Pairwise sequence alignment

Outline

- DNA Sequence Comparison: First Success Stories
- Strings
- Dot Matrix Methods
- Sequence Alignment – Edit Distance
- Scoring functions and matrices
- Global-, local-, repeat- and overlap alignment of two sequences using dynamic programming

DNA Sequence Comparison: First Success Story

- Finding sequence similarities with genes of known function is a common approach to infer a newly sequenced gene's function

- In 1984 Russell Doolittle and colleagues found similarities between cancer-causing gene and normal growth factor (PDGF) gene

Cystic Fibrosis

- **Cystic fibrosis** (CF) is a chronic and frequently fatal genetic disease of the body's mucus glands (abnormally high level of mucus in glands). CF primarily affects the respiratory systems in children.
- Mucus is a slimy material that coats many epithelial surfaces and is secreted into fluids such as saliva
- In early 1980s biologists hypothesized that CF is an autosomal recessive disorder caused by mutations in a gene that remained unknown till 1989

Shorter byway

Finding Similarities between the Cystic Fibrosis Gene and ATP binding proteins

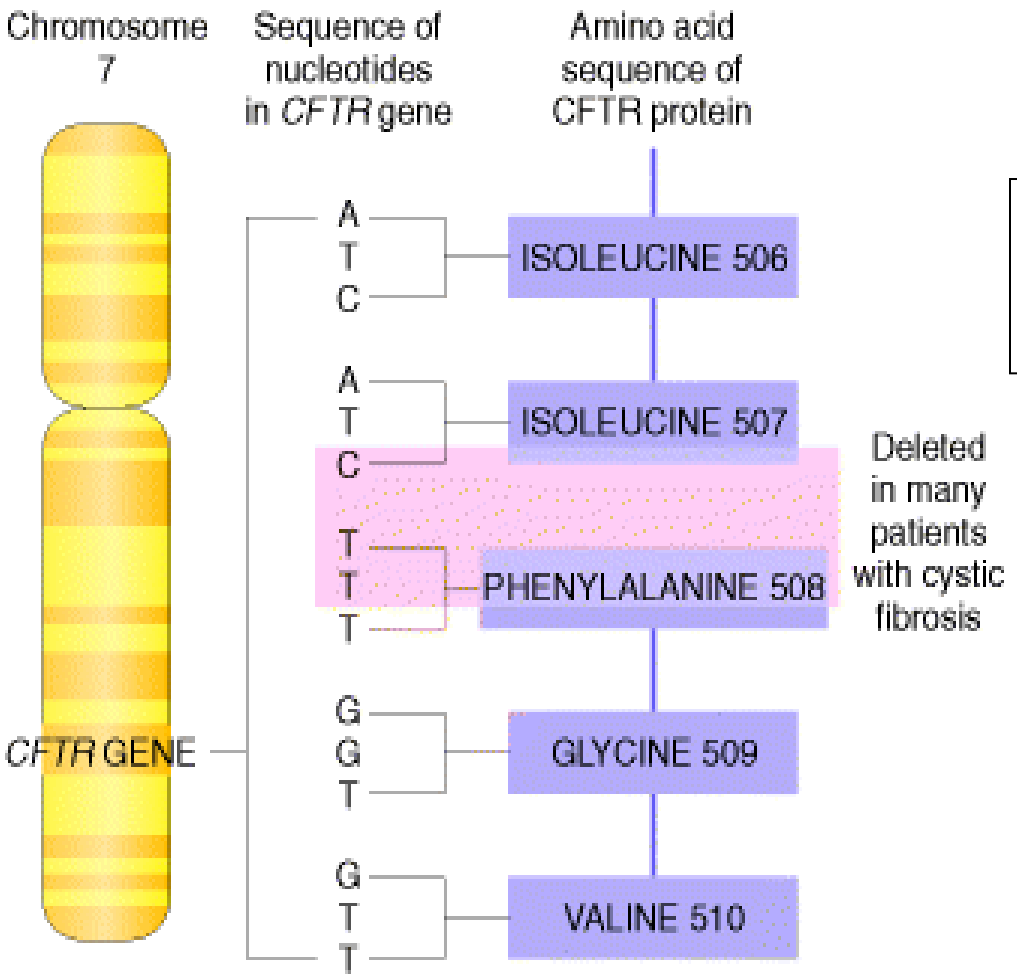
- ATP binding proteins are present on cell membrane and act as transport channel
- In 1989 biologists found similarity between the cystic fibrosis gene and ATP binding proteins
- A plausible function for cystic fibrosis gene, given the fact that CF involves sweet secretion with abnormally high sodium level

Cystic Fibrosis: Mutation Analysis

If a high % of cystic fibrosis (CF) patients have a certain mutation in the gene and the normal patients don't, then that could be an indicator of a mutation that is related to CF

A certain mutation was found in 70% of CF patients, convincing evidence that it is a predominant genetic diagnostics marker for CF

Cystic Fibrosis and CFTR Gene

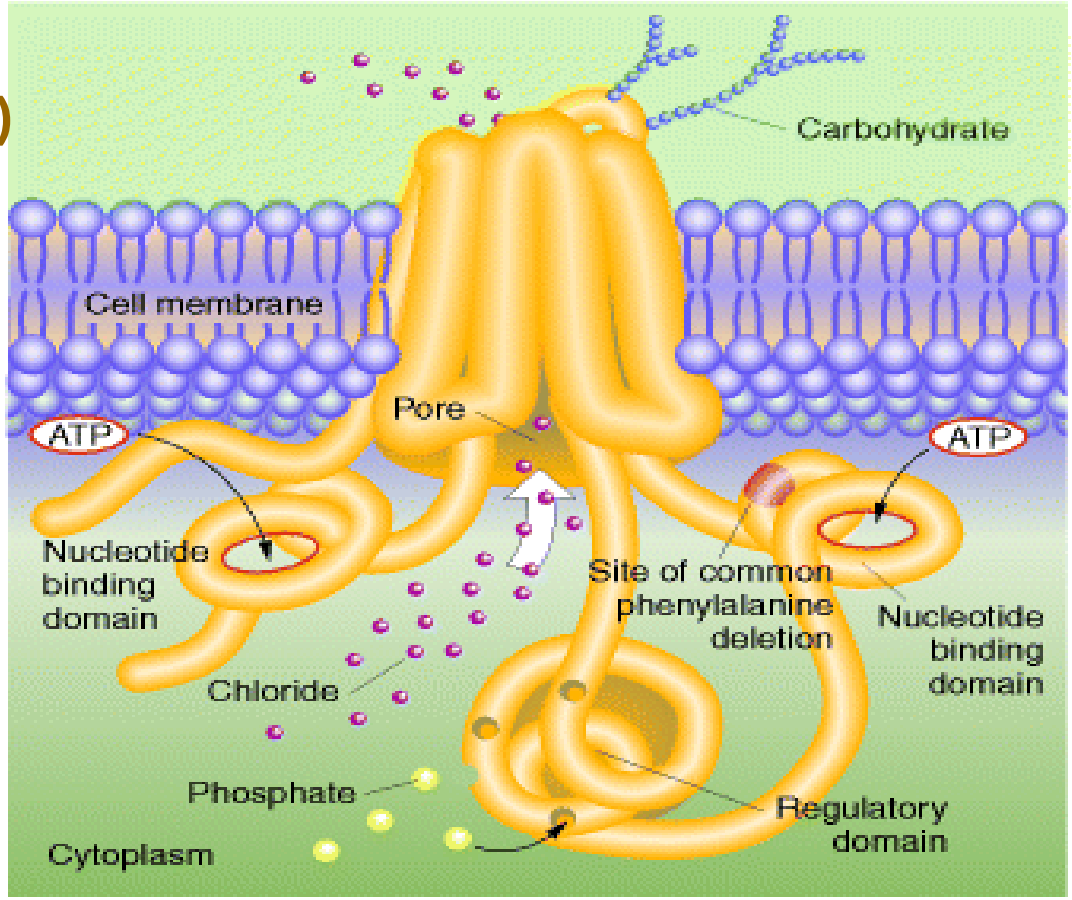


CFTR = Cystic Fibrosis Transmembrane conductance Regulator

Shorter byway

Cystic Fibrosis and the CFTR Protein

- **CFTR (Cystic Fibrosis Transmembrane conductance Regulator)** protein is acting in the cell membrane of epithelial cells that secrete mucus
- These cells line the airways of the nose, lungs, the stomach wall, etc.



Mechanism of Cystic Fibrosis

- The **CFTR protein** (1480 amino acids) regulates a chloride ion channel
- Adjusts the “wateriness” of fluids secreted by the cell
- Those with cystic fibrosis are missing one single amino acid in their CFTR
- Mucus ends up being too thick, affecting many organs

Bring in the Bioinformaticians

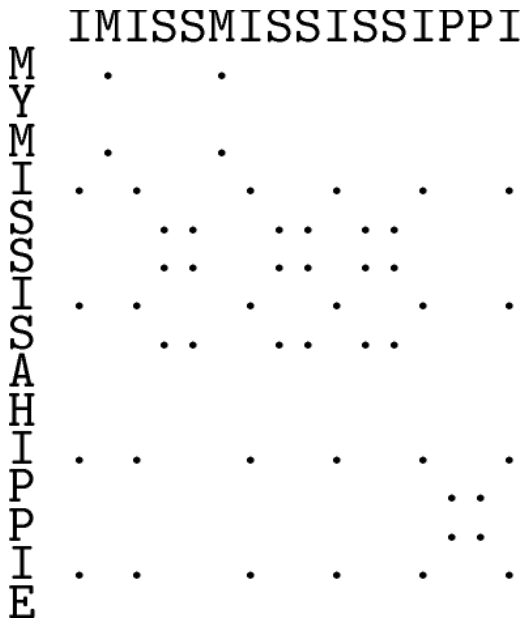
- Gene similarities between two genes with known and unknown function alert biologists to some possibilities
- Computing a similarity score between two genes tells how likely it is that they have similar functions

Strings

- **Definition:** An *alphabet* Σ is a finite nonempty set. The elements of an alphabet are called *symbols* or letters. A *string* S over an alphabet Σ is a (finite) concatenation of symbols from Σ. The *length* of a string S is the number of symbols in S, denoted by $|S|$. The set of strings of length n over Σ is denoted by Σ^n.
- For DNA-sequences $\Sigma = \{ A,G,C,T \}$.
- Known notions: a concatenation, substring, prefix, suffix
- Let Σ be an alphabet and let $S = s_1 \ldots s_n$ with $s_i \in \Sigma$. For all $i, j \in \{1, \ldots , n\}$, $i < j$, we denote the substring $s_i \ldots s_j$ by $S[i, j]$.

Dot matrix sequence comparison

- An ($n \times m$) matrix relating two sequences of length n and m respectively is produced: by placing a dot at each cell for which the corresponding symbols match. Here is an example for the two sequences `IMISSMISSISSIPPI` and `MYMISSISAHIPPIE`:

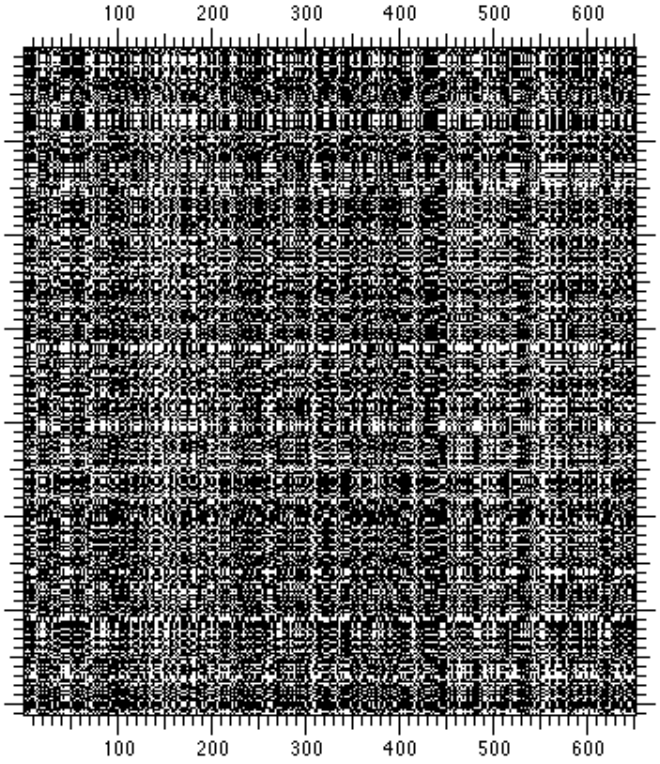


Dot plot

- **Definition:** Let $S = s_1 s_2 \ldots s_n$ and $T = t_1 \ldots t_m$ be two strings of length n and m respectively. Let M be an $n \times m$ matrix. Then M is a dot plot if for i, j, $1 \leq i \leq n$, $1 \leq j \leq m$: $M[i, j] = 1$ for $s_i = t_j$ and $M[i, j] = 0$ else.
- *Note*: The longest common substring within the two strings S and T is then the longest matrix subdiagonal containing only 1's. However, rather than drawing the letter 1 we draw a dot, and instead of a 0 we leave the cell blank. Some of the properties of a dot plot are
  - the visualization is easy to understand
  - it is easy to find common substrings, they appear as contiguous dots along a diagonal
  - it is easy to find reversed substrings (see assignment)
  - it is easy to discover displacements
  - it is easy to find repeats

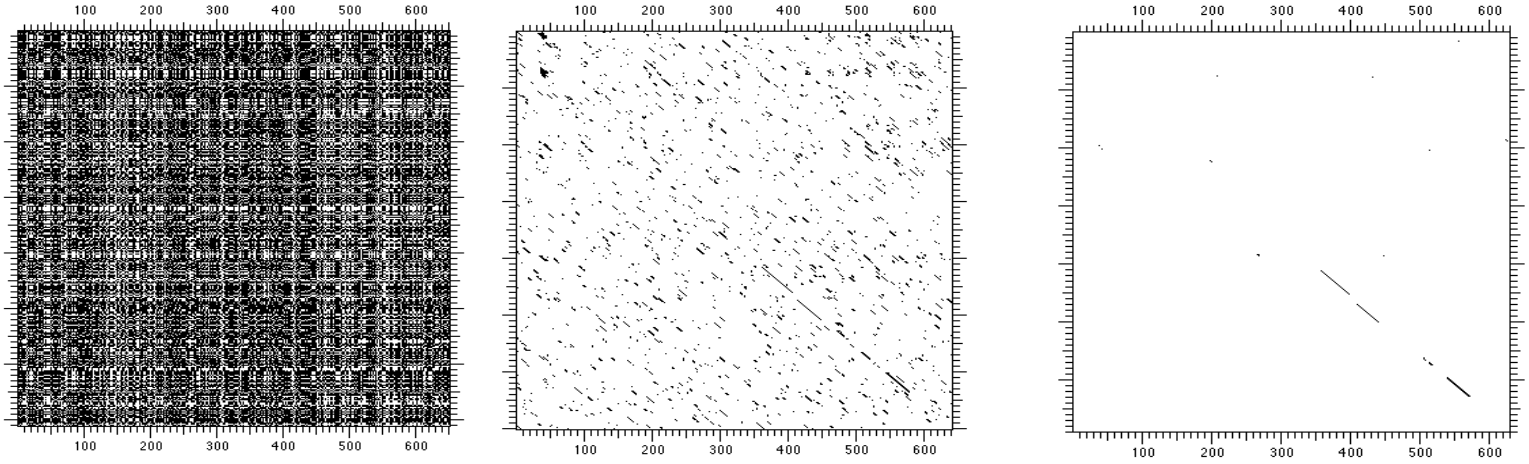
Dot matrix

- Example: DNA sequences which encode the Bacteriophage lambda and Bacteriophage P22 repressor proteins:



A window size and a stringency

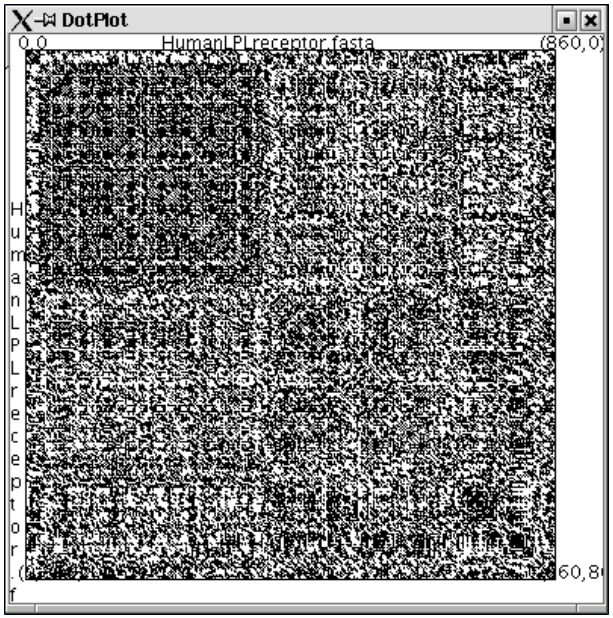
- Real dot plots of biological sequences will contain a lot of dots, many of which are considered as noise.
- To reduce the noise, a window size *w* and a stringency *s* are used and a dot is only drawn at point (*x*, *y*) if in the next *w* positions at least *s* characters are equal. For the example above:



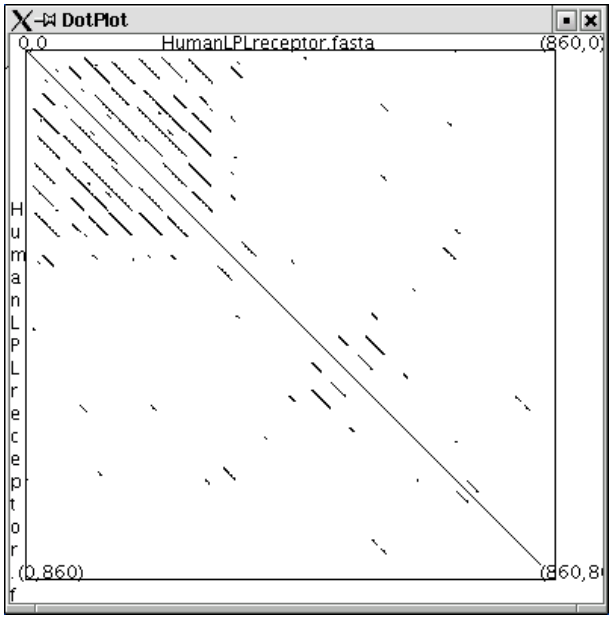
w = 1, *s* = 1 *w* = 11, *s* = 7 *w* = 23, *s* = 15

Dot matrix repeat detection

- Dot matrix analysis of human LDL receptor against itself (protein sequence):



w = 1, s = 1



w =?, s =?

Exercise: Determine which *w* and *s* are best to use in this case, and interpret the result.

Sequence alignment

- Procedure of comparing sequences by searching for a series of individual characters or character patterns that are in the same order in both sequences.
  - two (pair-wise alignment) or
  - more (multiple alignment)
- Two sequences are aligned by writing them in two rows. Identical or similar characters are placed in the same column, whereas non-identical characters are either placed in the same column as a mismatch or are opposite a gap in the other sequence.

Two strings:	→	Alignment:
IMISSMISSISSIPPI		I-MISSMISSISIPPI-
MYMISSISAHIPPIE		MYMISS-ISAH-IPPIE

String alignment

- Given two strings *X* and *Y* . An alignment *A* of *X* and *Y* is obtained by inserting dashes (‘-’) so that both resulting strings *X’* and *Y’* of equal length can be written one above the other in such a way that each character in the one string is opposite to a unique character in the other string.
- Usually, we require that no two dashes are aligned in this way.
- Example:

```
X =   Y E - S T E R D A Y
Y =   - E A S T E R S - -
```

- We need a scoring system.

Distance

- **Definition**: A set X of elements $x, y, \ldots \in X$ is called a metric space if for each pair $x, y \in X$ there exists a real number $d(x, y)$ with:
  1. $d(x, y) \geq 0, \quad d(x, y) = 0 \Leftrightarrow x = y$
  2. $d(x, y) = d(y, x)$
  3. $d(x, y) \leq d(x, z) + d(z, y) \quad \forall z \in X$

  $d(x, y)$ is called the distance of x and y.

Minkowski metric

- **Definition**: Let $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ be two elements of an n-dimensional space X. Then

$$d_M(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

- is called the Minkowski distance with parameter p.
- Note: For $p = 1$ the distance is also called the Manhattan (or city-block) distance, for $p = 2$ we have the well-known Euklidean distance.

Hamming metric

- **Definition**: Let $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ be two strings of length n over an alphabet S. Then

$$d_H(X, Y) = |\{ i \mid i \in \{1, \ldots, n\}, x_i \neq y_i\}|$$

  is called the Hamming distance.

- **Example:**

  The Hamming distance of the two sequences

  X = A T A T A T A T

  Y = T A T A T A T A

  is equal to $d_H(X, Y) =$ __.

Levenshtein or edit distance

- number of editing operations needed to transform one string into the other
- **Definition**: The Levenshtein distance or edit distance d_L between two strings X and Y is the minimum number of edit operations of type

  {Replacement, Insertion, or Deletion }

  that one needs to transform string X into string Y :

  $$d_L(X, Y) = \min\{R(X, Y) + I(X, Y) + D(X, Y)\}.$$

- the two strings need not be of equal length
- Using M for match, an edit transcript is a string over the alphabet {I,D,R,M} that describes a transformation of X to Y.

Edit distance

- **Example**:

  Given two strings

  X= YESTERDAY

  Y= EASTERS

  The edit distance is equal to 5, which can be easily seen from the minimum edit transcript:

```
    =  D M I M M M M R D D
X   =  Y E   S T E R D A Y
Y   =    E A S T E R S
```

- As we see from this example, edit transcripts and alignments are mathematically equivalent ways of describing a relationship between two strings.
- However, an edit transcript implies a set of putative mutational events, whereas an alignment presents a static picture of the relationship.

Calculation of edit distance

- Given two strings $X = x_1...x_n$ and $Y = y_1...y_m$. We want to compute the edit distance $D_L(X, Y)$ between X and Y .
- Let $D(i, j)$ denote the edit distance of the two prefixes $x_1...x_i$ and $y_1...y_j$.
- Clearly, it is $D_L(X, Y) = D(n,m)$, and we want to obtain $D(n,m)$ by computing $D(i, j)$ for all i, j with $0 \leq i \leq n$ and $0 \leq j \leq m$.
- This is the standard dynamic programming approach
  - the recurrence relation,
  - the tabular computation, and
  - the traceback.

Dynamic programming for computing edit distance

- the recurrence relation:

$$D(i,j) = \min \begin{cases} D(i, j-1) + 1 \\ D(i-1, j) + 1 \\ D(i-1, j-1) + t(i, j) \end{cases}$$

where

$$t(i, j) = \begin{cases} 0 & \text{if } x_i = y_j , \\ 1 & \text{else.} \end{cases}$$

Base conditions:

- We set $D(i, 0) = i$ for all $0 \le i \le n$. This corresponds to an alignment in which the first i characters of X are aligned to the left of the first character of Y .
- We set $D(0, j) = j$ for all $1 \le j \le m$. This corresponds to an alignment in which the first j characters of Y occur to the left of the first character of X.

Dynamic programming for computing edit distance

- The recursion is computed in tabular form↘:

	0	y_1	$\cdots$	y_{j-1}	y_j	$\cdots$	y_m
0	0	1	$\cdots$	$j-1$	j	$\cdots$	m
x_1	1						
$\vdots$							
x_{i-1}	$i-1$			$D(i-1,j-1)$	$D(i-1,j)$		
				$\searrow$	$\downarrow$		
x_i	i			$D(i,j-1)$	$\rightarrow$ $D(i,j)$		
$\vdots$							
x_n	n						$D(n,m)$

Dynamic programming for computing edit distance – traceback

- While computing the values $D(i, j)$ one also saves (in an independent matrix) which of the three terms in the recurrence relation was minimal and used for $D(i, j)$. Then from the final $D(n,m)$ the edit transcript (and therefore the alignment) can be achieved by backtracking or traceback of the entries in the second matrix.

D	0	G	A	T	T	A	G
0							
A							
T							
T							
A							
C							

Weighted edit distance

- We can **generalize** the edit distance by
  - weighting each of the edit operations I, D and R by a number
    - The operation weighted edit distance between two sequences X and Y is the minimum sum of weights of any edit transcript from X to Y .
  - making the score of an edit operation depend on the two characters involved
    - This gives rise to the **alphabet weighted edit distance**.
- **Definition:** Let Σ be a finite alphabet and d a metric on Σ. Let $\varepsilon \in \Sigma$ denote the gap symbol. Then for two strings X and Y of length n and m respectively

$$D_L(i, j) := \min \{D_L(i, j-1) + d(\varepsilon, y_j), D_L(i-1, j) + d(x_i, \varepsilon), D_L(i-1, j-1) + d(x_i, y_j)\},$$

and $D_L(n,m)$ is the alphabet weighted Levenshtein distance of X and Y .

Global distance alignment

- When comparing two biological sequences, we want to determine whether and how they diverged from a common ancestor by a process of mutation and selection.
  - basic mutational processes are substitutions, insertions and deletions. The latter two give rise to gaps.
- The total score assigned to an alignment is the sum of terms for each aligned pair of residues, plus terms for each gap.
- **We assume:** mutations at different sites occur independently of each other.
  - This is often reasonable for DNA and proteins, but not for structural RNA, where base pairing introduces very important long-range dependences.

Global distance alignment

- Let $X = x_1...x_n$ and $Y = y_1...y_m$ be two sequences over an alphabet Σ. Let A be a global alignment of length l_A of X and Y . Let ε be the gap symbol. Let $X' = x'_1...x'_{lA}$ and $Y' = y'_1...y'_{lA}$ denote the two strings obtained after inserting dashes (for the gap symbol ε). Let $d(a, b)$, $a, b \in \Sigma \cup \{\varepsilon\}$ be a distance on the alphabet. This represents the cost of a mutation of a into b or the cost of inserting or deleting a letter. Then define

$$D(X,Y) = \min_A \left(\sum_{i=1}^{l_A} d(x'_i, y'_i) \right)$$

- The alignment for which the total score is minimal is called optimal.

String distance → string similarity

- We have seen how to express string relatedness using the Levenshtein or edit distance. In biology, we are usually interested in **similarity** rather than distance, as we will see further below.
- A similarity score matrix $S : \Sigma \cup \{\varepsilon\} \times \Sigma \cup \{\varepsilon\} \rightarrow R$ assigns a similarity score to each pair of characters
- For a given alignment $A = (X', Y')$ of X and Y of length l_A, the value of A is defined as

$$\sum_{i=1}^{l_A} s\left(x'_i, y'_i\right)$$

String similarity – example

- For $\Sigma = \{A, B, L, -\}$ consider the following similarity score matrix S:

	A	B	L	−
A	3	1	−1	−2
B		2	0	−3
L			1	2
−				0

- matches of symbols are rewarded, mismatches and gaps penalized.

X= B L A - B L A

Y= A L A B B L -

1 +1 +3 -3 +2 +1 -2 = 3

String similarity – example

- The similarity of two sequences *X* and *Y* is the value of any alignment *A* of *X* and *Y* that **maximizes** the alignment value. Such an alignment is called optimal.

Example 1:

- Alignment between very similar human alpha- and beta globins:

```
HBA_HUMAN  GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKL
             G+  +VK+HGKKV   A++++++AH+D++  +++++LS+LH   KL
HBB_HUMAN  GNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKL
```

- there are many positions at which the two corresponding residues are identical. Many others are functionally conserved, e.g. the D-E pairs, both negatively charged amino acids – marked by + sign.

String similarity – example

Example 2:

- Plausible alignment to leg haemoglobin from yellow lupin:

```
HBA_HUMAN   GSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL
            ++ ++++H+ KV    + +A   ++            +L+ L+++H+ K
LGB2_LUPLU NNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG
```

- also a biologically meaningful alignment, as it is known that the two proteins are evolutionarily related, have the same 3D structure and both have the same function. However, there are many fewer identities and gaps have been introduced in the sequences.

Simple Scoring

- When
  - mismatches are penalized by $-\mu$,
  - indels are penalized by $-\sigma$, and
  - matches are rewarded with $+1$,

  the resulting score is:

$$\#matches - \mu(\#mismatches) - \sigma(\#indels)$$

The Global Alignment Problem

Find the best alignment between two strings under a given scoring schema

Input : Strings x and y and a scoring schema
Output : Alignment of maximum score

$\uparrow \rightarrow = -\sigma$

$\searrow$ $= 1$ if match
$= -\mu$ if mismatch

μ : mismatch penalty
σ : indel penalty

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + 1 & \text{if } x_i = y_j \\ F(i-1,j-1) - \mu & \text{if } x_i \neq y_j \\ F(i-1,j) - \sigma \\ F(i,j-1) - \sigma \end{cases}$$

Scoring Matrices

- In general, for DNA we consider a (4+1) x (4+1) **scoring matrix** *s*
- In the case of an amino acid sequence alignment, the scoring matrix would be a (20+1)x(20+1) size. The addition of 1 is to include the score for comparison of a gap character "**-**"
- This will simplify the algorithm as follows:

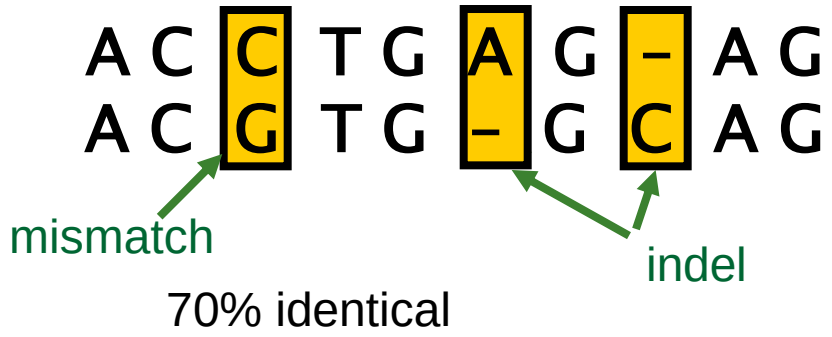
$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) + s(x_i, \text{'-'}) \\ F(i,j-1) + s(\text{'-'}, y_j) \end{cases}$$

General scoring model

- Computation of an alignment critically depend on the choice of parameters. Generally no existing scoring model can be applied to all situations.
  - When evolutionary relationships between the sequences are reconstructed – scoring matrices based on mutation rates are usually applied – computed from sequences with high percent identity.
  - When protein domains are compared – then the scoring matrices should be based on composition of domains and their substitution frequency – computed from sequences with high conservation.

Percent Sequence Identity

- The extent to which two nucleotide or amino acid sequences are invariant



Conservation

- Amino acid changes that tend to preserve the physico-chemical properties of the original residue
  - Polar to polar
    - aspartate → glutamate
  - Nonpolar to nonpolar
    - alanine → valine
  - Similarly behaving residues
    - leucine to isoleucine

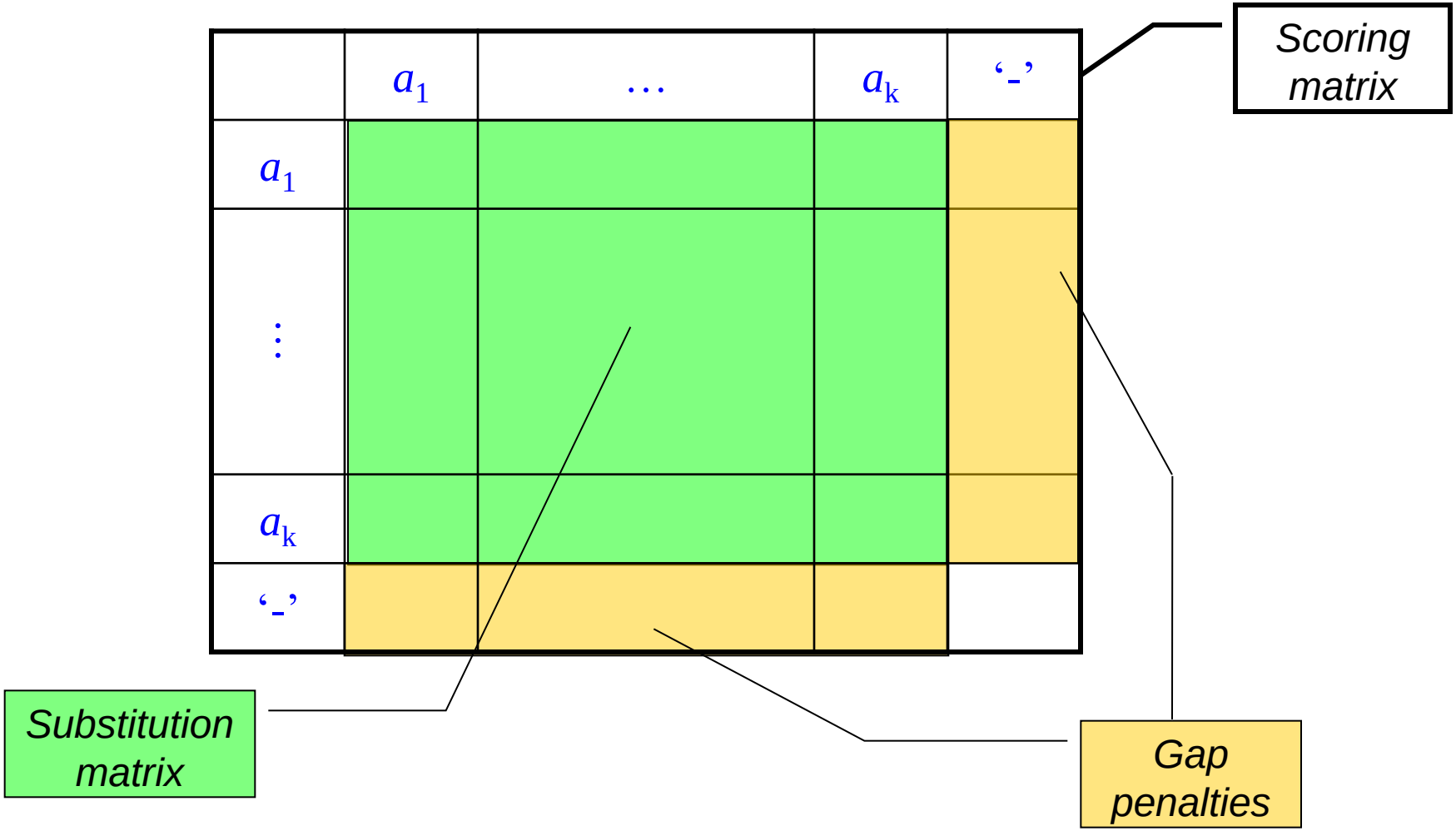
Scoring Matrix: Example

	A	R	N	K
A	5	-2	-1	-1
R	-	7	-1	3
N	-	-	7	0
K	-	-	-	6

- Notice that although R and K are different amino acids, they have a positive score.
- Why? They are both positively charged amino acids→ will not greatly change function of protein.

```
A   K   R   A   N   R
K   A   A   A   N   K
(-1) + (-1) + (-2) +  5  +  7  +  3  =  11
```

General scoring model



Scoring matrices

- DNA substitution matrices
  - DNA is less conserved than protein sequences
  - Less effective to compare coding regions at nucleotide level
- Amino acid substitution matrices
  - PAM
  - BLOSUM

Substitution matrices

- To be able to score an alignment, we need to determine score terms for each aligned residue pair.

- **Definition:** A substitution matrix S over an alphabet $\Sigma = \{a_1, \ldots, a_k\}$ has $k \times k$ entries, where each entry (i, j) assigns a score for a substitution of the letter a_i by the letter a_j in an alignment.

Substitution matrices

- **Basic idea:** Follow scheme of statistical hypothesis testing.

$$f\begin{pmatrix} a \\ b \end{pmatrix} = \frac{f(a,b)}{f(a) \cdot f(b)}$$

- Frequencies of the letters $f(a)$ as well as substitution frequencies $f(a, b)$ stem from a representative data set.

Null hypothesis / Random model

- Given a pair of aligned sequences (without gaps), the null hypothesis states that the two sequences are unrelated (not homologous). The alignment is then random with a probability described by the model *R*. The unrelated or random model *R* assumes that in each aligned pairs of residues the two residues occur independently of each other. Then the probability of the two sequences is:

$$P(X,Y \mid R) = P(X \mid R) \cdot P(Y \mid R) = \prod_i p_{x_i} \prod_i p_{y_i}$$

Match model

- In the match model *M*, describing the alternative hypothesis, aligned pairs of residues occur with a joint probability p_{ab}, which is the probability that *a* and *b* have each evolved from some unknown original residue *c* as their common ancestor. Thus, the probability for the whole alignment is:

$$P(X,Y \mid M) = \prod_i p_{x_i y_i}$$

Odds ratio
(pomer šancí)

- The ratio of the two gives a measure of the relative likelihood that the sequences are related (model M) as opposed to being unrelated (model R). This ratio is called odds ratio:

$$\frac{P(X,Y \mid M)}{P(X,Y \mid R)} = \frac{\prod_i p_{x_i y_i}}{\prod_i p_{x_i} \prod_i p_{y_i}} = \prod_i \frac{p_{x_i y_i}}{p_{x_i} p_{y_i}}$$

But this is not additive scoring!

Log-odds ratio

- To obtain an additive scoring scheme, we take the logarithm (base 2 is usually chosen) to get the log-odds ratio:

$$\log\left(\frac{P(X,Y\mid M)}{P(X,Y\mid R)}\right)=\log\left(\prod_i \frac{p_{x_i y_i}}{p_{x_i}p_{y_i}}\right)=\sum_i s(x_i,y_i)$$

where

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

- For amino-acid alignments, commonly used matrices are
  - the PAM and
  - BLOSUM matrices.

PAM matrices

Definition [Dayhoff et. al.]: One Point Accepted Mutation (1 PAM) is defined as an expected number of substitutions per site of 0.01. A 1 PAM substitution matrix is thus derived from any evolutionary model by setting the row sum of off-diagonal terms to 0.01 and adjusting the diagonal terms to keep the row sum equal to 1.

- After 100 PAMs of evolution, not every residue will have changed
  - some residues may have mutated several times
  - some residues may have returned to their original state
  - some residues may not changed at all

Jukes-Cantor Model (for DNA)

- The basic assumption is equality of substitution frequency for any nucleotide at any site. Thus, changing a nucleotide to each of the three remaining nucleotides has probability α per time unit. The rate of nucleotide substitution per site per time unit is then $r = 3\alpha$.
- PAM 1 matrix under a Jukes-Cantor model of sequence evolution is

$$\begin{pmatrix} 1-3\alpha & \alpha & \alpha & \alpha \\ \alpha & 1-3\alpha & \alpha & \alpha \\ \alpha & \alpha & 1-3\alpha & \alpha \\ \alpha & \alpha & \alpha & 1-3\alpha \end{pmatrix}$$

Jukes-Cantor Model

$$\begin{pmatrix} 1-3\alpha & \alpha & \alpha & \alpha \\ \alpha & 1-3\alpha & \alpha & \alpha \\ \alpha & \alpha & 1-3\alpha & \alpha \\ \alpha & \alpha & \alpha & 1-3\alpha \end{pmatrix}$$

- We scale matrix entries such that the expected number of substitutions per site is $0.01 = 3\alpha$ and obtain a probability matrix:

$$\begin{pmatrix} 0.99 & 0.003 & 0.003 & 0.003 \\ 0.003 & 0.99 & 0.003 & 0.003 \\ 0.003 & 0.003 & 0.99 & 0.003 \\ 0.003 & 0.003 & 0.003 & 0.99 \end{pmatrix}$$

From probability into scoring matrix

- A scoring matrix is then obtained by computing the log-odds ratios:

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

with $p_A = p_C = p_G = p_T = 0.25$ and joint probabilities as given by the PAM probability matrix. This leads to the following substitution score matrix:

for $a = b$ $\quad s(a,b) = \log(0.99/0.25^2) \cong 3.9855$

for $a \neq b$ $\quad s(a,b) = \log(0.003/0.25^2) \cong -4.3808$

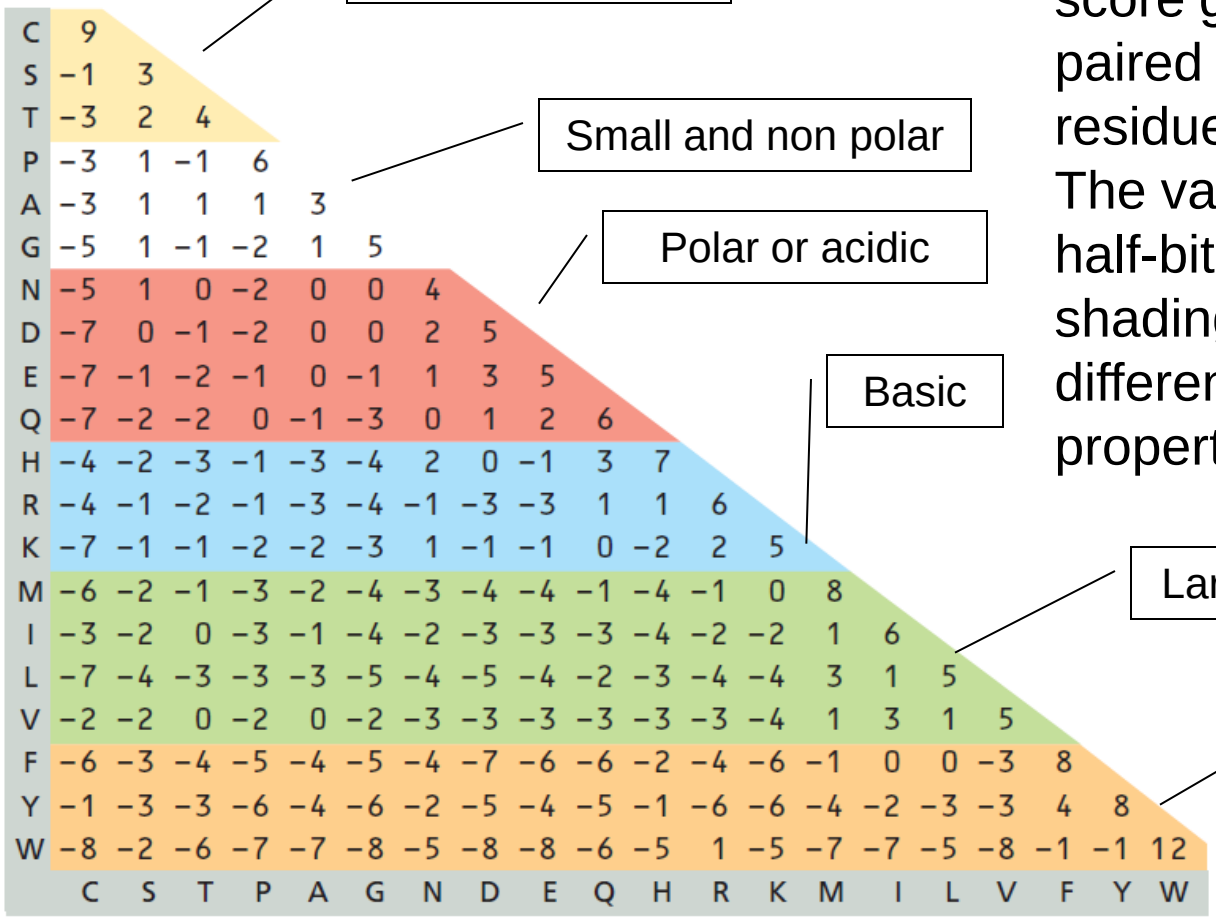
$$\begin{pmatrix} 398 & -438 & -438 & -438 \\ -438 & 398 & -438 & -438 \\ -438 & -438 & 398 & -438 \\ -438 & -438 & -438 & 398 \end{pmatrix}$$

PAM_X

- $PAM_x = PAM_1^x$
  - $PAM_{250} = PAM_1^{250}$
- PAM_{250} is a widely used scoring matrix for amino acids:

	Ala A	Arg R	Asn N	Asp D	Cys C	Gln Q	Glu E	Gly G	His H	Ile I	Leu L	Lys K	...
Ala A	13	6	9	9	5	8	9	12	6	8	6	7	...
Arg R	3	17	4	3	2	5	3	2	6	3	2	9	
Asn N	4	4	6	7	2	5	6	4	6	3	2	5	
Asp D	5	4	8	11	1	7	10	5	6	3	2	5	
Cys C	2	1	1	1	52	1	1	2	2	2	1	1	
Gln Q	3	5	5	6	1	10	7	3	7	2	3	5	
...													
Trp W	0	2	0	0	0	0	0	0	1	0	1	0	
Tyr Y	1	1	2	1	3	1	1	1	3	2	2	1	
Val V	7	4	4	4	4	4	4	4	5	4	15	10	

PAM_{120}



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

Each cell represents the score given to a residue paired with another residue (row × column). The values are given in half-bits. The colored shading indicates different physicochemical properties of the residues.

BLOCKS and BLOSUM matrices

- The BLOSUM matrices were derived from the database BLOCKS [1]. Blocks are multiply aligned **ungapped** segments corresponding to the most highly conserved regions of proteins.
- [1] Henikoff, S and Henikoff, JG (1992) Amino acid substitution matrices from protein blocks. Proc Natl Acad Sci U S A. 89(22):10915-9. BLOCKS database server: http://blocks.fhcrc.org/
- For the scoring matrices of the BLOSUM (=BLOcks SUbstitution Matrix) family all blocks of the database are evaluated columnwise. For each possible pair of amino acids the frequency $f(a_i, a_j)$ of common pairs (a_i, a_j) in all columns is determined.

BLOCKS and BLOSUM matrices

Block IPB001523

```
ID   Paired_box; BLOCK
AC   IPB001523; distance from previous block=(-26,400)
DE   Paired box protein, N-terminal
BL   ACI;  width=39; seqs=345; 99.5%=2041; strength=1195
GSBN_DROME|P09083  (  45) IVEMAASGVRPCVISRQLRVSHGCVSKILNRYQETGSIR   1
GSB_DROME|P09082   (  44) IVEMAAAGVRPCVISRQLRVSHGCVSKILNRFQETGSIR   2
HMPR_DROME|P06601  (  52) IVEMAADGIRPCVISRQLRVSHGCVSKILNRYQETGSIR   2
PAX1_CHICK|P47236  (  28) IVELAQLGIRPCDISRQLRVSHGCVSKILARYNETGSIL   1
PAX1_HUMAN|P15863  (  29) IVELAQLGIRPCDISRQLRVSHGCVSKILARYNETGSIL   1
PAX1_MOUSE|P09084  (  29) IVELAQLGIRPCDISRQLRVSHGCVSKILARYNETGSIL   1
PAX2_BRARE|Q90268  (  44) IVELAHQGVRPCDISRQLRVSHGCVSKILGRYYETGSIK   1
PAX2_HUMAN|Q02962  (  41) IVELAHQGVRPCDISRQLRVSHGCVSKILGRYYETGSIK   1
PAX2_MOUSE|P32114  (  40) IVELAHQGVRPCDISRQLRVSHGCVSKILGRYYETGSIK   1
PAX3_HUMAN|P23760  (  59) IVEMAHHGIRPCVISRQLRVSHGCVSKILCRYQETGSIR   1
PAX3_MOUSE|P24610  (  59) IVEMAHHGIRPCVISRQLRVSHGCVSKILCRYQETGSIR   1
PAX4_HUMAN|O43316  (  30) IVRLAVSGMRPCDISRILKVSNGCVSKILGRYYRTGVLE   6
PAX4_MOUSE|P32115  (  30) IVQLAIRGMRPCDISRSLKVSNGCVSKILGRYYRTGVLE   7
PAX4_RAT|O88436    (  30) IVQLAIRGMRPCDISRSLKVSNGCVSKILGRYYRTGVLE   7
PAX5_HUMAN|Q02548  (  41) IVELAHQGVRPCDISRQLRVSHGCVSKILGRYYETGSIK   1
PAX5_MOUSE|Q02650  (  41) IVELAHQGVRPCDISRQLRVSHGCVSKILGRYYETGSIK   1
PAX6_BRARE|P26630  (  48) IVELAHSGARPCDISRILQVSNGCVSKILGRYYETGSIR   1
PAX6_COTJA|P47238  (  29) IVELAHSGARPCDISRILQVSNGCVSKILGRYYETGSIR   1
PAX6_DROME|O18381  (  81) IVELAHSGARPCDISRILQVSNGCVSKILGRYYETGSIR   1
PAX6_HUMAN|P26367  (  29) IVELAHSGARPCDISRILQVSNGCVSKILGRYYETGSIR   1
PAX6_MOUSE|P32117  (  29) IVELAHSGARPCDISRILQVSNGCVSKILGRYYETGSIR   1
PAX6_ORYLA|O73917  (  48) IVELAHSGARPCDISRILQVSNGCVSKILGRYYETGSIR   1
```

BLOCKS and BLOSUM matrices

- Altogether there are $\binom{n}{2}$ possible pairs that we can draw from this alignment. We now assume that the observed frequencies are equal to the frequencies in the population. Then

$$p_{ab} = \frac{observed}{\binom{n}{2}}$$

BLOCKS and BLOSUM matrices

Example

Seq 1 A
Seq 2 A
Seq 3 A
Seq 4 A
Seq 5 A
Seq 6 A
Seq 7 A
Seq 8 A
Seq 9 A
Seq 10 C

Altogether there are 45 possible pairs that we can draw from this alignment, of which 36 are AA and 9 are AC pairs. We now assume that the observed frequencies are equal to the frequencies in the population. Then p_{AA} = 36/45 and p_{CA} = 9/45, p_A = 9/10 and p_C = 1/10.

BLOCKS and BLOSUM matrices

Example

- Different levels of the BLOSUM matrix can be created by differentially weighting the degree of similarity between sequences. For example, a BLOSUM62 matrix is calculated from protein blocks such that if two sequences are more than 62% identical, then the contribution of these sequences is weighted to sum to one.
- In this way the contributions of multiple entries of closely related sequences is reduced.
- Standard values are BLOSUM50 up to BLOSUM80, with the commonly used **BLOSUM62** matrix. Note that lower BLOSUMx values correspond to longer evolutionary time, and are applicable for more distantly related sequences.

BLOSUM62

- BLOSUM62 is scaled so that its values are in half-bits, i.e. the log-odds were multiplied by $2 / \log_2 2$ and then rounded to the nearest integer value. E.g.

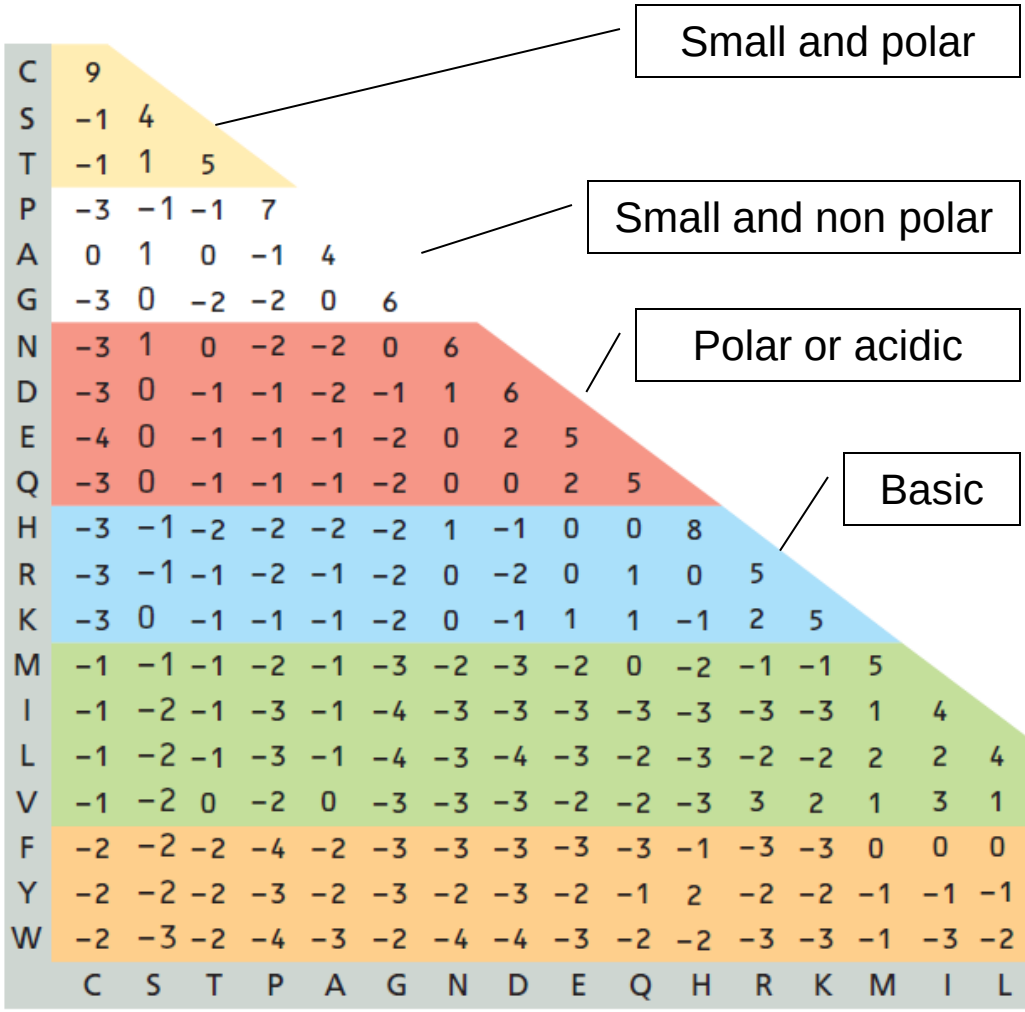
	A
A	4
R	-1

	p_a	p_b	p_{ab}	$p_{ab} / p_a p_b$	$2\log_2(p_{ab} / p_a p_b)$
A-A	0.074	0.074	0.0215	3.926	3.946
A-R	0.074	0.052	0.0023	0.598	-1.485

BLOSUM62

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

BLOSUM62



	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
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

Each cell represents the score given to a residue paired with another residue (row × column). The values are given in half-bits. The colored shading indicates different physicochemical properties of the residues.

Alignment algorithms

- Given a scoring scheme, we need to have an algorithm that computes the highest-scoring alignment of two sequences.
- As for the edit distance-based alignments we will discuss alignment algorithms based on *dynamic programming*. They are guaranteed to find the optimal scoring alignment.
- Note of caution: Optimal Pairwise alignment algorithms are of complexity $O(n \cdot m)$ – can be too slow and heuristics (such as BLAST, FASTA, MUMMER etc.) are then used that usually perform very well, but will miss the best alignment for some sequence pairs.
- Depending on the input data, there are a number of different variants of alignment that are considered, among them
  - *global alignment*,
  - *local alignment* and
  - *overlap alignment*.

Global alignment: Needleman-Wunsch algorithm

- Saul Needleman and Christian Wunsch (1970), improved by Peter Sellers (1974).
- **Idea**: Build up an optimal alignment using previous solutions for optimal alignments of smaller substrings.
- Given two sequences $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_m)$. We will compute a matrix

$$F : \{1, 2, \ldots, n\} \times \{1, 2, \ldots, m\} \rightarrow \mathbf{R}$$

- in which $F(i, j)$ equals the best score of the alignment of the two prefixes $(x_1, x_2, \ldots, x_i)$ and $(y_1, y_2, \ldots, y_j)$.
- This will be done recursively by setting $F(0, 0) = 0$ and then computing $F(i, j)$ from $F(i - 1, j - 1)$, $F(i - 1, j)$ and $F(i, j - 1)$:

Global alignment: Needleman-Wunsch algorithm

	0	y_1	$\cdots$	y_{j-1}		y_j	$\cdots$	y_m
0								
x_1								
$\vdots$								
x_{i-1}				$F(i-1, j-1)$		$F(i-1, j)$		
					$\searrow$	$\downarrow$		
x_i				$F(i, j-1)$	$\rightarrow$	$F(i, j)$		
$\vdots$								
x_n								$F(n, m)$

The recursion

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) & x_i \text{ aligns to } y_j \quad \begin{bmatrix} A\ G\ A\ x_i \\ A\ G\ G\ y_j \end{bmatrix} \\ F(i-1,j) - d & x_i \text{ aligns to a gap} \quad \begin{bmatrix} A\ G\ A\ x_i \\ A\ G\ G\ - \end{bmatrix} \\ F(i,j-1) - d & y_j \text{ aligns to a gap} \quad \begin{bmatrix} A\ G\ A\ - \\ A\ G\ G\ y_j \end{bmatrix} \end{cases}$$

- We set
  - $F(i, 0) = -id$ for $i=0,\ldots,n$
  - $F(0, j) = -jd$ for $j=0,\ldots,m$
- The final score will be in $F(n, m)$

Needleman-Wunsch algorithm

Input: two sequences X and Y
Output: optimal alignment and score

For $i := 1, 2, \ldots, n$ **do** $F(i, 0) := -i \cdot d$
For $j := 1, 2, \ldots, m$ **do** $F(0, j) := -j \cdot d$
For $i = 1, 2, \ldots, n$ **do**:
  For $j = 1, 2, \ldots, m$ **do**:

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

Set backtrace $T(i, j)$ to the maximizing pair (i', j')

The score is $\alpha := F(n, m)$
$(i, j) := (n, n)$
repeat
  if $T(i, j) = (i - 1, j - 1)$ **print** $\begin{pmatrix} x_i \\ y_j \end{pmatrix}$
  else if $T(i, j) = (i - 1, j)$ **print** $\begin{pmatrix} x_i \\ - \end{pmatrix}$
  else print $\begin{pmatrix} - \\ y_j \end{pmatrix}$
  $(i, j) := T(i, j)$
until $(i, j) = (0, 0)$.

Complexity of Needleman-Wunsch

- Space $(n+1)(m+1)$
- Time
  - $O(nm)$ for filling the matrix
  - $O(m+n)$ for backtrace
- If we need the score only then the space complexity can be reduced – how much?

Local alignment: Smith-Waterman algorithm

- Global alignment for similar (related) sequences only – whole sequences should be similar, e.g. homologous genes from related species
- In some cases the score between substrings could be larger than the score of the alignment between whole sequences – e.g. two proteins with common domain, two genes in chromosomes of different species

TCCCAGTTATGTCAGGGGACACGAGCATGCAGAGAC

AATTGCCGCCGTCGTTTTCAGCAGTTATGTCAGATC

- Let δ be a score function for an alignment. A local alignment of two strings *X* and *Y* is a global alignment of some substrings *X'* (of *X*) and *Y'* (of *Y*). An alignment *A* of substrings *X'* and *Y'* is an *optimal local alignment* of *X* and *Y* with respect to δ if

$$\delta(A) = \max_{A'=(X',Y')} \{ \delta(X',Y') \mid X' \text{ is a substring of } X,\ Y' \text{ is a sub-string of } Y \}$$

Local alignment – example

- Let X = AAAAACTCTCTCT and Y = GCGCGCGCAAAAA.
- Let $s(a, a) = +1$, $s(a, b) = -1$ and $s(a,-) = s(-, a) = -2$ be a scoring function.
- The optimal local alignment

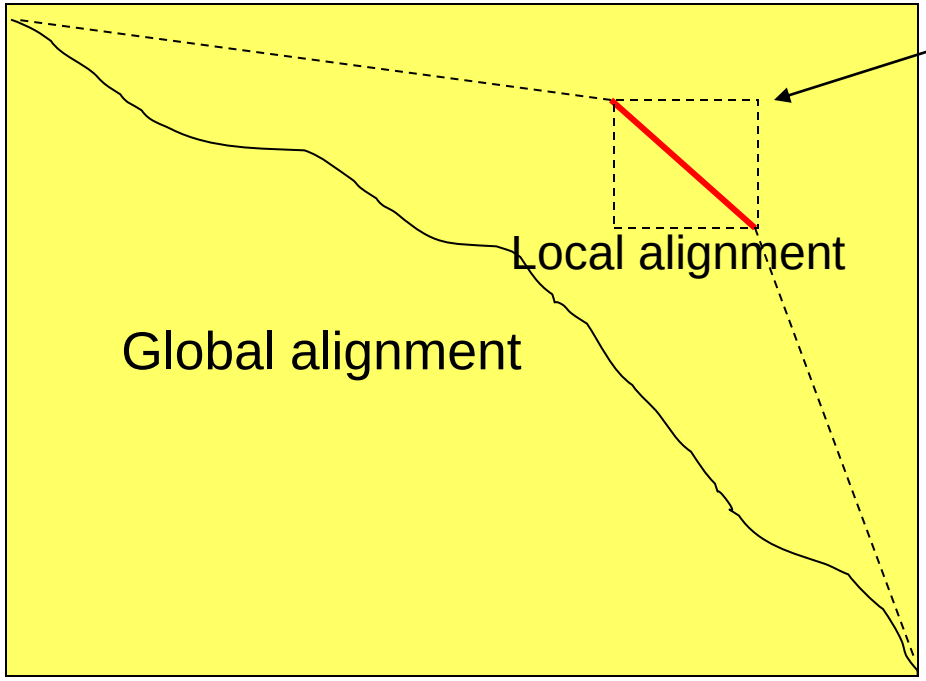
```
          AAAAA(CTCTCTCT)
          |||||
(GCGCGCGC)AAAAA
```

  in this case has a score 5
- The optimal global alignment

```
AAAAACTCTCTCT
     |  |
GCGCGCGCAAAAA
```

  has score -11.

Local Alignment: Example



Compute a “mini” Global Alignment to get Local

Local alignment: Smith-Waterman algorithm

- Smith, T. and Waterman, M. *Identification of common molecular subsequences.* J. Mol. Biol. 147:195-197, 1981
- local alignment algorithm is a modification of the global alignment algorithm – set the value of $F(i, j)$ to zero, if all attainable values at position (i, j) are negative:

$$F(i,j) = \max \begin{cases} 0 & \text{the best alignment up to } (i,j) \text{ has a negative score, then it is better to start a new one, rather than to extend the old one.} \\ F(i-1, j-1) + s(x_i, y_j) & x_i \text{ aligns to } y_j \\ F(i-1, j) - d & x_i \text{ aligns to a gap} \\ F(i, j-1) - d & y_j \text{ aligns to a gap} \end{cases}$$

Local alignment: Smith-Waterman algorithm

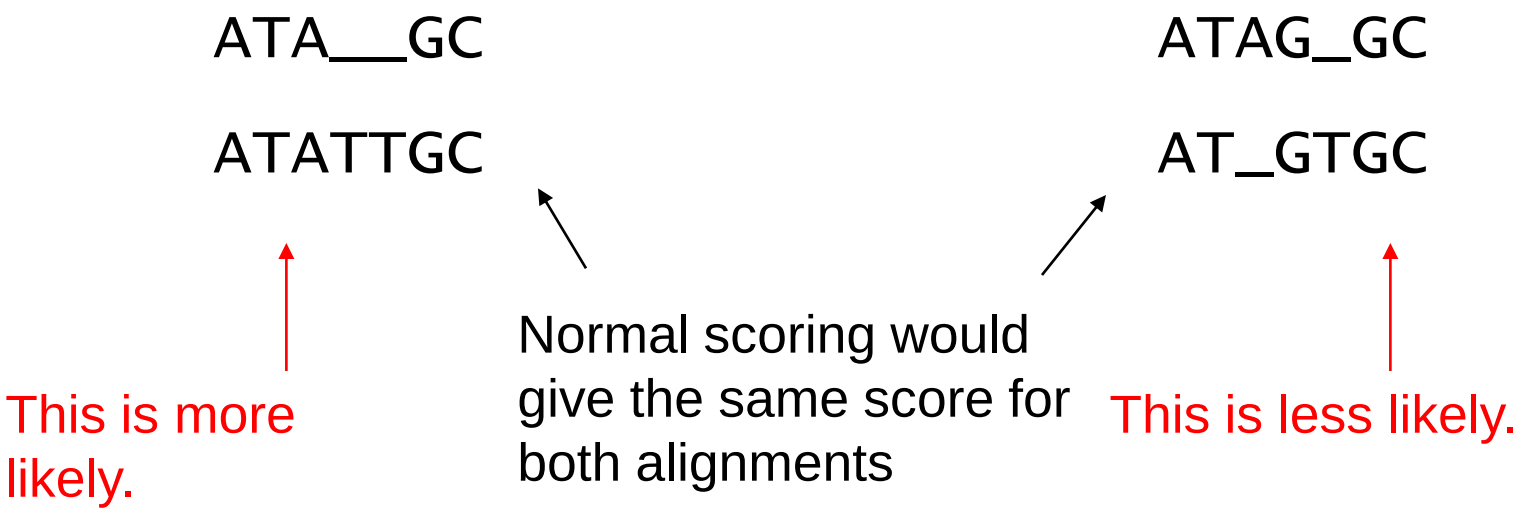
- **Base conditions**:
  - $F(i, 0) = 0$ for $i=0,\ldots,n$
  - $F(0, j) = 0$ for $j=0,\ldots,m$
- **Traceback**:
  - Instead of starting the traceback at (n,m), we start it at the cell with the highest score: argmax $F(i, j)$. The traceback ends upon arrival at a cell with score 0, which corresponds to the start of the alignment.
- **Requirement**: For this algorithm to work, we require that the expected score for a random match is negative, i.e. that

$$\sum_{a,b\in\Sigma} p_a \; p_b \; s(a,b) < 0$$

where p_a and p_b are the probabilities for seeing the symbol a or b respectively, at any given position. Otherwise, matrix entries will tend to be positive, producing long matches between random sequences.

Affine Gap Penalties

- In nature, a series of *k* indels often come as a single event rather than a series of *k* single nucleotide events:



Gap penalties

- Gaps are undesirable and thus penalized. The standard cost associated with a gap of length *g* is given either by a linear score

$$\gamma(g) = -gd$$

or an affine score

$$\gamma(g) = -d - (g-1)e\,,$$

where *d* is the gap open penalty and *e* is the gap extension penalty.

- Usually, $e < d$, with the result that less isolated gaps are produced, as shown in the following comparison:

Linear gap penalty:
```
GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKL
GSAQVKGHGKK--------VA--D----A-SALSDLHAHKL
```

Affine gap penalty:
```
GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKL
GSAQVKGHGKKVADA---------------SALSDLHAHKL
```

Affine-gap algorithm

- Instead of using one matrix $F(i, j)$ to represent the best score attainable up to x_i and y_j , we will now use three matrices M, I_x and I_y:
  - $M(i, j)$ is the best score up to (i, j), given that x_i is aligned to y_j ,
  - $I_x(i, j)$ is the best score up to (i, j), given that x_i is aligned to a gap, and
  - $I_y(i, j)$ is the best score up to (i, j), given that y_j is aligned to a gap.
- **Initialization**:

  $M(0, 0) = 0, \quad I_x(0, 0) = I_y(0, 0) = -\infty$

  $I_x(0, j) = -d - (j - 1)e, \quad M(0, j) = I_y(0, j) = -\infty,$ for $i = 1, \ldots, m$, and

  $I_y(i, 0) = -d - (i - 1)e, \quad M(i, 0) = I_x(i, 0) = -\infty,$ for $j = 1, \ldots, n$.
- We make the assumption that a gap in one sequence is not immediately followed by a gap in the other. This is true for the optimal path, if $-d - e$ is less than the lowest mismatch score.

Affine-gap algorithm

- $M(i, j)$ is the best score up to (i, j), given that x_i is aligned to y_j ,
- $I_x(i, j)$ is the best score up to (i, j), given that y_j is aligned to a gap in X, and
- $I_y(i, j)$ is the best score up to (i, j), given that x_i is aligned to a gap in Y.

- **Recursion**:

$$I_x(i,j) = \max\begin{cases} M(i,j-1) - d, & \text{begin gap in } X \\ I_x(i,j-1) - e; & \text{continue gap in } X \end{cases}$$

$$I_y(i,j) = \max\begin{cases} M(i-1,j) - d, & \text{begin gap in } Y \\ I_y(i-1,j) - e; & \text{continue gap in } Y \end{cases}$$

$$M(i,j) = \max\begin{cases} M(i-1,j-1) + s(x_i, y_j), & \text{match or mismatch} \\ I_x(i-1,j-1) + s(x_i, y_j); & \text{end gap in } X \\ I_y(i-1,j-1) + s(x_i, y_j); & \text{end gap in } Y \end{cases}$$

Affine-gap algorithm

Example

$s(a,a) = 1$

$s(a,b) = -1$

$\gamma(g) = -d - (g-1)e$ for $d = 1, e = 1$

		0	T	T	A	G	T
0	M I_x I_y						
T	M I_x I_y						
T	M I_x I_y						
G	M I_x I_y						

Simplification of the affine-gap algorithm

- Use only 2 matrices
  - *M* (corresponding to alignment) and
  - *I* corresponding to indel
- **Recursion**:

$$I(i,j) = \max\begin{cases} M(i-1,j) - d, & \text{begin gap in } Y \\ I(i-1,j) - e, & \text{continue gap in } Y \\ M(i,j-1) - d, & \text{begin gap in } X \\ I(i,j-1) - e; & \text{continue gap in } X \end{cases}$$

$$M(i,j) = \max\begin{cases} M(i-1,j-1) + s(x_i, y_j), & \text{match or mismatch} \\ I(i-1,j-1) + s(x_i, y_j); & \text{end gap in } X \text{ or in } Y \end{cases}$$

- Gives the same result if the lowest mismatch score is $> -2e$.

Simplification of the affine-gap algorithm

- Even if the lowest mismatch score is $\leq -2e$ then the difference in score and alignment will be insignificant (in a poorly matched gapped region):
  - Assume that the original algorithm (using M, I_x and I_y) produces the following optimal alignment:

    ```
    xxxx---axxxxxxxx
    yyyyyyyb----yyyy
    ```

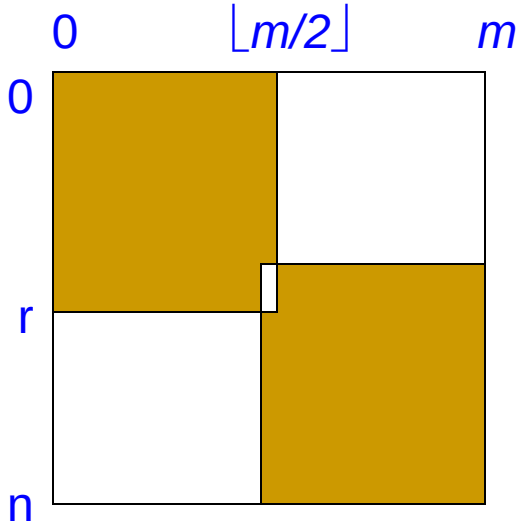
  - If $s(a, b) < -2e$, then the modified algorithm (using M and I) will produce the following higher scoring alignment, adding a gap before a and one after b:

    ```
    xxxx----axxxxxxxx
    yyyyyyyb-----yyyy
    ```

  - However, situations in which $\begin{pmatrix} - \\ b \end{pmatrix}$ is directly followed by $\begin{pmatrix} a \\ - \end{pmatrix}$ (or vice-versa) are uninteresting and so the original algorithm rules them out.

Alignment in linear space
(for global alignment)

- Computing the best score in linear space – easy.
- Problem: we must replace the trace-back matrix
- Idea: Divide et impera!
  - For the best score $F(n,m)$ compute through which row r of the middle column $\lfloor m/2 \rfloor$ the optimal trace passed.
  - Then recursively compute optimal trace from $(0,0)$ to $(r, \lfloor m/2 \rfloor)$ and from $(r, \lfloor m/2 \rfloor)$ to (n,m).



Alignment in linear space

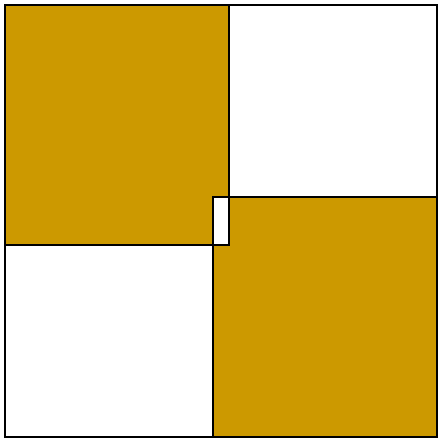
(for global alignment)

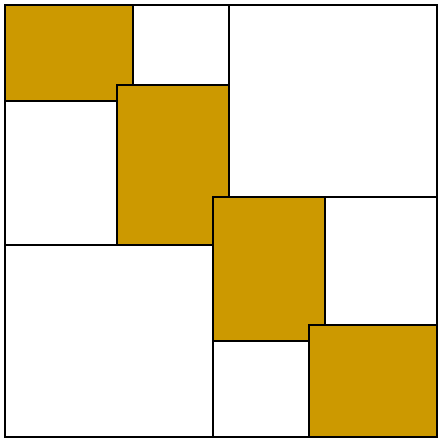
- Time complexity:
  - We first look at $1 \times mn$ cells, then at $\frac{1}{2} \times mn$ cells, then at $\frac{1}{4} \times mn$ cells etc. Further

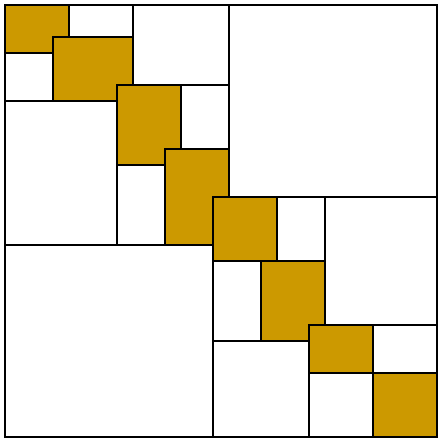
$$\sum_{i=0}^{n} \frac{1}{2^i} \times mn = 2mn$$

this algorithm is only twice as slow as the quadratic-space one!

- Space complexity: $O(n)$

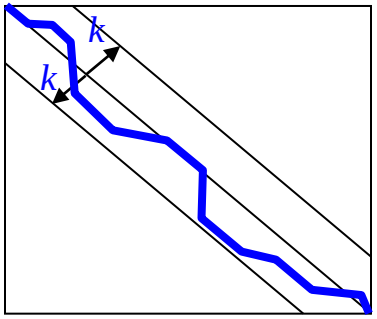


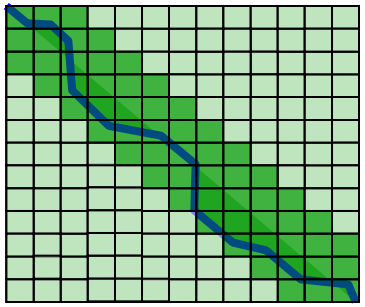




Can we do better?

- All considered variants till now have time complexity $O(mn)$ and space complexity $O(mn)$.
- For simplicity, we consider DNA sequences, assume $n = m$ and use a linear gap score d.
- **Idea:** Instead of computing the whole matrix F, use only a band of cells along the main diagonal:





- Let $2k$ denote the height of the band. Obviously, the time complexity of the banded algorithm will be $O(kn)$.
- **Questions:** Will this algorithm produce an optimal global alignment? What should k be set to?

KBand algorithm

- **Input**: two sequences X and Y of equal length n, integer k
- **Output**: the best score a of global alignment at most k diagonals away from the main diagonal
- **Computation**:

```
Set F(i, 0) := -i · d for all i = 0, 1, 2, . . . , k.
Set F(0, j) := -j · d for all j = 1, 2, . . . , k.
for i = 1 to n do
        for h = -k to k do
                j := i + h
                if 1 ≤ j ≤ n then
                        F(i, j) := F(i - 1, j - 1) + s(x_i, y_j)
                        if insideBand(i - 1, j, k) then
                                F(i, j) := max{F(i, j), F(i - 1, j) - d}
                        if insideBand(i, j - 1, k) then
                                F(i, j) := max{F(i, j), F(i, j - 1) - d}
return F(n, n)
```

To test whether (i, j) is inside the band, we use:

insideBand$(i, j, k) := (-k \leq i - j \leq k)$.

KBand algorithm

- Given two sequences X and Y of the same length n. Let M be the maximal match score (for two symbols) and d the gap penalty.

- **Question**: Let α_k be the best score obtained using the KBand algorithm for a given k. When is α_k equal to the optimal global alignment score α?

- **Lemma**: If $\alpha_k \geq M(n - k - 1) - 2(k + 1)d$, then $\alpha_k = \alpha$.

- **Proof:** If there exists an optimal alignment with score α that does not leave the band, then clearly $\alpha_k = \alpha$. Else, all optimal alignments leave the band somewhere. This requires insertion of at least $k+1$ gaps in each sequence, and allows only at most $n - k - 1$ matches, giving the desired bound.

KBand algorithm

The following algorithm computes an optimal alignment by repeated application of the KBand algorithm, with larger and larger k:

Input: two sequences X and Y of the same length n
Output: an optimal global alignment of X and Y
Computation:

```
k := 1
repeat
    Compute α_k using KBand
    if α_k ≥ M(n − k − 1) − 2(k + 1)d  then
        return α_k
    k := 2k
end
```

KBand algorithm – time complexity

The algorithm terminates when:

$$\alpha_k \geq M(n-k-1) - 2(k+1)d \quad \Leftrightarrow$$
$$\alpha_k - Mn + M + 2d \geq -(M+2d)k \quad \Leftrightarrow$$
$$-\alpha_k + Mn - (M+2d) \leq (M+2d)k \quad \Leftrightarrow$$
$$\frac{Mn - \alpha_k}{M+2d} - 1 \leq k$$

At this point, the total complexity is:

$n + 2n + 4n + \ldots + kn \leq 2kn$.

So far, this doesn't look better than nn. To bound the total complexity, we need a bound on k. When the algorithm stops for k, we must have:

$$\frac{k}{2} < \frac{Mn - \alpha_{k/2}}{M+2d} - 1$$

KBand algorithm – time complexity

$$\frac{k}{2} < \frac{Mn - \alpha_{k/2}}{M + 2d} - 1$$

There are two cases:

If $\alpha_{k/2} = \alpha_k = \alpha$, then

$$k < 2\left(\frac{Mn - \alpha}{M + 2d} - 1\right)$$

Otherwise, $\alpha_{k/2} < \alpha_k = \alpha$. Then any optimal alignment must have more than $k/2$ spaces, and thus

$$\alpha \le M\left(n - \frac{k}{2} - 1\right) + 2\left(\frac{k}{2} + 1\right)d \quad \Rightarrow \quad k \le 2\left(\frac{Mn - \alpha}{M + 2d} - 1\right)$$

As $M+2d$ is a constant, it follows that k is bounded by $O(\Delta)$, with $\Delta = Mn - \alpha$, and thus the total bound is $O(\Delta\, n)$.

In consequence, the more similar the sequences, the faster the KBand algorithm will run!

Using KBand algorithm

finding high identity alignments

- We can use the KBand algorithm as a fast method for finding high-identity alignments:
  - If we know that the two input sequences are highly similar and we have a bound *b* on the number of gaps that will occur in the best alignment, then the KBand algorithm with *k* = *b* will compute an
  - optimal alignment.

- For example, in forensics, one must sometimes determine whether a sample of human mtDNA obtained from a victim matches a sample obtained from a relative (or from a hair brush etc). If two such sequences differ by more than a couple of base-pairs or gaps, then they are not considered a match.

Alignments


Sequence alignments
dotplot
window size and stringency
repeat detection
measuring matches
% identity
conservation
scoring
Hamming distance
edit distance
computing edit distance
weighted edit distance
string distance -> string similarity
substitution matrices
PAM
BLOSUM
others
gap penalty
aligning sequences
pairwise
global
local
Alignment in linear space
KBand algorithm
multiple
global
local