5
```

LCS method for filling in a cell's score and pointer

```
protected void fillInCell(Cell currentCell, Cell cellAbove, Cell cellToLeft,
    Cell cellAboveLeft) {
    int aboveScore = cellAbove.getScore();
    int leftScore = cellToLeft.getScore();
    int matchScore;
    if (sequence1.charAt(currentCell.getCol() - 1) == sequence2
        .charAt(currentCell.getRow() - 1)) {
        matchScore = cellAboveLeft.getScore() + 1;
    } else {
        matchScore = cellAboveLeft.getScore();
    }
    int cellScore;
    Cell cellPointer;
    if (matchScore >= aboveScore) {
        if (matchScore >= leftScore) {
            // matchScore >= aboveScore and matchScore >= leftScore
            cellScore = matchScore;
            cellPointer = cellAboveLeft;
        } else {
            // leftScore > matchScore >= aboveScore
            cellScore = leftScore;
            cellPointer = cellToLeft;
        }
    } else {
        if (aboveScore >= leftScore) {
            // aboveScore > matchScore and aboveScore >= leftScore
            cellScore = aboveScore;
            cellPointer = cellAbove;
        } else {
            // leftScore > aboveScore > matchScore
            cellScore = leftScore;
            cellPointer = cellToLeft;
        }
    }
    currentCell.setScore(cellScore);
    currentCell.setPrevCell(cellPointer);
}
```

L.6

LCS traceback code

```
protected Object getTraceback() {  
    StringBuffer ICSBuf = new StringBuffer();  
    Cell currentCell = scoreTable[scoreTable.length - 1][scoreTable[0].length - 1];  
    while (currentCell.getScore() > 0) {  
        Cell prevCell = currentCell.getPrevCell();  
        if ((currentCell.getRow() - prevCell.getRow() == 1 && currentCell  
            .getCol()  
            - prevCell.getCol() == 1)  
            && currentCell.getScore() == prevCell.getScore() + 1) {  
            ICSBuf.insert(0, sequence1.charAt(currentCell.getCol() - 1));  
        }  
        currentCell = prevCell;  
    }  
  
    return ICSBuf.toString();  
}
```

L.7

Needleman-Wunsch initialization code

```
protected Cell getInitialPointer(int row,  
int col) {  
    if (row == 0 && col != 0) {  
        return scoreTable[row][col - 1];  
    } else if (col == 0 && row != 0) {  
        return scoreTable[row - 1][col];  
    } else {  
        return null;  
    }  
}  
  
protected int getInitialScore(int row,  
int col) {  
    if (row == 0 && col != 0) {  
        return col * space;  
    } else if (col == 0 && row != 0) {  
        return row * space;  
    } else {  
        return 0;  
    }  
} L.8  
}
```

Needleman-Wunsch code for filling in the table

```
protected void fillInCell(Cell currentCell, Cell cellAbove, Cell cellToLeft,
    Cell cellAboveLeft) {
    int rowSpaceScore = cellAbove.getScore() + space;
    int colSpaceScore = cellToLeft.getScore() + space;
    int matchOrMismatchScore = cellAboveLeft.getScore();
    if (sequence2.charAt(currentCell.getRow() - 1) == sequence1
        .charAt(currentCell.getCol() - 1)) {
        matchOrMismatchScore += match;
    } else {
        matchOrMismatchScore += mismatch;
    }
    if (rowSpaceScore >= colSpaceScore) {
        if (matchOrMismatchScore >= rowSpaceScore) {
            currentCell.setScore(matchOrMismatchScore);
            currentCell.setPrevCell(cellAboveLeft);
        } else {
            currentCell.setScore(rowSpaceScore);
            currentCell.setPrevCell(cellAbove);
        }
    } else {
        if (matchOrMismatchScore >= colSpaceScore) {
            currentCell.setScore(matchOrMismatchScore);
            currentCell.setPrevCell(cellAboveLeft);
        } else {
            currentCell.setScore(colSpaceScore);
            currentCell.setPrevCell(cellToLeft);
        }
    }
}
}
```

Traceback code used for both Needleman-Wunsch and Smith-Waterman

```
protected Object getTraceback() {
    StringBuffer align1Buf = new StringBuffer();
    StringBuffer align2Buf = new StringBuffer();
    Cell currentCell = getTracebackStartingCell();
    while (traceBackIsNotDone(currentCell)) {
        if (currentCell.getRow() - currentCell.getPrevCell().getRow() == 1) {
            align2Buf.insert(0, sequence2.charAt(currentCell.getRow() - 1));
        } else {
            align2Buf.insert(0, '-');
        }
        if (currentCell.getCol() - currentCell.getPrevCell().getCol() == 1) {
            align1Buf.insert(0, sequence1.charAt(currentCell.getCol() - 1));
        } else {
            align1Buf.insert(0, '-');
        }
        currentCell = currentCell.getPrevCell();
    }

    String[] alignments = new String[] { align1Buf.toString(),
        align2Buf.toString() };

    return alignments;
}
protected abstract boolean traceBackIsNotDone(Cell currentCell);
protected abstract Cell getTracebackStartingCell();
```

L.10

Needleman-Wunsch traceback code

```
protected boolean traceBackIsNotDone(Cell currentCell) {           L.11  
    return currentCell.getPrevCell() != null;  
}  
  
protected Cell getTracebackStartingCell() {  
    return scoreTable[scoreTable.length - 1][scoreTable[0].length - 1];  
}
```

Smith-Waterman initialization

```
protected Cell getInitialPointer(int row, int col) {  
    return null;  
}  
protected int getInitialScore(int row, int col) {  
    return 0;  
}                                     L.12
```

Smith-Waterman code for filling in a cell

```
protected void fillInCell(Cell currentCell, Cell cellAbove, Cell cellToLeft,
    Cell cellAboveLeft) {
    int rowSpaceScore = cellAbove.getScore() + space;
    int colSpaceScore = cellToLeft.getScore() + space;
    int matchOrMismatchScore = cellAboveLeft.getScore();
    if (sequence2.charAt(currentCell.getRow() - 1) == sequence1
        .charAt(currentCell.getCol() - 1)) {
        matchOrMismatchScore += match;
    } else {
        matchOrMismatchScore += mismatch;
    }
    if (rowSpaceScore >= colSpaceScore) {
        if (matchOrMismatchScore >= rowSpaceScore) {
            if (matchOrMismatchScore > 0) {
                currentCell.setScore(matchOrMismatchScore);
                currentCell.setPrevCell(cellAboveLeft);
            }
        } else {
            if (rowSpaceScore > 0) {
                currentCell.setScore(rowSpaceScore);
                currentCell.setPrevCell(cellAbove);
            }
        }
    } else {
        if (matchOrMismatchScore >= colSpaceScore) {
            if (matchOrMismatchScore > 0) {
                currentCell.setScore(matchOrMismatchScore);
                currentCell.setPrevCell(cellAboveLeft);
            }
        } else {
            if (colSpaceScore > 0) {
                currentCell.setScore(colSpaceScore);
                currentCell.setPrevCell(cellToLeft);
            }
        }
    }
    if (currentCell.getScore() > highScoreCell.getScore()) {
        highScoreCell = currentCell;
    }
}
```

L.13

Smith-Waterman traceback code

```
protected boolean traceBackIsNotDone(Cell currentCell) {  
    return currentCell.getScore() != 0;  
}  
  
@Override  
protected Cell getTracebackStartingCell() {  
    return highScoreCell;  
}
```

L.14

BioJava code sequence alignment code

```
public static void main(String[] args) throws Exception {
    // The alphabet of the sequences. For this example DNA is chosen.
    FiniteAlphabet alphabet = (FiniteAlphabet) AlphabetManager
        .alphabetForName("DNA");
    // Use a substitution matrix with equal scores for every match and every
    // replace.
    int match = 1;
    int replace = -1;
    SubstitutionMatrix matrix = new SubstitutionMatrix(alphabet, match,
        replace);
    // Firstly, define the expenses (penalties) for every single operation.
    int insert = 2;
    int delete = 2;
    int gapExtend = 2;
    // Global alignment.
    SequenceAlignment aligner = new NeedlemanWunsch(match, replace, insert,
        delete, gapExtend, matrix);
    Sequence query = DNATools.createDNASequence("GCCCTAGCG", "query");
    Sequence target = DNATools.createDNASequence("GCGCAATG", "target");
    // Perform an alignment and save the results.
    aligner.pairwiseAlignment(query, // first sequence
        target // second one
    );
    // Print the alignment to the screen
    System.out.println("Global alignment with Needleman-Wunsch:\n"
        + aligner.getAlignmentString());

    // Perform a local alignment from the sequences with Smith-Waterman.
    aligner = new SmithWaterman(match, replace, insert, delete, gapExtend,
        matrix);
    // Perform the local alignment.
    aligner.pairwiseAlignment(query, target);
    System.out.println("\nLocal alignment with Smith-Waterman:\n"
        + aligner.getAlignmentString());
}
```

L.15

Output

Global alignment with Needleman-Wunsch:

Length:9
Score:-0.0
Query:query,Length:9
Target:target,Length:8

```
Query:  1 gccctagcg 9
          || | | |
Target:  1 gcgc-aatg 8
```

Local alignment with Smith-Waterman:

Length:3
Score:3.0
Query:query,Length:9
Target:target,Length:8

```
Query:  7 gcg 9
          |||
Target:  1 gcg 3
```

L.16