

# Comparison of Protein Sequences and Practical Database Searching

## Higgins&Talyor Ch8 (Overview)

---

1. Alignments of sequences
  - (a) Global *vs* local alignment
  - (b) Optimal alignment and hueristics
2. Probabilities and statistics of sequence alignment
3. Practical Database searching
  - (a) Types of comparisons
  - (b) Databases
  - (c) Algorithms
  - (d) Filtering
  - (e) Scoring matrices and gap penalties
  - (f) Command line parameters
4. Interpretation of results

## Detecting Homology

---

- By definition, *homologous* proteins have evolved from the same ancestor protein.
- The degree of sequence conservation varies among protein families.
- Homologous proteins almost always have the same folds.
- *Similarity* is a way to *deduce* homology. Others?
- In this view, database searches should be treated as *experiments* analogous to wet-lab characterization.
- Planning a good experiment requires *understanding of the methods being applied*.

## Detecting Remote Homology

---

Detecting remote homology is a very subtle issue in sequence analysis. In my opinion, there are two aspects in the remote homology detection: completeness of the family of interest and the sensitivity of sequence search program.

- Sensitivity of sequence search programs:  
For example, HMM-based algorithm can detect more distantly related seqs than a pairwise alignment algorithm.
- Completeness of the family of interest:  
Most family based seq search algorithms utilize the fact that a set of seqs are in the same family, and build a model based on the family. Then, the completeness of the family being modeled determines the sensitivity of the search algorithm.

## Alignment of Sequences

---

Suppose that a sequence **a** evolved to the sequence **b** through substitutions, insertions, and deletions. The sequence alignment can represent this transformation (evolution).

<b>a</b>	=	A	C	-	T	T	G	A
<b>b</b>	=	A	G	A	T	T	-	A
<b>score</b>	=	1	0	-1	1	1	-1	1

The scores for identities and substitutions are from the *scoring matrix*, and the scores for gaps are defined by *gap penalties*. Altogether they are called the *scoring scheme*.

Thus, there are many scores for an alignment and the scoring scheme is empirically determined for the purpose of alignment.

## Rigorous Global Alignment Algorithms

---

The rigorous alignment algorithms use dynamic programming technique to compute the *optimal* alignment. See Fig. 1 in page 171.

The sequence alignment can be divided into *global* and *local* alignments.

The global alignment algorithm was developed by Needleman and Wunch in 1970. Let  $s(a_i, b_j)$  be the similarity score of  $a_i, b_j$  (scoring matrix) and let  $\alpha > 0$  be the gap paenalty. Then the score of an alignment with  $N_{ij}$  matches of  $a_i, b_j$  and  $N_{gap}$  insertions/deletions is defined by:

$$\sum_{i,j} N_{ij} s(a_i, b_j) - N_{gap} \alpha$$

## Rigorous Global Alignment Algorithms

(Continued)

---

**Affine (linear) gap penalties** are smaller than an opening gap. If the penalty for gap of length  $k$  is  $\alpha(k)$  and  $N_{k-gap}$  is the number of gaps of length  $k$  in a given alignment, the score is defined as:

$$\sum_{i,j} N_{ij} s(a_i, b_j) - \sum_k N_{gap} \alpha(k)$$

The **global similarity** of sequences **a** and **b** is defined by the largest score of any alignment of **a** and **b**:

$$S(a, b) = \max_{alignments} \left\{ \sum_{i,j} N_{ij} s(a_i, b_j) - \sum_k N_{gap} \alpha(k) \right\}$$

## Rigorous Global Alignment Algorithms

(using the dynamic programming technique)

---

As the number of possible alignments grows exponentially in relation to the length of sequences, searching for an optimal similarity score alignment seems computationally impossible. However, we can compute the best scoring alignment using dynamic programming technique.

For sequences,  $\mathbf{a} = a_1, a_2, \dots, a_n$  and  $\mathbf{b} = b_1, b_2, \dots, b_m$ , the partial scores are stored in a  $(n+1) \times (m+1)$  table as below. For simplicity, we assume that the gap penalty is equal and constant.

**Initialization**  $S_{0,0} = 0$   $S_{i,0} = -i \alpha$  for  $1 \leq i \leq n$   $S_{0,j} = -j \alpha$  for  $1 \leq j \leq m$

**Recursion**  $S_{i,j} = \max\{S_{i-1,j-1} + s(a_i, b_j), S_{i-1,j} - \alpha, S_{i,j-1} - \alpha\}$

## Rigorous Global Alignment Algorithms

(example)

---

Let's align two sequences, **ATGCATGC** and **ATGGATC**. Seqs are aligned in three steps; initialization and computation of the alignment matrix and traceback for generating the alignment. Assuming a simple scoring scheme with a gap score of -2, match score of 7, and mismatch score of -1.

		1A	2T	3G	4G	5A	6T	7C
0	0	-2	-4	-6	-8	-10	-12	-14
1A	-2	7	5	3	1	-1	-3	-5
2T	-4	5	14	12	10	8	6	4
3G	-6	3	12	21	19	17	15	13
4C	-8	1	10	19	20	18	16	22
5A	-10	-1	8	17	18	27	25	23
6T	-12	-3	6	15	16	25	34	32
7G	-14	-5	4	13	22	23	32	33
8C	-16	-7	2	11	20	21	30	39

## Rigorous Local Alignment Algorithms

(using the dynamic programming technique)

---

There are many locally optimal (best scoring) local alignments. How can we compute them? Smith and Waterman in 1981 discovered all local optimal alignments can be computed with a simple modification to the Needleman-Wunch algorithm!

**Initialization**  $S_{0,0} = 0$   $S_{i,0} = 0$  for  $1 \leq i \leq n$   $S_{0,j} = 0$  for  $1 \leq j \leq m$

**Recursion**  $S_{i,j} = \max\{0, S_{i-1,j-1} + s(a_i, b_j), S_{i-1,j} - \alpha, S_{i,j-1} - \alpha\}$

## Rigorous Local Alignment Algorithms

(example)

---

Let's align two sequences, **ACGCATGC** and **ATGGATC**. All things are the same as in the global alignment except the initialization step.

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

## Hueristic Sequence Alignment Algorithms

---

As the rigorous alignment algorithm needs a quadratic amount of time and space in relation to the length of sequences, we cannot use it to compare a query sequence to a database of a large number of sequences such as GenBank.

Several algorithms have been developed to speed up the alignment procedure. These algorithms speed up the alignment procedure but may fail to find an optimal alignment. We call these algorithms *hueristic algorithms*. Two well known, widely used algorithms are FASTA and BLAST.

## BLAST

---

BLAST (Basic Local Alignment Search Tool) works as follows.

1. BLAST starts by locating *seeds* of similarity among the query sequence and the database sequence that score at least  $T$ .
2. The seed segments are extended *without gaps* until their similarity scores exceed a certain threshold value. These segments are called *high scoring segment pairs (HSPs)* and the segment with the highest score is called *maximum segment pair (MSP)*.
3. Multiple MSP (HSP?) regions are combined.
4. For each combination, its probability is calculated using the Poisson or sum statistics and most significant hits (lowest probability) are reported.
5. The current implementation uses dynamic programming to extend a seed in both directions.

## **PSI-BLAST**

---

PSI-BLAST (position specific iterated BLAST) uses an iterative search method.

## FASTA

---

FASTA is another heuristic sequence alignment algorithm.

1. All  $k$ -tuples of the query sequence ( $k = 1$  or  $2$  for proteins) are stored in a hash table.
2. While scanning a database sequence, all occurrences of the  $k$ -tuples are marked.
3. Ten best diagonal regions are selected and rescanned.
4. Common  $k$ -tuples – not very far apart – on the same diagonal are joined together and the best region (high scoring region) is reported with a *init1* score.
5. Nearby high scoring regions are combined (not necessarily on the same diagonal) (their scores are called a *initn* score).
6. Finally, a banded dynamic programming is run in a band around the best scoring region.

## Probability and Statistics of Sequence Alignments

---

A sequence search against a database of a large number of sequences will result in a large number of matches with different similarity scores.

When the sequence similarity is statistically significant, we can deduce, with high confidence level, that sequences are biologically related. However, the reverse is not true. There are many examples of low similarity despite functional and structural similarity.

The statistical significance of similarity scores for real sequences is defined by the probability that the same score would have been obtained for random sequences.

The *the expectation value* and *zscore* are the most widely used statistics.

## Scores for Sequence Alignments

---

There are absolute and statistical scores for a pairwise sequence alignment. The absolute scores:

- Smith-Waterman score for FASTA and many others

The statistical scores are available for a search against a database of sequences:

- Bit score for BLAST
- Zscore denotes how many units of standard deviation apart the score is from the distribution of random sequences.

*The larger the more significant.*

*Note that statistical scores depend on the size of a database!*

In addition, the expectation value (Evalue) that denotes how many random matches would occur in the database is the most widely used criterium.

*Note that the smaller the Evalue is the more significant the match is.*

## Statistics of Global Alignment

---

The distribution of global similarity scores of *random* sequences has not been characterized yet. However, we know that the expected global similarity score grows linearly with the sequence length.

*Question: why are we interested in the distribution global similarity scores of random sequences ?.*

The mean and variance of the distribution are estimated empirically by shuffling the sequences and comparing the shuffled sequences repeatedly. With the estimated mean ( $\mu$ ) and variance  $\sigma$ , we may use some statistic like zscore for a given similarity score

$$zscore = \frac{S - \mu}{\sigma}$$

## Statistics of Local Alignment Without Gaps

---

Statistics of ungapped similarity scores are well mathematically studied. Karlin and Altshul showed how to compute the expectation scores for local alignment without gaps. Practically, the score for the best local alignment (the MSP score) is the *maximum* of the scores of many independent alignments.

For the two *random* seqs of lengths  $n$  and  $m$ , the score for the best ungapped local alignment is centered around  $\frac{\log(n \times m)}{\lambda}$ .

As the sum of many random variables is distributed normally, the maximum of many random variables is distributed as an extreme value distribution. So, the score  $S$  is distributed

$$\text{Prob}(S \geq x) \sim 1 - \exp(-e^{-\lambda(x-\mu)})$$

where  $\mu = (\log(Kmn))/\lambda$ .

## Statistics of Local Alignment Without Gaps (*Cont'd*)

---

By approximating  $1 - \exp(-e^{-x})$  to  $\exp(-e^{-x})$  and given  $\mu = (\log(Kmn))/\lambda$ ,

$$\text{Prob}(S \geq x) \sim \exp(-e^{-\lambda(x-\mu)}) = e^{-\lambda x} e^{-\lambda \mu} = Kmne^{-\lambda x}$$

Then, a pairwise seq alignment with score  $S$  has a p-value of  $p$  where  $p = Kmne^{-\lambda S}$ , i.e., there is a prob  $p$  that this score could have happened by chance.

Let **p-match** a match between two seqs with a p-value  $\leq 0$ . Then, the probability  $P$  of observing at least one p-match in  $D$  seqs is, by a Poisson approximation ( $P_Y(0) = \frac{e^{-\lambda} \lambda^0}{0!}$ ),

$$P = \text{Prob}(\text{at least one p-match}) = 1 - \text{Prob}(\text{no p-matches in } D \text{ seqs}) = 1 - e^{-Dp}$$

and  $P$  can be approximated by  $P \sim Dp$  for  $Dp < 0.1$ .

To make all seqs equally likely, for a database of size  $N$  and a segment of  $m$ ,  $D$  is rewritten as  $N/m$ . Then  $P \sim KNne^{-\lambda S}$ .

The expectation value  $E$  denotes the number of random matches in  $D$  seqs is  $E = Dp = KNne^{-\lambda S}$ .

## Statistics of Local Alignment With Gaps

---

Local *ungapped* similarity score grows logarithmically with the seq's length and the size of the search space,  $\frac{\log(n \times m)}{\lambda}$ . We know that local *gapped* have the same asymptotic characteristic from the empirical study (theoretical study is quite complicated. why?). Note that empirical study should use not too low gap penalties.

After removing very high seqs (to ensure the randomness), the mean value is estimated by calculating the regression line  $S = a + b \log(n)$ . By repeating this many times, we can compute

$$zscore = \frac{S - (a + b \log(n))}{var}$$

and the distribution of z-score is approximated by the extreme value distribution

$$P = \text{prob}(zscore > x) = 1 - \exp(-e^{c_1 x - c_2})$$

and the expectation value is

$$E(zscore > x) = N \times p$$

where  $N$  is the number of seqs in the database.

## Sequence Comparison Tools

---

Programs	Query	DB	Comments
blastn, fasta, ssearch	DNA	DNA	
blastp, fasta, ssearch	Protein	Protein	
blastx, fastx	DNA	Protein	
tblastn, tfasta, tfastx	protein	DNA	
tblastx	DNA	DNA	

## Sequence Databases

---

### Protein databases

nr (Expasy at <http://www.expasy.ch/>),  
nr (NCBI at <http://www.ncbi.nlm.nih.gov/>),  
SwissProt and TrEmbl at <http://www.expasy.ch/sprot>),  
PIR at <http://pir.georgetown.edu>,  
PDB at <http://www.pdb.org>)

### DNA databases

GenBank at <http://www.ncbi.nlm.nih.gov/>,  
EMBL at <http://www.embl-heidelberg.de/> or <http://www.ebi.ac.uk>,  
DDBJ at [www.ddbj.nig.ac.jp](http://www.ddbj.nig.ac.jp)

## Filtering

---

The statistics for database searches assumes that unrelated sequences look essentially random with respect to each other. However, there are many cases where the distribution is biased. For example, repeats. In general, these sequences are called *low complexity* regions and need to be masked out *before* the database search using computational tools such as SEG or DUST.

Other sort of filtering is desirable; for example, iterative searches are prone to contamination by regions of proteins that resemble coiled-coils or transmembrane helices.

## PAM Scoring Matrix

---

PAM (Point Accepted Mutations) matrices are constructed as follows.

Given the alignments of closely related proteins,

1. the frequencies of substitution of each amino acid pair were extracted from the alignments of proteins of small evolutionary distance, below 1%, i.e., *at most one mutation per 100 amino acids*, on average.
2. These frequencies, normalized to account for the frequencies of random occurrences of single amino acids, resulted in the PAM-1 matrix.
3. Each PAM-k matrix is computed from PAM-1 by k consecutive multiplications.  $PAM-250 = (PAM - 1)^{250}$  and the score

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

## BLOSUM

---

BLOSUM matrices were constructed by direct observation of sequence alignments of related proteins, at different levels of sequence divergence.

The matrices are based on blocks – a collection of multiple alignments of similar segments without gaps, each block representing a conserved region of a protein family. These blocks provide a list of accepted substitutions, and a log-odds scoring matrix can be defined based on the observed relative frequency of aligned pairs,  $q_{ab}$ , and the expected probability,  $e_{ab}$ :

$$S_{ab} = \log \frac{q_{ab}}{e_{ab}}$$

To reduce the bias in the amino acid pair frequencies caused by multiple counts from closely related sequences, segments in a block at least  $x\%$  identity are clustered and pairs are counted between clusters.

## Choosing the Scoring Matrix

---

When comparing two sequences, the most effective matrix is to use the one which corresponds to the evolutionary distance between them. However, we do not know the evolutionary distance between them! We may have to try several matrices. To do that, we have to know the characteristics of matrices.

In general, low PAM matrices are well suited to finding short but strong similarities, while high PAM matrices are best for finding long regions of weak similarities.

It is empirically shown that BLOSUM matrices outperform PAM matrices and BLOSUM 62 and 50 are effective in detecting weak similarities.

## Position-dependent Scores

---

It may be appropriate to use position-dependent scores for mismatches and gaps, i.e, *profiles*.

1. Profiles are obtained from a multiple sequence alignment of related sequences.
2. The frequency of each amino acid at each position along the multiple alignment is then calculated.
3. These counts are normalized and transformed into probabilities.
4. Finally, the scoring matrix is defined based on these probability distributions as well as on the similarities of pairs of amino acids,

$$s_i(a) = \sum_b \text{Prob}(b \text{ at position } i) \times s(a, b)$$

## Command line Parameters

---

ssearch -Q qfile -O outfile database

fasta -Q qfile -O outfile database

blastp database qfile

blastall -p blastp -d database -i qfile -o outfile

blastpgp -i qfile -o outfile -j 2 -d database

## Interpretation of Search Results

---

1. Interpretation of the search results involves first evaluating the matches, to determine whether they are significant and therefore imply homology. Evaluate is known to work very well for this purpose.
2. However, it is dangerous to put too much trust in the query having the same function as the matched ones: functions do diverge, for example, the notion of *paralogues*.
3. Given a very high scoring hit region, additional matches to other regions need to be sought much lower in the scoring ranking. (a good term project).
4. Due to distantly related seqs, we cannot conclude that the database does not contain seqs related to the query.
5. The most powerful tools today are those that incorporate information from a group of *related* seqs.

## An Example to Show the Importance of Log-odd Ratio

---

This example is from a chapter by Anders Krogh in *Computational Methods in Molecular Biology* edited by Salzberg, Searls, and Kasif.

Given a set  $S$  of aligned sequences:

ACA---ATG  
TCAACTATC  
ACAC--AGC  
AGA---ATC  
ACCG--ATC

Given two sequences, **TCAACTATC** that is a member of  $S$  and **TGCTAGG** that is unlikely belong to  $S$ ,

$\text{Prob}(\text{TCAACTATC}) = 0.0075$  while  $\text{Prob}(\text{TGCTAGG}) = 0.00023$ .

Note that they are clearly distinguishable!

But if we compute the log-odd scores,

$\text{Prob}(\text{TCAACTATC}) = 3.0$  while  $\text{Prob}(\text{TGCTAGG}) = -0.97$ .