Sequence Comparison

Sequence comparison has become the basic of bioinformatic operation, and it is usually the first a few things you do before conducting an experiment to tell whether two macromolecules are functionally similar or related. Although it is possible two proteins showing little sequence similarity have similar biological function, for example: So that the underlying principle of sequence comparison is based on molecular evolution, genes with similar sequences are descendants from a common ancestor in evolution history; similar primary sequences usually predicate similarity in secondary and ternary structure, as well as molecular functions.

Sequence -> Structure -> Function

There are three types of sequence comparisons or alignments we usually do with DNA and protein sequences, as listed below, different tools have been developed to be applied to each situation:

One-to-One: Pairwise alignment.
One-to-Many: Database searches.
Many-to-Many: Multiple sequence alignment, phylogenetic tree construction, discovery of motif and profiles in sequences.

Before we introduce the programs or web services for sequence comparisons, we first give a brief introduction to some basics of sequence alignments. Readers are encouraged to get more detailed background information from handouts and at our web site at http://biocomp.health.unm.edu/ and
FASTA: http://www.ebi.ac.uk/fasta33/fasta33_help.html
CLUSTALW: http://www.ebi.ac.uk/clustalw/clustalw_help.html

First, here are some helpful concepts that you will encounter when reading about sequence comparisons

Identity – the extent to which two sequences are invariant.

Similarity – The extent to which sequences are related, based on sequence identity and/or conservation.

Alignment – an alignment is an hypothesis of positional homology between nucleic acid bases or amino acids.

Conservation – changes in an amino acid sequence (proteins) that preserve the biological properties of the original residue. This is measured in most sequence comparison algorithms by substitution matrices in which scores for each position are derived from observations of the frequencies of substitutions in blocks of local alignments in related proteins.

Homology – similarity attributed to descent from a common ancestor. It may or may not result in similar function. Identification and analysis of homologies is central to phylogenetic systematics.

Orthologous – homologues sequences in different species that arose from a common ancestral gene.

Paralogous – homologues sequences within a single species that arose by gene duplication.

Sequence similarity

When aligning two related sequences, you will observe regions of perfect matches or mismatches, which includes insertion and deletions (Indels or gaps) of varying length, as well as terminal and internal mismatches, as shown below in Figure 1.

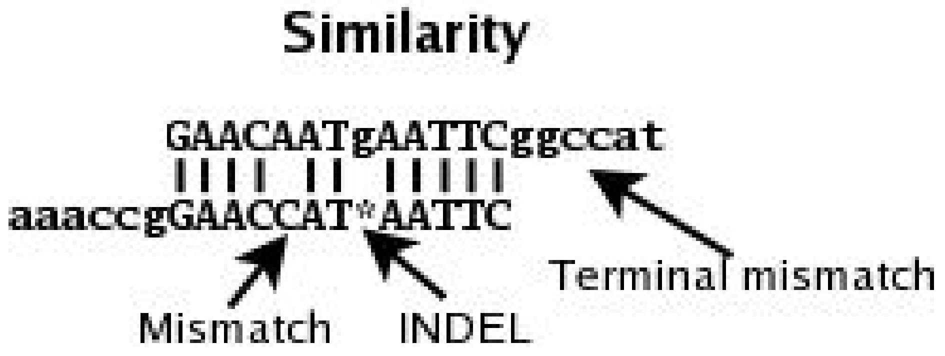


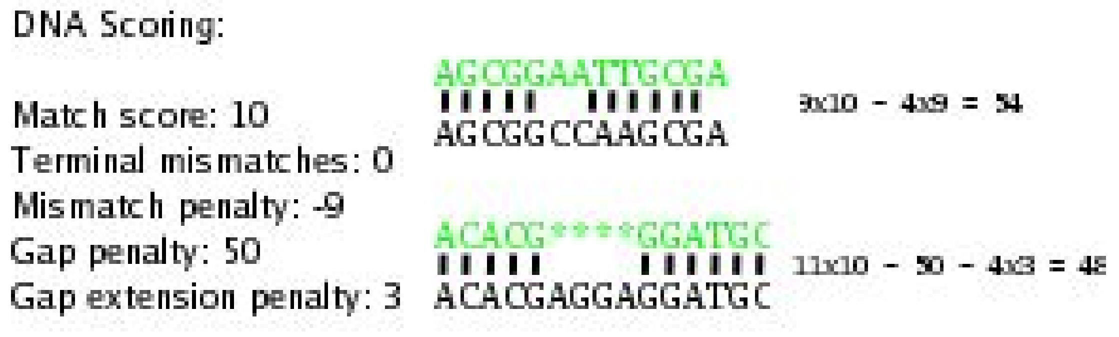
Figure 1

Understanding the scoring methods

1) Nucleic Acid Alignment:

When scoring a DNA sequence alignment, perfect matches receive positive scores, mismatches, and indels receive negative scores. There is no difference if A is substituted with a G or C or T, i.e. all positions are equally mutable and that all substitutions occur at an equal frequency – an assumption that is not always true. Figure 2 shows the default scores for DNA alignment.

An important problem is the treatment of gaps, i.e., spaces inserted to optimize the alignment score. A penalty is subtracted from the score for each gap opened (the 'gap open' penalty) and a penalty is subtracted from the score for the total number of gap spaces multiplied by a cost (the 'gap extension' penalty). Typically, the cost of extending a gap is set to be 5-10 times lower than the cost for opening a gap.



DNA Scoring:

Match score: 10
Terminal mismatches: 0
Mismatch penalty: -9
Gap penalty: 50
Gap extension penalty: 3

Figure 2

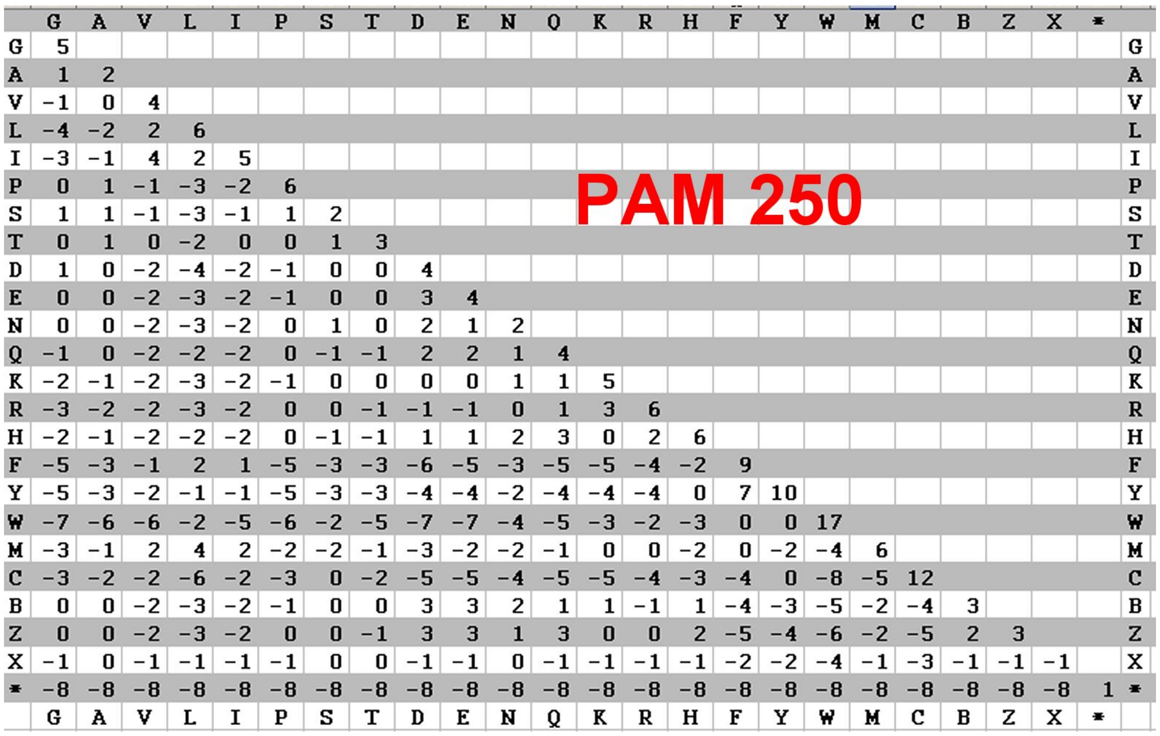
2) Substitution Matrices for Proteins

The scoring scheme for protein sequences are much complicated than nucleic acid sequences, **P**ercent **A**ccepted **M**utation [PAM :Dayhoff et. al] and **Blo**cks **Su**bstitution **M**atrix [BLOSUM :Henikoff & Henikoff] are two types of substitution matrices commonly used for calculating the alignment scores.

PAM matrices are based on related proteins, and substitution of residues are constrained by evolution and function such that 1.0 PAM unit is the amount of evolution which will change about 1% of amino acids in a protein sequence. A PAM(n) substitution matrix is a look-up table in which scores for each amino acid substitution have been calculated based on the frequency of that substitution in closely related proteins that have experienced a certain amount (n) of evolutionary divergence. For example, PAM250 has 250 changes for every 100 amino acids. In fact, different PAM matrices are derived from the PAM 1 matrix by matrix multiplication, and the matrices were further converted to log odds matrices (i.e. Log(Frequency of change / Probability of chance alignment)).

BLOSUM matrices are based on aligned sequence "blocks" [derived from the Blocks database], and adjacent "blocks" are merged at a given percentage of similarity to a larger "block", and occurrance frequencies of residues are factored in to calculate the "similarity" score; So For BLOSUM62 the similarity is set at 62%, and BLOSUM30 is set at 30%, so BLOSUM30 will be useful for detecting weak similarity sequences, while BLOSUM62 works well with closely related sequences. BLOSUM62 is the default matrix for the standard Protein BLAST program. Figure 3 shows the PAM250 and Figure 4 shows the BLOSUM62 matrices. A PAM250 matrix is roughly equivalent to a BLOSUM45 matrix.

Figure 3



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

Figure 4

3) What Matrices to use?

Here are a list of rules to consider when choosing substitution matrices:

1. BLOSUM matrices tend to be more sensitive to distant relationships than PAM.
2. BLOSUM tends to give higher scores to substitutions involving hydrophilic amino acids and lower scores to substitutions involving hydrophobic amino acids than PAM.
3. Substitutions of rare amino acids are more tolerated by BLOSUM.
4. The differences in derivation lead to some general rules:
    A. Use higher PAM or lower BLOSUM matrices for more divergent sequences.
    B. Use lower PAM or higher BLOSUM matrices for more closely related sequences.

Sequence Alignment Algorithms:

1) Global Alignment [Needleman-Wunsch algorithm]

The Needleman-Wunsch algorithm is a member of the class of algorithms that can calculate the best score and alignment in the order of m x n steps, (where 'n' and 'm' are the lengths of the two sequences). These dynamic programming algorithms were first developed for protein sequence comparison by Needleman and Wunsch, though similar methods were independently devised during the late 1960's and early 1970's for use in the fields of speech processing and computer science.

What is the optimal alignment? Dynamic programming methods ensure the optimal global alignment by exploring all possible alignments and choosing the best. It does this by reading in a scoring matrix that contains values for every possible residue or nucleotide match. Needle finds an alignment with the maximum possible score where the score of an alignment is equal to the sum of the matches taken from the scoring matrix.

In a Needleman-Wunsch global alignment, the entire length of each sequence is aligned. This can be thought of as an overlap between the two sequences (one can be completely within the other, or their ends can overlap).

This leaves no penalty for the hanging ends of the overlap. In bioinformatics, it is usually reasonable to assume that the sequences are incomplete and there should be no penalty for failing to align the missing bases.

In short, the algorithm finds the best alignment of two complete sequences that maximizes the number of matches and minimizes the number of gaps. Implementations of the algorithm include **GAP** in the GCG package and **needle** in the **EMBOSS** package.

2) Local Alignment [Smith-Waterman algorithm]

The Smith-Waterman algorithm is a member of the class of algorithms that can calculate the best score and local alignment in the order of m x n steps, (where 'n' and 'm' are the lengths of the two sequences). These dynamic programming algorithms were first developed for protein sequence comparison by Smith and Waterman, though similar methods were independently devised during the late 1960's and early 1970's for use in the fields of speech processing and computer science.

A local alignment searches for regions of local similarity between two sequences and need not include the entire length of the sequences. Local alignment methods are very useful for scanning databases or other circumstances when you wish to find matches between small regions of sequences, for example between protein domains.

Dynamic programming methods ensure the optimal local alignment by exploring all possible alignments and choosing the best. It does this by reading in a scoring matrix that contains values for every possible residue or nucleotide match. The algorithm finds an alignment with the maximum possible score where the score of an alignment is equal to the sum of the matches taken from the scoring matrix.

The Smith-Waterman algorithm contains no negative scores in the path matrix it creates. The algorithm starts the alignment at the highest path matrix score and works backwards until a cell contains zero.

Optimal alignments are found by inserting gaps to maximize the number of matches. it works really well when compare sequences of different lengths, or when looking at a particular regions of interest. Examples include **water** in the EMBOSS package and **BESTFIT** in the GCG package.

3) Heuristic Approximations to Smith-Waterman

Here are some examples in this category:

FASTA
WU-BLAST
BLAST
BLAST-2 (Gapped BLAST)

Optimal alignment algorithm like Smith-Waterman is guaranteed to generate an optimal alignment given two sequences, however with the cost of longer computation time, sometime 100 times or even 1000 times slower than heuristic algorithms, so large database searches, heuristic algorithms like BLAST or FASTA are used almost exclusively. You can find detailed algorithmic descriptions about FASTA and BLAST from the NCBI's web site, particularly there is one page focusing on the statistics of sequence similarity scores of BLAST at http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html.

Rule of thumb for Sequence Alignment

The shortest possible word size (protein:2; nt:7) gives the most sensitivity, but prolongs the search time. (default values of protein:3; nt:11) usually gives good results within reasonable amount of time.

The assumption that all point mutations occur at equal frequencies is not true. The rate of transition mutations (purine to purine or pyrimidine to pyrimidine) is approximately **1.5--5 x** that of transversion mutations (purine to pyrimidine or vice-versa) in all genomes where it has been measured (see *e.g.* Wakely *Mol Biol Evol* 11(3):436-42, 1994). This is another good reason to use protein BLAST rather than nucleic acid BLAST searches if at all possible. Matrices that take this mutation frequency bias into account have been constructed (States *et al. Methods* 3(1):66-70, 1991) and may be used in other similarity searching algorithms.

BLAST and FASTA

Very detailed background information about BLAST and FASTA has been provided in the handout. We'll use the BLAST server at the NCBI and the FASTA server at the EBI for exercises.