Moreover, the circular logic

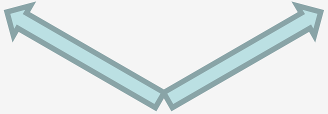
- How do we know what is the right distance without a “good” alignment?
- And how do we construct a good alignment without knowing what substitutions were made previously?

```
ATGCGT--GCAAGT

--GCGGCGGCAGTT
```

```
ATGCGT--GCAAGT-

--GCGGCGGC-AGTT
```



Which is better?

Less gaps but 7matches

More gaps but 8 matches

Substitution matrices for proteins

We need to compare, DNA or protein sequences, estimate their distances etc (e.g. Helpful for inferring molecular function by finding similarity to a sequence with known function).

For that purpose we need: “Good alignment” of sequences.

For that purpose we need: A measure for judging the quality of an alignment in relation to other possible alignments, a scoring system.

We use additive scoring systems:

Look at each position of a given alignment, and assign a score for the "quality of the match" at this position (forget about gaps for now). The total (or *cumulative*) score is obtained by adding the scores for the individual positions.

Simple example: Two DNA sequences; score for a match: $+1$, score for a mismatch -1.

E.g.:	a	a	g	t	t	t	c	t	t	g
	a	a	a	c	t	c	c	c	t	g
Individual scores:	1	1	-1	-1	1	-1	1	-1	1	1

$\Longrightarrow$ Cumulative score: $6 - 4 = 2$

Maybe more realistic: score for a match: $+1$, score for a transition: $-1/2$, score for a transversion: -1. Cumulative score in that case: $6 - 2 = 4$.

Scoring matrices

The scores for the individual positions can be displayed in a so-called **substitution matrix** (also called **scoring matrix**). This is a usually symmetrical 4×4 (DNA) resp. 20×20 (protein) matrix which has as entry (i, j) the score that we assign if at a position the nucleotides resp. the amino acids i and j are aligned.

E.g. for the second example from the last slide:

$$S = \begin{pmatrix} s_{a,a} & s_{a,c} & s_{a,g} & s_{a,t} \\ s_{c,a} & s_{c,c} & s_{c,g} & s_{c,t} \\ s_{g,a} & s_{g,c} & s_{g,g} & s_{g,t} \\ s_{t,a} & s_{t,c} & s_{t,g} & s_{t,t} \end{pmatrix} = \begin{pmatrix} 1 & -1 & -1/2 & -1 \\ -1 & 1 & -1 & -1/2 \\ -1/2 & -1 & 1 & -1 \\ -1 & -1/2 & -1 & 1 \end{pmatrix}$$

How can we find a biologically sensible scoring matrix?

For DNA sequences: simple scoring matrices (like the one presented) are often effective.

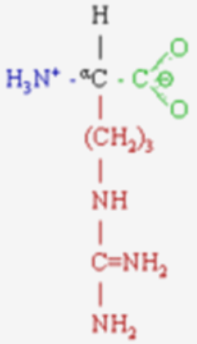
⟶ Usually, no need to worry.

A practical aspect as well, since we don’t study very diverged DNA sequences

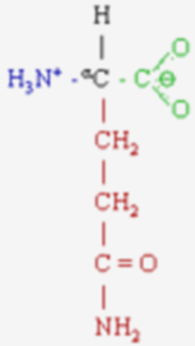
For protein sequences: some substitutions are clearly more likely to occur than others (presumably due to similar chemical properties of the amino acids involved); e.g. isoleucine for valine, serine for threonine, so-called *conservative substitutions*.

We get considerably better alignments if we take this into account.

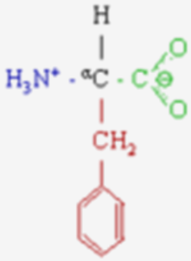
⟶ Use scoring matrices that are derived by statistical analysis of protein data.



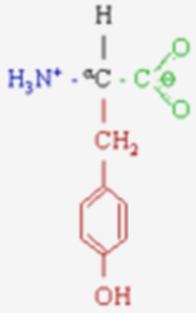
Arginine
(Arg / R)



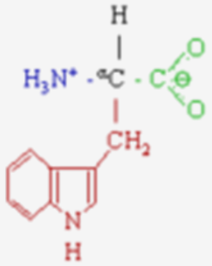
Glutamine
(Gln / Q)



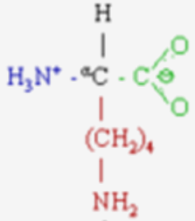
Phenylalanine
(Phe / F)



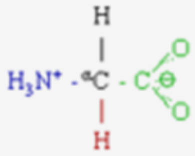
Tyrosine
(Tyr / Y)



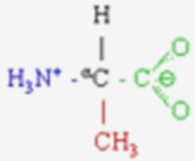
Tryptophan
(Trp, W)



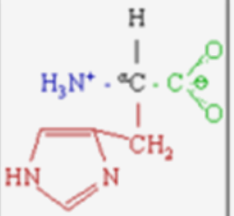
Lysine
(Lys / K)



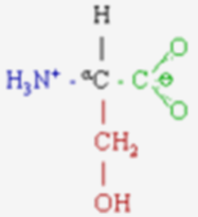
Glycine
(Gly / G)



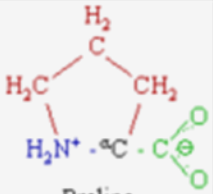
Alanine
(Ala / A)



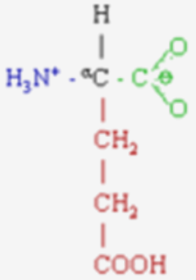
Histidine
(His / H)



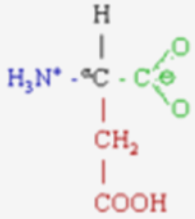
Serine
(Ser / S)



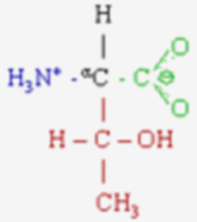
Proline
(Pro / P)



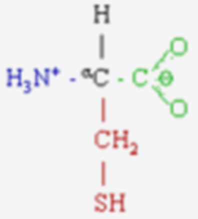
Glutamic Acid
(Glu / E)



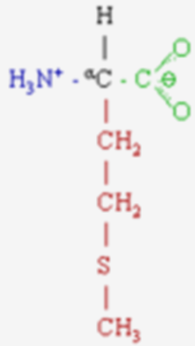
Aspartic Acid
(Asp / D)



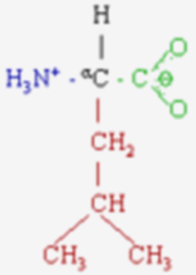
Threonine
(Thr / T)



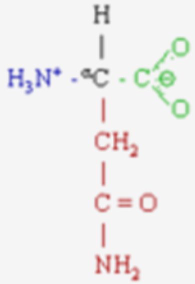
Cysteine
(Cys / C)



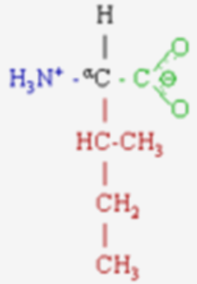
Methionine
(Met / M)



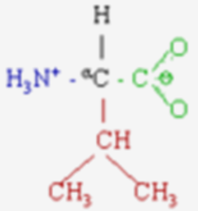
Leucine
(Leu / L)



Asparagine
(Asn / N)



Isoleucine
(Ile / I)



Valine
(Val / V)

Biologically sensible scoring matrices for proteins

Specifications:

- *identical amino acids should be given greater score than any substitution;*
- *conservative substitutions should be given greater score than non-conservative ones;*
- *different sets of values* may be desired for comparing very similar sequences (*e.g. homologies in mouse and rat*) as opposed to highly divergent sequences (*e.g. homologies in mouse and yeast*); i.e. *we usually want our scoring matrices to take into account the evolutionary distance between the sequences involved!*

Scoring matrices, overview

There are two frequently used approaches to finding substitution matrices. They lead to

1) **the PAM family of substitution matrices**

   Main concepts used:

   Markov chains and phylogenetic trees

   (for "fitting" an evolutionary model)

   log-likelihood ratios

   (for getting a scoring matrix from an estimated transition matrix)

2) **the BLOSUM family of substitution matrices**

   Main concept used:

   log-likelihood ratios

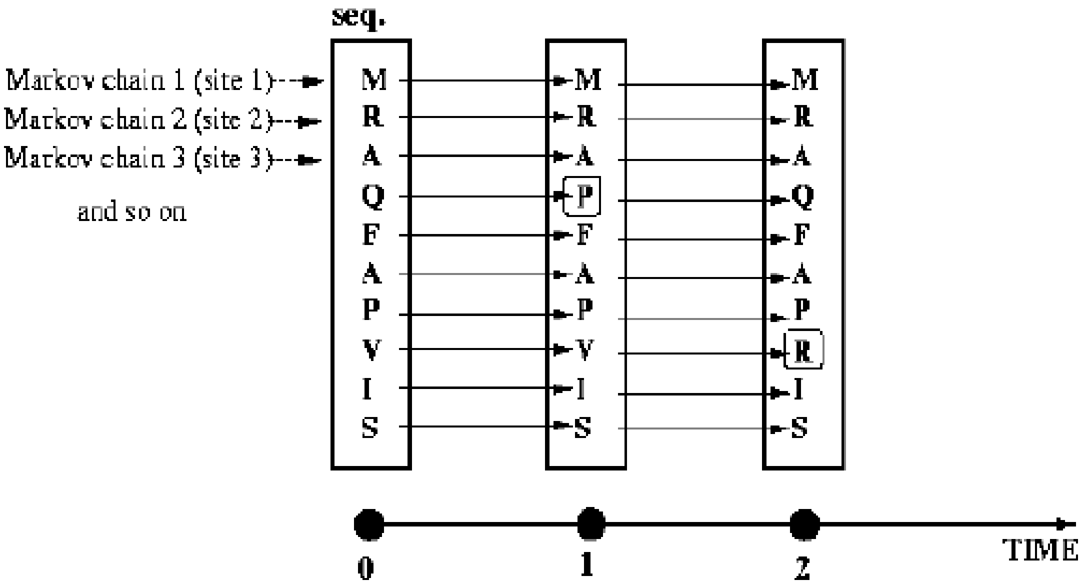
   (for getting a scoring matrix from a matrix of estimated substitution probabilities)

PAM matrices

In fact, there are two types of matrices involved here:

- a PAM **Markov transition** matrix
  (= *the table of estimated transition probabilities for the underlying evolutionary model*);
- a PAM **substitution** matrix
  (= *the table of scores for all possible pairs of amino acids*).

Underlying model: Each site in the sequence evolves *according to a Markov chain*, and *independently* of the other sites.



All the Markov chains have the *same* transition matrix P (matrix with dimension 20×20).

Dayhoff et al. (1978) *estimated* the one-step transition matrix P from protein sequence data.

How...?

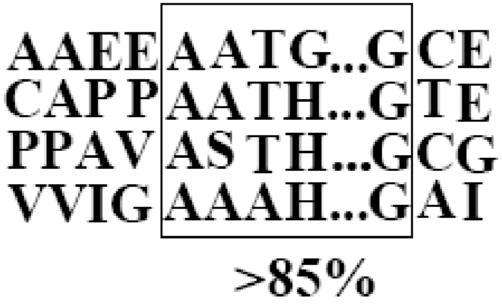
Construction of a PAM1 transition matrix

A **PAM1 transition matrix** is the Markov transition matrix applying for a time period over which we expect 1% *of the amino acids to undergo accepted point mutations.* The steps involved in the estimation:

- *Align protein sequences* that are at least 85 % identical.
- Reconstruct phylogenetic trees and *infer ancestral sequences*.
- *Count the amino acid replacements* that occurred along the trees (i.e. count mutations accepted by natural selection).
- Use these counts to *estimate probabilities* for the replacements.

The first step was to find reliable data.

Dayhoff et al. (1978) used ungapped multiple alignments of certain well-conserved regions from closely related proteins.
(71 *groups of proteins, all in all* 1572 *changes.*)



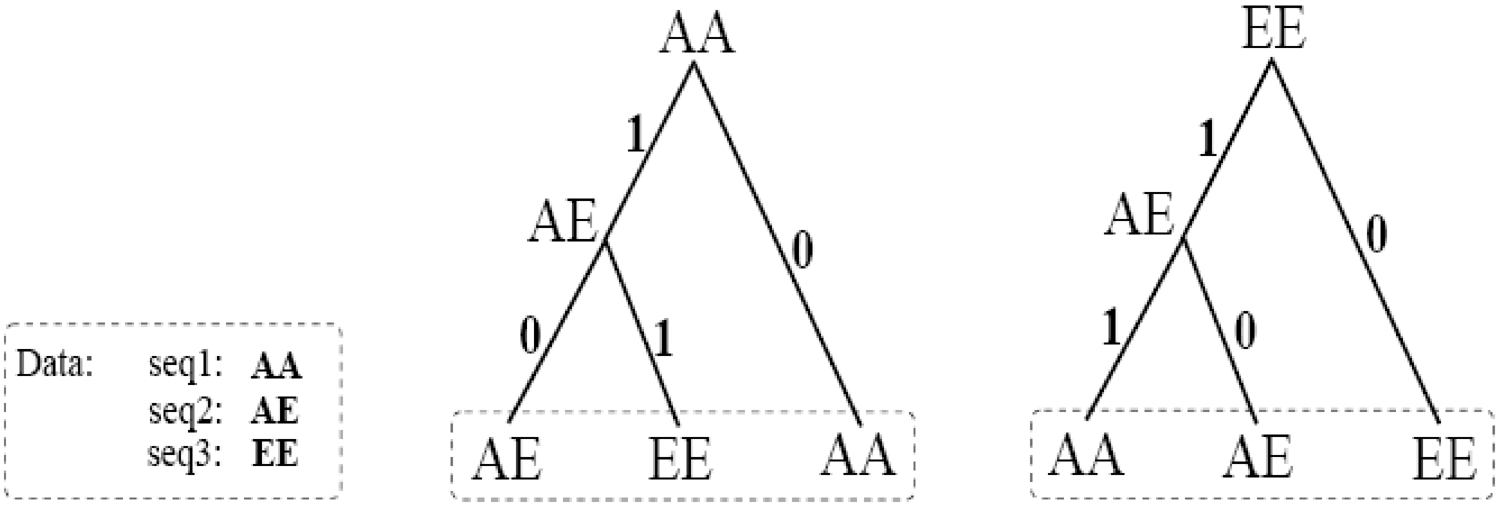
```
AAEE AATG...G CE
CAPP AATH...G TE
PPAV ASTH...G CG
VVIG AAAH...G AI
       >85%
```

In any block, any two sequences did not differ more than 15%.
(*The idea was to keep the number of sites that have encountered several changes low.*)

These aligned regions then were used to infer the underlying evolutionary tree[s] (there might be more than one).

Maximum parsimony was used to infer the *underlying evolutionary tree[s]* and *ancestral sequences*.

A *most parsimonious* tree is a tree structure such that the total number of substitutions across the tree is minimal. Ex:



All kinds of amino acid substitutions that occurred along the tree[s] were then counted.

For example, in each tree above substitutions between A and E occurred 2 *times.*

Why do we use trees?

To avoid overcounting!

Our count might be biased by closely related sequences that are overrepresented in our database.

Trees ⟹ sequences are grouped in the “right” way (in general: very similar sequences succeed one another in the tree)

⟹ we have mainly transitions between these sequences, and only a few transitions to other, more different sequences, so the corresponding substitutions do not get an unnatural importance.

Suppose that the amino acids are numbered from 1 to 20. (For simplicity, we assume that there was only one tree).

Let $A_{j,k}$ be the number of times substitutions from j to k were observed in the tree.

Result: a 'count' matrix.

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} & ... & A_{1,19} & A_{1,20} \\ A_{2,1} & A_{2,2} & ... & A_{2,19} & A_{2,20} \\ ... & ... & ... & ... & ... \\ A_{19,1} & A_{19,2} & ... & A_{19,19} & A_{19,20} \\ A_{20,1} & A_{20,2} & ... & A_{20,19} & A_{20,20} \end{pmatrix}$$

This matrix was then used to *estimate a Markov transition matrix.*

First, for any pair (j, k) define

$$a_{j,k} := \frac{A_{j,k}}{\sum_{m=1}^{20} A_{j,m}}.$$

This is the *observed relative frequency* for the substitution $j \to k$.

The $a_{j,k}$'s are *estimated probabilities.*

These probabilities were then *scaled* in a certain way:
For $j \neq k$,

$$p_{j,k} := c \cdot a_{j,k}$$

and

$$p_{j,j} := 1 - \sum_{k \neq j} c \cdot a_{j,k},$$

where the scaling constant c is sufficiently small so that $p_{j,j} \geq 0$ for all j.

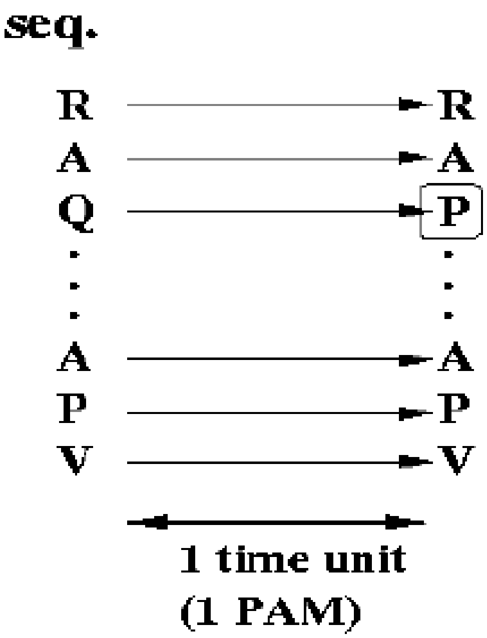
... but why the scaling factor c?

Why the scaling factor c?

To account for the evolutionary distance!

Goal: to choose a value of c which renders a transition matrix that is *'useful for short evolutionary periods'*.

More exactly: choose a value of c, such that 1% *of the amino acids are expected to undergo accepted point mutations* during one time unit.



Expected proportion
of amino acid changes
= 1 %

Such a time unit is called **an evolutionary distance of 1 PAM**.

How to choose the c?

To determine c, it *suffices to consider one of the sites* in the sequence, i.e. we consider only one of the parallel Markov chains.

Let $Z_n =$ *the amino acid present at the site considered at time* n, $n \geq 0$. (hence $1 \leq Z_n \leq 20$, since the AA's are coded as 1 to 20).

The probability that the site will change after 1 PAM time unit (i.e. after one step) is given by

$$\mathbf{P}(Z_1 \neq Z_0) = \sum_{j=1}^{20} \mathbf{P}(Z_0 = j, Z_1 \neq j)$$

$$= \sum_{j=1}^{20} \mathbf{P}(Z_1 \neq j | Z_0 = j) \cdot \mathbf{P}(Z_0 = j) \approx \sum_{j=1}^{20} \mathbf{P}(Z_1 \neq j | Z_0 = j) \cdot q_j,$$

where q_j is the *observed frequency of the amino acid no.* j in the original blocks of aligned proteins.

One wants the probability that the site will change after 1 PAM to be equal to 0.01. (*That implies an average change of* 1%.)

$$\begin{aligned}
0.01 &= \sum_{j=1}^{20} \mathbf{P}(Z_1 \neq j | Z_0 = j) \cdot q_j \\
&= \sum_{j=1}^{20} \Big(\sum_{k \neq j} \mathbf{P}(Z_1 = k | Z_0 = j) \Big) \cdot q_j \\
&\approx \sum_{j=1}^{20} \Big(\sum_{k \neq j} p_{j,k} \Big) \cdot q_j \\
&= \sum_{j=1}^{20} \Big(\sum_{k \neq j} c \cdot a_{j,k} \Big) \cdot q_j \\
&= c \cdot \sum_{j=1}^{20} \sum_{k \neq j} q_j \cdot a_{j,k}.
\end{aligned}$$

That is, we want

$$0.01 = c \cdot \sum_{j=1}^{20} \sum_{k \neq j} q_j \cdot a_{j,k}.$$

Therefore, using the estimated probabilities q_j and $a_{j,k}$, just put

$$c = \frac{0.01}{\sum_{j=1}^{20} \sum_{k \neq j} q_j \cdot a_{j,k}}.$$

Thus, with this choice for c, the *PAM transition matrix* is obtained ('one-step', i.e. for the evolutionary distance of 1 PAM) .

How can this transition matrix be turned into a scoring matrix?

How can the transition matrix be turned into a scoring matrix?

Consider two given protein sequences $\boldsymbol{s} = a_1 a_2 \cdots a_n$ and $\boldsymbol{s'} = b_1 b_2 \cdots b_n$ (at a evolutionary distance of 1 PAM, say).

The score for aligning $\boldsymbol{s}$ with $\boldsymbol{s'}$ is generated *by comparing two different hypothesis* H_0 and H_A:

- H_0: *$\boldsymbol{s}$ and $\boldsymbol{s'}$ are not evolutionarily related* *(i.e. a chance alignment).*

- H_A: *$\boldsymbol{s}$ and $\boldsymbol{s'}$ are evolutionarily related* *(i.e. $\boldsymbol{s'}$ depends on $\boldsymbol{s}$ via the Markov model).*

Under H_0, we have a chance alignment

$$\begin{array}{ll} \boldsymbol{s}: & a_1a_2\cdots a_n \\ \boldsymbol{s'}: & b_1\,b_2\cdots b_n \end{array}$$

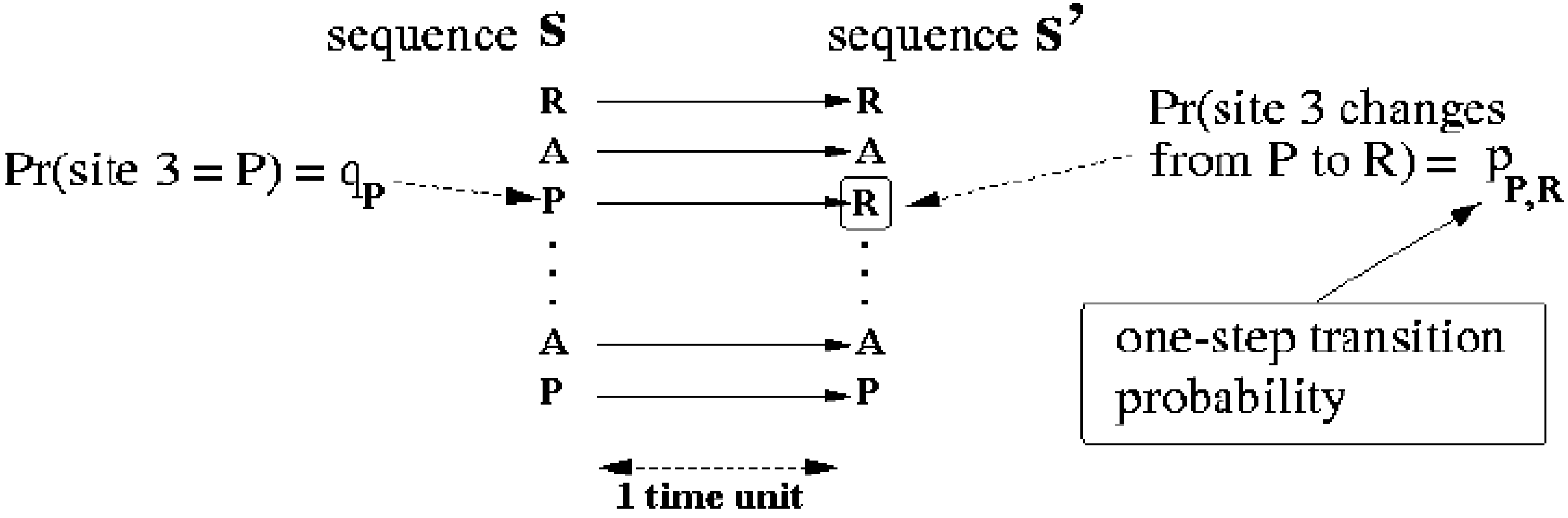
That is, all sites in both sequences are randomly generated, all sites independent of each other.

Amino acid j appears with probability q_j.

The probability for getting this *chance* alignment is equal to

$$\begin{aligned} \mathbf{P}_{H_0}(\textit{the alignment}) &= \Big(\prod_{i=1}^{n} q_{a_i}\Big) \cdot \Big(\prod_{i=1}^{n} q_{b_i}\Big) \\ &= \prod_{i=1}^{n} (q_{a_i} \cdot q_{b_i}). \end{aligned}$$

Under H_A, the sites in the sequences are dependent, according to the Markov model described earlier.


sequence S
sequence S'
R
A
P
A
P
R
A
R
A
P
Pr(site 3 = P) = q_P
Pr(site 3 changes from P to R) = p_{P,R}
one-step transition probability
1 time unit

Example: $\mathbf{P}_{H_A}(\textit{align P and R in a given site}) = q_P \cdot p_{P,R}$.

Since the different sites evolve independently of each other, we get

$$\mathbf{P}_{H_A}(\textit{the alignment}) = \prod_{i=1}^{n} (q_{a_i} \cdot p_{a_i,b_i}).$$

In principle, we want our score to reflect the 'chance' (or the *odds*) that with $\boldsymbol{s}$ and $\boldsymbol{s'}$ we have aligned evolutionarily related sequences (i.e. basically we want a high score if the odds are high that we have aligned related sequences).

A natural choice for the score is then a comparison of the probabilities under H_A and H_0, respectively:

The **likelihood ratio**:

$$\begin{aligned}\text{alignment score} &= \frac{\mathbf{P}_{H_A}(\textit{the alignment})}{\mathbf{P}_{H_0}(\textit{the alignment})} \\ &= \frac{\prod_{i=1}^{n}(q_{a_i} \cdot p_{a_i,b_i})}{\prod_{i=1}^{n}(q_{a_i} \cdot q_{b_i})} \\ &= \prod_{i=1}^{n} \frac{q_{a_i} \cdot p_{a_i,b_i}}{q_{a_i} \cdot q_{b_i}} = \prod_{i=1}^{n} \frac{p_{a_i,b_i}}{q_{b_i}}.\end{aligned}$$

Or, equivalently, but better for theoretical reasons, one can use the **log likelihood ratio** (Dayhoff et al.: “the **log odds ratio**”):

$$\begin{aligned}\text{alignment score} &= \log\Big(\frac{\mathbf{P}_{H_A}(\textit{the alignment})}{\mathbf{P}_{H_0}(\textit{the alignment})}\Big)\\ &= \log\Big(\prod_{i=1}^{n}\frac{p_{a_i,b_i}}{q_{b_i}}\Big)\\ &= \sum_{i=1}^{n}\log\left(\frac{p_{a_i,b_i}}{q_{b_i}}\right).\end{aligned}$$

The entry (a,b) in the **PAM substitution matrix** is then of the form

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

(or rounded to the nearest integer for convenience).

Commonly multiplied by a power of 10 to deal with decimals

Due to the logarithm, we have obtained an *additive* scoring system in a natural way:

alignment:

$\boldsymbol{s}$: $a_1 a_2 \cdots a_n$

$\boldsymbol{s'}$: $b_1\, b_2 \cdots b_n$

$$\boxed{\text{Total score: } S(alignment) = \textstyle\sum_{i=1}^{n} S_{a_i,b_i}.}$$

***

Adding the scores for each position is equivalent to *multiplying the probabilities* (due to the logarithm)!

$$\begin{aligned} S(alignment) &= \log\left(\frac{\mathbf{P}_{H_A}(the\ alignment)}{\mathbf{P}_{H_0}(the\ alignment)}\right) \\ &= \log\left(\frac{(q_{a_1}\cdot p_{a_1,b_1})\cdot(q_{a_2}\cdot p_{a_2,b_2})\cdots(q_{a_n}\cdot p_{a_n,b_n})}{(q_{a_1}\cdot q_{b_1})\cdot(q_{a_2}\cdot q_{b_2})\cdots(q_{a_n}\cdot q_{b_n})}\right) \\ &= \sum_{i=1}^{n}\log\left(\frac{p_{a_i,b_i}}{q_{b_i}}\right) = \sum_{i=1}^{n} S_{a_i,b_i}. \end{aligned}$$

PAMn substitution matrix?

For sequences having an evolutionary distance of n PAM units.

Careful: "n PAM units" does not mean that we expect n% of the amino acids to differ... because substitutions can occur at the same site many times!!

Let P be the 1 PAM transition matrix. As always with Markov chains: the n-step transition probabilities $p_{a,b}^{(n)}$ are given as the entries in

$$P^n.$$

The scores are

$$S_{a,b}^{(n)} = \log\left(\frac{p_{a,b}^{(n)}}{q_b}\right).$$

A sample substitution matrix

```
134 LQQGELDLVMTSDILPRSELHYSPMFDFEVRLVLAPDHPLASKTQITPEDLASETLLI
    |     |||         |       |        ||||||    |    || ||
137 LDSNSVDLVLMGVPPRNVEVEAEAFMDNPLVVIAPPDHPLAGERAISLARLAEETFVM
```

D:D = +6

D:R = -2

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

BLOSUM62

BLOSUM – Brief Overview

- Based purely on counts and alignment
- BLOSUM65 < BLOSUM45 in evolutionary distance

a set of sequences $\mathrm{S} = \mathrm{S}_1 \ldots \mathrm{S}_k$, where $\mathrm{S}_\mathrm{i} = \mathrm{s}_{\mathrm{i}1} \ldots \mathrm{s}_{\mathrm{in}}$,

s_{ij} is the j-th amino acid in sequence S_i.

The probability of observing each amino acid X =p(X)

$$\mathrm{p(X)} = \sum_{i=1}^{k} c(X_\mathrm{j}, \mathrm{S}_\mathrm{i}) \,/ \sum_{j=1}^{20} \sum_{i=1}^{k} c(X_\mathrm{j}, \mathrm{S}_\mathrm{i})$$

where $c(X_j, S_i)$ is the count of amino acid X_j in sequence S_i

$$P(X_l, X_m \mid random) = 2\,p(X_l).p(X_m) \text{ or } p(X_l)^2, if \;\; l = m$$

In general ...log-odds for alignment

$P(X_l, X_m \mid data) = \#\text{ of } X_l, X_m$ combinations/all possible pairwise combinations

finally take the log2-odds likelihood ratio and multiply by 2 for easy representation

Note: all possible pairwise combinations=k.$\{n(n-1)/2\}$

Blosum does use pseudo count to accomadate for combinations not observed:

add 1 in numerator, add 210 (20*19+20) in denominator

ie $P(\mathrm{X}_1, \mathrm{X}_\mathrm{m} \mid data) = \dfrac{count(\mathrm{X}_1, \mathrm{X}_\mathrm{m})}{\mathrm{k}.\{n(n-1)/2\} + 210}$

$$\lambda.\log \frac{P(\mathrm{X}_1, \mathrm{X}_\mathrm{m} \mid data)}{P(\mathrm{X}_1, \mathrm{X}_\mathrm{m} \mid random)} = subs.matrix$$

λ being a scaling factor for representation

BLOSUM vs PAM vs others?

BLOSUM:

Based on a range of evolutionary periods -- Each matrix constructed separately

Indirectly accounts for interdependence of residues

Range of sequences, range of replacements

PAM

Based on extrapolation from a short evolutionary period -- Errors in PAM1 are magnified through PAM250

Assumes Markov process – too much extrapolation

Rare replacements too infrequent to be represented accurately

Others

Incorporation of secondary structure/structural alignments

Use of structural alignments

Transmemberane protein-specific matrices

How do we use the matrices for sequence alignment?

AGGCTATCACCTGACCTCCAGGCCGATGCCC
TAGCTATCACGACCGCGGTCGATTTGCCCGAC

⇩

-**AG**G**CTATCAC**CT**GACC**T**C**CA**GG**C**CGA**--**TGCCC**---
T**AG**-**CTATCAC**--**GACC**GC**C**--**GG**T**CGA**TT**TGCCC**GAC

Definition

Given two strings $x = x_1x_2...x_M$, $y = y_1y_2...y_N$,

An alignment of two sequences ***x*** and ***y*** is an arrangement of ***x*** and ***y*** by position, where ***a*** and ***b*** can be padded with gap symbols to achieve the same length.

Why do we bother aligning?

- Finding important molecular regions that are conserved across species
- Given a new sequence, infer its function based on similarity to another Seq.
- Determining the evolutionary constraints at work
- Mutations in a population or a family of genes
- Find similar looking sequences in a database
- Inferring the secondary or tertiary structure of a sequence of interest -Molecular modeling using a template (homology modeling)

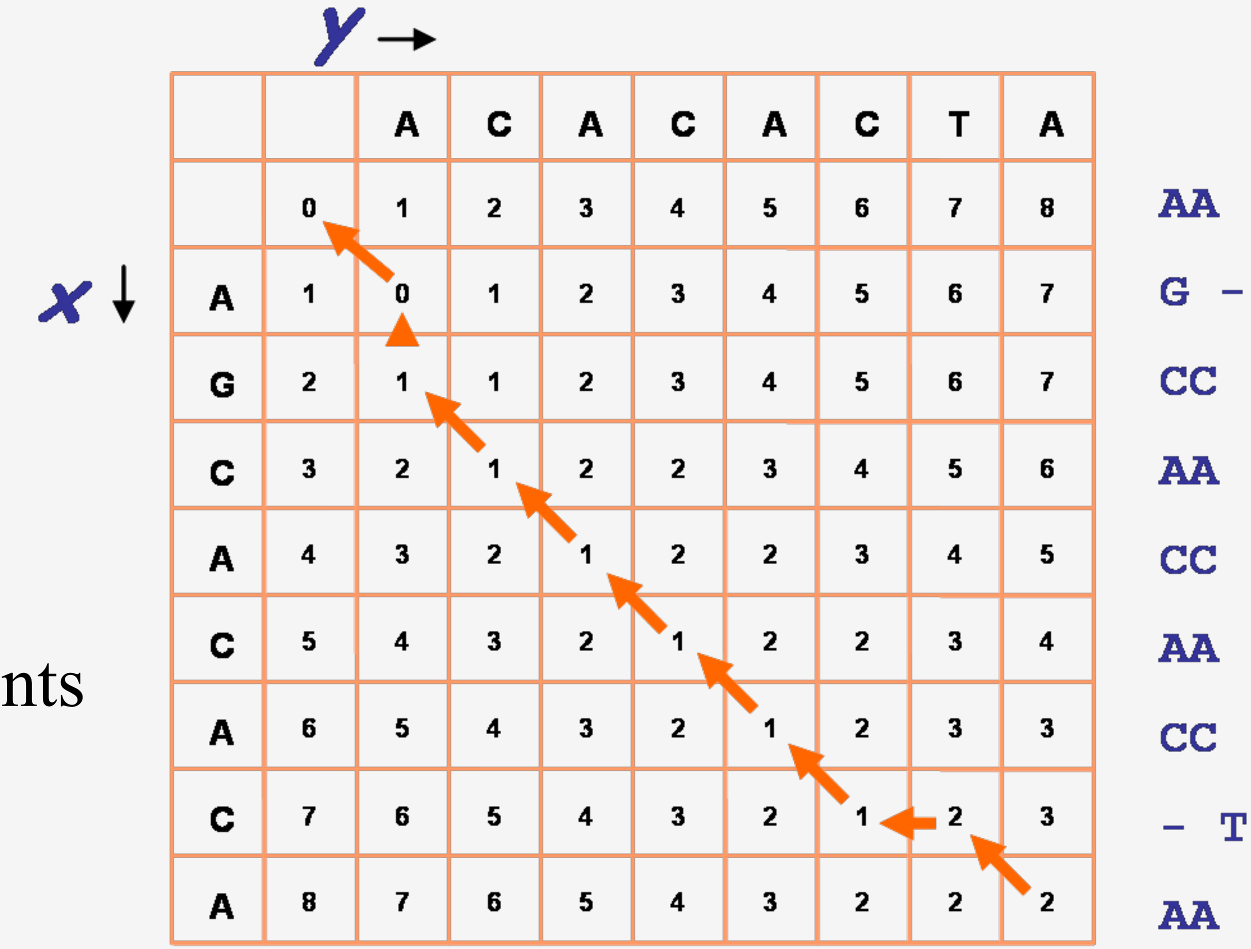
Alignment is a path in the alignment matrix

X: ACACACTA
Y: AGCACACA

X: AGCACAC-A
Y: A-CACACTA

$$\frac{m+n!}{m!n!} \approx 2^{m+n}$$ Alignments

possible!



		A	C	A	C	A	C	T	A	
	0	1	2	3	4	5	6	7	8	AA
A	1	0	1	2	3	4	5	6	7	G -
G	2	1	1	2	3	4	5	6	7	CC
C	3	2	1	2	2	3	4	5	6	AA
A	4	3	2	1	2	2	3	4	5	CC
C	5	4	3	2	1	2	2	3	4	AA
A	6	5	4	3	2	1	2	3	3	CC
C	7	6	5	4	3	2	1	2	3	- T
A	8	7	6	5	4	3	2	2	2	AA

Dynamic programming for global alignment – simple case

Consider two sequences: x[1 .. M], and y[1 .. N]
for a new position to be aligned we have ONLY three choices

1. x_i aligns to y_j

   $x_1 \ldots\ldots x_{i-1} \quad x_i$

   $y_1 \ldots\ldots y_{j-1} \quad y_j$

Align x[1 .. i] with y[1 .. j]

2. x_i aligns to a gap

   $x_1 \ldots\ldots x_{i-1} \quad x_i$

   $y_1 \ldots\ldots y_j \quad -$

x[1 ..(i-1)] is already aligned with y[1 ..(j)], so align x[i] with a gap in y

3. y_j aligns to a gap

   $x_1 \ldots\ldots x_i \quad -$

   $y_1 \ldots\ldots y_{j-1} \quad y_j$

x[1 .. i] is already aligned with y[1 .. (j-1)], so align a gap in x to y[j]

1. x_i aligns to y_j

$$\begin{array}{ll} x_1 \ldots\ldots x_{i-1} & x_i \\ y_1 \ldots\ldots y_{j-1} & y_j \end{array}$$

$F\ (i,j) = F\ (i\text{-}1,\ j\text{-}1) + \mathrm{s}(\mathrm{i},\mathrm{j})$

2. x_i aligns to a gap

$$\begin{array}{ll} x_1 \ldots\ldots x_{i-1} & x_i \\ y_1 \ldots\ldots y_j & - \end{array}$$

$F\ (i,j) = F\ (i\text{-}1,\ j) – gap\_open\_penalty\ (go)$

3. y_j aligns to a gap

$$\begin{array}{ll} x_1 \ldots\ldots x_i & - \\ y_1 \ldots\ldots y_{j-1} & y_j \end{array}$$

$F\ (i,j) = F\ (i,\ j\text{-}1) – gap\_open\_penalty\ (go)$

If we could make *F* (*i, j-1*), *F* (*i-1, j*), *F* (*i-1, j-1*) optimal, then we can make the next ones optimal as well

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

Where $s\,(x_i, y_j)$ = *Score for a match*, if $x_i = y_j$;

score for a mismatch, if $x_i \neq y_j$;

The Needleman-Wunsch Algorithm

– pioneering application of DP to biological sequences

1. Initialization.

   a. $F(0, 0) = 0$

   b. $F(0, j) = - j \times go$

   c. $F(i, 0) = - i \times go$

2. Main Iteration. Filling-in partial alignments

```
For each   i = 1……M
    For each   j = 1……N
```

$$F(i, j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) & \text{[case 1]} \\ F(i-1, j) - go & \text{[case 2]} \\ F(i, j-1) - go & \text{[case 3]} \end{cases}$$

$$Ptr(i,j) = \begin{cases} \text{DIAG,} & \text{if [case 1]} \\ \text{LEFT,} & \text{if [case 2]} \\ \text{UP,} & \text{if [case 3]} \end{cases}$$

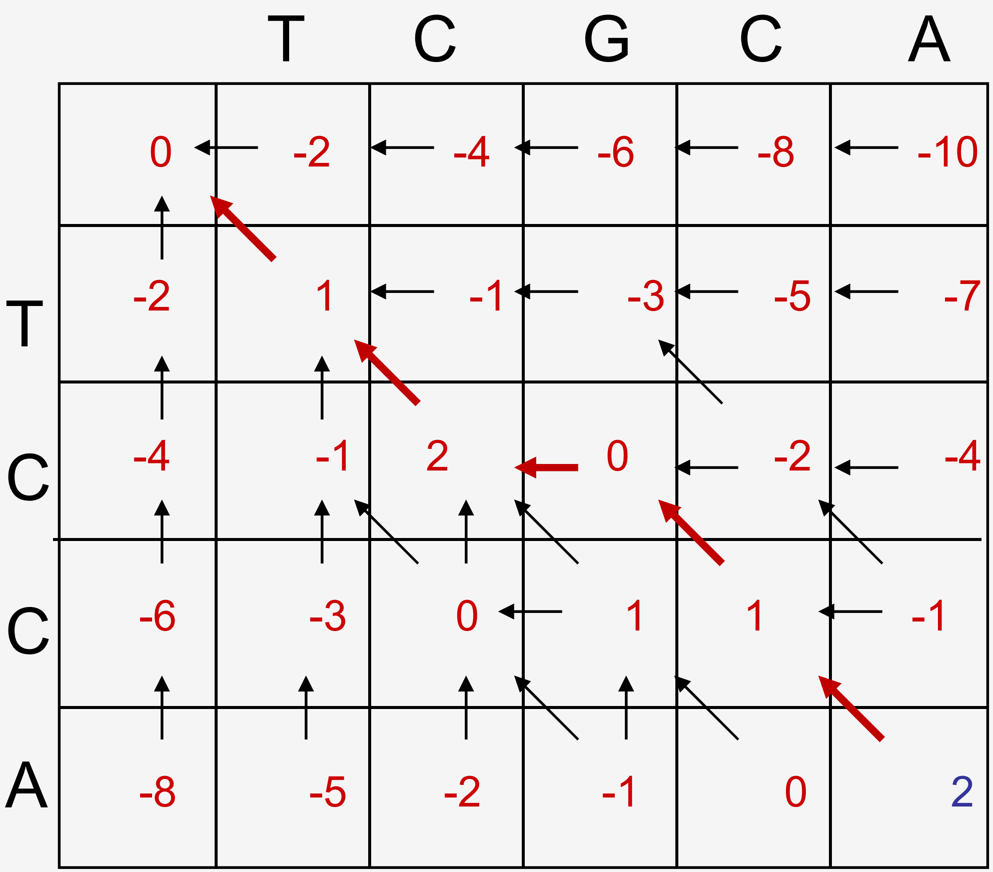
3. Termination. F(M, N) is the optimal score, and from Ptr(M, N) can trace back optimal alignment

An example

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

$s(x_i, y_j)$=1 for match,
-1 for mismatch

go=2



		T	C	G	C	A
	0	-2	-4	-6	-8	-10
T	-2	1	-1	-3	-5	-7
C	-4	-1	2	0	-2	-4
C	-6	-3	0	1	1	-1
A	-8	-5	-2	-1	0	2

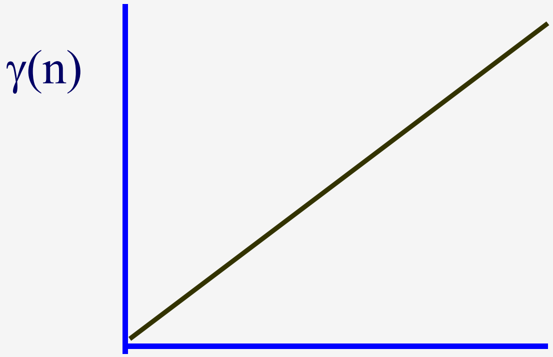
Trace back arrows from lower right to upper left and
Maximize the scores

Alignment score = 2

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

Are gaps that bad?

- Current model:
  - Gap of length n
  - incurs penalty n.go

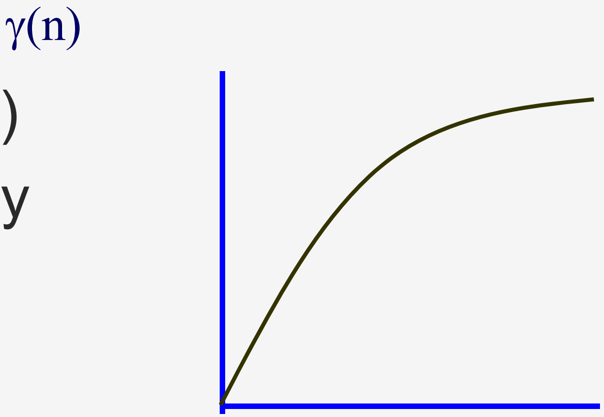


- Bunch of gaps are better than individual gaps
  - Convex (saturating) gap penalty function:

    γ(n):

    for all n, $\gamma(n+1) - \gamma(n) \leq \gamma(n) - \gamma(n-1)$

  A common function is Affine gap penalty



$$\gamma(n) = \underbrace{go}_{\text{gap open}} + (n-1)\times\underbrace{ge}_{\text{gap extend}}$$

Convex gap dynamic programming

Initialization: same as before

Iteration:

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ \max_{k=0\ldots i-1} F(k,j) - \gamma(i-k) \\ \max_{k=0\ldots j-1} F(i,k) - \gamma(j-k) \end{cases}$$

Termination: same

Running Time: $O(N^2M)$ (assume N>M)

Space: $O(NM)$

Fast implementation of affine gap penalty

We need three matrices for tracking scores

Matrix-1: a[i,j]= to store maximum score of an alignment that ends in x[i] matched to y[j]

$\ldots x_i$
$\ldots y_j$

Matrix-2: b[i,j]= to store maximum score of an alignment that ends in gap matched to y[j]

…-
$\ldots y_j$

Matrix-3: c[i,j]= to store maximum score of an alignment that ends in gap matched to x[i]

$\ldots. x_i$
… -

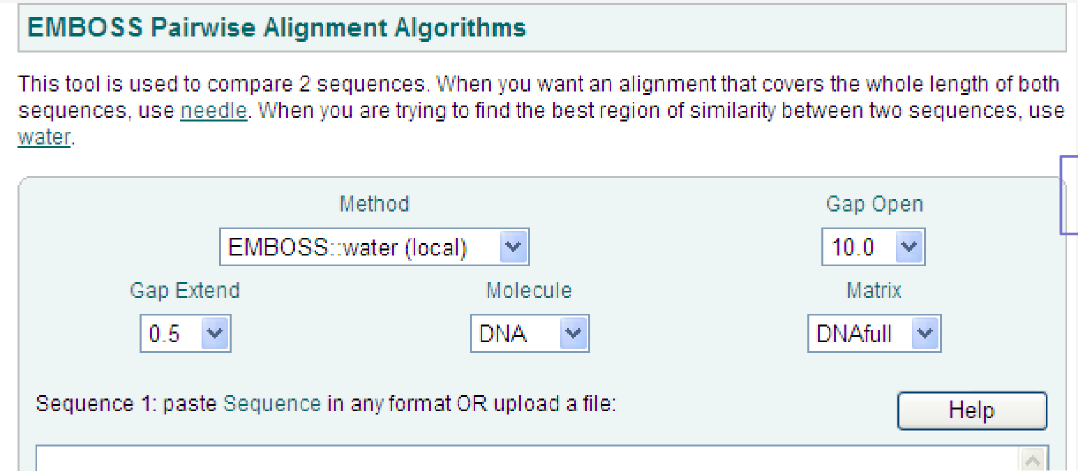
Implementation – Cont’d

$$a[i,j]=\max\begin{cases}a[i-1,j-1]\\ b[i-1,j-1]+s(i,j)\\ c[i-1,j-1]\end{cases}\qquad b[i,j]=\max\begin{cases}a[i,j-1]+go\\ b[i,j-1]+ge\\ c[i,j-1]+go\end{cases}$$

$$c[i,j]=\max\begin{cases}a[i-1,j]+go\\ b[i-1,j]+go\\ c[i-1,j]+ge\end{cases}$$

Pointer-matrices: Three matrices to figure out which state within each score matrix maximization was used to obtain the optimal alignment of position i,j. Of course you need to know which matrix yielded the best score in the end (m,n) as well – Covered by assignment!

http://www.ebi.ac.uk/Tools/emboss/align/index.html



EMBOSS Pairwise Alignment Algorithms

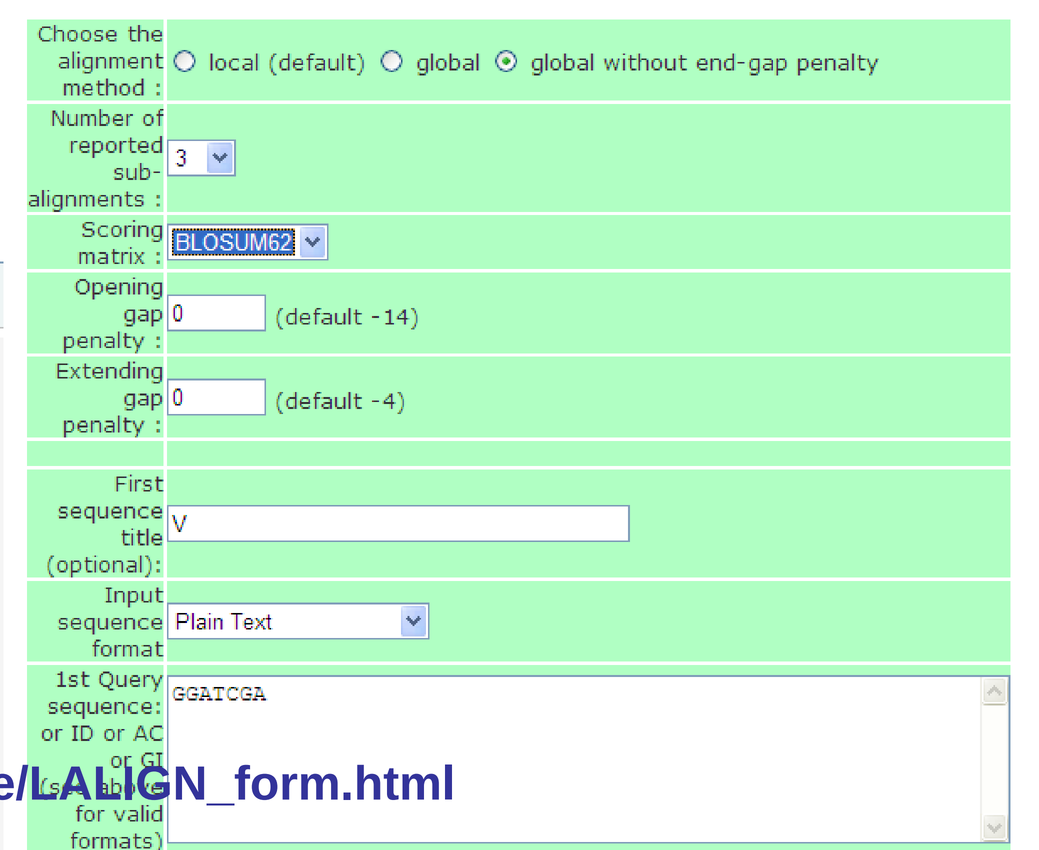
This tool is used to compare 2 sequences. When you want an alignment that covers the whole length of both sequences, use needle. When you are trying to find the best region of similarity between two sequences, use water.

Method	EMBOSS::water (local)
Gap Open	10.0
Gap Extend	0.5
Molecule	DNA
Matrix	DNAfull

Sequence 1: paste Sequence in any format OR upload a file:

Seq. 1 Upload a file:

For checking results of your code



Choose the alignment method :	○ local (default) ○ global ● global without end-gap penalty
Number of reported sub-alignments :	3
Scoring matrix :	BLOSUM62
Opening gap penalty :	0 (default -14)
Extending gap penalty :	0 (default -4)
First sequence title (optional):	V
Input sequence format	Plain Text
1st Query sequence: or ID or AC or GI (see above for valid formats)	GGATCGA

http://www.ch.embnet.org/software/LALIGN_form.html

Pair HMMs for alignment

HMM for sequence alignment, which incorporates affine gap scores.

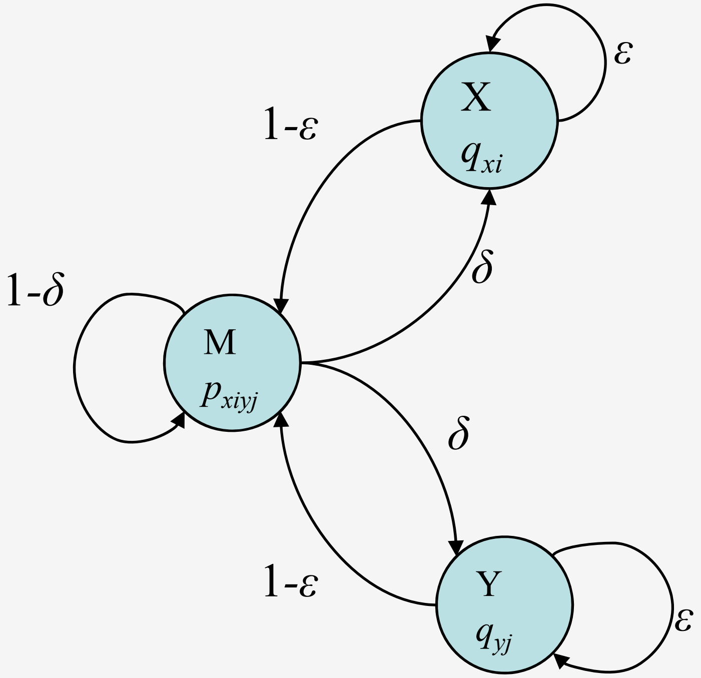
“Hidden” States

- Match (M)
- Insertion in *x* (X)
- insertion in *y* (Y)

Observation Symbols

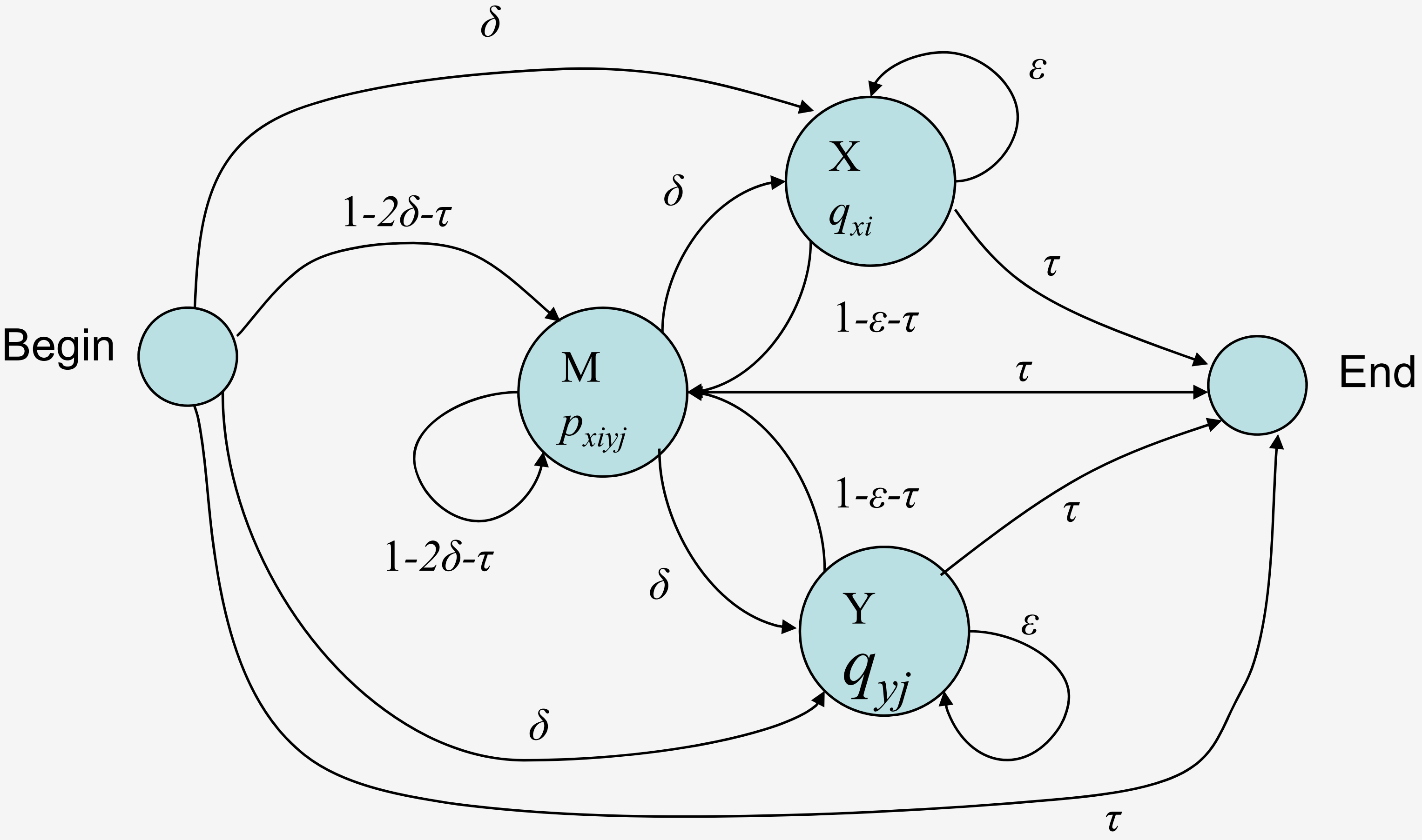
- Match (M): {(*a*,*b*)| *a*,*b* in ∑ }.
- Insertion in *x* (X): {(*a*,-)| *a* in ∑ }.
- Insertion in *y* (Y): {(-,*a*)| *a* in ∑ }.

Simple representation



	M	X	Y
M	1-2δ	δ	δ
X	1- ε	ε	0
Y	1- ε	0	ε

Full representation



Algorithm: Viterbi algorithm for pair HMMs

- Initialization:
  - $v^M(0, 0) = 1$. $v^X(0, 0) = v^Y(0, 0) = 0$ $v^*(-1, j) = v^*(i, -1) = 0$.
- Recurrence: $i = 0,\ldots,n$, $j = 0,\ldots,m$, except for(0,0);

$$v^{\mathrm{M}}(i,j) = p_{x_i y_j} \max \begin{cases} (1-2\delta-\tau)v^{\mathrm{M}}(i-1,j-1) \\ (1-\varepsilon-\tau)v^{\mathrm{X}}(i-1,j-1) \\ (1-\varepsilon-\tau)v^{\mathrm{Y}}(i-1,j-1) \end{cases}$$

$$v^{\mathrm{X}}(i,j) = q_{x_i} \max \begin{cases} \delta v^{\mathrm{M}}(i-1,j) \\ \varepsilon v^{\mathrm{X}}(i-1,j) \end{cases}$$

$$v^{\mathrm{Y}}(i,j) = q_{y_j} \max \begin{cases} \delta v^{\mathrm{M}}(i,j-1) \\ \varepsilon v^{\mathrm{X}}(i,j-1) \end{cases}$$

- Termination: $v^{\mathrm{E}} = \tau \max(v^{\mathrm{M}}(n,m), v^{\mathrm{X}}(n,m), v^{\mathrm{Y}}(n,m))$

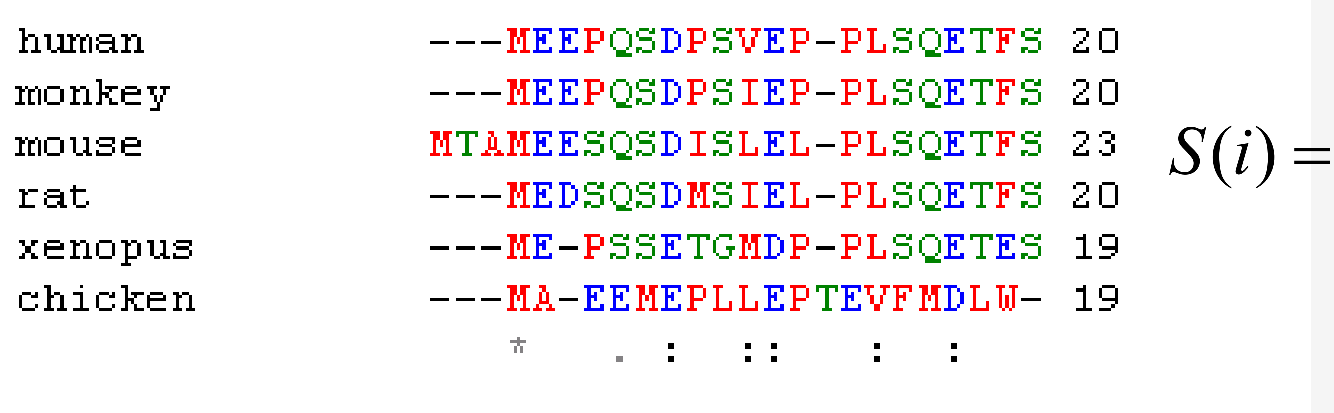
Viterbi Pair HMM is equivalent to global dynamic programming with affine gap!

- Proof (sketch)
  - Add the following missing details to DEKM’s summary
  - Take the viterbi matrices and divide by the terms for random sequence probability, and take the log of resulting transformation i.e,

$$v^{\mathrm{M}}(i,j) = \max \begin{cases} (1-2\delta-\tau)v^{\mathrm{M}}(i-1,j-1).p_{xiyj} \big/ q_{xi}q_{yi}(1-\eta)^2 \\ (1-\varepsilon-\tau)v^{\mathrm{X}}(i-1,j-1).p_{xiyj} \big/ q_{xi}q_{yi}(1-\eta)^2 \\ (1-\varepsilon-\tau)v^{\mathrm{Y}}(i-1,j-1).p_{xiyj} \big/ q_{xi}q_{yi}(1-\eta)^2 \end{cases}$$

$$v^{\mathrm{X}}(i,j) = q_{x_i} \max \begin{cases} \delta v^{\mathrm{M}}(i-1,j).q_{xi} \big/ q_{xi}(1-\eta) \\ \varepsilon v^{\mathrm{X}}(i-1,j).q_{xi} \big/ q_{xi}(1-\eta) \end{cases}$$

Multiple Sequence Alignment (MSA)



```
human           ---MEEPQSDPSVEP-PLSQETFS 20
monkey          ---MEEPQSDPSIEP-PLSQETFS 20
mouse           MTAMEESQSDISLEL-PLSQETFS 23
rat             ---MEDSQSDMSIEL-PLSQETFS 20
xenopus         ---ME-PSSETGMDP-PLSQETES 19
chicken         ---MA-EEMEPLLEPTEVFMDLW- 19
                   *   . :  ::   :  :
```

$$S(i) = \sum_{\text{all possible pairs}} s(x_i, y_i)$$

How to score? and How to align?

Scoring, common approach: Sum of pairs or its variants for each column

Consider L aligned at all N positions, LL score =5

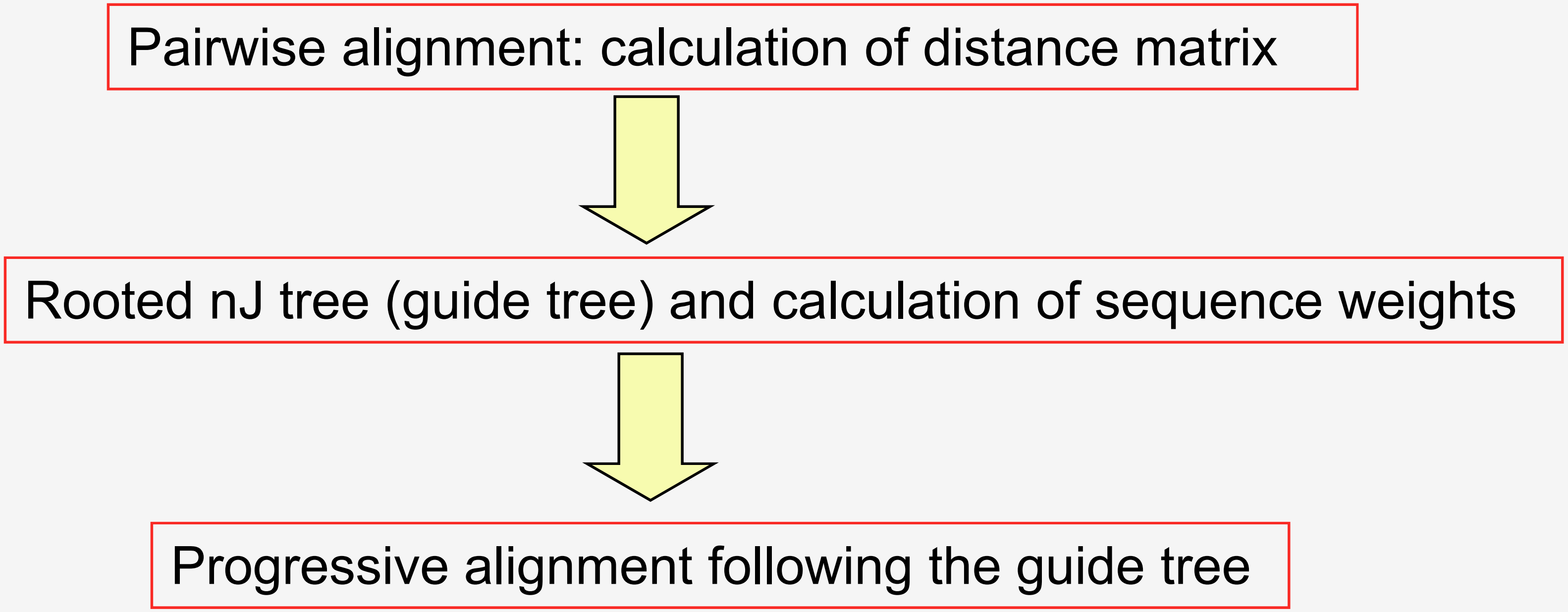
total score, $S(i) = 5 \times N(N-1)/2$

Now consider one position being G, rest being L GL score=-4

$$\frac{\Delta S(i)}{S(i)} = \frac{-5(N-1) + -4(N-1)}{S(i)} = \frac{-9(N-1)}{2.5N(N-1)} = \frac{-9}{2.5N}$$

Relative evidence decreases with N, opposed to having increased confidence in L being the "correct" residue at that position

Clustal W



Pairwise alignment: calculation of distance matrix

Rooted nJ tree (guide tree) and calculation of sequence weights

Progressive alignment following the guide tree

Step 1-Calculation of Distance Matrix using pairwise alignment

Use the Distance Matrix to create a Guide Tree to determine the “order” of the sequences.

Hbb-Hu	1	-						
Hbb-Ho	2	.17	-					
Hba-Hu	3	.59	.60	-				
Hba-Ho	4	.59	.59	.13	-			
Myg-Ph	5	.77	.77	.75	.75	-		
Gib-Pe	6	.81	.82	.73	.74	.80	-	
Lgb-Lu	7	.87	.86	.86	.88	.93	.90	-
		1	2	3	4	5	6	7

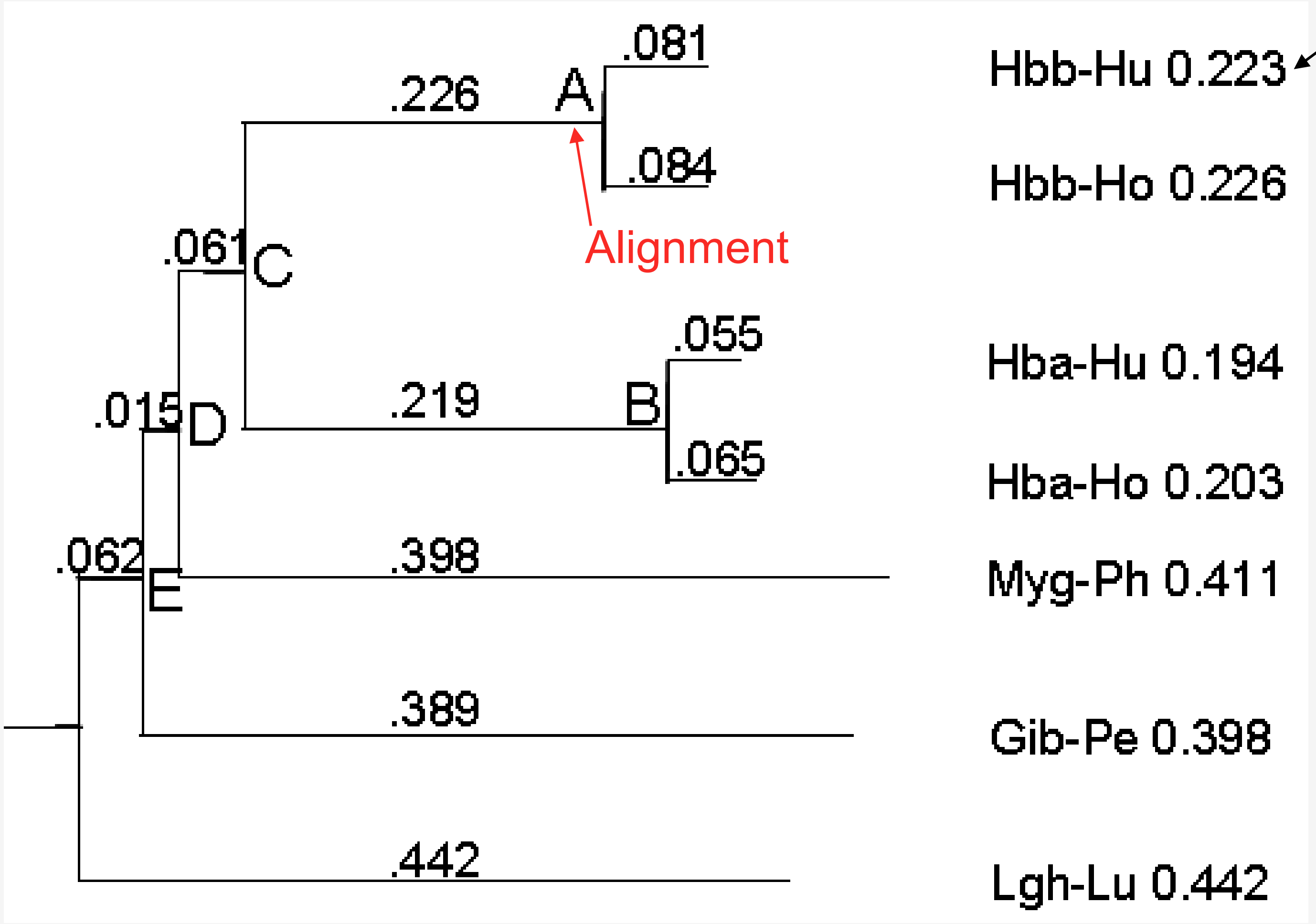
D = 1 – (I)

D = Difference score

$$I = \frac{\text{\# of identical aa's in pairwise global alignment}}{\text{total number of aa's in shortest sequence}}$$

Step 2-Create Rooted Tree and calculate weights

Neighbor joining algorithm – simple, to be discussed later



Weight

Order of alignment:

1 Hba-Hu vs Hba-Ho

2 Hbb-Hu vs Hbb-Ho

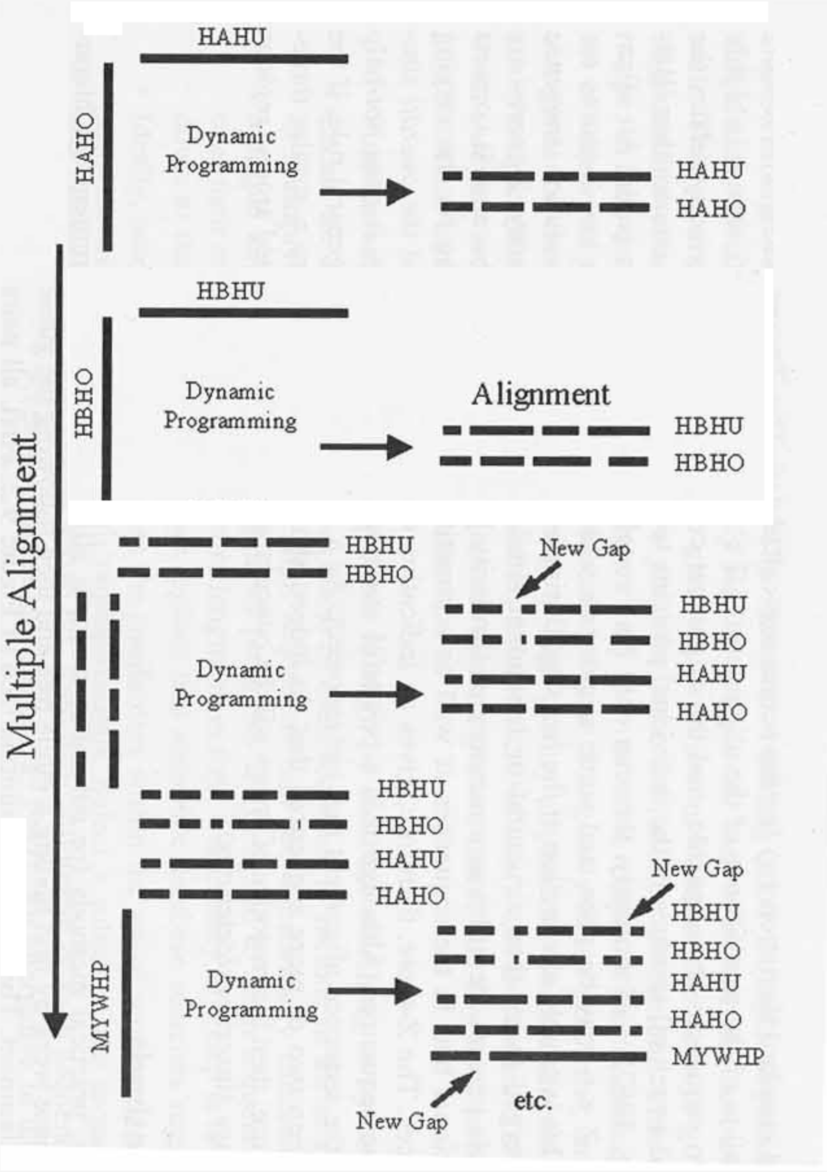
3 A vs B

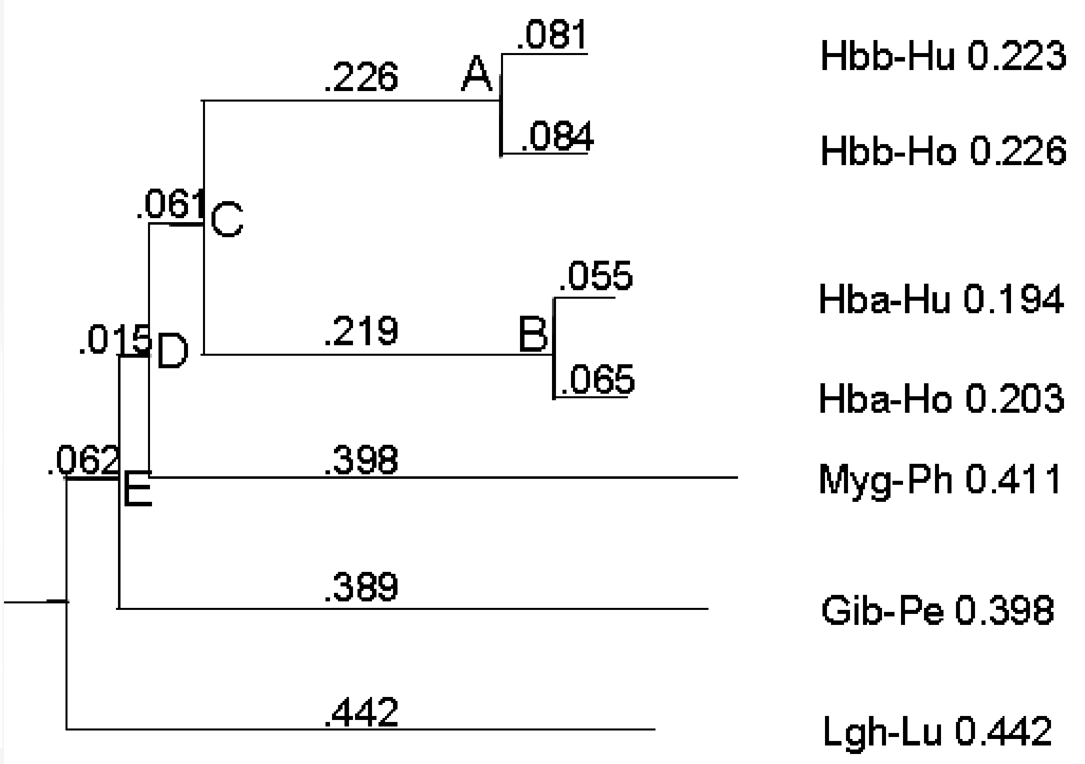
4 Myg-Ph vs C

5 Gib-Pe vs D

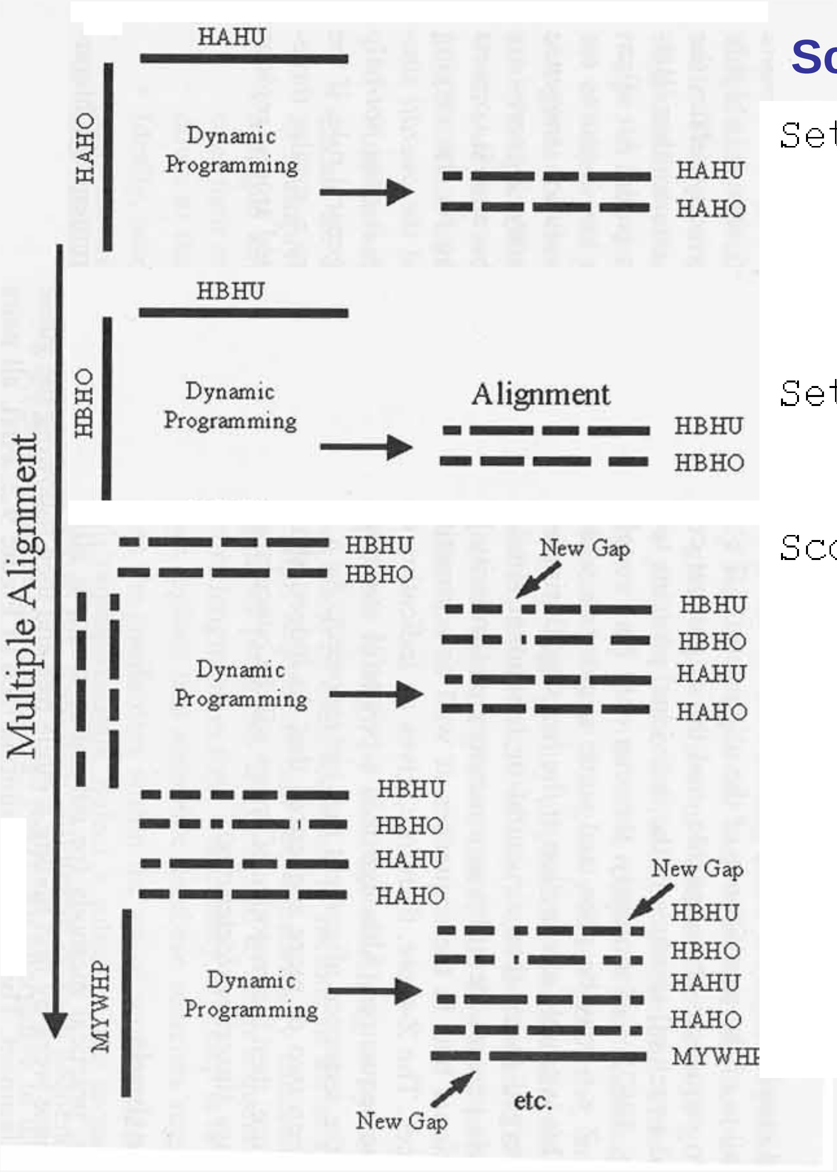
6 Lgh-Lu vs E

Step 3-Progressive alignment





Step 3-Progressive alignment



Scoring during progressive alignment

```
Set of 4:      1 eeksavtal
               2 eekaavlal
               3 adktnvkaa
               4 adktnvkaa

Set of 2:      5 gewqlvlhv
               6 aektkirsa
```

$$\text{Score} = \left.\begin{array}{rl} & M(t,v)*W_1*W_5 \\ + & M(t,i)*W_1*W_6 \\ + & M(l,v)*W_2*W_5 \\ + & M(l,i)*W_2*W_6 \\ + & M(k,v)*W_3*W_5 \\ + & M(k,i)*W_3*W_6 \\ + & M(k,v)*W_4*W_5 \\ + & M(k,i)*W_4*W_6 \end{array}\right\} \text{divided by 8}$$

Recommended MSA Programs

- MUSCLE (fast and accurate)
- MAVID (genome-scale alignment)
- SAM (hidden markov, powerful and wide range of options)