Database Similarity Searching:

What's available — the algorithms and programs, and ascertaining significance — in pairwise sequence comparison

Content: The dynamic programming algorithm, symbol comparison tables, dot matrices, motifs, hashing, and heuristics. What do database searches tell us; what can we gain from them; why even bother? Given a nucleotide or amino acid sequence, what can we know about its function? We may find biologically relevant information in it by searching for particular patterns, which may reflect some potential function of the molecule. But, what about comparisons with other sequences? Can we learn about one molecule by comparing it to another? Yes, naturally we can; inference through homology is fundamental to all the biological sciences. But what sort of a comparison is significant, and what level of significance implies homology? These questions are a major focus of the workshop. By comparing the conserved portions of sequence amongst a set, all of the sensitivity and power of the computational techniques is magnified.

Fall 2006; a GCG¥ Wisconsin Package™ SeqLab® tutorial for Florida State University sponsored by the School of Computational Science (SCS).

Author and Instructor: Steven M. Thompson

Steve Thompson
BioInfo 4U
2538 Winnwood Circle
Valdosta, GA, USA 31601-7953
stevet@bio.fsu.edu
229-249-9751

Introduction

I've got a sequence: Now what? Search the databases — what shows up?

Given the nucleotide or amino acid sequence of a biological molecule, what can we know about that molecule? We can find biologically relevant information in it by searching for particular patterns that may reflect some function of the molecule. These can be catalogued motifs, secondary structure predictions and physical attributes such as hydrophobicity, or even the content of the DNA itself as in some of the gene finding techniques. But what about comparisons with other sequences? Can we learn about one molecule by comparing it to another? Yes, certainly we can; inference through homology is fundamental to all the biological sciences.

By comparing the conserved portions of sequence amongst a set, all of the sensitivity and power of the computational techniques is magnified. The basic assumption is that those portions of sequence of crucial functional value are most constrained against evolutionary change. They will not tolerate many mutations. Not that mutations don't happen in these portions, just that most mutations in the region are lethal so we never see them. Other areas of sequence are able to drift more readily and are subject to less evolutionary pressure. Therefore, the sequence ends up a mosaic of quickly and slowly changing regions over evolutionary time. However, in order to learn anything by comparing sequences, we need to know how to compare them. We can use those constrained portions as 'anchors' to create a sequence alignment so that we can compare them. But this brings up the alignment problem and 'similarity.' It is easy to see that two sequences are aligned when they have identical symbols at identical positions, but what happens when symbols are not identical or the sequences are not the same length? How can we know that the most similar portions of our sequences are aligned, when is an alignment optimal, and does optimal mean biologically correct? How can anybody figure any of this out?

A 'brute force' approach just won't work. Even without considering the introduction of gaps, the computation required to compare all possible alignments between two sequences requires time proportional to the product of the lengths of the two sequences. Therefore, if the two sequences are approximately the same length (N), this is a N^2 problem. To include gaps, we would have to repeat the calculation 2N times to examine the possibility of gaps at each possible position within the sequences, now a N^{4N} problem. Waterman illustrated the problem in 1989 stating that to align two sequences 300 symbols long, 10^{88} comparisons would be required, about the same number as the number of elementary particles estimated to exist in the universe! Part of the solution to this problem is known as the dynamic programming algorithm.

In dynamic programming simplest implementation we will consider matching symbols to be worth one point and will not consider gapping at all. This simple example will be illustrated first. The solution occurs in two stages and is illustrated on the following page. The first stage begins very much like the dot matrix methods described later in the tutorial; the second is totally different. I will use an over-simplified example here. Instead of calculating the 'score matrix' on the fly, as is often shown as one proceeds through the graph, I like to completely fill in an original 'match matrix' first, and then add points to those positions which produce favorable alignments next. Points are added based on a "looking back over-your-left-shoulder" algorithm rule:

a) A completed 'match matrix' using one point for matching and zero points for mismatching:

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

b) Now begin to add points based on the best path through the matrix, always working diagonally, left to right and top to bottom. The second row is completed here:

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

c) Continue adding points based on the best previous path through the matrix. The third row is completed here:

		A	A	T	G	C
	A	1	1	0	0	0
	G	0	1	1	2	1
	G	0	0+1	0+1	1+1	0+2
	C	0	0	0	0	1

d) The score matrix is now complete:

		A	A	T	G	C
	A	1	1	0	0	0
	G	0	1	1	2	1
	G	0	1	1	2	2
	C	0	0+1	0+1	0+1	1+2

e) Now pick the bottom, right-most, highest scores in the matrix and work your way back through it, in the opposite direction as before. This is called the traceback stage and the matrix is now referred to as the path graph. In this case that highest score is in the right-hand corner, but it need not be:

		A	A	T	G	C
	A	1	1	0	0	0
	G	0	1	1	2	1
	G	0	1	1	2	2
	C	0	1	1	1	**3**

f) The completed traceback is shown with outline characters; these are all optimal alignments:

		A	A	T	G	C
	A	**1**	**1**	0	0	0
	G	0	**1**	**1**	**2**	1
	G	0	1	1	**2**	2
	C	0	1	1	1	**3**

The following alignments are all generated from the above path graph (f). All five have three matches. Gap penalties would have eliminated the last two of them; however, that still leaves three:

```
AG.GC        A.GGC        .AGGC        A..GGC        .A.GGC
|  ||        |  ||         | ||        |  | |         | | |
AATGC        AATGC        AATGC        AATG.C        AATG.C
```

The software will arbitrarily (based on some rule) choose only one of these to report as optimal. This decision can be partly controlled in some of the GCG programs such as BestFit and Gap with the HighRoad/LowRoad option. This brings up an important point. Just because dynamic programming is guaranteed to find an optimal alignment, it is not necessarily the only optimal alignment. Furthermore, the optimal alignment is not necessarily the 'right' or biologically relevant alignment! As always, question the results of any computerized solution based on what you know about the biology of the system. The above example illustrates the Needleman and Wunsch (1970) global solution (the Gap program in GCG). Later refinements (Smith and Waterman, 1981) demonstrated how dynamic programming could also be used to find optimal local alignments. The GCG program BestFit performs dynamic programming using this local method. To solve dynamic programming using local alignment (without going into all the gory details) programs use the following tricks:

1) An identity match matrix that uses negative numbers for mismatches is incorporated. Therefore, bad paths quickly become very bad. This leads to a traceback path matrix with many alternative paths, most of which do not extend the full length of the graph.
2) The best traceback within the graph is chosen. This does not have to begin or end at the edges of the graph — it is looking for the best segment of alignment!

The next example will be slightly more difficult. Unlike the previous example without gap penalties, I will now impose a very simple gap penalty function. I am going to penalize the scoring scheme by subtracting 1 point for every gap inserted unless they are at the beginning or end of the sequence. In other words, end gaps will not be penalized; both sequences do not have to begin or end at the same point in the alignment. This zero penalty end-weighting scheme is the default for most alignment programs, but can often be changed with a program option, if desired. However, the gap function being used in the example below is a much simpler gap penalty function than normally used in alignment programs. Normally an 'affine,' i.e. a linear, function is used:

total penalty = gap opening penalty + ([length of gap] * [gap extension penalty]).

(To run the GCG alignment programs with the type of simple gap penalty used in the following example, you would have to designate a 'gap creation' penalty of zero and a 'gap extension' penalty of ten for DNA sequences, since GCG's default DNA match matrix uses a score of ten for identical base matches.)

The example illustrated on the following page uses two randomly generated sequences that happen to fit the tata consensus regions of Eukaryotes and Bacteria. The sample Eukaryote promoter sequence is along the X-axis, the Bacteria along the Y-axis below.

a) First complete a match matrix using one point for matching and zero points for mismatching between bases, just like before:

	c	T	A	T	A	t	A	a	g	g
c	1	0	0	0	0	0	0	0	0	0
g	0	0	0	0	0	0	0	0	1	1
T	0	1	0	1	0	1	0	0	0	0
A	0	0	1	0	1	0	1	1	0	0
t	0	1	0	1	0	1	0	0	0	0
A	0	0	1	0	1	0	1	1	0	0
a	0	0	1	0	1	0	1	1	0	0
T	0	1	0	1	0	1	0	0	0	0

b) Now add and subtract points based on the best path through the matrix, working diagonally, left to right and top to bottom. When you have to jump a box to make the path, subtract one point per box jumped, except at the beginning or end of the alignment. Fill in all additions and subtractions and calculate the totals as you go:

	c	T	A	T	A	t	A	a	g	g
c	1	0	0	0	0	0	0	0	0	0
g	0	0+1=1	0+1-1=0	0+0-0=0	0+0-0=0	0+0-0=0	0+0-0=0	0+0-0=0	1+0-0=1	1+0=1
T	0	1+1-1=1	0+1=1	1+1-1=1	0+0-0=0	1+0-0=1	0+0-0=0	0+0-0=0	0+0-0=0	0+1-0=1
A	0	0+0-0=0	1+1=2	0+1=1	1+1=2	0+1-1=0	1+1=2	1+1-1=1	0+0-0=0	0+0-0=0
t	0	1+0-0=1	0+1-1=0	1+2=3	0+1=1	1+2=3	0+2-1=1	0+2=2	0+1=1	0+0-0=0
A	0	0+0-0=0	1+1=2	0+2-1=1	1+3=4	0+3-1=2	1+3=4	1+3-1=3	0+2=2	0+1=1
a	0	0+0-0=0	1+0-0=1	0+2=2	1+3-1=3	0+4=4	1+4-1=4	1+4=5	0+3=3	0+2=2
T	0	1+0-0=1	0+0-0=0	1+1=2	0+2=2	1+3=4	0+4=4	0+4=4	0+5=5	0+5-1=4

c) I'll clean up the score matrix next, only showing the totals in each cell:

	c	T	A	T	A	t	A	a	g	g
c	1	0	0	0	0	0	0	0	0	0
g	0	1	0	0	0	0	0	0	1	1
T	0	1	1	1	0	1	0	0	0	1
A	0	0	2	1	2	0	2	1	0	0
t	0	1	0	3	1	3	1	2	1	0
A	0	0	2	1	4	2	4	3	2	1
a	0	0	1	2	3	4	4	5	3	2
T	0	1	0	2	2	4	4	4	5	4

d) Finally, convert the score matrix into a traceback path graph by picking the bottom-most, furthest right and highest scoring coordinate (this is the overall alignment score) tracing back, choosing those routes that led to that highest score, to connect them all the way back to the beginning:

	c	T	A	T	A	t	A	a	g	g
c	**1**	0	0	0	0	0	0	0	0	0
g	0	**1**	0	0	0	0	0	0	1	1
T	0	1	1	**1**	0	1	0	0	0	1
A	0	0	2	1	**2**	0	2	1	0	0
t	0	1	0	3	1	**3**	1	2	1	0
A	0	0	2	1	4	2	**4**	3	2	1
a	0	0	1	2	3	4	4	**5**	3	2
T	0	1	0	2	2	4	4	4	**5**	4

There may be more than one best path through the matrix. This time, starting at the top and working down as we did, then tracing back, I found one optimum alignment, but there are probably more:

```
cTATAtAagg
|  |||||
cg.TAtAaT.
```

This solution has a final score of 5; this is the number optimized by the algorithm, not any type of a similarity or identity percentage score! It is the GCG HighRoad solution found when running the program Gap with the above example's parameter settings (here the score is 50 since GCG's default DNA match score is 10, i.e. 5X10):

```
GAP of: Euk_Tata.Seq  to: Bact_Tata.Seq

Euk_Tata: A random example Eukaryotic promoter TATA Box
Preferred region:  center between -36 and -20.

Bact_Tata: A random sequence that fits the consensus from the
standard E. coli RNA polymerase promoter 'Pribnow' box -10 region.

           Gap Weight:   0        Average Match:  10.000
        Length Weight:  10     Average Mismatch:   0.000

            HighRoad option               LowRoad option

           Quality:     50              Quality:     50
             Ratio:  6.250                Ratio:  6.250
Percent Similarity: 75.000    Percent Similarity: 62.500
            Length:     10                Length:     10
              Gaps:      2                  Gaps:      0
  Percent Identity: 75.000      Percent Identity: 62.500

       1 cTATAtAagg 10               1 cTATAtAagg 10
         |  |||||                         |||||
       1 cg.TAtAaT. 8                1 .cgTAtAaT. 8
```

Do you have any ideas about how others, such as GCG's LowRoad solution, could be discovered by hand? Often if you reverse the solution of the entire dynamic programming process, other solutions will be found!

Now that we have worked through a couple examples of dynamic programming by hand, let's formalize the steps. An optimal alignment is defined as an arrangement of two sequences, 1 of length *i* and 2 of length *j*, such that:

1) you maximize the number of matching symbols between 1 and 2;
2) you minimize the number of indels (gaps) within 1 and 2; and
3) you minimize the number of mismatched symbols between 1 and 2.

Therefore, the actual solution can be represented by:

$$S_{ij} = s_{ij} + \max \begin{Bmatrix} S_{i-1\ j-1} & \text{or} \\ \max S_{i-x\ j-1} + w_{x-1} & \text{or} \\ 2 < x < i & \\ \max S_{i-1\ j-y} + w_{y-1} & \\ 2 < y < i & \end{Bmatrix}$$

where Sij is the score for the alignment ending at i in sequence 1 and j in sequence 2,
sij is the score for aligning i with j,
wx is the score for making a x long gap in sequence 1,
wy is the score for making a y long gap in sequence 2,
allowing gaps to be any length in either sequence.

But what about protein sequences — conservative replacements and similarities — as opposed to identities? This is definitely an additional complication to consider. Certain amino acids are very much alike, structurally,

chemically, and genetically. How can we take advantage of the similarity of amino acids in our alignments? People have been struggling with this problem since the late 1960's. Margaret Dayhoff (Schwartz and Dayhoff, 1979) unambiguously aligned closely related datasets (no more than 15% difference) available at that point in time and noticed that certain residues, if they mutate at all, are prone to change into certain other residues. As it works out, these propensities for change fell into the same categories that chemists had known for years — those same chemical and structural classes, conserved through the evolutionary constraints of natural selection. However, Dayhoff's empirical observation quantified these changes. Based on the multiple sequence alignments that she created, the assumption that estimated mutation rates in closely related proteins can be extrapolated to more distant relationships, and matrix and logarithmic mathematics, she empirically specified the relative probabilities at which different residues mutate into other residues through evolutionary history as appropriate within some level of divergence between the sequences considered. This is the basis of the famous PAM (corrupted acronym of accepted point mutation) 250 (meaning that the matrix has been multiplied by itself 250 times) table. Since Dayhoff's time other biomathematicians (esp. see Henikoff and Henikoff's [1992] BLOSUM series of tables) have created newer, more powerful tables than Dayhoff's, but the concept remains the same and Dayhoff's original PAM 250 table remains the classic as historically the most widely used one.

Collectively these types of tables are known as symbol comparison, substitution, or scoring matrices and they are fundamental to all sequence comparison techniques. The default matrix for most protein similarity comparison programs is the BLOSUM62 table (Henikoff and Henikoff, 1992). It follows below; values whose magnitude is ≥ ±4 are drawn in outline characters. Notice that positive values for identity range from 4 to 11 and negative values for those substitutions that rarely occur go as low as –4. The most conserved residue is tryptophan with a score of 11; cysteine is next with a score of 9; both proline and tyrosine get scores of 7 for identity.

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

Database searches use these concepts and some more tricks discussed below. But what do database searches tell us and what can we gain from them? Why even bother? As I stated earlier, inference through homology is a fundamental principle in biology. When a sequence is found to fall into a preexisting group we can infer function, mechanism, evolution, and possibly even structure based on homology with its neighbors. Database searches can even provide valuable insights into enzymatic mechanism. Are there any 'families' that your newly discovered sequence falls into? Even if no similarity can be found, the very fact that your sequence is new and different could be very important. Granted, it's going to be a lot more difficult to discover functional and structural data about it, but in the long run its characterization might prove very rewarding.

One small database that you should always screen when working with a protein sequence, just as a matter of course, is PROSITE. The GCG program Motifs performs this search. Motifs searches for recognized structural, regulatory and enzymatic consensus sequences in the *PROSITE Dictionary of Protein Sites and Patterns* (Bairoch, 1992). This approach is wonderful for trying to ascertain function in an unidentified peptide sequence, but keep in mind inherent problems related to either too much or too little specificity that consensus style searches can have. The program can tolerate mismatches with a mismatch option and it displays an abstract with selected references for each motif signature found.

A big question and a very common mistake that is made in this whole area of searching and alignment is the concept of homology versus similarity: There is a huge difference! Similarity is merely a statistical parameter that describes how much two sequences, or portions of them, are alike according to some set scoring criteria. Homology, by definition, implies an evolutionary relationship — more than just the fact that all of life evolved from the same primordial 'urancestral schmooze.' You need to be able to demonstrate some type of lineage between the organisms or genes of interest in order to claim homology. Even better, be able show some experimental evidence, morphological, genetic, or fossil, that corroborates your assertion. There is really no such thing as percent homology; something is either homologous or it is not. Walter Fitch likes to relate the joke "homology is like pregnancy — you can't be 45% pregnant, just like something can't be 45% homologous. You either are or you are not." Do not make the mistake of calling any old sequence similarity homology; it is a serious blunder, especially offensive to evolutionists, though it is a particularly common misnomer.

So, how do you tell if a similarity, that is, an alignment discovered by some program, means anything? Is it statistically significant, is it truly homologous, and even more importantly, does it have anything to do with real biology? Many programs generate percent similarity scores; however, these really don't mean a whole lot. Do not use percent similarities or identities to compare sequences except in the roughest way. They are not optimized or normalized in any manner. The raw 'quality' score means a lot more, but is difficult to interpret. At least it takes the length of similarity, all of the necessary gaps introduced, and the matching of symbols all into account, but quality scores are only relevant within the context of a particular comparison or search. The quality ratio is the metric optimized by dynamic programming divided by the length of the shorter sequence. As such it represents a fairer comparison metric, but it also is relative to the particular scoring matrix and gap penalties used in the procedure. Some of the programs can generate histograms of score distributions, but again, they can be

confusing. To get a better handle on what these various scores mean, read about the algorithms in available textbooks, the primary literature, and the GCG Program Manual — statistics can be confusing, but it will help.

A traditional way of deciding alignment significance relies on an old statistics trick — Monte Carlo simulations. This type of significance estimation has implicit statistical problems; however, few practical alternatives exist for just comparing two sequences. Monte Carlo techniques continue to be used because of their ease and speed and will remain important in the field for a long time. Monte Carlo methods compare an actual score, in this case the quality score of an alignment, against the distribution of scores of alignments of a randomized sequence. Therefore, one way of deciding alignment significance is to take advantage of the Monte Carlo style randomizations option available in the two GCG dynamic programming pairwise comparison programs BestFit and Gap. To utilize this strategy, compare two sequences using the appropriate algorithm, either Gap or BestFit depending on whether you're trying to compare the entire length of each sequence or only the best regions of similarity of each, respectively, and specify the command option "–Randomizations=100". This option jumbles the second sequence of the comparison 100 times after the initial alignment is produced and then generates scores and a standard deviation based on the jumbled matches. Comparing the quality scores of the randomized alignments to the initial alignment can help give a feeling for the relative meaning of the scores. You can compare the mean of the random scores to the unjumbled score using a 'Z score' calculation to help decide significance. An old 'rule-of-thumb' that people often use is, if the actual score is much more than three standard deviations above the mean of the randomized scores, the analysis may be significant; if it is much more than five, than it probably is significant; and if it is above ten, than it definitely is significant. Many Z scores measure the distance from a mean using this simplistic Monte Carlo model assuming a normal distribution, in spite of the fact that 'sequence-space' actually follows what is know as an 'extreme value distribution;' however, the method does approximate significance estimates quite well and is calculated with the following formula:

$$\text{Z score} = \frac{[\,(\text{actual score}) - (\text{mean of randomized scores})\,]}{(\text{standard deviation of randomized score distribution})}$$

When the two TATA sequences from the previous dynamic programming example are compared to one another using the same scoring parameters as before, but incorporating a Monte Carlo Z score calculation, their similarity is found, surprisingly, not to be at all significant. It is merely a reflection of the compositional bias of the two sequences to contain lots of T's and A's. Those results follow:

> Average quality based on 100 randomizations: 41.8 +/– 7.4. Plugged into the formula: (50 – 41.8) / 7.4 = 1.11, i.e. there is no significance to the match!

The Fast and SSearch (Pearson and Lipman, 1988, and Pearson, 1998), BLAST (Altschul, et al., 1990 and 1997), HMMerSearch (Eddy, 1996 and 1998), and ProfileSearch (Gribskov, et al., 1987) algorithms use a similar approach but base their statistics on the distance of the query matches from the actual, or a simulated (in the case of BLAST), extreme value distribution from the rest of the, 'insignificantly similar,' members of the database searched. BLAST, Fast, SSearch, and HMMerSearch generate Expectation functions in this manner. ProfileSearch returns Z scores, which follow the same guidelines as described above. Expectation values are

printed in scientific notation and the smaller the number, i.e. the closer it is to 0, the more significant the match is. Expectation values show us how often we could expect that particular alignment match to occur merely by chance alone in a search of that size database. In all cases, these are the numbers to pay attention to, not the raw 'scores.'

Rough, conservative guidelines to Z scores and Expectation values from a typical search:

~Z score	~E value	Inference
≤3	≥0.1	little, if any evidence for homology
≅5	$\cong 10^{-2}$	probably homologous, but may be due to convergent evolution
≥10	$\leq 10^{-3}$	definitely homologous

Be very careful with any guidelines such as these, though, because they are entirely dependent on both the size and content of the database being searched as well as how often you perform the search! Think about it — the odds are way different for rolling a "Yahtzee" depending on how many dice you roll, whether they are 'loaded' or not, and how often you try. That's the way probabilities work.

Another very powerful empirical method of determining significance is to repeat a database search with the entry in question. If that entry finds more significant 'hits' with the same sorts of sequences as the original search, then the entry in question is undoubtedly homologous to the original entry. If it finds entirely different types of sequences, then it probably is not. That is, homology is transitive. Modular proteins with distinctly separate domains confuse issues considerably, but the principles remain the same, and can be explained through domain swapping and other examples of non-vertical transmission. And, finally, the 'Gold-standard' of homology is shared structural folds — if you can demonstrate that two proteins have the same structural fold, then, regardless of similarity, at least that particular domain is homologous between the two.

Database searching algorithms use all these concepts; however, classic dynamic programming techniques take far too long when used against most databases with a 'normal' computer. Therefore, the programs use tricks to make things happen faster. These tricks fall into two main categories, hashing and approximation. Hashing is the process of breaking your sequence into small 'words' or 'ktuples' of a set size and creating a 'look-up' table with those words keyed to numbers. Then when any of the words match part of an entry in the database, that match is saved. In general, hashing reduces the complexity of the search problem from N^2 for dynamic programming to N, the length of all the sequences in the database. Approximation techniques are collectively known as 'heuristics.' Webster's defines heuristic as "serving to guide, discover, or reveal; . . . but unproved or incapable of proof." In database searching algorithms the heuristic usually restricts the necessary search space by calculating some sort of a statistic that allows the program to decide whether further scrutiny of a particular match should be pursued. This statistic may miss things depending on the parameters set — that's what makes it heuristic. The exact implementation varies between the different programs, but the basic idea follows in most all of them.

Two predominant versions exist: the Fast and BLAST programs. Both return local alignments. Both are not a single program, but rather a family of programs with implementations designed to compare a sequence to a

database in about every which way imaginable; the acronyms run amuck. Comprehensively: a DNA sequence against a DNA database, as implemented by FastA and BLASTN (not recommended unless forced to do so because you are dealing with a non-translated region of the genome); a translated (where the translation is done 'on-the-fly' in all six frames) version of a DNA sequence against a translated ('on-the-fly') version of the DNA database, as done by TBLASTX (not available in the Fast program family); a translated ('on-the-fly') version of a DNA sequence against a protein database, this is FastX and BLASTX; a protein sequence against a translated ('on-the-fly') version of the DNA database, as TFastA and TFastX and TBLASTN do; or a protein sequence against a protein database, FastA again and BLASTP. The TFastX implementation allows for the recognition of frame shifts in translated comparisons. In more detail:

FastA and Family — University of Virginia (Pearson and Lipman, 1988, and Pearson, 1998)

1) Works well for DNA against DNA searches (within limits of possible sensitivity);
2) Can find only one gapped region of similarity;
3) Relatively slow, should usually be run in the background;
4) Does not require specially prepared, preformatted databases.

FastA is an older algorithm than BLAST. It was the first widely used, powerful sequence database searching program. Pearson continually refines the algorithm such that it remains a viable alternative to BLAST, especially if one is restricted to searching DNA against DNA without translation. It is also very helpful in situations where BLAST searches find no significant alignments — arguably, FastA may be more sensitive than BLAST in these situations. Furthermore, it's flexible database requirements make it a great way to search subsets of data.

The algorithm:

FastA builds words of a set k-tuple size, by default two for peptides. It then identifies all exact word matches between the sequence and the database members. Scores are assigned to each continuous, ungapped, diagonal by adding all of the exact match BLOSUM values. The ten highest scoring diagonals for each query-database pair are then rescored using BLOSUM similarities as well as identities and ends are trimmed to maximize the score. The best of each of these is called the *Init1* score. Next the program 'looks' around to see if nearby off-diagonal *Init1* alignments can be combined by incorporating gaps. If so, a new score, *Initn*, is calculated by summing up all the contributing *Init1* scores, penalizing gaps with a penalty for each. The program then constructs an optimal local alignment for all *Initn* pairs with scores better than some set threshold using a variation of dynamic programming "in a band." A sixteen-residue band centered at the highest *Init1* region is used by default with peptides. A score is generated from this step known as the *opt* score. Next, FastA uses a simple linear regression against the natural log of the search set sequence length to calculate a normalized *z-score* for the sequence pair. Finally, it compares the distribution of these *z-scores* to the actual extreme-value distribution of the search. Using this distribution, the program estimates the number of sequences that would be expected to have, purely by chance, a *z-score* greater than or equal to the *z-score* obtained in the search. This is reported as the Expectation value. Unfortunately, the *z-score* used in FastA and the previously discussed Monte Carlo style Z

score are quite different and can not be directly compared (see William R. Pearson, *Protein Science* **4**; 1145-1160 [1995] for an explanation of how this z-score is calculated). If the user requests pairwise alignments in the output, then the program uses full Smith-Waterman local dynamic programming, not 'restricted to a band,' to produce its final alignments.

BLAST — Basic Local Alignment Search Tool — NCBI (Altschul et al. 1990 and 1997)

1) Normally not a good idea to use for DNA against DNA searches (not optimized);
2) Prefilters repeat and "low complexity" sequence regions by default;
4) Can find more than one region of gapped similarity;
5) Very fast heuristic and parallel implementation;
6) Restricted to precompiled, specially formatted databases;

The algorithm:

After BLAST has sorted its lookup table, it tries to find all double word hits along the same diagonal (see the dot matrix graphs below) within some specified distance using what NCBI (National Center for Biotechnological Information, the division of the National Library of Medicine [NLM], at the National Institute of Health [NIH] responsible for maintaining GenBank and providing worldwide access to sequence analysis resources and software) calls a Discrete Finite Automaton (DFA). These word hits of size *W* do not have to be identical; rather, they have to be better than some threshold value *T*. To identify these double word hits, the DFA scans through all strings of words (typically *W*=3 for peptides) that score at least *T* (usually 11 for peptides). Each double word hit that passes this step then triggers a process called ungapped extension in both directions, such that each diagonal is extended as far as it can, until the running score starts to drop below a predefined value *X* within a certain range *A*. The result of this pass is called a High-Scoring segment Pair or HSP.

Those HSPs that pass this step with a score better than *S* then begin a gapped extension step utilizing dynamic programming. Those gapped alignments with Expectation values better than the user specified cutoff are reported. The extreme value distribution of BLAST Expectation values is precomputed against each precompiled database — this is one area that speeds up the algorithm considerably.

The math can be generalized thus: for any two sequences of length *m* and *n*, local, best alignments are identified as HSPs. HSPs are stretches of sequence pairs that cannot be further improved by extension or trimming, as described above. For ungapped alignments, the number of expected HSPs with a score of at least *S* is given by the formula: $E = Kmne^{-\lambda s}$

This is called an *E*-value for the score *S*. In a database search *n* is the size of the database in residues, so *N*=*mn* is the search space size. K and λ are be supplied by statistical theory, and, as mentioned above, can be calculated by comparison to precomputed, simulated distributions. These two parameters define the statistical significance of an *E*-value. The *E*-value defines the significance of the search. As mentioned above, the smaller

an *E*-value is, the more likely it is significant. A value of 0.001 to 0.010 is a good starting point for significance in most typical searches. In other words, in order to assess whether a given alignment constitutes evidence for homology, it helps to know whether the alignment can be expected by chance alone.

In review, both the Fast and BLAST family of programs base their Expectation "*E*" values on a more realistic 'extreme value distribution,' based on either real or simulated 'not significantly similar' database alignments, than Monte Carlo style Z scores do. Regardless, they follow Monte Carlo style Z scores fairly well. The higher the *E* value is, the more probable that the observed match is due to chance in a search of the same size database and the lower its Z score will be, i.e. is not significant. Therefore, the smaller the *E* value, i.e. the closer it is to zero, the more significant it is and the higher its Z score will be! The *E* value is the number that really matters.

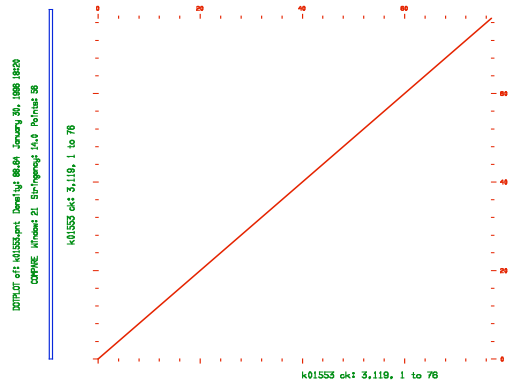
Furthermore, all database searching, regardless of the algorithm used, is far more sensitive at the amino acid level than at the DNA level. This is because proteins have twenty match criteria versus DNA's four and those four DNA bases can only be identical, not similar, to each other; and many DNA base changes (especially third position changes) do not change the encoded protein. All of these factors drastically increases the 'noise' level of a DNA against DNA search, and gives protein searches a much greater 'look-back' time, typically at least doubling it. Therefore, whenever dealing with coding sequence, it is always prudent to search at the protein level. Even though protein searching is more sensitive, the DNA databases are larger. This can be overcome with programs that take a protein query and compare it to translated nucleotide databases, but one still needs to know if the translation is 'real.' The general rule is to query with a peptide sequence, if at all possible, and screen whichever databases you choose.

Another powerful method that should always be considered in similarity analysis is the dot matrix procedure. In dot matrix analysis one sequence is plotted on the vertical axis against another on the horizontal axis using a very simple approach; a dot is generated wherever they match, according to some scoring system that you specify. Dot matrix analysis can point out areas of similarity between two sequences that all other methods might miss. This is because most other methods align either the overall length of two sequences or just the 'best' parts of each to achieve one optimal alignment. Dot matrix methods enable the operator to visualize the entirety of both sequences; if you will, they allow the 'Gestalt' of the alignment to be seen. Because your own mind and eyes are still better than computers at discerning complex visual patterns, especially when more than one pattern is being considered, dot matrix analysis can be extremely powerful. However, their interpretation is entirely up to the user — you must know what the plots mean and how to successfully filter out extraneous background noise when running them. Using this method correctly you can identify areas within sequences with significant matches that no other method would ever notice.

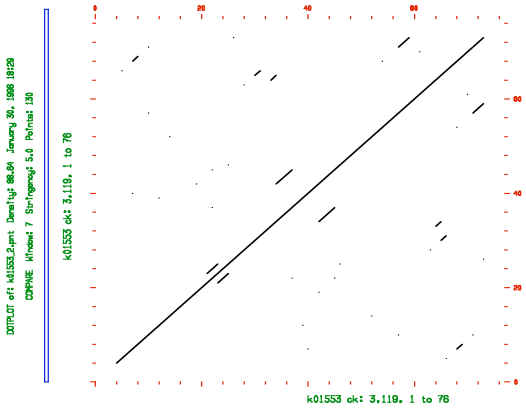
The Wisconsin package's implementation of dot matrix analysis, the paired programs Compare and DotPlot, use a window/stringency method by default. You need to be very careful with these programs as the default window size and stringency (14 in a window of 21 for nucleic acid sequences) may not be appropriate for the analysis at hand. Consider the following set of examples from the phenylalanine transfer RNA molecule from yeast, GenBank:K01553. The sequence and structure are both known for this molecule and this illustration will show

how simple dot-matrix procedures can quickly lead to functional and structural insights (even without complex folding algorithms).

a) If you run the programs at their default settings, the dotplot from a comparison of this sequence with itself is quite uninformative, only showing the main identity diagonal:

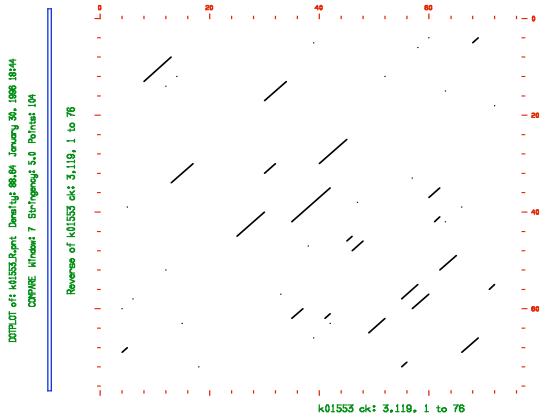


b) However, if you adjust the window size down to find finer features, some elements of symmetry become apparent. Here I changed the window size to 7 and the stringency to 5. (As a general guide to stringency levels, pick whatever window size is most appropriate for the analysis at hand, e.g. about the size of the feature that you are trying to recognize, and then choose a stringency that produces a point file with the number of points found to be of a similar order of magnitude as that of the length of your longest sequence.)



Several direct repeats are now obvious as off-diagonal alignment segments that remained obscured in the previous analysis.

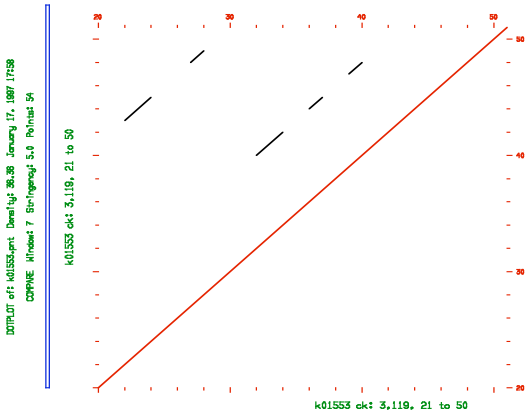
c) When dealing with RNA/DNA, even more insight can be gained by comparing the reverse, complement of a sequence to itself. This is easy in GCG by reversing the specified sequence. (In this context, the GCG –Reverse option is the reverse, complement of a sequence, not just the reverse since just the reverse is not biologically relevant.) In SeqLab use the “Edit” and then “Reverse” button. Compare the following dotplot to the previous ones; here the yeast tRNA sequence is compared to its reverse, complement using the same 5 out of 7 stringency setting:



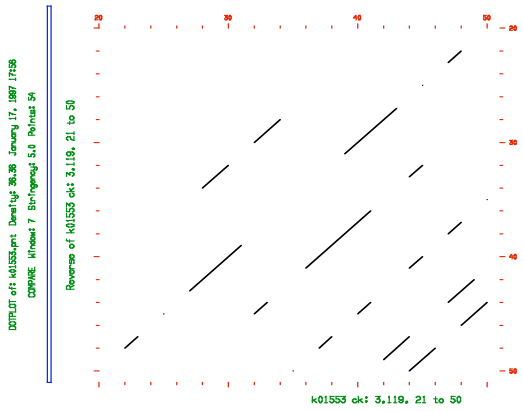
Now the potential for inverted repeats becomes obvious; these are the well-characterized stem-loop structures of the tRNA cloverleaf molecular shape. They appear as clearly delineated diagonals. These diagonals are now perpendicular to an imaginary main diagonal running opposite to the previous case, since we reversed the orientation of the second sequence. Take for instance the middle stem; the region of the molecule centered at approximately base number 38 has a clear propensity to base pair with itself without creating a loop since it crosses the main diagonal and then just after a small unpaired gap another stem is formed between the region

from about base number 24 through 30 with 46 through 40.

d) That same region 'zoomed in on' has some small direct repeats that can be seen when you use the Compare "–All" option on the sequence against itself without reversal. This is illustrated below:



e) But looking at the same region of the sequence against its reverse-complement shows a wealth of potential stem-loop structure in the tRNA (again with "–All"):

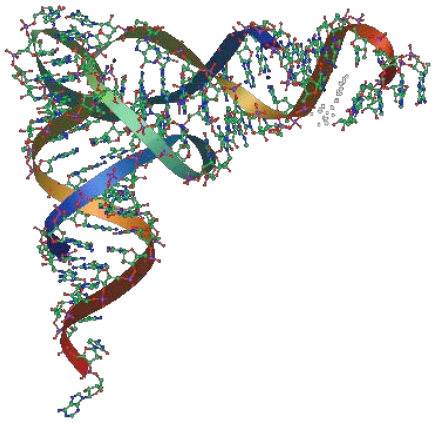


f) The GCG program StemLoop can show some of these match-ups base by base. Depending on what parameters you use, this is one:

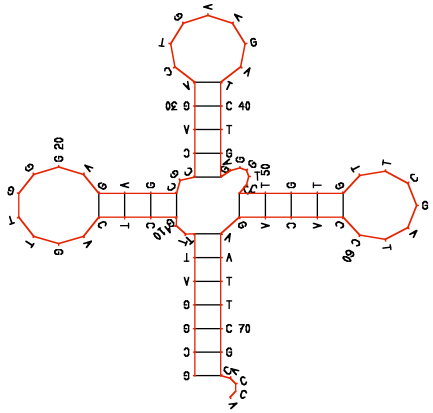
```
22 GAGCGCCAGACT  G    12, 22
   || | ||||| |   A
48 CTGGAGGTCTAG  A    3
```

So, as noted above, the region of K01553 from base position 22 through position 33 base pairs with (think — is quite similar to the reverse-complement of) itself from base position 37 through position 48. Got it?

g) The yeast phenylalanine 3D tRNA model as solved by Sundaralingam et al. (1976), deposited in the PDB under access code 1TRA, is shown below. The stem above corresponds to the right-most stem below:



h) A GCG 'Squiggle' plot of this same molecule from the output of MFold, Zuker's (1989) RNA folding algorithm that uses base pairing energy minimization to find the family of optimal and suboptimal structures, shows the most stable structure with a stem at positions 27 to 31 with 39 to 43. However the region around position 38 is represented as a loop. Reality, as modeled above, is seen to lie somewhere in between these two interpretations, but the simple dotplot analysis quickly provided valuable insight.



Another very powerful approach that should always be used, if possible, is the Profile suite (Gribskov, et al., 1987), and the statistically driven, Hidden Markov Model based enhancements developed subsequently (see e.g. HMMer, Eddy, 1996 and 1998). To use this strategy you need to prepare and refine a multiple sequence alignment of significantly similar sequences or domains first. Profile searching then involves forming a "profile" from this set of related sequences and searching the databases with that profile. A 'motif' style profile library has also been prepared by Gribskov and made available within the GCG system. The program ProfileScan searches your query protein sequence against this library. Similarly, a library of HMMer profiles, the Pfam library (A database of protein domain family alignments and HMMs © 1996-2000 The Pfam Consortium) is available for searching by the program HMMerPfam.

Profile searching is tremendously powerful and will find remote homologues that no other method will discover. The popular psi-BLAST program at NCBI (Altschul, et al., 1997) is an automated, somewhat crude implementation of the same idea. A very appropriate strategy is to find similar genes to a newly sequenced gene using traditional database searching techniques, and then align all of the significantly similar proteins or protein domains using a multiple sequence alignment program such as PileUp or ClustalW. Then you can generate a profile of the family to create a very sensitive probe for further analysis.

Often profile analysis can show features not obvious to individual members. A distinct advantage is evolutionary issues are considered in further manipulations and database searches, by virtue of the profile algorithms. Gaps are penalized more heavily in conserved areas than they are in variable regions, and the more highly conserved a residue is, the more important it becomes. Generated consensus sequences are not based merely on the positional frequency of particular residues but rather utilize the evolutionary conservation of substitutions based on the BLOSUM62 (Henikoff and Henikoff, 1992) table (by default, other substitution matrices can also be specified).

Most database searches work best when submitted as a batch or background process. This is because of the sizes of the databases involved. The October 2006 release of GenBank has 67 billion bases, and GenBank doubles in size nearly every year! In spite of the fast hashing, heuristic style algorithms incorporated, most programs can take quite a while to search through that much data. There is no way you want to wait in front of a unusable terminal while the computer cranks away comparing your query to that many sequences, therefore, take advantage of batch and/or background capabilities. All of the GCG database searches accept an automatic batch submission option from the command line, or you can run background searches while in SeqLab. An exception to the standard 'submit the search and wait' style of most database searching is the NetBLAST program. This program uses the same fast heuristic, statistical hashing algorithm as GCG's local BLAST program but it runs on a very fast parallel computer system located at NCBI. Typical searches run in just a few minutes, after you get through the waiting queue; however, realize that this algorithm (all forms of BLAST) is optimized for protein comparisons only. With its default parameters it is not at all appropriate for DNA to DNA comparisons without going through translation steps 'on the fly.' It will find closely similar DNA sequences, but will not be able to locate sequences that are only somewhat similar. If you are dealing with a nontranslated locus and are forced to compare a DNA query against a DNA database without translation, then use FastA instead of BLAST; it is the more appropriate tool.

Database searching: A 'real-life,' project oriented tutorial. How and where do we start?

I write these tutorials from a 'lowest-common-denominator' biologist's perspective. That is, I only assume that you have fundamental molecular biology knowledge, but are relatively inexperienced regarding computers. As a consequence of this they are written quite explicitly. Therefore, if you do exactly what is written, it will work. However, this requires two things: 1) you must read very carefully and not skim over vital steps, and 2) you mustn't take offense if you already know what I'm discussing. I'm not insulting your intelligence. This also makes the tutorials longer than otherwise necessary. Sorry.

I use **bold** type for those commands and keystrokes that you are to type in at your console or for buttons that you are to click in SeqLab. I also use bold type for **section headings**. Screen traces are shown in a "`typewriter`" style Courier font and "`//////////`" indicates abridged data. The arrow symbol, ">" indicates the system prompt and should not be typed as a part of commands. Really important statements may be underlined.

SeqLab is a part of the Genetics Computer Group's (GCG) Wisconsin Package. This comprehensive package of sequence analysis programs is used worldwide. It has become the global standard in sequence analysis software. The Wisconsin Package provides a comprehensive toolkit of around 150 integrated DNA and protein analysis programs, from database, pattern, and motif searching; fragment assembly; mapping and sequence comparison to gene finding; protein and evolutionary analysis; primer selection; and DNA and RNA secondary structure prediction. The SeqLab X-windows based Graphical User Interface (GUI) is a 'front-end' to the package. It provides an intuitive alternative to the UNIX command line by allowing menu-driven access to most of GCG's programs. It is based on Steve Smith's (1994) GDE (the Genetic Data Environment) and makes running the Wisconsin Package much easier by providing a common editing interface from which most programs can be launched. This workshop will show you how to use SeqLab to investigate pairwise sequence similarity in a number of ways. Once you gain an appreciation for SeqLab's power and ease of use, I don't think you'll be satisfied with any other sequence analysis system.

Specialized "X-server" graphics communications software is required to use GCG's SeqLab interface. This needs to be installed separately on personal style 'Wintel' or Macintosh machines but comes standard with UNIX operating systems. The details of X and of connecting to the GCG server on campus (mendel.scs.fsu.edu) will not be covered in this workshop. If you are unsure of these procedures ask for assistance in the computer laboratory. I am also available for individualized personal help in your own laboratories. If you are having difficulties connecting to the GCG server from there, just contact me at stevet@bio.fsu.edu. A couple of tips at this point should be mentioned though. X-windows are only active when the mouse cursor is in that window. Furthermore, rather than holding mouse buttons down, to activate items, just click on them; and buttons are turned on when they are pushed in and shaded. Also, do not close windows with the X-server software's close icon in the upper right- or left-hand window corner, rather, always use GCG's "Close" or "Cancel" or "OK" button, usually at the bottom of the window.

The protein example used in this tutorial is involved in Alzheimer's Disease. The Alzheimer's peptide is a small portion of a nexin II type protease present in normal organisms. The intact protein is a neuronal receptor that

couples to intracellular signaling pathways through the GTP-binding protein G(0). Some isoforms also have clathrin-binding and BPTI/Kunitz type protease inhibitor domains. However, for unknown reasons in individuals with Alzheimer's and in some older Down's syndrome cases, aberrant catabolism of the precursor creates the small insoluble, pathological protein known as the beta-amyloid protein. This small protein product comes from part of the transmembrane and extracellular domains of the intact protein. Several different mutations have been identified in the gene of afflicted individuals; however, this is only the case in a small fraction of all Alzheimer's patients. It is a very complex system as at least six different alternatively spliced forms of the protein are naturally found in Human beings, all coded by nineteen exons located on the long arm of chromosome 21 at map position 21q21.3 (molecular biology review from NCBI OMIM, 1994, and Gene, 2005, databases). In this tutorial you will perform all of the analyses with the intact protein and various small subset databases so that the entire exercise can be finished in real time within one workshop setting.

1) Log onto your GCG account and launch SeqLab

Each participant in the session should use a different UNIX account. Either login with your existing account and password or the new one supplied to you at the beginning of the workshop. Use the appropriate connection commands on the personal computer or terminal that you are sitting at to launch X and log onto the UNIX host computer that runs GCG at your site. An X-style terminal window should appear on the desktop after a few moments, if it doesn't, launch one with the appropriate command. Get my assistance for this step if you are unsure of yourself. There are too many variations in method for them all to be described here. In general:

Log on to the campus GCG server Mendel with an X tunneled ssh terminal connection (that's a capital X!):

```
> ssh -X user@mendel.scs.fsu.edu
```

The GCG package should initialize automatically as soon as your terminal window launches. If it didn't, type the command "**gcg**" (without the quotes) at the system prompt in the terminal window to start it up now. This process activates all of the programs within the package and displays the current version of both the software and all of its accompanying databases.

Issue the GCG command "**typedata sw:a4_Human**" in your terminal window and 'pipe' it through "**more**" so you can read the text a page at a time to see what this beta amyloid protein entry looks like. The command line follows:

```
> typedata sw:a4_Human | more
```

Use the 'space' bar not the 'return' key to proceed from one page to the next while using the UNIX more utility.

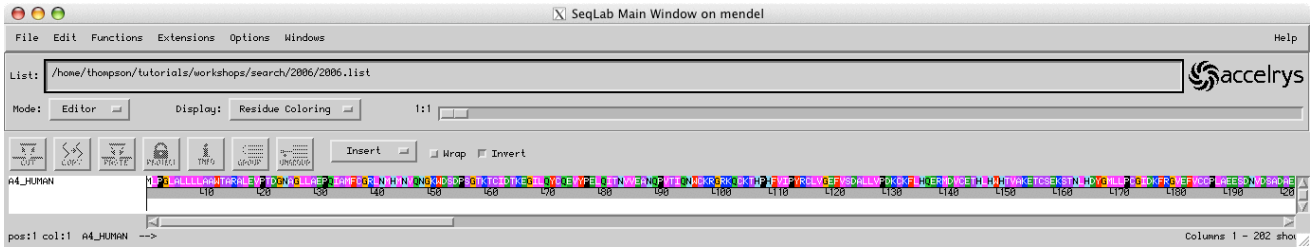
Now issue the command "**seqlab &**" to fire up the SeqLab interface. The ampersand, "**&**," is not necessary but really helps out by launching SeqLab as a background process so that you can retain control of your initial terminal window:

```
> seqlab &
```

This should produce two new windows, the first an introduction with an “OK” box; check “**OK**.” You should now be in SeqLab’s List mode. Before beginning the analyses, go to the “**Options**” menu and select “**Preferences . . .**” There’s no need to do this if you’ve already checked your settings in an earlier session. Otherwise, we should check a few options to insure that SeqLab runs its most intuitive manner. The defaults are usually fine, but I want you to see what’s available to change. Remember, buttons are turned on when they’re pushed in and shaded.

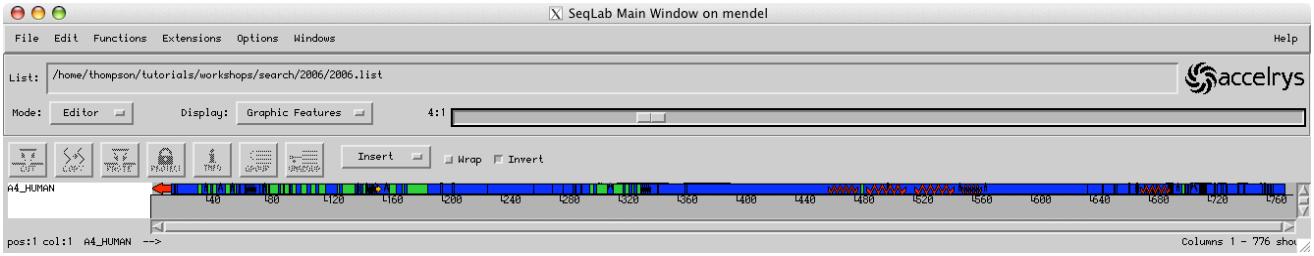
First notice that there are three different “**Preferences**” settings that can be changed: “**General**,” “**Output**,” and “**Fonts**;” start with “**General**.” The “**Working Dir. . .**” setting will be the directory from which SeqLab was initially launched. This is where all SeqLab’s working files will be stored; it can be changed in your accounts if desired, however, it is appropriate to leave it as is for now. Be sure that the “**Start SeqLab in:**” choice has “**Main List**” selected and that “**Close the window**” is selected under the “**After I push the “Run” button**:” choice. Next select the “**Output**” “**Preference**.” Be sure “**Automatically display new output**” is selected. Finally, take a look at the “**Fonts**” menu. We will leave all these choices as is but I want to point out that if you are dealing with very large alignments, then picking a smaller Editor font point size may be desirable in order to see more of your alignment on the screen at once. Click “**OK**” to accept any changes.

Be sure the “**Mode:**” “**Main List**” choice is selected in your main window and then go to the “**File**” menu. Pick “**Add sequences from**” and select “**Databases. . ..**” This will produce a “**SeqLab Database Browser**” window from which you can specify sequences to add to your “**`working.list`**.” Type the entry that we want, “**sw:a4_Human**,” in the “**Database Specification:**” box and then press the “**Show Matching Entries**” button. Select the sequence displayed, and then punch the “**Add to Main Window**” button, and then the “**Close**” button at the bottom of the window, to put the sequence in your “**`working.list`**.” It will appear in the SeqLab “**Main List**” window. Be sure it is selected and then switch to “**Editor**” “**Mode:**” to load the sequence into the SeqLab editor. Notice that the sequence now appears in the editor window with the residues color-coded. Any portion of or the entire sequence loaded is now available for analysis by any of the appropriate GCG programs. Drag the window to an appropriate size by ‘grabbing’ the bottom-right corner of its ‘frame’ and ‘pulling’ it out as far as desired. The display will look something like the following graphic:



Explore the editor interface for a while. Nearly all GCG programs are accessible through the "Functions" menu. The scroll bar at the bottom allows you to move through the sequences linearly. You can select the sequence or any position(s) within it by ‘capturing’ them with the mouse. Quickly double-click the sequence’s name (or click on the “**INFO**” icon with its name selected) to get its full database reference documentation and then “**Close**” the “**Sequence Information**” window. The “**pos:**” and “**col:**” indicators show you where the cursor is located on the sequence without including and with including gaps respectively. The “**1:1**” scroll bar near the upper right-hand

corner allows you to 'zoom' in or out on the sequences; move it to "**2:1**" and beyond and notice the difference in the display. Go to the "**Display:**" box and change it from "**Residue Coloring**" to "**Feature Coloring**." The colors are now based on the information from the database Feature Table for the entry. Change the "**Display:**" to "**Graphic Features**;" now the features are represented using the same colors as before but in a 'cartoon' fashion. Use the mouse to move your cursor to one of the colored areas; quickly double-click it (or use the "Features" selection under the "Windows" menu). This will produce a new window that describes the features located at the cursor. Click on one of the features to get more information on it and to select that feature in its entirety. All the features are fully editable through the "Edit" check box in this panel and new features can be added with several desired shapes and colors through the "Add" check box. The display will look something like my example shown below at a 4:1 zoom ratio so that the entire sequence can be seen at once:



2) Searching the PROSITE database. A 'quick and dirty' method — GCG's Motifs search of PROSITE

Many, many features have been described and catalogued in sequences over the years. Many of these have recognizable consensus patterns that allow you to screen an unknown sequence for their occurrence. This database of catalogued structural, regulatory, and enzymatic consensus patterns is Amos Bairoch's protein signature database, the *PROSITE Dictionary of Protein Sites and Patterns* (1992). It is one of the quickest and easiest databases to search with a peptide sequence. The GCG program Motifs performs this search. The program can tolerate mismatches with a mismatch option and it displays an abstract with selected references for each motif signature found. In many cases this can be a tremendous aid in ascertaining the function of an unknown peptide sequence, though motifs can be false positive and may not have anything to do with function at all. It can often lead to immediate answers and routes of investigation. Regardless, it should always be utilized — it's just too fast and simple to ignore.

Start the Motifs program by selecting the A4 protein entry's name in SeqLab and then going to the "**Functions**" "**Protein Analysis**" menu and picking "**Motifs. . ..**" A "**Which selection**" window may pop up asking if you want to use the "**selected sequences**" or "**selected regions**;" choose "**selected sequences**" to run the program on the full A4 sequence. Check the "**Save results as features in file motifs.rsf**" button. We'll use this file a little further below to superimpose the newfound Motifs features on our sequence. We don't need any of the other options so press the "**Run**" button. After a few moments you should get output. The file displayed, "**`motifs.rsf`**," isn't very interesting to read so "**Close**" it and use the "**Output Manager**" to display the file with the "**`.motifs`**" extension. Carefully look over the text file that is displayed. Notice the sites that have been characterized in this sequence and the extensive bibliography associated with them:

MOTIFS from: /home/thompson/.seqlab/input_84.rsf{A4_HUMAN}

```
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
A4_Extra              G(V,T)E(F,Y)VCCP
                        G(V)E(F)VCCP
           181: IDKFR     GVEFVCCP     LAEES
```

```
**************************************
* Amyloidogenic glycoprotein signatures *
**************************************
```

Amyloidogenic glycoprotein (A4 protein or APP) is an integral, glycosylated membrane brain protein [1,2,E1,E2]. APP is associated with Alzheimer's disease (AD). This responsibility stems from the fact that a small peptide (of 43 residues), called the amyloid beta protein, which is part of the sequence of A4, is the major constituent of amyloid deposits in AD and in Down's syndrome. As shown in the schematic representation below, the amyloid beta protein both precedes and forms part of the unique transmembrane region of A4.

```
       +---------------------------------------xxxxxxx-------------+
       |  Extracellular                         XXXXXXX Cytoplasmic |
       +-----------------------------------BBBBBBBBxxx-------------+
```

'X': Transmembrane region.
'B': Position of the amyloid beta protein in A4.

The exact function of A4 protein is not yet known, but it has been suggested that it mediates cell-cell interactions. The sequence of A4 from mammalian species is well conserved and is also similar to that of other proteins:

- Drosophila APPL (gene vnd) [3].
- Mammalian protein APLP1 [4].
- Mammalian protein APLP2 (APPH) (YWK-II) (CDEI-binding protein) [5].

We have derived two patterns specific to these proteins, the first one is a perfectly conserved octapeptide located in the beginning of the extracellular domain; the second is a conserved octapeptide located at the C-terminal end of the cytoplasmic domain.

-Consensus pattern: G-[VT]-E-[FY]-V-C-C-P
-Sequences known to belong to this class detected by the pattern: ALL.
-Other sequence(s) detected in SWISS-PROT: NONE.

-Consensus pattern: G-Y-E-N-P-T-Y-[KR]
-Sequences known to belong to this class detected by the pattern: ALL.
-Other sequence(s) detected in SWISS-PROT: NONE.

-Last update: July 1998 / Text revised.

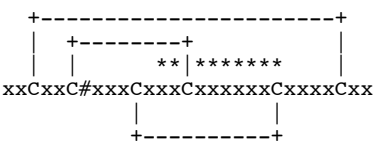
[1] Dyrks T., Weidemann A., Multhaup G., Salbaum J.M., Lemaire H.-G., Kang J., Muller-Hill B., Masters C.L., Beyreuther K.
EMBO J. 7:949-957(1988).
[2] Ashall F., Goate A.M.
Trends Biochem. Sci. 19:42-45(1994).
[3] Rosen D.R., Martin-Morris L., Luo L., White K.
Proc. Natl. Acad. Sci. U.S.A. 86:2478-2482(1989).
[4] Wasco W., Bupp K., Magendantz M., Gusella J.F., Tanzi R.E., Solomon F.
Proc. Natl. Acad. Sci. U.S.A. 87:10758-10762(1992).
[5] Sprecher C.A., Grant F.J., Grimm G., O'Hara P.J., Norris F., Norris K., Foster D.C.
Biochemistry 32:4481-4486(1993).
[E1] http://werple.mira.net.au/~dhs/adapp.html
[E2] http://bioinformatics.weizmann.ac.il/hotmolecbase/entries/app.htm


```
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
Bpti_Kunitz           Fx3GCx6(F,Y)x5C
                      Fx{3}GCx{6}(F)x{5}C
           319: GKCAP FFYGGCGGNRNNFDTEEYC MAVCG
```

```
******************************************************
* Pancreatic trypsin inhibitor (Kunitz) family signature *
******************************************************
```

The pancreatic trypsin inhibitor (Kunitz) family [1,2,3] is one of the numerous families of serine proteinase inhibitors. The basic structure of such a type of inhibitor is shown in the following schematic representation:



```
<------50 residues------>

'C': conserved cysteine involved in a disulfide bond.
'#': active site residue.
'*': position of the pattern.

In addition to the prototype sequence  for this type of inhibitor - the bovine
pancreatic  trypsin  inhibitor (BPTI)  (also known as basic protease inhibitor
(BPI)) -  this family also includes many other members which are  listed below
(references are only provided for recently determined sequences):

 - Mammalian  inter-alpha-trypsin  inhibitors  (ITI).    ITI's   contain   two
   inhibitory domains.
 - Tissue  factor  pathway  inhibitor  precursor  (TFPI)  (previously known as
   lipoprotein-associated coagulation inhibitor (LACI)), which inhibits factor
   X (Xa) directly and, in a Xa-dependent way,  inhibits VIIa / Tissue  factor
   activity. TFPI contains three inhibitory domains.
 - TFPI-2  [4]  (also  known  as placental protein 5), a protein that contains
   two inhibitory domains.
 - Bovine colostrum, serum and spleen trypsin inhibitors.
 - Trypstatin, a rat mast cell inhibitor of trypsin.
 - A number of venom basic protease  inhibitors  (including dendrotoxins) from
   snakes.
 - Isoinhibitor K from garden snail.
 - Protease inhibitor from the hemocytes of horseshoe crab.
 - Basic protease inhibitor from red sea turtle.
 - Sea anemone protease inhibitor 5 II.
 - Chymotrypsin inhibitors SCI-I,- II, and -III from silk moth.
 - Trypsin inhibitors A and B from the hemolymph of the tobacco hornworm.
 - Trypsin inhibitor from the hemolymph of the flesh fly [5].
 - Acrosin inhibitor from the male accessory gland of Drosophila.

 - A domain found in  one of the alternatively spliced  forms  of  Alzheimer's
   amyloid beta-protein (APP) (also known as protease nexin II) as well as the
   closely related amyloid-like protein 2 (or APPH).
 - A domain at the C-terminal  extremity  of  the  alpha(3)  chain  of type VI
   collagen.
 - A domain at the C-terminal  extremity  of  the  alpha(1)  chain of type VII
   collagen.

We developed a  pattern  which  will only  pick up sequences belonging to this
family of  inhibitors. It spans a region starting after the third cysteine and
ending with the fifth one.

-Consensus pattern: F-x(3)-G-C-x(6)-[FY]-x(5)-C
                    [The two C's are involved in disulfide bonds]
-Sequences known to belong to this class detected by the pattern: ALL,  except
 for trypsin inhibitor IV from the sea anemone Radianthus macrodactylus  which
 has Asp instead of Phe/Tyr.
-Other sequence(s) detected in SWISS-PROT: NONE.

-Expert(s) to contact by email:
           Ikeo K.; kikeo@genes.nig.ac.jp

-Last update: November 1997 / Text revised.

[ 1] Laskowski M., Kato I.
     Annu. Rev. Biochem. 49:593-626(1980).
[ 2] Salier J.-P.
     Trends Biochem. Sci. 15:435-439(1990).
[ 3] Ikeo K., Takahashi K., Gojobori T.
     J. Mol. Evol. 34:536-543(1992).
[ 4] Sprecher C.A., Kisiel W., Mathewes S., Foster D.C.
     Proc. Natl. Acad. Sci. U.S.A. 91:3353-3357(1994).
[ 5] Papayannopoulos I.A., Biemann K.
     Protein Sci. 1:278-288(1992).
```

Extensive abstract and reference lists follow the identified sequence locations for each site. This information can save anybody a tremendous amount of work! The sites themselves are shown with their sequence locations

below each consensus pattern. Sites commonly found in many proteins, such as glycosylation, phosphorylation, amidation, and myristoylation motifs, will be listed if you specify the –Frequent option. However, realize that sites may be false positives. This is always a danger with simple consensus style searches. The GCG programs ProfileScan and HMMerPfam use a more sensitive profile matrix of many PROSITE domains and protein families to search against your sequence. Notice in the example that Bairoch included the A4 protein signatures in his database; the characteristic octapeptides are only found in the A4 protein. The other pattern found, BPTI/Kunitz protease inhibitor, is also characteristic of A4 proteins. Both motifs are true positives in this case.

"**Close**" the motifs output window when you've looked it over and then load the "**`motifs.rsf`**" file into SeqLab. This will add the feature annotation that we created when we activated the –RSF option. The location of the PROSITE signatures will then be included in the editor sequence display. To do this use the "**SeqLab Output Manager**." This very important window, always available through the SeqLab "**Windows**" menu, contains all of the output from your current SeqLab session. Files may be displayed, printed, saved with other names or other locations, and/or deleted from this window. We need to use an extremely important function at this point. Select the file "**`motifs.rsf`**," then press the "**Add to Editor**" button and specify "**Overwrite old with new**" in the next window when prompted, to take the "`motifs.rsf`" feature file and merge it with the RSF (Rich Sequence Format: the sequence data as well as any reference information) file in the open editor. "**Close**" the "**Output Manager**" after loading your new RSF file. Look at your display using color or graphics to display the feature annotation and see if you can recognize any differences.

3) Traditional database searching: Word and Fast approaches — running the algorithms

The oldest heuristic, hashing style, symbol-matching algorithm for similarity searching is WordSearch (Wilbur and Lipman, 1983). It is included in the GCG Package, but is rarely used anymore since both Fast and BLAST style searches outperform this early algorithm, although, since the algorithms do differ, the output results will also differ.

a) A great solution: TFastX — takes advantage of the sensitivity of a protein query, the size of the nucleic acid databases, and allows for frame shifts due to sequencing errors

The original TFastA (Pearson and Lipman, 1988) program was one of the most robust database similarity searching programs around. It compares your peptide sequence against all six translations of the DNA database. This way you can take advantage of the size of the DNA sequence databases yet still retain the vastly increased sensitivity level of protein searches. The newer TFastX program makes the method even more robust by allowing for frame changes that minor sequencing mistakes can cause. These types of errors are especially prevalent in the tags databases. GCG classifies tags as EST's (expressed sequence tags), HTC's (high throughput cDNA sequences), and GSS's (genome survey sequences) — be warned. Furthermore TFastX+ allows you to search your protein sequence against the custom genomic databases that I've installed on our GCG server. These genomic databases have sequences too long for the normal GCG programs to use, but GCG version 11 added a suite of plus (+) programs that can use take advantage of genomic length sequences. Not all GCG sites will have these types of genomic databases; the GCG welcome banner should display their status.

A great feature of the entire Fast family of database search programs is you can search any valid GCG sequence set specification. You are not restricted to specific prebuilt databases, as you are with BLAST (though you can build your own custom BLAST databases in GCG). This can often work to your advantage by allowing you to search a more specific subsection of data. There are several advantages to this: 1) GenBank is highly redundant with many copies of nearly the same sequence, sorting through all this redundancy is a pain; 2) you may only be interested in particular taxa, rather than every life form on earth; 3) searching a smaller dataset increases the statistical discriminatory power of the search; and 4) a smaller search dataset is faster to scan!

Therefore, let's take advantage of this feature, and try to answer a realistic question at the same time, by preparing a LookUp based list file of all the 'primitive' animal sequences. LookUp is GCG's version of EMBL's Sequence Retrieval System (SRS) (Etzold and Argos, 1993), but it provides a GCG compatible list file as output. This list file can serve as the database to be searched by TFastX. To use LookUp, go the "**Functions**" "**Database Reference Searching** " menu and pick "**LookUp**." Check "**RefSeqNuc**" as the library to search, and be sure that "**Search the chosen sequence libraries**," is checked, in your new "**LookUp**" window. RefSeqNuc (RS_RNA) is GCG's version of NCBI's largely non-redundant RefSeq model organism cDNA collection. If there are way too many sequences available in GenBank, then RefSeqNuc is a very convenient place to look. Under the main query section of the window, type the words and symbols "**metazoa ! chordata**" in the "**Organism**" category. This will find all RefSeqNuc cDNA sequences that are Metazoan, but not Chordates.

You need to use Boolean operator symbols to connect individual query strings in LookUp because the databases are indexed using individual words for most fields. The "Organism" field is an exception; it will accept 'Genus species' designations as well as any single word supported level of taxonomy used in the sequence records, e.g. "metazoa." The Boolean operators supported by LookUp are the ampersand, "&," meaning "AND," the pipe symbol, "I," to denote the logical "OR," and the exclamation point, "!," to specify "BUT NOT." Other LookUp query construction rules are case insensitivity, parenthesis nesting, "*" and "?" wildcard support, and automatic wildcard extension (e.g. "transcript" will find "transcriptional" and "transcript"). Press the LookUp "**Run**" button. The program will display the results of the search; look through the output and then "**Close**" the window. Select your LookUp output file in the "**SeqLab Output Manager**," "**Save As . . .**" with a more appropriate name, and then press the "**Add to Main List**" button and "**Close**" the "**SeqLab Output Manager**" window afterwards. This will add the results of the LookUp search to your "`working.list`."

Now select the A4 sequence entry name in the Editor and go to the "**Functions**" "**Database Sequence Searching** " menu and select "**TFastX. . .**" to start the Translation FastX program. If a "**Which selection**" window pops up asking if you want to use the "**selected sequences**" or "**selected region**;" choose "**selected sequences**" to run the program on the full length of the A4 protein. The default database to search at most sites, "**Search Set. . .**" "**Using genbank:***" is all of GenBank, without the tags division, and it's fine to use, if you need it, but we're going to change it here so that we can get through the tutorial in one session. We'll take advantage of the LookUp file that we just created instead, to find remote homologues of A4 in non-Chordate Metazoans. Therefore, push the "**Search Set. . .**" button, select "**genbank:***" in the "**Build TFastX's Search Set**" box that pops up, and then "**Remove from Search Set**." Next, press the "**Add Main List Selection**" button and select

your new LookUp non-Chordate Metazoan LookUp list from the "**List Chooser**" window that pops up; press "**Add to Search Set**" and then "**Close**" the "**List Chooser**" and "**Build Search Set**" windows. The other parameters in the main TFastX window are fine at their default settings, though you may want to decrease the cutoff Expectation value to reduce the output list size. I set my search cutoff value to **1.00**. Press the "**Options. . .**" button to check out the optional parameters. Scroll down the window and notice the "Show sequence alignments in the output file" button. This toggles the command line option –NoAlign off and on to suppress the pairwise alignment section; this can be helpful if you are not interested in the pairwise alignments and wish to have smaller output files produced. Some of the other options, such as restricting your search by the database sequence length or by the date of their deposition in the database, can be helpful depending on your specific situation and should be explored in your own research. "**Close**" the "**Options**" window, be sure that the "**TFastX**" program window shows "**How:**" "**Background Job**," and then press the "**Run**" button. To check on the progress of the job you can go to SeqLab's "**Windows**" menu and choose "**Job Manager**." Select the "**TFastX**" entry to see its progress and then close the window. Go on with the rest of the tutorial now rather than waiting for the TFastX results at this point.

b) Contrast TFastX with normal FastA — protein against protein

Now run the standard FastA program. There's also a FastA+ version of this program so that you can search a 'normal' sized DNA query against a genomic sized sequence database, but we won't be running that today either. We're going to search the A4 protein sequence against the "SwissProt:*_Human" logical protein sequence specification. We can take advantage of UniProt's consistent sequence naming convention to search all of the human SwissProt sequences this way, without preparing another LookUp list. UniProt, which contains SwissProt, is the only database in which this is possible. All other sequence databases use either inconsistent or non-informative naming conventions. This search will find all of the human A4 paralogues in SwissProt as well as human sequences that just have a similar BPTI domain. Start and run the program just like above with TFastX, only this time pick "**FastA. . .**" off of the "**Functions**" "**Database Sequence Searching**" menu. As above, push the "**Search Set. . .**" button, select "**uniprot:***" in the "**Build FastA's Search Set**" box that pops up, and then "**Remove from Search Set**." This time, press the "**Add Database Sequences**" button, and then type "**sw:*_human**" (or SwissProt:*_Human) in the "**SeqLab Database Browser**" window; press "**Add to Search Set**" and then "**Close**" the "**Database Browser**" and "**Build Search Set**" windows. The options are the same between the two programs. "**Run**" the FastA program as a "**How:**" "**Background Job**." Again, proceed with the remainder of the exercise, as these programs will run for a while. Their results will appear as they finish (or be there next time you log on). We will review the results of all of the searches later on.

4) BLAST: Internet and local similarity searching

The BLAST heuristic database-searching algorithm (Altschul, et al., 1990 and 1997) was developed at NCBI. The acronym stands for Basic Local Alignment Search Tool. The original BLAST program only looked for ungapped segments; however, the current version (Altschul, et al., 1997) adds a dynamic programming step to produce gapped alignments. As with the Fast programs, BLAST ranks matches statistically and provides expectation values for each to help evaluate significance. It is best for identifying shorter regions of high similarity — exactly

what you might want with a sequence of unknown function. It is very fast, almost an order of magnitude over traditional sequence similarity database searching, yet maintains the sensitivity of older methods for local similarity in protein sequences! Another advantage with BLAST is it not only shows you the best alignment for each similar sequence found (as in the BestFit type alignments of the Fast programs) but also shows the next best alignments for each up to a certain preset cutoff point. This combines the power of dot-matrix type analyses and the interpretative ease of traditional sequence alignments. One can fine-tune BLAST by altering its operating parameters and taking advantage of the many options available in it; however, BLAST is not very appropriate for comparing non-protein-coding nucleotide sequences against the nucleotide database. When you are forced to perform this type of nucleotide-to-nucleotide search, it is usually best to use FastA style algorithms instead. For more help with BLAST refer to the GCG BLAST documentation, or NCBI's BLAST tutorial available on the Web.

The BLAST server at NCBI can provide the most up to date and quickest database search available. It runs on a large parallel computer system, and it accesses the latest (NCBI updates every night) database by default, nucleotide or protein. The GCG implementation of NCBI's BLAST server, called NetBLAST, runs in a remote client-server mode such that NCBI's database and computers perform the analysis. Unlike other GCG programs though, the list generated by NetBLAST is not appropriate as input to other GCG analyses. NetBLAST returns files in NCBI's own format, incompatible with GCG. For that reason I will be showing local BLAST here, though the same procedures and logic apply to NetBLAST. If you use NetBLAST, because your site does not maintain local BLAST databases, or because you need the very latest sequence data available, or because you want to use NCBI's specialized organism specific databases, then you will have to either use GCG's NetFetch program to retrieve entries from NetBlast's output, or you will have to manually modify NCBI's format to make it comply with GCG standards to use it as input in GCG programs. Alternatively you can run GCG's local BLAST program if you have local BLAST databases assembled at your site, which FSU does. An advantage of running GCG's local BLAST program is the output file is in valid GCG list file format so that it can be fed directly to other GCG programs. Both programs have plus versions, NetBlast+ and BLAST+, that allow the use of genomic sized sequences and databases, but we won't need them today.

To launch GCG's local BLAST program, be sure "**A4_Human**" is selected and then pick "**Blast. . .**" off of the "**Functions**" "**Database Sequence Searching**" menu. As above, if a "**Which selection**" window pops up asking if you want to use the "**selected sequences**" or "**selected region**," choose "**selected sequences**." The default database with a protein query, "**Search a protein database**" and "**Search Set. . .**," is "**Using local uniprot**." This will use BLASTP to search the local version of the UniProt database, but it would take a little longer than we want. BLAST requires precompiled special databases though, and will not accept the general type of GCG sequence specification that the Fast programs will. Therefore, we'll again take advantage of NCBI's RefSeq database, this time searching GCG's protein version of it, RefSeqProt (RS_Prot). RefSeqProt is a bit smaller than UniProt and doesn't contain as much redundancy as UniProt. It will work perfect for our example. You can also tell BLAST to "Search a nucleotide database" in which the default "Search Set. . ." is "Using local genbank." Using BLAST in this manner, that is a protein query against a nucleotide database, activates TBLASTN and provides maximum

sensitivity and database size just as it did with TFastX. However, as with TFastX, searching the local BLAST GenBank database takes quite a while, so we won't use TBLASTN today.

As in the FastA programs, decreasing the Expectation cutoff value will decrease the output list size; I set mine to "**1.0**." Push the "**Options. . .**" button to get a chance to review them. Notice that "**Filter input sequences for complex / repeat regions**" is turned on by default. This activates a very powerful option that should generally be taken advantage of. This option, the –Filter=xs switch, causes troublesome repeat/low information portions of the query sequence to be ignored in the search. This screening of low complexity sequences from your query minimizes search confusion due to random noise. (The programs that perform this function, Xnu and Seg, are available separately in GCG for prescreening your sequences prior to other types analyses besides BLAST.) Also notice the "Display alignments from how many sequences" button; this generates the –Align= option, useful for suppressing unneeded segment alignments and hence reduce the size of the output file. The standard output file is very long because BLAST in SeqLab automatically aligns the best 250 matches so you may wish to reduce this parameter. However, beginning and ending attributes are only saved in the BLAST output list file from those segment alignments that you request. "**Close**" the "**Options**" window and then press the "**Run**" button in BLAST's main window. Results may take a while so go on with the rest of the tutorial at this point.

5) The Profile methods: 'traditional' profiles ala. Gribskov and HMMer profiles (just read this section)

Gribskov et al. (1992) (see additional references in GCG program manual) invented a method for associating distantly related proteins and discovering structural motifs called Profile analysis. John Devereux wrote an excellent overview essay of the method in the GCG program manual ("genmanual" from the command line or the "Help" buttons in SeqLab). Profile analysis can be used with an aligned family of protein sequences assembled from the significant results of database searches such as we are doing here, but we just don't have the time to do that today, so you will not be running any profile analysis in this tutorial. However, you can find remote similarities that all other methods will miss using Profile analysis properly — it is extremely powerful — so, for your information, I'll provide the following . . . and do please read it!

A profile, and its inherent consensus, is created with the GCG program ProfileMake. If you do create your own profiles, all of the members should be appropriately weighted to even out their contributions to the profile. The profile refinement procedure, including repeatedly searching the databases and including or excluding members and adjusting their weights as well as adjusting the profile's length, is known as validating the profile. If using Profile analysis in your own research, following the validation procedures outlined in the GCG Program Manual in the ProfileScan description is prudent. Be sure that all of your alignment sequences (except a mask, if you have one) are selected and then, based on your previous observations, select the longest, most conserved, overall length available. Another effective strategy is to develop multiple, shorter profiles just centered around the similarity peaks of your alignment. Use the "Functions" "Multiple Comparison" menu to launch "ProfileMake." The output profile is a big table of numbers, a bit complicated to understand, but a tremendously powerful tool in subsequent analyses. As mentioned in the Introduction, other programs can read and interpret this alignment

customized scoring matrix to perform very sensitive database searches and alignments by utilizing the information within the matrix that penalizes misalignments in phylogenetically conserved areas more than in variable regions.

To run ProfileSearch in SeqLab use the "Functions" menu selecting "Database Sequence Searching" "ProfileSearch." Specify the "Query profile. . ." in the "File Chooser." Uncheck "ProfileSegments. . ." to prevent ProfileSearch's output from automatically being passed to ProfileSegments. I prefer to run ProfileSegments separately after my ProfileSearch is done. This way I can edit the ProfleSearch output file so that ProfileSegments only makes pairwise or multiple alignments of the sequences that I am interested in to my profile. I also use the –MinList option, changing "Lowest Z score to report in output list" from 2.5 to 3.5 or higher, to set a Z score cut-off value. ProfileSearch Z scores are normalized and reflect the significance of the results. Here rather than randomizing sequences to evaluate the Z score as is done in the Monte Carlo approach with –Randomization in Gap and BestFit, they are calculated based on all of the non-similar sequences from the database search, somewhat similar to the way that BLAST and FastA calculates their Expectation scores.

The program ProfileSegments makes BestFit style alignments of the results of a ProfileSearch. An option in ProfileSegments, –MSF, allows you to prepare a multiple sequence alignment of the ProfileSearch segments. The full information content of the profile is utilized in this alignment procedure. The importance of the conserved portions of an alignment as reflected in the content of a profile is fully utilized in this alignment procedure. Another option is –Global versus the –Local default; this will force full-length alignments, which might be what you would want, if you are trying to build up a multiple sequence alignment.

The other profile method mentioned in the Introduction is Sean Eddy's HMMer package. As powerful as Gribskov style profiles are, they require a lot of time and skill to prepare and validate, and they are heuristics based. An excess of subjectivity and a lack of formal statistical rigor also contribute as drawbacks. In collaboration with the author (Eddy, 1996 and 1998) GCG has incorporated the HMMer (pronounced "hammer") package into the Wisconsin Package. HMMer uses Hidden Markov modeling, with a formal probabilistic basis and consistent gap insertion theory, to build and manipulate HMMer profiles and profile databases, to search sequences against HMMer profile databases and visa versa, and to easily create multiple sequence alignments using HMMer profiles as a 'seed.' Again, GCG has taken the time to write an excellent essay in the Program Manual on HMMer, what Hidden Markov Models are, and how the algorithms work. I urge you to read it, as well as each individual HMMer program description, at some point. The 'take-home' message is HMMer profiles are much easier to build than traditional profiles and they do not need to have nearly as many sequences in their alignments in order to be effective. Furthermore, they offer a statistical rigor not available in Gribskov profiles, and they have all the sensitivity of any profile technique.

As with Gribskov profiles, HMMer profiles are built from a set of pre-aligned sequences. It's just not as important that the alignment be as comprehensive and perfect. But, as stated above, do not do this procedure today. To build a HMMer profile of an alignment in SeqLab select all of the relevant sequences and use the "Functions" "HMMER" menu "HmmerBuild" choice. Accept the default "create a new HMM" and specify some "Internal name for profile HMM." Also specify the "Type of HMM to be Built" — "multiple global" is the default. This is a big

difference between HmmerBuild and other profile building programs; when the profile is built you need to specify the type of eventual alignment it will be used with, rather than when the alignment is run. The HMMer profile will either be used for global or local alignment, and it will occur multiply or singly in a given sequence. Weighting is also handled differently in HMMer than it is with Gribskov profiles. To use a custom weighting scheme, e.g. if you've modified your RSF file weight values for ProfileBuild, you need to tell HmmerBuild not to use one of its built-in weighting schemes with the –Weighting=N option. Otherwise HmmerBuild's internal weighing algorithm will calculate the best weights for you automatically based on the sequences' similarities using a cluster analysis approach. HmmerCalibrate is checked by default. The completion of HmmerBuild automatically launches this calibration procedure to increase the speed and accuracy of subsequent analyses with the resultant profile. The output is an ASCII text profile representation of a statistical model, a Hidden Markov Model, of the consensus of a sequence family, deduced from a multiple sequence alignment.

A utility program, HmmerConvert, can change HMMer style profiles into Gribskov profiles, however information is lost in the process. Normally you would use your HMMer profile directly as a search probe for extremely sensitive database searching with the HMMerSearch program or as a template with which to build ever-larger alignments with the HMMer Align program. Both methods work great!

To use a HMMer profile as a search probe use the "Functions" menu "HMMER" "HmmerSearch" choice. Specify the HMMer profile with "Profile HMM to use as query. . ." using the "File Chooser" window to select the correct HMMer profile. Also specify the "Sequence search set. . .." HmmerSearch has similar cutoff parameters as other GCG database searches, that is, you can restrict the size of the output based on significance scores and you can limit the number of pairwise alignments displayed. HmmerSearch is very slow because, like ProfileSearch, it is a true dynamic programming implementation, a profile matrix against a whole database. So run it in the background when using SeqLab or, if at a terminal session, use the –Batch command line option. If your server has multiple processors, both programs, and all the BLAST and Fast programs, support the multithreading –Processors=x option to speed things up. The output is huge but very informative. Everything is based on significance Expectation value scores. The top portion shows best hits based on all domains, the second section is the list file portion of the best domain hits, next pairwise alignments are given, and finally a score distribution is plotted. It is a valid list file, so other GCG programs, particularly HmmerAlign, can read it.

HmmerAlign can be an incredible help to people working with very large multiple alignments and for adding newly found sequences to an existing alignment regardless of size. Somewhat similar in concept to the –MSF option of ProfileSegments, it takes a specified profile, in this case a HMMer profile, and aligns a specified set of sequences to it, to produce a multiple sequence alignment based on that profile. Unlike ProfileSegments, HmmerAlign takes any GCG sequence specification as input, not just the output from its own database-searching program. It is much faster to create very large multiple alignments this way, versus using PileUp or ClustalW, on an entire large dataset. The rationale being — take the time to make a good small alignment and HMMer profile, then use that to build up the original larger and larger. The alignment procedure used by HmmerAlign is a full-blown, recursive, dynamic programming implementation, the profile's matrix against every sequence individually, until an entire alignment is built. HmmerAlign can also use its profile to align one multiple alignment to another and produce a

merged result of the two. A heuristic (optimality is not guaranteed) solution is provided in this instance. To use this option choose "Combine output alignment and . . ." "another alignment" in the SeqLab "HmmerAlign Options" window. This will launch the –Heuristic= option. Using the original alignment that you made the profile with, against another sequence set is very fast; it is the –MapAlignment= option and it provides an exact, non-heuristic alignment. Launch HmmerAlign off the "Functions" "HMMER" menu by picking "HammerAlign." Specify the correct HMMer profile with the "profile HMM to use . . ." button and pick the sequences that you want to align to the profile with the "Sequences to align . . ." button.

Two other extremely powerful search algorithms available in GCG should be mentioned at this point but I do not want you to run them either, because they are incredibly cpu intensive. These are FrameSearch and SSearch. They are both full dynamic programming searches, that is, they use no hashing or heuristics. The former allows reading frame transitions in protein to nucleotide comparisons; the latter is used for comparing sequences of the same type. FrameSearch is a 'native' GCG program written by Irv Edelman, SSearch uses William Pearson's implementation of the method of Smith and Waterman (1981) for searching a database.

6) What Next? Comparisons, interpretations, and further analyses

Use the "**Output Manager**" located under SeqLab's "**Windows**" menu to display and manage your TFastX, FastA, and BLAST output files. You can also use the "**Job Manager**" located there to check on the status of any running jobs. Just select the job to see its status and/or any associated error statements.

By now, after reading about Profiles and HMMs, all of your jobs should be done. Let's check them out. Naturally, the topmost 'hits' will turn out to be A4 proteins. That is to be expected, and isn't all that interesting; it's the ones below the expected hits that may prove particularly interesting for this workshop. In our A4 protein case I expect to see the best E values for other A4 proteins and close homologues, and then another bracket of values for other sequences with BPTI protease inhibitor domain signatures, and finally, a category of not so good scores that reflect background noise. I'll show my abridged TFastX, FastA, and local BLASTP results next.

First a look at my abridged TFastX output from the search of the non-Chordate Metazoan list from RefSeqNuc:

```
!!SEQUENCE_LIST 1.0

 (Peptide) TFASTX of: a4_human  from: 1 to: 770  November 6, 2006 13:45

ID   A4_HUMAN       STANDARD;      PRT;   770 AA.
AC   P05067; P09000; P78438; Q13764; Q13778; Q13793; Q16011; Q16014;
DT   13-AUG-1987, integrated into UniProtKB/Swiss-Prot.
DT   25-JUL-2006, entry version 119. . . .

 TO: @nonchordate_metazoa.list  Sequences:     95,789  Symbols:   157,707,062  Word Size: 2

 Databases searched:
   REFSEQ_RNA, Release 18.0, Released on 21Jul2006, Formatted on 20Jul2006

 Searching both strands.
 Scoring matrix: GenRunData:blosum50.cmp
 Variable pamfactor used
 Gap creation penalty: 15  Gap extension penalty: 2  Frameshift penalty: 20

Histogram Key:
```

```
 Each histogram symbol represents 169 search set sequences
 Each inset symbol represents 5 search set sequences
 z-scores computed from opt scores

z-score obs    exp
        (=)    (*)

< 20      0      0:
  22      1      0:=
  24      0      0:
  26      4      2:*
  28      5     22:*
  30     28    132:*
  32    168    509:=  *
  34    710   1380:=====   *
  36   2017   2834:============    *
  38   4145   4684:=========================  *
  40   6803   6533:=======================================*==
  42   8960   7986:================================================*======
  44  10056   8809:=========================================================*=======
  46  10125   8972:=========================================================*======
  48   9429   8590:=====================================================*=====
  50   8301   7838:=============================================*===
  52   7042   6891:=======================================*=
  54   5761   5886:==================================*
  56   4537   4917:============================  *
  58   3660   4037:====================== *
  60   2934   3270:================== *
  62   2262   2622:============== *
  64   1675   2085:==========  *
  66   1384   1648:=========*
  68   1031   1296:=======*
  70    885   1016:======*
  72    680    794:====*
  74    503    619:===*
  76    382    482:==*
  78    360    374:==*
  80    282    291:=*
  82    256    222:=*
  84    215    176:=*
  86    154    136:*
  88    127    105:*
  90    111     82:*
  92     84     63:*         :============*====
  94     85     49:*         :=========*=======
  96     53     38:*         :=======*===
  98     77     29:*         :=====*==========
 100     46     23:*         :====*=====
 102     57     18:*         :===*========
 104     35     14:*         :==*====
 106     36     10:*         :=*======
 108     27      8:*         :=*====
 110     23      6:*         :=*===
 112     17      5:*         :*===
 114     19      4:*         :*===
 116     24      3:*         :*====
 118     11      2:*         :*==
>120    202      2:*=        :*=======================================

Joining threshold: 38, opt. threshold: 26, opt. width:  16, reg.-scaled

The best scores are:                  strand init1 initn   opt    z-sc E(95519)..

RS_RNA:XM_312126    Begin: 100  End: 2592
! Anopheles gambiae str. PEST ENSANGP...(f)   295   759   686   704.0   2e-32
RS_RNA:XM_624121    Begin: 164  End: 2287
! PREDICTED: Apis mellifera similar t...(f)   306   902   559   571.6  4.7e-25
RS_RNA:NM_057278    Begin: 205  End: 2832
! Drosophila melanogaster amyloid pro...(f)   241   797   485   492.2  1.2e-20
```

```
RS_RNA:XM_785222    Begin: 16  End: 1875
! PREDICTED: Strongylocentrotus purpu...(f)   274   589   375   384.3  1.3e-14
RS_RNA:NM_076469    Begin: 18  End: 2051
! Caenorhabditis elegans Amyloid Prec...(f)   173   935   304   308.8  2.1e-10
RS_RNA:NM_076470    Begin: 25  End: 2052
! Caenorhabditis elegans Amyloid Prec...(f)   173   935   295   299.4  6.8e-10
RS_RNA:XM_320745    Begin: 2113  End: 2274
! Anopheles gambiae str. PEST ENSANGP...(f)   234   265   245   248.0   5e-07
RS_RNA:NM_171931    Begin: 4124  End: 4468
! Caenorhabditis elegans PaPiliN (Dro...(f)   233   233   235   234.3  2.9e-06
RS_RNA:NM_072617    Begin: 4337  End: 4681
! Caenorhabditis elegans PaPiliN (Dro...(f)   226   226   235   234.1   3e-06
RS_RNA:XM_970371    Begin: 1849  End: 2025
! PREDICTED: Tribolium castaneum simi...(f)   192   227   230   233.3  3.3e-06
RS_RNA:NM_072616    Begin: 4337  End: 4669
! Caenorhabditis elegans PaPiliN (Dro...(f)   226   226   231   228.4  6.2e-06
RS_RNA:NM_075592    Begin: 2164  End: 2481
! Caenorhabditis elegans MoLTing defe...(f)   191   220   230   227.2  7.2e-06
RS_RNA:XM_394300    Begin: 1864  End: 2100
! PREDICTED: Apis mellifera similar t...(f)   219   219   222   225.0  9.6e-06
RS_RNA:NM_176574    Begin: 5917  End: 6087
! Drosophila melanogaster Papilin CG3...(f)   222   222   222   217.3  2.6e-05
RS_RNA:NM_176575    Begin: 5917  End: 6087
! Drosophila melanogaster Papilin CG3...(f)   214   214   222   217.1  2.6e-05
RS_RNA:NM_073094    Begin: 523  End: 1389
! Caenorhabditis elegans ZK287.4 (ZK2...(f)   164   187   212   212.1   5e-05
RS_RNA:NM_136586    Begin: 172  End: 330
! Drosophila melanogaster CG13748-RA ...(f)   168   168   199   211.6  5.4e-05
RS_RNA:XM_781686    Begin: 514  End: 678
! PREDICTED: Strongylocentrotus purpu...(f)   207   261   210   211.1  5.7e-05
RS_RNA:XM_784051    Begin: 2116  End: 2280
! PREDICTED: Strongylocentrotus purpu...(f)   207   325   210   208.4  8.1e-05
RS_RNA:XM_394721    Begin: 154  End: 538
! PREDICTED: Apis mellifera similar t...(f)   196   221   210   206.5  0.0001
RS_RNA:NM_134957    Begin: 139  End: 366
//////////////////////////////////////////////////////////////////////////////
RS_RNA:NM_063072    Begin: 1906  End: 2127
! Caenorhabditis elegans F10E7.4 (F-s...(f)    75    75   189   190.1  0.00084
RS_RNA:NM_164548    Begin: 154  End: 315
! Drosophila melanogaster Acp24A4 CG3...(f)    97    97   176   188.1  0.0011
RS_RNA:NM_170073    Begin: 249  End: 449
! Drosophila melanogaster CG6784-RB, ...(f)   116   116   178   187.2  0.0012
RS_RNA:XM_553610    Begin: 4  End: 162
! Anopheles gambiae str. PEST ENSANGP...(f)   144   144   171   186.3  0.0014
RS_RNA:NM_142888    Begin: 63  End: 227
! Drosophila melanogaster CG17380-RA ...(f)   133   133   180   185.9  0.0014
RS_RNA:NM_059619    Begin: 1530  End: 1703
! Caenorhabditis elegans K10D3.4 (K10...(f)   152   234   186   185.5  0.0015
RS_RNA:NM_077467    Begin: 2645  End: 2797
! Caenorhabditis elegans C34F6.1 (C34...(f)   167   258   185   184.4  0.0018
RS_RNA:NM_073283    Begin: 250  End: 414
! Caenorhabditis elegans F22E12.1 (F2...(f)   142   178   181   181.8  0.0025
RS_RNA:NM_142498    Begin: 193  End: 330
! Drosophila melanogaster CG14298-RA ...(f)   113   142   170   181.7  0.0025
RS_RNA:NM_134955    Begin: 165  End: 290
! Drosophila melanogaster CG16713-RA ...(f)    60    60   167   178.8  0.0036
RS_RNA:NM_134954    Begin: 133  End: 258
! Drosophila melanogaster CG3513-RA (...(f)    52    52   158   170.5    0.01
//////////////////////////////////////////////////////////////////////////////
RS_RNA:NM_134950    Begin: 236  End: 448
! Drosophila melanogaster CG15418-RA ...(f)    97    97   144   152.9   0.099
RS_RNA:NM_001028945    Begin: 130  End: 288
! Caenorhabditis elegans MEChanosenso...(f)   112   200   153   151.2    0.12
RS_RNA:NM_134956    Begin: 178  End: 303
! Drosophila melanogaster CG16712-RA ...(f)    60    60   140   150.2    0.14
RS_RNA:XM_795293    Begin: 13  End: 327
! PREDICTED: Strongylocentrotus purpu...(f)   128   128   141   150.1    0.14
RS_RNA:NM_071949    Begin: 40  End: 216
! Caenorhabditis elegans K11D12.6 (K1...(f)    86   118   142   149.9    0.15
```

```
RS_RNA:XM_393227    Begin: 354  End: 596  Strand: -
! PREDICTED: Apis mellifera similar t...(r)   104   132   156   149.8    0.15
RS_RNA:NM_067005    Begin: 506  End: 744
! Caenorhabditis elegans W05B2.2 (W05...(f)   151   175   151   148.7    0.17
RS_RNA:XM_795657    Begin: 150  End: 389
! PREDICTED: Strongylocentrotus purpu...(f)   139   139   139   148.6    0.17
RS_RNA:XM_795558    Begin: 27  End: 329
! PREDICTED: Strongylocentrotus purpu...(f)   133   133   138   148.6    0.17
RS_RNA:NM_064805    Begin: 1880  End: 2097
! Caenorhabditis elegans F30H5.3 (F30...(f)   134   231   152   147.9    0.19
RS_RNA:NM_066634    Begin: 2014  End: 2130
! Caenorhabditis elegans ZC84.6 (ZC84...(f)   117   145   151   147.6     0.2
RS_RNA:XM_795472    Begin: 34  End: 357
////////////////////////////////////////////////////////////////////////
! Drosophila melanogaster CG12017-RA ...(f)    77    77   136   135.6    0.92
RS_RNA:XM_965595    Begin: 1191  End: 1404
! PREDICTED: Tribolium castaneum simi...(f)   106   106   133   135.5    0.92
\\End of List


a4_human
RS_RNA:XM_312126

LOCUS       XM_312126               2664 bp    mRNA    linear   INV 16-MAR-2005
DEFINITION  Anopheles gambiae str. PEST ENSANGP00000022077
            (ENSANGG00000019588), partial mRNA.
ACCESSION   XM_312126
VERSION     XM_312126.2  GI:58382725
KEYWORDS    . . . .


SCORES  Strand: (f) Init1: 295   Initn: 759   Opt: 686   z-score: 704.0 E(): 2e-32
>>RS_RNA:XM_312126                                        (2664 nt)
Frame: 1 initn: 759 init1: 295 opt: 686 Z-score: 704.0 expect(): 2e-32
Smith-Waterman score: 779;    26.9% identity in 841 aa overlap
 (31-766:100-2589)

                        40        50           60        70        80
a4_human     EPQIAMFC---GRLNMHMNVQNGKWDSDPS---GTKTCIDTKEGILQYCQEVYPELQITN
             ||||:::|      : ::  ::|:| :| |     :||:  |  :|:||::|||  :|||
XM_312126    EPQISVLCEAGQTYHPQFLSEEGRWTTDLSIKVPGSTCLRDKMDLLDYCKKVYPGRDITN
           100       130       160       190       220       250

                90       100          110       120       130       140
a4_human     VVEANQPVTIQNWCKRGR---KQCKTHPHFVIPYRCLVGEFVSDALLVPDKCKFLHQERM
             :||:::   | :||::|     :||   ::: |:||| | | |||||||: | | | :
XM_312126    IVESSHYQKIGGWCRQGALNAAKCKGAQRWIKPFRCLEGPFQSDALLVPEGCLFDHIHNA
           280       310       340       370       400       430

//////////////////////////////////////////////////////////////////////

a4_human     YKFFE
             ||:||
XM_312126    YKYFE
              2590

//////////////////////////////////////////////////////////////////////

a4_human
RS_RNA:NM_063072

LOCUS       NM_063072               2577 bp    mRNA    linear   INV 06-FEB-2006
DEFINITION  Caenorhabditis elegans F10E7.4 (F-spondin) mRNA, complete cds.
ACCESSION   NM_063072
VERSION     NM_063072.4  GI:86561927
KEYWORDS    .
SOURCE      Caenorhabditis elegans . . .
```

```
SCORES  Strand: (f) Init1: 75    Initn: 75    Opt: 189   z-score: 190.1 E(): 0.00084
>>RS_RNA:NM_063072                                        (2577 nt)
Frame: 1 initn:  75 init1:  75 opt: 189 Z-score: 190.1 expect(): 0.00084
Smith-Waterman score: 189;    35.1% identity in 74 aa overlap
 (283-354:1906-2124)

                290       300       310       320       330       340
a4_human     VEEVVREVCSEQAETGPCRAMISRWYFDVTEGKCAPFFYGGCGGNRNNFDTEEYCMAVC-
             :|   :|:|:|: ::| | : : |::::  : :|  |:: || ||||:|:||| |  :|
NM_063072    IEINSEEICQEDKQAGQCAGNFPRYWYNHEKTQCERFIFTGCKGNRNQFETEEECKQICL
                1930      1960      1990      2020      2050      2080

                  350
a4_human     -GSAMSQSLLKTTQ
              |   |:||: ::|
NM_063072    PGYEKSKSLIPNNQ
                2110

! Distributed over 1 thread.
!      Start time: Mon Nov  6 13:40:54 2006
! Completion time: Mon Nov  6 13:46:13 2006

! CPU time used:
!        Database scan:  0:04:02.6
! Post-scan processing:  0:00:51.0
!       Total CPU time:  0:04:53.6
! Output File: nonchordate_metazoa.tfastx
```

A histogram of the score distribution is displayed in all Fast program outputs. This can be helpful to get a feeling for the statistical significance of the search and in ascertaining whether you ran your search list large enough. For the search statistics to be valid, the expected distribution, as indicated by the line of asterisks, should approximate the actual distribution, as shown by the equal signs. Here that correspondence is great. Normally you want your list size big enough to include some of the population of random low scores to help you ascertain the significance of the alignments; the default FastA Expectation cutoff of 10.0 assures this, though our use of 1.0 is usually safe. The inset shows a blowup of the highly significant score end of the graph — these are the best alignments found by the program, not the worst! The histogram can be suppressed with the –NoHistogram option if desired.

Since we searched a relatively small dataset, just the subset of RefSeqNuc containing 95,789 sequences from non-Chordate Metazoans, the scores fall off pretty fast. It's fairly easy to see at least two categories of Expectation values: A4 homologues having E values from 2 x 10^{-32} through 6.8 x 10^{-10}, and then a jump in E values to BPTI domain homologues with E values from 5 x 10^{-7} through around 5 x 10^{-1} where they merge into random noise. Also notice that only Insects and Nematodes, of all the non-Chordate Metazoans appear to possess true A4 proteins, though lack of finding a sequence certainly is not proof that they don't exist. They may just not have been sequenced yet, or they aren't one of the model organisms in RefSeqNuc. The TFastX output file is an acceptable GCG list file that can serve as input to other programs such as PileUp or ClustalW+. TFastX output shows the sequence alignment for each pair, unless you suppress it with the –NoAlign option. Identities are displayed as vertical pipe symbols (|) and similarities are represented by colons (:). The beginning and ending alignment points, along with the translation frames, in this section of the output can be used to go back to the original nucleotide entry to check whether the match-ups correspond to actually translated areas.

The entries are sorted by a "*z*" score parameter based on a normalization of the "*opt*" scores and their distribution from the rest of the database. As mentioned earlier this *z-score* is different than the traditional Monte Carlo style

distribution Z score. Here it is calculated from a linear regression against the natural log of the search set sequence length. Both describe the statistical significance of an alignment. The Expectation function, based on the number of sequences searched, here *E(95519)*, is calculated from the *z-score*, and is the most important column. It describes the number of search set sequences that would be needed to obtain a *z-score* greater than or equal to the *z-score* obtained in any particular search purely by chance; in-other-words, the smaller the number, the better. As a conservative rule-of-thumb, for a search against a protein database of around 100,000 sequences, as long as optimization is not turned off, E values of much less than 0.01 are probably homologous, and scores from 0.01 to 1 may be homologous, whereas scores between 1 to 10 are only perhaps homologous, although these guidelines can be skewed by compositional biases. Furthermore, a lack of significance cannot disprove homology, it only works the other way, that is, significance can only prove homology.

Sometimes the sequences found by TFastX (or TBLASTN) will not show up in any other searches, because the region may not be considered a 'normal' coding sequence. This could be important, especially if sometime during your peptide's evolutionary history it incorporated (was 'infected' by) any type of mobile DNA element.

Next my abridged example FastA output file. Remember this was against all the human SwissProt sequences:

```
!!SEQUENCE_LIST 1.0

(Peptide) FASTA of: a4_human  from: 1 to: 770  November 6, 2006 14:32

ID   A4_HUMAN       STANDARD;      PRT;   770 AA.
AC   P05067; P09000; P78438; Q13764; Q13778; Q13793; Q16011; Q16014;
DT   13-AUG-1987, integrated into UniProtKB/Swiss-Prot.
DT   25-JUL-2006, entry version 119. . . .

 TO: sw:*_human  Sequences:     14,445  Symbols:     7,944,717  Word Size: 2

 Databases searched:
   UNIPROT, Release 8.4, Released on 25Jul2006, Formatted on 14Aug2006

 Scoring matrix: GenRunData:blosum50.cmp
 Variable pamfactor used
 Gap creation penalty: 12  Gap extension penalty: 2

Histogram Key:
 Each histogram symbol represents 27 search set sequences
 Each inset symbol represents 1 search set sequences
 z-scores computed from opt scores

z-score obs    exp
        (=)    (*)

< 20     13      0:=
  22      0      0:
  24      0      0:
  26      1      0:=
  28      0      3:*
  30      4     20:*
  32     14     77:= *
  34     97    208:====   *
  36    260    427:==========     *
  38    653    705:========================= *
  40   1094    984:====================================*====
  42   1475   1203:============================================*==========
  44   1597   1327:==================================================*==========
  46   1571   1352:==================================================*========
  48   1403   1294:============================================*====
```

```
  50   1196   1181:==========================================*=
  52    946   1038:===================================  *
  54    802    887:==============================  *
  56    658    741:=========================  *
  58    564    608:====================  *
  60    412    493:================  *
  62    317    395:============  *
  64    264    314:==========  *
  66    175    248:=======  *
  68    166    195:=======*
  70    114    153:=====*
  72     91    120:====*
  74     91     93:===*
  76     79     73:==*
  78     63     56:==*
  80     47     44:=*
  82     40     33:=*
  84     38     27:*=
  86     19     21:*
  88     26     16:*
  90     23     12:*
  92     19     10:*         :=========*=========
  94     15      7:*         :======*========
  96     14      6:*         :=====*========
  98     14      4:*         :===*==========
 100      3      3:*         :==*
 102     10      3:*         :==*=======
 104      6      2:*         :=*====
 106      9      2:*         :=*=======
 108      6      1:*         :*=====
 110      3      1:*         :*==
 112      3      1:*         :*==
 114      3      1:*         :*==
 116      0      0:          *
 118      3      0:=         *===
>120     24      0:=         *========================

Joining threshold: 38, opt. threshold: 26, opt. width:  16, reg.-scaled

The best scores are:                    init1 initn   opt    z-sc E(14388)..

UNI_SPROT:A4_HUMAN    Begin: 1  End: 770
! P05067 h amyloid beta a4 protein pr... 5160  5160  5160  4207.0       0
UNI_SPROT:APLP2_HUMAN    Begin: 15  End: 762
! Q06481 homo sapiens (human). amyloi...  925  2460  1295  1062.9  3.1e-53
UNI_SPROT:APLP1_HUMAN    Begin: 23  End: 648
! P51693 homo sapiens (human). amyloi...  667  1431   683   565.9  1.5e-25
UNI_SPROT:SPIT2_HUMAN    Begin: 131  End: 183
! O43291 homo sapiens (human). kunitz...  207   207   236   207.5  1.4e-05
UNI_SPROT:TFPI1_HUMAN    Begin: 22  End: 131
! P10646 homo sapiens (human). tissue...  190   190   213   187.8  0.00017
UNI_SPROT:CO6A3_HUMAN    Begin: 3071  End: 3167
! P12111 homo sapiens (human). collag...  206   206   226   185.5  0.00023
UNI_SPROT:EPPI_HUMAN    Begin: 74  End: 127
! O95925 homo sapiens (human). eppin ...  169   169   200   181.7  0.00037
UNI_SPROT:AMBP_HUMAN    Begin: 287  End: 338
! P02760 homo sapiens ambp protein pr...  198   198   203   178.8  0.00054
UNI_SPROT:SPIT1_HUMAN    Begin: 233  End: 371
! O43278 homo sapiens (human). kunitz...  180   180   203   176.6  0.00072
UNI_SPROT:TFPI2_HUMAN    Begin: 34  End: 86
! P48307 homo sapiens (human). tissue...  154   154   191   171.3  0.0014
UNI_SPROT:WFDC8_HUMAN    Begin: 92  End: 156
! Q8iua0 homo sapiens (human). wap fo...  123   123   175   158.1  0.0077
UNI_SPROT:SPIT3_HUMAN    Begin: 11  End: 66
! P49223 homo sapiens (human). kunitz...  136   136   165   156.8   0.009
UNI_SPROT:CO7A1_HUMAN    Begin: 2825  End: 2936
! Q02388 homo sapiens (human). collag...  122   175   184   151.7   0.017
UNI_SPROT:MYT1L_HUMAN    Begin: 91  End: 165
! Q9ul68 homo sapiens (human). myelin...  136   136   173   147.8   0.029
```

```
UNI_SPROT:KINH_HUMAN    Begin: 483  End: 839
! P33176 homo sapiens (human). kinesi...   42    42   167   144.0   0.047
UNI_SPROT:CALR_HUMAN    Begin: 335  End: 407
! P27797 homo sapiens (human). calret...  107   107   147   132.3    0.21
UNI_SPROT:NUCL_HUMAN    Begin: 181  End: 275
! P19338 homo sapiens (human). nucleo...  106   106   148   130.2    0.27
UNI_SPROT:YTDC1_HUMAN    Begin: 116  End: 245
! Q96mu7 homo sapiens (human). yth do...  121   121   144   126.9    0.42
UNI_SPROT:AN32E_HUMAN    Begin: 149  End: 215
! Q9btt0 homo sapiens (human). acidic...   99    99   131   121.7    0.81
UNI_SPROT:FA13A_HUMAN    Begin: 192  End: 606
! O94988 homo sapiens (human). protei...   49    49   137   121.4    0.85
UNI_SPROT:MYT1_HUMAN    Begin: 236  End: 328
! Q01538 homo sapiens (human). myelin...  117   117   140   121.2    0.87
UNI_SPROT:PTMA_HUMAN    Begin: 10  End: 83
! P06454 homo sapiens (human). prothy...  100   100   124   120.9     0.9
UNI_SPROT:UBF1_HUMAN    Begin: 683  End: 746
! P17480 homo sapiens (human). nucleo...   86    86   137   120.9    0.91
UNI_SPROT:NFL_HUMAN    Begin: 452  End: 526
! P07196 homo sapiens (human). neurof...   99    99   134   120.3    0.98
\\End of List

a4_human
UNI_SPROT:A4_HUMAN

ID   A4_HUMAN       STANDARD;      PRT;   770 AA.
AC   P05067; P09000; P78438; Q13764; Q13778; Q13793; Q16011; Q16014;
AC   Q16019; Q16020; Q9BT38; Q9UCA9; Q9UCB6; Q9UCC8; Q9UCD1; Q9UQ58;
DT   13-AUG-1987, integrated into UniProtKB/Swiss-Prot.
DT   01-NOV-1991, sequence version 3.
DT   25-JUL-2006, entry version 119. . . .


SCORES   Init1: 5160  Initn: 5160  Opt: 5160  z-score: 4207.0 E(): 0
>>UNI_SPROT:A4_HUMAN                                    (770 aa)
 initn: 5160 init1: 5160 opt: 5160 Z-score: 4207.0 expect():    0
Smith-Waterman score: 5160;   100.0% identity in 770 aa overlap
 (1-770:1-770)

                     10        20        30        40        50        60
a4_human     MLPGLALLLLAAWTARALEVPTDGNAGLLAEPQIAMFCGRLNMHMNVQNGKWDSDPSGTK
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
A4_HUMAN     MLPGLALLLLAAWTARALEVPTDGNAGLLAEPQIAMFCGRLNMHMNVQNGKWDSDPSGTK
                     10        20        30        40        50        60

//////////////////////////////////////////////////////////////////////////////

a4_human
UNI_SPROT:NFL_HUMAN

ID   NFL_HUMAN      STANDARD;      PRT;   542 AA.
AC   P07196; Q16154; Q8IU72;
DT   01-APR-1988, integrated into UniProtKB/Swiss-Prot.
DT   01-FEB-2005, sequence version 2.
DT   27-JUN-2006, entry version 71.
DE   Neurofilament triplet L protein (68 kDa neurofilament protein) . . .


SCORES   Init1: 99    Initn: 99    Opt: 134   z-score: 120.3 E(): 0.98
>>UNI_SPROT:NFL_HUMAN                                    (542 aa)
 initn:  99 init1:  99 opt: 134 Z-score: 120.3 expect(): 0.98
Smith-Waterman score: 134;    38.7% identity in 75 aa overlap
 (191-264:452-526)

                    170       180       190       200       210
a4_human     KSTNLHDYGMLLPCGIDKFRGVEFVCCPLAEESDNVDSADAEE-DDSDVWWGGADTDYAD
                                        |  :::::| |||  |     | |: :  |
NFL_HUMAN    AYGGLQTSSYLMSTRSFPSYYTSHVQEEQIEVEETIEAAKAEEAKDEPPSEGEAEEEEKD
                   430       440       450       460       470       480
```

```
           220       230       240       250       260       270
a4_human     GSEDKVVEVAEEEEVAEVEEEEADDDEDDEDGDEVEEEAEEPYEEATERTTSIATTTTTT
               | :  |:|||||:|: | ||| ::|:  :|:| ||  |   ||
NFL_HUMAN    KEEAEEEEAAEEEEAAKEESEEAKEEEEGGEGEEGEETKEAEEEEKKVEGAGEEQAAKKK
                   490       500       510       520       530       540

           280       290       300       310       320       330
a4_human     TESVEEVVREVCSEQAETGPCRAMISRWYFDVTEGKCAPFFYGGCGGNRNNFDTEEYCMA

NFL_HUMAN    D


! Distributed over 1 thread.
!      Start time: Mon Nov  6 14:32:34 2006
! Completion time: Mon Nov  6 14:32:56 2006

! CPU time used:
!        Database scan:  0:00:02.9
! Post-scan processing:  0:00:02.4
!       Total CPU time:  0:00:05.3
! Output File: sw_human.fasta
```

Again the histogram score distribution shows excellent correlation to the expected extreme value distribution, so the search's statistics are absolutely valid. The sorted list of scores falls off even more quickly than the previous example since we just looked at human sequences, a much smaller dataset, only 14,388 sequences. The *E(14388)* significance value clearly demarcates three human A4 paralogues with E values from zero to 1.5 x 10^{-25}, then there's a huge jump to the BPTI domain Kunitz protease inhibitor homologues with E values from 1.4 x 10^{-5} up to around 1 x 10^{-2}, at which point the matches wander off into randomness. Compositional biases in the database are responsible for some of the seemingly good scores that do not possess a BPTI domain. We'll investigate this phenomenon further below. As with all the Fast program family, BestFit style alignments are displayed of the requested number of hits, if alignments are not suppressed with the –NoAlign option.

Finally let's look at my greatly abridged BLAST output. As with the previous two Fast style searches, especially pay attention to the E values in its output file. Remember, these are the likelihoods (expectations) that the observed matches could be due to chance: the smaller the E value, the more significant the match. They are much easier to interpret than the information bits score in the adjacent column:

```
!!SEQUENCE_LIST 1.0
BLASTP 2.2.10 [Oct-19-2004]

Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
"Gapped BLAST and PSI-BLAST: a new generation of protein database search
programs",  Nucleic Acids Res. 25:3389-3402.

Query= UNIPROT_SPROT:A4_HUMAN
         (770 letters)

Database:  rs_prot
           2,762,164 sequences; 974,374,765 total letters

Searching. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .done

                                                          Score    E
 Sequences producing significant alignments:              (bits)  Value ..

RS_PROT:NP_000475  Begin: 19 End: 770
!amyloid beta A4 protein precursor, isoform a [Homo sap..  1305   0.0
```

```
RS_PROT:NP_001013036  Begin: 19 End: 770
!amyloid beta A4 protein [Pan troglodytes].                1304   0.0
RS_PROT:NP_999537  Begin: 19 End: 770
!amyloid precursor protein variant 2 [Sus scrofa].         1280   0.0

//////////////////////////////////////////////////////////////////////

RS_PROT:NP_033821  Begin: 42 End: 204
!amyloid beta (A4) precursor-like protein 2 [Mus muscul..
RS_PROT:XP_508884  Begin: 198 End: 537
!PREDICTED: amyloid beta (A4) precursor-like protein 2 ..   342   4e-92
RS_PROT:XP_508884  Begin: 25 End: 94
!PREDICTED: amyloid beta (A4) precursor-like protein 2 ..
RS_PROT:XP_856039  Begin: 310 End: 528
!PREDICTED: similar to Amyloid-like protein 2 precursor..   304   1e-80
RS_PROT:XP_856039  Begin: 42 End: 204
!PREDICTED: similar to Amyloid-like protein 2 precursor..
RS_PROT:XP_856039  Begin: 536 End: 612
!PREDICTED: similar to Amyloid-like protein 2 precursor..
RS_PROT:XP_870796  Begin: 27 End: 171
!PREDICTED: similar to Amyloid beta A4 protein precurso..   273   2e-71
RS_PROT:XP_577781  Begin: 236 End: 611
!PREDICTED: similar to Amyloid-like protein 1 precursor..   245   4e-63
RS_PROT:XP_577781  Begin: 14 End: 170
!PREDICTED: similar to Amyloid-like protein 1 precursor..
RS_PROT:XP_001076010  Begin: 180 End: 555

//////////////////////////////////////////////////////////////////////

RS_PROT:XP_790315  Begin: 145 End: 425
!PREDICTED: similar to Beta-amyloid-like protein precur..   112   7e-23
RS_PROT:XP_790315  Begin: 6 End: 119
!PREDICTED: similar to Beta-amyloid-like protein precur..
RS_PROT:XP_790315  Begin: 583 End: 625
!PREDICTED: similar to Beta-amyloid-like protein precur..
RS_PROT:XP_693670  Begin: 73 End: 129
!PREDICTED: similar to amyloid precursor protein [Danio..   101   9e-20
RS_PROT:XP_693670  Begin: 180 End: 252
!PREDICTED: similar to amyloid precursor protein [Danio..
RS_PROT:XP_512603  Begin: 682 End: 768
!PREDICTED: similar to kin of IRRE-like 2 isoform c; ne..    91   2e-16
RS_PROT:XP_512603  Begin: 786 End: 901
!PREDICTED: similar to kin of IRRE-like 2 isoform c; ne..
RS_PROT:XP_512603  Begin: 999 End: 1042
!PREDICTED: similar to kin of IRRE-like 2 isoform c; ne..
RS_PROT:NP_001006317  Begin: 530 End: 606
!amyloid beta (A4) precursor-like protein 2 [Gallus ...      82   8e-14
RS_PROT:XP_320745  Begin: 650 End: 719
!ENSANGP00000008856 [Anopheles gambiae str. PEST].           73   5e-11
RS_PROT:NP_058896  Begin: 53 End: 115
!tissue factor pathway inhibitor [Rattus norvegicus].        71   2e-10
RS_PROT:NP_058896  Begin: 222 End: 282
!tissue factor pathway inhibitor [Rattus norvegicus].
RS_PROT:NP_058896  Begin: 124 End: 180
!tissue factor pathway inhibitor [Rattus norvegicus].
RS_PROT:XP_524249  Begin: 133 End: 183
!PREDICTED: similar to serine protease inhibitor, Kunit..    70   2e-10
RS_PROT:XP_524249  Begin: 38 End: 90
!PREDICTED: similar to serine protease inhibitor, Kunit..

//////////////////////////////////////////////////////////////////////

RS_PROT:NP_064789  Begin: 27 End: 77
!putative peptidylglycine alpha-amidating monooxygenase..    55   8e-06
RS_PROT:NP_725647  Begin: 509 End: 559
!fat-spondin CG6953-PB, isoform B [Drosophila melanogas..    55   8e-06
RS_PROT:NP_477435  Begin: 644 End: 694
!fat-spondin CG6953-PA, isoform A [Drosophila melanogas..    55   8e-06
RS_PROT:XP_967522
```

```
!PREDICTED: similar to CG5639-PA [Tribolium castaneum].      55   8e-06
RS_PROT:XP_853242
!PREDICTED: similar to Collagen alpha 3(VI) chain precu..    55   8e-06

////////////////////////////////////////////////////////////////////

RS_PROT:XP_871958
!PREDICTED: similar to tissue factor pathway inhibitor ..    49   7e-04
RS_PROT:NP_505945
!Temporarily Assigned Gene name family member (tag-290)..    49   7e-04
RS_PROT:XP_001113422
!PREDICTED: similar to alpha 3 type VI collagen isof...      49   7e-04
RS_PROT:NP_499406
!W05B2.2 [Caenorhabditis elegans].                            49   0.001
RS_PROT:XP_699239
!PREDICTED: similar to WFIKKN2 protein [Danio rerio].        49   0.001
RS_PROT:NP_001008881
!trypsin protease inhibitor-like 4 [Rattus norvegicus].      48   0.001

////////////////////////////////////////////////////////////////////

RS_PROT:XP_814845
!DNA repair protein RAD50 [Trypanosoma cruzi strain CL ..    44   0.030
ZP_00679090
!Protease inhibitor I2, Kunitz metazoa [Pelobacter pr...     43   0.039
RS_PROT:XP_552480
!ENSANGP00000028836 [Anopheles gambiae str. PEST].           43   0.039
RS_PROT:XP_691668
!PREDICTED: similar to chromatin assembly factor 1, sub..    43   0.039

////////////////////////////////////////////////////////////////////

RS_PROT:XP_001108669
!PREDICTED: similar to epididymal trypsin inhibitor ...      41   0.20
RS_PROT:XP_686943
!PREDICTED: similar to restin [Danio rerio].                 41   0.20
RS_PROT:XP_814069
!hypothetical protein [Trypanosoma cruzi strain CL Bren..    41   0.20
RS_PROT:XP_723990
!hypothetical protein PY03788 [Plasmodium yoelii yoelii..    41   0.20
RS_PROT:XP_792448
!PREDICTED: similar to kielin/chordin-like protein 1 [S..    41   0.20

////////////////////////////////////////////////////////////////////

RS_PROT:XP_392064
!PREDICTED: similar to GA18234-PA [Apis mellifera].          40   0.33
RS_PROT:YP_205882
!hypothetical protein VF2499 [Vibrio fischeri ES114].        40   0.33
\\End of List

>RS_PROT:NP_000475 amyloid beta A4 protein precursor, isoform a [Homo sapiens].
          Length = 770

 Score = 1305 bits (3378), Expect = 0.0
 Identities = 646/752 (85%), Positives = 646/752 (85%)

Query: 19  EVPTDGNAGLLAEPQIAMFCGRLNMHMNVQNGKWDSDPSGTKTCIDTKEGILQYCQEVYP 78
           EVPTDGNAGLLAEPQIAMFCGRLNMHMNVQNGKWDSDPSGTKTCIDTKEGILQYCQEVYP
Sbjct: 19  EVPTDGNAGLLAEPQIAMFCGRLNMHMNVQNGKWDSDPSGTKTCIDTKEGILQYCQEVYP 78

Query: 79  ELQITNVVEANQPVTIQNWCKRGRKQCKTHPHFVIPYRCLVGEFVSDALLVPDKCKFLHQ 138
           ELQITNVVEANQPVTIQNWCKRGRKQCKTHPHFVIPYRCLVGEFVSDALLVPDKCKFLHQ
Sbjct: 79  ELQITNVVEANQPVTIQNWCKRGRKQCKTHPHFVIPYRCLVGEFVSDALLVPDKCKFLHQ 138

Query: 139 ERMDVCETHLHWHTVAKETCSEKSTNLHDYGMLLPCGIDKFRGVEFVCCPLXXXXXXXXX 198
           ERMDVCETHLHWHTVAKETCSEKSTNLHDYGMLLPCGIDKFRGVEFVCCPL
Sbjct: 139 ERMDVCETHLHWHTVAKETCSEKSTNLHDYGMLLPCGIDKFRGVEFVCCPLAEESDNVDS 198
```

```
Query: 199 XXXXXXXXXXWWGGADTDYADGSXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 258
                   WWGGADTDYADGS
Sbjct: 199 ADAEEDDSDVWWGGADTDYADGSEDKVVEVAEEEEVAEVEEEEADDDEDDEDGDEVEEEA 258

Query: 259 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXCSEQAETGPCRAMISRWYFDVTEGKCAP 318
                                        CSEQAETGPCRAMISRWYFDVTEGKCAP
Sbjct: 259 EEPYEEATERTTSIATTTTTTTESVEEVVREVCSEQAETGPCRAMISRWYFDVTEGKCAP 318

////////////////////////////////////////////////////////////////////

>RS_PROT:NP_741569 PaPiliN (Drosophila ECM protein) homolog family member (ppn-1)
           [Caenorhabditis elegans].
           Length = 1487

 Score = 68.6 bits (166), Expect = 9e-10
 Identities = 34/111 (30%), Positives = 49/111 (44%)

Query: 291  CSEQAETGPCRAMISRWYFDVTEGKCAPFFYGGCGGNRNNFDTEEYCMAVCGSAMSQSLL 350
            C E  +TGPC   +++WY++  +G C  F YGGC G  N FD E+ C A C +      L
Sbjct: 1376 CDEAKDTGPCTNFVTKWYYNKADGTCNRFHYGGCQGTNNRFDNEQQCKAACQNHKDACQL 1435

Query: 351  KTTQEPLARDPVKLPTTAASTPDAVDKYLETPGDENEHAHFQKAKERLEAK 401
               Q P +          AS       Y    G+ N  A  ++ + R  +K
Sbjct: 1436 PKVQGPCSGKHSYYYYNTASHQCETFTYGGCLGNTNRFATIEECQARCPSK 1486

Score = 58.2 bits (139), Expect = 1e-06
 Identities = 24/53 (45%), Positives = 35/53 (66%), Gaps = 2/53 (3%)

Query: 291  CSEQAETGP-CRAMIS-RWYFDVTEGKCAPFFYGGCGGNRNNFDTEEYCMAVC 341
            C++  E+G  C A     W++D TEG+C  F+YGGCGGN NNF +++ C  +C
Sbjct: 1089 CNQTQESGTVCGAGYKLAWHYDTTEGRCNQFWYGGCGGNDNNFASQDMCETIC 1141

Score = 52.4 bits (124), Expect = 7e-05
 Identities = 18/51 (35%), Positives = 28/51 (54%)

Query: 291  CSEQAETGPCRAMISRWYFDVTEGKCAPFFYGGCGGNRNNFDTEEYCMAVC 341
            C  + + GPC     +W+++    +C  F YGGCGGN N F +++ C   C
Sbjct: 1271 CRSRQDAGPCETYSDQWFYNAFSQECETFTYGGCGGNLNRFRSKDECEQRC 1321

Score = 49.7 bits (117), Expect = 4e-04
 Identities = 32/134 (23%), Positives = 60/134 (44%), Gaps = 20/134 (14%)

Query: 300  CRAMISRWYFDVTEGKCAPFFYGGCGGNRNNFDTEEYCMAVCGSAMSQSLLKTTQEPLAR 359
            C  +  R+Y+D ++  C  F++ GC GN NNF++ E C   C
Sbjct: 1161 CDQLQPRYYYDHSKKHCVAFWWRGCLGNANNFNSFEECSMFCKDV--------------- 1205

Query: 360  DPVKLPTTAASTP---DAVDKYLETPGDENEHAHFQKAKERLEAKHRERMSQVMREWEEA 416
             P   PTTAA  P       +YL TP  E +    Q A++    + +++  Q  ++ ++
Sbjct: 1206 GPYDAPTTAAPPPPQQNAQQYLPTP--EVQQIEIQSAEQPQPQQPQQQQQQQQQQPQQP 1263

Query: 417  ERQAKNLPKADKKA 430
             +  +++ ++ + A
Sbjct: 1264 RQSMEDICRSRQDA 1277


>RS_PROT:NP_954518 serine protease inhibitor, Kunitz type 2 [Rattus norvegicus].
          Length = 193

 Score = 68.2 bits (165), Expect = 1e-09
 Identities = 26/51 (50%), Positives = 35/51 (68%)

Query: 291 CSEQAETGPCRAMISRWYFDVTEGKCAPFFYGGCGGNRNNFDTEEYCMAVC 341
           C  +A TGPCRA   RWY+DV +  C+ F YGGC GN+N++ ++E CM  C
Sbjct: 74  CVPKAVTGPCRAAFPRWYYDVEKNSCSSFIYGGCRGNKNSYLSQEACMQHC 124

////////////////////////////////////////////////////////////////////

>RS_PROT:NP_990865 collagen, type VI, alpha 3 [Gallus gallus].
          Length = 3137
```

```
 Score = 62.0 bits (149), Expect = 8e-08
 Identities = 23/51 (45%), Positives = 32/51 (62%)

Query: 291  CSEQAETGPCRAMISRWYFDVTEGKCAPFFYGGCGGNRNNFDTEEYCMAVC 341
            C  Q E G CR  + +W++D+    CA F+YGGCGGN N F+T++ C   C
Sbjct: 3072 CLLQKEEGTCRDFVLKWHYDLKTKSCARFWYGGCGGNENRFNTQKECEKAC 3122


>RS_PROT:NP_037033 polyprotein 1-microglobulin/bikunin [Rattus norvegicus].
          Length = 349

 Score = 61.6 bits (148), Expect = 1e-07
 Identities = 25/52 (48%), Positives = 29/52 (55%)

Query: 291 CSEQAETGPCRAMISRWYFDVTEGKCAPFFYGGCGGNRNNFDTEEYCMAVCG 342
           C+     GPCRA    W FD  +GKC  F YGGC GN N F +E+ C   CG
Sbjct: 286 CNLPIVQGPCRAFAELWAFDAAQGKCIQFIYGGCKGNGNKFYSEKECKEYCG 337

Score = 49.7 bits (117), Expect = 4e-04
 Identities = 20/51 (39%), Positives = 27/51 (52%)

Query: 291 CSEQAETGPCRAMISRWYFDVTEGKCAPFFYGGCGGNRNNFDTEEYCMAVC 341
           C      GPC  M  ++Y++     C  F YGGC GN NNF +E+ C+  C
Sbjct: 230 CQLNYSEGPCLGMQQKYYYNGASMACETFQYGGCLGNGNNFASEKECLQTC 280

//////////////////////////////////////////////////////////////////////

>RS_PROT:NP_477435 fat-spondin CG6953-PA, isoform A [Drosophila melanogaster].
          Length = 763

 Score = 55.5 bits (132), Expect = 8e-06
 Identities = 23/51 (45%), Positives = 28/51 (54%)

Query: 291 CSEQAETGPCRAMISRWYFDVTEGKCAPFFYGGCGGNRNNFDTEEYCMAVC 341
           C +  + GPCR    R+ +D     C  F YGGC GNRNNF TE  C+  C
Sbjct: 644 CVQAPDPGPCRGTYMRYAYDPQNQHCYSFTYGGCRGNRNNFLTENDCLNTC 694


  Database: /newdisk/raja/gcgblast/rs_prot
    Posted date:  Jul 21, 2006  4:41 PM
  Number of letters in database: 974,374,765
  Number of sequences in database:  2,762,164

Lambda     K      H
   0.317    0.132    0.398

Gapped
Lambda     K      H
   0.267   0.0410    0.140


Matrix: BLOSUM62
Gap Penalties: Existence: 11, Extension: 1
Number of Hits to DB: 1,237,085,072
Number of Sequences: 2762164
Number of extensions: 48275617
Number of successful extensions: 171084
Number of sequences better than  1.0: 500
Number of HSP's better than  1.0 without gapping: 370
Number of HSP's successfully gapped in prelim test: 316
Number of HSP's that attempted gapping in prelim test: 169157
Number of HSP's gapped (non-prelim): 1730
length of query: 770
length of database: 974,374,765
effective HSP length: 136
effective length of query: 634
effective length of database: 598,720,461
effective search space: 379588772274
```

```
effective search space used: 379588772274
T: 11
A: 40
X1: 16 ( 7.3 bits)
X2: 38 (14.6 bits)
X3: 64 (24.7 bits)
S1: 41 (21.7 bits)
S2: 88 (38.5 bits)
```

BLAST (but not NetBLAST) output is also a GCG format list file, complete with beginning and ending attributes for those pairwise alignments specified (and complementary strand attributes when necessary if using DNA). This particular list is much longer, and the scores fall off much more slowly, than the previous Fast program examples, because we searched all of RefSeqProt, by far the largest dataset we searched today. However, you should still be able to see somewhat of a demarcation where the scores drop off between the significant A4 hits, BPTI domains, and background noise. There's just a lot more of each, especially the BPTI domain hits!

Several features of BLAST should be noted in this output. First off, notice that some sequences have multiple alignment segments. This can be seen both in the output's list section and in the pairwise alignment portion, where each segment's individual E value is shown. This is particularly helpful in cases of multiple domain proteins, or where the database entry is from genomic DNA and has several dispersed exons each with separate homologies to your query. Secondly, notice that BLAST screened out a long stretch of "low-complexity" sequence in our query. This can plainly be seen in the best pairwise alignment above, SW:A4_Human against itself in RefSeqProt, NP_000475. The E value is 0.0, but the identity is 85%, due to the fact that the "X" symbols in the query don't match the database entry. BLAST represents identities and similarities differently than the Fast programs do in the pairwise section. It uses the actual amino acid symbol rather than the pipe symbol (|) to represent identities, and the plus sign (+) rather than a colon (:) to denote similarities. Finally, from all three searches, notice that BPTI domain sequences seem to be mixed in with all sorts of other things including lots of collagens and kinesins. We'll check this out in the next section.

Interpreting results—what is significant — beyond Expectation values alone

We know how the A4 sequence aligns to itself and other close homologues; we know those alignments are significant. They don't cause anybody any problems; they're obvious. Therefore, we'll pick a few sequences where the similarity is not so obvious for the remainder of the tutorial. These 'interesting' sequences, not recognizably from the same gene family as A4, will still have somewhat decent scores, but not great. We want to see how some of the not so similar, as Russell Doolittle calls them 'twilight zone,' sequences, align to our query and the significance of those alignments. This often happens with a newly sequenced piece of DNA — you suspect that it encodes for something, yet the best E values you can find in any type of a protein based database similarity search are pretty mediocre, around 0.1 to 0.001. What do E values in this range 'really' mean?

Therefore, I'll choose three completely different 'twilight zone' entries, one each, from each program run, TFastX, FastA, and, BLAST, with mediocre scores, yet among the higher of the dissimilar matches, that do not obviously belong to the same gene family as the A4 protein. Work through my examples, or pick a couple of your own. Select different types of proteins for each, e.g. only choose one obvious BPTI homolog. We just want to see

some interesting comparisons so that we can experiment with various methods of analyzing similarity significance beyond E values. Relevant lines showing my three choices from the search files follow.

From my TFastX search of the non-Chordate Metazoans in RefSeqNuc I chose a non-A4 sequence that turns out to have BPTI domain in spite of its marginal Expectation value of 10^{-1}:

```
RS_RNA:NM_134950    Begin: 236  End: 448
! Drosophila melanogaster CG15418-RA ...(f)    97    97   144   152.9   0.099
```

And from the FastA output, something completely different, a kinesin, with an Expectation value of 5×10^{-2}:

```
UNI_SPROT:KINH_HUMAN    Begin: 483  End: 839
! P33176 homo sapiens (human). kinesi...   42    42   167   144.0   0.047
```

From the BLASTP search of RefSeqProt I chose the second from the bottom of the list, a predicted honeybee protein, with the nearly insignificant Expectation value of 3×10^{-1}:

```
RS_PROT:XP_392064
!PREDICTED: similar to GA18234-PA [Apis mellifera].          40   0.33
```

Load my choices, and/or a couple of your own, perhaps one of those collagens (don't bother with another RefSeqNuc sequence, for time's sake), into SeqLab by going to the "**File**" "**Add sequences from**" "**Databases. . .**" menu. Type the name of the entry, including its database logical name and colon separator (`LogicalName:EntryName`), in the "**Database Specification:**" box of the "**SeqLab Database Browser**" and then press the "**Add to Main Window**" button. "**Close**" the browser box after adding the sequences. Double-click on each new entry's name, or use the "**INFO**" icon with the sequence's name selected, to read each sequence's database annotation.

An additional step is necessary for the RefSeqNuc sequence — it's DNA. I'll need to translate the protein-coding region in it that corresponds to the region discovered by the search algorithm. The easiest way to do this is to use the sequence's CDS (coding sequence) feature annotation, if it has any and it makes sense. My NM_134950 example is an easy one — there's only one CDS listed in the entry — it should be the one that was found to be similar to A4, unless the similarity isn't 'real.' Double-click anywhere within the sequence to launch the "**Sequence Features**" window, just as at the beginning of the tutorial, however, this time switch the features being displayed from "**Show:**" "**Features at the cursor**" to "**All features in current sequence**." This will allow you to scroll through the entire sequence's feature list and select any CDSs that are relevant. In my NM_134950 case there's only one: "`CDS  176. .469  /gene="CG15418"`."

> However, if the sequence came from eukaryotic genomic DNA, then you would have to choose every CDS of each exon associated with the particular gene that was found to be similar to your query then translate to concatenate all of the selected CDS regions. Don't select "mRNA" or "exon" features — UTR's and/or splicing variants can mess things up. Furthermore, if a nucleotide database entry isn't annotated with CDS feature data, such as in most of the Tags databases, then you would have to translate the entry using some other criteria. TBLASTN and TFastX outputs list beginning and ending attributes in the list portion of the file (unless suppressed) and they indicate whether the similarity was

found on the forward or reverse strand. One way to translate just the desired regions would be to select the DNA sequence, and then use the "Edit" "Select Range. . ." function to select just that portion identified by the search algorithm. Then you could go to the "Edit" "Translate. . ." button and translate the selected region of the sequence to produce a translation of only that segment. However, if the sequence similarity was found on the reverse strand, you would need to use the "Edit" "Reverse. . ." menu to specify "Reverse and Complement" and then use the opposite strand's numbering scheme before doing this.

Select the relevant CDS region in my example. Go to the "**Edit**" menu and select "**Translate. . .**" Choose "**Selected regions**" rather than "Selected sequences" when prompted in the "**Which selections**" window, and then press "**OK**" in the next window. The new protein sequence will appear at the bottom of your SeqLab Editor display, and now we're ready to start some in-depth pairwise comparisons. Go up to SeqLab's "**File**" menu and "**Save As . . .**" the dataset; give it a name that makes sense like A4, but leave the "**`.rsf`**" extension.

6) Dot matrix methods. Compare and DotPlot: the GCG implementation of dot matrix analysis

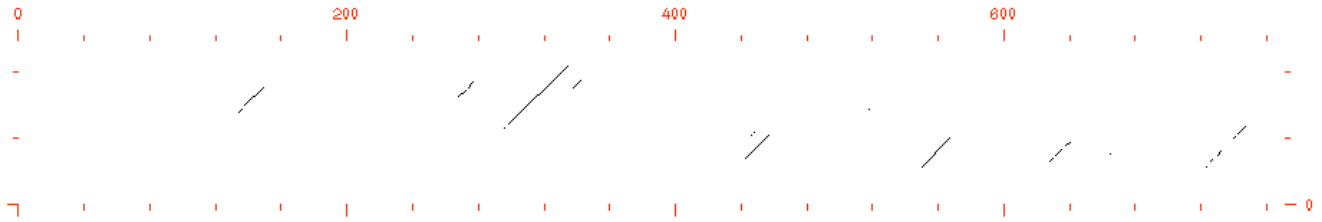
Dot matrix analysis is one of the few ways to identify other elements beyond what dynamic programming algorithms show to be similar between two sequences. It provides a 'Gestalt' of all the possible alignments between two sequences. GCG implements dot matrix methods with two programs. Compare generates the data that serves as input to DotPlot, which actually draws the matrix. Compare the A4 sequence to each of my three 'twilight zone,' neighbors (as described above) using these methods. (In general, put the longer sequence along the horizontal axis of the final dotplot by having it first in the SeqLab display. Dotplots just look better that way, though it is not necessary.)

I'll run the programs three times, once comparing the human A4 sequence to my interesting non-family member found by TFastX (don't use the DNA sequence: use the newly translated protein product), once to an analogous find by FastA, and finally another to a match found by regular BLASTP. You can do all of these as you follow along, or just do the couple of sequences that you've chosen — it's completely up to you and how much time is left in the session. After all, this is just practice, don't get too serious!

Start the programs by selecting "**A4_HUMAN**" and each new entry, one per program run at a time, pairwise (select nonadjacent entries with the <**ctrl**> key and [only in Linux implementations, like here at FSU, the <**right-click**>] mouse button) in the SeqLab Editor display. Next go to the "**Functions**" menu and select "**Pairwise Comparison**" "**Compare. . .**" to produce a Compare program window. Notice that "**DotPlot. . .**" is checked by default; this takes the output from Compare and automatically passes it to DotPlot, so the graphic will be drawn after the "**Run**" button is punched automatically. This will run the program at the GCG protein stringency default of 10 points within a window of 30 residues. Just as in all windowing algorithms, you want to use a window size of approximately the same size as the feature that you're trying to recognize. You can leave the window at its default setting of 30 for these runs, unless one of your sequences is too short for this size of a window to find much, in which case you should reduce the window size appropriately. To clean up the graph, rerun the program increasing the stringency of the comparisons until the number of points generated is of the same order of

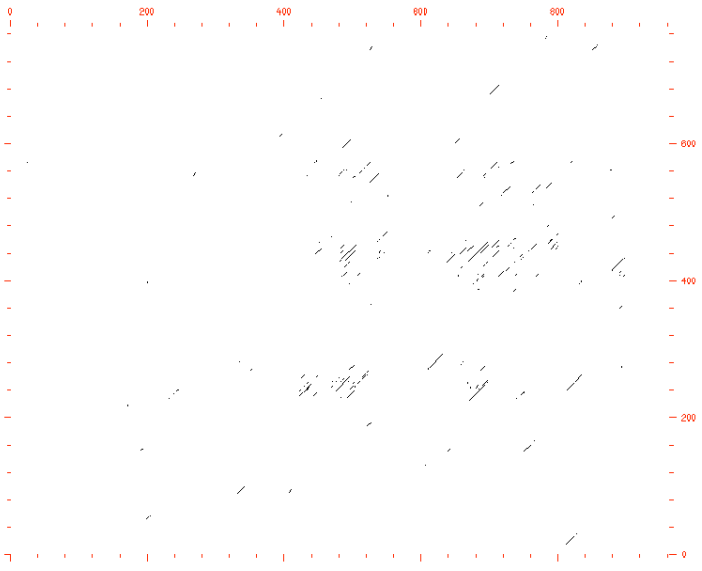
magnitude as the length of the longest sequence being compared. This, and changing the window size, is done through the "**Options**" menu.

Below, in my first example, human A4 against the translated product of my RefSeqNuc 'interesting' hit, I found that the default parameters worked OK and resulted in 126 points — somewhere between the two sequences' lengths being compared. In this case, wherever the sum of BLOSUM62 match scores within the 30 residue window is equal to or exceeds 10, a point will be drawn at the middle of the window, then the window will be slid over one position at which point the process is repeated. When run at this stringency the graphic looks like the following plot with the human A4 sequence along the horizontal axis and the fruit fly sequence vertical:



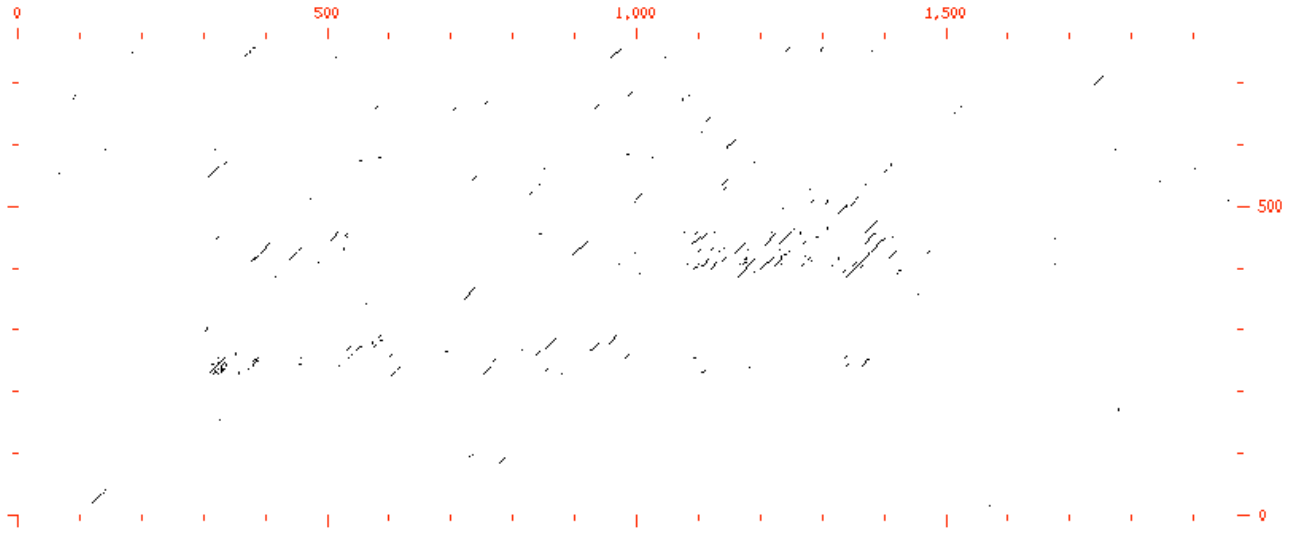
Notice that running the comparison at an appropriate stringency, in this case the default, produces a clean plot with little confusing noise. Obvious are several strong diagonals that clearly show alignments across various portions of the fruit fly sequence. The biggest is centered around residue 300 through 340 in the A4 sequence. This corresponds to the BPTI/Kunitz domain of A4. There is little doubt about the significance of this alignment, even though its BLASTP E value is only10^{-1} — the translated product of NM_134950 has a BPTI/Kunitz domain. Still, sometimes interpreting a dotplot can be a major accomplishment in itself — just remember that diagonals are regions of similarity between the two sequences and that any diagonal off the main center line is indicative of regions that do not correspond in linear placement between the two sequences yet are still similar. Columns or rows of multiple diagonals always mean direct repeat sequences. Here there are four repeated diagonals in human A4 that correspond to a region in the fruit fly sequence from around residue 20 through 40. The BPTI domain also has a weak repeat upstream of the main diagonal at about residue 120 through 150 in A4.

My second example, human A4 against human kinesin, greatly contrasts with the clear similarity seen above. My first run at the default stringency resulted in a very messy dotplot with 5,305 points. Therefore, I increased the stringency up to 20 within the default window of 30 and reran the program to produce a nice clean plot with 770 points. The A4 sequence is now along the vertical axis while kinesin is along the horizontal. The dotplot graphic is shown here to the right. Even though it is a 'clean' plot there's not much to see:



The similarity is largely restricted to several weak areas of overlapping direct repeats which can be seen as columns and rows of diagonals — various stretches of residues in the region of 200 through 600 of A4 repeat themselves several times in a region of kinesin between residues 400 through 800. However, none of these repeats occur around A4's position 320, where that BPTI domain sits. The dotplot hints that even though FastA returned an E value of 5 x 10^{-2} for this alignment, the best of my 'interesting' hits, it is likely not significant.

I returned all the running parameters back to their defaults by pressing the "**GCG Defaults**" button, and then analyzed my A4 sequence against the honeybee sequence, XP_392064, that I picked from our BLAST search of RefSeqProt. The default stringency again returned an extremely noisy plot, so I increased the stringency back to 20 within a 30-mer window to produce the plot below. The human A4 sequence is again along the vertical axis:



Not much is obvious here either. There's a region in A4 between residues 400 to 450, that repeats itself several times along the length of the bee sequence, but it's pretty messy and not very long. And it's certainly not in the BPTI domain of A4. When running all the dotplots, take notes of those particular regions in the proteins that have the strongest diagonals, and the protein's corresponding names in SeqLab. For example, as noted in the above plot, the region of A4_Human from about residue 400 through 450 has similarities to itself in XP_392064 from around residue 1350 through 1400. We will need these numbers and names in the next section.

7) The pairwise dynamic programming alignment algorithms: use the right one for the right job — Gap, BestFit, and FrameAlign

You need to understand the difference between these algorithms! Gap is a 'global' alignment program, BestFit is 'local,' both between two sequences of the same type, whereas FrameAlign can be global or local depending on the options that you set, but it always aligns DNA to protein. Using one versus the other implies that you are looking for distinctly different relationships. Know what they mean. If you already know that the full length of two sequences of the same type are pretty close, that they probably belong to the same family, then Gap is a good choice; if you only suspect an area of one is similar to an area of another, then you should use BestFit. To force BestFit to be even more local, you can specify a more stringent alternative symbol comparison table, such as

`pam250.cmp` or `blosum100.cmp`. If you suspect that a DNA sequencing error is affecting the alignment, then FrameAlign is the program to use. All three programs can generate 'gapped' output files in standard sequence format as an option; this can be handy as direct input to other GCG routines, such as multiple sequence analysis.

BestFit and Gap: how to use them to estimate significance

What is significant? GCG provides an easy way to estimate significance beyond search algorithm Expectation values in Gap and BestFit. When running them specify the –Randomizations=100 option and the second input sequence will be jumbled a 100 times after the initial alignment is produced (unfortunately GCG doesn't support 1,000 jumbles). Comparing the quality scores of the randomized alignments to the initial alignment can help you get a feeling for the relative meaning of the scores. This distance above the mean is often known as a "Z score" and is calculated with the formula given in the Introduction on page ten.

This type of significance analysis is known as a Monte Carlo approach, because of the shuffling aspect. It has many implicit statistical problems, but few practical alternatives exist for just comparing two sequences. I will use the BestFit –Randomizations option with A4_Human to further analyze my three 'interesting' proteins to illustrate the concept. Before beginning though, study your dotplot notes from before. This approach works best when applied to local areas where you already know some similarity exists and you wish to further test that similarity, otherwise you are just throwing extra noise into the analysis, which inflates the alignment's apparent significance. Therefore, restrict your analyses to those regions identified by the dotplots. However, remember that dotplots show us all the regions that are similar, whereas dynamic programming only gives us one optimal solution.

Unfortunately SeqLab's Editor will not allow us to choose two different ranges on two different sequences. Some things are still simpler from the command line. In lieu of switching to the command line, you could create new spaces to hold duplicate sequence data through the "File" "New Sequence. . ." "Protein" menu. But you would have to repeat the procedure six times to create spots for the three pairwise comparisons, and you would need to rename the newly created sequences so that you could recognize what they are through the "INFO" icon, changing their name fields. Then you would need to copy and paste all of the desired subsequences into the new spots. The "Text clipboard" holds portions of a sequence, rather than an entire entry including its references. After creating all the new subsequences, then you could use the "Functions" "Pairwise Comparison" "BestFit. . ." "Options" button to take advantage of "Generate statistics from randomized alignments" changing the "Number of randomizations" up to "100." What a pain!

Therefore, exit SeqLab now with the "**File**" menu "**Exit**" choice; we won't need it any more in the workshop. However, we do need the protein translation that we generated from the RefSeqNuc sequence, so save your RSF file again, if prompted; use the same RSF file name as before and "**Overwrite**" the previous version. Also save your working list with the appropriate response. SeqLab will close. Next use your ssh terminal window to manually launch BestFit –Randomizations. Designate the A4_Human sequence from SwissProt and each of the three protein sequences in turn, along with each respective region from the dotplots. The region constraints are supplied with –Begin1, –Begin2, –End1, and –End2 options, or you can interact with the program to supply the length restrictions when prompted. The RefSeqProt and SwissProt entries are quite straightforward; the

RefSeqNuc translation is a bit more complicated. Notice the peculiar specification for the RSF file — you need to use both the RSF file name and the set of braces with the RefSeqNuc's translated sequence's name inside. There are spaces between each parameter and GCG logical name database specifications are case insensitive, but file names and commands are case sensitive. Follow my examples or use your own. Accept the program defaults, but specify output file names that make sense to you. I'll provide command lines for my examples here:

```
> bestfit -random=100 -begin1=290 -end1=340 -begin2=40 -end2=90 sw:a4_human
A4.rsf{NM_134950_frame1}
> bestfit -random=100 -begin1=200 -end1=600 -begin2=400 -end2=800 sw:a4_human
sw:kinh_human
> bestfit -random=100 -begin1=400 -end1=450 -begin2=1350 -end2=1400 sw:a4_human
rs_prot:xp_392064
```

You'll end up with three BestFit output files, if you do all my examples. These will show those best local portions of A4 aligned to those best local regions of my three 'interesting' comparisons, as localized in the previous dotplots. My first output file, the BPTI/Kunitz domain of human A4 against a seemingly homologous region in my translated RefSeqNuc fruit fly sequence, is shown below:

```
BESTFIT of: a4_human  check: 2126  from: 290  to: 340

ID   A4_HUMAN       STANDARD;      PRT;   770 AA.
AC   P05067; P09000; P78438; Q13764; Q13778; Q13793; Q16011; Q16014;
AC   Q16019; Q16020; Q9BT38; Q9UCA9; Q9UCB6; Q9UCC8; Q9UCD1; Q9UQ58;
DT   13-AUG-1987, integrated into UniProtKB/Swiss-Prot.
DT   01-NOV-1991, sequence version 3.
DT   25-JUL-2006, entry version 119. . . .

 to: 2006.rsf{NM_134950_frame1}  check: 516  from: 40  to: 90


 Symbol comparison table: /usr/local/gcg/share/matrix/blosum62.cmp
 CompCheck: 1102

         Gap Weight:      8      Average Match:  2.778
      Length Weight:      2   Average Mismatch: -2.248

            Quality:     88             Length:     47
              Ratio:  1.872               Gaps:      0
 Percent Similarity: 42.553   Percent Identity: 36.170

 Average quality based on 100 randomizations: 22.9 +/- 4.0

        Match display thresholds for the alignment(s):
                    | = IDENTITY
                    : =   2
                    . =   1

 a4_human x 2006.rsf{NM_134950_frame1} November 10, 2006 12:32  ..

                  .         .         .         .
     291 CSEQAETGPCRAMISRWYFDVTEGKCAPFFYGGCGGNRNNFDTEEYC 337
         | :    | ||    |: :.   | |  | | ||    ||| | | |
      41 CRQPKAPGLCRGHQLRYAYNKKTGNCESFIYTGCASTENNFLTFEEC 87
```

Notice that the similarity is spread over the entire length analyzed. Also notice the value of the original quality compared to the randomized quality. The Z score calculates to be 16.3. Therefore, the interpretation is that the similarity is extremely significant, in spite of the two regions only having 36% identity, and its poor TFastX

Expectation value of 10^{-1}. This alignment clearly shows how percent identities and/or similarities can be quite misleading. However, often a seemingly decent alignment will not be significant upon further inspection — do not blindly accept the output of any computer program! Always investigate further for similarities can be strictly artifactual. The second comparison that I chose, that big 'smear' of an area of A4 against the kinesin protein, turned out to be entirely insignificant in spite of what seemed, upon first inspection, to be an OK FastA Expectation score, 5×10^{-2}. From the previous dotplot analysis, I saw that the region of A4 between residues 200 through 600 has several short overlapping direct repeats in kinesin between residues 400 through 800. The Monte Carlo analysis of this regional comparison is shown below. The Z score is 3.3, just below Doolittle's "Twilight Zone" and generally considered to be insignificant:

```
BESTFIT of: a4_human  check: 2126  from: 200  to: 600

ID   A4_HUMAN       STANDARD;      PRT;   770 AA.
AC   P05067; P09000; P78438; Q13764; Q13778; Q13793; Q16011; Q16014;
AC   Q16019; Q16020; Q9BT38; Q9UCA9; Q9UCB6; Q9UCC8; Q9UCD1; Q9UQ58;
DT   13-AUG-1987, integrated into UniProtKB/Swiss-Prot.
DT   01-NOV-1991, sequence version 3.
DT   25-JUL-2006, entry version 119. . . .

 to: kinh_human  check: 5105  from: 400  to: 800

ID   KINH_HUMAN     STANDARD;      PRT;   963 AA.
AC   P33176; Q5VZ85;
DT   01-OCT-1993, integrated into UniProtKB/Swiss-Prot.
DT   01-OCT-1993, sequence version 1.
DT   11-JUL-2006, entry version 57.
DE   Kinesin heavy chain (Ubiquitous kinesin heavy chain) (UKHC). . . .

 Symbol comparison table: /usr/local/gcg/share/matrix/blosum62.cmp
 CompCheck: 1102

         Gap Weight:      8      Average Match:  2.778
      Length Weight:      2   Average Mismatch: -2.248

            Quality:     89             Length:    336
              Ratio:  0.285              Gaps:     15
 Percent Similarity: 33.550   Percent Identity: 25.733

 Average quality based on 100 randomizations: 60.0 +/- 8.8

        Match display thresholds for the alignment(s):
                    | = IDENTITY
                    : =   2
                    . =   1

 a4_human x kinh_human     November 10, 2006 13:02  ..

                  .         .         .         .         .
     229 AEEEEVAEV.EEEEADDDEDDEDGDEVEEEAEEPYEEATERTTSIATTTT 277
         | .||| || :  |      |:   |||:. .| ||  .:     . |
     483 ASKEEVKEVLQALEELAVNYDQKSQEVEDKTKE.YELLSDELNQKSATLA 531
                  .         .         .         .         .
     278 TTTESVEEVVREVCSEQAETGPCRAMISRWYFDVTEGKCAPFFYGGCGGN 327
         .    .:. .:|. . | .      |..    |. |        |   ||
     532 SIDAELQK.LKEMTNHQKKR..AAEMMASLLKDLAE.......IGIAVGN 571
                  .         .         .         .         .
     328 RNNFDTEEYCMAVCGSAMSQSLLKTTQEPLARDPVKLPTTAASTPDAVDK 377
          |.    |    :        |  .  .   :  ||      ||    .|
     572 .NDVKQPEGTGMIDEEFTVARLYISKMKSEVKTMVKRCKQLESTQTESNK 620
                  .         .         .         .         .
     378 YLETPGDENEHAHFQKAKERLEAKHRERMSQVMREWEEAERQAKNLPKAD 427
          :|   .| | |  |    . ||| :  :.: :.  |: .|| .    |
```

```
621 KMEE..NEKELAACQLRISQHEAKIKS.LTEYLQNVEQKKRQLEESVDAL 667
              .         .         .         .         .
428 KKAVIQ.HFQEKVESLEQEAANERQQLVETHMARVEAMLNDRRRLALENY 476
     . .:|   ||||  :|.|  |. |   |   | ||  :   |    :
668 SEELVQLRAQEKVHEMEKEHLNKVQTANEVKQA.VEQQIQSHRE.THQKQ 715
              .         .         .         .         .
477 ITALQAVPPRPRHVFNMLKKYVRAEQKDRQHTLKHFEHVRMVDPK.KAA. 524
    |..|       |       | :   |   |  :   | .|.   | ||
716 ISSL.......RDEVEAKAKLITDLQDQNQKMMLEQERLRVEHEKLKATD 758
              .         .         .
525 QIRSQVMTHLRVIYERMNQSLSLLYNV.PAVAEEIQ 559
    | :|. :  | |. :|  |.   |  .   ||.|:|
759 QEKSRKLHELTVMQDRREQARQDLKGLEETVAKELQ 794
```

It appears to be a decent alignment; it's quite long, and the percent similarity isn't all that worse than the previous example. However, the Monte Carlo test suggests that this similarity is not at all significant; it's merely the result of compositional bias. The program mainly found both regions to be very glutamate rich. This low-complexity, repeat type of sequence (that BLAST filters out by prerunning Seq and Xnu) can cause huge problems. The programs Xnu and Seg are available outside of BLAST for prefiltering your sequences. This is particularly prudent in situations such as the above with molecules where you know that a lot of repeat and/or low complexity sequence composition has the potential to confound search algorithms.

The final BestFit output from my examples follows next. Its Z score of 3.9 also argues for marginal to no significance, though it is more clearly in Doolittle's "Twilight Zone." This alignment, which received the BLAST E value of 3×10^{-1}, is most likely a sequence composition artifact also:

```
BESTFIT of: a4_human  check: 2126  from: 400  to: 450

ID   A4_HUMAN       STANDARD;      PRT;   770 AA.
AC   P05067; P09000; P78438; Q13764; Q13778; Q13793; Q16011; Q16014;
AC   Q16019; Q16020; Q9BT38; Q9UCA9; Q9UCB6; Q9UCC8; Q9UCD1; Q9UQ58;
DT   13-AUG-1987, integrated into UniProtKB/Swiss-Prot.
DT   01-NOV-1991, sequence version 3.
DT   25-JUL-2006, entry version 119. . . .

 to: xp_392064  check: 4330  from: 1350  to: 1400

LOCUS       XP_392064               1971 aa            linear   INV 24-MAY-2005
DEFINITION  PREDICTED: similar to GA18234-PA [Apis mellifera].
ACCESSION   XP_392064
VERSION     XP_392064.2  GI:66525100
DBSOURCE    REFSEQ: accession XM_392064.2
KEYWORDS    . . . .

 Symbol comparison table: /usr/local/gcg/share/matrix/blosum62.cmp
 CompCheck: 1102

        Gap Weight:      8      Average Match:  2.778
     Length Weight:      2   Average Mismatch: -2.248

           Quality:     51             Length:     47
             Ratio:  1.085              Gaps:      0
 Percent Similarity: 40.426   Percent Identity: 21.277

 Average quality based on 100 randomizations: 29.8 +/- 5.4

       Match display thresholds for the alignment(s):
                   | = IDENTITY
                   : =   2
                   . =   1
```

```
a4_human x xp_392064      November 10, 2006 13:44  ..

                .         .         .         .
  401 KHRERMSQVMREWEEAERQAKNLPKADKKAVIQHFQEKVESLEQEAA 447
      ||  .: :|  : :   :: . |    :| ..:|   |.. ||:  |
 1353 KHSRQLEEVRAQSDATIKELEQLQNEKRKMLMEHETLKLKELEEAYA 1399
```

If you suspect a frame shift sequencing error in a DNA sequence being considered, a very powerful pairwise alignment program, FrameAlign, is available, but we will not be running it today. This program uses dynamic programming to align a protein to a DNA sequence with the allowance of frame shifts. Frame shift errors will appear in the pair output alignment as gaps that are not multiples of three.

8) Conclusions — do you have any?

At the conclusion of tonight's computer laboratory you can either log out of Mendel and the Classroom workstation or leave the computers as they are with the terminals active. If you don't log out, I'll go around and log you out and clean up any loose ends. I hope you've learned that there's a whole lot more to database searching than just NCBI's Web BLAST service in this evening's workshop. It's been a very long one; I apologize. However, database searching is one of the most commonly misunderstood areas in bioinformatics today. There is a tremendous amount of confusion in this field, and anything that can be done to try and clear up some of the mess is entirely worthwhile. I still need to make one point before quitting though: the previous techniques were performed largely using GCG's suggested defaults. This will usually work, but it's a good idea to think about what these default values imply and adjust them accordingly, especially if the results seem inappropriate after running through a first pass with the default parameters intact. For more help in this area and for personal sequence analysis consultation, contact me for more information: Steve Thompson, stevet@bio.fsu.edu.

References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990) Basic Local Alignment Tool. *Journal of Molecular Biology* **215**, 403-410.

Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: a New Generation of Protein Database Search Programs. *Nucleic Acids Research* **25**, 3389-3402.

Bairoch A. (1992) PROSITE: A Dictionary of Sites and Patterns in Proteins. *Nucleic Acids Research* **20**, 2013-2018.

Eddy, S.R. (1996) Hidden Markov models. *Current Opinion in Structural Biology* **6**, 361-365.

Eddy, S.R. (1998) Profile hidden Markov models. *Bioinformatics* **14**, 755-763.

Etzold, T. and Argos, P. (1993) SRS — an Indexing and Retrieval Tool for Flat File Data Libraries. *Computer Applications in the Biosciences* 9, 49–57.

Genetics Computer Group (GCG), a part of Accelrys, Inc., (Copyright 1982-2006) *Program Manual for the Wisconsin Package*, Version 11.1, San Diego, California, U.S.A.

Gribskov, M. and Devereux, J., editors (1992) *Sequence Analysis Primer*. W.H. Freeman and Company, New York, N.Y., U.S.A.

Henikoff, S. and Henikoff, J.G. (1992) Amino Acid Substitution Matrices from Protein Blocks. *Proceedings of the National Academy of Sciences U.S.A.* **89**, 10915-10919.

Maglott, D., Ostell, J., Pruitt, K.D., and Tatusova, T. (2005) Entrez Gene: Gene-Centered Information at NCBI. *Nucleic Acids Research* **33**, 54-58.

Needleman, S.B. and Wunsch, C.D. (1970) A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of Molecular Biology* **48**, 443-453.

Pearson, P., Francomano, C., Foster, P., Bocchini, C., Li, P., and McKusick, V. (1994) The Status of Online Mendelian Inheritance in Man (OMIM) medio 1994. *Nucleic Acids Research* **22**, 3470-3473.

Pearson, W.B. (1998) Empirical Statistical Estimates for Sequence Similarity Searches. *Journal of Molecular Biology* **276**, 71–84.

Pearson, W.R. and Lipman, D.J. (1988) Improved Tools for Biological Sequence Analysis. *Proceedings of the National Academy of Sciences U.S.A.* **85**, 2444-2448.

Smith, S.W., Overbeek, R., Woese, C.R., Gilbert, W., and Gillevet, P.M. (1994) The Genetic Data Environment, an Expandable GUI for Multiple Sequence Analysis. *Computer Applications in the Biosciences* **10**, 671-675.

Schwartz, R.M. and Dayhoff, M.O. (1979) Matrices for Detecting Distant Relationships. In *Atlas of Protein Sequences and Structure*, (M.O. Dayhoff editor) **5**, Suppl. **3**, 353-358, National Biomedical Research Foundation, Washington D.C., U.S.A.

Smith, T.F. and Waterman, M.S. (1981) Comparison of Bio-Sequences. *Advances in Applied Mathematics* **2**, 482-489.

Sundaralingam, M., Mizuno, H., Stout, C.D., Rao, S.T., Liedman, M., and Yathindra, N. (1976) Mechanisms of Chain Folding in Nucleic Acids. The Omega Plot and its Correlation to the Nucleotide Geometry in Yeast tRNAPhe1. *Nucleic Acids Research* **10**, 2471-2484.

Wilbur, W.J. and Lipman, D.J. (1983) Rapid Similarity Searches of Nucleic Acid and Protein Data Banks. *Proceedings of the National Academy of Sciences U.S.A.* **80**, 726-730.

Zuker, M. (1989) On Finding All Suboptimal Foldings of an RNA Molecule. *Science* **244**, 48-52.