



Master Course

Algorithms in Sequence Analysis

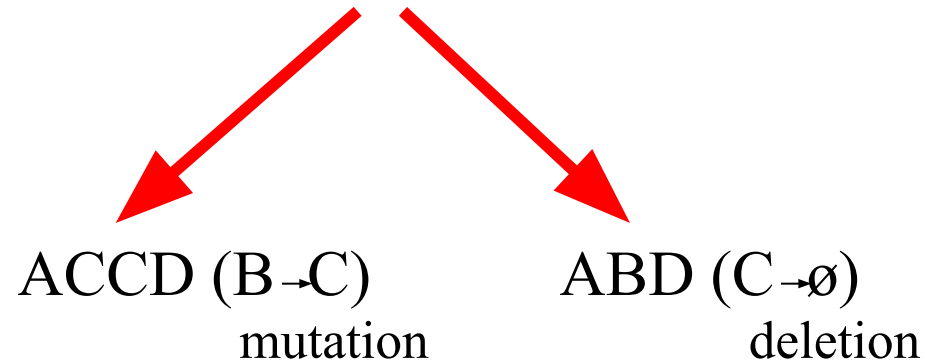
Lecture 3: Substitution Matrices



Centre for Integrative Bioinformatics VU
vrije Universiteit amsterdam

Divergent evolution – searching the true alignment

Ancestral sequence: ABCD



ACCD
AB—D

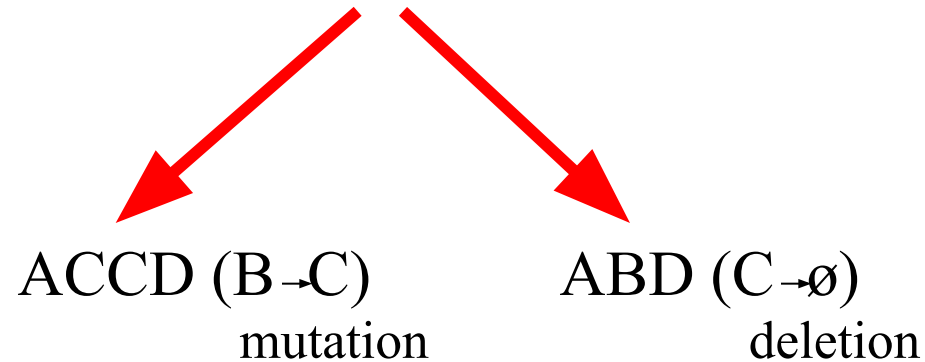
or

ACCD
A—BD

Pairwise Alignment

Divergent evolution – searching the true alignment

Ancestral sequence: ABCD



ACCD
AB—D

true alignment

or

ACCD
A—BD

Pairwise Alignment

How many pair-wise alignments

T	D	W	V	T	A	L	K
T	D	W	L	-	-	I	K

Combinatorial explosion

- 1 gap in 1 sequence: $n+1$ possibilities
- 2 gaps in 1 sequence: $(n+1)n$
- 3 gaps in 1 sequence: $(n+1)n(n-1)$, etc.

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

Given the exceeding number of possible alignments for biological sequences, scoring all of them and picking the highest scoring one is out of the question.

2 sequences of 300 a.a.: $\sim 10^{88}$ alignments

2 sequences of 1000 a.a.: $\sim 10^{600}$ alignments!

Technique to overcome the combinatorial explosion:

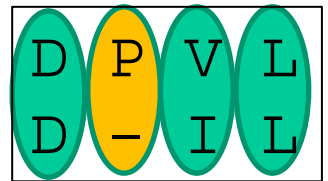
Dynamic Programming

- Break alignment problem up in smaller subproblems and solve these iteratively
- Alignment is simulated as a Markov process, all sequence positions are seen as independent (independent and identically distributed - i.i.d)
- Chances of sequence events are taken as independent
 - Therefore, probabilities per aligned position can be multiplied
 - Amino acid matrices contain so-called log-odds values (\log_{10} of the probabilities), so probabilities can be summed [$\log(ab)=\log(a)+\log(b)$]

Needleman, S.B. & Wunsch, C. D. (1970), "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins", *J.Mol. Biol.*, vol. 48, pp. 443-453.

Treating sequence alignment statistically...

- To perform statistical analyses on sequence alignment, we need a reference model allowing us to compute the statistical significance of an alignment.
- The model: each letter in a sequence is selected from a defined alphabet in an ***independent and identically distributed (i.i.d.)*** manner.
 - All residues, given their base frequency, have an equal chance to position anywhere in the sequence, independent from other positions
- Given a probability distribution, P_i , for matched residues in an alignment, the probability of seeing a particular sequence of $i, j, k, \dots n$ is simply $P_i P_j P_k \dots P_n$



As an alternative to multiplication of the probabilities, we could sum their logarithms and exponentiate the result. The probability of the same sequence of letters can be computed by exponentiating $\log P_i + \log P_j + \log P_k + \dots + \log P_n$.

In practice, when aligning sequences we only add log-odds values (from residue exchange matrices) but we do not exponentiate the final score.

$$\log ab = \log a + \log b$$

Dynamic Programming (DP)

- A global optimisation algorithm for sequential segmentation (e.g., fragments or letters)
- Given a scoring system to enable segmentation, the DP algorithm guarantees an optimal solution.
- The method was developed by applied mathematician **Richard E. Bellman** in 1953 and has found applications in diverse fields, such as economics, control engineering, or bioinformatics.
- DP requires that sub-problems can be nested recursively inside larger problems, where the relation between the value of the larger problem and the values of the sub-problems are described by the **Bellman equation**.

Needleman, S.B. & Wunsch, C. D. (1970), "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins", *J.Mol. Biol.*, vol. 48, pp. 443-453.

Scoring alignments

Gap penalties

Gap penalty types:

- **Linear:** $gp(k)=ak$
- **Affine:** $gp(k)=b+ak$
- **Concave;** e.g., $gp(k)=\log(k)$

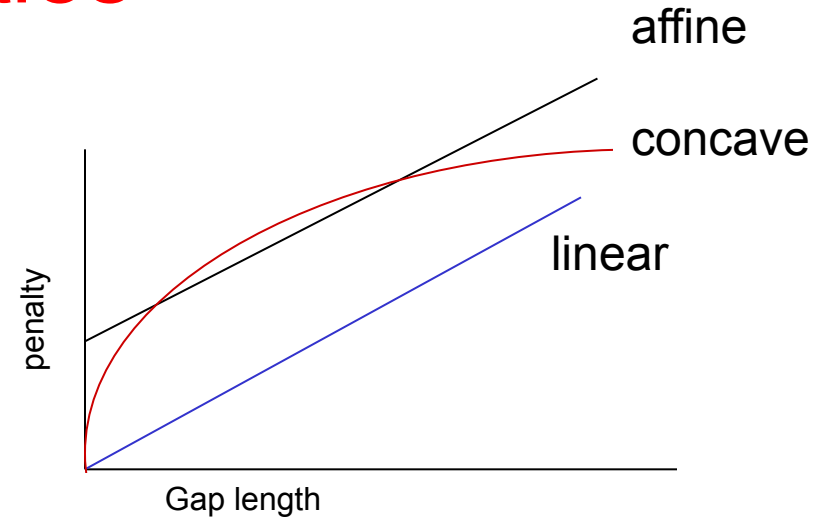
```
GTA--  
--ATG
```

```
G-T-A  
-ATG-
```

Alignment scoring (DNA or protein) should give reasonable alignments

We have to assign scores to:

- Substitution (or match/mismatch)
- insertions /deletions



General alignment score:

$$S_{a,b} = \sum_l s(a_i, b_j) - \sum_k N_k \cdot gp(k)$$

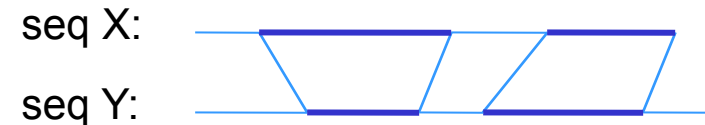
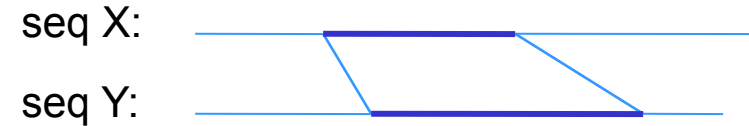
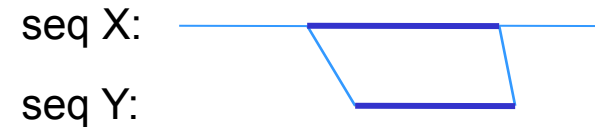
Substitution
(match/mismatch) score

Gap penalty

The score of an alignment is the sum of the scores over all alignment columns (incl. gapped positions)

Summary – types of alignment

1. **Global**
e.g Needleman-Wunsch
algorithm
2. **Semi-global**
3. **Local**
e.g. Smith-Waterman
algorithm
4. **Multiple local
alignments**
aka Waterman-Eggert
algorithm



There are three kinds of alignments

- Global alignment (preceding slides)
- **Semi-global alignment**
- Local alignment

Variation on global alignment

- *Global* alignment: previous algorithm is called *global* alignment because it uses all letters from both sequences.

CAGCACTTGGATTCTCGG

CAGC-----G-T-----GG

- *Semi-global* alignment: uses all letters but does not penalize for end gaps

CAGCA-CTTGGATTCTCGG

---CAGCGTGG-----

Semi-global alignment

- Global alignment: all gaps are penalised
- Semi-global alignment: N- and C-terminal (5' and 3') gaps (end-gaps) are not penalised

MSTGAVLIY--TS-----
--GGILLFHRTSGTSNS

End-gaps

End-gaps

Semi-global alignment

Applications of *semi-global*:

- Finding a gene in genome
- Placing marker onto a chromosome
- One sequence much longer than the other



Risk: if gap penalties high -- really bad alignments for divergent sequences



Protein sequences have N- and C-terminal amino acids that are often small and hydrophilic. When sequences are divergent, this may lead to 'head-to-tail' alignments (see alignment on the left)

Semi-global alignment

$$M[i,j] = \max \begin{cases} M[i-1,j-1] + 1 \\ M[i,j-1] - 2 \\ M[i-1,j] - 2 \end{cases}$$

- Ignore 5' or N-terminal end gaps
 - First row/column set to 0
- Ignore C-terminal or 3' end gaps
 - Read the result from **last row/column** (select the highest scoring cell)

	-	G	A	G	T	G
-	0	0	0	0	0	0
A	0	-1	1	-1	-1	-1
G	0	1	-2	2	0	0
T	0	-1	0	0	3	1

If the highest scoring cell is not the lowest-rightmost one, meaning there are one or more end-gaps, then pad the remaining sequence stretch (x or y sequence) with (zero) end-gaps

Measuring Alignment Similarity

- **Sequence identity** (number of identical exchanges per unit length)
- **Raw alignment score** (the score obtained by DP)
- **Sequence similarity** (alignment score normalised to a maximum possible)
- **Alignment score normalised** to a randomly expected situation (database/homology searching – later lectures)

Statistical alignment score

Similarity by chance or common ancestry?

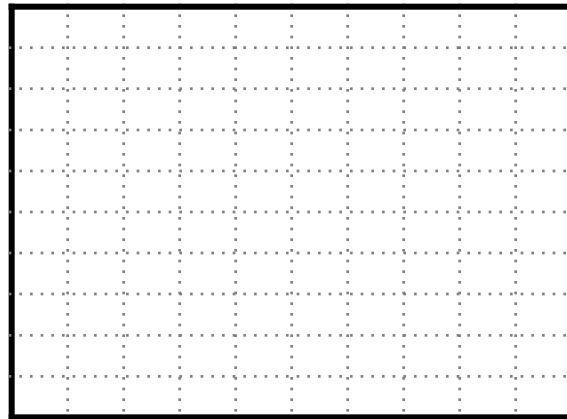
Z-Score:

- Alignment score normalised to a randomly expected situation
 - Shuffle (scramble) one or both sequences
 - Calculate alignment score
 - Repeat N times and calculate mean and stdev over the N random (scrambled) alignment scores
 - Convert real alignment score x to Z-score

$$\text{Z-score} = (x - \text{mean}) / \text{stdev}$$

Substitution matrices

Algorithms in Sequence
Analysis



Substitution scores $S(i,j)$

Given an alignment, what is its score?

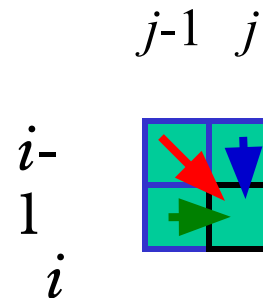
```
MSTGAVLIY--TSPS---  
---GGILLFHRTSGTSNS
```

$S(L,L)$

$S(P,G)$

$S(I,L)$

Substitution score in sequence alignment:

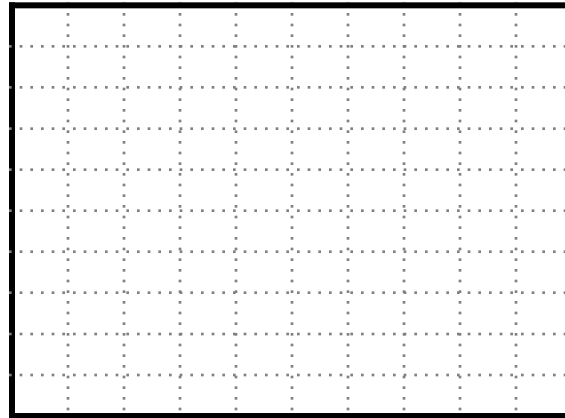


$$M[i,j] = \max \begin{cases} M[i-1,j-1] + \text{score}(X[i], Y[j]) \\ M[i,j-1] - g \\ M[i-1,j] - g \end{cases}$$

How
do we
get
these?

Definition

- Two-dimensional matrix with score values describing the probability of one amino acid or nucleotide being replaced by another during sequence evolution.



What is simplest substitution table you can think of for:

- DNA sequences
- Protein sequences

Scoring matrices for nucleotide sequences

- Can be simple:
 - e.g. positive value for match and zero (or -1) for mismatch.
 - frequencies of mutation are equal for all bases (Jukes-Cantor model).
- Can be more complicated:
 - taking into account transitions ($>$) and transversions ($<$) (Kimura model)

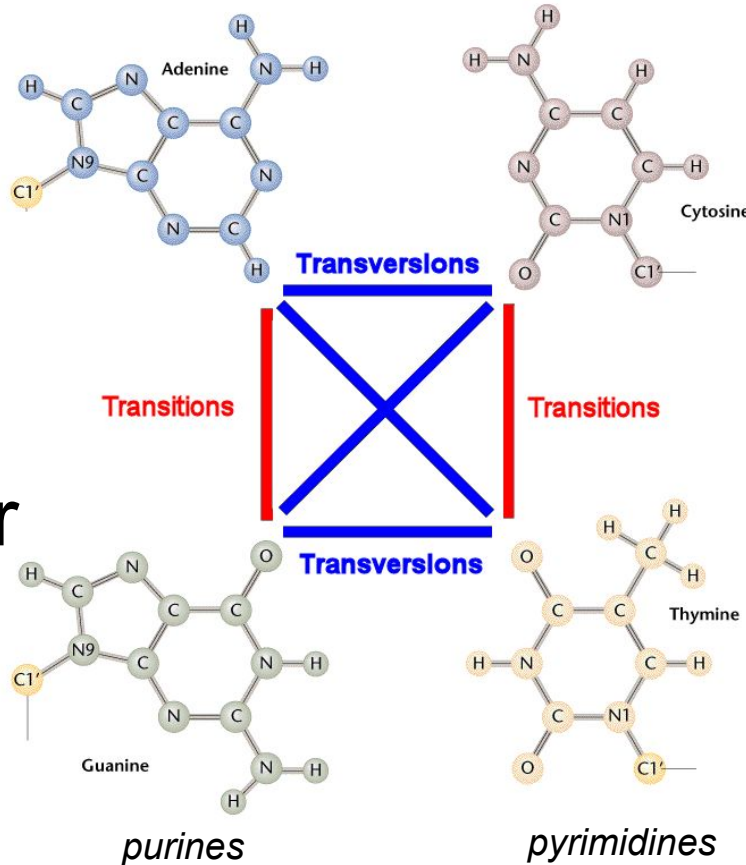
Scoring matrices for nucleotide sequences

- Simple mode

	A	C	T	G
A	1	0	0	0
C	0	1	0	0
T	0	0	1	0
G	0	0	0	1

- Jukes-Cantor

	A	C	T	G
A	\square	α	α	α
C	α	\square	α	α
T	α	α	\square	α
G	α	α	α	\square



- Kimura

	A	C	T	G
A	\square	β	β	α
C	β	\square	α	β
T	β	α	\square	β
G	α	β	β	\square

transversions are less common than transitions

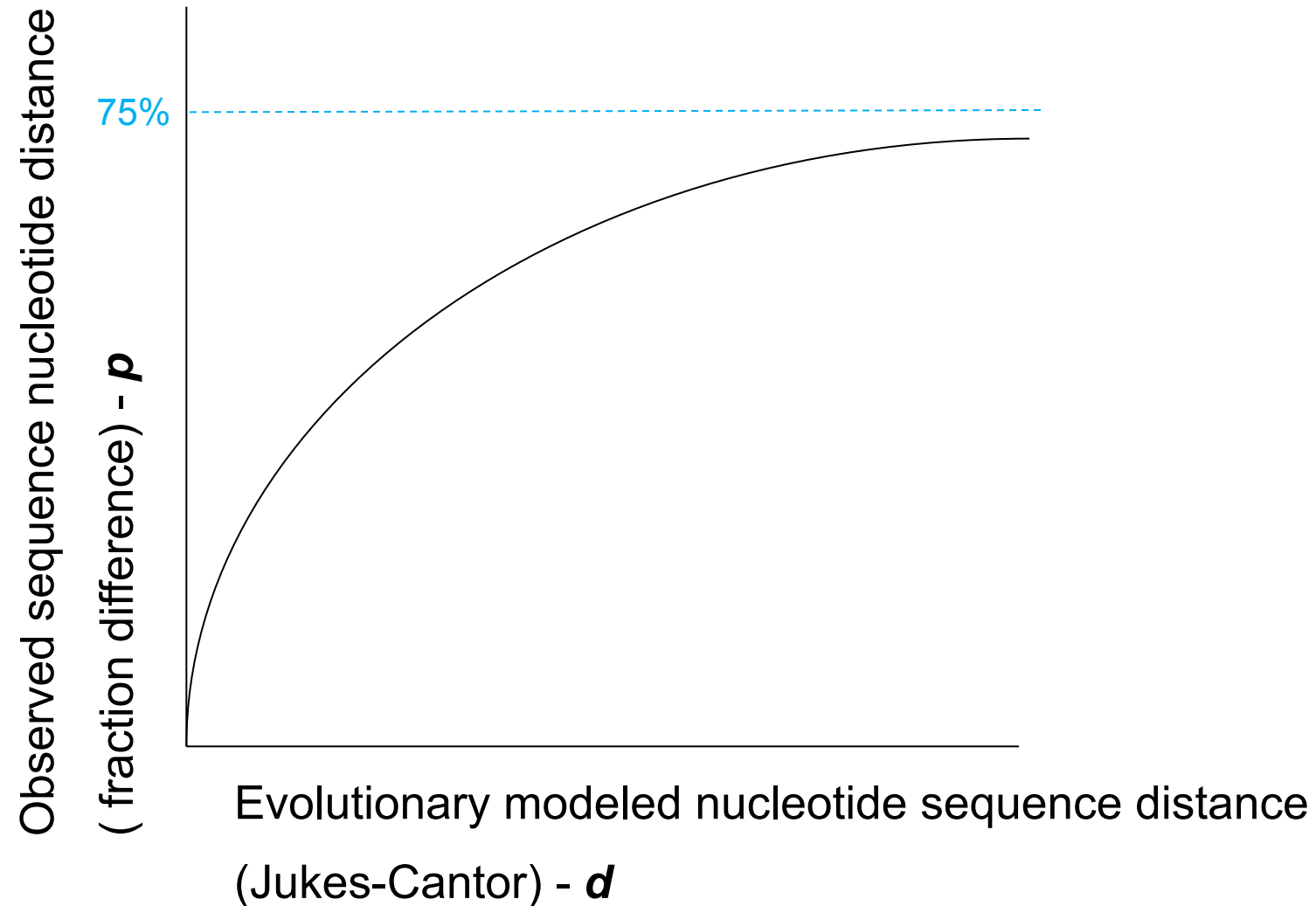
Question:

What sequence similarity do you expect for 2 random DNA sequences?

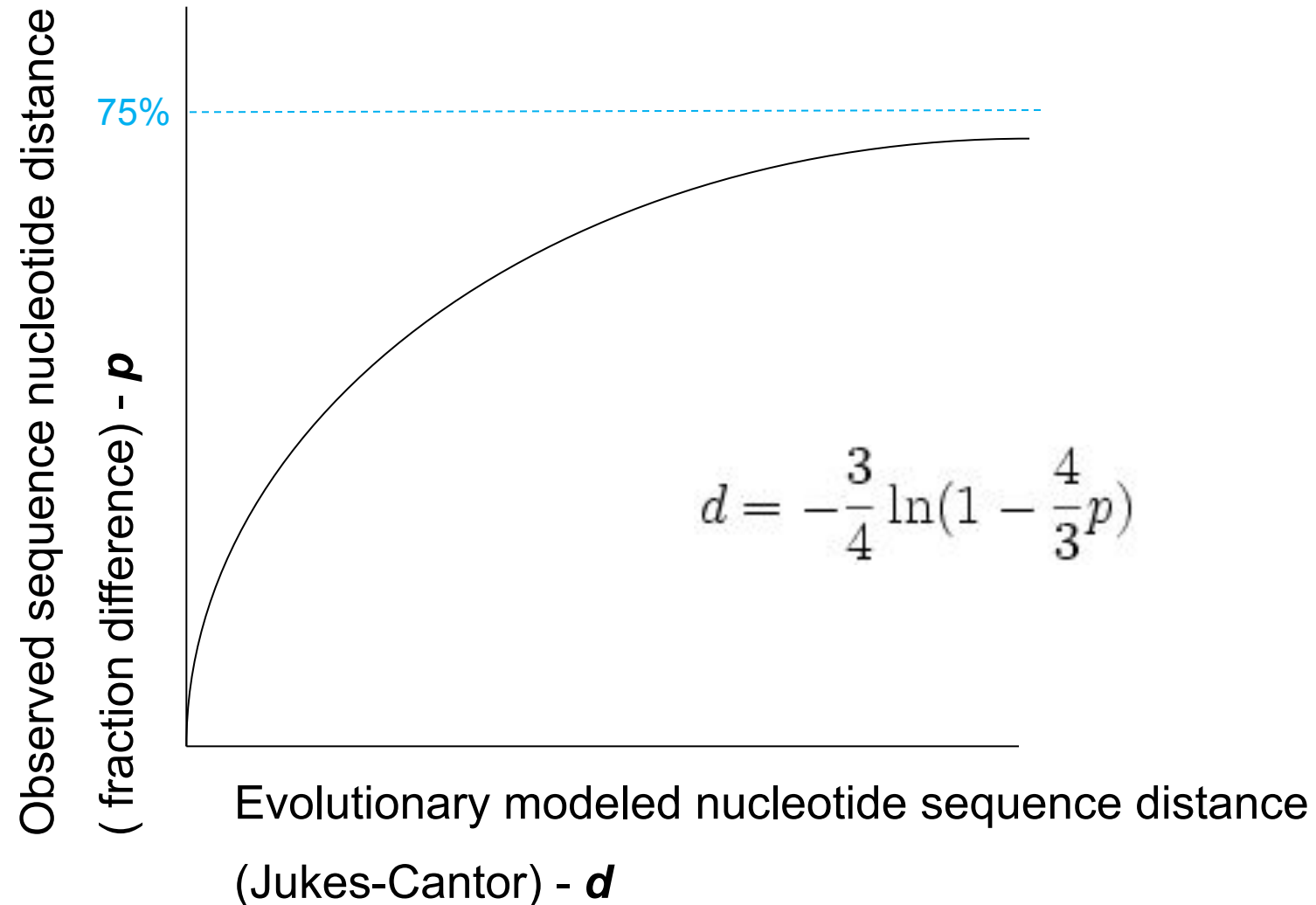
Sequence similarity measures

- There are various models to correct for the fact that the true rate of evolution cannot be observed through nucleotide (or amino acid) exchange patterns (e.g. due to back mutations)
- Saturation level for nucleotide sequences is ~75%, higher real mutations are no longer observable; e.g.
observed A->T through A->C->G->T
or observed G->G through G->C->G (back mutation)
- Saturation level for protein sequences is ~94% (see next slide)

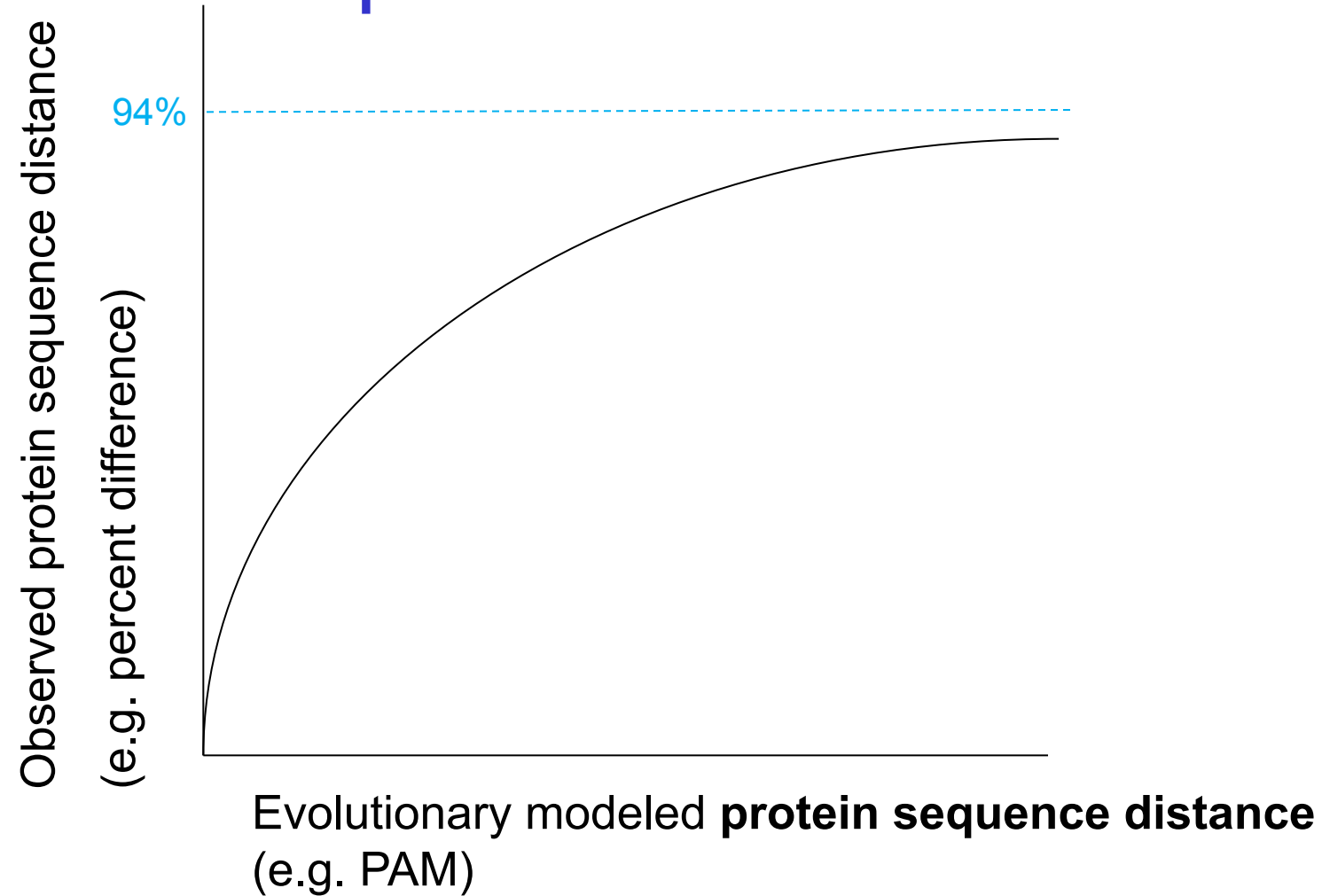
Similarity criterion for phylogeny



Similarity criterion for phylogeny



Protein evolution and observed sequence distance



Evolutionary models for DNA evolution

A				
T	α			
G	α	α		
C	α	α	α	
	A	T	G	C

JC69

A				
T	β			
G	α	β		
C	β	α	β	
	A	T	G	C

Kimura

The Jukes–Cantor (left) and Kimura (right) models for DNA substitutions.

In the Jukes–Cantor model, all nucleotides have equal substitution rates (α).

In the Kimura model, there are unequal rates of transitions (α) and transversions (β). The probability values for identical matches are shaded because evolutionary distances only count different residue positions.

Motoo Kimura

- Neutral theory of molecular evolution (1968)
- Famous monograph (1983) *The Neutral Theory of Molecular Evolution*
- Neutral theory: At molecular level most evolutionary changes and most of variation within and between species is not caused by natural selection but by genetic drift of mutant alleles that are neutral.
- A neutral mutation does not affect an organism's ability to survive and reproduce.
- Theory allows for possibility that most mutations are deleterious (Darwin), but since these are rapidly removed by natural selection, they do not make significant contributions to variation within and between species at the molecular level.
- Mutations that are not deleterious are assumed to be mostly neutral rather than beneficial.

Calculating true (corrected) rate of DNA evolution - example

- **Jukes-Cantor model** (nucleotides):

$$d_{AB} = -(3/4) \ln[1 - (4/3)p_{AB}]$$

- E.g. $p_{AB} = 0.3$ (i.e. observed sequence difference is 30%)
-> $d_{AB} = -3/4 \ln[1 - (4/3 \times 0.3)] = 0.38$

(so corrected evolutionary distance is larger than the observed one)

- Model only works for closely related sequences

Calculating true rate of DNA evolution

- **Kimura model** (nucleotides):

$$d_{AB} = -(1/2) \ln(1 - 2p_{ti} - p_{tv}) - (1/4) \ln(1 - 2p_{tv})$$

- An example: let sequences A and B differ by 30%. If 20% of changes are a result of transitions (*ti*) and 10% of changes are a result of transversions (*tv*), the evolutionary distance can be calculated using

$$d_{AB} = -1/2 \ln(1 - 2 \times 0.2 - 0.1) - 1/4 \ln(1 - 2 \times 0.1) = 0.40$$

Calculating true rate of evolution for protein sequences:

use sequence identity with ***Kimura correction***:

- Express sequence distance as (1 – fraction identity)
- Protein sequences:

$$d_{AB} = -\ln(1.0 - p_{AB} - (p_{AB})^2/5.0),$$

where d_{AB} is the corrected distance and p_{AB} is observed percentage divergence between the two aligned sequences A and B

$$\text{e.g. } p_{AB} = 0.3 \rightarrow d_{AB} = -\ln(1.0 - 0.3 - 0.3^2/5.0) = 0.38$$

$$p_{AB} = 0.5 \rightarrow d_{AB} = -\ln(1.0 - 0.5 - 0.5^2/5.0) = 0.80$$

- Heuristic formula that is only defined for $p < 0.86$ (earlier slide shows max observable limit is at $p = 0.94$) .

Other evolutionary models

The Jukes-Cantor and Kimura models assume equal base rates (stationary frequencies becoming equal in infinite evolutionary time)

There are more complex models, such as

- F81 model (Felsenstein 1981)
- HKY85 (Hasegawa, Kishino and Yano 1985)
- T92 (Tamura 1992)
- TN93 model (Tamura and Nei 1993)
- GTR: Generalised time-reversible

These models do not assume equal base rates and take many more parameters into consideration. However, these more complex models are normally not used in practice because the calculations are complicated and the variance levels resulting from their formulae are typically high.

How are these models used in sequence alignment

The main use in alignment is 'correcting' the alignment scores (Jukes-Cantor and Kimura models for DNA; (another) Kimura model for proteins)

However, the majority of (multiple) alignment methods do not use this correction, since it is based upon crude sequence identity scores that only score residue matches

An important use of these methods however is in Phylogeny, where attempts are made to reconstruct the most-likely evolutionary decendance of sequences, particularly in Maximum Likelihood methods (see later lecture)

What is better to align?

DNA or protein sequences?

1. Many mutations within DNA are synonymous
⇒ divergence overestimation

		Second letter				
		U	C	A	G	
First letter	U	UUU } Phe UUC } UUA } Leu UUG }	UCU } UCC } Ser UCA } UCG }	UAU } Tyr UAC } UAA Stop UAG Stop	UGU } Cys UGC } UGA Stop UGG Trp	U C A G
	C	CUU } CUC } Leu CUA } CUG }	CCU } CCC } Pro CCA } CCG }	CAU } His CAC } CAA } Gln CAG }	CGU } CGC } Arg CGA } CGG }	U C A G
	A	AUU } AUC } Ile AUA } AUG Met	ACU } ACC } Thr ACA } ACG }	AAU } Asn AAC } AAA } Lys AAG }	AGU } Ser AGC } AGA } Arg AGG }	U C A G
	G	GUU } GUC } Val GUA } GUG }	GCU } GCC } Ala GCA } GCG }	GAU } Asp GAC } GAA } Glu GAG }	GGU } GGC } Gly GGA } GGG }	U C A G

2. Evolutionary relationships can be more accurately expressed using a **20×20 amino acid exchange table**
3. DNA sequences contain **non-coding regions**, which should be avoided in homology searches.
4. Still an issue when translating into (six) protein sequences through a codon table.
5. Searching at protein level: **frameshifts** can occur, leading to stretches of incorrect amino acids and possibly elongation or truncation.



However, frameshifts normally result in stretches of highly unlikely amino acids.

...ACT TTT CAT AGT...
 ...Thr Phe His Ser...
 ...ACT TTT TCA TAG T...
 ...Thr Phe Ser Stop

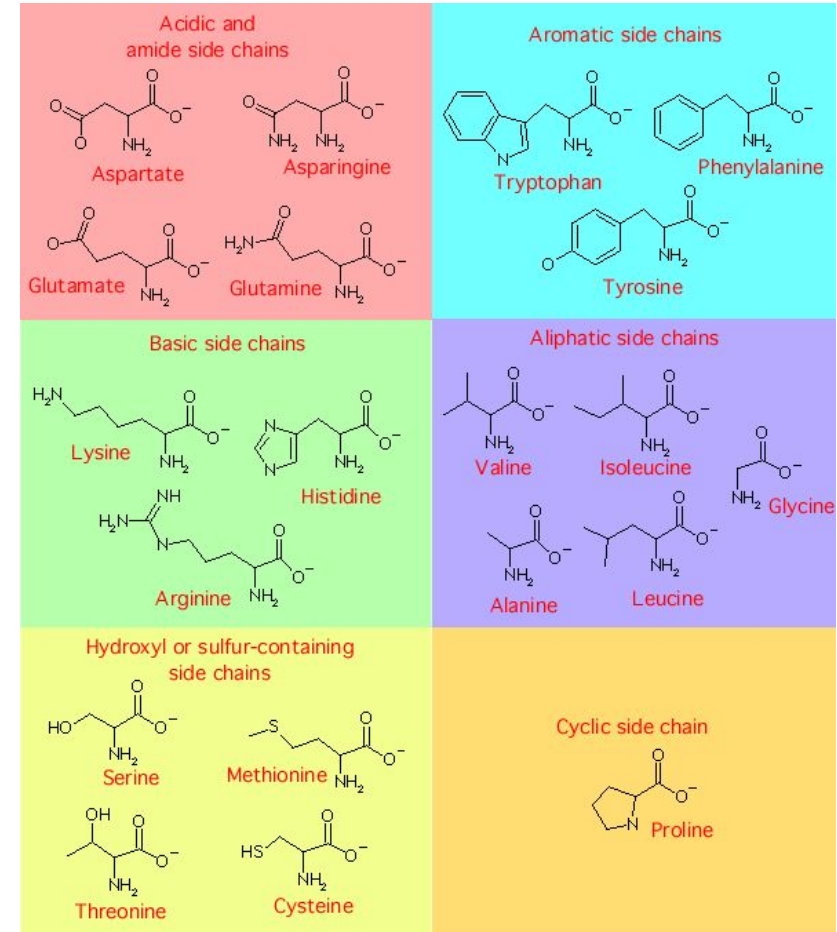
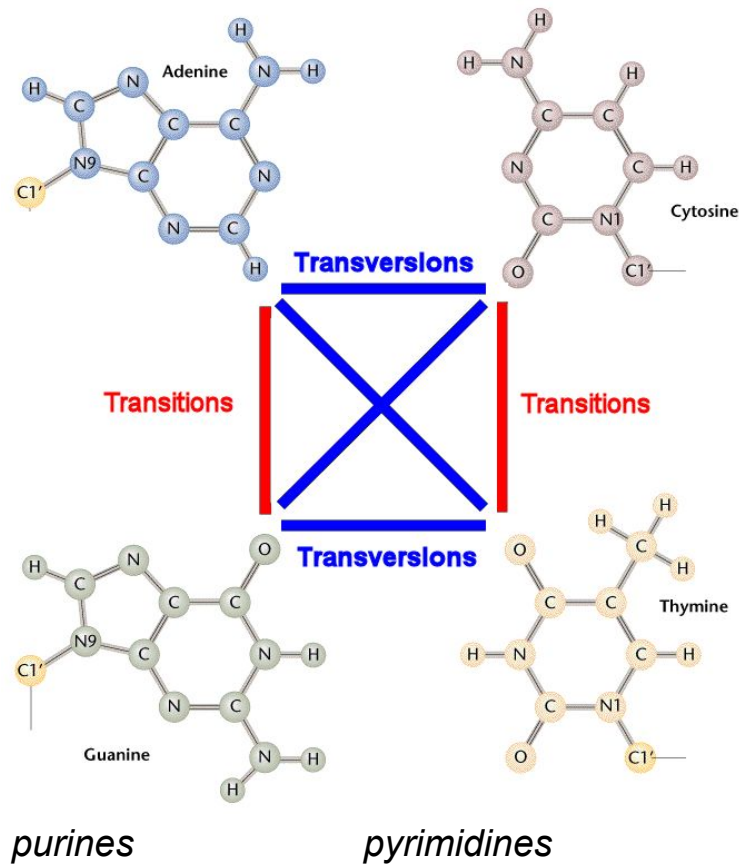
So?

Rule of thumb:

⇒ if ORF (open reading frame) exists, then align at protein level

Scoring matrices

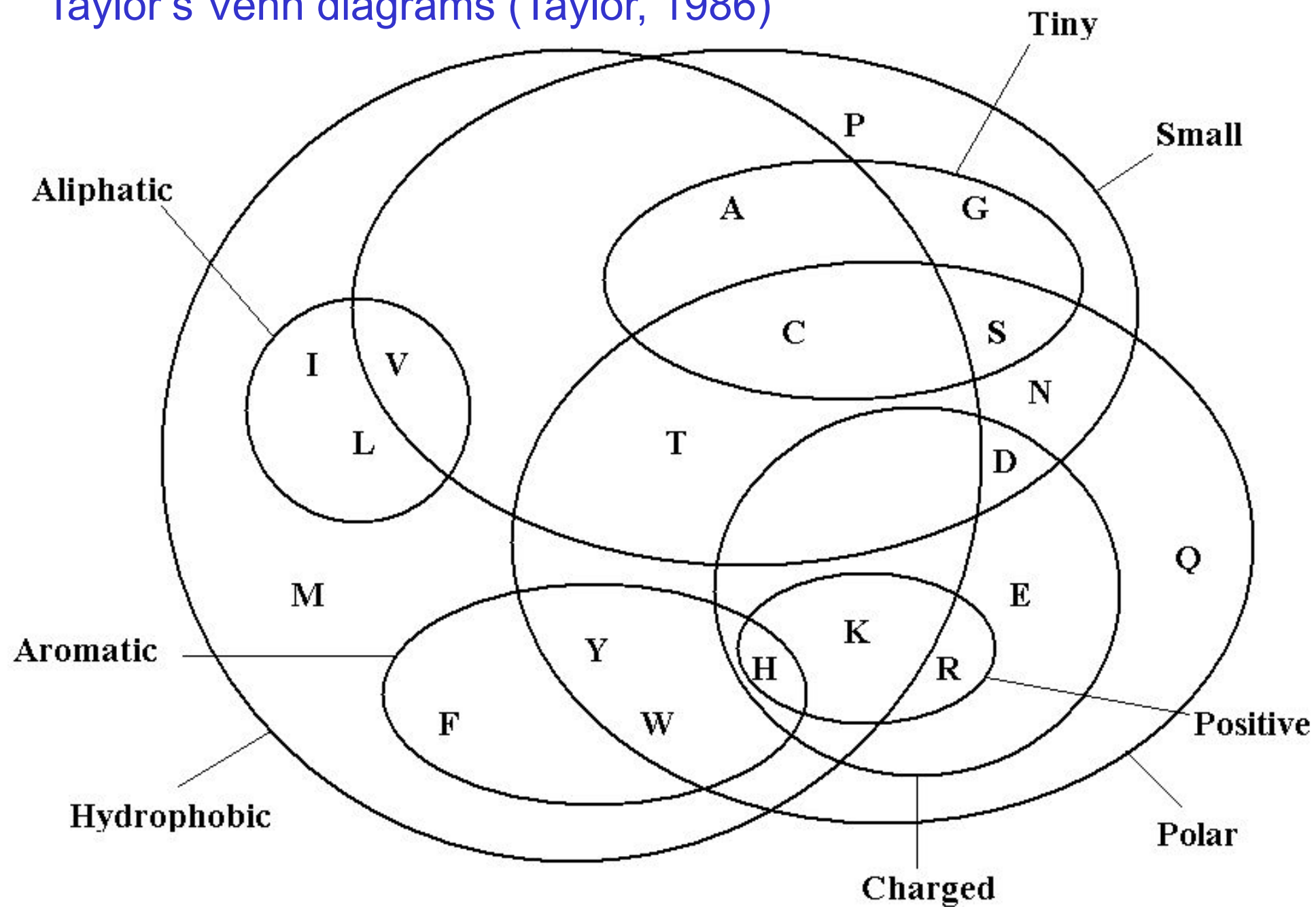
4 nucleotides - 20 amino acids



Scoring matrices for amino acid sequences

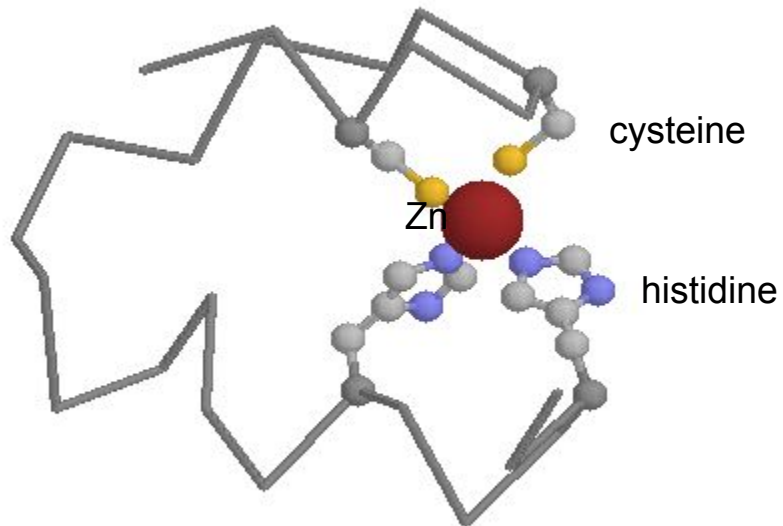
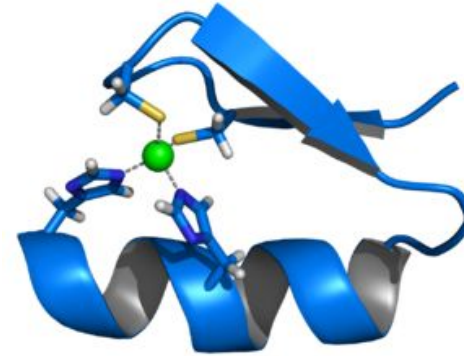
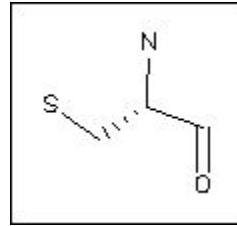
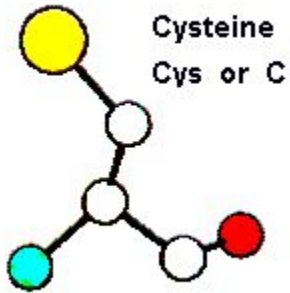
- Are complicated, scoring has to reflect:
 - Physio-chemical properties of aa's
 - Likelihood of residues being substituted among truly homologous sequences
- Certain aa with similar properties can be more easily substituted: preserve structure/function
- “Disruptive” substitution is less likely to be selected in evolution (non functional proteins)

Taylor's Venn diagrams (Taylor, 1986)



Example of environment:

Cysteines are very common in metal binding motifs



Zinc finger (@wiki)

Now let's think about alignments

- Let's consider a simple alignment: ungapped global alignment of two (protein) sequences, x and y , of length n .
- In scoring this alignment, we would like to assess whether these two sequences have a common ancestor, or whether they are aligned by chance.

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

← sequences have common ancestor (Match)

← sequences are aligned by chance (Random)

- We therefore want our amino acid substitution table (matrix) to score an alignment by estimating this ratio (= improvement over random).
- In brief, each substitution score is the **log-odds probability** that amino acid a could change (mutate) into amino acid b through evolution, based on the constraints of our evolutionary model.

Target and background probabilities

- Background probability

If q_a is the frequency of amino acid a in one sequence and q_b is the frequency of amino acid b in another sequence, then the probability of the alignment being random is given by:

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

A	A	R	S
V	V	K	S

- Target probability

If p_{ab} is now the probability that amino acids a and b have derived from a common ancestor, then the probability that the alignment is due to common ancestry is given by:

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

A	A	R	S
V	V	K	S

Source of target and background probabilities:

high confidence alignments

- Target frequencies
 - The “evolutionary true” alignments allow us to get statistics on biologically permissible amino acid mutations and derive the frequencies of observed pairs.
These are the TARGET frequencies (20x20 combinations).
- Background frequencies
 - The BACKGROUND frequencies are simply the frequencies at which each amino acid type is observed in these “trusted” data sets (20 values).

Log-odds

- Substitution matrices apply logarithmic conversions to describe the probability of amino acid substitutions
- The converted values are the so-called **log-odds** scores
- So they are simply the logarithmic ratios of the observed mutation frequency *divided* by the probability of substitution expected by random chance (*target – background*)

Formulas

- Odds-ratio of two probabilities

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

- Log-odds probability of an alignment being random is therefore given by

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

$$\log \left(\prod_i x \right) = \sum_i \log x$$

So... for a given substitution matrix:

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

- **a positive score**

means that the frequency of amino acid substitutions found in the high-confidence alignments is greater than would have occurred by random chance

- **a zero score**

... that the frequency is equal to that expected by chance

- **a negative score**

... that the frequency is less than that expected by chance

Alignment score

- The alignment score S is given by the sum of all amino acid pair substitution scores:

$$S = \sum_i s(x_i, y_i) = \log \frac{\Pr(x, y | M)}{\Pr(x, y | R)}$$

- Where the substitution score for any amino acid pair $[a, b]$ is given by:

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

Alignment score

- The total score of an alignment:

EAAS
VF-T

- would be:

$$S = s(E, V) + s(A, F) + \gamma(1) + s(S, T)$$

Empirical matrices

- Are based on surveys of actual amino acid substitutions among related proteins as observed in sets of protein multiple sequence alignments
- Most widely used: **PAM** and **BLOSUM**

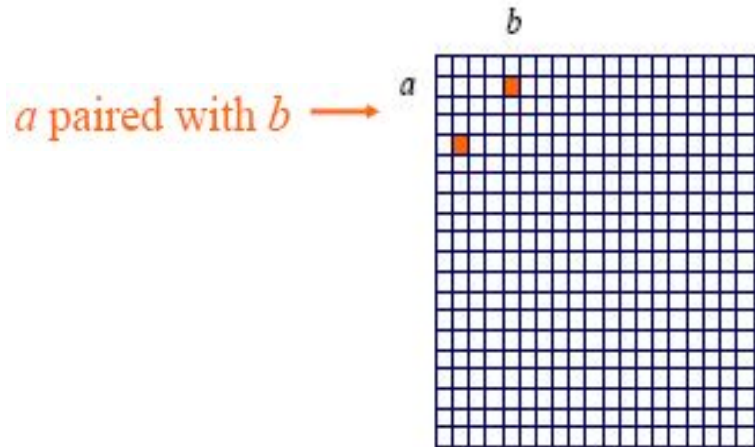
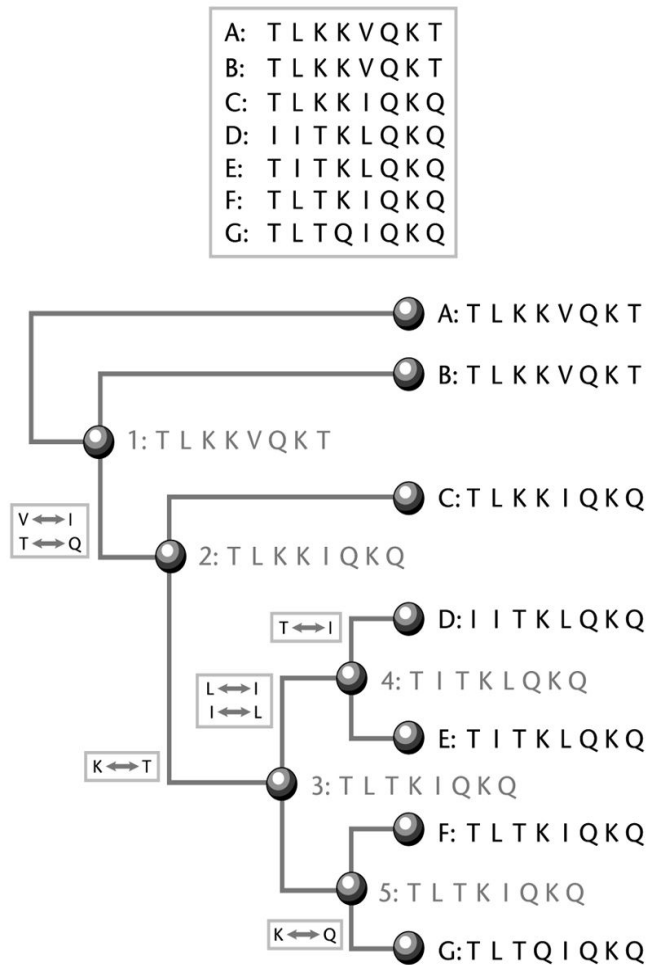
The PAM series



- The first systematic method to derive amino acid substitution matrices was done by Margaret Dayhoff et al. (1978) *Atlas of Protein Structure*.
- These widely used substitution matrices are frequently called Dayhoff, MDM (Mutation Data Matrix), or **PAM** (**Point Accepted Mutation**) matrices.
- **Key idea:** trusted alignments of closely related sequences provide information about biologically permissible mutations.

The PAM design

- **Step 1.** Dayhoff used 71 protein families (each family having closely related family members), made hypothetical phylogenetic trees and recorded the number of observed substitutions (along each branch of the tree) in a 20x20 target matrix.



Mutations are counted in both directions, i.e. $A \rightarrow B = B \rightarrow A$. This leads to a symmetrical mutation matrix A (holding accepted point mutations)

The PAM design

- **Step 2.** The target matrix was then converted to frequencies by dividing each cell (a,b) over the sum of all other substitutions of a.

$$\Pr(b \mid a) = \frac{A_{ab}}{\sum_c A_{ac}}$$

- **Step 3.** The target matrix was normalized so that the expected number of substitutions covered 1% of the protein (PAM-1).

$$\Pr(b \mid a, t = 1)$$

- **Step 4.** Determine the final substitution matrix.

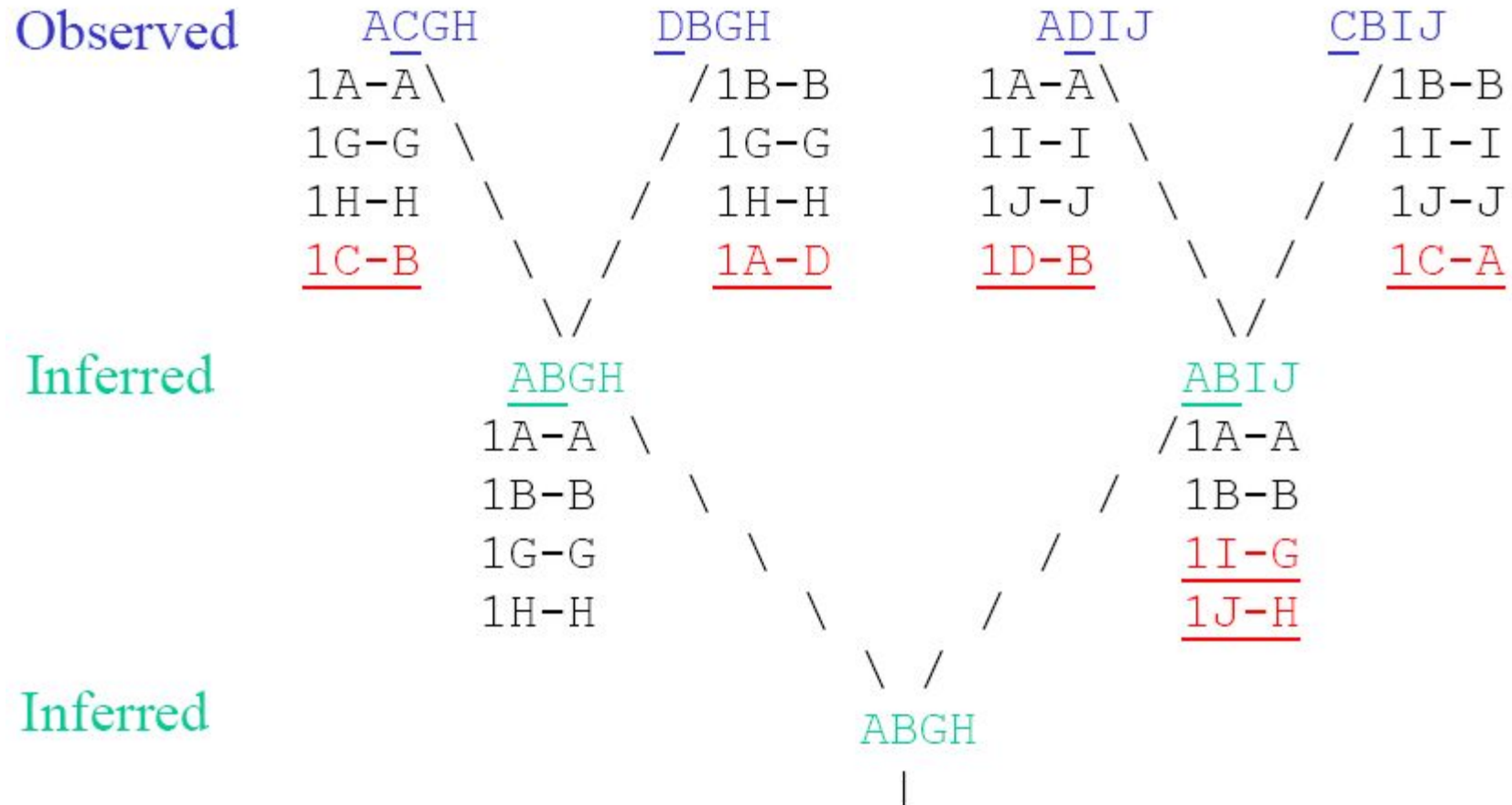
$$s(a, b \mid t) = \log \frac{p_{ab}}{q_a q_b} = \log \frac{P(b \mid a, t)}{q_b}$$

So, how does it work with real data?

- the PAM calculation steps -

Step 1: PAM - Phylogenetic Tree

inferred ancestral sequences and derived mutations



Step 1: PAM - Accepted Point Mutation

	A	B	C	D	G	H	I	J
A	8	0	1	1	0	0	0	0
B	0	8	1	1	0	0	0	0
C	1	1	0	0	0	0	0	0
D	1	1	0	0	0	0	0	0
G	0	0	0	0	6	0	1	0
H	0	0	0	0	0	6	0	1
I	0	0	0	0	1	0	4	0
J	0	0	0	0	0	1	0	4

A_{ij} : number of times amino acid j mutates to amino acid i .

A mutation could go in both directions, therefore the tally of mutation i - j enters both A_{ij} and A_{ji} entries, while the tally of conservation i - i enters A_{ii} entry twice.

The main diagonal elements should be 'doubled' to allow for correct downstream calculations; e.g. the probability of occurrence of each amino acid type (frequencies P_j)

Step 2: Mutability of Residue j

$$m_j = 1 - \frac{A_{jj}}{\sum_{i=1,20} A_{ij}} = \frac{\sum_{i=1,20; i \neq j} A_{ij}}{\sum_{i=1,20} A_{ij}}$$

m_j is the probability that amino acid j will change in a given evolutionary interval. The absolute values of m_j depend on how similar the sequences used to tally A_{ij} are .

Probability of staying the same – normalization is by the total column sum.

Probability of mutating from type j to any other amino acid type – normalization is by the total column sum.

Mutability of Residue j scaled arbitrarily to that of Alanine (which is set to 100)

Ser	149	Met	122	Asn	111	Ile	110	Glu	102
Ala	100	Gln	98	Asp	90	Thr	90	Trp	22
Val	80	Lys	57	Pro	56	His	50	Gly	48
Phe	45	Arg	44	Leu	38	Tyr	34	Cys	27
Gap	84								

The value of Ala (m_{Ala}) has been set arbitrarily to 100 and the values of all other amino acids scaled accordingly. (Adapted from Table 21. Atlas of Protein Sequence and Structure, Suppl 3, 1978, M.O. Dayhoff, ed. National Biomedical Research Foundation, 1979.)

Step 2: Total Mutation Rate

P_j is the probability of random occurrence of amino acid j

$$P_j = \frac{\sum_{i=1,20} A_{ij}}{\sum_{i=1,20} \sum_{j=1,20} A_{ij}}$$

P_j is simply the frequency of amino acid type j (it is called q_a in the earlier equations as it represents the random probability of the amino acid type)

$\sum_{j=1,20} P_j m_j$ is the total mutation rate of all amino acids

Step 3: Normalize Total Mutation Rate to 1%

λ is a scaling constant to make sure that the total mutation rate is 1%

$$\lambda \cdot \sum_{j=1,20} P_j m_j = 1\% \Rightarrow \text{solve for } \lambda$$

This defines an evolutionary period: the period during which the 1% of all sequences are mutated (accepted of course)

Step 3: Mutation Probability Matrix

Normalized such that the total mutation rate is 1%

M_{ij} ($i \neq j$): Probability of amino acid j changing into i in the evolutionary period

$$M_{ij} = \lambda \frac{A_{ij}}{\sum_{i=1,20} A_{ij}} \quad \text{-- Normalised using the matrix columns}$$

M_{jj} : Probability of amino acid j not changing in PAM-1

$$M_{jj} = 1 - \sum_{i=1,20; i \neq j} M_{ij} = 1 - \lambda m_j$$

Step 3: Mutation Probability Matrix (transposed) M*10000

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

Note this matrix (based on 1% mutation) is conservative

Note this matrix is NOT symmetrical (due to column normalization)

PAM Matrix: Assumptions

- The likelihood of amino acid Y replacing X is the same as that of X replacing Y: $P(X \rightarrow Y) = P(Y \rightarrow X)$
- Replacement at any site depends only on the amino acid at that site and the probability given by a Markov Model; all positions in a protein are equally mutable (independently and identically distributed – i.i.d.)
- All sequences have average amino acid composition
- Mutations observed in closely related protein sequences are used to extrapolate to more divergent evolutionary distances: $X \rightarrow Z \rightarrow Y$

$M^{(1)}$ == PAM1 Mutation Probability Matrix

$M^{(2)}$ == PAM2 Mutation Probability Matrix?

-- Mutations that happen in twice the evolution period of that for a PAM1

In two PAM1 periods:

- $\{A \rightarrow R\} = \{A \rightarrow A \text{ and } A \rightarrow R\}$ or
 $\{A \rightarrow N \text{ and } N \rightarrow R\}$ or
 $\{A \rightarrow D \text{ and } D \rightarrow R\}$ or
... or
 $\{A \rightarrow V \text{ and } V \rightarrow R\}$

Entries in a PAM-2 Mutation Probability Matrix

$\Pr(A \rightarrow R \text{ in 2 periods}) =$

$\Pr(A \rightarrow A \text{ in 1st period}) \times \Pr(A \rightarrow R \text{ in 2nd period}) +$

$\Pr(A \rightarrow N \text{ in 1st period}) \times \Pr(N \rightarrow R \text{ in 2nd period}) +$

$\Pr(A \rightarrow D \text{ in 1st period}) \times \Pr(D \rightarrow R \text{ in 2nd period}) +$

\boxtimes

$$P_{AR}^{(2)} = P_{AA} \cdot P_{AR} + P_{AN} \cdot P_{NR} + P_{AD} \cdot P_{DR} + \boxtimes$$

PAM-*K* Mutation Prob. Matrix

$$M^{(2)} = M^{(1)} \times M^{(1)}$$

$$M^{(K)} = \{M^{(1)}\}^K$$

So you get a PAM-*K* Mutation Probability Matrix (representing *K* periods of time associated with PAM-1) by self-multiplying the PAM-1 Mutation Probability Matrix *K* times

PAM units

- One **PAM unit** is defined as 1% of the amino acids positions that have changed
- to construct the **PAM1** substitution table, Dayhoff used 71 protein families comprising closely related sequences (see earlier slide) from which she derived mutation frequencies corresponding to **one** PAM unit.
- One PAM unit corresponds to about 1 million years of evolutionary time.
- How do you extend from **PAM1** (1 million years) to **PAM2** (2 million years), **PAM3** (3 million years), etc.
 - A->C in 2 million years develops via A->B->C (over all possible B residues)

How to get to a whole series of matrices:

PAM10 ... PAM250

- These matrices are extrapolated from **PAM1** matrix (by matrix multiplication)

The diagram illustrates the process of extrapolating PAM matrices. It shows four 20x20 matrices, each representing a different PAM matrix (PAM1, PAM10, PAM25, and PAM50). The matrices are arranged in a sequence, with 'x' symbols between the first three and an '=' symbol before the fourth, indicating that the product of these matrices results in a higher-order PAM matrix (PAM125).

Self-multiply Matrix N times to make PAM ' N '; then take the Log (see next slide)

- So:** a PAM is a relative measure of evolutionary distance
 - 1 PAM = 1 accepted point mutation per 100 amino acids
 - 250 PAM = 250 mutations per 100 amino acids, so 2.5 accepted mutations per amino acid!

Step 4: PAM-k log-likelihood matrix

$$S_{ij} = 10 \log_{10} \frac{(M^K)_{ij}}{P_i}$$

Note that only the frequency of amino acid type i (P_i) is used to normalise the mutation probabilities, because the M_{ij} values are already normalised with respect to the frequency of amino acid type j .

P_i is the probability of random occurrence of amino acid i

$$P_i = \frac{\sum_{j=1,20} A_{ij}}{\sum_{i=1,20} \sum_{j=1,20} A_{ij}}$$

S is a symmetric matrix

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

The thus converted scores (after rounding) represent the PAM matrices as we know them.

PAM numbers vs. observed amino acid mutational rates

PAM Number	Observed Mutation Rate (%)	Sequence Identity (%)
0	0	100
1	1	99
30	25	75
80	50	50
110	40	60
200	75	25
250	80	20

The PAM250 matrix

[illegible]

W-R exchange probability
is too large (due to
paucity of data)

Scoring matrices for amino acid sequences

- Are complicated, scoring has to reflect:
 - Physio-chemical properties of aa's
 - Likelihood of residues being substituted among truly homologous sequences
- Certain aa with similar properties can be more easily substituted: preserve structure/function
- “Disruptive” substitution is less likely to be selected in evolution (e.g. rendering non-functional proteins)
- Dayhoff (and Blosom) approach is an empirical way of getting residue exchange probabilities using observed mutations in groups of aligned sequences

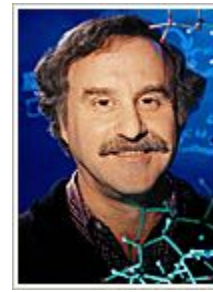
PAM model

- The scores derived through the PAM model are an accurate description of the information content (or the relative entropy) of an alignment (Altschul, 1991).
- **PAM1** corresponds to about 1 million years of evolution.
- **PAM120** has the largest information content of the PAM matrix series: “best” for general alignment (Altschul, 1991)
- Nonetheless, **PAM250** is the traditionally most popular matrix: “best” for detecting distant sequence similarity.

Summary Dayhoff's PAM-matrices

- Derived from global alignments of closely related sequences.
- Matrices for greater evolutionary distances are extrapolated from those for smaller ones
 - Evolutionary clock (mutation rate) is supposed to be uniform (equal over evolutionary time)
- The number with the matrix (**PAM40**, **PAM100**) refers to the evolutionary distance; greater numbers are greater distances.
- Attempts to extend Dayhoff's methodology or re-apply her analysis using databases with more examples:
 - Jones, Thornton and coworkers used the same methodology as Dayhoff but with modern databases (CABIOS 8:275, 1992)
 - Gonnett and coworkers (Science 256:1443, 1992) used a slightly different (but theoretically equivalent) methodology

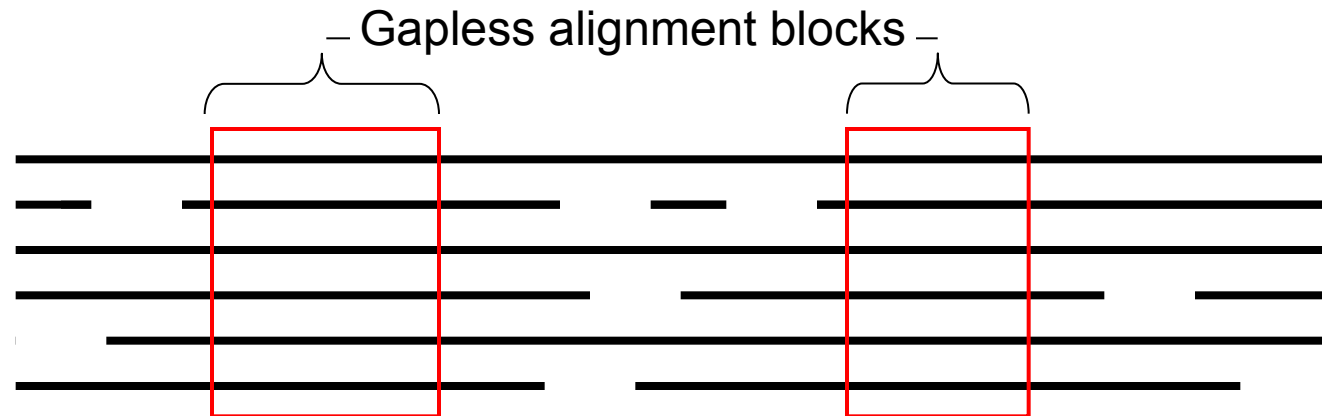
The BLOSUM series



- BLOSUM stands for: **B**LOcks **S**Ubstitution **M**atrices
- Created by Steve Henikoff and Jorja Henikoff (PNAS 89:10915).
- Derived from local, un-gapped alignments of distantly related sequences (the BLOCKS database)
- All matrices are directly calculated; no extrapolations (PAM) are used.
- As with PAM: compare observed freqs of each pair to expected freqs -- Then: Log-odds matrix.

The **Blocks** database

- The Blocks Database contains multiple alignments of conserved regions in protein families.
- Blocks are multiply aligned un-gapped segments corresponding to the most highly conserved regions of proteins.
- The blocks for the BLOCKS database are made automatically by looking for the most highly conserved regions in groups of proteins represented in the PROSITE database. These blocks are then calibrated against the SWISS-PROT database to obtain a measure of the random distribution of matches. It is these calibrated blocks that make up the BLOCKS database.



The BLOSUM series

- **BLOSUM**30, 35, 40, 45, 50, 55, 60, 62, 65, 70, 75, 80, 85, 90.
- The number after the matrix (**BLOSUM**62) refers to the minimum percent identity of the blocks (in the BLOCKS database) used to construct the matrix (for BLOSUM62 all blocks have $\geq 62\%$ sequence identity);
 - e.g., for BLOSUM30 blocks are used with $\geq 30\%$ sequence identity
- No extrapolations are made in going to higher evolutionary distances
 - Each time another selection of the BLOCKS alignments is used
- High number - closely related sequences
Low number - distant sequences
- BLOSUM62 is the most popular exchange matrix: best for general alignment.

A toy example of constructing
a BLOSUM matrix from 4
training sequences

Constructing a BLOSUM matr.

1. Counting mutations

VVAPV

AAAPA

PVAPV

PAAAV

$$N_{AA} = 0 + 1 + (4*3/2) + 0 + 0 = 7$$

$$N_{VV} = 0 + 1 + 0 + 0 + (3*2)/2 = 4$$

$$N_{PP} = 1 + 0 + 0 + (3*2)/2 + 0 = 4$$

$$N_{AV} = N_{VA} = 1 + 2*2 + 0 + 0 + 3 = 8$$

$$N_{AP} = N_{PA} = 2 + 0 + 0 + 3 + 0 = 5$$

$$N_{PV} = N_{VP} = 2 + 0 + 0 + 0 + 0 = 2$$

N_{VP} is the number of V-P pairs

2. Tallying mutation frequencies

$q_{ij} =$	A	V	P
A	14	8	5
V	8	8	2
P	5	2	8

q_{ij} : number of times amino acid j mutates to amino acid i .

A mutation could go in both directions, therefore the tally of A-V pair enters both q_{AV} and q_{VA} entries, while the tally of A-A pair enters q_{AA} entry twice.

3. Matrix of mutation probs.

p_{ij} is the probability that a mutation occurs between amino acid i and amino acid j

$$p_{ij} = \frac{q_{ij}}{\sum_{i=1,20} \sum_{j=1,20} q_{ij}}$$

$p_{ij} =$	A	V	P
A	14/60	8/60	5/60
V	8/60	8/60	2/60
P	5/60	2/60	8/60

4. Calculate abundance of each residue (Marginal prob)

p_i is the marginal probability, meaning the expected probability of occurrence of amino acid i

$$p_i = \frac{\sum_{j=1,20} q_{ij}}{\sum_{i=1,20} \sum_{j=1,20} q_{ij}}$$

$p_i =$	A	V	P
	9/20	6/20	5/20
	(27/60)	(18/60)	(15/60)

5. Obtaining a BLOSUM matrix

The BLOSUM log-likelihood matrix:

$$S_{ij} = 2 \log_2 \frac{p_{ij}}{p_i p_j}$$

$S_{ij} =$	A	V	P
A	0.409		
V	-0.036	1.134	
P	-0.866	-2.34	2.19

Constructing the real BLOSUM62 Matrix

1.2.3.Mutation Frequency Table

$P_{ij} \cdot 1000$

A	215																				
R	23	178																			
N	19	20	141																		
D	22	16	37	213																	
C	16	4	4	4	119																
Q	19	25	15	16	3	73															
E	30	27	22	49	4	35	161														
G	58	17	29	25	8	14	19	378													
H	11	12	14	10	2	10	14	10	93												
I	32	12	10	12	11	9	12	14	6	184											
L	44	24	14	15	16	16	20	21	10	114	371										
K	33	62	24	24	5	31	41	25	12	16	25	161									
M	13	8	5	5	4	7	7	7	4	25	49	9	40								
F	16	9	8	8	5	5	9	12	8	30	54	9	12	183							
P	22	10	9	12	4	8	14	14	5	10	14	16	4	5	191						
S	63	23	31	28	10	19	30	38	11	17	24	31	9	12	17	126					
T	37	18	22	19	9	14	20	22	7	27	33	23	10	12	14	47	125				
W	04	3	2	2	1	2	3	4	2	4	7	3	2	8	1	3	3	65			
Y	13	9	7	6	3	7	9	8	15	14	22	10	6	42	5	10	9	9	102		
V	51	16	12	13	14	12	17	18	6	120	95	19	23	26	12	24	36	4	15	196	
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	

4. Calculate Amino Acid Abundance

$$p_i = \sum_j p_{ij}$$

A:	0.074	R:	0.052	N:	0.045	D:	0.054	C:	0.025
Q:	0.034	E:	0.054	G:	0.074	H:	0.026	I:	0.068
L:	0.099	K:	0.058	M:	0.025	F:	0.047	P:	0.039
S:	0.057	T:	0.051	W:	0.013	Y:	0.032	V:	0.073

5. Obtaining BLOSUM62 Matrix

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

$$S_{ij} = 2 \cdot \log_2 \frac{p_{ij}}{p_i p_j}$$

The log-odds matrix for BLOSUM62

BLOSUM62																				
A	4																			
R	-1	5																		
N	-2	0	6																	
D	-2	-2	1	6																
C	0	-3	-3	-3	9															
Q	-1	1	0	0	-3	5														
E	-1	0	0	2	-4	2	5													
G	0	-2	0	-1	-3	-2	-2	6												
H	-2	0	1	-1	-3	0	0	-2	8											
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4
X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	0	0	-2	-1	-1
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V

Positive for chemically similar substitution

Common amino acids have low weights

Rare amino acids have high weights

Higher random chance to become aligned (observed freqs versus expected freqs)

PAM *versus* BLOSUM

- Based on an explicit evolutionary model
- Derived from small, closely related proteins with ~15% divergence
- Higher PAM numbers to detect more remote sequence similarities
- Errors in PAM 1 are scaled 250X in PAM 250

- Based on empirical frequencies
- Uses much larger, more diverse set of protein sequences (30-90% ID)
- Lower BLOSUM numbers to detect more remote sequence similarities
- Errors in BLOSUM arise from errors in alignment

Comparing exchange matrices

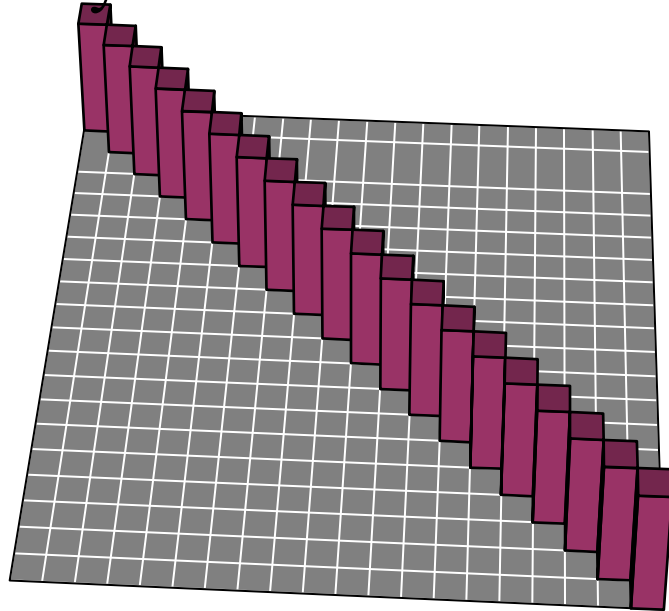
- To compare amino acid exchange matrices, the "Entropy" value can be used. This is a relative entropy value (H) which describes the amount of information available per aligned residue pair.

$$H = \sum s_{ij} \log_2(s_{ij} / p_i p_j)$$

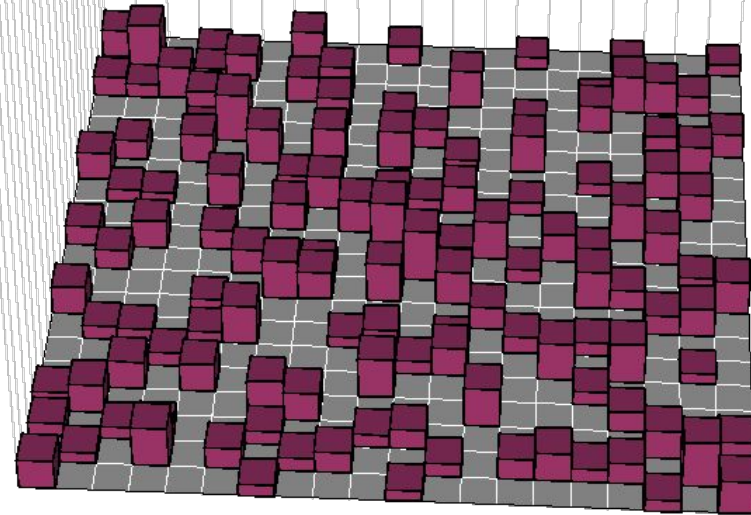
NAME	CLUSTER	#blocks	#PAIRS	ENTROPY	EXPECTED	NAME	CLUSTER	#blocks	#PAIRS	ENTROPY	EXPECTED
	#	USED					#	USED			
Blosum30	30*	261	70,394	.1424	-.1074	Pam160				.70	-.57
Pam350				.20		Blosum63	63	1606	1,322,720	.7218	-.5389
Pam340				.21		Blosum65	65	1671	1,503,060	.7576	-.5675
Blosum35	35	439	136,328	.2111	-.1550	Pam150				.76	-.61
Pam330				.22		Pam140				.82	-.66
Pam290				.28		Blosum70	70	1803	1,930,219	.8391	-.6313
Blosum40	40	672	227,179	.2851	-.2090	Pam130				.90	
Pam280				.30		Blosum75	75	1896	2,365,918	.9077	-.6845
Pam250a				.36	-.28	Pam120				.98	-.81
Blosum45	45	900	357,558	.3795	-.2789	Blosum80	80	1941	2,907,119	.9868	-.7442
Pam240				.39		Pam110				1.08	
Pam210				.48		Blosum85	85	1993	3,861,389	1.0805	-.8153
Blosum50	50	1161	557,053	.4808	-.3573	Pam100				1.18	-.98
Pam200				.51		Blosum90	90	2022	5,196,613	1.1806	-.8887
Pam190				.55		Blosum95	95	2050	7,185,859	1.2926	-.9743
Blosum55	55	1331	753,984	.5637	-.4179	Pam90				1.30	
Pam180				.60		Pam80				1.44	
Pam170				.65		Blosum100	100	2089	11,066,715	1.4516	-1.0948
Blosum60	60	1529	1,087,140	.6603	-.4917	Blosumn	no	2106	15,255,643	1.5172	-1.1484
Blosum62	62	1572	1,245,852	.6979	-.5209	Pam70				1.60	

Evolution and Matrix “landscape”

- Recent evolution
→ identity matrix

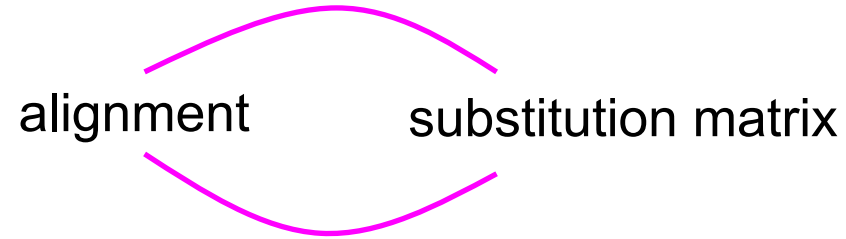


- Ancient evolution
→ convergence to random model



A note on reliability

- All these matrices are designed using empirical evolutionary models.
- Circular problem
- It is important to understand that evolution is not the same for all proteins, not even for the same regions of proteins.

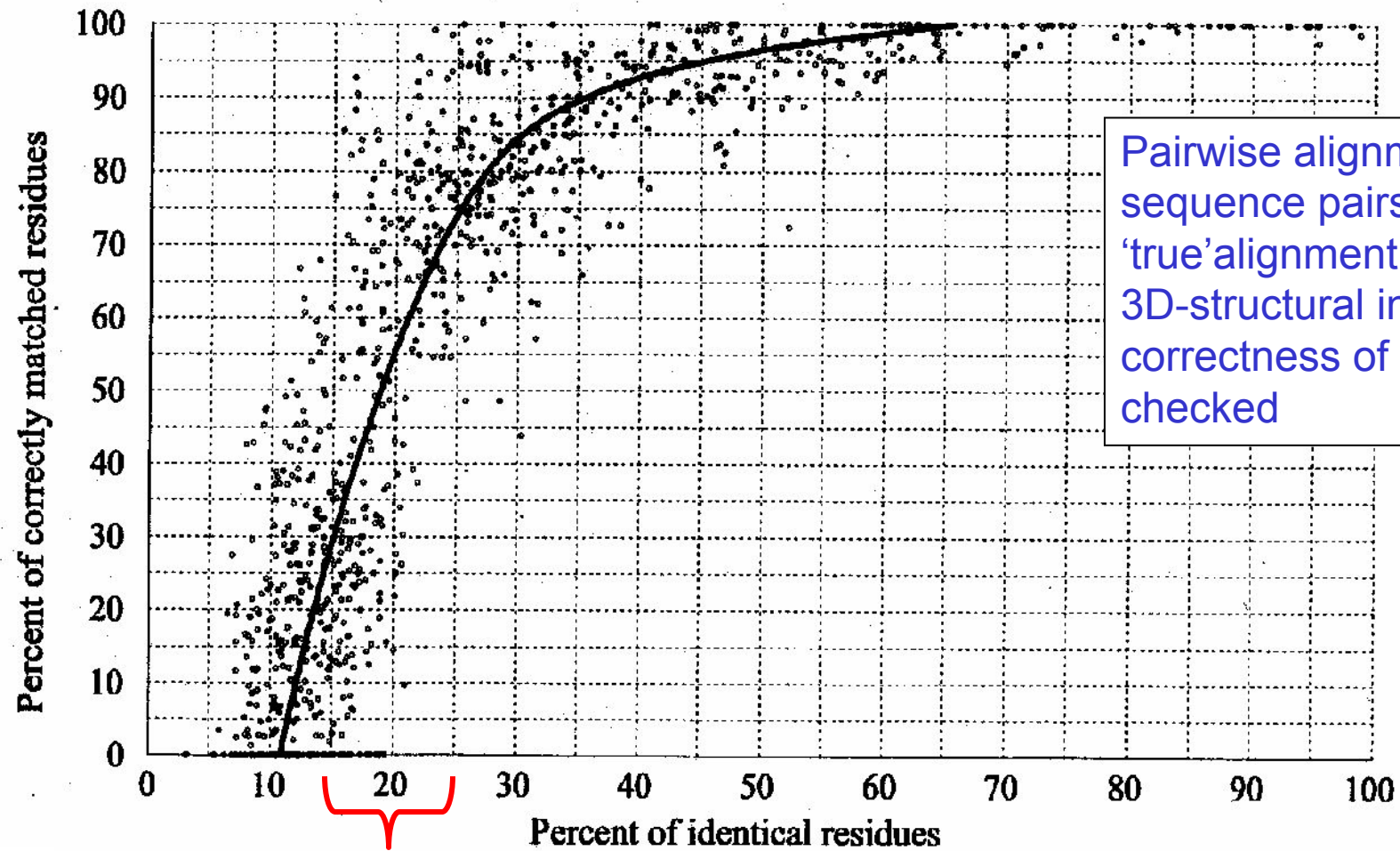


In practice

- No single matrix performs best on all sequences. Some are better for sequences with few gaps, and others are better for sequences with fewer identical amino acids.
- Therefore, when aligning sequences, applying a general model to all cases is not ideal.
Rather, re-adjustment can be used to make the general model better fit the given data.
- In practice, matrices such as BLOSUM62 (or sometimes BLOSUM50) are used in the vast majority of cases

Pair-wise alignment quality *versus* sequence identity

- Vogt et al., JMB 249, 816-831, 1995



Pairwise alignments were made of sequence pairs for which the 'true' alignment was known from 3D-structural information, so the correctness of the alignments could be checked

The twilight zone (alignment identities between 15% and 25%) shows alignment error rates between 25% and 70%...

Twilight zone

Take-home messages - 1

- If ORF exists, then align at protein level.
- Amino acid substitution matrices reflect the log-odds ratio between the evolutionary and random model and can therefore help in determining homology via the alignment score.
- The evolutionary and random models depend on generalized data sets used to derive them. This is not an ideal solution.

Take-home messages - 2

- Apart from the PAM and BLOSUM series, a great number of further matrices have been developed.
- Matrices have been made based on DNA, protein structure, information content, etc.
- For local alignment, BLOSUM62 is often superior; for distant (global) alignments, BLOSUM50, GONNET, JTT or (still) PAM250 work well.
- **Remember that gap penalties are always a problem:** unlike the matrices themselves, there is no formal way to calculate their values -- you can follow recommended settings, but these are based on trial and error and not on a formal framework.

APPENDIX

Review of three pairwise
alignment types

There are three kinds of alignments

- **Global alignment**

(Saul B. Needleman and Christian D. Wunsch, 1970)

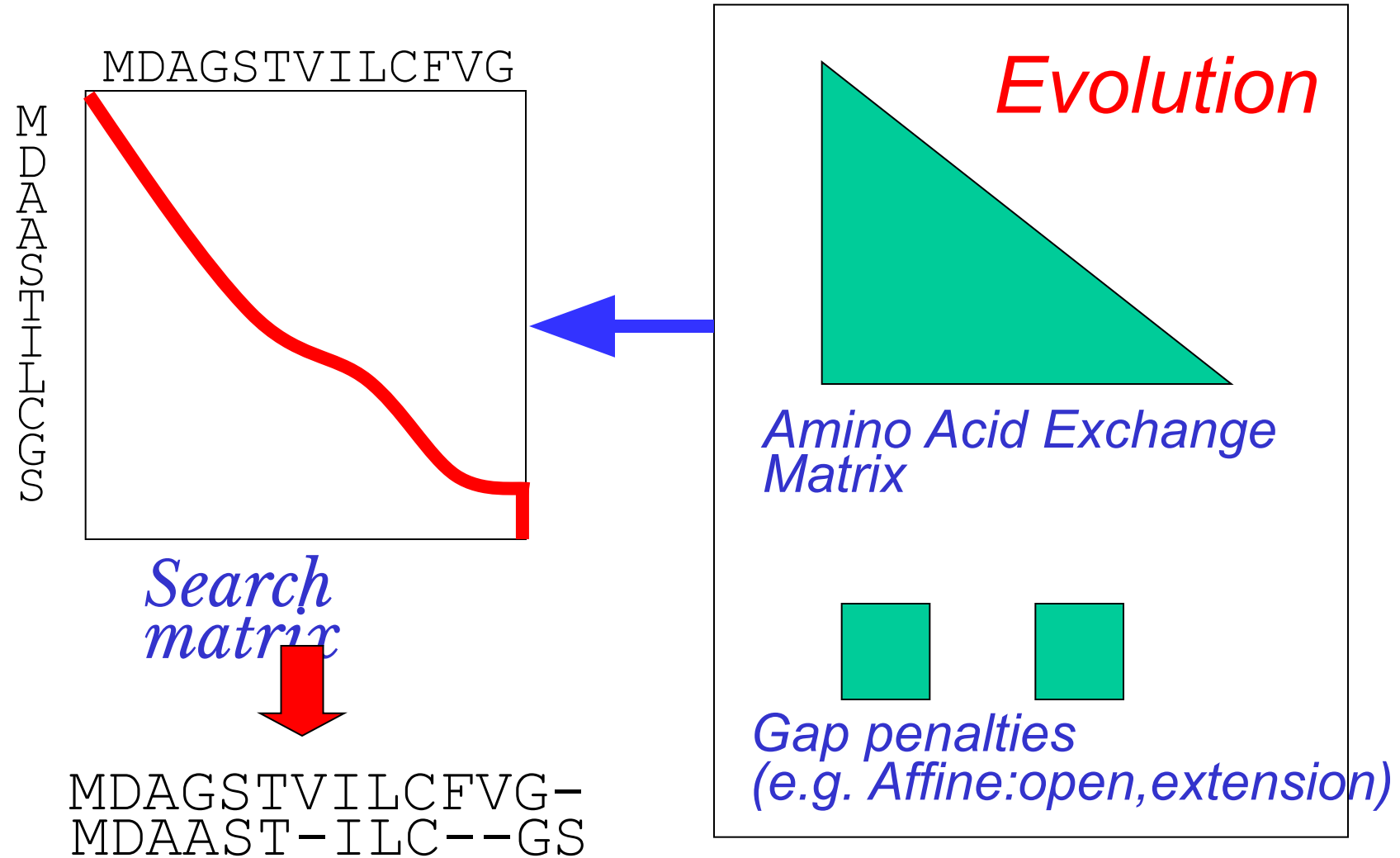
- Semi-global alignment

- Local alignment

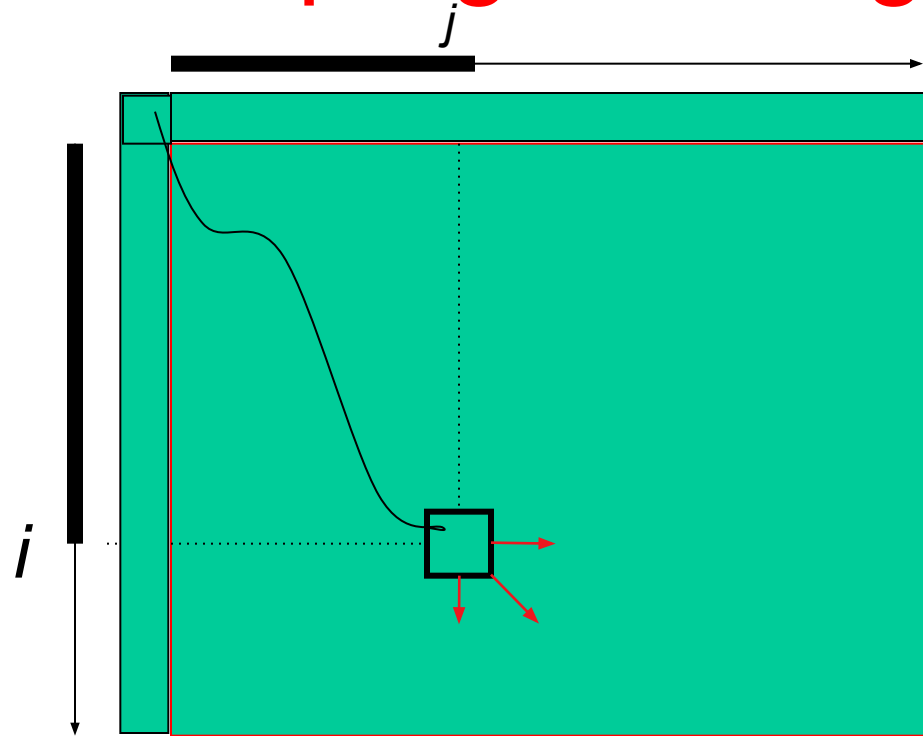
Alignment examples work with scoring alignments (higher scores denoting more similarity between sequences)

Pairwise sequence alignment

Global dynamic programming



Dynamic programming

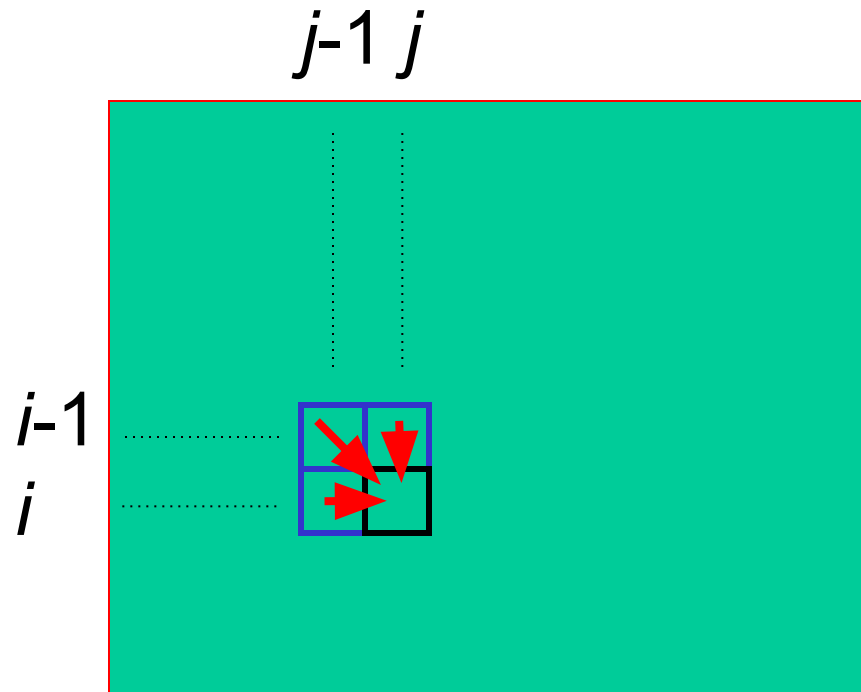


The cell $[i, j]$ contains the alignment score of the best scoring alignment of subsequence $1..i$ and $1..j$, that is, the subsequences up to $[i, j]$

Cell $[i, j]$ does not 'know' what that best scoring alignment is; it is one (or a number of alternatives) out of very many possibilities, leading to $[i, j]$

Extend alignment from cell $[i, j]$

Global dynamic programming



Value from residue
exchange matrix (or
match/mismatch)

$$H(i,j) = \text{Max} \begin{cases} H(i-1,j-1) + s(i,j) & \text{diagonal} \\ H(i-1,j) - g & \text{vertical} \\ H(i,j-1) - g & \text{horizontal} \end{cases}$$

This is a recursive formula

Example:

global alignment of two sequences

- Align two DNA sequences:
 - GAGTGA
 - GAGGCGA (note the length difference)
- Parameters of the algorithm:
 - Match: $\text{score}(A,A) = 1$
 - Mismatch: $\text{score}(A,T) = -1$
 - Gap: $g = 2$

$$M[i, j] = \max \begin{cases} M[i-1, j-1] & \pm 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \end{cases}$$

The algorithm. Step 1: init

- Create the matrix
- Initiation
 - 0 at [0,0]
 - Apply the equation...

$$M[i, j] = \max \begin{cases} M[i-1, j-1] + 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \end{cases}$$

	j→	0	1	2	3	4	5	6
i↓		-	G	A	G	T	G	A
0	-	0						
1	G							
2	A							
3	G							
4	G							
5	C							
6	G							
7	A							

The algorithm. Step 1: init

$$M[i, j] = \max \begin{cases} M[i-1, j-1] + 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \end{cases}$$

Initialisation of the matrix:


- 0 at pos [0,0]
- Fill in the initialisation row using the “→” rule
- Fill in the initialisation column using “↓” rule

		<i>j</i>							
		<i>j</i> →	-	G	A	G	T	G	A
<i>i</i> ↓			0	-2	-4	-6	-8	-10	-12
1	G	-2							
2	A	-4							
3	G	-6							
4	G	-8							
5	C	-10							
6	G	-12							
7	A	-14							

First row

First column

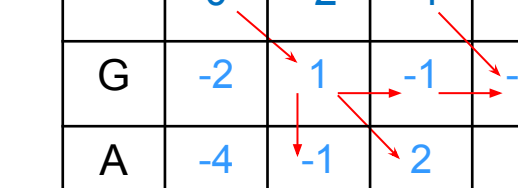
The algorithm. Step 2: fill in

$$M[i, j] = \max \begin{cases} M[i-1, j-1] + 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \end{cases}$$


- Continue filling in the matrix, remembering from which cell the result comes (arrows)

j

	-	G	A	G	T	G	A
-	0	-2	-4	-6	-8	-10	-12
G	-2	1	-1	-3			
A	-4	-1	2				
<i>i</i> G	-6						
G	-8						
C	-10						
G	-12						
A	-14						



The algorithm. Step 2: fill in

$$M[i, j] = \max \begin{cases} M[i-1, j-1] + 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \end{cases}$$

- Continue filling in the matrix, remembering from which cell the result comes (arrows)

j

	-	G	A	G	T	G	A
-	0	-2	-4	-6	-8	-10	-12
G	-2	1	-1	-3	---	---	---
A	-4	-1	2				
G	-6						
G	-8						
C	-10						
G	-12						
A	-14						


i

$\begin{matrix} i \\ j \end{matrix} \begin{matrix} G & - & - \\ G & A & G \end{matrix}$

or

$\begin{matrix} i \\ j \end{matrix} \begin{matrix} - & - & G \\ G & A & G \end{matrix}$

The algorithm. Step 2: fill in

$$M[i, j] = \max \begin{cases} M[i-1, j-1] + 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \end{cases}$$



- We are done...
- Where's the result?

j

	-	G	A	G	T	G	A
-	0	-2	-4	-6	-8	-10	-12
G	-2	1	-1	-3	-5	-7	-9
A	-4	-1	2	0	-2	-4	-6
<i>i</i> G	-6	-3	0	3	1	-1	-3
G	-8	-5	-2	1	2	2	0
C	-10	-7	-4	-1	0	1	1
G	-12	-9	-6	-3	-2	1	0
A	-14	-11	-8	-5	-4	-1	2

(arrows left out for clarity)

The algorithm. Step 2: fill in

$$M[i, j] = \max \begin{cases} M[i-1, j-1] \pm 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \end{cases}$$


- We are done...
- Where's the result?

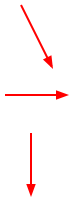
The
lowest-rightmo
st cell

j

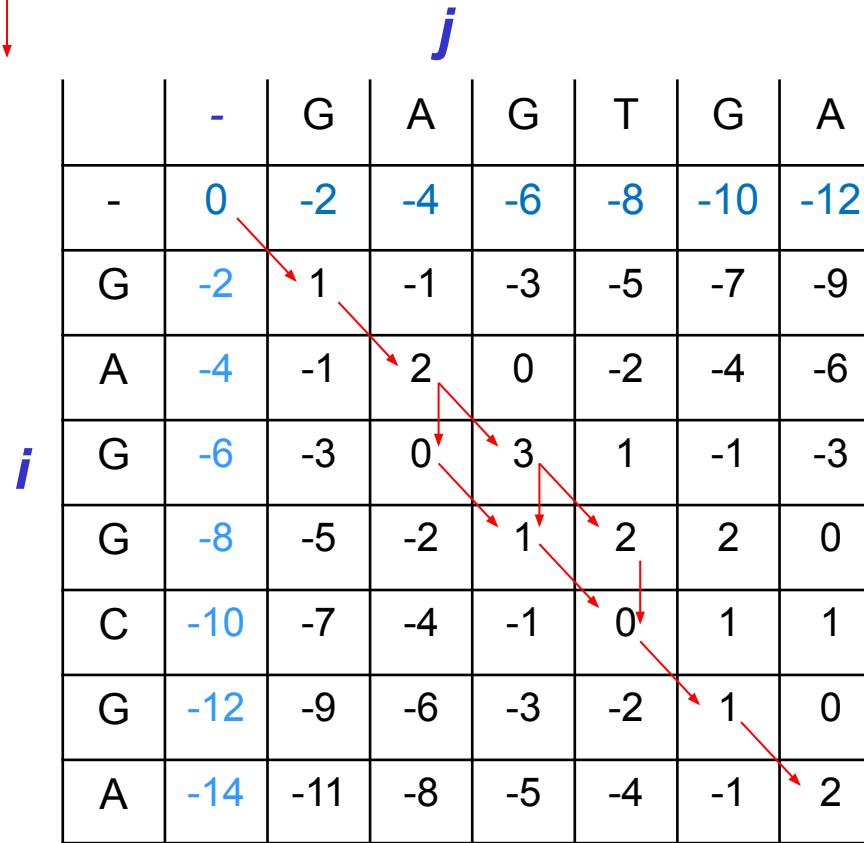
	-	G	A	G	T	G	A
-	0	-2	-4	-6	-8	-10	-12
G	-2	1	-1	-3	-5	-7	-9
A	-4	-1	2	0	-2	-4	-6
<i>i</i> G	-6	-3	0	3	1	-1	-3
G	-8	-5	-2	1	2	2	0
C	-10	-7	-4	-1	0	1	1
G	-12	-9	-6	-3	-2	1	0
A	-14	-11	-8	-5	-4	-1	2

(arrows left out for clarity)

The algorithm. Step 3: trace-back

$$M[i, j] = \max \begin{cases} M[i-1, j-1] + 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \end{cases}$$


- Start at the lowest rightmost cell of the matrix
- Go against the direction of arrows
- Sometimes the value may be obtained from more than one cell (which ones?)



	-	G	A	G	T	G	A
-	0	-2	-4	-6	-8	-10	-12
G	-2	1	-1	-3	-5	-7	-9
A	-4	-1	2	0	-2	-4	-6
G	-6	-3	0	3	1	-1	-3
G	-8	-5	-2	1	2	2	0
C	-10	-7	-4	-1	0	1	1
G	-12	-9	-6	-3	-2	1	0
A	-14	-11	-8	-5	-4	-1	2

Trace-back: follow arrows back
 -- with vertical/horizontal arrows
 the aligned cell is at base of
 arrow (top/leftmost point)

The algorithm. Step 3: trace-back

$$M[i, j] = \max \begin{cases} M[i-1, j-1] + 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \end{cases}$$

Extract the alignments:

a) high road

GAGT-GA

GAGGCGA

b) low road

GA-GTGA

GAGGCGA

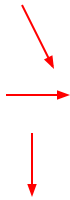
c) 'middle road'

GAG-TGA

GAGGCGA

		<i>j</i>						
		-	G	A	G	T	G	A
	-	0	-2	-4	-6	-8	-10	-12
	G	-2	1	-1	-3	-5	-7	-9
	A	-4	-1	2	0	-2	-4	-6
<i>i</i>	G	-6	-3	0	3	1	-1	-3
	G	-8	-5	-2	1	2	2	0
	C	-10	-7	-4	-1	0	1	1
	G	-12	-9	-6	-3	-2	1	0
	A	-14	-11	-8	-5	-4	-1	2

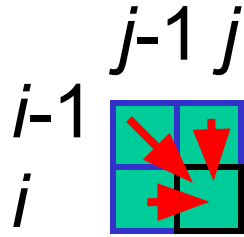
The algorithm. Step 3: trace-back

$$M[i, j] = \max \begin{cases} M[i-1, j-1] + 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \end{cases}$$


QUESTION:

How can one change a 'high road' algorithm into a 'low road' one, or vice versa?

Global dynamic programming



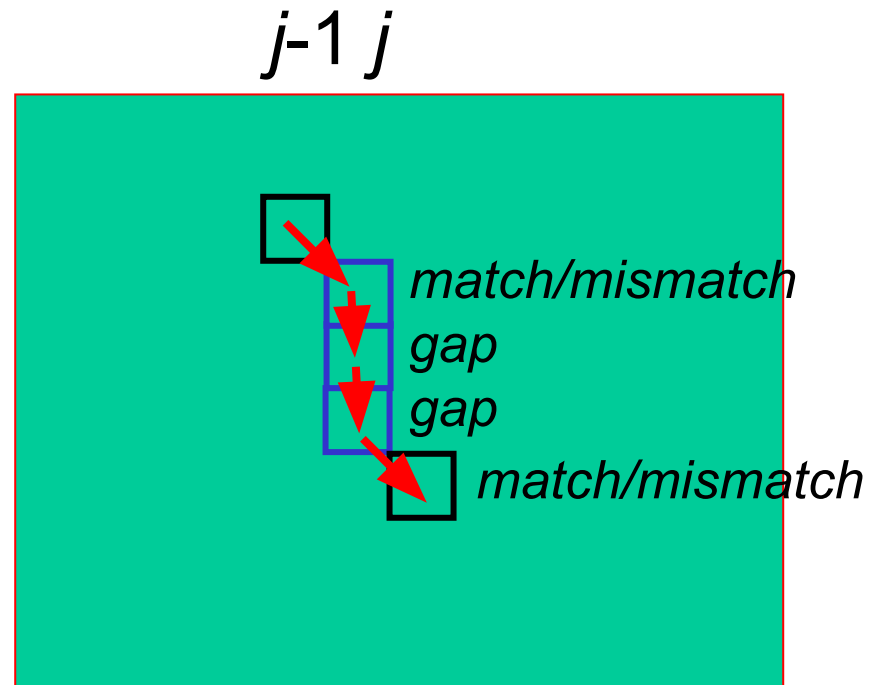
$$H(i,j) = \text{Max} \begin{cases} H(i-1,j-1) + S(i,j) & \text{diagonal} \\ H(i-1,j) - g & \text{vertical} \\ H(i,j-1) - g & \text{horizontal} \end{cases}$$

Two-step approach (+ initialisation of preceding row and column with gap penalties):

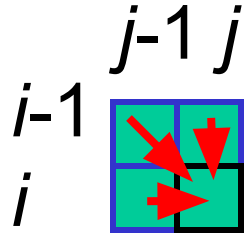
1. Forward step: calculating matrix cells
 - Final alignment score in lower rightmost cell
2. Traceback step: reconstructing the alignment path and hence the alignment itself

Global dynamic programming

Trace back



Global dynamic programming



$$H(i,j) = \text{Max} \begin{cases} H(i-1,j-1) + S(i,j) & \text{diagonal} \\ H(i-1,j) - g & \text{vertical} \\ H(i,j-1) - g & \text{horizontal} \end{cases}$$

Problem with simple DP approach:

- **Can only do linear gap penalties**
- Not suitable for affine or concave penalties, but algorithm can be extended to deal with affine penalties

There are three kinds of alignments

- Global alignment (preceding slides)
- **Semi-global alignment**
- Local alignment

Variation on global alignment

- *Global* alignment: previous algorithm is called *global* alignment because it uses all letters from both sequences.

CAGCACTTGGATTCTCGG

CAGC-----G-T-----GG

- *Semi-global* alignment: uses all letters but does not penalize for end gaps

CAGCA-CTTGGATTCTCGG

---CAGCGTGG-----

Semi-global alignment

- Global alignment: all gaps are penalised
- Semi-global alignment: N- and C-terminal (5' and 3') gaps (end-gaps) are not penalised

MSTGAVLIY--TS-----
---GGILLFHRTSGTSNS

End-gaps

End-gaps

Semi-global alignment

Applications of *semi-global*:

- Finding a gene in genome
- Placing marker onto a chromosome
- One sequence much longer than the other



Risk: if gap penalties high -- really bad alignments for divergent sequences



Protein sequences have N- and C-terminal amino acids that are often small and hydrophilic. When sequences are divergent, this may lead to 'head-to-tail' alignments (see alignment on the left)

Semi-global alignment

- Ignore 5' or N-terminal end gaps
 - First row/column set to 0
- Ignore C-terminal or 3' end gaps
 - Read the result from **last row/column** (select the highest scoring cell)

	-	G	A	G	T	G
-	0	0	0	0	0	0
A	0	-1	1	-1	-1	-1
G	0	1	-2	2	0	-2
T	0	-1	0	0	3	1

If the highest scoring cell is not the lowest-rightmost one, so there are one or more end-gaps, then pad the remaining sequence stretch (x or y sequence) with (zero) end-gaps

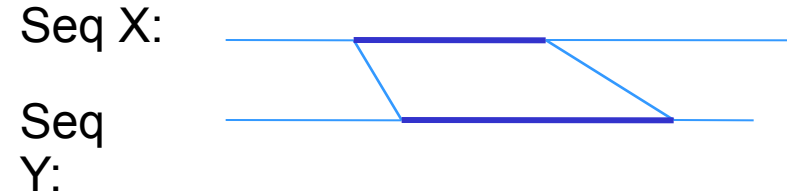
There are three kinds of alignments

- Global alignment (preceding slides)
- Semi-global alignment
- **Local alignment** (Temple F. Smith & Michael S. Waterman, 1981)

There are three kinds of pairwise alignments

- *Global alignment* – align all residues in both sequences; all gaps are penalised
- *Semi-global alignment* – align all residues in both sequences; end gaps are **not** penalised (zero end gap penalties)
- *Local alignment* – align one fragment of each sequence; end gaps are not applicable
 - they can be used in the DP algorithm, but end-gaps are never displayed

Local alignment



- What's local?
 - Allow only parts of the sequence to match
 - Results in High Scoring Segments
 - Locally maximal: cannot make it better by trimming/extending the alignment

Local alignment

Seq X:

Seq
Y:



- Why local?

- Parts of sequence **diverge faster** evolutionary pressure does not put constraints on *the whole* sequence

seq X:

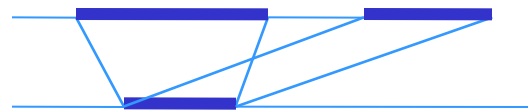
seq Y:



- Proteins have **modular construction** sharing domains between sequences

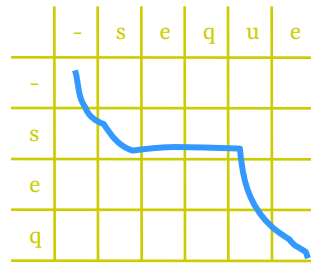
seq X:

seq Y:



Global → local alignment

a) global alignment



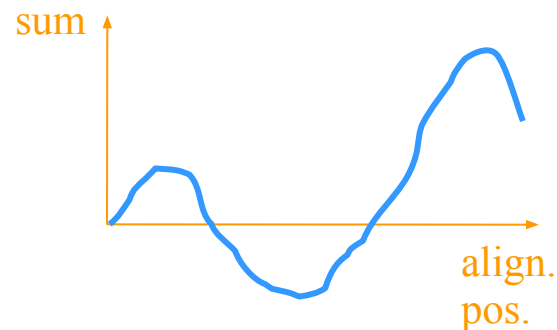
b) retrieve the result

CAGCACTTGGATTCTCG

-

CA-C-----GATTCGT-

c) sum score along the result

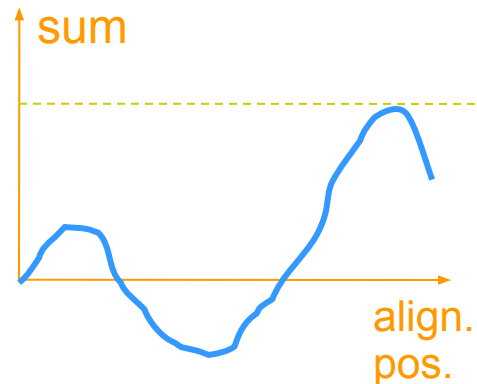
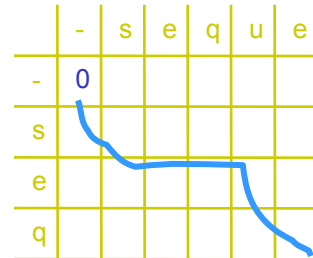


- Take the old, good equation
- Look at the result of the *global* alignment

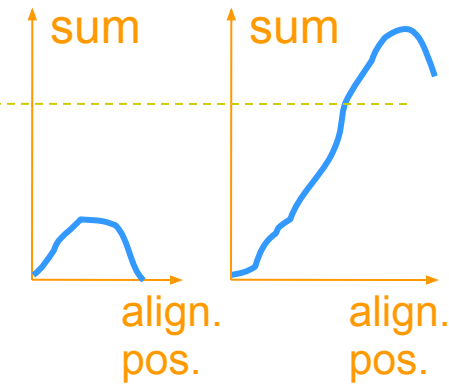
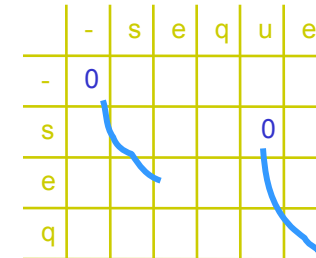
Local alignment – breaking the alignment

- A recipe
 - Just don't let the score go below 0
 - Start new alignment when it happens
 - Where is the result in the matrix?

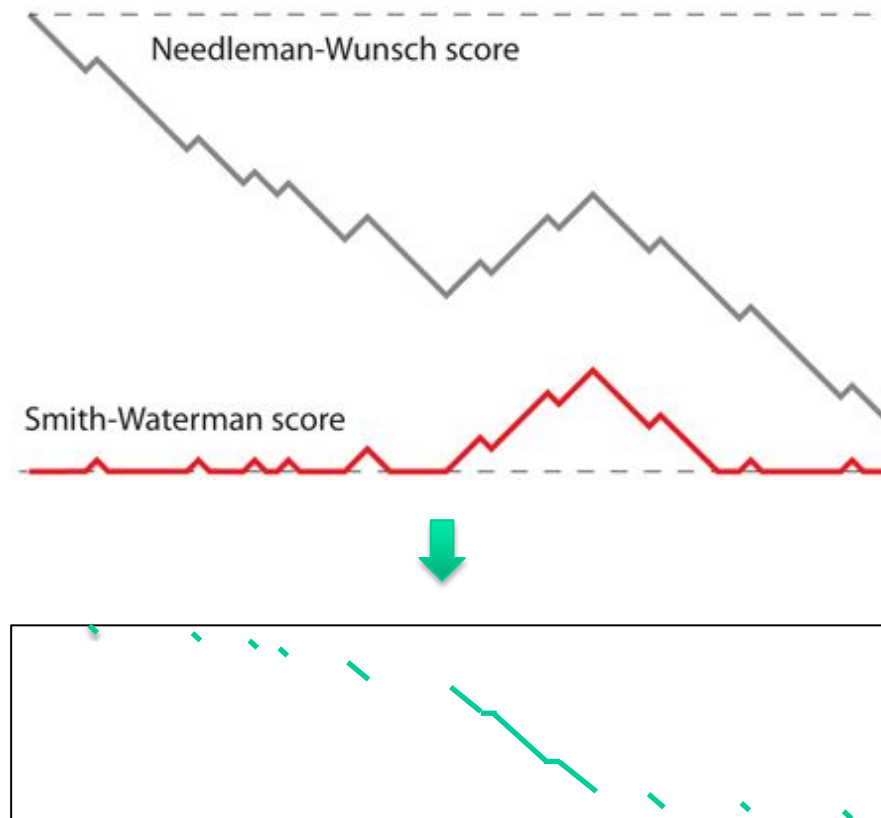
Before:



After:

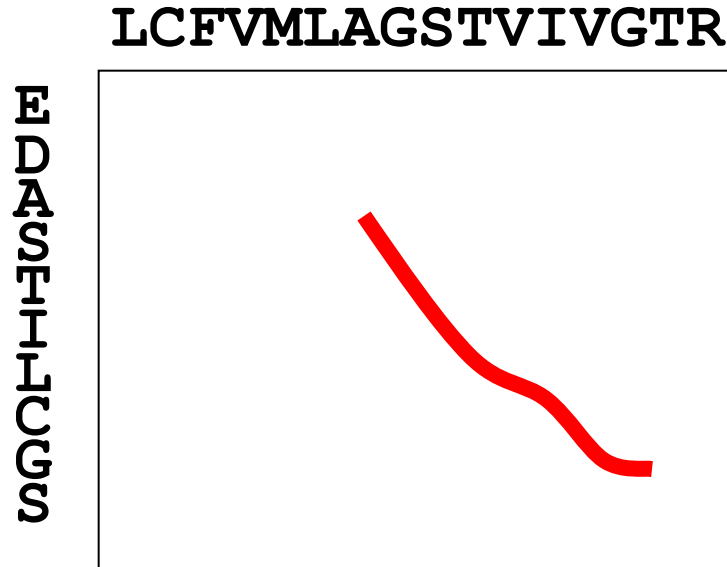


Local alignment – breaking the alignment



Local dynamic programming

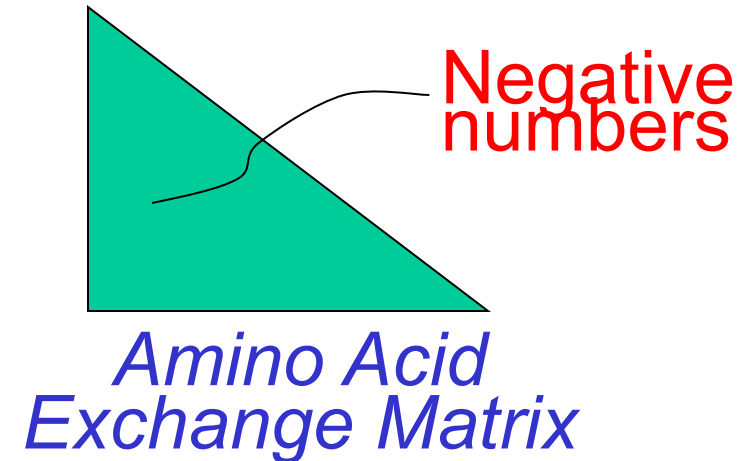
(Smith & Waterman, 1981)



Search matrix



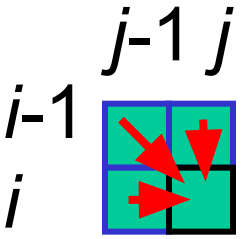
AGSTVIVG
A-STILCG



*Gap penalties
(e.g., open,
extension)*

Local dynamic programming (Smith and Waterman, 1981)

basic algorithm


$$H(i,j) = \text{Max} \left\{ \begin{array}{ll} H(i-1,j-1) + S(i,j) & \text{diagonal} \\ H(i-1, j) - g & \text{vertical} \\ H(i, j-1) - g & \text{horizontal} \\ 0 & \end{array} \right.$$

Smith, T. F. & Waterman, M. S. (1981), "Identification of common molecular subsequences", *J. Mol. Biol.*, vol. 147, pp 195-197.

Local alignment – the equation

$$M[i, j] = \max \begin{cases} M[i-1, j-1] + \text{score}(X[i], Y[j]) \\ M[i, j-1] - g \\ M[i-1, j] - g \\ 0 \end{cases}$$

Great contribution to science!

- Init the matrix with 0's
- Fill in the search matrix (forward step)
- Read the maximal value **from anywhere** in the matrix
- Find the result by performing trace-back

	-	s	e	q	u	e
-						
s						
e						
q						

Example:

local alignment of two sequences

- Align two DNA sequences:
 - GAGTGA
 - GAGGCGA (note the length difference)
 - Parameters of the algorithm:
 - **Match:** $\text{score}(A,A) = 1$
 - **Mismatch:** $\text{score}(A,T) = -1$
 - **Gap:** $g = -2$
- $$M[i, j] = \max \begin{cases} M[i-1, j-1] \pm 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \\ 0 \end{cases}$$

The algorithm. Step 1: init

- Create the matrix
- Initiation
 - No beginning row/column
 - Just apply the equation...

$$M[i, j] = \max \begin{cases} M[i-1, j-1] + 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \\ 0 \end{cases}$$

	$j \rightarrow$	1	2	3	4	5	6
$i \downarrow$		G	A	G	T	G	A
1	G						
2	A						
3	G						
4	G						
5	C						
6	G						
7	A						

The algorithm. Step 2: fill in

- Perform the forward step...

$$M[i, j] = \max \begin{cases} M[i-1, j-1] \pm 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \\ 0 \end{cases}$$

	$j \rightarrow$	1	2	3	4	5	6
$i \downarrow$		G	A	G	T	G	A
1	G	1	0	1	0	1	0
2	A	0	2	0	0	0	2
3	G	1	0	3	1	1	0
4	G	1	0	1	2		
5	C						
6	G						
7	A						

Bold '0's represent cells would have scored negative otherwise (SW's 0)

The algorithm. Step 2: fill in

- Perform the forward step...

$$M[i, j] = \max \begin{cases} M[i-1, j-1] \pm 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \\ 0 \end{cases}$$

	$j \rightarrow$	1	2	3	4	5	6
$i \downarrow$		G	A	G	T	G	A
1	G	1	0	1	0	1	0
2	A	0	2	0	0	0	2
3	G	1	0	3	1	1	0
4	G	1	0	1	2	2	0
5	C	0	0	0	0	1	1
6	G	1	0	1	0	1	0
7	A	0	2	0	0	0	2

Bold '0's represent cells would have scored negative otherwise (SW's 0)

The algorithm. Step 2: fill in

- We're done
- Find the highest cell anywhere in the matrix

$$M[i, j] = \max \begin{cases} M[i-1, j-1] \pm 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \\ 0 \end{cases}$$

	$j \rightarrow$	1	2	3	4	5	6
$i \downarrow$		G	A	G	T	G	A
1	G	1	0	1	0	1	0
2	A	0	2	0	0	0	2
3	G	1	0	3	1	1	0
4	G	1	0	1	2	2	0
5	C	0	0	0	0	1	1
6	G	1	0	1	0	1	0
7	A	0	2	0	0	0	2

Bold '0's represent cells would have scored negative otherwise (SW's 0)

The algorithm. Step 3: trace back

- Reconstruct path leading to highest scoring cell
- Trace back until zero (not including the zero cell) or start of sequence: alignment path can begin and terminate anywhere in matrix
- Alignment: GAG
GAG

$$M[i, j] = \max \begin{cases} M[i-1, j-1] \pm 1 \\ M[i, j-1] - 2 \\ M[i-1, j] - 2 \\ 0 \end{cases}$$

	$j \rightarrow$	1	2	3	4	5	6
$i \downarrow$		G	A	G	T	G	A
1	G	1	0	1	0	1	0
2	A	0	2	0	0	0	2
3	G	1	0	3	1	1	0
4	G	1	0	1	2	2	0
5	C	0	0	0	0	1	1
6	G	1	0	1	0	1	0
7	A	0	2	0	0	0	2

Bold '0's represent cells would have scored negative otherwise (SW's 0)

Summary – types of alignment

1. **Global**
e.g Needleman-Wunsch
algorithm
2. **Semi-global**
3. **Local**
e.g. Smith-Waterman
algorithm
4. **Multiple local alignments**
aka Waterman-Eggert
algorithm

