Chapter 7 Class Notes – Rapid Alignment Methods: FASTA and BLAST

7.1. The Biological Problem: we discuss methods to speed up the alignment process involving a query/target sequence and a search space; the researcher is usually interested in seeking info about the query sequence regarding its possible function by comparing this sequence with related sequences in so-called "model organisms" (organisms chosen for intensive genomic, genetic and/or biochemical studies), such as *Escherichia coli* (a bacterium), *Saccharomyces cerevisiae* (baker's yeast), *Caenorhabdidtis elegans* (a nematode), *Drospohila melanogaster* (fruit fly) *Arabidopsis thaliana* (mustard weed), and *Mus musculus* (the common mouse). Due to rigorous testing, many of the genes, gene products, and functions of gene products are known for these model organisms. "The important point is that a target gene is likely to have a function similar or related to functions of a homolog in one or more model organisms… rapid [alignment] methods are necessary because of the very large number of comparisons that must be made…"

7.2. Search Strategies: "one way to speed up sequence comparison is by reducing the number of sequences to which any candidate sequence must be compared. This can be done by restricting the search for a particular matching sequence to 'likely' sequence entries." [Example: our textbook (key words "genome", "probabilistic", "statistics", "alignment") and a library.] Given two sequences I and J, if I and J do not have at least some words in common, then we can decide that the strings are not similar.

We should be mindful that "the appearance of a subset of [certain common] words is a necessary but not sufficient condition for declaring that I and J have at least some sequence similarity.

7.2.1. Word Lists and Comparison by Content: instead of scanning each of 2 sequences for each k-word, we will form a new (look-up) table. For words $J =$ CCATCGCCATCG ($m = 12$) and $I =$ GCATCGGC ($n = 8$) & searching for 2-tuples ($k = 2$), we get:

Table 7.2. k-word lists for J = CCATCGCCATCG and I = GCATCGGC, $k = 2$.

J		I	
AA		AA	
AC		AC	
AG		AG	
AT	3, 9	AT	3
CA	2, 8	CA	2
CC	1, 7	CC	
CG	5, 11	CG	5
CT		CT	
GA		GA	
GC	6	GC	1, 7
GG		GG	6
GT		GT	
TA		TA	
TC	4, 10	TC	4
TG		TG	
TT		TT	

In our new notation, for the word $w =$ CG, $L_{CG}(J) = \{5, 11\}$ and $L_{CG}(J) = \{5\}$. A statistic that counts k-words in common is

$$C_k = \sum_{w} (\#L_w(I))(\#L_w(J))$$

Here, we obtain $C_2 = 0^2 + 0^2 + 0^2 + 2 \times 1 + 2 \times 1 + 2 \times 0 + 2 \times 1 + 0^2 + 0^2 + 1 \times 2 + 0 \times 1 + 0^2 + 0^2 + 2 \times 1 + 0^2 + 0^2 = 10$; then we can compare this value with some cut-off threshold value. A criticism of this statistic is that it ignores the relative positions of the k-words in the sequence.

7.2.2. Binary Searches: given that I and J have the k-words ($k = 4$ here) listed in Table 7.1 (below at right), how do we find the first word in list I, TGAT, within list J?

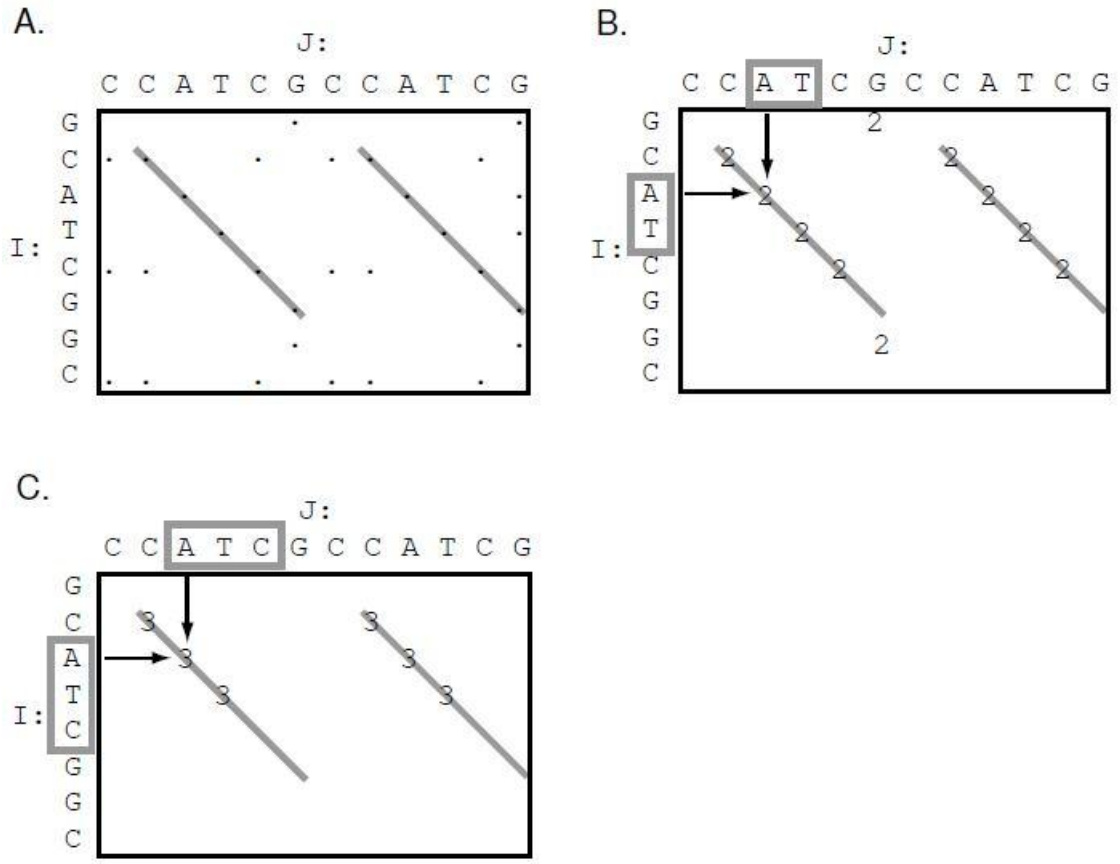
Once the list of k-words of J has been ordered, the technique of a binary search first cuts the total number of k-words in J ($m = 32$) in half and asks: does TGAT occur after entry $m/2 = 16$ in this ordered list? [yes]. Step 2: after entry $m/2 + m/4 = 24$? [yes]. Step 3: after entry $m/2 + m/4 + m/8 = 28$? [no]. Step 4: after entry $m/2 + m/4 + m/16 = 26$? [no]. Step 5: after entry $m/2 + m/4 + m/32 = 25$? [no]. Takes only 5 steps, and in general will take only $\log_2 m$ steps (which here is 5 since $\log_2(32) = 5$). Otherwise, we would have used 25 steps, and the expected number would have been $32/2 = 16$.

I		J	
TGAT		AAAT	1
GATG		AATC	2
ATGA		AATG	3
...		ACAA	4
		ATCC	5
		ATGA	6
		ATGT	7
		ATTT	8
		CAAA	9
		CCGA	10
		CGAA	11
		CGAC	12
		CGTT	13
		CTTT	14
		GAAT	15
		GACA	16
		GATG	17
		GATT	18
		GGAT	19
		GGGA	20
		GTCG	21
		TCAC	22
		TCCG	23
		TCGA	24
		TGAT	25
		TGGG	26
		TGTC	27
		TTGG	28
		TTTA	29
		TTTC	30
		TTTG	31
		TTTT	32

7.2.3. Rare Words and Sequence Similarity: for words of length k, first take sequence I of length n and sequence J of length m, and order the $(n - k + 1)$ k-words of I and the $(m - k + 1)$ k-words of J. Next, find matches in the list by alternating between the lists as needed (recording locations too).

7.3. Looking for Regions of Similarity Using FASTA: technique put forth in Wilbur & Lipman (1983) and Pearson & Lipman (1988); it relies on the Smith-Waterman local sequence alignment method.

7.3.1. Dot Matrix Comparisons: Again for $J =$ CCATCGCCATCG and $I =$ GCATCGGC example, we obtain the following dot matrices for $k = 1$ (in A), $k = 2$ (in B), and $k = 3$ (in C).



These are a special type of alignment matrices; dots in A coincide with agreement of nucleotides, '2's in B coincide with first letters of matching 2-tuples, and '3's in C for first letters of matching 3-tuples. Notice here there are two predominant diagonals (denoting regions of similarity of the sequences), and also that (# of 3s in C) ≤ (# of 2s in B) ≤ (# of dots in A).

7.3.2. FASTA – Rationale: the rationale is observed in the above dot matrices for $k > 1$, only focusing on the diagonals. We index diagonals by letting $l = i - j$, and we need to calculate scores S_l

as follows: work through Table 7.2 one word at a time and tally the number of times we observe a match between the k-word at position i in I and $j + l$ in J (for $k - m \leq l \leq n - k$).

To illustrate with the example, $k = 2, m = 12, n = 8$, so $-10 \leq l \leq 6$; each S_l is initialized at 0. We then get $S_{-6} = 4$ since $l = i - j = -6$ for words **AT**, **CA**, **CG** and **TC**. We also find $S_{-5} = 1, S_0 = 4, \; and \; S_1 = 1$; this coincides with dot matrix B above.

Table 7.2. k-word lists for J = CCATCGCCATCG and I = GCATCGGC, $k = 2$.

J		I	
AA		AA	
AC		AC	
AG		AG	
AT	3, 9	AT	3
CA	2, 8	CA	2
CC	1, 7	CC	
CG	5, 11	CG	5
CT		CT	
GA		GA	
GC	6	GC	1, 7
GG		GG	6
GT		GT	
TA		TA	
TC	4, 10	TC	4
TG		TG	
TT		TT	

The FASTA pseudo-code is given in Box 7.1 on p.176; the 5 steps in FASTA are:

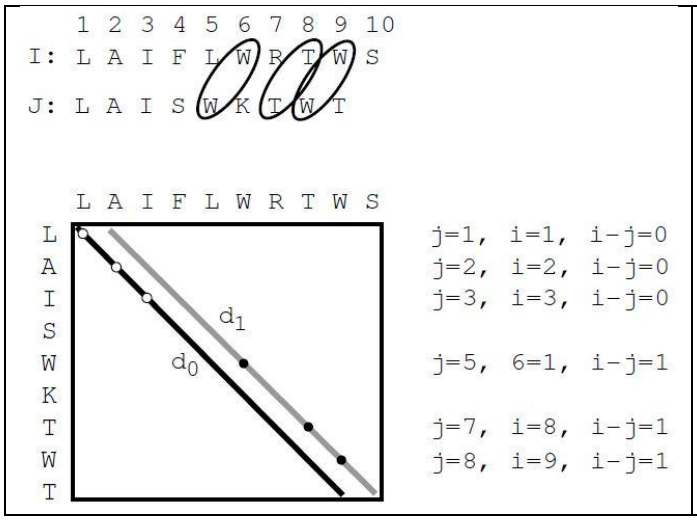
1. Create the "look-up table" as above at right, creating the table first by passing through I and recording the positions i $(1 \leq i \leq n - 1)$ of the k-words, and then pass through J to one-by-one look up the k-words of I and record the position
2. Identify high-scoring diagonals as described above (by finding the respective $S_l, k - m \leq l \leq n - k$), and choose the top 10 of these diagonals
3. Rescore these diagonals; this rescoring is needed as shown:

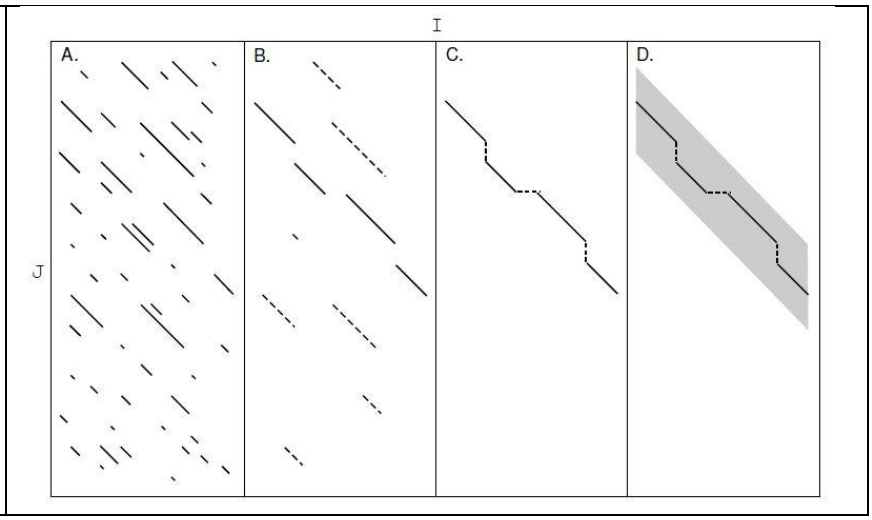
```
I:  C C A T C G C C A T C G      (Number 4-tuple matches: 0)
J:  C C A A C G C A A T C A

I': C C A T C G C C A T C G
J': A C A T C A A A T A A A
```

They're looking for 4-tuple matches here. In the first instance, there are no matches even though the sequences are 75% identical; in the second instance, there is a match, but the sequences are only 33% identical. "Rescoring reveals sequence similarity not detected because of the arbitrary demand for uninterrupted identities of length k."

4. Join the initial regions with the aid of appropriate joining or gap penalties for short alignments on offset diagonals; that is, some diagonals may be offset from each other (i.e. via a gap) as below at left (for two proteins). Diagonals d_0 and d_1 should be joined – the offset here is $l = 1$.
5. The final step is to perform a rigorous Smith-Waterman local alignment.

The above steps are displayed below at right: 'A' illustrates Step 2 (finding diagonals), 'B' shows the rescored plot of Step 3, 'C' gives joined regions in Step 4, and 'D' illustrates the results of the local alignment.





7.4. BLAST (Basic Local Alignment Search Tool): High-scoring local alignments are called "high scoring segment pairs", or HSPs. The output of BLAST is a set of HSPs and associated p-values.

7.4.1. Finding Local Matches: a given query sequence is used as a template to construct a set of subsequences of length w that can score at least T when compared to the query; a substitution matrix containing neighborhood sequences is used. To illustrate:

I **is query and** J **is search space:**

$$J = C\ C\ A\ T\ C\ G\ C\ C\ A\ T\ C\ G$$
$$I = G\ C\ A\ T\ C\ G\ G\ C$$

5-words in I	5-words in J	J position(s)	Score
CATCG	CATCG	2, 8	5
GCATC	CCATC	1	4
ATCGG	ATCGC	3	4
TCGGC	TCGCC	4	4

$$\begin{Bmatrix} A \\ C \\ T \end{Bmatrix} CATC, \quad G \begin{Bmatrix} A \\ G \\ T \end{Bmatrix} ATC, \quad GC \begin{Bmatrix} C \\ G \\ T \end{Bmatrix} TC, \quad GCA \begin{Bmatrix} A \\ C \\ G \end{Bmatrix} C, \quad GCAT \begin{Bmatrix} A \\ G \\ T \end{Bmatrix}$$

Here, subsequence length is $k = 5$ and the threshold value is $T = 4$; start with GCATC in I, build the neighborhood of 15 new words as above (bottom), so the neighborhood contains 16. Then, do the same for CATCG, ATCGG and TCGGC in I, giving 64 5-word patterns in the neighborhood. Now look for exact matches (called "seed" hits) of each of these in J as above at top right. The final step of BLAST is in producing un-gapped extensions from these seed hits.

7.4.2. Scores: here a p-value will be the probability of a more extreme score than the one found (in the complete database); if a random sequence is Y, the database is D, and the score is s, this is $P(S(D, Y) \geq s)$. Here, matching score is 1 and mismatch is $-\infty$

(no mismatches and no indels), and the alignment matrix is $n \times m$. Let $p = P(\text{two random letters are equal})$, so a mismatch followed by t identities has probability $(1-p)p^t$. Since there are nm places to begin the alignment, the expected number (mean) of local alignments of at least length t is $\lambda = nm(1-p)p^t$. We want this to be a rare event, so this is well-modelled by the Poisson distribution with mean λ. Thus, the probability there is a local alignment of length t or longer (p-value) is approximately

$$1 - P(no\ such\ event) = 1 - e^{-\lambda} = 1 - e^{-nm(1-p)p^t}$$

Even though some of these calculations are approximate, this is the reasoning behind BLAST's calculation of E-values (on right):

$$P(S(D,Y) \geq s) \approx 1 - e^{-nm\gamma\xi^t}$$

Here, $\gamma > 0$ and $0 < \xi < 1$ are parameters to be estimated.

7.5. Scoring Matrices for Protein Sequences: as pointed out in Chapter 6 (for DNA), scoring matrices must make good (biological) sense. We now turn to aligning residues of proteins such as:

$$X = \cdots NVSDVNLNK \ldots$$
$$Y = \cdots NASNLSLSK \ldots$$

Thus, we need to find the probability p_{ab} of matching amino acid a with amino acid b: we wish to find the scoring. In general, for the sequences $A = a_1 a_2 \ldots a_n$ and $B = b_1 b_2 \ldots b_n$, under a random (indep.) model (R), the probability of having A and B is

$$P(A,B|R) = \prod_i q_{a_i} \prod_i q_{b_i}$$

Here, for example, q_{a_1} is the probability of amino acid of type a_1 irrespective of position. Under the "match" (M) model, we have:

$$P(A, B|M) = \prod_i p_{a_i b_i}$$

Alignments of proteins A and B use the score calculation:

$$S = \log_2 \frac{P(A, B|M)}{P(A, B|R)} = \sum_{i=1}^{n} \log_2 \left(\frac{p_{a_i b_i}}{q_{a_i} q_{b_i}} \right) = \sum_{i=1}^{n} s(a_i, b_i)$$

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

The above BLOSUM62 matrix is a substitution matrix for proteins, and it essentially contains the scores, $s(a_i, b_i) = \log_2\left(\frac{p_{a_i b_i}}{q_{a_i} q_{b_i}}\right)$. To develop these: $s(A, A), s(R, A), s(R, R), \ldots$ (with A = alanine, R = arginine, … as in the matrix on p.183), we can first estimate the "random" probabilities such as q_a by counting the number of each amino acid type in an appropriate collection of protein sequences and then dividing by the total number of amino acids present. Estimating the p_{ab} is as follows (see pp.186-7):

First "blocks" (using the Blocks database: each block consists of n aligned sequences each having w residues), and we count the number of pairwise matches and mismatches for each amino acid type. Each block provides $w \times n(n-1)/2$ possible pairings, and this is done for more than 24,000 blocks. Frequencies are then arranged into a 20×20 frequency matrix of the form:

	A	R	N	D	…	V
A	$f_{A,A}$					
R	$f_{R,A}$	$f_{R,R}$				
N	$f_{N,A}$	$f_{N,R}$	$f_{N,N}$			
D	$f_{D,A}$	$f_{D,R}$	$f_{D,N}$	$f_{D,D}$		
…	…	…	…	…	…	
V	$f_{V,A}$	$f_{V,R}$	$f_{V,N}$	$f_{V,D}$	…	$f_{V,V}$

Then, for each of the $\frac{20(19)}{2} + 20 = 210$ entries, $p_{ab} = f_{ab} / \sum_{a=1}^{20} \sum_{b=1}^{a} f_{ab}$. Then, we take $s(a, b) = \log_2\left(\frac{p_{ab}}{q_a q_b}\right)$, and in practice, these are rounded off and scaled. For BLOSUM, the scores are reported in half-bits: $s(a, b)[half - bits] = 2s(a, b)$

Variation on Computational Example 7.2: using the BLOSUM62 matrix, give the alignment score for MQLEANADTSV and LQECAEAQGEM. Here, we get the score:

$$S = 2 + 5 - 3 - 4 + 4 + 0 + 4 + 0 - 2 + 0 + 1 = 7\ (half - bits)$$

Next, check the BLOSUM62 matrix to make sure that it makes sense. First, since tryptophan (W) is a relatively rare amino acid, it makes sense that the score for it being conserved (11) is high. Second, the score for matching D (aspartic acid) and L (leucine) is very low (−4) since the codons for the former, {GAC, GAU}, are quite "far" or "distant" from those for the latter, {UUA, UUG, CUU, CUC, CUA, CUG} – in the sense that they are two mutations away; also, D is polar and negatively charged whereas L is nonpolar and neutral. Note too how "close" (3) are I {AUU, AUC, AUA} and V {GUU, GUC, GUA, GUG}.

7.6. Tests of Alignment Methods: what are the chances of finding in a database search HSPs that are not homologs? Interestingly, "the three dimensional structures of homologs and their domain structures will be conserved, even though the proteins may have diverged in sequence." Also, a good alignment programs must meet two criteria – it must maximize the number of homologs found (true positives; sensitivity) and it must minimize the number of nonhomologous proteins found (false positives; specificity). Brenner *et al* (1998) reported some testing of Smith-Waterman, FASTA and the earlier BLAST.

Not great news: they found that at best only about 35% of homologs were detectable at a false positive error frequency of 0.001 per query sequence.

In the past, a rule of thumb was that sequence identities of 25-30% in an alignment signified true homology. To test this, these authors used a database of known proteins annotated with respect to homology/non-homology relationships to test the relationship between sequence identity and homology. Their results are below (plot of percent identity within the alignment versus alignment length) for proteins that are not homologs. Note that for alignments 100 residues in length, about *half* of the non-homologous proteins show more that 25% sequence identity.

This clearly shows why the statistical analysis of HSPs (i.e. E-values) is needed.