Biomolecular Sequence Alignment and Analysis:

Part II — Database Searching, Pairwise Comparisons, and Multiple Sequence Alignment.

A GCG¥ Wisconsin Package SeqLab Tutorial for the Woods Hole Oceanographic Institution.

c.o. Timothy M. Shank, PhD
MS 33 Redfield
Biology Department
Woods Hole Oceanographic Institution
Woods Hole, MA 02543
Tel. 508-289-3392
Fax. 508-457-2134
http://www.whoi.edu/WHOI/SciTechDir/timothy_m_shank.html

February 21 – 22, 2003

author: **Steven M. Thompson**

Florida State University
School of Computational Science & Information Technology
Tallahassee, Florida 32306-4120
telephone: 850-644-1010
fax: 850-644-0098

corresponding address:

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

¥GCG is the Genetics Computer Group, part of Accelrys Inc., a subsidiary of Pharmacopeia Inc., producer of the Wisconsin Package for sequence analysis.
 2003 BioInfo 4U

1. Introduction.

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 sequences by searching for particular patterns that reflect some function of the molecule. These can be catalogued motifs and domains, secondary structure predictions, physical attributes such as hydrophobicity, or even the content of 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, naturally we can; inference through homology is fundamental to all the biological sciences. We can learn a tremendous amount by comparing our sequence against others. And . . .

The power and sensitivity of sequence based computational methods dramatically increases with the addition of more data. More data yields stronger analyses — if done carefully! Otherwise, it can confound the issue. The patterns of conservation become clearer by comparing the conserved portions of sequences amongst a larger and larger dataset. Those areas most resistant to change are functionally the most important to the molecule. 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 do not occur 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 being less subject to evolutionary pressure. Therefore, sequences end 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 allowing comparison, 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 the dynamic programming algorithm.

1.1. Dynamic Programming.

Let's begin with a review of pairwise dynamic programming. 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. The first begins very much like the dot matrix methods described further below; the second is totally different. I will use a simplified example here. Instead of calculating the 'score matrix' on the fly, as is often taught 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 where the only allowable trace-back is diagonally behind and above. The illustration follows below:

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.

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. Matching symbols will still be worth one point, non-matching symbols will still be worth zero points, but we will penalize the scoring scheme by subtracting one 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, i.e. 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 described here and 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; the standard y = mx + b equation:

> total penalty = gap opening penalty + ([length of gap] * [gap extension penalty]).
>
> (To run most alignment programs with the type of simple DNA gap penalty used here, you would have to designate a gap 'creation' or 'opening' penalty of zero and a gap 'extension' penalty of whatever counts in that particular program as an identical base match for DNA sequences.)

As we will see, the oversimplified gap function used in this example does have a rather strange effect.

This example uses two randomly generated sequences that happen to fit the tata consensus regions of eukaryotes and prokaryotes. The most conserved bases within the consensus are capitalized. The sample eukaryote promoter sequence is along the X-axis, the prokaryote 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 sums and differences 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) Clean up the score matrix next. I'll only show the totals in each cell here:

	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 trace-back path graph by picking the bottom-most, furthest right and highest scoring coordinates. Then choose the highest scoring trace-back route, to connect them all the way back to the beginning using the same 'over-your-left-shoulder' rule:

	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 will probably 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 two optimum alignments:

```
cTATAtAagg          cTATAtAagg
|  |||||           |   ||||
cg.TAtAaT.          cgT.AtAaT.
```

Each of these solutions yields a trace-back total score of 22. This is the number optimized by the algorithm, not any type of a similarity or identity score! Even though one of these alignments has six exact matches and the other has five, they are both optimal according to the rather strange gap penalty criteria in which we solved the algorithm.

Some programs offer the highroad/lowroad option mentioned above to help explore this solution space. One of the above solutions is the GCG HighRoad solution found when running the program Gap with the above example's parameter settings:

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

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

Prok_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 could be discovered? Answer: Often if you reverse the solution of the entire dynamic programming process, other solutions are found! In other words, reverse the sequences in software programs to see alternative alignments.

To recap, and for those people that like equations, an optimal pairwise 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 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:

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

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.

However, just because dynamic programming guarantees 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. Later refinements (Smith and Waterman, 1981) demonstrated how dynamic programming could also be used to find optimal local alignments. To solve dynamic programming using local alignment (without going into all the gory details) programs use the following two 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 trace-back path matrix with many alternative paths, most of which do not extend the full length of the graph.
2) The best trace-back 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!

1.2. Scoring Matrices.

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 protein 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 mentioned above, 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 fancy matrix and logarithmic mathematics that smooth out the statistics of the system, she was able to empirically specify the relative probabilities at which different residues mutated 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) log odds matrix. Since Dayhoff's time other biomathematicians (esp. see Henikoff and Henikoff's [1992] BLOSUM series of tables, and for a quite controversial matrix see Gonnet et al. [1992]) have created newer matrices with more or less success than Dayhoff's original but the concept remains the same and Dayhoff's original PAM 250 table remains a classic as historically the most widely used one. Collectively these types of tables are known as symbol comparison tables, log odds matrices, or scoring matrices, and they are fundamental to all sequence comparison techniques. Dayhoff's PAM250 table follows below; the main identity diagonal is highlighted with outline characters:

The Wisconsin Package's version of the Dayhoff PAM250 matrix (Dayhoff, et al., 1979).

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

The standard default scoring matrix for many protein similarity comparison programs is now the BLOSUM62 table. It follows below; values whose magnitude is ≥ ±4 are drawn in outline characters to make them easier to recognize.

BLOSUM62 amino acid substitution matrix (Henikoff and Henikoff, 1992).

	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

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 an identity score of 11; cysteine is next with a score of 9; histidine gets 8; both proline and tyrosine get scores of 7. Also check out the hydrophobic substitution triumvirate — isoleucine, leucine, valine, and to a lesser extent methionine — all easily swap places. So rather than using the one/zero match function that we used in the simple tata dynamic programming example above, protein sequence alignments use

the match function provided by a scoring matrix such as this. The concept of similarity becomes very important with some amino acids being way 'more similar' than others!

1.3. Database Searching.

After all of these concepts are considered we can screen databases to look for sequences to compare ours to. 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.

1.3.1. Significance.

A big question and a particularly common misnomer made in this area 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. It can be normalized to ascertain statistical significance as seen in the database searching methods described below, but it's still just a number. Homology, in contrast and by definition, implies an evolutionary relationship — more than just the fact that we have all evolved from the same old 'hot spring/pond scum.' 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, structural, morphological, genetic, or fossil, that corroborates your assertion. There really is no such thing as percent homology; something is either homologous or it is not. The famous molecular evolutionist 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 all too commonly made mistake of calling any old sequence similarity homology. Highly significant similarity can argue for homology, but never the other way around.

So, how do you tell if a similarity, in other words, 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 of the 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 by the programs. The 'quality' scores mean a lot more but are difficult to interpret. At least they take 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 the algorithm section of the GCG Program Manual for the various methods — statistics can be confusing but the descriptions are written well and they do 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 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 line 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 nine, 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 in spite of 75% identity! Composition can make a huge difference!

The FastA (Pearson and Lipman, 1988; and Pearson, 1998), BLAST (Altschul, et al., 1990), and ProfileSearch (Gribskov, et al., 1987) algorithms, described below, all use a similar approach but base their statistics on the distance of the query matches from the actual, or a simulated, extreme value distribution from the rest of the, 'insignificantly similar,' members of the database being searched. BLAST and FastA generate Expectation values 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 and very conservative guidelines to Z scores and Expectation values from a typical search:

~Z score	~E value	Inference
≤3	≥0.05	little, if any, evidence for homology, but impossible to disprove
≈5	≈10^{-5}	probably homologous, but may be due to convergent evolution
≥9	≤10^{-15}	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.

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. 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.

1.3.2. The Searching Programs.

Database searching programs use elements of all the concepts discussed above; 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, that of hashing and that of approximation. Hashing is the process of breaking your sequence into small 'words' or 'k-tuples' 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 techniques 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 BLAST and Fast 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. These include: a DNA sequence against a DNA database (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, a translated ('on-the-fly') version of a DNA sequence against a protein database, a protein sequence against a translated ('on-the-fly') version of the DNA database, or a protein sequence against a protein database. Many implementations allow the recognition of frame shifts in translated comparisons.

In more detail:

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

1) Normally not a good idea to use for DNA against DNA searches (not optimized);
2) Pre-filters 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 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 un-gapped extension in both directions, such that each diagonal is extended as far as it can, until the running score starts to drop below a pre-defined 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 pre-computed 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 can not 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 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 how strong an alignment can be expected from chance alone.

1.3.2.2 FastA — and Family, Developed at the 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 a 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.

The algorithm:

FastA also 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, un-gapped, diagonal by adding all of the exact match BLOSUM values. The ten highest scoring diagonals for each query-database pair are then re-scored 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. If the user requests pair-wise alignments in the output, then the program uses full Smith-Waterman local dynamic programming, not 'restricted to a band,' to produce its final alignments.

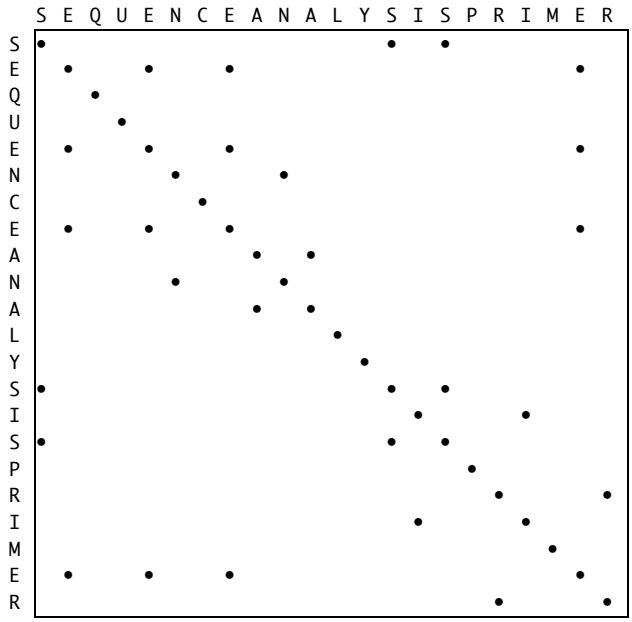
In review, both the BLAST and FastA family of programs base their Expectation "*E*" values on a more realistic 'extreme value distribution,' based on either real or simulated 'not significantly similar' database alignments, than Monte Carlo style Z scores do. Regardless, they follow Monte Carlo style 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 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 have more data. This drawback 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.' However, sometimes a preliminary untranslated search of the DNA databases with a DNA query will yield valuable information such as the important questions: "Am I working on something that's already been done; is it worth continuing?" or, "Am I doing something totally unique — or maybe, just 'junk' DNA?" An early nucleotide search may point out the necessity of continued research or the contemplation of abandonment, before any translational analysis has been completed. This could prove to be a tremendous time saver, so it may be worthwhile. Therefore, even though there are advantages and disadvantages to both types of searching, the general rule is to query with a peptide sequence, if at all possible, and screen whichever databases you choose.

1.4. Dot Matrix Procedures.

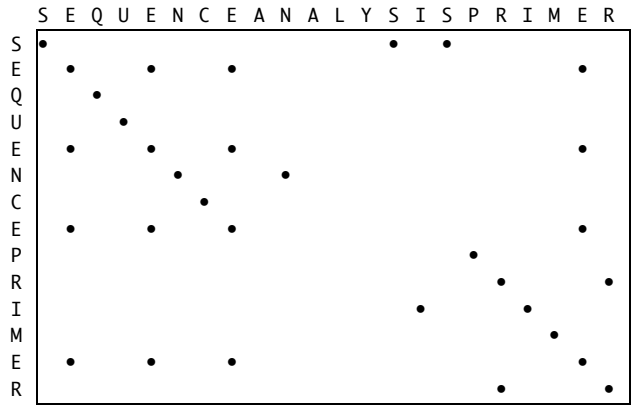
Another powerful method that should always be considered in similarity analysis is the dot matrix procedure. In dot matrix analysis one sequence is plotted on the vertical axis against another on the horizontal axis using a very simple approach; wherever they match according to some scoring criteria that you specify, a dot is generated. Why use dot matrix analysis? 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 optimal alignments. 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 that happen to have significant matches that no other method would ever notice. What approaches are used?

a) To illustrate, I will use a very simple 0, 1 (match, no-match) identity scoring function. More complex scoring functions such as the BLOSUM62 matrix are always used with real amino acid sequences. This example is based on an illustration in a dated but very addressable text, *Sequence Analysis Primer* (Gribskov and Devereux, 1992). The sequences compared are written out along the x and y axes of a matrix and a dot is placed wherever the two squences' symbols match:



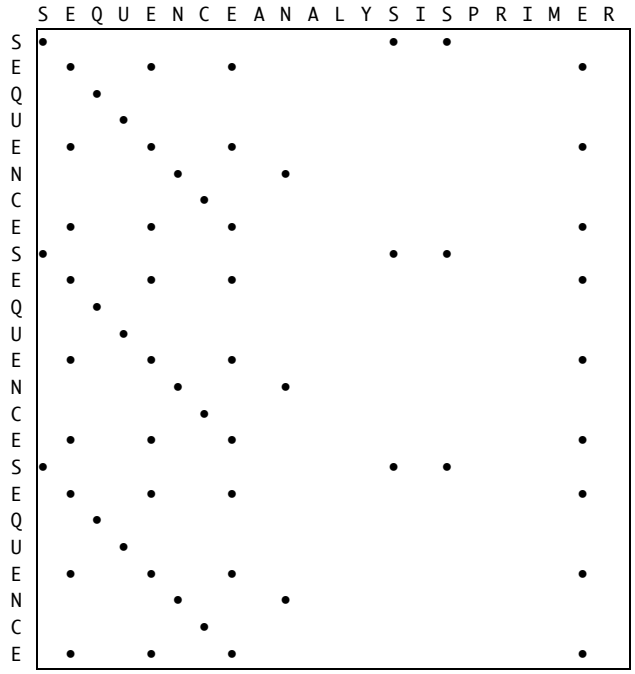
Since this is a comparison between two of the same sequences, an *intra*sequence comparison, the most obvious feature is the main identity diagonal. Two short perfect palindromes can be seen as crosses directly off the main diagonal; they are "ANA" and "SIS." If this were a double-stranded DNA or RNA sequence self comparison, these inverted repeat regions would be indicative of potential cruciform pseudoknots at that point. Direct internal repeats will show up as parallel diagonals off of the main diagonal. The biggest asset of dot matrix analysis is it allows you to visualize the entire comparison at once, not concentrating on any one 'optimal' region, but rather giving you the 'Gestalt' of the whole thing. You can see the 'less than best' comparisons as well as the main one and then 'zoom-in' on those regions of interest using more detailed procedures.

b) Check out the 'mutated' *inter*sequence comparison:



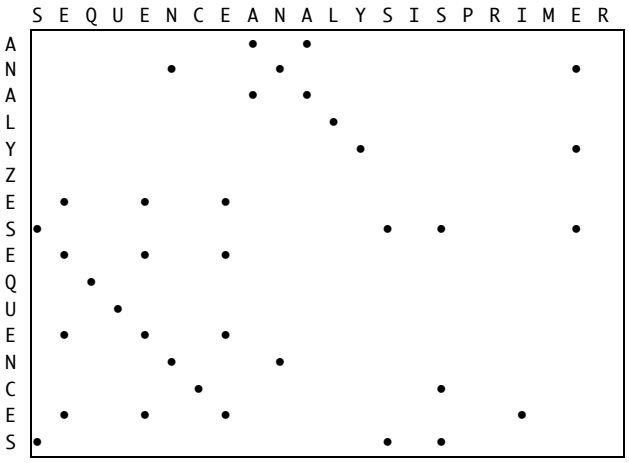
Here you can easily see the effect of a sequence 'insertion' or 'deletion.' It is impossible to tell whether the evolutionary event that caused the discrepancy between the two sequences was an insertion or a deletion and hence this phenomena is called an 'indel.' A jump or shift in the register of the main diagonal on a dotplot clearly points out the existence of an indel.

c) Other phenomena that are easy to visualize with dot matrix analysis are duplications and direct repeats. These are shown in the following example, still using the 0, 1 match function:



The 'duplication' here is seen as a distinct column of diagonals. Whenever you see either a row or column of diagonals in a dotplot, you are looking at direct repeats.

d) Now consider the more complicated mutation in the following comparison:

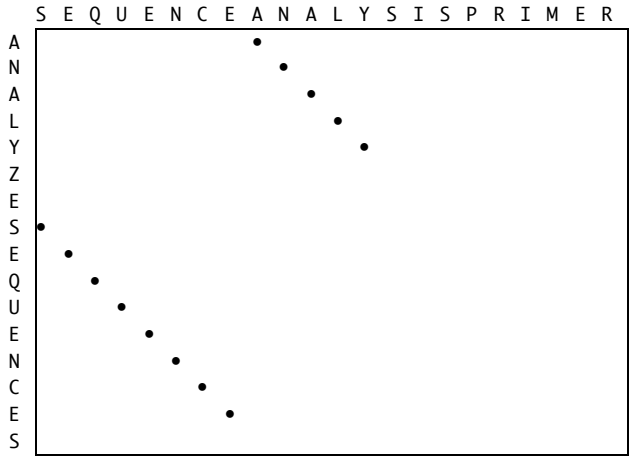


Again, notice the diagonals. However, they have now been displaced off the center diagonal of the plot, and, in fact, in this example, show the occurrence of a 'transposition.' Dot matrix analysis is the only sensible way to locate such transpositions in sequences. Inverted repeats still show up as perpendicular lines to the diagonals, they are just now not on the center of the plot. The 'deletion' of 'PRIMER' is shown by the lack of a corresponding diagonal.

Reconsider the same plot. Notice the extraneous dots that neither indicate runs of identity between the two sequences nor inverted repeats. These merely contribute 'noise' to the plot and are due to the 'random' occurrence of the letters in the sequences, the composition of the sequences themselves. How can we 'clean up' the plots so that this noise does not detract from our interpretations? Sequence analysis is all about balancing signal to noise.

Consider the implementation of a filtered windowing approach; a dot will only be placed if some 'stringency' is met. What is meant by this is that if within some defined window size, and when some defined criteria is met, then and only then, will a dot be placed at the middle of that window. The window is then shifted over one position and the entire process is repeated. This very successfully rids the plot of unwanted noise and is the default dotplot mechanism for most available programs. The previous dotplots are actually a special case of a filtered windowing approach, that of using a window and stringency filter both equal to one. The next case will show how using a stringency of two within a window of size three makes a huge difference.

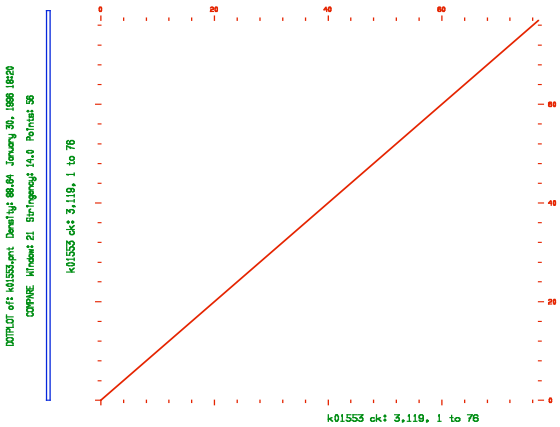
e) In the plot below a window of three with a stringency of two was used to considerably improve the signal to noise ratio:



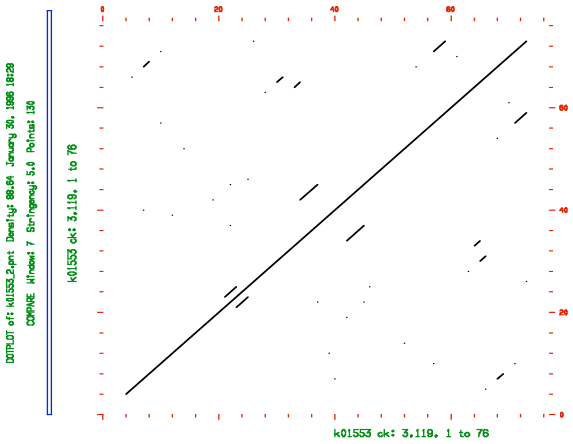
The only remaining dots indicate the two runs of identity between the two sequences, however, any indication of the palindrome, "ANA" has been lost. This is because our filtering approach was too stringent to catch such a short element. In general, you need to make your window about the same size as the element you are attempting to locate. In the case of our palindrome, "AN" and "NA"' are the inverted repeat sequences and since our window was set to three, we will not be able to see an element only two letters long. Had we set our filter to one out of two, then these would still be visible.

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). This example follows next:

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



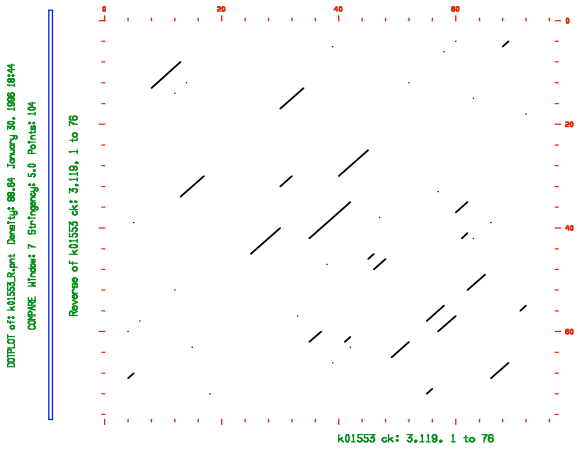
b) However, if you adjust the window size down to find finer features, some elements of symmetry become apparent. Here I have changed the window size to 7 and the stringency value to 5:



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

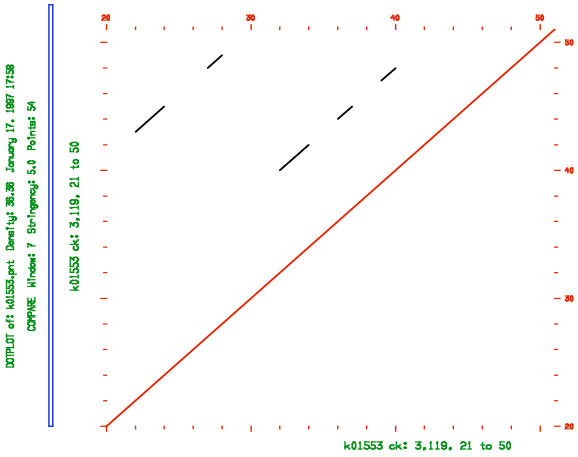
(As a general guide to stringency levels, pick whatever window size is most appropriate for the analysis at hand, e.g. about the size of the feature that you are trying to recognize, and then choose a stringency that produces a point file with the number of points found to be of a similar order of magnitude as that of the length of your longest sequence.)

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 to do in the GCG command line programs by specifying that you want the program to reverse the specified sequence. (In this context, the GCG option -Reverse is the reverse, complement of a sequence, not just the reverse since just the reverse is not biologically relevant.) (In SeqLab use the Edit-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:

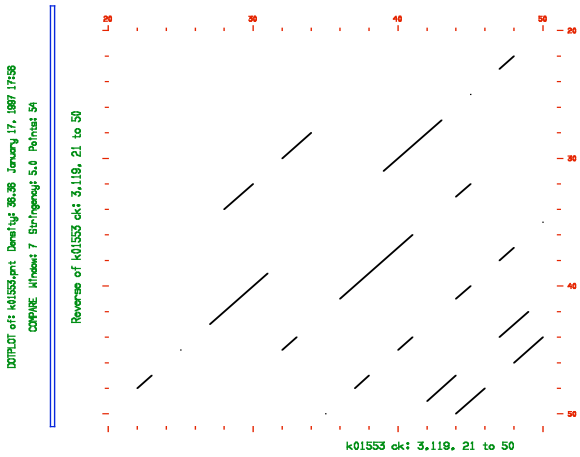


Now the potential for inverted repeats becomes obvious; these are the well characterized stem-loop structures of the tRNA cloverleaf molecular shape. They appear as clearly delineated diagonals. These diagonals are now perpendicular to an imaginary main diagonal running opposite to the previous case, 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 approximately 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:



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



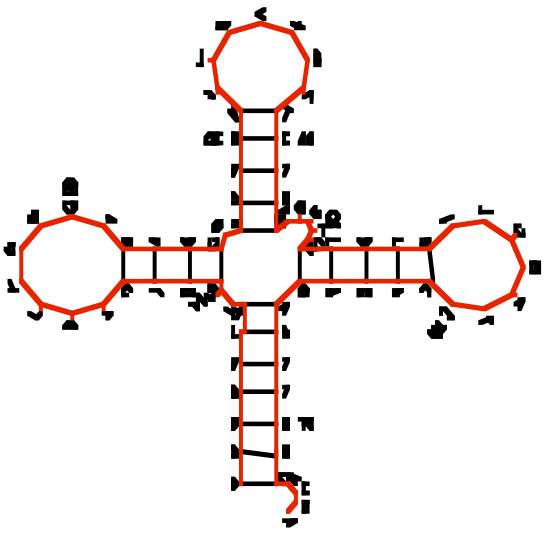
f) The GCG program StemLoop can show some of these match-ups base by base. Depending on what parameters you use, this is one of them:

```
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) This stem probably corresponds to the bottom-most stem representation in the standard model of tRNA. Here is a view in an orientation that allows you to visualize most of the stem structure of the yeast phenylalanine tRNA model as solved by Sundaralingam et al. (1976) deposited in the 3D PDB under access code 1TRA:

h) In a GCG 'Squiggle' plot of this same molecule from the output of MFold, Zuker's (1989) RNA folding algorithm which uses base pairing energy minimization to find the family of most optimal and suboptimal structures, the most stable structure found is shown to possess a stem at positions 27 to 31 with 39 to 43. However the region around position 38 is represented as a loop. Note that the molecule is upside down here as compared to the previous model. Reality, as modeled above, is seen to lie somewhere in between these two interpretations, but the simple dotplot analysis did quickly provide some valuable insights.



1.5. Multiple Sequence Dynamic Programming.

As seen in pairwise dynamic programming, looking at every possible position by sliding one sequence along every other sequence, just will not work for alignment. Therefore, dynamic programming reduces the problem back down to N^2. But how do you work with more than just two sequences at a time? It becomes a much harder problem. You could painstakingly manually align all your sequences using some type of editor, and many people do just that, but some type of an automated solution is desirable, at least as a starting point to manual alignment. However, solving the dynamic programming algorithm for more than just two sequences rapidly becomes intractable. Dynamic programming's complexity, and hence its computational requirements, increases exponentially with the number of sequences in the dataset being compared (complexity=[sequence length]$^{\text{number of sequences}}$). Mathematically this is an N-dimensional matrix, quite complex indeed. As we have seen, pairwise dynamic programming solves a two-dimensional matrix, and the complexity of the solution is equal to the length of the longest sequence squared. Well, a three member standard dynamic programming sequence comparison would be a matrix with three axes, the length of the longest sequence cubed, and so forth. You can at least draw a three-dimensional matrix, but more than that becomes impossible to even visualize. It quickly boggles the mind!

Several different heuristics have been employed over the years to simplify the complexity of the problem. One program, MSA (Gupta et al. [version 2.0, 1995] and version 2.1), does attempt to globally solve the N-dimensional matrix equation using a bounding box trick. However, the algorithm's complexity precludes its use in most situations, except with very small datasets. One way to still globally solve the algorithm and yet reduce its complexity is to restrict the search space to only the most conserved 'local' portions of all the sequences involved. This approach is used by the program PIMA (Smith and Smith, version 1.4, 1995). MSA and PIMA are both available through the Internet at several bioinformatics servers (in particular the Baylor College of Medicine's Search Launcher at http://searchlauncher.bcm.tmc.edu/) and you can investigate these resources on your own time.

1.5.1. How the Algorithm Works.

The most common implementations of automated multiple alignment modify dynamic programming by establishing a pairwise order in which to build the alignment. This modification is known as pairwise, progressive dynamic programming. Originally attributed to Feng and Doolittle (1987), this variation of the dynamic programming algorithm generates a global alignment, but restricts its search space at any one time to a local neighborhood of the full length of only two sequences. Consider a group of sequences. First all are compared to each other, pairwise, using normal

dynamic programming. This establishes an order for the set, most to least similar. Subgroups are clustered together similarly. Then take the top two most similar sequences and align them using normal dynamic programming. Now create a consensus of the two and align that consensus to the third sequence using standard dynamic programming. Now create a consensus of the first three sequences and align that to the forth most similar. This process continues until it has worked its way through all sequences and/or sets of clusters. The pairwise, progressive solution is implemented in several programs. Perhaps the most popular is Higgins' and Thompson's ClustalW (1994) and its multi-platform, graphical user interface ClustalX (Thompson, et al., 1997). These can be found at biocomputing sites around the globe and installed on your own machine or run through the World Wide Web (WWW). ClustalX has versions available for most windowing computing Operating Systems — most UNIX flavors, Microsoft Windows, and Macintosh. The ClustalX homesite guarantees the latest version: ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalX/. Complete documentation comes with the program and is accessed through a "Help" menu. The GCG program PileUp implements a very similar method.

As seen with pairwise alignments and sequence database similarity searching, all of this is much easier with protein sequences versus nucleotide sequences. Twenty symbols are just much easier to align then only four; the signal to noise ratio is again so much better. And, as in database searching, the concept of similarity applies to amino acids but generally not to nucleotides. Therefore, just like in database searching, multiple sequence alignment should always be done on a protein level if at all possible, unless the DNA sequences are so similar as to not cause any problem. Therefore, translate nucleotide sequences to their protein counterparts if you are dealing with coding sequences before performing multiple sequence alignment. The process is much more difficult if you are forced to align nucleotides because the region does not code for a protein. Automated methods may be able to help as a starting point, but they are certainly not guaranteed to come up with a biologically correct alignment. The resulting alignment will probably have to be extensively edited, if it works at all. Success will largely depend on the similarity of the nucleotide dataset.

One liability of global progressive, pairwise methods is they are entirely dependent on the order in which the sequences are aligned. Fortunately ordering them from most similar to least similar usually makes biological sense and works very well. However, the techniques are very sensitive to the substitution matrix and gap penalties specified. Programs such as ClustalW and PileUp that allow 'fine-tuning' areas of an alignment by re-alignment with different scoring matrices and/or gap penalties can be extremely helpful because of this. However, any automated multiple sequence alignment program should be thought of as only a tool to offer a starting alignment that can be improved upon, not the 'end-all-to-meet-all' solution, guaranteed to provide the 'one-true' answer.

1.5.2. Reliability?

To help assure the reliability of sequence alignments always use comparative approaches. A multiple sequence alignment is a hypothesis of evolutionary history. Insure that you have prepared a good one — be sure that it makes sense. Think about it — a sequence alignment is a statement of positional homology. It establishes the explicit homologous correspondence of each individual sequence position, each column in the alignment. Therefore, devote considerable time and energy toward developing the most satisfying multiple sequence alignment possible. Editing alignments is allowed and to be encouraged. Specialized sequence editing software such as GCG's SeqLab Editor help achieve this but any editor will do as long as the sequences end up properly formatted afterwards. After some automated solution has offered its best guess, go into the alignment and use your own brain to improve it. Use all available information and understanding to insure that all columns are truly homologous. Look for conserved functional sites to help guide your judgement. Assure that known enzymatic, regulatory, and structural elements all align, for the results of subsequent analyses are absolutely dependent upon the alignment.

Researchers have successfully used the conservation of co-varying sites in ribosomal and other structural RNA alignments to assist in alignment refinement. That is, as one base in a stem structure changes the corresponding Watson-

Crick paired base will change in a corresponding manner. This process has been used extensively by the Ribosomal Database Project at the Center for Microbial Ecology at Michigan State University to help guide the construction of their rRNA alignments and structures. The WWW Uniform Resource Locator (URL) is http://rdp.cme.msu.edu/html/.

Be sure an alignment makes biological sense — align things that make sense to align! Beware of comparing 'apples and oranges.' If creating alignments for phylogenetic inference, either make paralogous comparisons (i.e. evolution via gene duplication) to ascertain gene phylogenies within one organism, or orthologous (within one ancestral loci) comparisons to ascertain gene phylogenies between organisms which should imply organismal phylogenies. Try not to mix them up without complete data representation. Lots of confusion can arise, especially if you do not have all the data and/or if the nomenclature is contradictory; extremely misleading interpretations can result. Be wary of trying to align genomic sequences with cDNA when working with DNA; the introns will cause all sorts of headaches. Similarly, do not align mature and precursor proteins from the same organism and loci. It does not make evolutionary sense, as one is not evolved from the other, rather one is the other. These are all easy mistakes to make; try your best to avoid them.

Remember the old adage "garbage in — garbage out!" Some general guidelines to remember include the following:

- If the homology of a region is in doubt, then throw it out (or "mask" it, as will be shown in later using SeqLab).
- Avoid the most diverged parts of molecules; they are the greatest source of systematic error.
- Do not include sequences that are more diverged than necessary for the analysis at hand.

1.5.3. Applicability?

So what's so great about multiple sequence alignments; why would anyone want to bother? They are:

- very useful in the development of PCR primers and hybridization probes;
- great for producing annotated, publication quality, graphics and illustrations;
- invaluable in structure/function studies through homology inference;
- essential for building "Profiles" for remote homology similarity searching; and
- required for molecular evolutionary phylogenetic inference programs such as those from PAUP* (Phylogenetic Analysis Using Parsimony [and other methods]) and PHYLIP (PHYLogeny Inference Package).

A multiple sequence alignment is useful for probe and primer design by allowing you to visualize the most conserved regions of an alignment. This technique is invaluable for designing phylogenetic specific probes as it clearly localizes areas of high conservation and high variability in an alignment. Depending on the dataset that you analyze, any level of phylogenetic specificity can be achieved. Pick areas of high variability in the overall dataset that correspond to areas of high conversation in phylogenetic category subset datasets to differentiate between universal and specific potential probe sequences. After localizing general target areas on the sequence, you can then use any of a number of primer discovery programs to find the best primers within those regions and to test those potential probes for common PCR conditions and problems.

Graphics prepared from multiple sequence alignments can dramatically illustrate functional and structural conservation. These can take many forms of all or portions of an alignment — shaded or colored boxes or letters for each residue, cartoon representations of features, running line graphs of overall similarity, overlays of attributes, various consensus representations, etc. — all can be printed with high-resolution equipment, usually in color or gray tones. These can make a big difference in a poster or manuscript presentation.

Conserved regions of an alignment are functionally important. In addition to the conservation of primary sequence and function, structure is also conserved in these crucial regions. In fact, recognizable structural conservation between true

homologues extends way beyond statistically significant sequence similarity. An oft-cited example is in the serine protease superfamily. *S. griseus* protease A demonstrates remarkably little similarity when compared to the rest of the superfamily (Expectation values E()≥$10^{1.8}$ in a typical search) yet its three-dimensional structure clearly shows its allegiance to the serine proteases (Pearson, W.R., personal communication). These principles are the premise of 'homology modeling' and it works remarkably well.

Profiles are a position specific weight matrix description of an alignment or a portion of an alignment. Gap insertion is penalized more heavily in conserved areas of the alignment than it is in variable regions, and the more highly conserved a residue is, the more important it becomes. Originally described by Gribskov (1987), later refinements have added more statistical rigor (see e.g. Eddy's Hidden Markov Model Profiles [1996 and 1998]). Several profile style programs will be described in detail later in the tutorial. Generally, a profile is created from an alignment of related sequences and then used to search databases for remote sequence similarities. Profile searching is tremendously powerful and can provide the most sensitive, albeit extremely computationally intensive, database similarity searches possible.

Finally, we can use multiple sequence alignments to infer phylogeny. Based on the assertion of homologous positions in an alignment, several algorithms can estimate the most reasonable evolutionary tree for that alignment. This is a huge, complicated, and highly contentious field, hopefully to be delved into later in your lifelong learning experience. (See the Woods Hole Marine Biological Laboratory's excellent summer course, the Workshop on Molecular Evolution, at http://newfish.mbl.edu/Course/.) However, always remember that regardless of algorithm used, parsimony, any distance method, maximum likelihood, or even Bayesian Inference, all molecular sequence phylogenetic inference programs make the absolute validity of your input alignment their first and most critical assumption.

I reiterate, the most important factor in inferring reliable phylogenies is the accuracy of the multiple sequence alignment. The interpretation of your results is utterly dependent on the quality of your input. In fact, many experts advice against using any parts of the sequence data that are at all questionable. Only analyze those portions that assuredly do align. If any portions of the alignment are in doubt, throw them out. This usually means trimming down or masking out the alignment's terminal ends and may require internal trimming or masking as well. Biocomputing is always a delicate balance — signal against noise — and sometimes it can be quite the balancing act!

1.5.4. Complications.

One of the biggest problems in computational biology is that of molecular sequence data format. Each suite of programs to come along seems to require its own different sequence format. The major databases all have their own; Clustal has its own; even the database similarity searching program FastA has a sequence format associated with it. GCG Wisconsin Package sequence format exists both as single and Multiple Sequence Format (MSF) and GCG's SeqLab has its own format called Rich Sequence Format (RSF) that contains both sequence data and reference and feature annotation. PAUP* has a required format called the NEXUS file and PHYLIP has its own unique input data format requirements. The PAUP* interfaces in the GCG Wisconsin Package, PAUPSearch and PAUPDisplay, automatically generate their required NEXUS format directly from the GCG formatted files. Most systems are not nearly so helpful. Several different programs are available to convert formats back and forth between the required standards, but it all can get quite confusing. One program available, ReadSeq by Don Gilbert at Indiana University (1990), allows for the back and forth conversion between several different formats. I would heartily recommend installing it on all of your computers. It comes as an old 'tried-and-trued' C version or a new JAVA version with a graphical interface. I don't have much experience with the JAVA version but have relied on the C version for many years. Alignment gaps are another problem. Different program suites may use different symbols to represent them. Most programs use hyphens, "-", the Wisconsin Package uses periods, ".". Furthermore, not all gaps in sequences should be interpreted as deletions. Interior gaps are probably okay to represent this way, as regardless of whether a deletion, insertion or a duplication event created the gap, logically they

will be treated the same by the algorithms. These are indels. However, end gaps should not be represented as indels because a lack of information beyond the length of a given sequence may not be due to a deletion or insertion event. It may have nothing to do with the particular stretch being analyzed at all. It may just not have been sequenced! These gaps are just place holders for the sequence. Therefore, it is safest to manually edit an alignment to change leading and trailing gap symbols to "x"'s which mean "unknown amino acid," or "n"'s which mean "unknown base," or "?"'s which is supported by many programs, but not all, and means "unknown residue or indel." This will assure that the programs do not make incorrect assumptions about your sequences.

1.6. The Protein System.

I use members of the same dataset throughout this tutorial to make it more interesting and to provide a common focused objective. It should be somewhat analogous to what you would have to do in an actual laboratory setting and should provide a framework on which you can build. In fact, if you care, switch my example to your 'pet' molecule throughout.

The Elongation Factors are a vital protein family crucial to protein biosynthesis. They are ubiquitous to all of cellular life and, in concert with the ribosome, they must have been one of the very earliest enzymatic factories to evolve. I will use the Elongation Factor subunit known as 1-Alpha (EF-1α) in Eukaryota and Archaea and called Elongation Factor Tu in [Eu]Bacteria (and Euk' and Arch' plastids) as an example in this tutorial. It is essential in the universal process of protein biosynthesis and promotes the GTP-dependent binding of aminoacyl-tRNA to the A-site of the intact ribosome. GTP is hydrolyzed to GDP in the process. Because of strong evolutionary pressure resulting in very slow divergence and because of its ubiquity, it is an appropriate gene on which to estimate early life questions. In fact, a series of papers in the early-90's, notably those by Iwabe, et al. (1989), Rivera and Lake (1992), and Hasegawa, et al. (1993) all base 'universal' trees of life on this gene. Iwabe, et al. used the trick of aligning the α gene paralogue EF-1β to their α dataset to root the tree. Elongation Factor 1α/Tu has guanine nucleotide, ribosome, and aminoacyl-tRNA binding sites. There are three distinct types of elongation factors that all work together to help perform the vital function of protein biosynthesis. In [Eu]Bacteria and Eukaryota nuclear genomes they have the following names (the nomenclature in Archaea has not been completely worked out and is often contradictory):

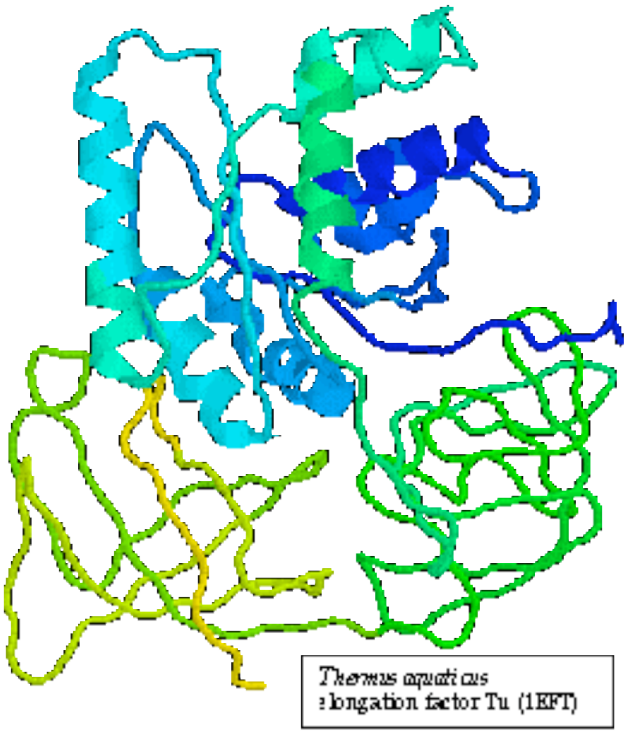
Eukaryota	[Eu]Bacteria	Function
EF-1α	EF-Tu	Binds GTP and an aminoacyl-tRNA; delivers the latter to the A site of ribosomes.
EF-1β	EF-Ts	Interacts with EF-1α/Tu to displace GDP and thus allows the regeneration of GTP-EF-1α/Tu
EF-2	EF-G	Binds GTP and peptidyl-tRNA and translocates the latter from the A site to the P site.

In EF-1α, a specific region is involved in a conformational change mediated by the hydrolysis of GTP to GDP. This region is conserved in both EF-1α/Tu and EF-2/G and seems to be typical of GTP-dependent proteins which bind non-initiator tRNAs to the ribosome.

In *E. coli* EF-Tu is encoded by a duplicated loci, *tuf*A and *tuf*B located about 15 minutes apart on the chromosome at positions 74.92 and 90.02 (ECDC). In humans at least twenty loci on seven different chromosomes demonstrate homology to the gene. However, only two of them are potentially active; the remainder appear to be retropseudogenes (Madsen, et al., 1990). It is encoded in both the nucleus and mitochondria and chloroplast genomes in eukaryotes and is a globular, cytoplasmic enzyme in all life forms.

The three-dimensional structure of Elongation Factor 1α/Tu has been solved in about 15 cases. Partial and complete *E. coli* structures have been resolved and deposited in the Protein Data Bank (1EFM, 1ETU, 1DG1, 1EFU, and 1EFC), the complete *Thermus aquaticus* and *thermophilus* structures have been determined (1TTT, 1EFT, and 1AIP), and even the cow EF-1β has been determined (1D2E). Most of the structures show the protein in complex with its nucleotide ligand, some

show the terniary complex. The *Thermus aquaticus* structure is shown below. Notice that half of the protein has well defined alpha helices and the rest is rather unordered coils. GTP fits right down in amongst all the helices in the pocket:



The *T. aquaticus* structure has six well-defined helices that occur from residue 24 through 38, 86 through 98, 114 through 126, 144 through 161, 175 through 184, and 194 through 207. There are also two short helices at residues 47 to 51 and 54 to 59. The guanine nucleotide binding site involves the following regions: residues 18 to 25, residues 81 to 85, and residues 136 to 139. Residue 8 is associated with aminoacyl-tRNA binding.

Sequence analysis of this dataset will explore these functional and structural regions as well as discover other interestingly conserved sites. For the tutorial here I will restrict my example to a subset of 'lower' eukaryotic EF-1 sequences. These will include many protists and algae but will exclude much of the "Crown" group, including all of the higher plants, true fungi, and metazoans. As such it may be an appropriate dataset with which to ask early branching order questions in deep eukaryotic evolution.

2. Database Searching and Multiple Sequence Alignment.

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

I will use **bold** type in this tutorial 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 Wisconsin Package. This comprehensive package of sequence analysis programs is used worldwide. The Wisconsin Package only runs on server computers running the UNIX operating system but it can be accessed from any networked terminal. It has arguably become the global 'industry-standard' in sequence analysis software. The Wisconsin Package provides a comprehensive toolkit of almost 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 powerful 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. SeqLab 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 and alignments can be manipulated. This workshop will show you how to use SeqLab to search for similar sequences, investigate pair-wise sequence similarity, and prepare and analyze multiple sequence alignments. 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. X server emulation software needs to be installed separately on personal style Microsoft Windows/Intel or Macintosh machines but genuine X-Windowing comes standard with most UNIX/Linux operating systems. 'Wintel' machines are often set up with either XWin32 or eXceed to provide this function; Pre OS X Macintoshes are often loaded with either MacX or eXodus software; OS X Macs can have true X windowing installed with the XDarwin package. The details of X and of connecting to your local GCG server will not be covered in this workshop. If you are unsure of these procedures ask for assistance in the computer laboratory. Your bio-computing support personnel are also available for individualized personal help in your own laboratories. I am also receptive to e-mail consultation, 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, and always close windows when you are through with them to conserve system memory. Furthermore, rather than holding mouse buttons down, to activate items, just click on them. Also buttons are turned on when they are pushed in and shaded. Finally, 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.

2.1. Log Onto Your GCG Account and Launch SeqLab.

Each participant in the session should use a different UNIX account. SeqLab behaves best when only one person uses it per UNIX GCG account. Either login with your existing account and password or use 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 assistance from your instructor or systems manager for this step if you are unsure of yourself. The details of X and of connecting to a GCG server are not covered here. There are just too many variations in method for them all to be described.

The Wisconsin Package usually initializes automatically as soon as your terminal window launches. If your site isn't configured this way, issue the command "**gcg**" (without the quotes) to initialize the software suite now. This initialization 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 command "**seqlab &**" (again without the quotes) in your terminal window 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. 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 any analyses, go to the "**Options**" menu and select "**Preferences . . .**." A few of the options should be checked there 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.

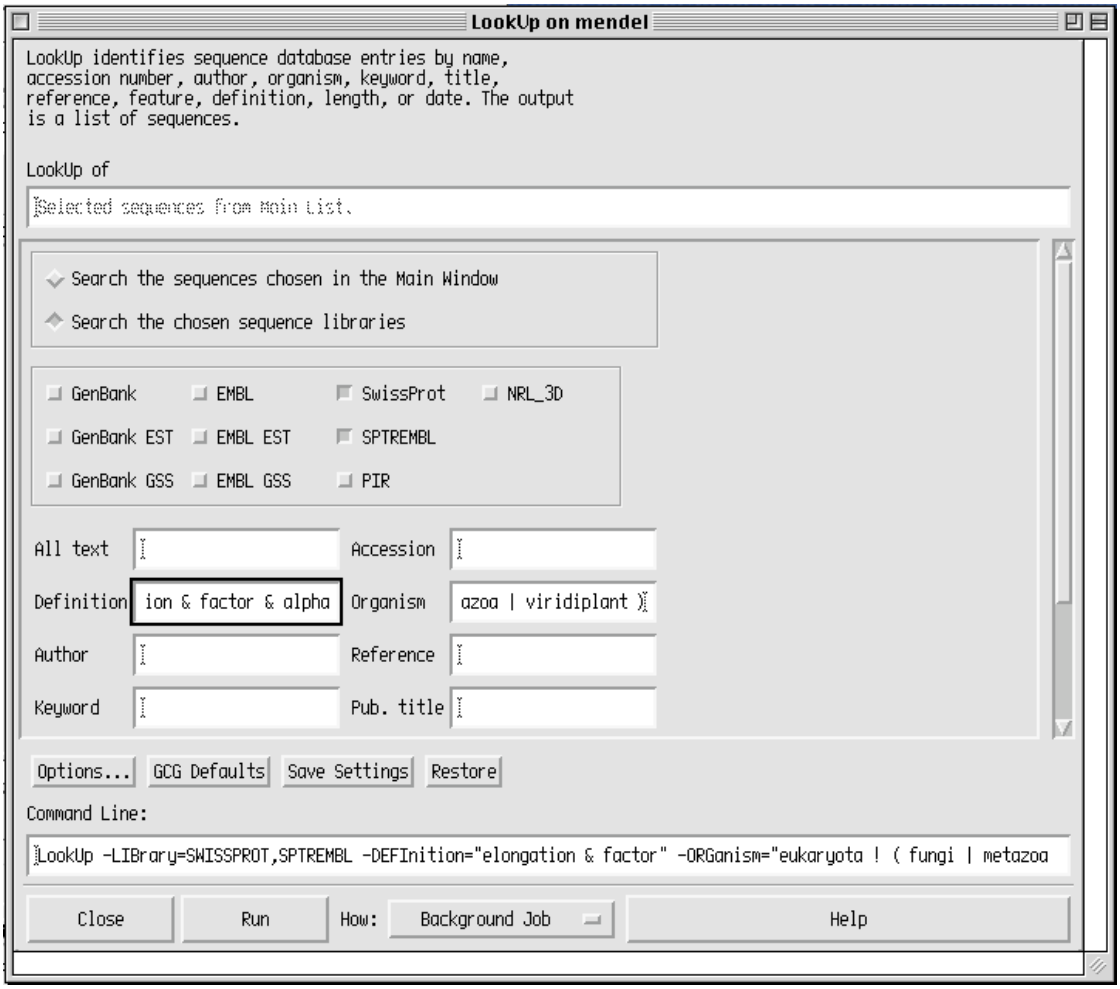
First notice that there are three different "**Preferences**" settings that can be changed: "**General**," "**Output**," and "**Fonts**;" start with "**General**." The "**Working Dir . . .**" setting will be the directory from which SeqLab was initially launched. This is where all SeqLab's working files 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. 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.

2.2. Find a Protein in the Database.

Given interest in a particular biological molecular sequence, you can use any of several available text string searching tools to find that entry's name in a sequence database. After an entry has been identified, a natural next step is to use a sequence similarity searching program such as FastA and/or BLAST to help prepare a list of sequences to be aligned. One of the more difficult aspects of multiple alignment analysis is knowing what sequences you should attempt it with. Any list from any program will need to be restricted to only those sequences that actually should be aligned. Make sure that the group of sequences that you align are in fact related, that they actually belong to the same gene family, that the alignment will be meaningful.

As described in the Introduction, the collection of sequences used throughout the tutorial contains representative EF-1 sequences from many 'lower' eukaryotes chosen based on taxonomy. The dataset was assembled using GCG's LookUp program, a Sequence Retrieval System (SRS) derivative (Etzold and Argos, 1993). But it could as well have been collected using Entrez at NCBI, either through the Web or installed as their client/server NetEntrez application, or SRS on the Web, available at all EMBL and many other biocomputing sites around the world (see e.g. http://srs.ebi.ac.uk/).

To find entries of interest in GCG sequence databases we need to know their proper database names or accession codes. Database text searching programs are often the easiest way to do this. Here I use GCG's LookUp program because it creates an output file that can be used as an input list file to other GCG programs. I use it here to find a representative set of elongation factor entries from the so-called 'primitive' eukaryotes. That is, those eukaryotes that exclude the Fungi, Metazoans, and true Plants. To start be sure that the "**Mode:**" "**Main List**" choice is selected in your main window and then launch "**LookUp**" through the "**Functions**" "**Database Reference Searching**" menu. In the new "LookUp" window be sure that "**Search the chosen sequence libraries**" is checked and then select "**SwissProt**" as well as "**SPTREMBL**" for the libraries to search. Under the main query section of the window, type the words and symbols "**elongation & factor & alpha**" following the category "**Definition**" and the words and symbols "**eukaryota ! (fungi | metazoa | viridiplantae)**" in the "**Organism**" category; next press the "**Run**" button. You need to use the Boolean operator symbols to connect the individual query strings 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 other single word supported level of taxonomy, e.g. "fungi." The Boolean operators supported by LookUp are the ampersand, "&", meaning "AND," the pipe symbol, "|", 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. This query should find most of the elongation factor alpha's from the 'lower' eukaryotes in the SwissProt and SPTREMBL databases and will provide a reasonable and interesting starting dataset for the tutorial. The "LookUp" window should look similar to the following:



The program will display the results of the search; scroll through the output and then "**Close**" the window. The beginning of the LookUp output file from the example follows below:

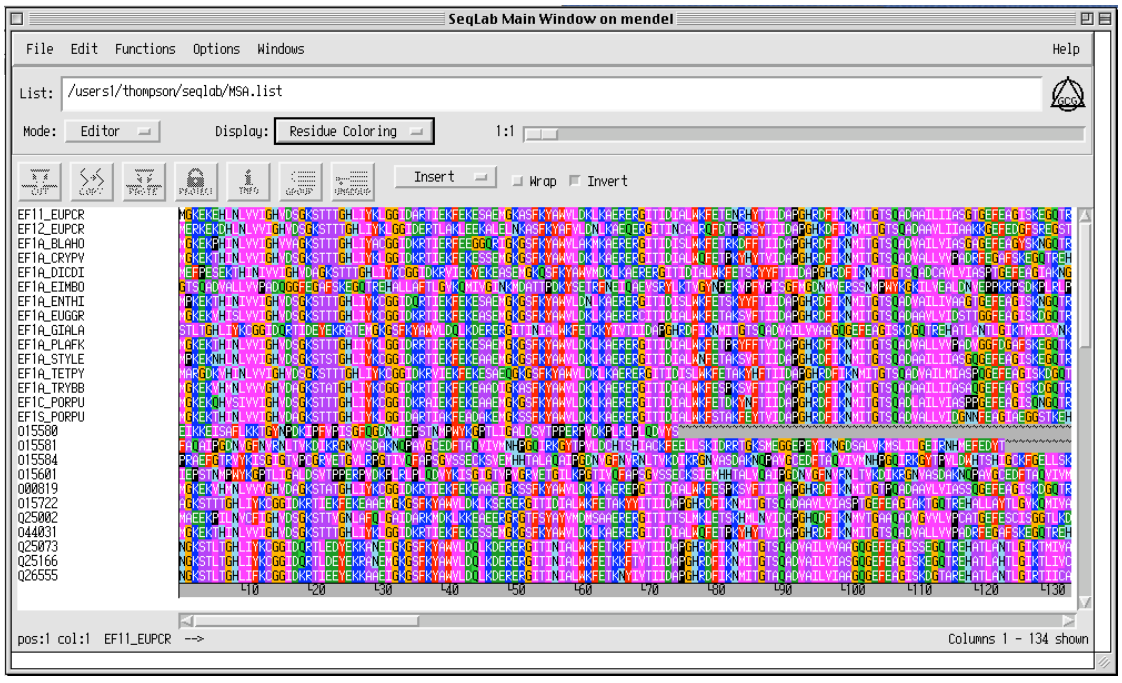
```
!!SEQUENCE_LIST 1.0
LOOKUP in: swissprot,sptrembl  of: "([SQ-DEF: elongation* & factor* & alpha*] &
[SQ-ORG: eukaryota* ! ( fungi* | metazoa* | viridiplant* )])"

 79 entries  May 10, 2001 16:08 ..

SWISSPROT:EF11_EUPCR ! ID: ef470001
! DE   ELONGATION FACTOR 1-ALPHA 1 (EF-1-ALPHA-1).
! GN   EFA1.
SWISSPROT:EF12_EUPCR ! ID: f8470001
! DE   ELONGATION FACTOR 1-ALPHA 2 (EF-1-ALPHA-2).
! GN   EFA2.
SWISSPROT:EF1A_BLAHO ! ID: 0d480001
! DE   ELONGATION FACTOR 1-ALPHA (EF-1-ALPHA).
SWISSPROT:EF1A_CRYPV ! ID: 14480001
! DE   ELONGATION FACTOR 1-ALPHA (EF-1-ALPHA).
SWISSPROT:EF1A_DICDI ! ID: 16480001
! DE   ELONGATION FACTOR 1-ALPHA (EF-1-ALPHA) (50 KDA ACTIN-BINDING PROTEIN)
! DE   (ABP-50).
! GN   EFAA.
SWISSPROT:EF1A_EIMBO ! ID: 17480001
! DE   ELONGATION FACTOR 1-ALPHA (EF-1-ALPHA) (FRAGMENT).
SWISSPROT:EF1A_ENTHI ! ID: 18480001
! DE   ELONGATION FACTOR 1-ALPHA (EF-1-ALPHA).
SWISSPROT:EF1A_EUGGR ! ID: 19480001
! DE   ELONGATION FACTOR 1-ALPHA (EF-1-ALPHA).
! GN   TEF.
SWISSPROT:EF1A_GIALA ! ID: 1a480001
! DE   ELONGATION FACTOR 1-ALPHA (EF-1-ALPHA) (14 NM FILAMENT-ASSOCIATED
```

```
! DE   PROTEIN) (FRAGMENT).
! GN   TEF1.
SWISSPROT:EF1A_PLAFK ! ID: 2a480001
! DE   ELONGATION FACTOR 1-ALPHA (EF-1-ALPHA).
! GN   MEF-1.
SWISSPROT:EF1A_STYLE ! ID: 35480001
! DE   ELONGATION FACTOR 1-ALPHA (EF-1-ALPHA).
! GN   EFAA.
SWISSPROT:EF1A_TETPY ! ID: 38480001
! DE   ELONGATION FACTOR 1-ALPHA (EF-1-ALPHA) (14 NM FILAMENT-ASSOCIATED
! DE   PROTEIN).
////////////////////////////////////////////////////////////////////////
```

Be careful that all of the proteins included in the output from any text searching program are appropriate. In this case the elongation factors found all look correct, but improper nomenclature and other database inconsistencies can always cause problems. If you find inappropriate proteins upon reading the output, you can either edit the output file to remove them, or "CUT" them from the SeqLab Editor display after loading the list. Another option, if you use an editor, is to comment out the undesired sequences by placing an exclamation point, "!," in front of the unwanted lines.

Select the LookUp output file in the "SeqLab Output Manager." This is a very important window and will contain all of the output from your current SeqLab session. Files may be displayed, printed, saved in other locations with other names, and deleted from this window. Press the "**Add to Main List**" button in the "**SeqLab Output Manager**" and "**Close**" the window afterwards. Go to the "**File**" menu next and press "**Save List**." Next, be sure that the LookUp output file is selected in the "**SeqLab Main Window**" and then switch "**Mode:**" to "**Editor**." This will load the file into the SeqLab Editor and allow us to perform further analyses on those entries.

Notice that all of the sequences now appear in the Editor window with the amino acid residues color-coded. The nine color groups are based on a UPGMA clustering of the BLOSUM62 amino acid scoring matrix, and approximate physical property categories for the different amino acids. Expand the window to an appropriate size by 'grabbing' the bottom-left corner of its 'frame' and 'pulling' it out as far as desired. Use the vertical scroll bar to see them all. Any portion of, or the entire alignment loaded, is now available for analysis by any of the GCG programs. The display should look similar to the following graphic after loading the dataset:



Another way to get sequences into SeqLab is to use the "Add sequences from" "Sequence Files. . ." choice under the "File" menu. Only GCG format compatible sequences or list files are accessible through this route. Use SeqLab's Editor "File" menu "Import" function to directly load GenBank format sequences or ABI binary trace files without the need to reformat. You can also directly load sequences from the online GCG databases with the "Databases. . ." choice under the "Add sequences" menu if you know their proper identifier name or accession code. The "Add Sequences" window's "Filter" box is very important! By default files are filtered such that only those that end with the extension ".seq" are displayed. This often won't do you any good as the sequences that you may want to add may have other extensions. Therefore, delete the ".seq" extension in the "Filter" box (including the period) if necessary, but be sure to leave the "*" wild card. Press the "Filter" button to display all of the files in your working directory. Select the file that you want from the "Files" box, and then check the "Add" and then "Close" buttons at the bottom of the window to put the desired file into your current list, if you're in List Mode, or directly into the Editor, if you're in "Editor Mode."

While you have sequences loaded in the Editor explore the interface for a bit. Each protein sequence is listed by its official SwissProt or SPTREMBL entry name (ID identifier). Use both scroll bars to move around within the sequences. The scroll bar at the bottom allows you to move through the sequences linearly; the one at the side allows you to scroll through all of your entries vertically. Quickly double click on various entries' names (or single click the "INFO" icon with the sequence entry name selected) to see the database reference documentation on them. (This is the same information that you can get with the GCG command "typedata -ref" at the command line.) "**Close**" the "**Sequence Information**" windows after reading them. You can also change the sequences' names and add any documentation that you want in this window. Change the "**Display:**" box from "**Residue Coloring**" to "**Feature Coloring**" and then "**Graphic Features**." Now the display shows a schematic of the feature information from each entry with colors based on the information from the database Feature Table for the entry. "Graphic Features" represents features using the same colors but in a 'cartoon' fashion. Quickly double-click on one of the various colored regions of the sequences (or use the "Features" choice under the "Windows" menu). This will produce a new window that describes the features located at the cursor. Select the feature to show more details 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.

Nearly all GCG programs are accessible through the "Functions" menu. Select various entry's names and then go to the "Functions" menu to perform different analyses on them. You can select sequences in their entirety by clicking on their names or you can select any position(s) within sequences by 'capturing' them with the mouse. You can select a range of sequence names by <shift><clicking> the top-most and bottom-most name desired, or <ctrl><click> sequence entry names to select noncontiguous entries. The "pos:" and "col:" indicators show you where the cursor is located on a 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.

It's probably a good idea to save the sequences in the display at this point and multiple times down the road as you work on a dataset. Do this occasionally the whole time you're in SeqLab just in case there's an interruption of service for any reason. Go to the "**File**" menu and choose "**Save As**." Accept the default "**.rsf**" extension but give it any file name and directory specification you choose. RSF (Rich Sequence Format) contains all the aligned sequence data as well as all the reference and feature annotation associated with each entry. It is "Richer" than most other multiple sequence formats and is SeqLab's default format.

2.3. Traditional Database Searching: FastA and Word Approaches.

Two different heuristic, hashing style, symbol matching algorithms have traditionally been utilized in database searching. These two algorithms (see the GCG Program Manual, take advantage of the Help buttons in SeqLab programs, or use the

genmanual command for details) are incorporated into GCG's FastA family (Pearson and Lipman, 1988 and Pearson, 1998) and WordSearch (Wilbur and Lipman, 1983) programs. Most of these programs 'eat' cpu; and work best when submitted as a batch or background process. This is because of the sizes of the databases involved. Version 131 of GenBank has over twenty-two billion bases, and GenBank doubles in size almost 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 that is really handy or you can run background searches while in SeqLab.

WordSearch is rarely used anymore since both FastA and BLAST style searches outperform this early algorithm, although, since the algorithms do differ, the output results will also differ. As in all computerized molecular biology analyses, the prudent may want to run as many strategies as practical and try to interpret the results in light of this. Here, however, due to time constraints, I will not be illustrating WordSearch today. If you are to run WordSearch sometime in the future, here's some relevant information on the program.

An advantage to WordSearch, like BLAST but unlike FastA, is multiple segments, i.e. more than one region of similarity between two sequences, can be found. Like all GCG database searching programs WordSearch accepts automatic batch submission with the -Batch option. It can also plot a histogram with the -Plot option. It will not cutoff at any particular score, rather you should specify a list size. A sometimes useful option is -Simplify. This simplifies the input sequences according to a scheme that generalizes amino acids into broad classes based on their chemical characteristics. Another helpful option available is -Mask. This option allows you to search with only certain characters of your sequence such as only the first and second codon positions in coding DNA. Unlike the FastA and BLAST programs, WordSearch does not automatically illustrate its alignments with sequence pairs. The separate program Segments does that. In order to visualize the alignments you can run Segments on a WordSearch output file. WordSearch has big problems with low-complexity, repeat types of sequence that BLAST filters out, therefore, prerunning Seq and Xnu on your sequences will avoid this potential pitfall. WordSearch has largely been superseded by FastA and BLAST approaches but does remain online as an alternative. As stated earlier, it is good to gather as much different data as you can. One situation where WordSearch does excel is in finding difficult DNA alignments such as aligning cDNA to genomic DNA. Peter Rice, formerly at EMBL, Heidelberg, and now at the Sanger Centre, U.K., wrote the following on the Info-GCG BIOSCI/bionet news group (http://net.bio.net/) concerning this aspect of WordSearch:

```
To: info-gcg@net.bio.net
From: rice@embl-heidelberg.de (Peter Rice)
Subject: Re: Way to gap around a 100 bp insertion?
Date: 11 Sep 93 13:57:19 GMT

In article <CD3st6.B28@watserv2.uwaterloo.ca>, kowalski@sciborg.uwaterloo.ca
(Paul Kowalski) writes:
> Is there some clever method to gap "around" a 100 bp insertion in genomic
> DNA? Lowering weights doesn't seem to help.

Sure. There is a program that does just that. Remember WORDSEARCH? The program
everyone used to use before FASTA . . . came along for GCG.

WORDSEARCH will find any decent sized exon (18 bases or so will get lost in
the noise of course). SEGMENTS is simply a BESTFIT run tied to each of
the hits. . . .

Use WORDSEARCH with a list size of 2 if you only have 2 exons. Use the
cDNA as the search sequence, and the genomic sequence as the "database".
I use a higher list size for safety, then edit out the lower scores from the
.WORD output file.

Use SEGMENTS to do the alignments.
```

```
WORDSEARCH is also wonderful for understanding fragments or contigs that
fail to overlap in Fragment Assembly.

I still wouldn't recommend WORDSEARCH for database searches, but for just a few
sequences (fragment assembly) or just one with several hits (as above) it
beats any other program.

You can lower the gap weights in SEGMENTS if you like. I often do for
Fragment Assembly matching, where you can expect a large number of 1-base
gaps (insertions or deletions) so you need a very low gap weight.
```

2.3.1 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 program is one of the more 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 databases and 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 (EST's [expressed sequence tags] and GSS's [genome survey sequences]) — be warned.

Select whichever EF-1 sequence entry name in the Editor that you wish to use for this section, I picked EF1A_GIALA, and then 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 EF-1 protein. A nice feature of the FastA family of database search programs is you can search any valid GCG sequence set specification. You are not restricted to specific prebuilt databases. The default database to search, "**Search Set. . .**" "**Using genembl:***" is all of GenBank and those sequences from EMBL that are not in GenBank, without the tags subdivisions of each, and it is often the one that you will want to use, but we're going to change it here so that we can get through the tutorial in a reasonable length of time. Therefore, push the "**Search Set. . .**" button, select "**genembl:***" in the "**Build TFastX's Search Set**" box that pops up, and then "**Remove from Search Set**." Next, press the "**Add Database Sequences**" button and then select "**Invertebrate**" from the "**SeqLab Database Browser**" window that popped up; press "**Add to Search Set**" and then "**Close**" the Database Browser and the Build Search Set windows. We will use the Invertebrate sequence specification as a convenient representation of GenEMBL that we can search relatively quickly. Also the Invertebrate specification will restrict our search to the subset of eukaryotes that we are primarily interested in although it will include metazoans that we are not interested in and exclude algae that we are. Regardless, we don't want to have to wait for all of GenEMBL to be searched, which could take an hour or so depending on the server's load at the time. (As mentioned above, the GenEMBL sequence specification does not include the "Tags" division, i.e. all EST's and GSS's; to search all nucleic acid databases including tags, use the sequence specification GenEMBLPlus [or the abbreviation GEP].) The other parameters in the main TFastX window are fine at their default settings, though you may want to decrease the cutoff Expectation value from its default of 10 to something more reasonable like 0.10 to reduce the output list size. 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 can be very helpful depending on your specific situation and should be explored in your own research. Restricting your search by the database sequence length or by date of their deposition in the database can be very handy. "Save and sort by optimized score" (the -OptAll option), in particular, is very useful and is now the program default. This causes the algorithm to sort its output based on a normalized derivative of the optimum score, the result of the program's dynamic programming 'in-a-band' pass, rather than the initn score, which is the longest combined

word score. "**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. Be sure not to submit the same job multiple times, and if you see that you have accidentally done so, you can use the "Job Manager" to "Stop" the given job. Go on with the rest of the tutorial now rather than waiting for the TFastX results at this point.

2.3.2. Contrast TFastX with Normal FastA — Protein Against Protein.

Now run the normal FastA program on the same EF-1 sequence searching either the "**PIR:***" or "**SwissProt:***" logical protein sequence specification. (As above, build a "**Search Set**" as desired.) Start and run the program just like above with TFastX only this time pick "**FastA. . .**" off of the "**Functions**" "**Database Sequence Searching**" menu. 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 in the workshop.

2.4. BLAST: Internet and Local Server Based Similarity Searching.

BLAST (Altschul, et al., 1990 and 1997) is a heuristic algorithm for searching sequence databases developed by the National Center for Biotechnology Information (NCBI), a division of the National Library of Medicine (NLM), at the National Institute of Health (NIH), the same people responsible for maintaining GenBank and for providing worldwide access to sequence analysis resources. The acronym stands for Basic Local Alignment Search Tool. The original BLAST algorithm only looked for un-gapped segments; however, the current version (Altschul, et al., 1997) adds a dynamic programming step to produce gapped alignments. As with the FastA family, BLAST ranks matches statistically and provides Expectation values for each to help evaluate significance. 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 of BLAST is it not only shows you the best alignment for each similar sequence found (as in the BestFit type alignments of FastA) but also shows the next best alignments for each up to a certain preset cutoff point. This combines some of 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 appropriate for comparing nucleotide sequences against the nucleotide database without going through translation steps 'on the fly.' In fact, in this situation, with its default parameters, it will only find closely similar DNA sequences, but will not be able to locate sequences that are only somewhat similar. Therefore, If you are dealing with a non-protein-coding, non-translated locus and are forced to compare a DNA query against a DNA database without translation, use FastA instead of BLAST; it is the far more appropriate tool.

The GCG implementation of NCBI's BLAST server, called NetBLAST, runs in a remote client-server mode such that NCBI's database and computers quickly perform the analysis, though you may have to wait for a few moments in a user waiting queue because the server tends to get quite busy. This program is an exception to the standard 'submit the search and wait' mode of most database searching programs. It 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 in Bethesda, MD. Typical searches run in just a few minutes, after you get through the waiting queue; however, realize that this algorithm, as with the local version of BLAST, is optimized for protein comparisons only. The BLAST server at NCBI can provide the most up to date database search available because NCBI updates GenBank and GenPept every night. Alternatively you can run GCG's local BLAST program if you have local BLAST databases assembled at your site, but it won't be as fast. For help in interpreting BLAST results refer to the GCG BLAST documentation, or NCBI's BLAST tutorial available on the Web, or their BLAST HELP file obtained by sending the single word "HELP" to BLAST@ncbi.nlm.nih.gov (leave

the subject line blank). An advantage to 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. Unlike other GCG programs, 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. For your information some features of NetBLAST's output format should be pointed out in case you ever need to deal with one of these files:

> 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 off of NetBlast's output from NCBI, or you will have to modify NCBI's format to make it comply with GCG standards. Since NCBI's computers don't know about GCG format requirements you won'to be able to use it as input to other GCG programs. To use it as input in other GCG programs you must manually edit the NetBLAST output list changing the database names to reflect the logicals that GCG understands. For example in a NetBLAST protein search output with the *Giardia* EF-1 protein query, you would need to change "sp|Q08046|EF1A_GIALA" to "SW:Q08046 EF1A_GIALA" (either insert a blank space or ! between the accession code and sequence name). The "gb" designations are translations from GenBank's CDS (CoDing Sequence) references. For instance, "gb|AAB81020.1| (U94406)" is a CDS translation from GenBank accession code U94406. This database, GenPept, is installed at most GCG sites. The GCG logicals "GP," "GenPep," and/or "GenPept" (case independent) usually point to the GenPept database, if your site maintains it. So to access gb|AAB40919.1 you would either have to translate the appropriate CDS region from GenBank:U82624 or you could directly specify the sequence from GenPept. Unfortunately, NetBLAST reports list "gb" numbers and GenBank accession codes, but GenPept is based on another designation, the "GI" number. Therefore, one needs to first look up the GenBank entry based on its accession code in order to find the corresponding CDS's GI number. This makes life a bit more complicated but is not that difficult to work around. You also must also edit the header portion of the file to separate it from the list portion with two consecutive periods (..).

To launch GCG's local BLAST program, be sure that your desired sequence entry name is still 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 program default on the main window is to "**Search a nucleotide database**" "**Search Set. . .**" "**Using local genembl**." 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, BLAST requires precompiled special databases and will not accept the general type of GCG sequence specification that the FastA programs will. Furthermore, searching any of the local BLAST nucleotide databases would take a while, so I suggest that we do not use TBLASTN at all and, instead, search one of the local BLAST protein databases using BLASTP. Therefore, change the selection to "**Search a protein database**" and be sure the "**Search Set. . .**" menu specifies "**Using local nrl**." This will search the local version of the NRL_3D database and will only take a moment, yet will still illustrate how BLAST works. Since NRL_3D contains all of the protein sequences from the Research Collaboratory for Structural Bioinformatics (RCSB) 3D structural database Protein Data Bank (PDB), http://www.rcsb.org/pdb/, this is also a quick way to gain insight into a sequence's potential structure, and may allow you to infer your protein's secondary structure based on an alignment with a known structure. As in the FastA programs, decreasing the Expectation cutoff value will decrease the output list size. 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 and 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= command line 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. You should get results almost instantly.

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

2.5. What Next? Comparisons, Interpretations, and Further Analyses.

I will show all the abridged database search output files next. Naturally, the topmost 'hits' will turn out to be EF-1 proteins; it's the ones below the expected hits that may prove interesting for this section of the workshop. Local BLASTP, FastA, and TFastX results follow below.

Especially pay attention to BLAST's E value scores in its output file. As explained in the Introduction, these are the likelihoods (expectations) that the observed matches could be due to chance; the smaller the E number, the more significant the match. They are much easier to interpret than the information bits score in the adjacent column. Here is the abridged BLASTP output from our search of NRL_3D:

```
!!SEQUENCE_LIST 1.0
BLASTP 2.1.2 [Nov-13-2000]

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= /users1/thompson/.seqlab-mendel/input_23.rsf{EF1A_GIALA}
         (396 letters)

Database: nrl
           23,291 sequences; 4,527,721 total letters

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

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

NRL_3D:1TTTC  Begin: 50 End: 304
!translation elongation factor EF-Tu, chain C - The...      148  3e-36
NRL_3D:1TTTB  Begin: 50 End: 304
!translation elongation factor EF-Tu, chain B - The...      148  3e-36
NRL_3D:1TTTA  Begin: 50 End: 304
!translation elongation factor EF-Tu, chain A - The...      148  3e-36
NRL_3D:1EFT  Begin: 50 End: 304
!translation elongation factor EF-Tu (with guanosine...     148  3e-36
NRL_3D:1TUIC  Begin: 42 End: 296
!translation elongation factor EF-Tu, chain C - The...      148  3e-36
NRL_3D:1TUIB  Begin: 42 End: 296
!translation elongation factor EF-Tu, chain B - The...      148  3e-36
NRL_3D:1TUIA  Begin: 42 End: 296
!translation elongation factor EF-Tu, chain A - The...      148  3e-36
NRL_3D:1B23P  Begin: 50 End: 304
!elongation factor tu, chain P - Thermus aquaticus          148  3e-36
NRL_3D:1D2ED  Begin: 25 End: 337
!elongation factor tu (ef-tu), chain D - bovine             146  9e-36
NRL_3D:1D2EC  Begin: 25 End: 337
!elongation factor tu (ef-tu), chain C - bovine             146  9e-36
NRL_3D:1D2EB
!elongation factor tu (ef-tu), chain B - bovine             146  9e-36
NRL_3D:1D2EA
!elongation factor tu (ef-tu), chain A - bovine             146  9e-36
NRL_3D:1AIPF2
!translation elongation factor EF-Tu, chain F, fra...       128  2e-30
NRL_3D:1AIPE2
!translation elongation factor EF-Tu, chain E, fra...       128  2e-30
```

```
NRL_3D:1AIPB2
!translation elongation factor EF-Tu, chain B, fra...      128  2e-30
NRL_3D:1AIPA2
!translation elongation factor EF-Tu, chain A, fra...      128  2e-30
NRL_3D:1EFCB
!elongation factor, chain B - bacteria                     124  4e-29
NRL_3D:1EFCA
!elongation factor, chain A - bacteria                     124  4e-29
NRL_3D:1DG1H
!elongation factor tu, chain H - bacteria                  124  4e-29
NRL_3D:1DG1G
!elongation factor tu, chain G - bacteria                  124  4e-29
NRL_3D:1EFUC2
!translation elongation factor EF-Tu, chain C, fra...      109  1e-24
NRL_3D:1EFUA2
!translation elongation factor EF-Tu, chain A, fra...      109  1e-24
NRL_3D:1ETU2
!translation elongation factor EF-Tu, domain I (wit...      68  3e-12
NRL_3D:1EFM2
!elongation factor Tu (trypsin-modified with GDP), ...      67  1e-11
\\End of List

>NRL_3D:1TTTC translation elongation factor EF-Tu, chain C - Thermus
           aquaticus
          Length = 405

 Score =  148 bits (369), Expect = 3e-36
 Identities = 99/271 (36%), Positives = 153/271 (55%), Gaps = 31/271 (11%)

Query: 40  LDQLKDERERGITINIALWKFETKKYIVTIIDAPGHRDFIKNMITGTSQADVAILVVAAG 99
           +D+  +ER RGITIN A  ++ET K   + +D PGH D+IKNMITG +Q D AILVV+A
Sbjct: 50  IDKAPEERARGITINTAHVEYETAKRHYSHVDCPGHADYIKNMITGAAQMDGAILVVSAA 109

Query: 100 QGEFEAGISKDGQTREHATLANTLGIKTMIICVNKMD-----------DGQVKYSKERYD 148
            G          QTREH  LA  +G+  +++ +NK+D           + +V+    +Y
Sbjct: 110 DGPMP-------QTREHILLARQVGVPYIVVFMNKVDMVDDPELLDLVEMEVRDLLNQY- 161

Query: 149 EIKGEMMKQLKNIGWKKAEEFDYIPTSGWTGDNIMEKSDKMPWYEGPCLIDAIDG-LKAP 207
           E  G+ +  ++       EE    P +   G+N  E  DK+ W     L+DAID  +  P
Sbjct: 162 EFPGDEVPVIRGSALLALEEMHKNPKTK-RGEN--EWVDKI-WE----LLDAIDEYIPTP 213

////////////////////////////////////////////////////////////////////////
```

The output is a perfectly suitable GCG list file, complete with beginning and ending attributes for those alignments specified and complementary strand attributes when necessary if using DNA. The pair-wise alignments requested are illustrated with identity positions highlighted by amino acid single letter symbols and similarity positions identified by plus signs. BLAST can even find more than one segment of alignment on the same sequence entry. This can be particularly helpful in those cases where the database entry is from genomic DNA and has several dispersed exons each with separate homologies to your query.

You will often be able to see somewhat of a demarcation where the Expectation values drop off between the significant hits and background noise. In our EF-1 protein case I expect to see the best E values for other EF-1 proteins and homologues, and then another bracket of very good values for other sequences with GTP-binding protein P-Loop signatures, and finally, a category of not so good scores that reflect background noise. However, in the BLASTP search of NRL_3D my Expectation cutoff of 0.10 restricted the search to only EF-1 /Tu proteins.

Next I'll show example FastA search output file of SW:EF1A_GIALA against the PIR database. Here you can see the first two E value brackets that I mention above, though there is no clean break between the two classes:

```
!!SEQUENCE_LIST 1.0

 (Peptide) FASTA of: input_20.rsf{ef1a_giala}  from: 1 to: 396  October 18, 2001 11:23

Description:  Q08046 giardia lamblia (giardia intestinalis). elongation factor 1-alpha (ef-1-a
Accession/ID: Q08046

====================General comments====================
```

```
ID   EF1A_GIALA     STANDARD;      PRT;   396 AA.

 TO: pir:*  Sequences:    232,624  Symbols:    80,607,033  Word Size: 2

 Databases searched:
   NBRF, Release 69.0, Released on 30Jun2001, Formatted on 10Sep2001

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

Histogram Key:
 Each histogram symbol represents 357 search set sequences
 Each inset symbol represents 11 search set sequences
 z-scores computed from opt scores

z-score obs    exp
        (=)    (*)

< 20    791      0:===
  22      0      0:
  24      4      0:=
  26      8      5:*
  28     44     53:*
  30    315    320:*
  32   1516   1236:===*=
  34   4135   3353:=========*==
  36   8281   6886:====================*====
  38  13459  11380:================================*======
  40  18055  15875:================================================*======
  42  20383  19405:==========================================================*===
  44  21415  21405:============================================================*
  46  21263  21802:============================================================*
  48  20066  20873:============================================================ *
  50  18038  19046:================================================  *
  52  15767  16745:============================================ *
  54  13502  14303:=====================================  *
  56  11090  11947:================================ *
  58   9139   9809:========================= *
  60   7496   7946:====================== *
  62   6041   6370:=================*
  64   4800   5066:==============*
  66   3737   4004:===========*
  68   2877   3149:========*
  70   2293   2468:======*
  72   1828   1929:=====*
  74   1294   1504:====*
  76   1057   1170:===*
  78    789    910:==*
  80    601    706:=*
  82    443    540:=*
  84    373    428:=*
  86    313    331:*
  88    204    256:*
  90    161    198:*
  92    121    153:*         :===========  *
  94     93    119:*         :========= *
  96     78     92:*         :=========*
  98     51     71:*         :===== *
 100     52     55:*         :====*
 102     39     43:*         :===*
 104     30     33:*         :==*
 106     23     25:*         :==*
 108     20     20:*         :=*
 110      9     15:*         :=*
 112      6     12:*         :=*
 114      7      9:*         :*
 116      2      7:*         :*
 118      1      5:*         :*
>120    514      4:*=        :*=======================================

Joining threshold: 37, opt. threshold: 25, opt. width:  16, reg.-scaled

The best scores are:                    init1 initn   opt    z-sc E(232096)..

PIR2:S70634    Begin: 4  End: 399
```

```
! translation elongation factor eEF-1... 2318  2318  2318  2569.3  6.1e-136
PIR2:S70635    Begin: 4  End: 399
! translation elongation factor eEF-1... 2125  2125  2125  2355.8  4.8e-124
PIR2:A54760    Begin: 21  End: 415
! translation elongation factor eEF-1...  996  1773  1816  2013.3  5.8e-105
PIR2:JC5117    Begin: 21  End: 415
! translation elongation factor eEF-1... 1003  1775  1799  1994.5  6.4e-104
PIR2:T43890    Begin: 1  End: 395
! translation elongation factor eEF-1...  986  1790  1792  1987.5  1.6e-103
PIR2:A49171    Begin: 22  End: 416
! translation elongation factor eEF-1...  971  1760  1785  1979.2  4.6e-103
PIR2:T43892    Begin: 1  End: 395
! translation elongation factor eEF-1...  990  1781  1783  1977.6  5.6e-103
PIR2:S16308    Begin: 21  End: 415
! translation elongation factor eEF-1...  983  1756  1776  1969.1  1.7e-102
PIR2:S11665    Begin: 24  End: 421
! translation elongation factor eEF-1...  838  1728  1737  1925.8  4.3e-100
PIR2:S07724    Begin: 21  End: 410
! translation elongation factor eEF-1...  922  1683  1708  1893.9  2.6e-98
PIR2:S10507    Begin: 21  End: 415
! translation elongation factor eEF-1...  932  1661  1696  1880.5  1.4e-97
//////////////////////////////////////////////////////////////////////////
PIR2:S57200    Begin: 21  End: 424
! translation elongation factor eEF-1...  846  1613  1596  1769.7  2.1e-91
PIR2:A49394    Begin: 1  End: 234
! translation elongation factor eEF-1... 1556  1556  1556  1729.7  3.6e-89
PIR2:S54734    Begin: 25  End: 418
! translation elongation factor aEF-1...  666  1426  1468  1628.4  1.6e-83
PIR2:S70636    Begin: 4  End: 275
! translation elongation factor eEF-1...  802  1420  1444  1604.8  3.2e-82
PIR2:H90162    Begin: 20  End: 413
! hypothetical protein tuF-1 [importe...  599  1374  1403  1556.6  1.6e-79
//////////////////////////////////////////////////////////////////////////
PIR2:T16218    Begin: 21  End: 427
! translation elongation factor EF-1 ...  799  1640   839   932.3  9.4e-45
PIR2:S26293    Begin: 1  End: 376
! translation elongation factor eEF-1...  834  1464   836   929.7  1.3e-44
PIR2:T51896    Begin: 311  End: 709
! probable translation release factor...  565   565   837   927.2  1.8e-44
PIR2:T23102    Begin: 123  End: 457
! hypothetical protein H19N07.1 - Cae...  579   637   819   908.8  1.9e-43
PIR2:T23393    Begin: 512  End: 918
! hypothetical protein K07A12.4 - Cae...  510   698   811   896.9  8.8e-43
PIR2:I54251    Begin: 5  End: 192
! translation elongation factor eEF-1...  600   739   749   837.2  1.9e-39
PIR2:T40165    Begin: 191  End: 519
! translation elongation factor eEF-1...  487   487   729   809.1  6.8e-38
PIR2:S38162    Begin: 181  End: 526
! translation elongation factor eEF-1...  465   529   594   659.5  1.5e-29
PIR1:S29293    Begin: 51  End: 324
! translation elongation factor EF-Tu...  202   431   507   565.8  2.4e-24
PIR1:S17146    Begin: 51  End: 356
! translation elongation factor EF-Tu...  202   429   506   564.7  2.8e-24
PIR1:S00229    Begin: 51  End: 356
! translation elongation factor EF-Tu...  202   429   506   564.7  2.8e-24
//////////////////////////////////////////////////////////////////////////
PIR1:EFEGT    Begin: 40  End: 385
! translation elongation factor EF-Tu...  188   370   441   492.8  2.8e-20
PIR2:S57945    Begin: 50  End: 364
! probable translation elongation fac...  175   366   432   482.9   1e-19
PIR2:S04391    Begin: 37  End: 371
! translation elongation factor EF-Tu...  185   363   431   481.7  1.2e-19
//////////////////////////////////////////////////////////////////////////
PIR2:G72243    Begin: 50  End: 343
! translation elongation factor EF-Tu...  194   405   315   353.5  1.6e-12
PIR2:A87433    Begin: 38  End: 350
! hypothetical protein CC1482 [import...  216   460   312   347.3  3.6e-12
//////////////////////////////////////////////////////////////////////////
PIR2:B86500    Begin: 40  End: 329
! elongation factor Tu [imported] - C...  198   453   271   305.0  8.2e-10
PIR2:A84979    Begin: 35  End: 375
! sulfate adenylyltransferase (EC 2.7...  177   407   268   300.5  1.5e-09
//////////////////////////////////////////////////////////////////////////
PIR2:T37295    Begin: 227  End: 492
! GTP-binding protein cgp-1 - Caenorh...   59   101   230   257.0  3.8e-07
PIR1:H69323    Begin: 103  End: 347
! translation initiation factor aIF-2...   80   173   226   254.7  5.2e-07
```

```
PIR1:D64457    Begin: 112  End: 373
! translation initiation factor aIF-2...   94   200   223   251.2  8.1e-07
PIR2:D86089    Begin: 50  End: 344
! hypothetical protein tufB [imported...  168   314   211   238.6  4.1e-06
PIR2:G64361    Begin: 16  End: 290
! translation initiation factor eIF-2...  156   320   211   237.5  4.7e-06
//////////////////////////////////////////////////////////////////////////
! tetracycline resistance protein tet...  156   156   192   214.5  8.9e-05
PIR2:D83046    Begin: 26  End: 256
! selenocysteine-specific elongation ...  111   180   191   213.4  0.0001
PIR1:E71178    Begin: 86  End: 322
! translation initiation factor aIF-2...  103   261   185   209.6  0.00017
PIR2:F75163    Begin: 86  End: 322
! translation initiation factor aif-2...  103   261   183   207.3  0.00022
PIR2:E64340    Begin: 136  End: 294
! hypothetical protein MJ0325 - Metha...  134   165   181   206.9  0.00024
PIR2:G84355    Begin: 87  End: 317
! hypothetical protein eif2g [importe...   87   184   181   205.1  0.0003
PIR2:A56779    Begin: 17  End: 185
! tetracycline resistance protein Tet...  146   146   183   204.6  0.00032
//////////////////////////////////////////////////////////////////////////
PIR1:A53048    Begin: 130  End: 367
! translation initiation factor eIF-2...   58    58   169   191.0  0.0018
PIR1:S46941    Begin: 129  End: 366
! translation initiation factor eIF-2...   81    81   168   189.8  0.0021
//////////////////////////////////////////////////////////////////////////
PIR2:T24472    Begin: 185  End: 522
! hypothetical protein T04H1.2 - Caen...   59    59   161   180.8  0.0068
PIR2:D89870    Begin: 24  End: 168
! peptide chain release factor 3 [imp...  131   155   160   180.4  0.0071
//////////////////////////////////////////////////////////////////////////
PIR2:H82039    Begin: 46  End: 367
! GTP-binding protein TypA VC2744 [im...   96   167   158   177.2   0.011
PIR2:A72233    Begin: 60  End: 299
! lepA protein - Thermotoga maritima ...   84   155   158   177.1   0.011
//////////////////////////////////////////////////////////////////////////
PIR2:E82295    Begin: 50  End: 173
! translation releasing factor RF-3 V...  123   123   151   170.3   0.026
PIR2:A81817    Begin: 496  End: 705
! translation initiation factor IF-2 ...  123   146   154   169.9   0.027
PIR2:G75524    Begin: 58  End: 262
//////////////////////////////////////////////////////////////////////////
PIR2:D96510    Begin: 79  End: 184
! probable mitochondrial elongation f...  117   117   144   160.4   0.092
PIR1:EFTWG    Begin: 35  End: 113
! translation elongation factor EF-G ...  126   126   143   159.8   0.099
\\End of List

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

input_20.rsf{ef1a_giala}
PIR2:G71023

P1;G71023 - translation elongation factor aEF-1 alpha chain - Pyrococcus
 horikoshii
C;Species: Pyrococcus horikoshii
C;Date: 14-Aug-1998 #sequence_revision 14-Aug-1998 #text_change 02-Feb-2001
C;Accession: G71023
R;Kawarabayasi, Y.; Sawada, M.; Horikawa, H.; Haikawa, Y.; Hino, Y.; Yamamoto,
 S.; Sekine, M.; Baba, S.; Kosugi, H.; Hosoyama, A.; Nagai, Y.; Sakai, M.;
 Ogura, K.; Otsuka, R.; Nakazawa, H.; Takamiya, M.; Ohfuku, Y.; Funahashi, T.;
 Tanaka, T.; Kudoh, Y.; Yamazaki, J.; Kushida, N.; Oguchi, A.; Aoki, K.;
 Yoshizawa, T.; Nakamura, Y.; Robb, F.T.; Horikoshi, K.; Masuchi, Y.; Shizuya,
 H.; Kikuchi, H.
DNA Res. 5, 55-76, 1998 . . .

SCORES   Init1: 365   Initn: 1130  Opt: 973   z-score: 1081.0 E(): 4.9e-53
>>PIR2:G71023                                              (428 aa)
 initn: 1130 init1: 365 opt: 973 Z-score: 1081.0 expect(): 4.9e-53
Smith-Waterman score: 1321;    51.9% identity in 405 aa overlap
 (1-396:21-410)

                                      10        20        30        40
input_20.rsf                       STLTGHLIYKCGGIDQRTIDEYEKRATEMGKGSFKYAWVL
                                   ||  |:|:|  |:| :  | ::|: : | || |||:|||:
G71023       MPKEKPHVNIVFIGHVDHGKSTTIGRLLYDTGNIPETIIKKFEEMG-EKGK-SFKFAWVM
                    10        20        30        40         50
```

```
                    50        60        70        80        90       100
input_20.rsf DQLKDERERGITINIALWKFETKKYIVTIIDAPGHRDFIKNMITGTSQADVAILVVAAGQ
             |:||:||||||||::|  |||| :  :|||||||||||:||||||:||||:|:||||| :
G71023       DRLKEERERGITIDVAHTKFETPHRYITIIDAPGHRDFVKNMITGASQADAAVLVVAATD
             60        70        80        90       100       110

                   110       120       130       140       150       160
input_20.rsf GEFEAGISKDGQTREHATLANTLGIKTMIICVNKMDDGQVKYSKERYDEIKGEMMKQLKN
             | :        ||:||| || ||||| :|: :||||  :|:|:::: ::::|::: | ||:
G71023       GVMP-------QTKEHAFLARTLGIKHIIVTINKMD--MVNYDQKVFEKVKAQVEKLLKT
           120              130       140          150       160

                   170       180       190       200       210       220
input_20.rsf IGWKKAEEFDYIPTSGWTGDNIMEKSDKMPWYEGPCLIDAIDGLKAPKRPTDKPLRLPIQ
             :|:|   :|  ||||:|:|||:::|||||||:|| ||:|:| :  |::| |||||:|||
G71023       LGYK---DFPVIPTSAWNGDNVVKKSDKMPWYNGPTLIEALDQIPEPEKPIDKPLRIPIQ
          170          180       190       200       210       220

                   230       240       250             260       270
input_20.rsf DVYKISGVGTVPAGRVETGELAPGMKVVFAPTSQV------SEVKSVEMHHEELKKAGPG
             |||:|:||||||:|||||||:|  |  |:| |:| :      :||||:|||||  |::| ||
G71023       DVYSIKGVGTVPVGRVETGKLKVGDVVIFEPASTIFHKPIQGEVKSIEMHHEPLQEALPG
              230       240       250       260       270       280

                 280       290       300          310       320       330
input_20.rsf DNVGFNVRGLAVKDLKKGYVVGDVTNDPPVGCK---SFTAQVIVMNHPKKIQPGYTPVID
             ||:|||||||:: :|:|:| |:|  |: ||:  :   :| ||:||:|||  |  ||:||:
G71023       DNIGFNVRGVSKNDIKRGDVAGH-TDKPPTVVRTKDTFKAQIIVLNHPTAITVGYSPVLH
              290       300       310       320       330       340

                   340       350       360       370       380       390
input_20.rsf CHTAHIACQFQLFLQKLDKRTLKPEMENPPDAGRGDCIIVKMVPQKPLCCETFNDYAPLG
              |||:|  :|: :| |:| || :   |||     ||  || : |:||:  |  ::   ||
G71023       AHTAQIPVRFEQILAKVDPRTGNIVEENPQFIKTGDSAIVVLRPMKPVVLEPVKEIPQLG
               350       360       370       380       390       400

input_20.rsf PFAVR
              ||:|
G71023       RFAIRDMGMTIAAGMVISIQKGE
               410       420

! Distributed over 1 thread.
!      Start time: Thu Oct 18 11:20:40 2001
! Completion time: Thu Oct 18 11:24:05 2001

! CPU time used:
!        Database scan:  0:01:34.7
! Post-scan processing:  0:00:11.7
!       Total CPU time:  0:01:46.3
! Output File: /users1/thompson/seqlab/ef1a_giala_20.fasta
```

The FastA output file is also an acceptable GCG list file with sequence attribute information that can serve as input to other GCG programs. The file shows a histogram of the score distribution, then a sorted list of the top scores and finally, if alignments are not suppressed with the -NoAlign option, a specified number of BestFit style alignments from the score list. These pairwise alignments show gaps as hyphens, identities as vertical bars and conservative replacement positions as colons. The histogram of the score distribution 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. 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. 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.

Another thing to notice in the output is that 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. This *z-score* is a bit different than the more traditional Monte Carlo style distribution Z score that I described in the introduction. Here it is calculated from a simple linear

regression against the natural log of the search set sequence length. (See William R. Pearson, Protein Science 4; 1145-1160 [1995] for an explanation of how this z-score is calculated.) Either type can be very helpful as they help describe the statistical significance of an alignment. Sometimes initial extended word scores, *initn*'s, are greatly improved after the *opt* dynamic programming and normalization pass. A good example from the above output is shown here:

```
PIR2:G72243    Begin: 50  End: 343
! translation elongation factor EF-Tu...  194   405   315   353.5  1.6e-12
PIR2:A87433    Begin: 38  End: 350
! hypothetical protein CC1482 [import...  216   460   312   347.3  3.6e-12
```

Notice PIR:A87433 had a higher *initn* score than PIR:G72243, yet its *opt* score is somewhat worse. This point underscores the importance of using appropriate options, this time taking advantage of the default -Opt option.

As in BLAST reports, the Expectation function, *E()*, is by far the most important column. It is very similar to the E value in BLAST reports and 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, just like with BLAST *E-values*, the smaller the number, the better. As a conservative rule-of-thumb, for a search against a protein database of around 10,000 sequences, as long as optimization is not turned off, *E()* scores 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. This was our most comprehensive search since I looked through all of PIR in it. That is why the scores fall off so slowly. In this case my Expectation cutoff of 0.10 got beyond the EF-1□/Tu sequences but did not get beyond all GTP-binding proteins.

Next let's take a look at the abridged TFastX output from the search of the Invertebrate nucleic acid database subdivision:

```
!!SEQUENCE_LIST 1.0

(Peptide) TFASTX of: input_19.rsf{ef1a_giala}  from: 1 to: 396  October 18, 2001 11:30

Description:  Q08046 giardia lamblia (giardia intestinalis). elongation factor 1-alpha (ef-1-a
Accession/ID: Q08046

====================General comments====================

ID   EF1A_GIALA     STANDARD;      PRT;   396 AA.

 TO: Invertebrate:*  Sequences:     95,557  Symbols:   471,181,336  Word Size: 2

 Databases searched:
   GenBank, Release 125.0, Released on 15Aug2001, Formatted on 30Aug2001

 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 151 search set sequences
 Each inset symbol represents 35 search set sequences
 z-scores computed from opt scores

z-score obs    exp
        (=)    (*)

< 20      2      0:=
  22      1      0:=
  24      2      0:=
  26     10      2:*
  28     53     21:*
  30    165    129:*=
  32    515    500:===*
  34   1217   1356:========*
  36   2616   2785:==================*
  38   4176   4603:============================  *
```

```
  40   6468   6421:=========================================*
  42   8362   7849:=============================================================*====
  44   8615   8658:============================================================*
  46   9027   8819:============================================================*=
  48   8867   8443:============================================================*===
  50   8000   7704:=========================================================*=
  52   7346   6773:=============================================*====
  54   5769   5785:=====================================*
  56   4795   4833:==================================*
  58   3938   3968:==========================*
  60   3030   3214:====================*
  62   2483   2577:=================*
  64   2034   2049:=============*
  66   1543   1620:==========*
  68   1160   1274:========*
  70    784    998:======*
  72    693    780:=====*
  74    561    608:====*
  76    400    473:===*
  78    293    368:==*
  80    185    286:=*
  82    173    219:=*
  84    124    173:=*
  86     89    134:*
  88     57    104:*
  90     74     80:*
  92     27     62:*         :=*
  94     18     48:*         :=*
  96     23     37:*         :=*
  98     24     29:*         :*
 100     31     22:*         :*
 102     32     17:*         :*
 104     18     13:*         :*
 106     11     10:*         :*
 108      7      8:*         :*
 110      4      6:*         :*
 112      5      5:*         :*
 114      4      4:*         :*
 116      4      3:*         :*
 118      8      2:*         :*
>120   1714      2:*=========:*======================================

Joining threshold: 37, opt. threshold: 25, opt. width:  16, reg.-scaled

The best scores are:                  strand init1 initn   opt    z-sc E(93881)..

GB_IN:GIAEF1A    Begin: 1  End: 1188
! D14342 G.lamblia mRNA for elongatio...(f)  2696  2696  2696  3581.8  9.9e-193
GB_IN:HSU29442    Begin: 11  End: 1198
! U29442 Diplomonad ATCC50330 elongat...(f)  2310  2310  2310  3067.9  4.2e-164
GB_IN:HIU37081    Begin: 11  End: 1198
! U37081 Hexamita inflata elongation ...(f)  2125  2125  2125  2821.6  2.2e-150
GB_IN:SVU94406    Begin: 11  End: 1204
! U94406 Spironucleus vortens elongat...(f)  1071  2068  2077  2757.7  8e-147
GB_IN:AF058283    Begin: 10  End: 1215
! AF058283 Naegleria andersoni elonga...(f)   802  1797  1821  2416.9  7.7e-128
GB_IN:TRBTEF1A    Begin: 11  End: 1195
! L25868 Trypanosoma brucei elongatio...(f)   996  1770  1813  2406.3  3e-127
GB_IN:D29834    Begin: 1  End: 1185
! D29834 Trypanosoma cruzi mRNA for e...(f)  1003  1788  1812  2405.0  3.5e-127
GB_IN:TRBEF1AE    Begin: 477  End: 1661
! L76077 Trypanosoma cruzi elongation...(f)  1003  1772  1796  2381.0  7.7e-126
GB_IN:AF230349    Begin: 1  End: 1185
! AF230349 Dinenympha exilis clone 1 ...(f)   986  1787  1789  2374.4  1.8e-125
////////////////////////////////////////////////////////////////////////////////
GB_IN:AF056107    Begin: 16  End: 1203
! AF056107 Stentor coeruleus translat...(f)   839  1636  1715  2275.7  5.6e-120
GB_IN:AF172083    Begin: 61  End: 1245
! AF172083 Paramecium tetraurelia tra...(f)   981  1689  1708  2266.0  1.9e-119
GB_IN:CPU71180    Begin: 263  End: 1435
! U71180 Crytposporidium parvum elong...(f)  1011  1665  1709  2265.5  2.1e-119
GB_IN:AF056096    Begin: 16  End: 1200
! AF056096 Blepharisma japonicum tran...(f)   938  1684  1707  2265.1  2.2e-119
////////////////////////////////////////////////////////////////////////////////
GB_IN:PFEF1    Begin: 378  End: 1556
! X60488 P.falciparum MEF-1 gene for ...(f)   922  1618  1665  2207.2  3.7e-116
GB_IN:PBAJ4150    Begin: 61  End: 1239
```

```
! AJ224150 Plasmodium berghei EF-1alp...(f)   915  1609  1656  2195.2  1.7e-115
GB_IN:PBAJ4151    Begin: 1113  End: 2291
! AJ224151 Plasmodium berghei EF-1alp...(f)   915  1609  1656  2192.2  2.5e-115
GB_IN:AF056100    Begin: 16  End: 1194
! AF056100 Euplotes aediculatus trans...(f)   850  1572  1647  2185.2  6.2e-115
GB_IN:AF240805    Begin: 5  End: 1093
! AF240805 Polyzonium germanicum elon...(f)   924  1559  1587  2105.9  1.6e-110
GB_IN:AF016242    Begin: 11  End: 1345
! AF016242 Dictyostelium discoideum p...(f)   838  1701  1567  2078.2  5.7e-109
//////////////////////////////////////////////////////////////////////////////
GB_IN:AF230353    Begin: 1  End: 870
! AF230353 Trichonympha agilis elonga...(f)   794  1311  1319  1750.4   1e-90
GB_IN:AF058284    Begin: 10  End: 822
! AF058284 Naegleria andersoni elonga...(f)   802  1268  1292  1714.7  9.9e-89
GB_IN:AF016243    Begin: 11  End: 1300
! AF016243 Physarum polycephalum prot...(f)   982  1744  1275  1689.6  2.5e-87
GB_IN:EIMDEVGEND    Begin: 3  End: 938
! M98839 Eimeria bovis developmental ...(f)  1175  1175  1252  1659.3  1.2e-85
GB_IN:AF124804    Begin: 1  End: 824
! AF124804 Amurotaenia decidua elonga...(f)   421  1022  1045  1385.9   2e-70
GB_IN:AF173393    Begin: 8  End: 994
! AF173393 Macrosoma sp. MC-2000 elon...(f)   800  1475   946  1253.1  5.1e-63
//////////////////////////////////////////////////////////////////////////////
GB_IN:AF234561    Begin: 20  End: 1240
! AF234561 Ceratomia catalpae elongat...(f)   895  1775   903  1194.6  9.2e-60
GB_IN:AF151631    Begin: 20  End: 1240
! AF151631 Heliothis terracottoides e...(f)   893  1787   903  1194.6  9.2e-60
GB_IN:HIU93087    Begin: 1  End: 534
! U93087 Hexamita inflata elongation ...(f)   901   901   901  1193.6  1.1e-59
GB_IN:AF003556    Begin: 19  End: 1239
! AF003556 Urosimulium aculeatum elon...(f)   890  1776   902  1193.3  1.1e-59
//////////////////////////////////////////////////////////////////////////////
GB_IN:AF300519    Begin: 4  End: 984
! AF300519 Alepidosceles sp. BBSL-221...(f)   558  1094   673   889.7  8.9e-43
GB_IN:AC005711    Begin: 26422  End: 27593  Strand: -
! AC005711 Drosophila melanogaster, c...(r)   556   657   694   889.7  8.9e-43
GB_IN:AF256527    Begin: 1  End: 567
! AF256527 Asternoseius sp. AL5467 el...(f)   558   738   670   888.7   1e-42
GB_IN:AF264789    Begin: 1  End: 1531
! AF264789 Lasioglossum conspicuum el...(f)   563  1482   674   888.6   1e-42
//////////////////////////////////////////////////////////////////////////////
GB_IN:AB002728    Begin: 25  End: 498
! AB002728 Entamoeba histolytica mRNA...(f)   582   624   628   833.5  1.2e-39
GB_IN:DRZ83648    Begin: 2  End: 418
! Z83648 D.rapae gene encoding elonga...(f)   616   616   625   830.5  1.8e-39
\\End of List

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

input_19.rsf{ef1a_giala}
GB_IN:AF056099

LOCUS       AF056099     1221 bp    DNA             INV       15-MAR-1999
DEFINITION  Euplotes aediculatus translation elongation factor 1-alpha (TEF1)
            gene, partial cds.
ACCESSION   AF056099
VERSION     AF056099.1  GI:4063577
KEYWORDS    . . . .


SCORES  Strand: (f) Init1: 935   Initn: 1615  Opt: 1684  z-score: 2234.4 E(): 1.
1e-117
>>GB_IN:AF056099                                          (1221 nt)
Frame: 1 initn: 1615 init1: 935 opt: 1684 Z-score: 2234.4 expect(): 1.1e-117
Smith-Waterman score: 1684;    61.9% identity in 396 aa overlap
 (1-396:16-1191)

                     10        20        30        40        50        60
input_19.rsf STLTGHLIYKCGGIDQRTIDEYEKRATEMGKGSFKYAWVLDQLKDERERGITINIALWKF
             || ||||||| || | |||:::||:::|||||:||||||||:|| ||||||||:||||||
AF056099     STTTGHLIYKLGGTDARTIEKFEKESAEMGKGTFKYAWVLDKLKAERERGITIDIALWKF
                    40        70        100       130       160       190

                     70        80        90       100       110       120
input_19.rsf ETKKYIVTIIDAPGHRDFIKNMITGTSQADVAILVVAAGQGEFEAGISKDGQTREHATLA
             || : : |||||||||||||||||||||:|||::|:|:|||||||||:||||||| ||
AF056099     ETTNRFYTIIDAPGHRDFIKNMITGTSQADAAILIIASGKGEFEAGISKEGQTREHALLA
```

```
                  220       250       280       310       340       370

                   130       140       150       160       170       180
input_19.rsf NTLGIKTMIICVNKMDDGQVKYSKERYDEIKGEMMKQLKNIGWKKAEEFDYIPTSGWTGD
             |:|:| |:: :|||| ::|:: || ||| |: : | ::|: |:: ::::| ||::||
AF056099     FTMGVKQMVVALNKMD--AAEYDETRYKEIKKEVSEYLDKVGY-KTDTMNFVPISGFNGD
                  400       430       460       490       520

                   190       200       210       220       230       240
input_19.rsf NIMEKSDKMPWYEGPCLIDAIDGLKAPKRPTDKPLRLPIQDVYKISGVGTVPAGRVETGE
             |::|:| :|||| || | :|:|::| |||| ||||||:||||||:|:|||||||||||
AF056099     NLLERSTNMPWYTGPTLTEALDSFKQPKRPILKPLRLPLQDVYKIGGIGTVPVGRVETGV
           550       580       610       640       670       700

                   250       260       270       280       290       300
input_19.rsf LAPGMKVVFAPTSQVSEVKSVEMHHEELKKAGPGDNVGFNVRGLAVKDLKKGYVVGDVTN
             |  |: ||||| :  :| |||||||| :::| ||:|||||||:|||:|:|:|:||  |
AF056099     LKSGIVVVFAPKGVSAECKSVEMHHEAVEEAIPGNNVGFNVKGLSVKDIKRGFVAGDSKN
           730       760       790       820       850       880

                   310       320       330       340       350       360
input_19.rsf DPPVGCKSFTAQVIVMNHPKKIQPGYTPVIDCHTAHIACQFQLFLQKLDKRTLKPEMENP
             |||:  ::|:|:|||:|||| :|: ||||||| |||||||:|: :| | |:|: |   : |
AF056099     DPPMDTENFVAHVIIMNHPGEIKAGYTPVIDVHTAHIACKFEELLTKADRRSGKKTDDPP
           910       940       970      1000      1030      1060

                   370       380       390
input_19.rsf PDAGRGDCIIVKMVPQKPLCCETFNDYAPLGPFAVR
                 ||   :::|| |||| |:|: ||||| ||||
AF056099     KFLKAGDAGQIRLVPTKPLCIENFSRYAPLGRFAVR
           1090      1120      1150      1180

! Distributed over 1 thread.
!      Start time: Thu Oct 18 11:19:32 2001
! Completion time: Thu Oct 18 11:38:16 2001

! CPU time used:
!        Database scan:  0:09:08.3
! Post-scan processing:  0:07:42.3
!       Total CPU time:  0:16:50.6
! Output File: /users1/thompson/seqlab/ef1a_giala_19.tfastx
```

TFastX output again shows a score distribution histogram, a sorted list with beginning and ending as well as strand attributes, and sequence alignments for as many pairs specified, unless you suppress them with the -NoAlign option. 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 same alignment cues are used as in FastA for the pair-wise alignment section, that is "-" gaps, "|"and ":" identity and similarity identifiers, as well as a new forward slash, "/," identifier that indicates frame shifts on the DNA strand necessary for the alignment to complete.

The scores fall off even more slowly in this search because there are so many elongation factor EF-1α's in the Invertebrate DNA database. In fact the search hits a program limit of 1000 on the number of sequences to report regardless of the Expectation cutoff used, and fails to get beyond EF-1α sequences at all! Both FastA and TFastX use the same type of *E()* significance statistic in their reports.

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

2.6. Interpreting Database Search Results — What is Significant?

We know how the EF-1α sequence aligns to itself and other close homologues; we know those alignments are significant. Those types of alignments don't cause anybody any problems; they're obvious. Therefore, we will try to use sequences where the similarity is not so obvious for the remainder of this section of the tutorial. Try to find interesting sequences

which are not recognizably from the same gene family as EF-1α yet still have somewhat decent scores. We are interested here in how some of the not so similar, as Russell Doolittle calls them 'twilight zone,' sequences, align and the significance of those alignments. Therefore, choose three completely different, hopefully, 'twilight zone' entries, one each, from each of your program runs, BLAST, FastA, and TFastX. This will be hard to do with the searches we did here since the EF-1α gene family is so huge most of our searches barely got beyond them if they did at all. I'll just go with the bottom entry from each list, but you can choose others as you wish. They should not be obvious EF-1α homologues, though you'll probably have to choose at least one; try for different types of proteins from each of the searches. There are no 'correct' answers here; we just want to see some interesting comparisons. Write down your choices. We will experiment with various methods for analyzing the significance of these sequences' similarity. Relevant lines showing my three choices from the search files follow below.

From the BLASTP search of NRL_3D I chose an Elongation Factor 1α orthologue, an EF-Tu fragment from *E. coli*. It has quite a significant E value but nothing else was available:

```
NRL_3D:1EFM2
!elongation factor Tu (trypsin-modified with GDP), ...      67  1e-11
```

And from the FastA output, The EF-1α paralogue, EF-G from *Thermus aquaticus*, with a much worse score:

```
PIR1:EFTWG    Begin: 35  End: 113
! translation elongation factor EF-G ...  126   126   143   159.8   0.099
```

The bottom of the TFastX Invertebrate search list was also an Elongation Factor, this time EF-1α from an Aphid. It has an extremely good Expectation value, but will serve as a worthwhile positive control:

```
GB_IN:DRZ83648    Begin: 2  End: 418
! Z83648 D.rapae gene encoding elonga...(f)   616   616   625   830.5  1.8e-39
```

And because I didn't find any negative controls for this portion of the tutorial, I ran one more TFastX search; this time of the Viral division of GenEMBL, just for fun. Only one sequence came up with an Expectation value less than 1.00. Relevant lines from that result follows

```
(Peptide) TFASTX of: ef1a_giala  from: 1 to: 396  October 19, 2001 11:11
TO: viral:*  Sequences:    130,083  Symbols:   114,600,849  Word Size: 2

GenBank, Release 125.0, Released on 15Aug2001, Formatted on 5Sep2001

GB_VI:SIVTBL209    Begin: 171  End: 413  Strand: -
! M61072 Simian immunodeficiency viru...(r)    51    51   109   139.4    0.77
```

Load each of your three sequence choices (and, if you would like, my negative control sequence, GB_VI:SIVTBL209) into SeqLab by going to the "**File**" "**Add sequences from**" "**Databases. . .**" menu. Merely type the name of the entry, including its database logical name and colon separator (logical_name:entry_name), 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 three sequences. They will be added to the bottom of the present dataset loaded in the SeqLab Editor. Double-click on each new entry's name or use the "**INFO**" icon with the sequence's name selected to read about each new sequence.

The sequences from Invertebrate and Viral are nucleic acid so an additional step is necessary. We need to translate the protein coding regions in them that correspond to the regions 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 Aphid EF-1α example is an easy one — there's only one CDS listed in the entry, so it should be the one that was found to be similar to

EF1A_GIALA, unless the similarity isn't real. Double-click anywhere within the sequence to launch the "**Sequence Features**" window, just as when we first began 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 that are relevant. In my Aphid EF-1 case there is only one CDS listed, but often with genomic DNA you have to choose every CDS of each exon associated with the particular gene that was found to be similar to your query. (Do not select "mRNA" or "exon" features — UTR's and/or splicing variants may mess you up.) Select the relevant CDS regions in your example. Be wary of translations that do not begin at position one. These are flagged in the entry's Features annotation with "/codon_start=2" or "3" and are often seen in cases such as this Aphid example where the actual CDS begins before the sequence begins and ends after the sequence ends: "CDS <1 . . >418." Since the Aphid CDS doesn't begin at position one, deselect the CDS region chosen in the Editor display by clicking once anywhere within the relevant sequence.

Go to the "**Edit**" menu and "**Select Range. . .**;" only that region that you want to translate by providing the correct beginning and ending numbers and then pressing "**Select**" and "**Close**" in turn. Next return to the "**Edit**" menu and select "**Translate. . .**;" specify "**Selected regions**" if asked. Press "**OK**" in the next window to translate (and concatenate all of the exons if you're dealing with that situation) the selected CDS region. The new protein sequence will appear at bottom of your SeqLab Editor display and now we're ready to start some in-depth pair-wise comparisons.

However, if a nucleotide database entry is not annotated with CDS feature data, such as the Simian Immune Deficiency Virus sequence that I am using as a negative control here, and as is the case in most of the tags database, then you have to translate the entry using some other criteria. TBLASTN and TFastX outputs should list beginning and ending attributes in the list portion of the file (unless suppressed) and they will indicate whether the similarity was found on the forward or reverse strand. One way to translate just the desired region is to select the DNA sequence and then click the "**PROTECT**" icon. Next push all the buttons on in the "Protections" window to allow all modifications and click "**OK**." You can then use the "**Edit**" "**Select Range. . .**" function to select the downstream, 3', region first that needs to be trimmed away. Select the region one base further than the area identified by the search algorithm all the way to end of the molecule; press "**Select**" and then "**Close**." Next you can use the "**CUT**" icon to trim that portion away; being sure to specify "**Selected regions**" when prompted, not "Selected sequences." You can then repeat this procedure on the upstream, 5', region that is not similar to your query to trim it away too. Next, if the sequence similarity was found on the reverse strand as it was in my SIV example, you need to use the "**Edit**" "**Reverse. . .**" menu and specify "**Reverse and Complement**." And finally, go to the "**Edit**" "**Translate. . .**" button and translate the "**First**" frame of the sequence by pressing the "**OK**" button. This will produce a translation of only that segment that the search algorithm identified.

2.6.1. Dot Matrix Methods. Compare and DotPlot — GCG's Implementation.

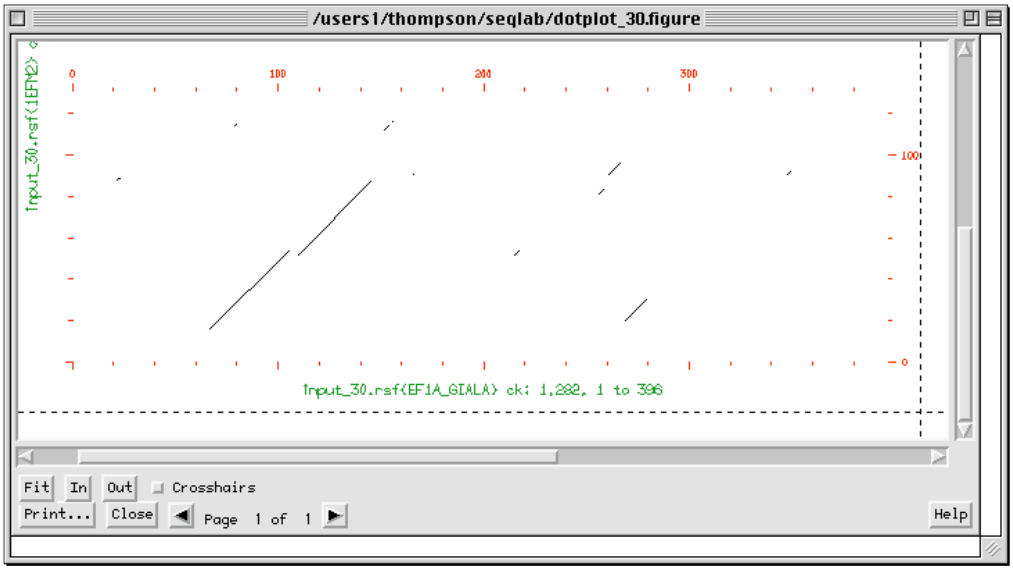
Dot matrix analysis is one of the few ways to identify other elements beyond what dynamic programming algorithms show to be similar between two sequences. GCG implements dot matrix methods with two programs. Compare generates the data that serves as input to DotPlot, which actually draws the matrix. Compare the your EF-1 query sequence to each of its three 'twilight zone,' near 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.)

You'll run the programs three times, once comparing your chosen EF-1 sequence to an interesting dataset sequence found by BLASTP, once to an analogous find by regular FastA, and finally another to a match found by TFastX. Start the program by selecting your query sequence and each new entry, one at a time per program run, pairwise in the SeqLab main Editor display. Select nonadjacent entries with the <**ctrl**> key. You may want to "**CUT**" and "**PASTE**" (it will go right below any sequence entry name that you have selected) or "**COPY**" your sequence to move it to the bottom of the

display near the database entries. Next go to the "**Functions**" menu and select "**Pairwise Comparison**" "**Compare. . .**" to produce a Compare program window. Notice that with "**DotPlot. . .**" checked the output from Compare will automatically be passed to DotPlot and the graphic will be drawn after the "**Run**" button is punched.

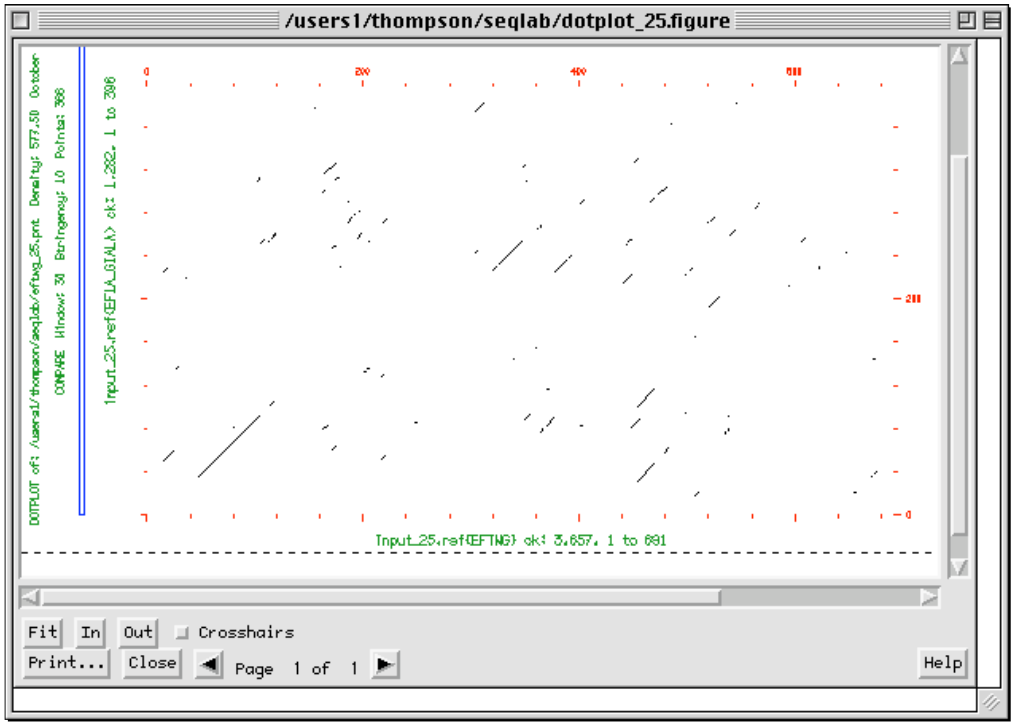
This will run the program at the GCG protein stringency default of 10 points within a window of 30 residues. That means wherever the average of BLOSUM62 match scores within the window is equal to or exceeds 10, a point will be drawn at the middle of the window, then the window is slid over one position at which point the process is repeated. Just as in all windowing algorithms, you want to use a window size 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 so short that size of window would cover much of the sequence, 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, *Giardia* EF-1α against the *E. coli* EF-Tu GTP-binding domain structure, I found the default stringency score of 10 within the window of 30 resulted in 109 points — almost the shorter sequence's length and of the correct magnitude. When run at the default stringency the graphic looks like the following:



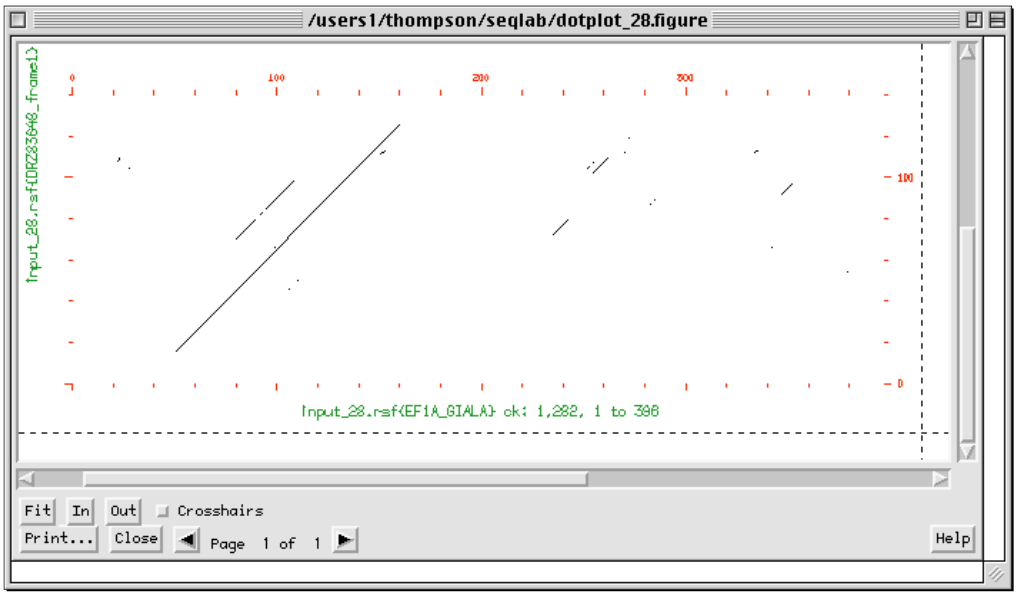
Notice that running the comparison at an appropriate stringency, in this case the default, produces a clean plot with little confusing noise. There is an obvious strong diagonal that clearly shows an alignment across much of the length of the NRL_3D sequence starting around residue 60 and running through residue 140 in the *Giardia* EF-1α sequence. There is also a short repeated domain further downstream. The section of *E. coli* EF-Tu from about position 20 to 30 corresponds to two sections of the Giardia sequence, a region at sequence positions 65 to 80 and also from about 265 to 280. These regions most likely correspond to GTP-binding motifs in both proteins. There is little doubt about the significance of this alignment as is reflected by its BLASTP E value of about 1.0×10^{-11} — they are both Elongation Factor 1α orthologues. 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.

To contrast the clear similarity seen above, I will illustrate my second example next, EF-1α against EF-G, two cross-phyla paralogues. The default parameters again worked fine resulting in 366 points. Take advantage of the "**GCG Defaults**" button to easily reset all of the program parameters, if you've previously changed them. That dotplot follows below:



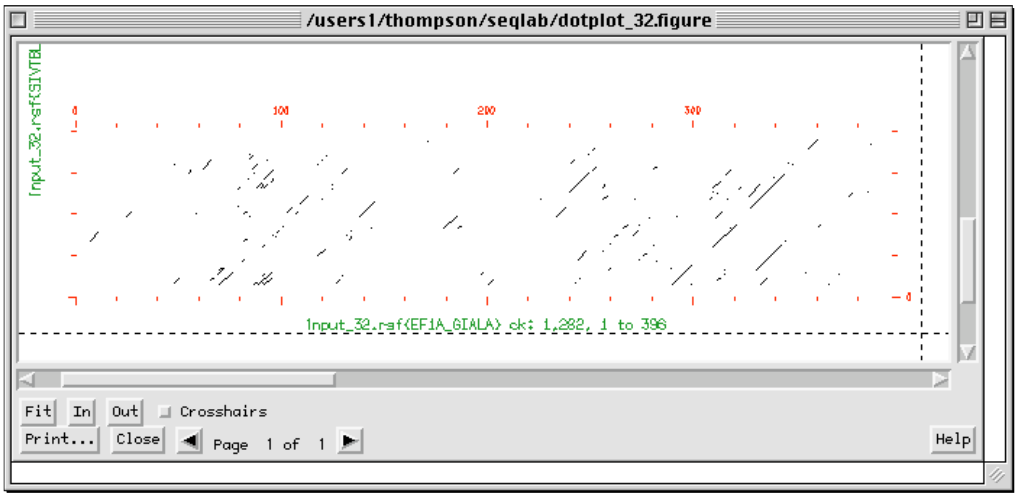
The only thing obvious here is a distinct region of both sequences that strongly align in the lower left-hand corner of the plot, probably that same GTP-binding domain. Many short direct repeats are scattered about elsewhere. Columns or rows of multiple diagonals always mean direct repeat sequences.

Running EF-1α against Aphid EF-1α, again with the default stringencies, produced the following plot:



This time it is obvious that the full length of the Aphid translation is almost identical to the amino half of the *Giardia* protein. No doubt about that and it is reflected in the extremely significant Expectation value of $1.8x10^{-39}$ for this comparison. However, notice that we still see direct repeat elements in the comparison. Dot matrix techniques are about the best available for recognizing repeats in biological sequences. When running all the dotplots, take notes of those particular regions in the proteins that have the longest running similarity. For example, as noted in the above plot, the region of EF1A_GIALA from about residue 50 through 160 has is almost identical to the full length of the Aphid sequence. We will need these numbers in the next section.

Finally, the negative control experiment. I had to adjust both the window size and the stringency of this comparison to pick up the signal in the relatively short SIV translation. Giardia EF-1□ compared and dotplotted to a hypothetical SIV CDS with a window of 11 and a stringency of 7 produces the following graphic:



Now a much foggier picture of the alignment emerges. Some small similarities and repeats are noticeable but nothing jumps right out at you. We'll test the longest diagonal obvious, that is EF1A_GIALA from about position 290 through 360 to the full length of my SIVTBL209 translation.

2.6.2. 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 scheme and BestFit is a 'local' algorithm, 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 the program for you; 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 located in the logical directory GenMoreData. 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 formats as an option; this can be handy as direct input to other GCG routines — particularly multiple sequence analysis programs.

BestFit and Gap both allow you to estimate significance. But what is significant? GCG provides an easy way to find out in these two programs. 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. 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. 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 more than five, then it most probably is significant, if it's above around nine, then certainly so. This distance above the mean is often known as a "Z score" and, as described in the Introduction, can be calculated with the following formula:

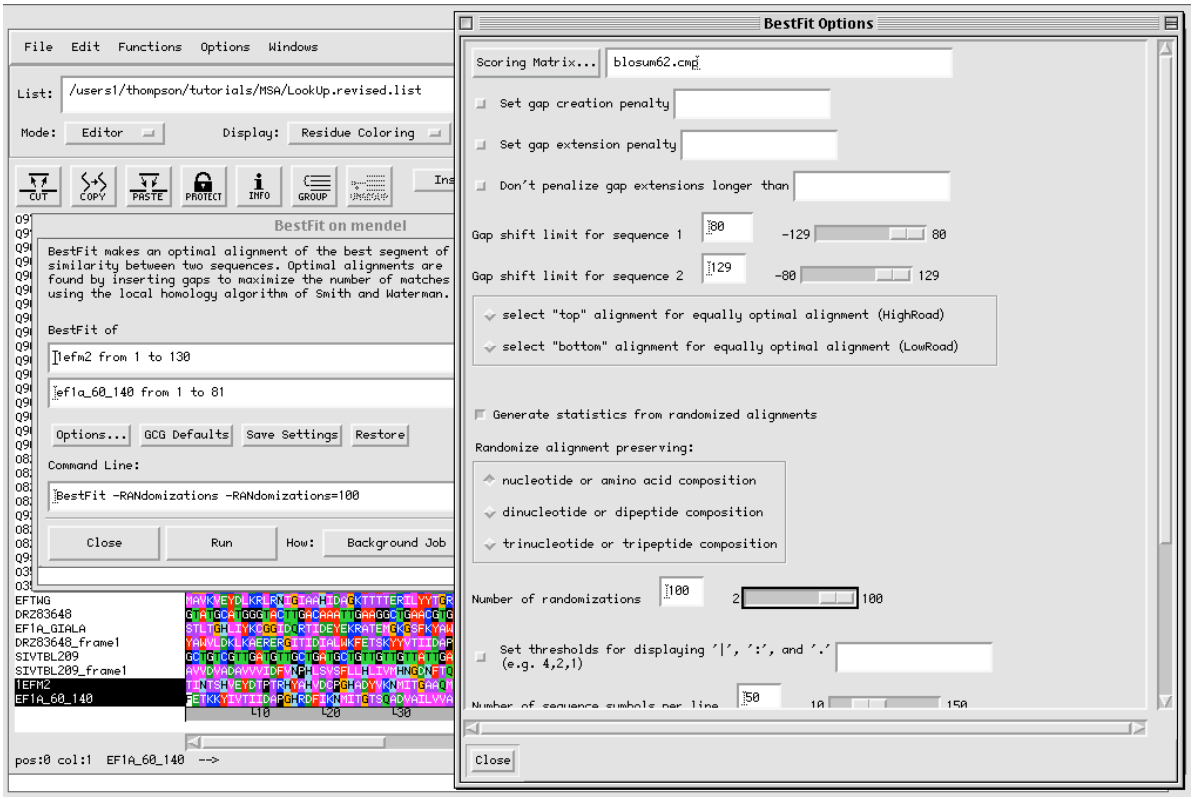
$$Z\ score = \frac{[\ (\ actual\ score\ ) - (\ mean\ of\ randomized\ scores\ )\ ]}{(\ standard\ deviation\ of\ randomized\ score\ distribution\ )}$$

This type of significance analysis is known as a Monte Carlo approach; it has many implicit statistical problems, however, few practical alternatives exist for comparing two sequences. I will use -Randomizations with BestFit to illustrate with the previous four comparisons. 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 noise into the analysis. 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 will not allow us to choose two different ranges on two different sequences, so we need to trick it into doing this analysis. Some things are still simpler from the command line where you can easily do this, and if you would rather use the command line for this portion of the tutorial, you are welcome to. A sample command line for my second comparison follows:

```
> bestfit pir:eftwg -begin1=40 -end1=80 sw:ef1a_giala -begin2=40 -end2=80 -random=100
```

In lieu of switching to the command line, first create new spaces to hold duplicate sequence data by going to the "**File**" "**New Sequence. . .**" menu and specifying "**Protein**." Repeat the procedure six (eight, if doing the negative control) times to create spots for the four pair-wise comparisons. Rename the newly created sequences so that you can recognize what they'll be by clicking on the "INFO" icon and changing their name fields. Next, select the sequence that you've based all of your searches on to start with, and then use the "**Edit**" function "**Select Range. . .**" to select just the desired region in it for your first Z test. In my first comparison, Giardia EF-1α against the *E. coli* EF-Tu GTP-binding domain, that roughly corresponds to a region from residue **60** through **140** of EF-1α and the full length of 1EFM2. Type the appropriate numbers into the "**Select Range**" "**Begin:**" and "**End:**" boxes and then press "**Select**" and "**Close**" to select the region. Press the "**COPY**" button, then answer "**Selected regions**" in the "**Which selection**" window that appears. Next select the new, empty EF-1α 60 through 140 sequence entry and place your cursor on the residue adjacent to the name in position one, then press the "**PASTE**" button. If you are asked "**Which clipboard**," answer "**Text clipboard**." This clipboard holds portions of a sequence, rather than an entire entry including its references. Repeat this procedure with the other sequence that you want to compare EF-1α to, if you are only using a range of it also. Select the new pair to analyze and return to the "**Functions**" "**Pairwise Comparison**" menu, only this time choose "**BestFit. . ..**" Press the "**Options**" button there to take advantage of -Randomizations. Don't mess with the top several options, but do check the box next to "**Generate statistics from randomized alignments**" and change the "**Number of randomizations**" up to "**100**." The display should look similar to the following screen snapshot:



"**Close**" the "**Options**" window. Press "**Run**" in the BestFit window and in a few moments your output file will appear.

Repeat this whole procedure with your other choices so that you end up with three (or four if you're doing my negative control test also) new pairs of sequences with just those portions of the originals that you want to test and three (or four) BestFit output files.

My first output file, all of 1EFM2 against a GTP-binding domain region of EF-1α, is shown below. Notice the 62% similarity is spread over much of the length analyzed. Also notice the high original quality and the low randomized quality. The Z score calculates to be 29.8; therefore, the interpretation is that the similarity is extremely significant and its BLAST Expectation value of 1.0×10^{-11} is corroborated.

```
BESTFIT of: input_33.rsf{1EFM2}  check: 222  from: 1  to: 130

Description:  elongation factor Tu (trypsin-modified with GDP), fragment 2 -
 Escherichia coli
Accession/ID: 000000

====================General comments====================

F1;1EFM2 - elongation factor Tu (trypsin-modified with GDP), fragment 2 -
 Escherichia coli

 to: input_33.rsf{EF1A_60_140}  check: 5792  from: 1  to: 81

Symbol comparison table: /usr/gcg/gcgcore/data/rundata/blosum62.cmp
 CompCheck: 1102

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

            Quality:    147             Length:     67
              Ratio:  2.450              Gaps:      1
 Percent Similarity: 61.667   Percent Identity: 53.333

 Average quality based on 100 randomizations: 24.9 +/- 4.1
```

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

input_33.rsf{1EFM2} x input_33.rsf{EF1A_60_140} October 19, 2001 18:10  ..

                .         .         .         .         .
     19 VDCPGHADYVKNMITGAAQMDGAILVVAATDGPMP.......QTREHILL 61
        :| ||| |::|||||| .| | |||||||  |          |||||  |
     11 IDAPGHRDFIKNMITGTSQADVAILVVAAGQGEFEAGISKDGQTREHATL 60
                .
     62 GRQVGVPYIIVFLNKCD 78
           .|:  .|: .|| |
     61 ANTLGIKTMIICVNKMD 77
```

EF-1α and EF-Tu are orthologues so this extremely significant alignment is not at all surprising.

The next one compares two cross-kingdom paralogues, a primitive Eukaryota protein, *Giardia*'s EF-1α, and a primitive [Eu]Bacterial protein, *Thermus aquaticus* EF-G. Here the same regions of both PIR:EFTWG and SW:EF1A_GIALA, residues 40 through 80, are tested against each other. The results of the sample BestFit command line from above follow below:

```
BESTFIT of: eftwg  check: 3657  from: 40  to: 80

P1;EFTWG - translation elongation factor EF-G - Thermus aquaticus
C;Species: Thermus aquaticus
C;Date: 30-Sep-1991 #sequence_revision 30-Sep-1991 #text_change 19-Jan-2001
C;Accession: S15928; S29294; S05978
R;Yakhnin, A.V.; Vorozheykina, D.P.; Matvienko, N.I.
Nucleic Acids Res. 17, 8863, 1989 . . .

 to: ef1a_giala  check: 1282  from: 40  to: 80

ID   EF1A_GIALA     STANDARD;      PRT;   396 AA.
AC   Q08046; Q94838;
DT   15-DEC-1998 (Rel. 37, Last annotation update)
DE   Elongation factor 1-alpha (EF-1-alpha) (14 NM filament-associated . . .

 Symbol comparison table: /usr/gcg/gcgcore/data/rundata/blosum62.cmp
 CompCheck: 1102

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

            Quality:     51             Length:     18
              Ratio:  2.833               Gaps:      0
 Percent Similarity: 66.667   Percent Identity: 55.556

 Average quality based on 100 randomizations: 18.2 +/- 3.6

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

 eftwg x ef1a_giala        October 19, 2001 18:40  ..

                .
     52 MDFMEQERERGITITAAV 69
        :| :. ||||||||  |.
     40 LDQLKDERERGITINIAL 57
```

Again the Z score for this domain turns out to be quite significant, 9.1, in spite of a mediocre FastA Expectation function, 0.099, and a very short homology 'patch.' Both sequences bind GTP in the same type of pocket, both have the same structural fold; therefore, those domains are clearly homologous.

The next comparison, two cross-phlya orthologues, is undoubtedly the most significant of all of them. This one had a tiny TFastX Expectation function of 1.8×10^{-39} so we expect a huge Z score:

```
BESTFIT of: prws.rsf{DRZ83648_frame1}  check: 4894  from: 1  to: 139

to: ef1a_giala  check: 1282  from: 50  to: 160

ID   EF1A_GIALA     STANDARD;      PRT;   396 AA.
AC   Q08046; Q94838;
DT   15-DEC-1998 (Rel. 37, Last annotation update)
DE   Elongation factor 1-alpha (EF-1-alpha) (14 NM filament-associated . . .

 Symbol comparison table: /usr/gcg/gcgcore/data/rundata/blosum62.cmp
 CompCheck: 1102

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

           Quality:    413             Length:    110
             Ratio:  3.755               Gaps:      0
 Percent Similarity: 80.909   Percent Identity: 72.727

 Average quality based on 100 randomizations: 27.3 +/- 4.9

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

 prws.rsf{DRZ83648_frame1} x ef1a_giala October 20, 2001 13:32  ..

                .         .         .         .         .
     15 GITIDIALWKFETSKYYVTIIDAPGHRDFIKNMITGTSQADCAVLIVAAG 64
        ||||.|||||||| || ||||||||||||||||||||||| |:|:||||
     50 GITINIALWKFETKKYIVTIIDAPGHRDFIKNMITGTSQADVAILVVAAG 99
                .         .         .         .         .
     65 TGEFEAGISKNGQTREHALLAFTLGVKQLIVGVNKMDSTEPPYSENRFEE 114
         |||||||||.|||||| || |||:| :|: :|||||  :  ||. |::|
    100 QGEFEAGISKDGQTREHATLANTLGIKTMIICVNKMDDGQVKYSKERYDE 149
                .
    115 IKKEVSSYIK 124
        || |.   :|
    150 IKGEMMKQLK 159
```

The Aphid EF-1α cross *Giardia* EF-1α alignment is >80% similar, >70% identical, over a length of over 100 residues. The Z score ends up 78.7, an enormously significant result.

Finally, my negative control experiment, the full length of my hypothetical Simian Immune Deficiency Virus translation against residues 290 through 360 of *Giardia* EF-1α:

```
BESTFIT of: prws.rsf{SIVTBL209_frame1}  check: 8711  from: 1  to: 74

 to: ef1a_giala  check: 1282  from: 290  to: 360

ID   EF1A_GIALA     STANDARD;      PRT;   396 AA.
AC   Q08046; Q94838;
DT   15-DEC-1998 (Rel. 37, Last annotation update)
DE   Elongation factor 1-alpha (EF-1-alpha) (14 NM filament-associated . . .

 Symbol comparison table: /usr/gcg/gcgcore/data/rundata/blosum62.cmp
 CompCheck: 1102

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

           Quality:     61             Length:     40
             Ratio:  1.564               Gaps:      1
 Percent Similarity: 38.462   Percent Identity: 35.897

 Average quality based on 100 randomizations: 20.6 +/- 3.9
```

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

prws.rsf{SIVTBL209_frame1} x ef1a_giala October 20, 2001 14:10  ..

                  .         .         .         .
     22 SFLLHLIVMHNGDNFTQGFIEVSKSCHTSSIACSVTVLFQ 61
        ||   .|||..      |:  |   |||. |||   .  |
    308 SFTAQVIVMNHPKKIQPGYTPVI.DCHTAHIACQFQLFLQ 346
```

Even this alignment turns out to be very significant, with a Z score of 10.4, in spite of its lousy TFastX Expectation function of 0.77 and a mediocre 36% identity over only 40 amino acids. So even though I expected this to be a negative control it clearly shows a region of EF-1 potentially homologous to a region of the Simian Immune Deficiency Virus Genome. What is interesting is that this is not the GTP-binding domain region identified in all of the previous comparisons. It also shows how Expectation values have to be considered in light of each search performed, completely dependent on the size and content of the database being searched and on how often you are performing that search.

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. Comparisons can turn out to be entirely insignificant in spite of what seems to be, upon first inspection, a very good alignment and a pretty high percent identity. A Monte Carlo style Z-test below around 3.5, near the bottom of Doolittle's "Twilight Zone," will often suggest that the similarity is not at all significant, that it is merely the result of compositional bias. As mentioned previously, the programs Xnu and Seg are now available in the Wisconsin Package outside of BLAST for pre-filtering your sequences. This is particularly prudent in situations with molecules where you know that a lot of repeat and/or low complexity sequence composition has the potential to confound search algorithms.

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 here. 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.

Next let's switch tracks. Rather than investigating pairwise comparisons anymore, let's move on to analyzing more than two sequences at a time. As mentioned in the Introduction, one of the harder aspects of multiple sequence alignment is knowing just what to align. In these days of huge genome projects and massive databases, one important slant to that is a data mining question, that is, figuring out just which sequences to align from a huge number available that are all homologous to your query. This question is particularly appropriate here since there are an enormous number of Elongation Factors present in the databases. So often it depends on the type of scientific question that you are asking in your research. Are you interested in predicting the structure or the function of your 'pet' molecule; how about in ascertaining the evolution of a paralogous gene family within a species as the result of gene duplications; what about the evolution of several species based on an analysis of the orthologues present in several different species? Clearly the dataset to be used is directly molded by the question that you ask.

This may be a logical break point between sessions. If you wish to finish up for the day, correctly leave SeqLab and pick up the tutorial at this point when you return. To do this exit SeqLab with the "**File**" menu "**Exit**" choice and save your RSF file and any changes in your list with appropriate responses. Accept the suggested changes and designate names that make sense to you; SeqLab will close. Log out of the current UNIX session on the GCG host that you were connected to and then log out of the workstation that you are sitting at.

2.7. Similarity Searching to Decrease (or Increase) Dataset Size.

A logical next step in preparing a multiple sequence alignment might be to run a similarity search to add those most similar sequences from the database to your dataset. As seen in previous sections, an advantage to running similarity searches within the context of GCG is the results are immediately available for further analyses without the need for any sequence downloading or reformatting because of the GCG list file format and the fact that all of the databases are mounted locally. In your own research settings, and depending on the type of questions that you are asking, you may want to create very large alignments by screening all available databases for sequences of significant similarity to your query. So let's talk about just how big you can go.

The Wisconsin Package's restrictions, as of version 10.3, allow individual sequences to be a maximum of 350 Kb in length (longer entries are cut into overlaps in database creation steps), though SeqLab can display longer sequences. You may want to load a longer sequence into SeqLab if you are working on the genome scale, and want to extract sub-ranges from that entry. The MSF file format can hold up to 500 sequences; RSF can hold much more, only limited by system memory. This allows programs such as HmmerAlign (described later) to produce multiple sequence alignment output larger than 500 sequences. PileUp itself can handle a sequence alignment up to 7,000 characters long, including gaps. Input sequences are restricted to a length of 5,000 characters by default. The 'overall surface-of-comparison' is restricted to 2,250,000 with the default program, a bit more than all the residues or bases plus all the gaps in the alignment. Alternative executables are provided with the Package for allowing 10,000, 15,000, and 20,000 character input, though these executables are usually not scripted into SeqLab. Launch them from the command line with "pileup_10000," "pileup_15000," and "pileup_20000" respectively. Take home message: You can make really huge alignments if you care to; it's all up to what you really need to do to answer the biological questions that you are asking.

But what about the opposite situation, when you have too many homologues? We'll return to FastA to illustrate this type of a data mining function next. FastA style database similarity searching can be very helpful for sorting any collection of GCG sequence specifications into order of alignment significance. This data mining function allows you to easily screen undesired sequences from the bottom of any list or combinations of lists. But, be warned, on some systems and some versions of GCG, you can not run FastA on too small of a dataset without causing core dumps! A trick is to add another small database such as NRL_3D, or the results of other database searches as I'll do here, to your Search List Set. This provides the necessary background randomization to allow proper normalization. Another point to remember is you can not use any of the BLAST programs to search against any sequence set that has not been preformatted into a BLAST compatible database. Because of this, BLAST is not an appropriate program to use for this type of list file sorting, data mining function. However, the FastA family of programs support all GCG sequence specifications, so it works great for this purpose.

Here I'll use FastA to search the existing LookUp output list file and the two previous protein database search output files from before all at the same time to narrow down all my data to a more manageable size. I will again use the *Giardia* sequence as a query for this search because many researchers consider *Giardia*'s most ancient ancestor to be rooted near the base of the universal tree of life on the eukaryote lineage (see e.g. Sogin, et al., 1996). Since my target dataset is all 'lower' eukaryotic, this seems like an appropriate choice.

If you're starting back up for the day, connect to the GCG server and relaunch SeqLab and then reselect your RSF dataset in "**List Mode**" and switch to "**Editor Mode**." Begin by s**electing all** of the sequences in the open SeqLab Editor. Several methods are available for selecting multiple sequence entry names. Either drag the mouse through them all (if they are all visible at once in the display), or <Shift>-click on the top- and bottom-most entries (select non-adjacent entries with <Cntrl>-clicks), or select "Select All" from the "Edit" menu. Deselect the "**EF1A_GIALA**" sequence entry name in the

Editor display by <**Ctrl**>-clicking its entry name. "**CUT**" all of the other sequences from the display. Now go back and select the remaining "**EF1A_GIALA**" sequence and launch FastA off the "**Functions**" "**Database Sequence Searching**" menu. 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 selected protein. At most sites the default protein database to search, "**Search Set. . .**," will be "**Using pir:***," but, as described above, here I'm going to change it to specify my LookUp output list file and the two protein database search result list files. Therefore, push the "**Search Set. . .**" button, select "**pir:***" in the "**Build FastA's Search Set**" box that pops up, and then "**Remove from Search Set**." Next, press the "**Add Main List Selection. . .**" button and then select your previous LookUp output list file from the "**List Chooser**" window that pops up; press "**Add to Search Set**." Repeat this process using the "**Add Sequences Files. . .**" button in the "**Build FastA's Search Set**" window, using the "**Filter**" function correctly to identify and load the two other database search files in your working directory. These will have names that end in "**.blast**" and "**.fasta**." "**Close**" the l**ist chooser** and the "**Build Search Set**" windows. Decrease the cutoff Expectation value in the main FastA window to something quite stringent like **0.01** to reduce the output list size. Be sure that the "**FastA**" program window shows "**How:**" "**Background Job**," and then press the "**Run**" button. The output will quickly return since it's a very small search set.

I searched a very small and definitely skewed collection of sequences; the scores again fall off quite slowly. Display your results and note the common players in this search from the previous ones. Another data mining trick that can be done at this point is to repeat the initial LookUp search, not back on a sequence database, but rather, on the new results of this latest FastA search. That would further restrict your dataset to only those taxonomic groups that you are interested in.

However you manage to get your dataset the size you want, you still need to load it into the SeqLab Editor. Use the "Output Manager" window again, always available through the SeqLab "Windows" menu. You need to use an extremely important Output Manager function at this point. Select your latest search output file in the "**Output Manager**" window and then press the "**Add to Editor**" button. Specify "**Overwrite old with new**" in the "**Reloading Same Sequence**" window when prompted, to take the search output and merge it with sequence already in the open Editor. Click "**Interrupt Loading**" in the "**Loading sequences**" window after as many sequences have loaded as you care to work with. If loading a FastA file, they are loaded in order of similarity to your query. In my example's case I restricted my analysis to about the top 50 entries of my final FastA file. The next prompt requires some thought if you're loading the results of a similarity search. You'll be asked whether to "Modify the sequences" or "Ignore all attributes" in a "**List file attributes set**" window. The answer will depend on the type of alignment you are creating and the biological questions that you asking. In many cases, especially if you are asking phylogenetic questions, then you will not want to modify the sequences. Load their full length to maximize available signal. However, if dealing with extremely diverse sequences and/or just domains of sequences, then trimming the sequences down to those most conserved portions identified by FastA can be very helpful. In this case I will not trim them down, so I press the "**Ignore all attributes**" button. "**Close**" the "**Output Manager**" after loading your new FastA list file and return your display to "**1:1**" and "**Residue Coloring**." Take a look at the new members in the display. As before, quickly double click on various entries' names to see the database reference descriptions for them (or click on the "INFO" button). The following graphic shows the Editor display after loading the top part of my latest FastA file:

Now would be a good time to go back to the "**File**" menu and save the RSF file. "**Overwrite**" in the "**File exists**" box if you've used the same name for this file earlier. I suggest that you do this, as RSF files are quite large and there's no need to save all the various versions of the data.

2.8. MEME.

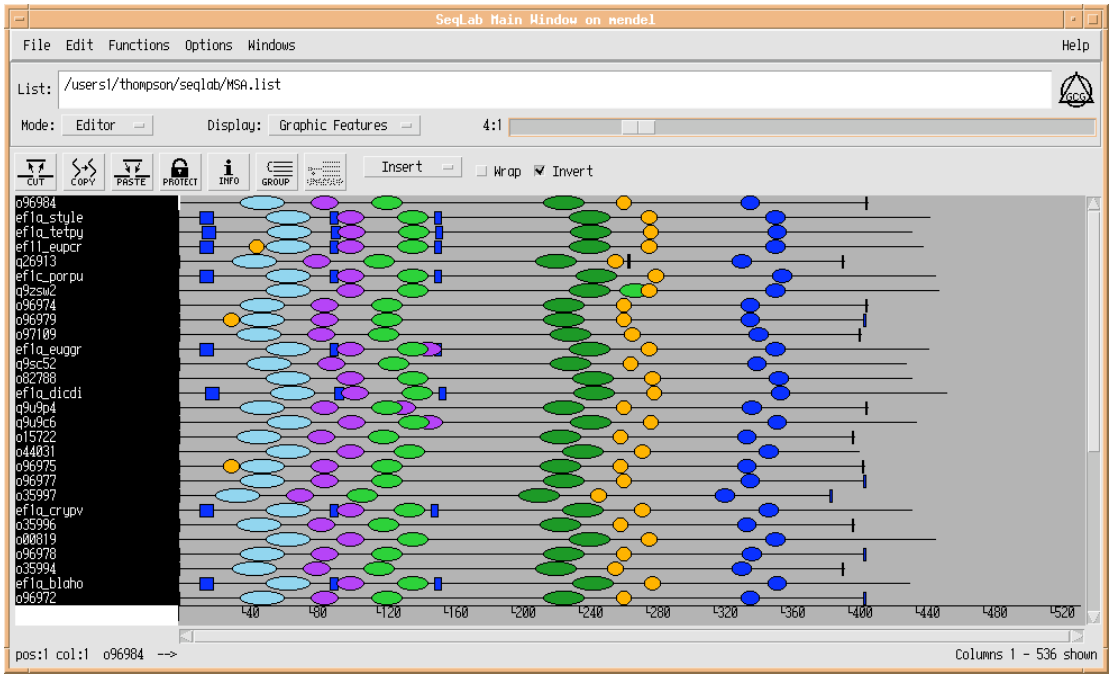
Before actually performing multiple sequence alignment on our dataset a powerful *de novo* motif discovery algorithm can be run. The algorithm is called Expectation Maximization; it uses Bayesian probabilities and unsupervised learning to find conserved motifs among a group of unaligned, ungapped sequences (Bailey and Elkan, 1994). The motifs do not have to be in congruent order among the different sequences; i.e. it has the power to discover 'unalignable' motifs between sequences. This characteristic differentiates MEME from most other profile building techniques. It is implemented in the Wisconsin Package as the MEME program and it produces output containing a multiple profile file as well as a readable report file. Its profile output serves as input to MotifSearch (Bailey and Gribskov, 1998). I would strongly suggest reading the MEME and MotifSearch chapters in the GCG Program Manual (genmanual at the command line or the Help buttons in the program in SeqLab) — they explain the details of the algorithms quite well.

Select all of the sequences in the Editor window so that MEME runs on them all. Launch "**MEME**" off of the "**Functions**" "**Multiple Comparisons**" menu. A "Which selection" window may pop 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 all the sequences. In most cases the default parameters will work fine but the algorithm can be sped up at the cost of sensitivity by decreasing the number of motifs to be found, by restricting the number of motifs found to exactly one in each sequence, and/or by decreasing the allowable motif window size. Again, I suggest reading the relevant GCG Program Manual chapters.

MEME output consists of two files; a .meme readable text file and a .prf multiple profile text file. MotifSearch will scan any dataset specified with the multiple profile file that MEME produced. A helpful thing to do is scan the original

'training' dataset that you created the profiles with. This will annotate those regions that MEME discovered in your SeqLab Editor RSF file. After alignment the MEME motifs that are alignable will all line up. Go to the "**Database Sequence Searching**" menu and select "**MotifSearch. . ..**" Specify your "**query profile(s)**," the one you just made, and change the "**Search set**" to the RSF dataset that you now have loaded in the Editor. Be sure to activate "**Save motif features to the RSF file**." The output will return with the .rsf file on top. Don't bother trying to read it; just "**Close**" it. It contains the SeqLab format "Rich Text Format" for all the feature data discovered by MEME in your dataset. The .ms file contains the readable results of the search in list file format with Expectation value statistics and the number of motif hits for each fit. After the list file portion a "Position diagram" schematically describes the hits in each sequence. Take a moment to look it over by pressing the "**Display**" button in the Output Manager and then "**Close**" it.

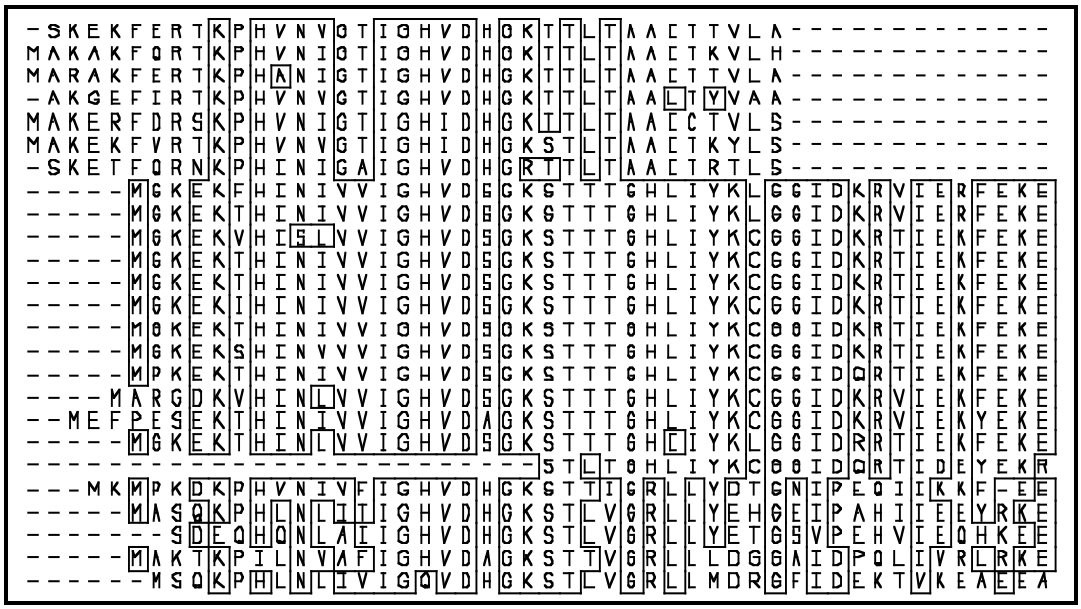
Use the Output Manager to merge the motifsearch.rsf feature file with the existing data already in the open SeqLab Editor. This will add the feature annotation created when I activated the MotifSearch -RSF option. The location of each motif will be included in the Editor sequence display. To do this again use the extremely important "**Add to Editor**" Output Manager function. As above, specify "**Overwrite old with new**" in the next window when prompted. "**Close**" the "**Output Manager**" after loading your new RSF file. Change "**Display:**" to "**Graphic Features**" and check out the additional annotation. The following figure illustrates my "Graphic Features" display at a "4:1" zoom ratio:



2.9. Searching PROSITE.

A 'quick and dirty' method — GCG's Motifs search.

Many, many features have been described and catalogued in biological sequences over the years. Most of these have recognizable consensus patterns that allow you to screen an unknown sequence for their occurrence. One very simplistic approach is to look at an alignment, see that certain regions are conserved, and create a consensus of that region. A multiple sequence alignment of Elongation Factor Tu/1 from many different organisms illustrates the conservation of the first of several GTP-binding domains in these proteins, that area around position twenty:



G H V D H G K S

Based on experimental evidence, we know that the indicated region bounded by the Glycine and Serine above is essential. So we merely count up the various residues in those locations and assign the most common one to the consensus. Simple. But what about the fact that the middle Histidine isn't always a Histidine; in this data set, just as often it's a Serine and sometimes it's an Alanine. Other positions are also seen not be invariant. And there's lots of other members of this gene family not being represented here at all. A consensus isn't necessarily the biologically "correct" combination. How do we include this other information? A simple consensus throws much of it away. Therefore, we need to adopt some sort of standardized ambiguity notation. The trick is to define a motif such that it minimizes false positives and maximizes true positives; i.e. it needs to be just discriminatory enough. The development of the exact motif is largely empirical; a pattern is made, tested against the database, then refined, over and over, although when experimental evidence is available, it is always incorporated. This approach is known as motif definition and fortunately a scientist in Switzerland, Dr. Amos Bairoch, has done it for tons of sequences!

His database of catalogued structural, regulatory, and enzymatic consensus patterns is a protein signature database, the *PROSITE Dictionary of Protein Sites and Patterns* (1992). Bairoch's compilation, now named the *PROSITE Database of protein families and domains*, contains 1079 documentation entries that describe 1459 different patterns, rules, and profile matrices (Release 16.33, of 25-Jan-2001). Pattern descriptions for these characteristic local sequence areas are variously and confusingly known as motifs, templates, signatures, patterns, and even fingerprints; don't let the terminology bewilder you. Those that GCG's Motifs program can access are one-dimensional descriptions, that encode ambiguity, of some sort of functional or otherwise constrained consensus region of a sequence alignment (e.g. glycosylation and phosphorylation sites, SH3-binding sites, nuclear localization sequence, and enzymatic active sites). Common motifs may or may not represent sequence homology and may or may not encompass an entire structural domain — they do not all signify known function nor common origin. Regardless, PROSITE 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. It can often lead to immediate answers and routes of investigation. It should always be utilized — it's just too fast and simple to ignore.

Start the Motifs program by selecting all of the protein entries' names in SeqLab, as in the previous MEME run, and then going to the "**Functions**" "**Protein Analysis**" menu and picking "**Motifs. . ..**" The "Motifs" program window will be

displayed. Check the "**Save results as features in file motifs.rsf**" button in the "Motifs" program window. As with MotifSearch, this file contains annotation discovered by the program and we'll use it below. None of the other options are required for this run 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 in my abridged Motifs output file below that have been characterized in these sequences and the extensive bibliography associated with them:

```
 MOTIFS from: @/users1/thompson/.seqlab-mendel/motifs_54.list

 Mismatches: 0               May 17, 2001 10:25  ..

input_54.rsf{GIARDIA_L}  Check: 6084  Length: 475   ! In situ PileUp of: @/users
1/thompson/.seqlab-mendel/pileup_36.list

_______________________________________________________________________________

Efactor_Gtp           D(K,R,S,T,G,A,N,Q,F,Y,W)x3E(K,R,A,Q)x(R,K,Q,D)(G,C)(I,V,M,
K)(S,T)(I,V)x2(G,S,T,A,C,K,R,N,Q)
                        D(Q)x{3}E(R)x(R)(G)(I)(T)(I)x{2}(A)
            64: YAWVL   DQLKDERERGITINIA   LWKFE

*****************************************
* GTP-binding elongation factors signature *
*****************************************

Elongation factors  [1,2]  are proteins  catalyzing the  elongation of peptide
chains in protein biosynthesis. In both prokaryotes and  eukaryotes, there are
three  distinct types  of elongation  factors,  as  described in the following
table:

 ------------------------------------------------------------------------
 Eukaryotes   Prokaryotes   Function
 ------------------------------------------------------------------------
 EF-1alpha    EF-Tu         Binds GTP and  an aminoacyl-tRNA;  delivers  the
                            latter to the A site of ribosomes.
 EF-1beta     EF-Ts         Interacts with EF-1a/EF-Tu  to  displace GDP and
                            thus allows the regeneration of GTP-EF-1a.
 EF-2         EF-G          Binds GTP and peptidyl-tRNA and translocates the
                            latter from the A site to the P site.
 ------------------------------------------------------------------------

The GTP-binding elongation factor family also includes the following proteins:

 - Eukaryotic  peptide  chain  release  factor GTP-binding subunits [3]. These
   proteins interact  with  release  factors  that bind to ribosomes that have
   encountered a  stop  codon  at  their decoding site and help them to induce
   release of  the  nascent  polypeptide.  The yeast protein was known as SUP2
   (and also as SUP35, SUF12 or GST1) and the human homolog as GST1-Hs.
 - Prokaryotic  peptide  chain release factor 3 (RF-3) (gene prfC).  RF-3 is a
   class-II RF, a GTP-binding  protein  that  interacts  with class I RFs (see
   <PDOC00607>) and enhance their activity [4].
 - Prokaryotic GTP-binding protein lepA and its  homolog  in yeast (gene GUF1)
   and in Caenorhabditis elegans (ZK1236.1).
 - Yeast HBS1 [5].
 - Rat statin S1 [6], a protein of unknown function which is highly similar to
   EF-1alpha.
 - Prokaryotic selenocysteine-specific elongation factor selB [7], which seems
   to replace  EF-Tu  for  the insertion of selenocysteine directed by the UGA
   codon.
 - The tetracycline resistance proteins tetM/tetO [8,9] from various  bacteria
   such as  Campylobacter jejuni, Enterococcus faecalis,  Streptococcus mutans
   and Ureaplasma urealyticum. Tetracycline binds to the prokaryotic ribosomal
   30S subunit and inhibits binding of aminoacyl-tRNAs. These proteins abolish
   the inhibitory effect of tetracycline on protein synthesis.
 - Rhizobium nodulation protein nodQ [10].
 - Escherichia coli hypothetical protein yihK [11].

In EF-1-alpha, a specific region  has  been  shown [12] to  be  involved  in a
conformational change mediated by the hydrolysis of GTP to GDP. This region is
conserved in  both EF-1alpha/EF-Tu as well as EF-2/EF-G and thus seems typical
for GTP-dependent proteins which bind non-initiator tRNAs to the ribosome. The
pattern we  developed  for  this  family  of  proteins  include that conserved
region.
```

```
-Consensus pattern: D-[KRSTGANQFYW]-x(3)-E-[KRAQ]-x-[RKQD]-[GC]-[IVMK]-[ST]-
                    [IV]-x(2)-[GSTACKRNQ]
-Sequences known to belong to this class detected by the pattern: ALL,  except
 for 11 sequences.
-Other sequence(s) detected in SWISS-PROT: NONE.
-Last update: November 1997 / Text revised.

[ 1] Concise Encyclopedia Biochemistry, Second Edition, Walter de Gruyter,
     Berlin New-York (1988).
[ 2] Moldave K.
     Annu. Rev. Biochem. 54:1109-1149(1985).
[ 3] Stansfield I., Jones K.M., Kushnirov V.V., Dagkesamanskaya A.R.,
     Poznyakovski A.I., Paushkin S.V., Nierras C.R., Cox B.S.,
     Ter-Avanesyan M.D., Tuite M.F.
     EMBO J. 14:4365-4373(1995).
[ 4] Grentzmann G., Brechemier-Baey D., Heurgue-Hamard V., Buckingham R.H.
     J. Biol. Chem. 270:10595-10600(1995).
[ 5] Nelson R.J., Ziegelhoffer T., Nicolet C., Werner-Washburne M., Craig E.A.
     Cell 71:97-105(1992).
[ 6] Ann D.K., Moutsatsos I.K., Nakamura T., Lin H.H., Mao P.-L., Lee M.-J.,
     Chin S., Liem R.K.H., Wang E.
     J. Biol. Chem. 266:10429-10437(1991).
[ 7] Forchammer K., Leinfeldr W., Bock A.
     Nature 342:453-456(1989).
[ 8] Manavathu E.K., Hiratsuka K., Taylor D.E.
     Gene 62:17-26(1988).
[ 9] Leblanc D.J., Lee L.N., Titmas B.M., Smith C.J., Tenover F.C.
     J. Bacteriol. 170:3618-3626(1988).
[10] Cervantes E., Sharma S.B., Maillet F., Vasse J., Truchet G., Rosenberg C.
     Mol. Microbiol. 3:745-755(1989).
[11] Plunkett G. III, Burland V.D., Daniels D.L., Blattner F.R.
     Nucleic Acids Res. 21:3391-3398(1993).
[12] Moller W., Schipper A., Amons R.
     Biochimie 69:983-989(1987).
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

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

input_54.rsf{CRYPTOSPORIDIUM_P}  Check: 6774  Length: 475   ! In situ PileUp of:
 @/users1/thompson/.seqlab-mendel/pileup_36.list
```

```
Atp_Gtp_A             (A,G)x4GK(S,T)
                       (G)x{4}GK(S)
            17: NLVVI    GHVDSGKS    TTTGH

****************************************
* ATP/GTP-binding site motif A (P-loop) *
****************************************
```

From sequence comparisons and crystallographic data analysis it has been shown [1,2,3,4,5,6] that an appreciable proportion of proteins that bind ATP or GTP share a number of more or less conserved sequence motifs. The best conserved of these motifs is a glycine-rich region, which typically forms a flexible loop between a beta-strand and an alpha-helix. This loop interacts with one of the phosphate groups of the nucleotide. This sequence motif is generally referred to as the 'A' consensus sequence [1] or the 'P-loop' [5].

There are numerous ATP- or GTP-binding proteins in which the P-loop is found. We list below a number of protein families for which the relevance of the presence of such motif has been noted:

- ATP synthase alpha and beta subunits (see <PDOC00137>).
- Myosin heavy chains.
- Kinesin heavy chains and kinesin-like proteins (see <PDOC00343>).
- Dynamins and dynamin-like proteins (see <PDOC00362>).
- Guanylate kinase (see <PDOC00670>).
- Thymidine kinase (see <PDOC00524>).
- Thymidylate kinase (see <PDOC01034>).
- Shikimate kinase (see <PDOC00868>).
- Nitrogenase iron protein family (nifH/frxC) (see <PDOC00580>).
- ATP-binding proteins involved in 'active transport' (ABC transporters) [7] (see <PDOC00185>).
- DNA and RNA helicases [8,9,10].
- GTP-binding elongation factors (EF-Tu, EF-1alpha, EF-G, EF-2, etc.).

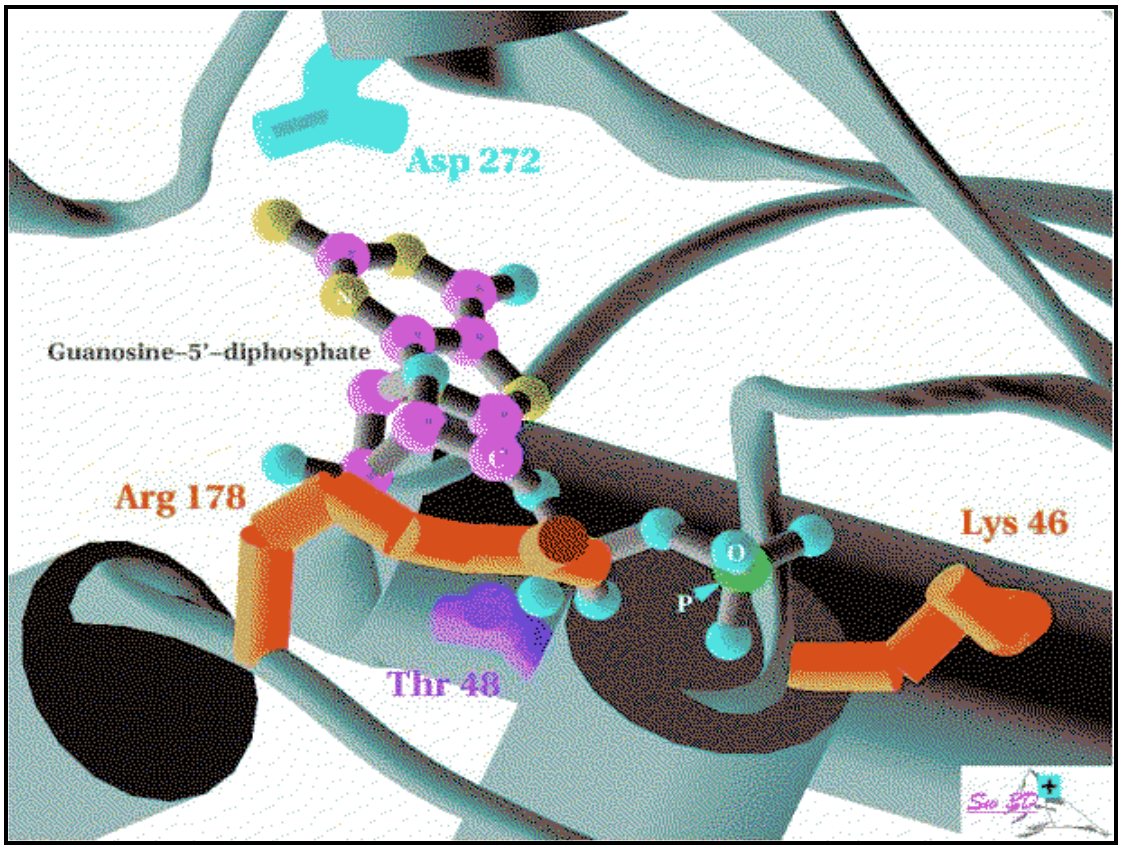
```
 - Ras family of GTP-binding proteins (Ras, Rho, Rab, Ral, Ypt1, SEC4, etc.).
 - Nuclear protein ran (see <PDOC00859>).
 - ADP-ribosylation factors family (see <PDOC00781>).
 - Bacterial dnaA protein (see <PDOC00771>).
 - Bacterial recA protein (see <PDOC00131>).
 - Bacterial recF protein (see <PDOC00539>).
 - Guanine nucleotide-binding proteins alpha subunits (Gi, Gs, Gt, G0, etc.).
 - DNA mismatch repair proteins mutS family (See <PDOC00388>).
 - Bacterial type II secretion system protein E (see <PDOC00567>).

Not all ATP- or GTP-binding proteins are picked-up by this motif.  A number of
proteins escape detection because the structure   of their ATP-binding site is
completely different from that of the P-loop.  Examples  of  such proteins are
the E1-E2 ATPases or  the  glycolytic kinases.   In  other ATP- or GTP-binding
proteins the flexible loop exists  in a  slightly different form; this is  the
case for tubulins or protein kinases.  A special mention must  be reserved for
adenylate  kinase,  in  which  there  is a  single  deviation  from the P-loop
pattern: in the last position Gly is found instead of Ser or Thr.

-Consensus pattern: [AG]-x(4)-G-K-[ST]
-Sequences known to belong to this class detected by the pattern: a majority.
-Other sequence(s) detected in SWISS-PROT: in addition to the proteins  listed
 above,  the 'A' motif is also  found in a number  of other proteins.  Most of
 these proteins  probably  bind  a nucleotide, but others are definitively not
 ATP- or GTP-binding (as for example  chymotrypsin,  or  human  ferritin light
 chain).

-Expert(s) to contact by email:
           Koonin E.V.; koonin@ncbi.nlm.nih.gov

-Last update: July 1999 / Text revised.

[ 1] Walker J.E., Saraste M., Runswick M.J., Gay N.J.
     EMBO J. 1:945-951(1982).
[ 2] Moller W., Amons R.
     FEBS Lett. 186:1-7(1985).
[ 3] Fry D.C., Kuby S.A., Mildvan A.S.
     Proc. Natl. Acad. Sci. U.S.A. 83:907-911(1986).
[ 4] Dever T.E., Glynias M.J., Merrick W.C.
     Proc. Natl. Acad. Sci. U.S.A. 84:1814-1818(1987).
[ 5] Saraste M., Sibbald P.R., Wittinghofer A.
     Trends Biochem. Sci. 15:430-434(1990).
[ 6] Koonin E.V.
     J. Mol. Biol. 229:1165-1174(1993).
[ 7] Higgins C.F., Hyde S.C., Mimmack M.M., Gileadi U., Gill D.R.,
     Gallagher M.P.
     J. Bioenerg. Biomembr. 22:571-592(1990).
[ 8] Hodgman T.C.
     Nature 333:22-23(1988) and Nature 333:578-578(1988) (Errata).
[ 9] Linder P., Lasko P., Ashburner M., Leroy P., Nielsen P.J., Nishi K.,
     Schnier J., Slonimski P.P.
     Nature 337:121-122(1989).
[10] Gorbalenya A.E., Koonin E.V., Donchenko A.P., Blinov V.M.
     Nucleic Acids Res. 17:4713-4730(1989).
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

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

input_54.rsf{BLASTOCYSTIS_H}  Check: 172   Length: 475   ! In situ PileUp of: @/
users1/thompson/.seqlab-mendel/pileup_36.list

Atp_Gtp_A             (A,G)x4GK(S,T)
                       (G)x{4}GK(S)
            17: NLVVI    GHVVAGKS    TTTGH

Find reference above under sequence: input_54.rsf{CRYPTOSPORIDIUM_P}, pattern: A
tp_Gtp_A.

____________________________________________________________________________

Prokar_Lipoprotein    ~(D,E,R,K)6(L,I,V,M,F,W,S,T,A,G)2(L,I,V,M,F,Y,S,T,A,G,C,Q)
(A,G,S)C
                                        ~(D,E,R,K){6}(L,I){2}(Y)(A)C
           24: VVAGK                            STTTGHLIYAC
         GGIDK

********************************************************
```

```
* Prokaryotic membrane lipoprotein lipid attachment site *
**********************************************************

In prokaryotes, membrane lipoproteins are synthesized  with a precursor signal
peptide, which is cleaved  by  a specific lipoprotein signal peptidase (signal
peptidase II). The peptidase recognizes a conserved sequence and cuts upstream
of a cysteine residue  to which a  glyceride-fatty acid lipid is attached [1].
Some of  the  proteins known to undergo such processing currently include (for
recent listings see [1,2,3]):

 - Major outer membrane lipoprotein (murein-lipoproteins) (gene lpp).
 - Escherichia coli lipoprotein-28 (gene nlpA).
 - Escherichia coli lipoprotein-34 (gene nlpB).
 - Escherichia coli lipoprotein nlpC.
 - Escherichia coli lipoprotein nlpD.
 - Escherichia coli osmotically inducible lipoprotein B (gene osmB).
 - Escherichia coli osmotically inducible lipoprotein E (gene osmE).
 - Escherichia coli peptidoglycan-associated lipoprotein (gene pal).
 - Escherichia coli rare lipoproteins A and B (genes rplA and rplB).
 - Escherichia coli copper homeostasis protein cutF (or nlpE).
 - Escherichia coli plasmids traT proteins.
 - Escherichia coli Col plasmids lysis proteins.
 - A number of Bacillus beta-lactamases.
 - Bacillus subtilis periplasmic oligopeptide-binding protein (gene oppA).
 - Borrelia burgdorferi outer surface proteins A and B (genes ospA and ospB).
 - Borrelia hermsii variable major protein 21 (gene vmp21) and 7 (gene vmp7).
 - Chlamydia trachomatis outer membrane protein 3 (gene omp3).
 - Fibrobacter succinogenes endoglucanase cel-3.
 - Haemophilus influenzae proteins Pal and Pcp.
 - Klebsiella pullulunase (gene pulA).
 - Klebsiella pullulunase secretion protein pulS.
 - Mycoplasma hyorhinis protein p37.
 - Mycoplasma hyorhinis variant surface antigens A, B, and C (genes vlpABC).
 - Neisseria outer membrane protein H.8.
 - Pseudomonas aeruginosa lipopeptide (gene lppL).
 - Pseudomonas solanacearum endoglucanase egl.
 - Rhodopseudomonas viridis reaction center cytochrome subunit (gene cytC).
 - Rickettsia 17 Kd antigen.
 - Shigella flexneri invasion plasmid proteins mxiJ and mxiM.
 - Streptococcus pneumoniae oligopeptide transport protein A (gene amiA).
 - Treponema pallidium 34 Kd antigen.
 - Treponema pallidium membrane protein A (gene tmpA).
 - Vibrio harveyi chitobiase (gene chb).
 - Yersinia virulence plasmid protein yscJ.

 - Halocyanin from Natrobacterium pharaonis [4], a membrane associated copper-
   binding protein.  This  is  the  first archaebacterial  protein known to be
   modified in such a fashion).

From  the  precursor sequences  of all  these proteins, we derived a consensus
pattern and  a  set  of  rules  to  identify  this  type of post-translational
modification.

-Consensus pattern: {DERK}(6)-[LIVMFWSTAG](2)-[LIVMFYSTAGCQ]-[AGS]-C
                    [C is the lipid attachment site]
 Additional rules:  1) The cysteine must be between positions 15 and 35 of the
                       sequence in consideration.
                    2) There must be at least one Lys or one Arg in the first
                       seven positions of the sequence.
-Sequences known to belong to this class detected by the pattern: ALL.
-Other sequence(s) detected in SWISS-PROT: some 100 prokaryotic proteins. Some
 of them are not membrane lipoproteins, but at least half of them could be.
-Last update: November 1995 / Pattern and text revised.

[ 1] Hayashi S., Wu H.C.
     J. Bioenerg. Biomembr. 22:451-471(1990).
[ 2] Klein P., Somorjai R.L., Lau P.C.K.
     Protein Eng. 2:15-20(1988).
[ 3] von Heijne G.
     Protein Eng. 2:531-534(1989).
[ 4] Mattar S., Scharf B., Kent S.B.H., Rodewald K., Oesterhelt D.,
     Engelhard M.
     J. Biol. Chem. 269:14939-14945(1994).
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

////////////////////////////////////////////////////////////////////////////
```

```
input_54.rsf{PHYTOPHTHORA_I}  Check: 9509  Length: 475   ! In situ PileUp of: @/
users1/thompson/.seqlab-mendel/pileup_36.list

Atp_Gtp_A             (A,G)x4GK(S,T)
                       (G)x{4}GK(S)
            17: ...VI    GHVDAGKS    TTTGH

Find reference above under sequence: input_54.rsf{CRYPTOSPORIDIUM_P}, pattern: A
tp_Gtp_A.


Efactor_Gtp           D(K,R,S,T,G,A,N,Q,F,Y,W)x3E(K,R,A,Q)x(R,K,Q,D)(G,C)(I,V,M,
K)(S,T)(I,V)x2(G,S,T,A,C,K,R,N,Q)
                        D(N)x{3}E(R)x(R)(G)(I)(T)(I)x{2}(A)
            64: YAWVL   DNLKAERERGITIDIA   LWKFE

Find reference above under sequence: input_54.rsf{GIARDIA_L}, pattern: Efactor_G
tp.

_____________________________________________________________________________

Fggy_Kinases_1        (M,F,Y,G,S)x(P,S,T)x2K(L,I,V,M,F,Y,W)xW(L,I,V,M,F)x(D,E,N,
Q,T,K,R)(E,N,Q,H)
                                              (G)x(T)x{2}K(Y)xW(V)x(D)(N)
            53: EAAEL                                GKTSFKYAWVLDN
                  LKAER

************************************************
* FGGY family of carbohydrate kinases signatures *
************************************************

It has been shown [1] that four different type of carbohydrate kinases seem to
be evolutionary related. These enzymes are:

 - L-fucolokinase (EC 2.7.1.51) (gene fucK).
 - Gluconokinase (EC 2.7.1.12) (gene gntK).
 - Glycerokinase (EC 2.7.1.30) (gene glpK).
 - Xylulokinase (EC 2.7.1.17) (gene xylB).
 - L-xylulose kinase (EC 2.7.1.53) (gene lyxK).

These enzymes are proteins of from 480 to 520 amino acid residues.

As consensus  patterns for this  family  of  kinases we selected two conserved
regions, one in the central section, the other in the C-terminal section.

-Consensus pattern: [MFYGS]-x-[PST]-x(2)-K-[LIVMFYW]-x-W-[LIVMF]-x-[DENQTKR]-
                    [ENQH]
-Sequences known to belong to this class detected by the pattern: ALL,  except
 for lyxK.
-Other sequence(s) detected in SWISS-PROT: 5.

-Consensus pattern: [GSA]-x-[LIVMFYW]-x-G-[LIVM]-x(7,8)-[HDENQ]-[LIVMF]-x(2)-
                    [AS]-[STAIVM]-[LIVMFY]-[DEQ]
-Sequences known to belong to this class detected by the pattern: ALL.
-Other sequence(s) detected in SWISS-PROT: 11.

-Expert(s) to contact by email:
           Reizer J.; jreizer@ucsd.edu

-Last update: November 1997 / Patterns and text revised.

[ 1] Reizer A., Deutscher J., Saier M.H. Jr., Reizer J.
     Mol. Microbiol. 5:1081-1089(1991).
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

////////////////////////////////////////////////////////////////////////////
```

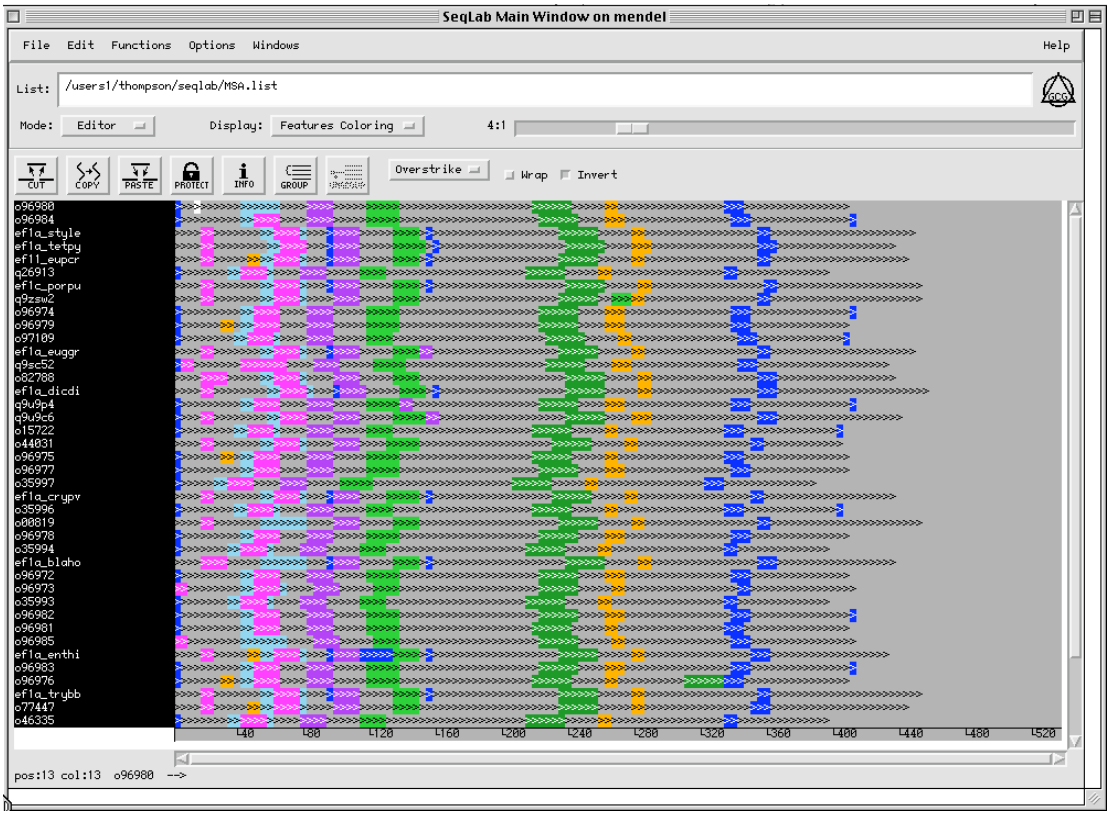
Extensive abstract and reference lists follow the identified sequence locations for each site. This information can save anybody a tremendous amount of work! The sites themselves are shown with their sequence locations below each consensus pattern. Among the other motifs discovered, the characteristic P-Loop is defined as (A,G)x4GK(S,T), i.e. either an Alanine or a Glycine, followed by four of anything, followed by an invariant Glycine-Lysine pair, followed by either a Serine or a Threonine. Exceptions are noted in the documentation. This particular site has been very well researched and many three-dimensional structures are available for it. It always has a beta/alpha/beta secondary structure conformation

and is sometimes known as the "Rossman Fold." Here the site is shown in the Guanine Nucleotide-Binding Protein G(I), Alpha-1 Subunit (Adenylate Cyclase-Inhibiting) from *Rattus norvegicus* (common rat), GBI1_RAT, courtesy ExPASy's Swiss-3DImage collection (ftp://ca.expasy.org/databases/swiss-3dimage/IMAGES/JPEG/S3D00521.jpg):



Post-translational modification sites commonly found in many proteins, such as glycosylation, phosphorylation, amidation, and myristylation, will only be listed if you specify the -Frequent option. However, realize that sites may be false positives, especially if you use the -Frequent option. This is always a danger with simple consensus style searches. The GCG programs ProfileScan and HmmerPfam use a much more sensitive profile matrix approach to search your sequence with profiles including most of PROSITE and will be discussed further later on. Notice in the example above that Motifs discovered the truly positive GTP-binding elongation factor signature and the ATP/GTP-binding P-loop site, yet it also found two probable false positives, the Prokaryotic membrane lipoprotein lipid attachment site and the FGGY family of carbohydrate kinases signature.

"**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 created with the -RSF option. The location of the PROSITE signatures will now be included in the Editor sequence display. Again use the "**SeqLab Output Manager**" to do this, as discussed previously. Select the file "**motifs.rsf**," then press the "**Add to Editor**" button and specify "**Overwrite old with new**" to take the new motifs.rsf feature file and merge it with the old RSF file in the open Editor. "**Close**" the "**Output Manager**" after loading your new RSF file. Look at your display using "**Features Coloring**" or "**Graphic Features**" to display the new annotation and see if you can recognize the differences. My dataset is illustrated below using "Features Coloring" now annotated with its original database features as well as MEME discoveries and Motifs patterns:



2.10. Performing the Alignment — the PileUp Program.

Next, I want to align all of these protein sequences. Select all of the entries in the Editor window using one of the methods discussed previously. Once all of your sequences are selected, go to the "**Functions**" menu and select "**Multiple comparison**." Click on "**PileUp. . .**" to align the entries. A new window will appear with the parameters for running PileUp. Often you'll accept all of the program defaults on a first run by pressing the "Run" button; however, here I am going to change the scoring matrix for the alignment from the default BLOSUM62 to the alternate BLOSUM30 matrix.

Depending on the level of divergence in a data set, better multiple sequence alignments can often be generated with alternate scoring matrices (the -Matrix option, specifying the desired matrix from the GCG logical directory GenMoreData) and/or different gap penalties. Beginning with GCG version 9.0, the BLOSUM62 (Henikoff and Henikoff, 1992) matrix file, "blosum62.cmp," is used as the default symbol comparison table in most programs. Furthermore, appropriate gap creation and extension penalties are now coded directly into the matrix, though they can still be adjusted within the program if desired. This is a greatly improved situation over the normalized Dayhoff PAM 250 table (Schwartz and Dayhoff, 1979) and the program encoded penalty values that GCG formerly used. The BLOSUM series are much more robust at handling a wider range of sequence divergence than the PAM table ever was — the BLOSUM30 table being most appropriate for the most divergent datasets, ranging to the BLOSUM100 table for the most conserved datasets. Since these sequences are from quite a wide spectrum of organisms, we'll use the BLOSUM30 matrix.

Therefore, click on the "**Options**" button. To specify the BLOSUM30 matrix select the check button next to and click on the "**Scoring Matrix. . .**" box in the "**Pileup Options**" window. This will launch a "**Chooser for Scoring Matrix**" window from which you can select the BLOSUM30 matrix file, "**blosum30.cmp**." Double-click the matrix's name to see what it looks like; click "**OK**" to close both windows. Scroll through the rest of "**PileUp Options**" window to see all those available. "**Close**" it when finished. Be sure that the "**How:**" box says "**Background Job**" and press then "**Run**" in the "**PileUp**" window to launch the program.

The program will first compare every sequence with every other one. This is the pairwise nature of the program, and then it will progressively merge them into an alignment in the order of determined similarity, from most to least similar (Feng and Doolittle, 1987). The window will go away and then, after a few moments, depending on the complexity of the alignment and the load on the server, new output windows will automatically display. The top window will be the Multiple Sequence Format (MSF) output from your PileUp run. Notice the BLOSUM30 matrix specification and the default gap introduction and extension penalties associated with that matrix, 15 and 5 respectively. As mentioned above, in most cases the default gap penalties will work fine with their respective matrixes, though they can be changed if desired. In fact, see below on improving regions within alignments, where it is absolutely required.

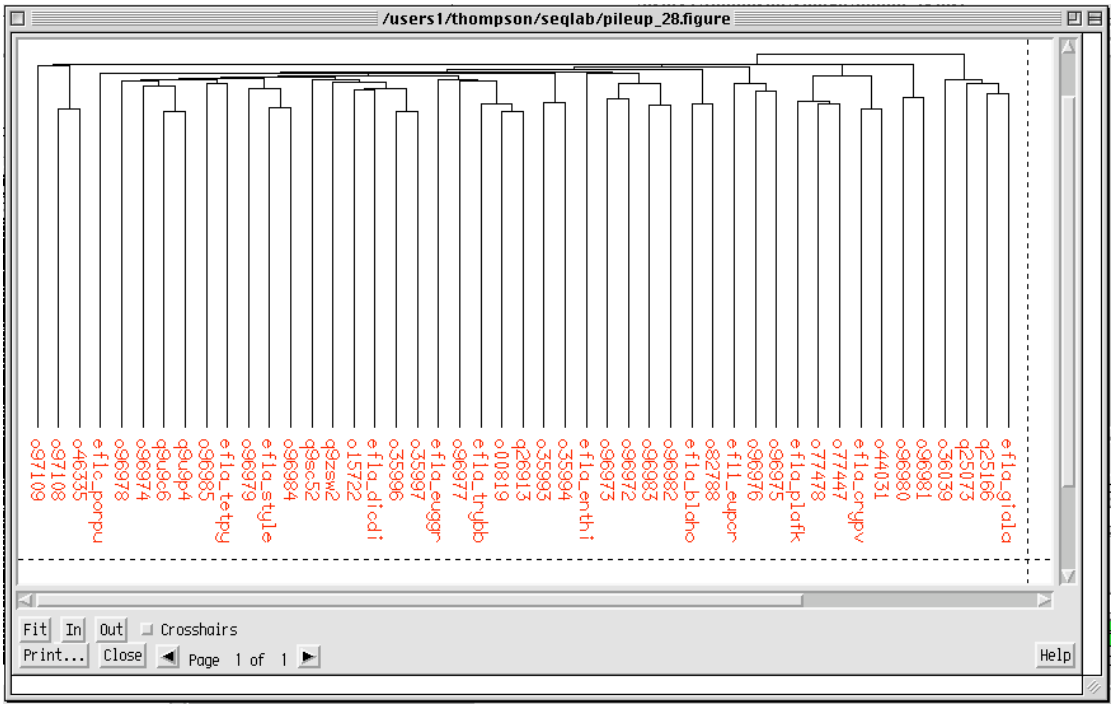
Scroll through your alignment to check it out and then "Close" the window afterwards. My abridged output file example follows below. Notice the interleaved character of the sequences, yet they all have unique identities, addressable through their MSF filename together with their own name in braces, {name}:

```
!!AA_MULTIPLE_ALIGNMENT 1.0
PileUp of: @/users1/thompson/.seqlab-mendel/pileup_28.list

 Symbol comparison table: /usr/gcg/gcgcore/data/moredata/blosum30.cmp  CompCheck
: 8599

                GapWeight: 15
           GapLengthWeight: 5

 pileup_28.msf  MSF: 472  Type: P  May 14, 2001 14:35  Check: 2476 ..

 Name: ef1a_giala       Len:   472  Check: 8631  Weight:  1.00
 Name: q25166           Len:   472  Check: 6209  Weight:  1.00
 Name: q25073           Len:   472  Check: 2914  Weight:  1.00
 Name: o36039           Len:   472  Check: 7560  Weight:  1.00
 Name: o96981           Len:   472  Check: 3858  Weight:  1.00
 Name: o96980           Len:   472  Check: 3082  Weight:  1.00
 Name: o44031           Len:   472  Check:  851  Weight:  1.00
 Name: ef1a_crypv       Len:   472  Check: 2406  Weight:  1.00
 Name: o77447           Len:   472  Check: 9210  Weight:  1.00
 Name: o77478           Len:   472  Check: 1123  Weight:  1.00
 Name: ef1a_plafk       Len:   472  Check: 1436  Weight:  1.00
//////////////////////////////////////////////////////////////
 Name: o96978           Len:   472  Check: 6796  Weight:  1.00
 Name: ef1c_porpu       Len:   472  Check: 6199  Weight:  1.00
 Name: o46335           Len:   472  Check: 7668  Weight:  1.00
 Name: o97108           Len:   472  Check: 5669  Weight:  1.00
 Name: o97109           Len:   472  Check: 6457  Weight:  1.00

//

            1                                                    50
ef1a_giala  ~~~~~~~~~~ ~~~~~~~~~~ ~~~STLTGHL IYKCGGIDQR TIDEYEKRAT
    q25166  ~~~~~~~~~~ ~~~~~~~~~~ NGKSTLTGHL IYKCGGIDQR TLDEYEKRAN
    q25073  ~~~~~~~~~~ ~~~~~~~~~~ NGKSTLTGHL IYKCGGIDQR TLEDYEKKAN
    o36039  ~~~~~~~~~~ ~~~~~~~~~~ NGKSTLTGHL IFKCGGIDQR TLDEYEKKAN
    o96981  ~~~~~~~~~~ ~~~~~~~~VD SGKSTSTGHL IYKCGGIDER TIEKFEKEAK
    o96980  ~~~~~~~~~~ ~~~~~~~~VD SGKSTSTGHL IYKCGGIDER TIEKFEKEAK
    o44031  ~~~MGKEKTH INLVVIGHVD SGKSTTTGHL IYKLGGIDKR TIEKFEKESS
ef1a_crypv  ~~~MGKEKTH INLVVIGHVD SGKSTTTGHL IYKLGGIDKR TIEKFEKESS
    o77447  ~~~MGKEKTH INLVVIGHVD SGKSTTTGHI IYKLGGIDRR TIEKFEKESA
    o77478  ~~~MGKEKTH INLVVIGHVD SGKSTTTGHI IYKLGGIDRR TIEKFEKESA
ef1a_plafk  ~~~MGKEKTH INLVVIGHVD SGKSTTTGHI IYKLGGIDRR TIEKFEKESA
    o96975  ~~~~~~~~~~ ~~~~~~~~VD SGKSTTTGHL IYKLGGTDAR TIEKFEKESA
    o96976  ~~~~~~~~~~ ~~~~~~~~VD SGKSTTTGHL IYKCGGIDAR TIEKFEKESA
ef11_eupcr  ~~~MGKEKEH LNLVVIGHVD SGKSTTTGHL IYKLGGIDAR TIEKFEKESA
    o82788  ~~~MGKEKPH INLVVIGHVD SGKSTTTGHL IYACGGIDKR TIERFEEGGQ
ef1a_blaho  ~~~MGKEKPH INLVVIGHVV AGKSTTTGHL IYACGGIDKR TIERFEEGGQ
    o96982  ~~~~~~~~~~ ~~~~~~~~VD SGKSTTIGHL IYKCGGIDKR TIDKFDKDAS
    o96983  ~~~~~~~~~~ ~~~~~~~~VD SGKSTSTGHL IYKCGGIDKR TIEKFEKEAS
    o96972  ~~~~~~~~~~ ~~~~~~~~VD SGKSTSCGHL IYKCGGIDKR TIEKYEKEAK
    o96973  ~~~~~~~~~~ ~~~~~~GHVD SGKSTSCGHL IYKCGGIDKR TIEKYEKEAN
ef1a_enthi  ~~~MPKEKTH INIVVIGHVD SGKSTTTGHL IYKCGGIDQR TIEKFEKESA
    o35994  ~~~~~~~~~~ ~~~~~~~~~~ ~~~STTTRHL IYKCGGIDQR TLDRFQKESE
    o35993  ~~~~~~~~~~ ~~~~~~~~~~ ~~~STTTGHL IYKCGGIDER TIKKFEQESE
    q26913  ~~~~~~~~~~ ~~~~~~~~~~ ~~~STATGHL IYKCGGIDKR TIEKFEKEAA
```

```
    o00819  ~~~MGKEKVH MNLVVVGHVD AGKSTATGHL IYKCGGIDKR TIEKFEKEAA
ef1a_trybb  ~~~MGKEKVH MNLVVVGHVD AGKSTATGHL IYKCGGIDKR TIEKFEKEAA
    o96977  ~~~~~~~~~~ ~~~~~~~~VD SGKSTSTGHL IYKCGGIDKR TIEKFDKEAA
ef1a_euggr  ~~~MGKEKVH ISLVVIGHVD SGKSTTTGHL IYKCGGIDKR TIEKFEKEAS
    o35997  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~CGGIDKR TIEKFEKEAK
    o35996  ~~~~~~~~~~ ~~~~~~~~~~ AGKSTTTGHL IYKCGGIDKR TIEKFEKEAK
ef1a_dicdi  MEFPESEKTH INIVVIGHVD AGKSTTTGHL IYKCGGIDKR VIEKYEKEAS
    o15722  ~~~~~~~~~~ ~~~~~~~~~~ AGKSTTTGHL IYKCGGIDKR TIEKFEKEAA
    q9zsw2  ~~~MGKQKTH INIVVIGHVD SGKSTTTGHL IYKCGGIDKR TIEKFEKEAA
    q9sc52  ~~~~~~~~~~ ~~~~VIGHVD AGKSTTTGHL IYKCGGIDKR TIEKFEKEAA
    o96984  ~~~~~~~~~~ ~~~~~~~~VD SGKSTSTGHL IYKCGGIDKR TIEKFEKEPA
ef1a_style  ~~~MPKEKNH LNLVVIGHVD SGKSTSTGHL IYKCGGIDKR TIEKFEKEAA
    o96979  ~~~~~~~~~~ ~~~~~~~~VD SGKSTTTGHL IYKCGGIDKR VIEKFEKESA
ef1a_tetpy  ~~MARGDKVH INLVVIGHVD SGKSTTTGHL IYKCGGIDKR VIEKFEKESA
    o96985  ~~~~~~~~~~ ~~~~~~GHVD SGKSTSTGHL IYKCGGIDKR TLEKFEKEAA
    q9u9p4  ~~~~~~~~~~ ~~~~~~~~VD SGKSTTTGHL IYKLGGIDER TIKKFEDEAN
    q9u9c6  ~~GTRKDKLH VNLVVIGHVD SGKSTTTGHL IYKLGGIDER TIKKFEDEAN
    o96974  ~~~~~~~~~~ ~~~~~~~~VD SGKSTSTGHL IYKCGGIHKR TIEKFEKEAN
    o96978  ~~~~~~~~~~ ~~~~~~~~VD SGKSTTTGHL IYKCGGIDKR TIEKFEKESA
ef1c_porpu  ~~~MGKEKQH VSIVVIGHVD SGKSTTTGHL IYKCGGIDKR AIEKFEKEAA
    o46335  ~~~~~~~~~~ ~~~~~~~~~~ ~~~STTTGHL IYKCGGLDKR KLAAMEKEAE
    o97108  ~~~~~~~~~~ ~~~~~~~~VD AGKSTTTGHL IYKCGGLDKR KLAAIEKEAE
    o97109  ~~~~~~~~~~ ~~~~~~~~~~ AGKSTTTGHL IYKCGGIDKR VIEKFEKEAA

            51                                                  100
ef1a_giala  EMGKGSFKYA WVLDQLKDER ERGITINIAL WKFETKKYIV TIIDAPGHRD
    q25166  EMGKGSFKYA WVLDQLKDER ERGITINIAL WKFETKKFTV TIIDAPGHRD
    q25073  EIGKGSFKYA WVLDQLKDER ERGITINIAL WKFETKKFIV TIIDAPGHRD
    o36039  ELGKGSFKYA WVLDQLKDER ERGITINIAL WKFETKKFIV TIIDAPGHRD
    o96981  QIGKESFKYA WVLDKLKAER ERGITIDIAL WKFESQKYSF TIIDAPGHRD
    o96980  QIGKESFKYA GLLDILKAER ARGITIDIAL WKFESQKYSF TIIDAPGHRD
    o44031  EMGKGSFKYA WVLDKLKAER ERGITIDIAL WQFETPKYHY TVIDAPGHRD
ef1a_crypv  EMGKGSFKYA WVLDKLKAER ERGITIDIAL WQFETPKYHY TVIDAPGHRD
    o77447  EMGKGSFKYA WVLDKLKAER ERGITIDIAL WKFETPRYFF TVIDAPGHKD
    o77478  EMGKGSFKYA WVLDKLKAER ERGITIDIAL WKFETPRYFF TVIDAPGHKH
ef1a_plafk  EMGKGSFKYA WVLDKLKAER ERGITIDIAL WKFETPRYFF TVIDAPGHKD
    o96975  EMGKGTFKYA WVLDKLKAER ERGITIDIAL WKFETTNRFY TIIDAPGHRD
    o96976  EMGKGSFKYA FVLDNLKAER ERGITIDIAL WKFETPKRFY TIIDAPGHRD
ef11_eupcr  EMGKASFKYA WVLDKLKAER ERGITIDIAL WKFETENRHY TIIDAPGHRD
    o82788  RIGKGSFKYA WVLDKMKAER ERGITIDISL WKFQTEKYFF TIIDAPGHRD
ef1a_blaho  RIGKGSFKYA WVLAKMKAER ERGITIDISL WKFETRKDFF TIIDAPGHRD
//////////////////////////////////////////////////////////////////
```

Return to the listing of sequence names near the top of the file. This listing contains an important number called the checksum. All GCG sequence programs use this number as a unique sequence identifier. There is a checksum line for the whole alignment as well as individual checksum lines for each member of the alignment. If any two of the checksum numbers are the same, then those sequences are identical. If they are, an editor can be used to place an exclamation point, "!" at the start of the checksum line in which the duplicate sequence occurs. Exclamation points are interpreted by GCG as remark delineators, therefore, the duplicate sequence will be ignored in subsequent programs. Or the sequence could be "CUT" from the alignment with the SeqLab Editor. Another important number on the individual checksum lines also needs to be pointed out. The "Weight" designation determines how much importance each sequence contributes to a profile made of the alignment. Sometimes it is worthwhile to adjust these values so that the contribution of a collection of very similar sequences does not overwhelm the signal from a few more divergent sequences. In the SeqLab interface the "Sequence Info . . ." window can be used to accomplish this, or you can use a simple text editor. However, I will not be bothering with it here.

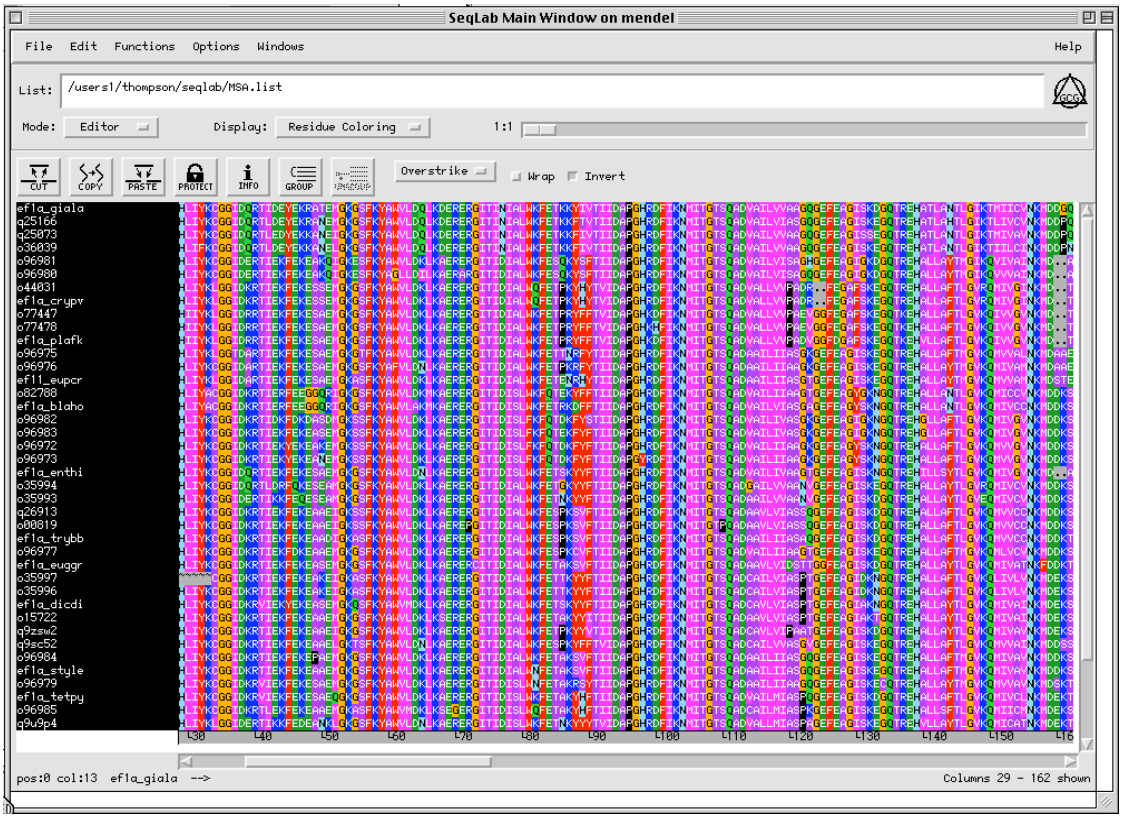
Scroll through the alignment and then "**Close**" its window. Again use the "**Output Manager**" to "**Add to Editor**" and "**Overwrite old with new**," to take your new MSF output and merge it with the old RSF file in the open Editor. This will keep all of the database feature annotation intact, yet renumber all of its reference locations based on the inclusion of gaps in the alignment. "**Close**" the "**Output Manager**" after loading your new alignment. The next window will contain PileUp's cluster dendrogram, in the EF-1 example, the following graphic:



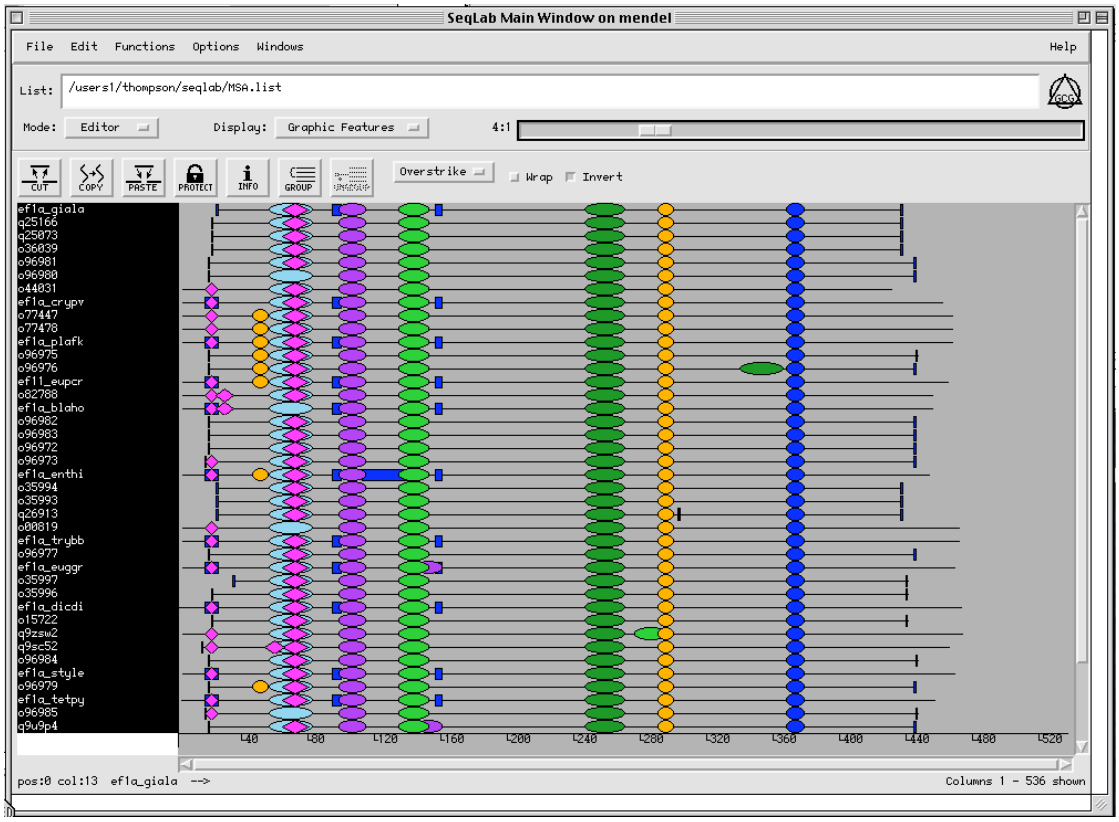
PileUp automatically creates this dendrogram of the similarity clustering relationships between the sequences. It can be very helpful for adjusting sequence Weight values, which even out each sequences' contribution to a profile. The lengths of the vertical lines are proportional to the differences in similarity between the sequences. However, realize that this tree is not an evolutionary tree, and it should never be presented as one. No phylogenetic inference optimality criteria algorithm, such as maximum likelihood, least-squares fit, or parsimony, nor any molecular substitution, multiple-hit correction models, such as Jukes-Cantor, Kimura, or any other subset of the GTR (General Time Reversible) model, nor any site rate heterogeneity models such as a Gamma correction, are used in its construction. (It is roughly an uncorrected UPGMA tree, prone to all the same errors seen in UPGMA. Therefore, if the rates of evolution for each lineage were exactly the same, then it could represent a 'true' phylogenetic tree, but this is seldom the case in nature.) PileUp's dendrogram merely indicates the relative similarity of the sequences based on the scoring matrix used, by default the BLOSUM62 but the BLOSUM30 in my example, and, therefore, the clustering order used to create the alignment.

If desired, you can directly print from SeqLab graphics Figure windows to PostScript files by picking "Print . . ." "[Encapsulated] PostScript File" "Output Device:" You can name the output file to anything that you want; click "Proceed" to create an EPSF output in your current directory. To actually print this file you may need to ftp it to a local machine attached to a PostScript savvy printer unless you have direct access to the UNIX system printer and it is PostScript compatible. (All Macintosh compatible laser printers run PostScript by default. Carefully check any laser printer connected to a 'Wintel' system to be sure that it is PostScript compatible.) "**Close**" the dendrogram window.

Now notice that your residues align by color. My Editor display looks like the following after loading the MSF file using "**Residue Coloring**" and a "**1:1**" zoom ratio:



Notice the nice columns of color representing columns of aligned residues. Change the "**Display:**" box from "**Residue Coloring**" to "**Graphic Features**." Now the display shows a schematic of the feature information from each entry, as well as all of the motifs discovered by the programs Motifs and MotifSearch, and will look like the following, at a "**4:1**" zoom:

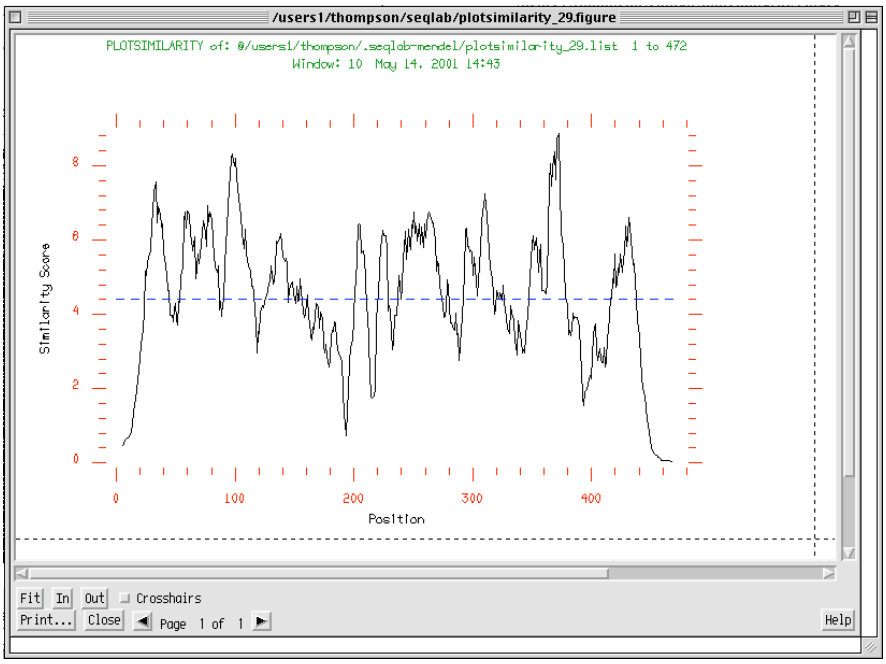


Remember, quickly double clicking on any of the color coded feature regions in the Editor display will produce a "Features" window where more information is available about that particular feature by selecting the Feature entry in the new window. Clicking once in the colored region and then using the "Features" option from the "Windows" menu will also produce the "Features" window. Now would also be another good time to save your work as an updated RSF file!

2.11. Visualizing Conservation in Multiple Sequence Alignments.

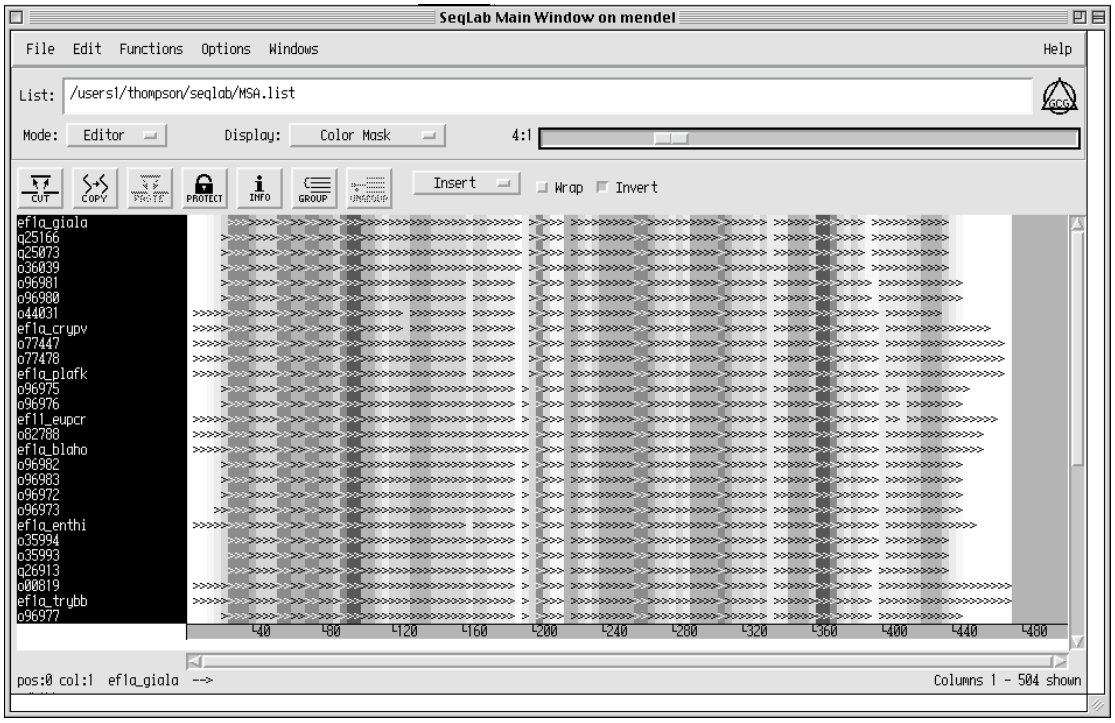
The most conserved portions of an alignment are those most resistant to evolutionary change, often due to some type of structural constraint. To easily visualize the positional conservation of a multiple sequence alignment use the graphics program PlotSimilarity. The program draws a graph of the running average similarity along a group of aligned sequences (or of a profile with the -Profile option). The PlotSimilarity peaks of a protein alignment represent the most conserved areas of the alignment, but even more so, those areas most resistant to evolutionary change due to the algorithm's use of the BLOSUM matrix in its calculations. PlotSimilarity is also a nice way to see those areas of an alignment that may need improving by pointing out the most variable regions. Furthermore, PlotSimilarity can be helpful for ascertaining alignment quality by noting changes in the overall average alignment similarity and in those regions of conservation within the alignment, as it is adjusted and refined.

Select all of the sequence names and then go to the "**Functions**" menu and under the "**Multiple comparison**" section choose "**PlotSimilarity . . ..**" I recommend changing some of the program defaults so choose "**Option**s" in the program window. Check "**Save SeqLab colormask to**" and "**Scale the plot between:**" the "**minimum and maximum values calculated from the alignment**." The first option's output file will be used in the next step. The second specification launches the program's command line -Expand option. This blows up the plot, scaling it between the maximum and minimum similarity values observed, so that the entire graph is used, rather than just the portion of the Y axis that your alignment happens to occupy. The Y-axis of the resulting plot uses the similarity values from whichever scoring matrix you used to create your alignment unless you specify an alternative. The default matrix, BLOSUM62, begins its identity value at 4 and ranges up to 11; mismatches go as low as -4. "**Close**" the "**Options**" window; notice that the "**Command Line:**" box now reflects your updated options. Click the "**Run**" box to launch the program. The output will quickly return. "**Close**" the plotsimilarity.cmask display and the "**Output Manager**" and then take a look at the similarity plot. My example follows next:



This example shows a great deal of sequence similarity. Strong peaks are seen centered around positions 30, 100, and 375. The ordinate scale is dependent on the scoring matrix used by the program, here the BLOSUM30 table, which ranges in score from -7 to +20. The dashed line across the middle shows the average similarity value for the entire alignment, here about 4.4. Make a PostScript file of this plot too, if desired. As before, to print a SeqLab graphics Figure to a PostScript file: select "Print . . ." off the Figure window, choose "Output Device:" "[Encapsulated] PostScript File," and click "Proceed," to create EPSF output. Regardless of whether you print this plot or not, take notes of where the similarity significantly falls off within and at the beginning and end of the alignment. In my example above, this is the first 25 residues or so, a region around 190 and 220, around 390, and about the last 25 residues. "**Close**" the PlotSimilarity window after noting where these deepest valleys, the least similar regions of the alignment, lay.

Now go to the "**File**" menu and click on "**Open Color Mask Files**." This will produce another window from which you should select your new "**plotsimilarity.cmask**" file; click on "**Add**" and "**Close**" the window. This will produce a gray scale overlay on your sequences that describes their regional similarity where darker gray corresponds to higher similarity values. My sample alignment, using a zoom factor of 4 to 1, looks like the following. Notice the strong conservation peak centered just before residue 100 in the alignment, one of EF-1α's GTP binding regions:



2.12. Improving Alignments within SeqLab.

The beauty of this representation is you can now easily select those regions of low similarity to try to improve their alignment automatically. This is possible because of PileUp's incredibly effective -InSitu option that can realign regions within an alignment. Be sure that all of your sequences are selected and then zoom back in your alignment to "**1:1**" so that you can see individual residues and then scroll to the carboxy end. It's best to start at the carboxy termini in this process so that the positions of the low similarity regions do not become skewed as you proceed through the procedure. Now select a region of low similarity across the complete sequence set. This can be done using the mouse if it's all on the screen in front of you, which is not the case here. Otherwise, use the "**Edit**" "**Select Range**" function (determine the positions by placing your cursor at the beginning and end of the range to be selected and noting the column number in the lower left-hand of the Editor display). Once all of your sequences and the region that you wish to improve are

selected, go to the "**Functions**" menu and again select "**Multiple comparison**." Click on "**PileUp . . .**" to realign all of the sequences within that region. (The "Windows" menu also contains a 'shortcut' listing of all of the programs that you have used in the current session; you can launch any of them from there as well.) You will be asked whether you want to use the "Selected sequences" or "Selected region;" it is very important to specify "**Selected region**." This will produce a new window with the parameters for running PileUp. Next, be sure to click on "**Options . . .**" to change the way that PileUp will perform the alignment. In the "**Options**" window check the gap creation and extension boxes and change their respective values to much less than the default. Changing them to about a third the default value works pretty well for a start, so for the BLOSUM30 matrix change the values to "**5**" and "**2**" respectively. Most importantly, check "**Realign a portion of an existing alignment**;" this calls up the command line -InSitu option. Otherwise only that portion of your alignment selected will be retained in the output. Furthermore, we really don't need another similarity dendrogram, so uncheck the "**Plot dendrogram**" box. "**Close**" the window and notice the new options in the PileUp "Command Line:" "**Run**" the program to improve your alignment. The window will go away and your results will return very quickly since you are only realigning a portion of the alignment; new output windows will automatically display. The top window will be the MSF output from your PileUp run. Notice the BLOSUM30 matrix specified (others available through the options menu) and the lowered gap introduction and extension penalties of 5 and 2. Scroll through your alignment to check it out and then "**Close**" the window. The next window will be the "Output Manager." Just like before, click on "**Add to Editor**" and then specify "**Overwrite old with new**" in the new "**Reloading Same Sequences**" window to merge the new alignment with the old one and retain all feature annotation. This feature information may help guide your alignment efforts in subsequent steps. "**Close**" the "**Output Manager**" window after loading your new alignment.

Your alignment should now be a bit better within the specified region. Repeat this process in all areas of low similarity, again, working from the carboxy termini toward the amino end. Notice that all of the options that you last specified are retained by the program so you don't need to respecify them. You can also save these run parameters so that they will come up in subsequent sessions by clicking on the "Save Settings" box in any of the program run windows. You may want to go to the "**File**" menu periodically to save your work using the "**Save as . . .**" function in case of a computer or network problem. It's also probably a good idea to reperform the PlotSimilarity and color mask procedure after going through the entire alignment to see how things have improved after you've finished the various InSitu PileUps. If you discover an area that you can not improve through this automated procedure, then it is time to either manually 'correct' it or 'throw it away.' Again, note those 'problem' areas and then switch back to "**Residue Coloring**." This will ease manual alignment by allowing your eyes to work with columns of color.

Other things that can help manual alignment are "GROUP"ing and "Protections." The "GROUP" function allows you to manipulate 'families' of sequences as a whole — any change in one will be propagated throughout them all. To "GROUP" sequences, select those that you want to behave collectively and then click on the "GROUP" icon right above your alignment. You can have as many groups as you want. The space bar will introduce a gap into the sequence and the delete key will take a gap away. However, you can not delete a sequence residue without changing that sequence's (or the entire alignment's) "Protections." Click on the padlock icon to produce a "Protections" window. Notice that the default protection allows you to modify "Gap Characters" and "Reversals" only. Check "All other characters" to allow you to "CUT" regions out of your alignment and/or delete individual residues and then click "OK" to close the window. A very powerful manual alignment function can be thought of as the 'abacus' function. To take advantage of this function select the region that you want to slide and then press the shift key as you move the region with the right or left arrow key. You can slide residues greater distances by prefacing the command keystrokes with the number of spaces that you want them to slide.

Make subjective decisions regarding your alignment. Is it good enough; do things line up the way that they should? If, after all else, you decide that you just can't align some region, or even an entire sequence, then perhaps get rid of it with

the "CUT" function. Another alternative is the mask function that I will describe below. Cutting out an entire sequence may leave some columns of gaps in your alignment. If this is the case, then reselect all of your sequences and go to the "Edit" menu and select "Remove Gaps . . ." "Columns of gaps." Notice the extreme amino and carboxy ends of the alignment. Amino and carboxy termini seldom align properly and are often jagged and uncertain. This is fairly common in multiple sequence alignments and subsequent analyses should probably not include these regions. If loading sequences from a database search, allowing SeqLab to trim the ends automatically based on beginning and ending constraints considerably improves this situation. Overall, things to look for include columns of strongly conserved residues such as tryptophans, cysteines, and histidines, important structural amino acids such as prolines, tyrosines and phenylanines, and conserved isoleucine, leucine, valine substitutions; make sure they all align. After you have finished tweaking, evaluating, and readjusting your alignment to make it as 'satisfying' as possible, change back to "**Feature Coloring**" "**Display**." Those features that are annotated should now align perfectly. This is another way to assure that your alignment is as biologically 'correct' as possible. Everything you do from this point on, and especially later if you use alignments to ascertain molecular phylogenies, is absolutely dependent on the quality of the alignment! You need a very clean, unambiguous alignment that you can have a very high confidence in — truly a biologically meaningful alignment. Each column of symbols must actually contain homologous characters.

Many other alignment editors are available for cleaning up multiple sequence alignments. However, I think that you will find SeqLab most satisfying, and only using a GCG compatible editor assures that the format will not be corrupted. If you do make any changes to a GCG sequence data file with a non-GCG compatible editor, you must reformat the alignment afterwards. However, reformatting GCG MSF or RSF files requires a couple of tricks. If this step is not done exactly correct, you will get very weird results. If you do need to do this for any reason, you must use the appropriate Reformat option (either -MSF or –RSF respectively) and you must specify all the sequences within the file using the brace specifier, i.e. "{*}," for example:

```
>  reformat -msf your_favorite.msf{*}
```

You should never need to do this, unless for some perverse reason you decide to edit an alignment with a non-GCG compliant editor; however, it may prove necessary in some situations. After reformatting, the new MSF or RSF file will follow GCG convention, with updated format, numbering, and checksums.

2.13. SeqLab Editor On-Screen Annotation.

Something that you may want to do to your alignment after you've gotten it all cleaned up is add text annotation to the display. Changing the entries' names for presentation purpose might also be helpful. Both are easy to do in the SeqLab Editor. Double-click on an entry's name to get its "Sequence Information" window and directly edit the name there. Selecting the entry name and then pressing the "INFO" icon does the same thing. To put text lines directly into your display go to the SeqLab "**File**" menu "**New sequence . . .**" entry and select the "**Text**" button to the "**What type of sequence?**" question. This will put a "**NewText**" line at the bottom of the Editor display that you can directly type annotation into. You can also add customized "Graphic Features" and "Features Coloring" annotation with the "Windows" "Features" window. Select a desired region across an alignment and launch the "**Features**" window. Press "**Add**" to get a "**Feature Editor**" window where you can designate the feature's "**Shape:**" "**Color:**" and "**Fill:**" as well as give the region a "**Keyword:**" and "**Comments:**." Warning: You can add feature annotation to a region across an entire alignment, but you can not delete or edit the annotation from the whole region collectively afterwards. You can only edit or delete feature annotation from an RSF file with the SeqLab Editor one sequence feature at a time!

Subsequent screen snapshots of my example dataset will reflect changed entry names and on-screen annotation, as described above. I'll also pare down my dataset to 38 sequences by excluding the farthest outliers least similar to *Giardia* EF-1 and by removing redundancies where two sequences were almost identical.

2.14. Profile Analysis — Position Specific, Weighted Score Matrices of Multiple Sequence Alignments.

OK, so one-dimensional motifs are a way to 'capture' the information of an important portion of an alignment. However, motifs can not convey any degree of residue 'importance.' For instance, in the GTP-binding P-Loop seen in previous sections, is it better to have an Alanine or a Glycine in that first position or doesn't it matter? This lack of sense of importance causes a loss of sensitivity. More 'robust' methods can convey the importance of each residue in the region.

Given a multiple sequence alignment, how can we use the extra information contained in it to find ever more remotely similar sequences? How do we search and explore into and past Russell Doolittle's "Twilight Zone," i.e. those similarities below ~25% identity, those Z scores below ~4, those BLAST/Fast *E* values above ~10^{-3} or so? Just because a similarity score between two sequences is quite low, we do not automatically know that the two structures do not fold in a similar manner or perform a similar function, we have no idea of homology at all!

Obviously much of the information in a multiple sequence alignment is 'noise' at this similarity level. Searching with the full-length of any of its members would not gain us anything. Too much evolution has happened over its full length — the 'history' of most of it has been lost. However, certain regions of the alignment have been constrained throughout evolutionary history. They are somehow very 'important' to the sequence — functionally, structurally, or whatever — we can use them to find other sequences with similarly constrained regions.

Enter two-dimensional consensus techniques. The basic idea is to tabulate how often every possible residue occurs at each position. This information is stored in a matrix twenty residues wide by the length of your pattern. Does this remind you of anything? We're talking about the same concept as a symbol substitution table or scoring matrix, in other words a very special PAM style table — a matrix custom built based on a specific pattern in a collection of related sequences.

This powerful approach is called Profile analysis (Gribskov, et al., 1987 and 1989). It, and later refinements thereof (e.g. Eddy, 1996 and 1998) is great for discovering distantly related proteins and structural motifs. John Devereux, past president of GCG, wrote an excellent overview essay of the method in the GCG Program Manual. It's worth the time to read this section at some point ("genmanual" from the command line or the "Help" buttons in SeqLab). This strategy is used after you've prepared and refined as much as possible (and saved!) your multiple sequence alignment of significantly similar sequences or regions within sequences. A good plan is to find similar sequences to a newly sequenced section of DNA using traditional database searching techniques and then align all of the significantly similar translationed sequences or domains. Next, run the aligned sequences through the Profile package to generate a profile of the family — a very sensitive and tremendously powerful probe for further searching analyses.

Profile methods enable the researcher to recognize features that may otherwise be invisible to individual sequence members. Profile analysis uses the full information content of an alignment. The greatly enhanced information content, over that of individual sequences, has the potential to find similar motifs in sequences that are only distantly related, more so than any other class of search algorithm. All other methods of describing an alignment such as consensus or pattern description either through away too much information or become too ambiguous. Profiles achieve additional sensitivity with a two-dimensional weight matrix approach versus a simple one-dimensional string technique. Furthermore, profiles are a special type of two-dimensional weight matrix in which conserved areas of the alignment receive the most importance and variable regions hardly matter!

A distinct advantage is further manipulations and database searches consider evolutionary issues by virtue of the Profile algorithms. The creation of gaps is highly discouraged in conserved areas and occurs easily in variable regions in subsequent profile alignments and searches. This occurs because gaps are penalized more heavily in conserved areas than they are in variable regions. Furthermore, the more highly conserved a residue is, the greater its position-specific matrix score is. These two factors are what give profiles so much power. The matrix and its associated consensus sequence are not based merely on the positional frequency of particular residues, but rather utilize the evolutionary conservation of amino acid substitutions within the alignment based on the scoring matrix specified, by default the BLOSUM62 table (Henikoff and Henikoff, 1992) (other substitution matrices can also be specified). Therefore, the resultant consensus residues are the most evolutionarily conserved, rather than just statistically the most frequent. This can mean much more to us than an ordinary consensus and is especially appropriate for the design of hybridization and PCR probes for unknown sequences when data is available in several related species.

2.14.1. 'Traditional' Profiles.

The Gribskov et al. (1987) method is implemented in the Wisconsin Package with a series of five programs:

ProfileMake —	creates the profile from a multiple sequence alignment.
ProfileSearch —	searches other sequences (the database) with a profile.
ProfileSegments —	aligns the output list of a ProfileSearch.
ProfileGap —	aligns individual sequences to a profile.
ProfileScan —	searches sequences against validated profile library built by Gribskov and based on PROSITE.

A profile, and its inherent consensus, is created with the GCG program ProfileMake. When you create a profile all of its members should be appropriately weighted to even out each contribution. Each sequence, by default, contributes an equal importance, i.e. weight, to the profile. This may or may not be appropriate for your situation. Consider a multiple sequence alignment with several very similar sequences and a few more divergent ones. In this case, the contribution of the more divergent sequences would be 'lost' among the overpowering signal of all the similar ones. It may be appropriate to increase the weight of the more divergent sequences to even out all the sequences' contribution. This is often done in an 'ad-hoc' manner, although a similarity dendrogram, as seen earlier, can aid the decision. Those clusters with less than their 'fair share' of contribution, have their weights increased. To figure out the appropriate weighting factors, choose the largest cluster, assign each member a weight of one and then propagate that up throughout the clusters. (If you're interested, I can explain further personally.) The process of weighting your sequences appropriately and repeatedly searching the database with your profile and then adjusting the weights and including or excluding subsequent members of the profile is known as "validating" your profile. If using Traditional Profile analysis in your own research, following the validation procedures outlined in the GCG Program Manual in the ProfileScan description is very prudent. A 'motif' style profile library based on the PROSITE Dictionary of Protein Sites and Patterns has been prepared by Gribskov and made available within the GCG system. The program ProfileScan searches your query protein sequence against this library. The present version of GCG has 632 validated profiles in its ProfileScan library.

To run ProfileMake be sure that all of your alignment sequences (except a mask, see later) are selected and then, based on your previous observations and your experimental objectives, select the longest, most conserved, overall sequence length available. Restrict the length of your profile so that jagged ends in the alignment are excluded. In SeqLab do this through the "**Edit**" "**Select Range. . .**" menu. "**Select**" and then "**Close**" the box. Another effective strategy is to develop multiple shorter profiles just centered about the similarity peaks of your alignment. These most likely will correspond to functional or structural domains in your protein. After your range is selected use the "**Functions**" "**Multiple Comparison**" "**ProfileMake**" menu and reply "**Selected region**" in the "**Which selection**" dialog box. You can also use the "**Options. . .**" menu from the "**ProfileMake**" dialog box to specify the -SeqOut command option by checking "**Write**

the consensus into a sequence file" and giving it an appropriate name. This will generate a normal sequence file of the consensus in addition to the profile file. Play with any of the other options that you would like, such as the scoring matrix, and then "**Close**" the "**Options**" box and "**Run**" ProfileMake. After running ProfileMake, the top window returned will display your profile consensus sequence. All positions will be filled; there will be no gaps. This is because the Profile algorithm will decide on the most conserved residue for each position, regardless. The header contains information relating to the sequence's creation through ProfileMake. "**Close**" the consensus window. The "Output Manager" will also list a "**.prf**" file. This is the profile itself.

I created a small profile of just the P-Loop region to show you how to interpret a profile matrix. The greatest amount of conservation of the P-Loop region is centered about absolute residue position twenty or so. What happens if I prepare a profile around just this region? What does it look like? It's a big table of numbers that doesn't make a whole lot of sense to us mere mortals, but it is a tremendously powerful tool in subsequent analyses. As described above, other programs can read and interpret this alignment customized scoring matrix to perform very sensitive database searches and further alignments by utilizing the information within the matrix that penalizes misalignments in phylogenetically conserved areas more than in variable regions.

Let's check it out next:

Cons	A	B	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y	Z	Gap	Len..
E	11	20	-11	27	33	-21	16	10	-4	10	-9	-6	16	6	18	0	8	17	-3	-29	-15	26	12	12
K	0	27	-40	21	22	-47	-6	7	-13	100	-20	13	27	7	27	53	14	13	-13	5	-40	28	12	12
! 11																								
P	13	3	4	3	3	-13	9	2	3	3	-2	-1	1	28	4	3	11	20	9	-21	-16	4	12	12
H	-7	26	-6	26	26	-6	-14	99	-18	6	-12	-19	33	13	46	33	-13	-6	-19	-7	20	33	12	12
I	3	-7	2	-7	-6	19	-6	-9	43	-7	29	22	-10	-4	-6	-10	-4	6	38	-17	1	-5	12	12
N	14	73	-19	47	33	-34	27	33	-20	27	-27	-20	100	0	26	7	22	14	-20	-20	-7	27	12	12
I	1	-10	-1	-10	-8	26	-9	-10	46	-8	34	27	-12	-6	-8	-12	-6	5	40	-12	4	-7	12	12
V	**15**	2	7	3	1	-1	**20**	-9	**24**	-6	**14**	11	-3	6	-3	-11	4	10	**37**	-30	-9	-1	12	12
V	9	-4	7	-5	-4	5	7	-8	29	-4	20	15	-6	4	-7	-9	0	19	36	-21	-2	-5	12	12
I	0	-16	16	-16	-16	55	-24	-24	118	-16	63	47	-24	-16	-24	-24	-8	16	87	-39	8	-16	12	12
G	55	47	16	55	39	-47	118	-16	-24	-8	-39	-24	31	24	16	-24	47	31	16	-79	-55	24	12	12
H	-6	27	-7	27	27	-8	-13	100	-20	7	-13	-20	34	14	48	34	-13	-7	-20	-7	19	34	12	12
! 21																								
V	11	-12	12	-12	-12	13	11	-18	67	-12	48	36	-18	5	-12	-18	-6	12	89	-47	-6	-12	12	12
D	24	87	-39	118	79	-79	55	31	-16	24	-39	-31	55	8	55	0	16	16	-16	**-87**	-39	71	12	12
S	9	12	11	11	11	-8	8	22	-7	5	-10	-10	14	11	11	9	23	4	-6	1	-2	9	12	12
G	55	47	16	55	39	-47	118	-16	-24	-8	-39	-24	31	24	16	-24	47	31	16	-79	-55	24	12	12
K	0	27	-40	20	20	-47	-7	7	-14	100	-20	13	27	7	27	55	13	13	-14	8	-40	27	12	12
S	19	14	30	10	10	-14	27	-9	-2	10	-17	-12	14	19	-5	3	63	24	-2	7	-19	1	100	100
T	40	20	20	20	20	-30	40	-10	20	20	-10	0	20	30	-10	-10	30	**150**	20	-60	-30	10	100	100
T	8	-4	-9	-4	0	13	1	-6	18	0	23	22	-2	2	-4	-9	0	34	18	-6	-2	-1	100	100
T	19	8	10	8	8	-12	19	-6	16	8	1	4	7	14	-6	-6	13	69	18	-32	-14	3	100	100
G	40	24	10	28	21	-27	61	-8	-11	-4	-19	-11	16	16	9	-14	26	18	9	-44	-28	13	100	100
! 31																								
H	10	11	-1	11	11	-10	1	34	-8	7	-8	-5	13	11	19	18	0	1	-6	-1	0	14	100	100
L	-4	-20	-27	-20	-13	50	-21	-10	43	-13	62	53	-17	-13	-7	-17	-15	-2	40	13	12	-9	100	100
*	20	0	0	27	12	3	73	70	65	46	38	0	24	11	5	6	33	85	65	0	0	0		

On closer inspection, the matrix begins to make some sense. Across the top are all possible residues. The first column is that residue that received the highest score in the program — the consensus. But notice the interior of the matrix. Numbers bounce all over the place, from 150 to -87. What's that all about? Well, without going into all the fancy mathematics involved, based on the alignment we fed it and the initial scoring matrix used (by default the BLOSUM62 matrix but you can specify others) the program has scaled those positions which are most important up and those positions least important down. For instance the Threonine at position 27 in our alignment is the only residue absolutely conserved throughout — it gets the highest score! The Aspartate at position 22 substituted with a Tryptophan would never happen, hence the -87 score. Tryptophan is the most conserved residue on both the PAM and BLOSUM matrix series and the Aspartate is conserved at all positions in our alignment that have residues at that position — the negative matrix score of any substitution to Tryptophan times the high conservation at that position for Aspartate equals the most negative score in the profile. How about those positions where the conservation is not as striking? Position 16 is a good one to pick on. Valine is the assigned consensus residue because it has the highest score, 37, but Glycine also occurs several times, a score of 20. However, other residues are ranked in the substitution matrices as being quite similar to

Valine; therefore Isoleucine and Leucine also get similar scores, 24 and 14, and Alanine occurs some of the time in the alignment so it gets a comparable score, 15. But realize that all of these numbers are way less than the highest numbers in the matrix — because the position is not well conserved all the values are fairly mediocre at that position.

OK, but what about the last two columns in the matrix, and the last row? The last row is the composition of the whole profile. Our alignment has twenty Alanines overall and no Cysteines — big deal. However, the last two columns are very important! They relate to gap penalties in any subsequent analysis with this particular profile. I stated that gaps are more easily introduced into variable regions than conserved regions in profile analysis. Well, this is where that comes from. The first column is the gap opening penalty and the second is the gap extension penalty for that particular spot in any subsequent analysis (both as a percentage). Unlike other implementations of dynamic programming, the penalties are not constant throughout the length of the profile. Those regions where conservation is highest, receive 100% of the assigned gap penalty. Those regions with less conservation, receive less gap penalty. Here, everywhere else only gets 12% of the assigned gap penalty!

"**Save As . . .**" the profile in your "**Output Manager**" giving it an appropriate name that you can recognize; retain the "**.prf**" extension. "**Close**" the "**Output Manager**."

ProfileSearch is launched through SeqLab with the "**Functions**" menu; select "**Database Sequence Searching**" "**ProfileSearch**." Specify the "**Query profile. . .**" in the "**File Chooser**" and click "**OK**." Search whichever protein database you prefer, though to reduce cpu load I suggest you just use "**NRL_3D**" for now. I like to run ProfileSegments separately after my ProfileSearch is done. Therefore, uncheck "**ProfileSegments. . .**" to prevent ProfileSearch's output from automatically being passed to ProfileSegments. 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. Also, under "**Options. . .** " I like to use the -MinList option by changing "**Lowest Z score to report in output list**" from 2.5 to 3.5 or higher. MinList sets a list Z score cut-off value — a handy way to limit your output list size. "**Close**" the "**Option**s" window and be sure that "**How:**" "**Background Job**" is selected and then click "**Run**."

As in BLAST and FastA searches, ProfileSearch estimates a realistic significance parameter. In the case of profile searching it is a Z score based on the distance, in the number of standard deviations, from the rest of the 'insignificant' database matches. ProfileSearch Z scores are normalized and reflect the significance of the results. Here rather than randomizing sequences to evaluate a Z score, as is done in Monte Carlo approaches (see previous discussion of the -Randomization option within the GCG's programs Gap and BestFit), it is calculated based on all of the nonsimilar sequences from the database search, similar to the way that FastA calculates its Expectation values. Pay particular attention to the reported Z scores in the output. As with Monte Carlo approaches, Z scores below 3 are probably not worth considering, from around 4 to 7 may be interesting, and above 7 are most probably significant and should definitely be checked out further. You can find remote similarities that all other methods will miss using Profile analysis properly — it is extremely powerful.

2.14.2. Interpreting Profile Analysis — Why Even Bother; What Can it Show Us?

Even though ProfileSearches do require some work to setup and run — a meaningful multiple sequence alignment must be assembled and refined, ProfileMake needs to be run, and the search job itself takes quite a long time to run — it is well worth the bother. ProfileSearches are incredibly CPU intensive, together with HmmerSearch some of the most so in the GCG package, so be sure to submit them as early as possible (if launched from the command line, use the -Batch option). When you return to a completed ProfileSearch take a careful look at the output. There is a good chance that other search algorithms will have missed some of the sequences listed as significant matches. If launched from SeqLab, the output will be located in the your working directory and it will have a cryptic name of the form profilesearch_some-number.pfs.

ProfileSearch finds all of the Elongation Factors in the PIR/NBRF protein database plus many other interesting nucleotide binding proteins down near the end of the list, all with Z scores >4. The nucleotide binding motifs in the EF-1α profile are among the most highly conserved portions of the alignment; therefore, more importance is placed on them by the search resulting in other proteins with similar domains also being found.

A greatly abridged screen trace of a sample ProfileSearch output with a profile built form most of the length of my Elongation Factor 1α alignment follows below. I've excluded many of the entries that I would expect and left some of the surprises.

```
!!SEQUENCE_LIST 1.0
(Peptide) PROFILESEARCH of: /users1/thompson/seqlab/primitive.prf Length: 428 to: pir:*

         Scores are not corrected for composition effects

                 Gap Weight: 60.00
          Gap Length Weight: 0.67
         Sequences Examined: 188196
         CPU time (seconds): 2713
*    *    *    *    *    *    *    *    *    *    *    *    *    *    *
Profile information:
(Peptide) PROFILEMAKE v4.50 of:
 @/users1/thompson/.seqlab-mendel/profilemake_63.list  Length: 428
  Sequences: 38  MaxScore: 1798.78  July 11, 2001 20:11
                          Gap: 1.00             Len: 1.00
                     GapRatio: 0.33         LenRatio: 0.10
       input_63.rsf{GIARDIA_L}  From: 19        To: 446       Weight: 1.00
   input_63.rsf{DIPLOMONAD_SP}  From: 19        To: 446       Weight: 1.00 . . .

*    *    *    *    *    *    *    *    *    *    *    *    *    *    *    *
Normalization:                                  July 11, 2001 21:21

         Curve fit using 49 length pools
         0 of 49 pools were rejected

         Normalization equation:

                 Calc_Score = 66.96 * ( 1.0 - exp(-0.0023*SeqLen - 0.6191) )

         Correlation for curve fit: 0.973

         Z score calculation:
         Average and standard deviation calculated using 99616 scores
         384 of 100000 scores were rejected

                 Z_Score = ( Score/Calc_Score - 1.010 ) / 0.164

          Sequence  Strd ZScore   Orig Length ! Documentation  ..
PIR2:A49171           +  158.30 1454.17    435 ! translation elongation factor e
EF-1 alpha chain - Tetrahymena pyriformis
PIR2:A54760           +  157.48 1458.18    449 ! translation elongation factor e
EF-1 alpha chain - Trypanosoma brucei
PIR2:S11665           +  156.90 1458.53    456 ! translation elongation factor e
EF-1 alpha chain - slime mold (Dictyostelium discoid
PIR2:S16308           +  156.81 1449.85    446 ! translation elongation factor e
EF-1 alpha chain - Stylonychia lemnae
PIR2:JC5117           +  155.73 1442.59    449 ! translation elongation factor e
EF-1 alpha - Trypanosoma cruzi
PIR2:T43890           +  154.38 1385.87    395 ! translation elongation factor e
EF-1 alpha [similarity] - Dinenympha exilis (fragmen
PIR2:T43892           +  154.08 1383.28    395 ! translation elongation factor e
EF-1 alpha [similarity] - unidentified Oxymonadida A
PIR2:A60491           +  152.65 1425.02    462 ! translation elongation factor e
EF-1 alpha chain - African clawed frog
PIR2:JU0133           +  152.61 1424.67    462 ! translation elongation factor e
EF-1 alpha chain - Chinese hamster
PIR2:S21055           +  152.61 1424.67    462 ! translation elongation factor e
EF-1 alpha chain - rat
PIR2:I50226           +  152.35 1422.28    462 ! translation elongation factor e
////////////////////////////////////////////////////////////////////////////
F-1 alpha chain - Thermococcus celer
PIR2:F69007           +  100.48 930.88    413 ! translation elongation factor aE
F-1 alpha chain - Methanobacterium thermoautotrophi
```

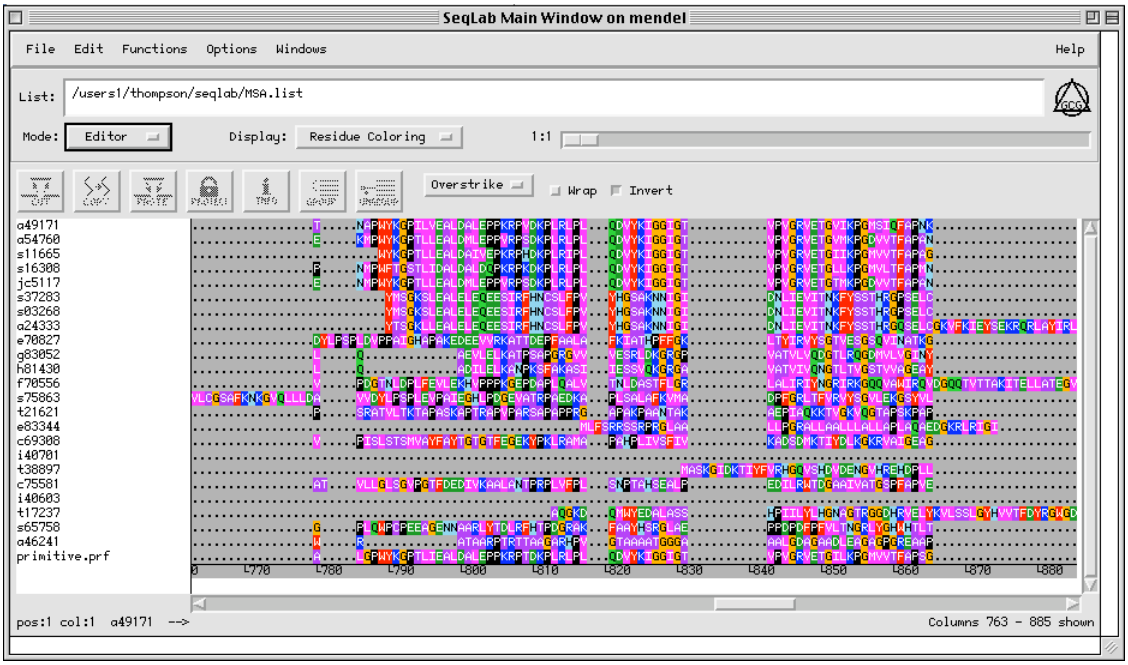
```
PIR2:A37159           +   86.47 760.89    325 ! translation elongation factor eE
F-1 alpha-related centrosphere protein - sea urchin
PIR2:I54251           +   74.38 604.24    227 ! translation elongation factor eE
F-1 alpha - human (fragment)
PIR2:T03718           +   71.88 679.53    409 ! suppressor 2 protein homolog - c
ommon tobacco (fragment)
PIR2:T03717           +   70.48 705.60    515 ! GTP-binding protein SUP1, EF-1-a
lpha-related - common tobacco
////////////////////////////////////////////////////////////////////////////////
PIR2:S37283           +    9.80 154.03    639 ! tetracycline resistance protein
tetM - Neisseria meningitidis
PIR2:S03268           +    9.80 154.03    639 ! tetracycline resistance protein
tetM - Ureaplasma urealyticum
PIR2:A24333           +    9.69 153.00    639 ! tetracycline resistance protein
tetM - Enterococcus faecalis transposon Tn1545
PIR2:E70827           +    9.66 155.56    701 ! probable fusA protein - Mycobact
erium tuberculosis (strain H37RV)
PIR2:G83052           +    9.66 160.60    840 ! translation initiation factor IF
-2 PA4744 [imported] - Pseudomonas aeruginosa (stra
PIR2:H81430           +    9.44 159.24    871 ! translation initiation factor IF
-2 Cj0136 [imported] - Campylobacter jejuni (strain
PIR2:F70556           +    9.35 149.14    628 ! hypothetical protein Rv1165 - My
cobacterium tuberculosis (strain H37RV)
////////////////////////////////////////////////////////////////////////////////
PIR2:A34347           +    5.01 113.10    830 ! translation elongation factor eE
F-2 - slime mold (Dictyostelium discoideum)
PIR2:F70180           +    4.99  78.70    176 ! ferric uptake regulation protein
 (fur) homolog - Lyme disease spirochete
PIR2:T21362           +    4.99 113.39    852 ! hypothetical protein F25H5.4 - C
aenorhabditis elegans
PIR2:S53707           +    4.99 110.33    727 ! translation initiation factor eI
F-2 — bovine
PIR2:T21621           +    4.98 113.85    878 ! hypothetical protein F32A7.5 - C
aenorhabditis elegans
PIR2:E83344           +    4.98  90.86    317 ! probable adhesion protein PA2407
 [imported] - Pseudomonas aeruginosa (strain PAO1)
PIR2:C69308           +    4.97  93.48    355 ! immunogenic protein (bcsp31-1) h
omolog - Archaeoglobus fulgidus
PIR2:I40701           +    4.97  89.04    294 ! glyceraldehyde-3-phosphate dehyd
rogenase (EC 1.2.1.12) - Citrobacter freundii (frag
PIR2:T38897           +    4.96  82.42    216 ! hypothetical protein SPAC513.02
- fission yeast (Schizosaccharomyces pombe)
PIR2:C75581           +    4.96 105.14    580 ! malate oxidoreductase - Deinococ
cus radiodurans (strain R1)
PIR2:I40603           +    4.96  82.02    212 ! hypothetical protein A - Clostri
dium acetobutylicum
PIR2:T17237           +    4.96  85.14    247 ! hypothetical protein DKFZp434P10
6.1 - human (fragment)
PIR2:S65758           +    4.96 110.29    737 ! nitrate reductase (EC 1.7.99.4)
chain A narB - Oscillatoria chalybea
PIR2:A46241           +    4.95  87.60    277 ! interferon response element-bind
ing factor IREBF-2 - mouse (fragment)
////////////////////////////////////////////////////////////////////////////////
```

Notice the very clean demarcation in Z scores between the EF-1α orthologues, with Z scores above around 100, and all the GTP-binding proteins below that with Z scores from below 100 to almost 5, and what is most likely just noise, with Z scores of around 5 and less. Notice that this search has many entries in common with the previous searches but there are also substantial differences. This is another reason why it is always a good idea to run as many different types of analyses as practical.

The program ProfileSegments makes BestFit style alignments of the results of a ProfileSearch. A great option in ProfileSegments, -MSF, allows you to prepare a multiple sequence alignment of the ProfileSearch segments. This can be very helpful for merging ever-increasingly distant sequences into an alignment. The full information content of the profile including the importance of the conserved portions of your alignment is used in this alignment procedure. When checking out a ProfileSearch output, something I'll often do is edit it to exclude (or comment out by placing an exclamation point at the beginning of the entry's line) most of the sequences that I expected to be found by the search, except a few positive controls; i.e. in my example most of the EF-1α's. If you ever do this, be sure not mess with the header portion of the file, it specifies your profile's directory location! Alignments are made from the modified

ProfileSearch output file with ProfileSegments. When running ProfileSegments be sure to set your list size big enough to include all of the relevant sequences remaining in the ProfileSearch output. Another handy option is -Global versus the -Local default; this will force full-length alignments, which might be what you would want, especially if you are trying to build up a multiple sequence alignment.

A screen snapshot centered about the t-RNA binding region of a ProfileSegments -MSF -Global alignment made from many of the entries from the above ProfileSearch example aligned against my example EF-1α profile follows below:



Notice the difference between this alignment and examples seen with other algorithms. Profile alignments are often much more 'gappy' than other alignments, more so than just that caused by the extreme divergence of this example. The conserved portions of the profile do not allow the corresponding portion of alignment to gap; yet gaps are easily put in the non-conserved regions of the alignment. 'Clustering' is much more critical to Profile analyses than other methods. This is because of profile analysis' variable gap penalties where conserved areas are not allowed to gap and variable regions are. This can be a very handy strategy for pregapping new sequences to introduce them into existing alignments.

2.14.3. HMMER — Hidden Markov Modeling and Profiles.

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, Sean 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. To build a HMMER profile of an alignment in SeqLab, select all of the relevant sequences, and perhaps a region within them to exclude jagged, unalignable ends. Do not select a Mask sequence, as profiles need to include all of the ambiguity of the alignment within the region being used, and would be the wrong length if any alignment columns were excluded. Go to the "**Functions**" "**HMMER**" menu and pick "**HmmerBuild**." Specify "**Selected region**" rather than "**Selected sequences**" if restricting your profile's length. 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 on 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. It again becomes important to understand the types of biological questions that you are asking to rationally set many of the program parameters.

Notice HmmerCalibrate is checked by default. The completion of HmmerBuild automatically launches a calibration procedure that increases the speed and accuracy of subsequent analyses with the resultant profile. The other HmmerBuild options can be explored, but read the Program Manual first. For now accept the default HmmerBuild parameters and press "**Run**." 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.

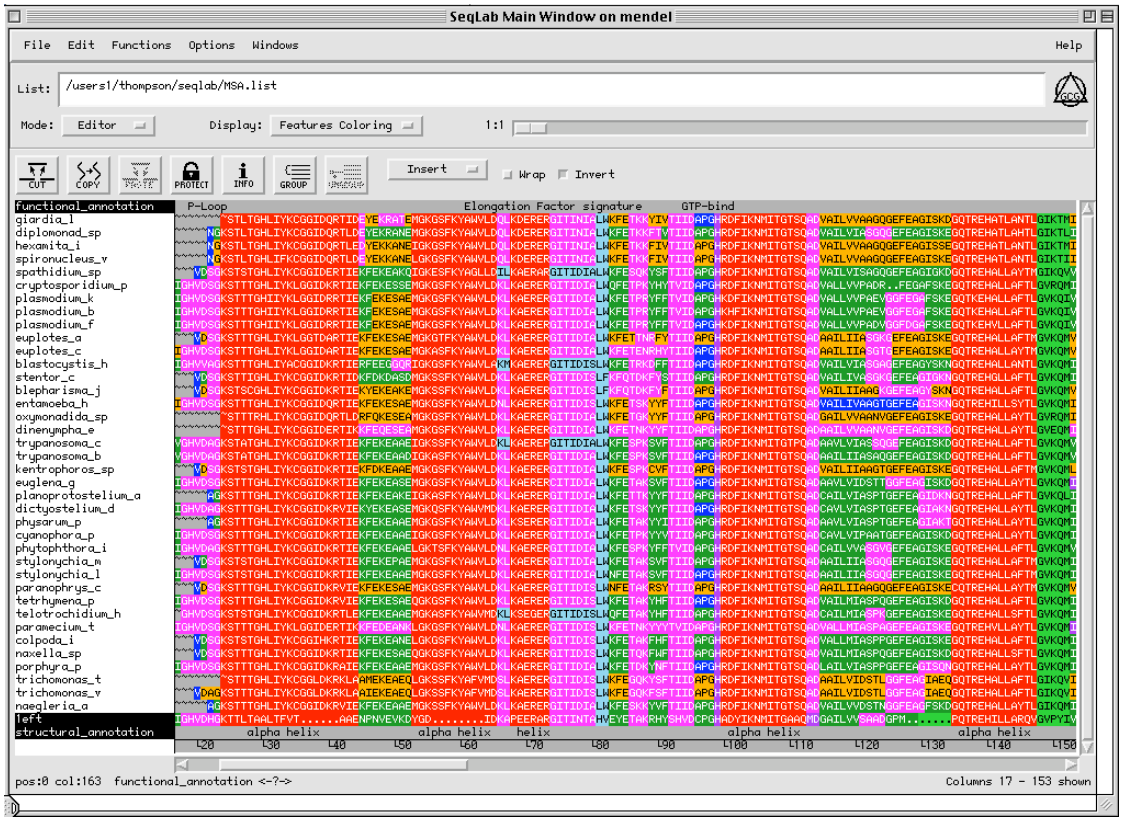
A utility program, HmmerConvert, can change HMMER style profiles into Gribskov profiles, however information is lost in the process. Normally you would use your new HMMER profile as either a search probe for extremely sensitive database searching or as a template upon which to build ever-larger multiple sequence alignments.

To use a HMMER profile as a search probe go to the "**Functions**" menu and pick "**HMMER**" "**HmmerSearch**." Specify the new HMMER profile by clicking "**Profile HMM to use as query. . .**" and using the "**File Chooser**" window to select the correct HMMER profile. Either accept the default "Sequence search set. . ." "PIR:*" specification or choose other sequences to search. 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 it is a true dynamic programming implementation, a HMMER profile matrix against a whole database. So definitely 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, HmmerSearch supports the multithreading -Processors=x option to speed things up. "Run" the program when you've got the options set the way you want them. The output is huge but very informative. Everything is based on significance Expectation value scores. The top portion is a list of best hits based on all domains, the second section is the GCG list file portion of the best domain hits, next pairwise alignments are given, and finally a score distribution is plotted. Since it is a GCG list file, it can be read by other GCG programs, in particular HmmerAlign.

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, 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 command line -Heuristic=some.msf{*} option. Using the original alignment that you made the profile with, against another sequence set is very fast; it is the -MapAlignment=some.rsf{*} 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.

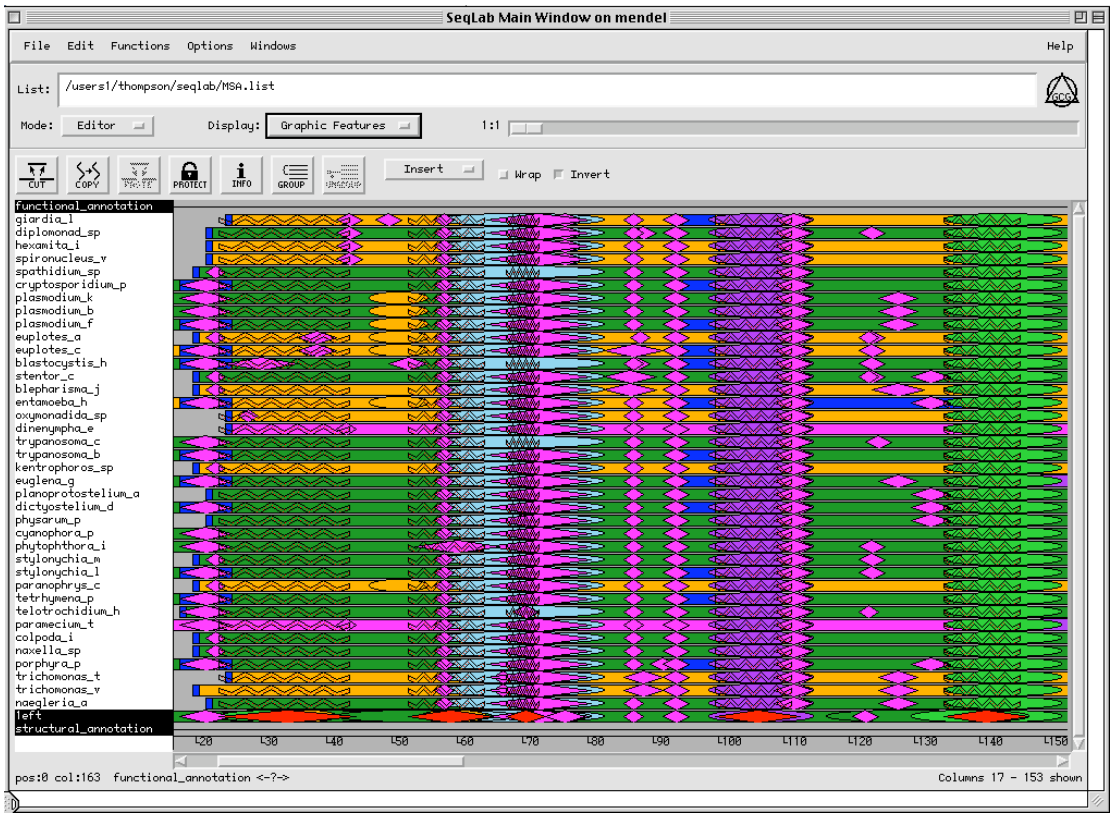
1EFT is one of the most similar Elongation Factor 1α homologues to my example 'lower' eukaryote EF-1α profile that has a solved structure. Therefore, an alignment of its primary sequence with structural annotation against my sample dataset should allow a decent inference of secondary structure across the entire alignment. This is the basis of homology modeling. Here I've loaded the results of a HmmerAlign run on NRL_3D:1EFT, the EF-Tu structure from *Thermus aquaticus*, against my example EF-1α HMMER profile and its associated alignment. My inferred secondary structure is illustrated in the following "Features Coloring" graphic by highlighting the alpha helices in red:



2.14.4. HmmerPfam.

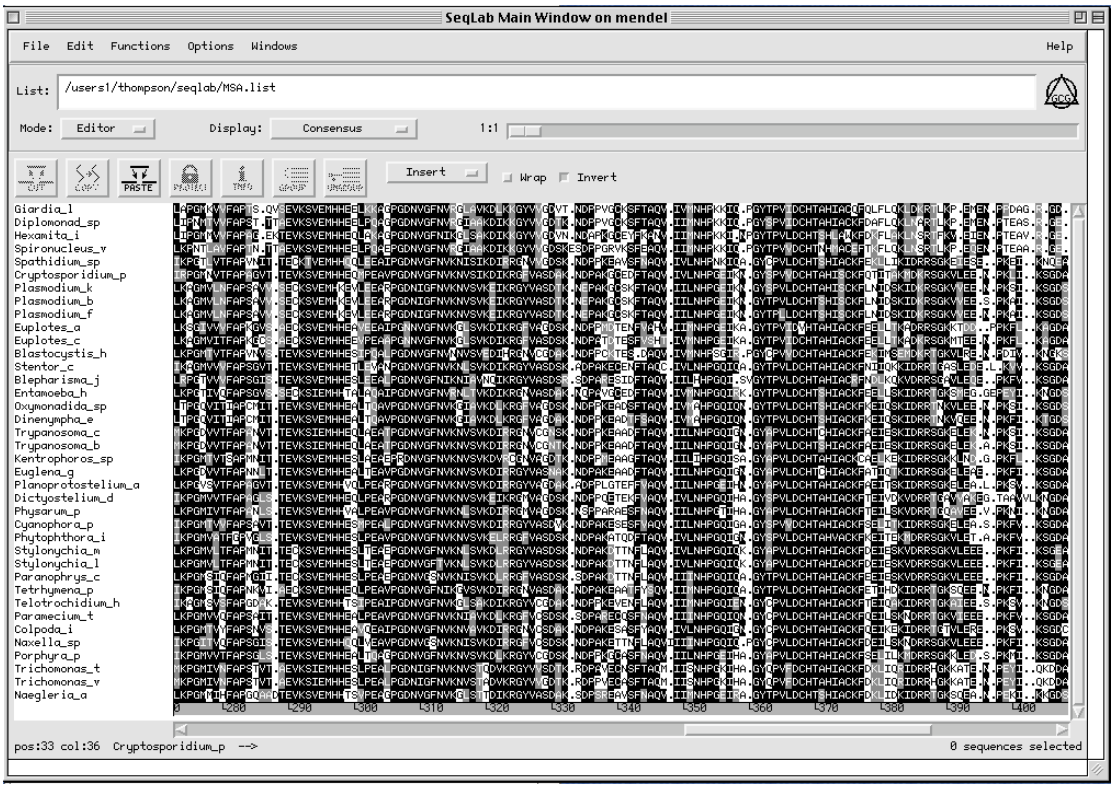
As with Motifs and MotifSearch, HmmerPfam can help build up the annotation of an RSF file. This program scans sequences against a library of HMMER profiles, by default the Pfam library (A database of protein domain family alignments and HMMs © 1996-2000 The Pfam Consortium). Select all of your protein sequences (do not select annotation or mask lines) and launch the program through the "**Functions**" "**HMMER**" "**HmmerPfam. . .**" menu. "**Save the best**

scoring profile HMMs as an RSF file" and give an appropriate name. You can check out the options if desired; you may want to reduce the Expectation cutoff values. "**Run**" the program. When its finished (It can take quite a while to run — don't wait for it to finish.) add it's RSF output file to the Editor display as before with the "**Output Manager**"'s "**Add to Editor**" and "**Overwrite old with new**" functions. The output .hmmerpfam file lists Pfam domain matches ranked by Expectation values and with the -RSF option writes the domain identification and Expectation value as a feature in an RSF file. The screen snapshot below shows my sample alignment over the same span as above but now including additional HmmerPfam annotation using "Graphic Features" "Display:" mode. Inferred alpha helices are now seen as transparent red coils:



2.15. Consensus and Masking Issue — GCG's Mask operation.

Consensus methods are another powerful way to visualize similarity within an alignment besides GCG's PlotSimilarity program. The SeqLab "Edit" menu allows you to easily create several types of consensus representations. To create a standard protein sequence consensus select all your sequences and use the "**Edit**" "**Consensus . . .**" menu and specify "**Consensus type:**" "**Protein Sequence**." When making a normal sequence consensus of a protein alignment you can generate figures with black highly similar residues, gray intermediate similarities, and white non-similar amino acids. This is a nice way to prepare alignment figures for publication. The default mode is to create an identity consensus at the 2/3'rds plurality level ("Percent required for majority") with a threshold of 5 ("Minimum score that represents a match"). Try different lower plurality and threshold values as well as different scoring comparison matrices to see the difference that it can make in the appearance of your alignment. Be sure that "**Shade based on similarity to consensus**" is checked to generate a color mask overlay on the display to help in the visualization process. The following screen illustrates a region near the carboxy termini of my example using the BLOSUM30 matrix, a "Percent required for majority" (plurality) of 33%, and a "Minimum score that represents a match" (threshold) cutoff value of 4:



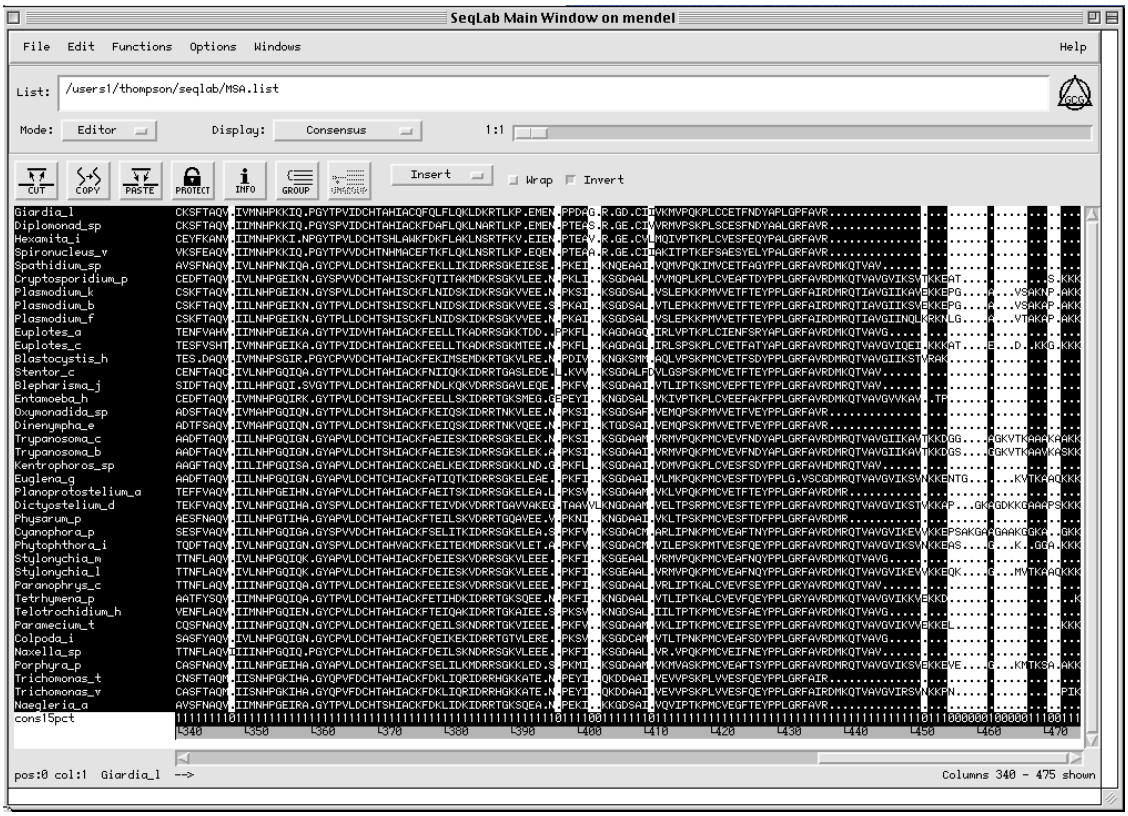
When you've found a plurality combination that you like, an available option is to go to the "File" "Print. . ." command and change the "Output Format:" to "PostScript" in order to prepare a PostScript file of your SeqLab display. Whatever color scheme that is being displayed by the Editor at the time will be captured by the PostScript file. Play around with the other parameters — notice that as you change the font size the number of pages to be printed varies. In the "Print Alignment" menu specify "Destination. . . File" and give it an appropriate filename and then click "OK." This command will produce a PostScript language graphics file in the directory that you launched SeqLab from and is a great way to prepare presentations of your research. This PostScript file can be sent to a color PostScript printer, or a black and white laser printer that will simulate the colors with gray tones, or it can be imported into a PostScript savvy graphics program for further manipulation. Unfortunately, if it's longer than one page, the 'raw' PostScript format is so different from standard Encapsulated PostScript that you may have to use a different UNIX print queue. Discuss these matters with your system administrator. It may require some variation of the following type of command:

```
> lpr -PPostScript_que seqlab_alignment.ps
```

In addition to standard consensus sequences using various similarity schemes, SeqLab also allows you to create consensus "Masks" that screen specified areas of your alignment from further analyses by specifying 0 or 1 weights for each column. A SeqLab Mask allows the user to differentially weight different parts of their alignment to reflect their confidence in it. It can be a handy trick with some data sets, especially those with both highly conserved and highly variable regions. Masks can be modified by hand or they can be created manually through the "New Sequences" menu. They can have position values all the way up to 9, though I doubt anyone would want any column of an alignment to be nine times as important as some other column. Masking is especially helpful for phylogenetic analysis by excluding those less reliable columns in your alignment where you are not confident in the positional homology without actually getting rid of the data.

Once a Mask has been created in SeqLab, most of the programs available through the "Functions" menu will use that Mask, if the Mask is selected along with the desired sequences, to weight the columns of the alignment data matrix appropriately. This only occurs through the "Functions" menu.

To create a Mask style sequence consensus select all your sequences and then use the "**Edit**" "**Consensus . . .**" menu and specify "**Consensus type:**" "**Mask Sequence**." As above, the default mode uses an identity consensus at the 2/3'rds plurality level with a threshold of 5. However, these are very high values for phylogenetic analysis and would likely not leave much phylogenetically informative data. Therefore, again experiment with different lower pluralities, threshold values, and scoring comparison matrices. Be sure that "**Shade based on similarity to consensus**" is still checked. The following screen illustrates the carboxy terminal end of my example using a weight Mask generated from the BLOSUM30 matrix, a plurality of 15%, and a threshold of 4:



Few areas are excluded by the Mask in this alignment because of the high similarity of this group of sequences. This is as it should be, for excluding many more columns in this particular alignment would likely leave nearly identical sequences and it would be impossible to ascertain how they are related.

2.16. Coding DNA Issues.

When dealing with very similar sequences, it is usually best to align DNA sequences along with their corresponding proteins (the "Group" function is very helpful for this). Phylogenetic analyses is then performed on the DNA rather than on the proteins. This is especially important when dealing with datasets that are quite similar since the proteins may not reflect many differences hidden in the DNA. Furthermore, many people prefer to run phylogenetic analyses on DNA rather than protein regardless of how similar they are — the multiple substitution models are much more robust for DNA. In fact, many phylogenetic inference algorithms do not even take advantage of amino acid similarity when dealing with protein sequences; they only count identities! However, the more diverged a dataset becomes, the more random

third and eventually first codon positions become, which introduces noise (error) into the analysis. Therefore, often third positions and sometimes first positions are masked out of datasets. Just like in most of computational molecular biology, one is always balancing signal against noise. Too much noise or too little signal both degrade the analysis to the point of nonsense.

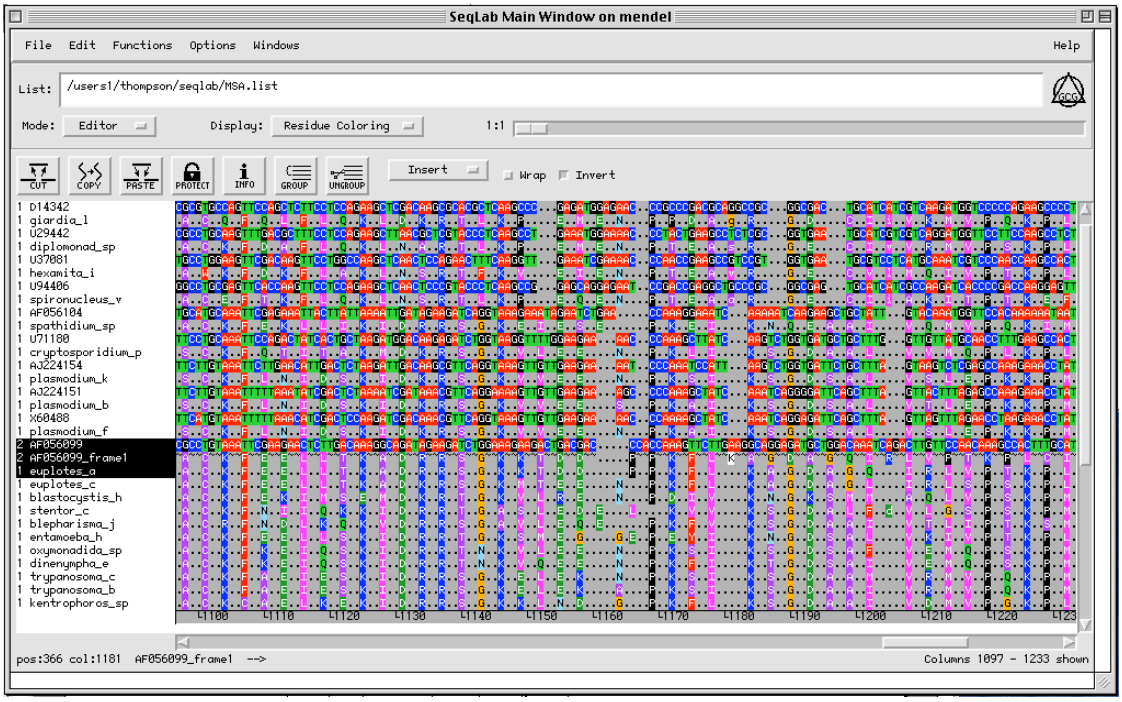
The logic to this paired protein and DNA alignment approach is as follows:

1) The easy case where you can align the DNA directly. If the DNA sequences are directly alignable because they are quite similar, then merely create your DNA alignment. Next use the "Edit" menu "Translate" function and the "align translations" option to create aligned corresponding protein sequences. Select the region to translate based on the CDS reference in each DNA sequence's annotation. Be careful of CDS entries that do not begin at position 1 — the GenBank CDS feature annotation "/codon_start=" identifies which position the translation begins within the first codon listed. You may also have to trim sequences down to just the relevant gene, especially if they're genomic. You'll have to change their protections with the padlock icon if this is the case. Group each protein to its corresponding DNA sequence so that subsequent manipulations will keep them together.

2) The way more difficult case where you need to use the protein sequences to create the alignment because the DNA is not directly alignable. In this case you need to load the protein sequences first, create their alignment, and then load their corresponding DNA sequences. You can find the DNA sequence accession codes in the annotation of the protein sequence entries. Next translate the unaligned DNA sequences into new protein sequences with the Edit-Translate function using the "align translations" option and Group these to their corresponding DNA sequences, just as above. However, this time the DNA along with their translated sequences are not aligned as a set, just the other protein set is aligned. Also, Group all of the aligned protein dataset together, separately from the DNA/aligned translation set. Now comes the manual part; painstakingly rearrange your display to place the DNA, its aligned translation, and the original aligned protein sequence side-by-side and then manually slide one set to match the other. Use the "CUT" and "PASTE" buttons to move the sequences around. When pasting realize that the "Sequence clipboard" contains complete sequence entries, whereas the "Text clipboard" only contains sequence data, amino acid residues or DNA bases as the case may be. The translated sequence entries can be "CUT" away after they're aligned to the rest of the set. Merge the newly aligned sequences into the existing alignment Group as you go and then start on the next one. It sounds difficult, but since you're matching up two identical protein sequences, the DNA translation and the original aligned protein, it's really not too bad. The Group function keeps everything together the way it should be so that you don't lose your original alignment as you space residues apart to match them up to their respective codons. Some codons may become spaced apart in this process and will have to be adjusted afterwards. As usual, save your work often.

My final, completely aligned, sample data RSF file with the *Thermus aquaticus* sequences aligned to the primitive eukaryotic protein and DNA sequences, and all annotation, is available at my home URL in the Data_Files directory, in case you wish to play with it (http://bio.fsu.edu/~stevet/Data_Files/EF1a-primitive.rsf).

A screen dump of my sample dataset part way through the DNA-protein alignment process follows below:



2.17. Multiple Alignment Format and Phylogenetics.

As mentioned in the previous chapter, multiple sequence alignment is a necessary prerequisite for biological sequence based phylogenetic inference, and phylogenetic inference guides our understanding of molecular evolution. The famous Darwinian Theodosius Dobzhansky summed it up succinctly in 1973, provided as an inscription on the inner cover of the classic organic evolution text *Evolution*: "Nothing in biology makes sense except in the light of evolution" (Dobzhansky, et al., 1977). These words ring true. To me, evolution provides the single, unifying, cohesive force that can explain all life. It is to the life sciences what the long sought holy grail of the unified field theory is to astrophysics.

2.17.1. GCG's Interface to PAUP* —

Multiple alignment format issues and conversion to two well accepted phylogenetic formats.

GCG implements David Swofford's PAUP* (usually pronounced 'pop star') phylogenetic analysis package (Swofford, 1989–2003) with the paired programs PAUPSearch and PAUPDisplay. These interface programs provide an easy to use access to much of PAUP* within GCG. However, their use for evolutionary inference will not be covered here. For serious phylogenetic analysis you may want to consider running PAUP* exterior to GCG by getting the latest version directly from Sinauer Associates, the publishing company that distributes the software (http://www.sinauer.com/), and installing it on your own machine. The version of PAUP*, included in GCG version 9.1 through 10.3, either run in native mode or through the PAUPSearch and PAUPDisplay programs, is an old 4.0.0d55 version. If you do not have access to the latest and greatest version of PAUP*, which contains many bugs fixes and enhancements since 4.0.0d55, then using it within GCG is a legal alternative. Use the following command in a terminal window to read the license agreement with GCG, if you're curious:

```
> typedata paup-license.txt
```

The PAUP package was originally written to only perform parsimony analysis with either DNA sequences or morphological character data using a Macintosh. It latest incarnation, version 4.0+, changed the package's name by adding the asterisk which means "and other methods" referring to the incorporation of the minimum evolution distance

method and the maximum likelihood method to the package. It was also expanded into a "portable" package capable of being run on many different platforms using a command line interface in addition to its original Mac graphical user interface. PAUP* is weak at dealing with protein sequences as it does not incorporate any protein models of evolution other than a crude like/not-like model. However, more sophisticated protein models can be used by embedding the necessary commands and matrices in the NEXUS file used as input to the package. Though, as I discussed previously, many people prefer to perform evolutionary inference with DNA sequences anyway. Furthermore, PAUP*'s DNA models are perhaps the most sophisticated available in any molecular phylogenetic inference software, and I, therefore, heartily recommend using it for DNA datasets.

2.17.2. NEXUS Format.

Within the context of GCG NEXUS format files are most easily and reliably built from alignments with GCG's own interface to the PAUP* package. PAUPSearch within SeqLab can be used to generate NEXUS format files which can then be fed directly to any version of PAUP*.

Begin the NEXUS conversion process by selecting all relevant sequences, and any desired weight Masks, in the "Main Window" display. Select "**PAUPSearch. . .**" from the "**Functions**" "**Evolution**" menu to launch the dialogue box. To only generate a NEXUS file, run PAUPSearch in its fastest mode without actually performing a search. Accept the default "**Tree Optimality Criterion**" "**maximum parsimony**" and the "**heuristic tree search (fast)**" "**Method for Obtaining Best Tree(s)**." Be sure that the "**perform bootstrap replications. . .**" button is not pressed and then launch the "**Options**" menu by pressing the appropriate button. In the "**PAUPSearch Options**" menu check in the top box to save the PAUPscript file. This is not required for running the programs but since we are just generating NEXUS format, it is essential. You can change or leave the file name as you wish. The PAUPscript output file results from the automatic conversion of the alignment to NEXUS format and contains all the PAUP commands as well as ther alignment. (If needed, the PAUPlog file keeps track of all that happened during the program run and is a good place to look for any error messages. It is, therefore, a handy file to save to avoid otherwise frustrating troubleshooting.) Uncheck the next box, "**Perform the analysis**." This makes the program do the conversion to generate the NEXUS script but prevents it from performing the heuristic search for the best tree (equivalent to the command line option –NoRun). Scroll down through the options menu, leaving the rest of the options at their default settings, but do check them out. "**Close**" the options menu. Normally PAUPSearch and PAUPDisplay are linked to each other when you run them from the SeqLab interface. Therefore, uncheck the "**PAUPDisplay. . .**" button in PAUPSearch's main window to turn PAUPDisplay off. Be sure that "**How:**" "**Background Jo**b" is specified on the main PAUPSearch menu and then press "**Run**" there. After a moment the output PAUPscript file will be displayed. Here's my abridged Elongation Factor protein dataset example:

```
#NEXUS

[! Aligned sequences from GCG file(s) '@/users1/thompson/.seqlab-mendel/paupsear
ch_51.list' ]

[Length: 441  Type: P  May 15, 2001 15:07]

[ Name: GIARDIA_L        Len:   441  Check: 2966  Weight:  1.00]
[ Name: DIPLOMONAD_SP    Len:   441  Check: 4608  Weight:  1.00]
[ Name: HEXAMITA_I       Len:   441  Check: 9530  Weight:  1.00]
[ Name: SPIRONUCLEUS_V   Len:   441  Check: 2245  Weight:  1.00]
[ Name: SPATHIDIUM_SP    Len:   441  Check: 2937  Weight:  1.00]
[ Name: CRYPTOSPORIDIUM_P  Len:   441  Check: 7665  Weight:  1.00]
[ Name: PLASMODIUM_K     Len:   441  Check: 9956  Weight:  1.00]
[ Name: PLASMODIUM_B     Len:   441  Check: 9937  Weight:  1.00]
[ Name: PLASMODIUM_F     Len:   441  Check:  796  Weight:  1.00]
[ Name: EUPLOTES_A       Len:   441  Check: 8831  Weight:  1.00]
[ Name: EUPLOTES_C       Len:   441  Check: 8653  Weight:  1.00]
[ Name: BLASTOCYSTIS_H   Len:   441  Check: 9014  Weight:  1.00]
[ Name: STENTOR_C        Len:   441  Check: 5386  Weight:  1.00]
[ Name: BLEPHARISMA_J    Len:   441  Check: 7915  Weight:  1.00]
```

```
[ Name: ENTAMOEBA_H      Len:   441  Check: 8365  Weight:  1.00]
[ Name: OXYMONADIDA_SP   Len:   441  Check: 8531  Weight:  1.00]
[ Name: DINENYMPHA_E     Len:   441  Check: 5471  Weight:  1.00]
[ Name: TRYPANOSOMA_C    Len:   441  Check: 9945  Weight:  1.00]
[ Name: TRYPANOSOMA_B    Len:   441  Check:  960  Weight:  1.00]
[ Name: KENTROPHOROS_SP  Len:   441  Check: 1567  Weight:  1.00]
[ Name: EUGLENA_G        Len:   441  Check:  492  Weight:  1.00]
[ Name: PLANOPROTOSTELIUM_A  Len:   441  Check: 8843  Weight:  1.00]
[ Name: DICTYOSTELIUM_D  Len:   441  Check: 6233  Weight:  1.00]
[ Name: PHYSARUM_P       Len:   441  Check:  320  Weight:  1.00]
[ Name: CYANOPHORA_P     Len:   441  Check: 4176  Weight:  1.00]
[ Name: PHYTOPHTHORA_I   Len:   441  Check:  804  Weight:  1.00]
 //////////////////////////////////////////////////////////////
[ Name: PARAMECIUM_T     Len:   441  Check: 3452  Weight:  1.00]
[ Name: COLPODA_I        Len:   441  Check: 8135  Weight:  1.00]
[ Name: NAXELLA_SP       Len:   441  Check: 6970  Weight:  1.00]
[ Name: PORPHYRA_P       Len:   441  Check: 1559  Weight:  1.00]
[ Name: TRICHOMONAS_T    Len:   441  Check: 6212  Weight:  1.00]
[ Name: TRICHOMONAS_V    Len:   441  Check: 6532  Weight:  1.00]
[ Name: NAEGLERIA_A      Len:   441  Check: 7736  Weight:  1.00]

begin data;
     dimensions ntax=38 nchar=441;
     format datatype=protein interleave gap=.;
     matrix
[                    1                                                50]
            GIARDIA_L  .......... .......... STLTGHLIYK CGGIDQRTID EYEKRATEMG
          DIPLOMONAD_SP  .......... .......NGK STLTGHLIYK CGGIDQRTLD EYEKRANEMG
             HEXAMITA_I  .......... .......NGK STLTGHLIYK CGGIDQRTLE DYEKKANEIG
           SPIRONUCLEUS_V  .......... .......NGK STLTGHLIFK CGGIDQRTLD EYEKKANELG
            SPATHIDIUM_SP  .......... .....VDSGK STSTGHLIYK CGGIDERTIE KFEKEAKQIG
       CRYPTOSPORIDIUM_P  MGKEKTHINL VVIGHVDSGK STTTGHLIYK LGGIDKRTIE KFEKESSEMG
            PLASMODIUM_K  MGKEKTHINL VVIGHVDSGK STTTGHIIYK LGGIDRRTIE KFEKESAEMG
            PLASMODIUM_B  MGKEKTHINL VVIGHVDSGK STTTGHIIYK LGGIDRRTIE KFEKESAEMG
            PLASMODIUM_F  MGKEKTHINL VVIGHVDSGK STTTGHIIYK LGGIDRRTIE KFEKESAEMG
              EUPLOTES_A  .......... .....VDSGK STTTGHLIYK LGGTDARTIE KFEKESAEMG
              EUPLOTES_C  MGKEKEHLNL VVIGHVDSGK STTTGHLIYK LGGIDARTIE KFEKESAEMG
           BLASTOCYSTIS_H  MGKEKPHINL VVIGHVVAGK STTTGHLIYA CGGIDKRTIE RFEEGGQRIG
                STENTOR_C  .......... .....VDSGK STTIGHLIYK CGGIDKRTID KFDKDASDMG
            BLEPHARISMA_J  .......... .....VDSGK STSCGHLIYK CGGIDKRTIE KYEKEAKEMG
              ENTAMOEBA_H  MPKEKTHINI VVIGHVDSGK STTTGHLIYK CGGIDQRTIE KFEKESAEMG
           OXYMONADIDA_SP  .......... .......... STTTRHLIYK CGGIDQRTLD RFQKESEAMG
             DINENYMPHA_E  .......... .......... STTTGHLIYK CGGIDERTIK KFEQESEAMG
            TRYPANOSOMA_C  MGKEKVHMNL VVVGHVDAGK STATGHLIYK CGGIDKRTIE KFEKEAAEIG
            TRYPANOSOMA_B  MGKEKVHMNL VVVGHVDAGK STATGHLIYK CGGIDKRTIE KFEKEAADIG
         KENTROPHOROS_SP  .......... .....VDSGK STSTGHLIYK CGGIDKRTIE KFDKEAAEMG
                EUGLENA_G  MGKEKVHISL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEASEMG
      PLANOPROTOSTELIUM_A  .......... .......AGK STTTGHLIYK CGGIDKRTIE KFEKEAKEIG
          DICTYOSTELIUM_D  MESEKTHINI VVIGHVDAGK STTTGHLIYK CGGIDKRVIE KYEKEASEMG
               PHYSARUM_P  .......... .......AGK STTTGHLIYK CGGIDKRTIE KFEKEAAEMG
             CYANOPHORA_P  MGKQKTHINI VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAAEIG
           PHYTOPHTHORA_I  .......... .VIGHVDAGK STTTGHLIYK CGGIDKRTIE KFEKEAAELG
            STYLONYCHIA_M  .......... .....VDSGK STSTGHLIYK CGGIDKRTIE KFEKEPAEMG
            STYLONYCHIA_L  MPKEKNHLNL VVIGHVDSGK STSTGHLIYK CGGIDKRTIE KFEKEAAEMG
            PARANOPHRYS_C  .......... .....VDSGK STTTGHLIYK CGGIDKRVIE KFEKESAEMG
             TETRHYMENA_P  M.GDKVHINL VVIGHVDSGK STTTGHLIYK CGGIDKRVIE KFEKESAEQG
         TELOTROCHIDIUM_H  .......... ...GHVDSGK STSTGHLIYK CGGIDKRTLE KFEKEAAEMG
             PARAMECIUM_T  G.KDKLHVNL VVIGHVDSGK STTTGHLIYK LGGIDERTIK KFEDEANKLG
                COLPODA_I  .......... .....VDSGK STSTGHLIYK CGGIHKRTIE KFEKEANELG
              NAXELLA_SP  .......... .....VDSGK STTTGHLIYK CGGIDKRTIE KFEKESAEQG
              PORPHYRA_P  MGKEKQHVSI VVIGHVDSGK STTTGHLIYK CGGIDKRAIE KFEKEAAEMG
            TRICHOMONAS_T  .......... .......... STTTGHLIYK CGGLDKRKLA AMEKEAEQLG
  ////////////////////////////////////////////////////////////////////////////
[                    401                                          441]
            GIARDIA_L  CCETFNDYAP LGPFAVR... .......... .......... .
          DIPLOMONAD_SP  SCESFNDYAA LGRFAVR... .......... .......... .
             HEXAMITA_I  CVESFEQYPA LGRFAVR... .......... .......... .
           SPIRONUCLEUS_V  SAESYELYPA LGRFAVR... .......... .......... .
            SPATHIDIUM_SP  VCETFAGYPP LGRFAVRDMK QTVAV..... .......... .
       CRYPTOSPORIDIUM_P  CVEAFTDYPP LGRFAVRDMK QTVAVGVIKS VKKE....KK K
            PLASMODIUM_K  VVETFTEYPP LGRFAIRDMR QTIAVGIIKA VKKEAAKNAK K
            PLASMODIUM_B  VVETFTEYPP LGRFAIRDMR QTIAVGIIKS VKKEAAKAAK K
            PLASMODIUM_F  VVETFTEYPP LGRFAIRDMR QTIAVGIINQ LRKNAAKAAK K
              EUPLOTES_A  CIENFSRYAP LGRFAVRDMK QTVAVG.... .......... .
              EUPLOTES_C  CVETFATYAP LGRFAVRDMR QTVAVGVIQE IKKKE.KKKK K
           BLASTOCYSTIS_H  CVETFSDYPP LGRFAVRDMR QTVAVGIIKS TRAK...... .
                STENTOR_C  CVETFTEYPP LGRFAVRDMK QTVAV..... .......... .
```

```
          BLEPHARISMA_J  CVEPFTEYPP LGRFAVRDMR QTVAV..... .......... .
            ENTAMOEBA_H  CVEEFAKFPP LGRFAVRDMK QTVAVGVVKA V.TP...... .
          OXYMONADIDA_SP  VVETFVEYPP LGRFAVR... .......... .......... .
            DINENYMPHA_E  VVETFVEYPP LGRFAVR... .......... .......... .
           TRYPANOSOMA_C  CVEVFNDYAP LGRFAVRDMR QTVAVGIIKA VKKDAAAAAK K
           TRYPANOSOMA_B  CVEVFNDYAP LGRFAVRDMR QTVAVGIIKA VKKDGAAVSK K
        KENTROPHOROS_SP  CVESFSDYPP LGRFAVHDMR QTVAV..... .......... .
              EUGLENA_G  CVESFTDYPP LG.VSCGDMR QTVAVGVIKS VKKE.TKAKK K
    PLANOPROTOSTELIUM_A  CVETFTEYPP LGRFAVRDMR .......... .......... .
        DICTYOSTELIUM_D  CVESFTEYPP LGRFAVRDMR QTVAVGVIKS TKKAAAAAKK K
             PHYSARUM_P  CVESFTDFPP LGRFAVRDMR .......... .......... .
           CYANOPHORA_P  CVEAFTNYPP LGRFAVRDMR QTVAVGVIKE VKKEAGKAGK K
         PHYTOPHTHORA_I  TVESFQEYPP LGRFAVRDMR QTVAVGVIKS VKKEG.GGKK K
          STYLONYCHIA_M  CVEAFNQYPP LGRFAVRDMK QTVAVG.... .......... .
          STYLONYCHIA_L  CVEAFNQYPP LGRFAVRDMK QTVAVGVIKE VKKEGTKAKK K
          PARANOPHRYS_C  CVEVFSEYPP LGRYAVRDMK QTVAV..... .......... .
           TETRHYMENA_P  CVEVFQEYPP LGRYAVRDMK QTVAVGVIKK VKKD...... K
      TELOTROCHIDIUM_H  CVESFAEYPP LGRFAVRDMK QTVAVG.... .......... .
          PARAMECIUM_T  CVEIFSEYPP LGRFAVRDMK QTVAVGVIKV VKKE....KK K
             COLPODA_I  CVEAFSDYPP LGRFAVRDMK QTVAVG.... .......... .
           NAXELLA_SP  CVEIFNEYPP LGRFAVRDMK QTVAV..... .......... .
           PORPHYRA_P  CVEAFTSYPP LGRFAVRDMR QTVAVGVIKS VKKEGTKSAK K
          TRICHOMONAS_T  VVESFQEYPP LGRFAIR... .......... .......... .
          TRICHOMONAS_V  VVESFQEYPP LGRFAIRDMK QTVAVGVIRS VKKP....PI K
            NAEGLERIA_A  CVEGFTEYPP LGRFAVR... .......... .......... .

    ;
endblock;

begin paup;
set errorstop;
set criterion=parsimony;
set increase=no;
pset collapse=maxbrlen;
hsearch start=stepwise addseq=simple swap=tbr;
savetrees /brlens file='/users1/thompson/seqlab/paupsearch_51.pauptrees' replace
;
quit;
endblock;
```

The PAUPscript file contains the NEXUS format file that was generated by GCG to run PAUP*. Notice that columns of your alignment with zeroes in their Mask are excluded from the NEXUS alignment. This file can be used to run the latest version of PAUP*, if available, in its native mode by 'ftping' it to an appropriate machine. Using a Macintosh may be desirable in order to take advantage of PAUP*'s very friendly Macintosh graphical user interface. Since GCG automatically creates this file for you, correctly encoding all of the required format data, when you run PAUPSearch, there is no need to hassle with a later conversion of your alignment to NEXUS. File format conversion can be a huge headache and here GCG has done all of that work for you. When using this file as input to native PAUP* you will want to comment out or remove any inappropriate commands within the command block near the end of the file with a simple text editor. Likewise, this file can be greatly expanded by encoding any desired commands and rate matrices within its command block.

As stated above, I would recommend running the latest version of PAUP* available, but whatever version you run, learn how to run the most robust searches possible, before accepting any output as valid phylogenetic inference. Unfortunately the techniques of molecular phylogenetics are beyond the scope of this tutorial. I encourage you to investigate further.

2.17.3 PHYLIP Format.

Dr. Joseph Felsenstein's PHYLIP (PHYLogenetic Inference Package [1993]) programs from the University of Washington (http://evolution.genetics.washington.edu/phylip.html) use their own distinct file format. PHYLIP is a comprehensive freeware suite of thirty different programs for inferring phylogenies that can handle molecular sequence, restriction digest, gene frequency, and morphological character data. Complete documentation comes with the package. Methods available in the package include parsimony, distance matrix, and likelihood, as well as bootstrapping and consensus

techniques. A menu controls the programs and asks for options to set and starts the computation. Data is automatically read into the program from a text file in PHYLIP format called "infile." If it is not found, the user types in the proper data file name. Output is written into special files with names like "outfile" and "treefile". Trees written into "treefile" are in the Newick format, an informal standard agreed to in 1986 by authors of a number of major phylogeny packages. PHYLIP has been in distribution since 1980, and has over 6,000 registered users. It is the most widely distributed phylogeny package worldwide, and competes with PAUP/PAUP* as that responsible for the largest number of published trees.

To reliably generate PHYLIP format from GCG alignments in SeqLab we'll use a combination approach — GCG's ToFastA and Don Gilbert's ReadSeq (1990). But first go to the "**SeqLab Main Window**" "**File**" "**Export**" menu; click "**Format**" and notice that "**MSF**," "**GenBank**," and "**GDE2.2**" are all available for saving a copy of an RSF file in a few alternative formats. At this point do not export any of these formats and "**Cancel**" the window. Realize that using this export route does not use the Mask data to include or exclude columns from your alignment. To take advantage of the Mask data for subsequent phylogenetic analyses, export your alignment using another method. Therefore, after being sure that all of the relevant sequences, as well as any Mask sequence that you wish to experiment with, are selected. Next, go to the "**Functions**" menu, where all choices will be affected by the Mask that you've chosen, and choose "**Importing/Exporting**" "**ToFastA. . ..**" No options are required here; just press "**Run**" to convert the portion of the alignment that is not masked out into FastA format. FastA is a good intermediate format on the way to PHYLIP's required format. The new file will be displayed by SeqLab. The first part of my protein dataset FastA format output file is shown below:

```
>GIARDIA_L In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
--------------------STLTGHLIYKCGGIDQRTIDEYEKRATEMGKGSFKYAWVL
DQLKDERERGITININIALWKFETKKYIVTIIDAPGHRDFIKNMITGTSQADVAILVVAAGQ
GEFEAGISKDGQTREHATLANTLGIKTMIICVNKMDDGQVKYSKERYDEIKGEMMKQLKN
IGWK-EEFDYIPTSGWTGDNIMEKSDKMPWYEGPCLIDAIDGLKAPKRPTDKPLRLPIQD
VYKISGVGTVPAGRVETGELAPGMKVVFAPTS-QSEVKSVEMHHEELKKAGPGDNVGFNV
RGLAVKDLKKGYVVGDVTNDPPVGCKSFTAQVIVMNHPKKIQ-PGYTPVIDCHTAHIACQ
FQLFLQKLDKRTLKP-EMENPPDAR-GD-CIVKMVPQKPLCCETFNDYAPLGPFAVR---
-------------------
>DIPLOMONAD_SP In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
-----------------NGKSTLTGHLIYKCGGIDQRTLDEYEKRANEMGKGSFKYAWVL
DQLKDERERGITININIALWKFETKKFTVTIIDAPGHRDFIKNMITGTSQADVAILVIASGQ
GEFEAGISKEGQTREHATLAHTLGIKTLIVCVNKMDDPQVNYSEARYKEIKEEMQKNLKQ
IGYK-DEFDFIPTSGWTGDSIMEKSPNMPWYSGPCLIDAIDGLKAPKRPTDKPLRLPIQD
VYKINGVGTVPAGRVESGLLIPNMTVVFAPST-TAEVKSVEMHHEELPQAGPGDNVGFNV
RGIAAKDIKKGYVVGDTKNDPPVGCKSFTAQVIIMNHPKKIQ-PGYSPVIDCHTAHIACK
FDAFLQKLNARTLKP-EMENPTEAR-GE-CIVRMVPSKPLSCESFNDYAALGRFAVR---
-------------------
>HEXAMITA_I In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
-----------------NGKSTLTGHLIYKCGGIDQRTLEDYEKKANEIGKGSFKYAWVL
DQLKDERERGITININIALWKFETKKFIVTIIDAPGHRDFIKNMITGTSQADVAILVVAAGQ
GEFEAGISSEGQTREHATLANTLGIKTMIVAVNKMDDPQVNYSEARYTEIKTEMQKTFKQ
IGFK-EEFDFVPLSGWTGDNIMEASPKTPWYKGKCLIECIDGLKAPKRPNDKPLRLPIQD
VYKINGVGTVPAGRVESGELIPGMMVVFAPAG-ETEVKSVEMHHEQLAKAGPGDNVGFNI
KGLSAKDIKKGYVVGDVNNDAPKGCEYFKANVIIMNHPKKI-NPGYTPVLDCHTSHLAWK
FDKFLAKLNSRTFKV-EIENPTEAR-GE-CVMQIVPTKPLCVESFEQYPALGRFAVR---
-------------------
>SPIRONUCLEUS_V In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
-----------------NGKSTLTGHLIFKCGGIDQRTLDEYEKKANELGKGSFKYAWVL
DQLKDERERGITININIALWKFETKKFIVTIIDAPGHRDFIKNMITGTSQADVAILVVAAGQ
GEFEAGISKEGQTREHATLANTLGIKTIILCINKMDDPNVNYSKDRYNEIKTEMTKTLVA
IGYK-PEFNYIPTSGWTGLNIMEKTEKTGWYDGPCLIEAIDSLKPPKRPTDKCLRLPIQD
VYKINGVGTVPAGRVESGCLKPNTLAVFAPTN-TAEVKSVEMHHEELPQAEPGDNVGFNV
RGIAAKDIKKGYVVGDSKSDPPGRVKSFEAQVIIMNHPKKIQ-PGYTPVVDCHTNHMACE
FTKFLQKLNSRTLKP-EQENPTEAR-GE-CIAKITPTKEFSAESYELYPALGRFAVR---
-------------------
>SPATHIDIUM_SP In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
---------------VDSGKSTSTGHLIYKCGGIDERTIEKFEKEAKQIGKESFKYAGLL
DILKAERARGITIDIALWKFESQKYSFTIIDAPGHRDFIKNMITGTSQADVAILVISAGQ
GEFEAGIGKDGQTREHALLAYTMGIKQVVVAINKMD--AVQYNEERFTDIKKEVIDYLKK
MGSKKKMLMSLPISGFMGDNLIEKSDKMPWYKGDTILEALDRVERPKRPVAKPLRLPLQD
VYKITGVGTVPVGRVETGVIKPGTLVTFAPVNITTECKTVEMHHQQLEEAIPGDNVGFNV
//////////////////////////////////////////////////////////
```

Notice that it excludes those positions that were masked with zero and that it now follows all FastA format conventions including the automatic conversion of all GCG style gap periods and tildes to the more universal gap hyphen representation. This step, therefore, circumvents the common 'dot to dash' problem often encountered in sequence format conversion. "**Close**" the ToFastA output window. You may want to use the "**Output Manager**" to save the file under a name that makes more sense to you through the "**Save As . . .**" menu. Next, ReadSeq is used to convert this FastA format file to PHYLIP compatible format.

To do this either exit SeqLab with the "**File**" menu "**Exit**" choice, or temporarily switch to your background terminal window. If you exit, you will probably be asked if you want to save your RSF file and any changes in your list. Accept the suggested changes giving appropriate names, if you're interested in saving your data, and SeqLab will close. This will return you to your terminal window, formerly behind the SeqLab display, where we can run ReadSeq. This program can be used to change your FastA format file into something acceptable for PHYLIP use. A limitation of ReadSeq is it does not allow you to only choose a portion of an alignment, nor does it automatically convert dots and tildes to hyphens. However, since we've taken care of these points while in SeqLab, it'll work just fine for us here. ReadSeq runs a bit backward from what most people are used to though.

Begin the program by typing "**readseq**" at your command prompt in the terminal window. ReadSeq first prompts you for an appropriate output file name, not an input file. Do not make a mistake in this step by giving the name of your input file first. If you do, you will overwrite the input file while running the program, and then when it tries to read it, there will be nothing left to read! Next choose "**12**" off of the ReadSeq menu for the current PHYLIP format and then designate the input sequence. (Do not use the GCG {*} designator; this is not a GCG program.) Finally, after the program has read all of the input sequences, specify "All" the sequences by typing the word "**all**." When the program again asks for an input sequence, press return to inform it that you are done, and let it do its thing. A sample terminal session screen trace is shown below; user responses are in bold:

```
> readseq
readSeq (1Feb93), multi-format molbio sequence reader.

Name of output file (?=help, defaults to display):
EF1A.phy
         1. IG/Stanford           10. Olsen (in-only)
         2. GenBank/GB            11. Phylip3.2
         3. NBRF                  12. Phylip
         4. EMBL                  13. Plain/Raw
         5. GCG                   14. PIR/CODATA
         6. DNAStrider            15. MSF
         7. Fitch                 16. ASN.1
         8. Pearson/Fasta         17. PAUP/NEXUS
         9. Zuker (in-only)       18. Pretty (out-only)

Choose an output format (name or #):
12

Name an input sequence or -option:
EF1A.tfa
Sequences in EF1A.tfa  (format is  8. Pearson/Fasta)
 1)  GIARDIA_L In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 2)  DIPLOMONAD_SP In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 3)  HEXAMITA_I In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 4)  SPIRONUCLEUS_V In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 5)  SPATHIDIUM_SP In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 6)  CRYPTOSPORIDIUM_P In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 7)  PLASMODIUM_K In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 8)  PLASMODIUM_B In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 9)  PLASMODIUM_F In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 10)  EUPLOTES_A In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 11)  EUPLOTES_C In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 12)  BLASTOCYSTIS_H In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 13)  STENTOR_C In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 14)  BLEPHARISMA_J In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 15)  ENTAMOEBA_H In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
```

```
 16)  OXYMONADIDA_SP In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 17)  DINENYMPHA_E In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 18)  TRYPANOSOMA_C In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 19)  TRYPANOSOMA_B In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 20)  KENTROPHOROS_SP In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 21)  EUGLENA_G In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 22)  PLANOPROTOSTELIUM_A In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 23)  DICTYOSTELIUM_D In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 24)  PHYSARUM_P In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 25)  CYANOPHORA_P In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 26)  PHYTOPHTHORA_I In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 27)  STYLONYCHIA_M In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 28)  STYLONYCHIA_L In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 29)  PARANOPHRYS_C In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 30)  TETRHYMENA_P In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 31)  TELOTROCHIDIUM_H In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 32)  PARAMECIUM_T In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 33)  COLPODA_I In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 34)  NAXELLA_SP In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 35)  PORPHYRA_P In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 36)  TRICHOMONAS_T In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 37)  TRICHOMONAS_V In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
 38)  NAEGLERIA_A In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list


Choose a sequence (# or All):
all

Name an input sequence or -option:<rtn>
```

Never mind if you get a ". . . padded to fit" error message — the program is just doing what it is supposed to do. Do realize, though, that had I not used ReadSeq on the output from ToFastA to convert to PHYLIP, and had rather used a GCG MSF file as input, then an essential change would have to be made before it would be correct for PHYLIP. As mentioned before, periods and tildes will not work to represent indels (gaps); they must all be changed to hyphens (dashes). The following, rather strange, UNIX command works well for this step from the command line, but you should not need to use it, if you've followed my suggested procedure:

```
> tr \~\. \- < infile.phy > outfile.phy
```

The first part of my example PHYLIP output file is displayed below:

```
 38 439
GIARDIA_L    ---------- ---------- STLTGHLIYK CGGIDQRTID EYEKRATEMG
DIPLOMONAD   ---------- -------NGK STLTGHLIYK CGGIDQRTLD EYEKRANEMG
HEXAMITA_I   ---------- -------NGK STLTGHLIYK CGGIDQRTLE DYEKKANEIG
SPIRONUCLE   ---------- -------NGK STLTGHLIFK CGGIDQRTLD EYEKKANELG
SPATHIDIUM   ---------- -----VDSGK STSTGHLIYK CGGIDERTIE KFEKEAKQIG
CRYPTOSPOR   MGKEKTHINL VVIGHVDSGK STTTGHLIYK LGGIDKRTIE KFEKESSEMG
PLASMODIUM   MGKEKTHINL VVIGHVDSGK STTTGHIIYK LGGIDRRTIE KFEKESAEMG
PLASMODIUM   MGKEKTHINL VVIGHVDSGK STTTGHIIYK LGGIDRRTIE KFEKESAEMG
PLASMODIUM   MGKEKTHINL VVIGHVDSGK STTTGHIIYK LGGIDRRTIE KFEKESAEMG
EUPLOTES_A   ---------- -----VDSGK STTTGHLIYK LGGTDARTIE KFEKESAEMG
EUPLOTES_C   MGKEKEHLNL VVIGHVDSGK STTTGHLIYK LGGIDARTIE KFEKESAEMG
BLASTOCYST   MGKEKPHINL VVIGHVVAGK STTTGHLIYA CGGIDKRTIE RFEEGGQRIG
STENTOR_C    ---------- -----VDSGK STTIGHLIYK CGGIDKRTID KFDKDASDMG
BLEPHARISM   ---------- -----VDSGK STSCGHLIYK CGGIDKRTIE KYEKEAKEMG
ENTAMOEBA_   MPKEKTHINI VVIGHVDSGK STTTGHLIYK CGGIDQRTIE KFEKESAEMG
OXYMONADID   ---------- ---------- STTTRHLIYK CGGIDQRTLD RFQKESEAMG
DINENYMPHA   ---------- ---------- STTTGHLIYK CGGIDERTIK KFEQESEAMG
TRYPANOSOM   MGKEKVHMNL VVVGHVDAGK STATGHLIYK CGGIDKRTIE KFEKEAAEIG
TRYPANOSOM   MGKEKVHMNL VVVGHVDAGK STATGHLIYK CGGIDKRTIE KFEKEAADIG
KENTROPHOR   ---------- -----VDSGK STSTGHLIYK CGGIDKRTIE KFDKEAAEMG
EUGLENA_G    MGKEKVHISL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEASEMG
PLANOPROTO   ---------- -------AGK STTTGHLIYK CGGIDKRTIE KFEKEAKEIG
DICTYOSTEL   MESEKTHINI VVIGHVDAGK STTTGHLIYK CGGIDKRVIE KYEKEASEMG
PHYSARUM_P   ---------- -------AGK STTTGHLIYK CGGIDKRTIE KFEKEAAEMG
CYANOPHORA   MGKQKTHINI VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKEAAEIG
PHYTOPHTHO   ---------- -VIGHVDAGK STTTGHLIYK CGGIDKRTIE KFEKEAAELG
STYLONYCHI   ---------- -----VDSGK STSTGHLIYK CGGIDKRTIE KFEKEPAEMG
STYLONYCHI   MPKEKNHLNL VVIGHVDSGK STSTGHLIYK CGGIDKRTIE KFEKEAAEMG
PARANOPHRY   ---------- -----VDSGK STTTGHLIYK CGGIDKRVIE KFEKESAEMG
```

```
TETRHYMENA   M-GDKVHINL VVIGHVDSGK STTTGHLIYK CGGIDKRVIE KFEKESAEQG
TELOTROCHI   ---------- ---GHVDSGK STSTGHLIYK CGGIDKRTLE KFEKEAAEMG
PARAMECIUM   G-KDKLHVNL VVIGHVDSGK STTTGHLIYK LGGIDERTIK KFEDEANKLG
COLPODA_I    ---------- -----VDSGK STSTGHLIYK CGGIHKRTIE KFEKEANELG
NAXELLA_SP   ---------- -----VDSGK STTTGHLIYK CGGIDKRTIE KFEKESAEQG
PORPHYRA_P   MGKEKQHVSI VVIGHVDSGK STTTGHLIYK CGGIDKRAIE KFEKEAAEMG
TRICHOMONA   ---------- ---------- STTTGHLIYK CGGLDKRKLA AMEKEAEQLG
TRICHOMONA   ---------- -----VDAGK STTTGHLIYK CGGLDKRKLA AIEKEAEQLG
NAEGLERIA_   ---------- -------AGK STTTGHLIYK CGGIDKRVIE KFEKEAAEMG

             KGSFKYAWVL DQLKDERERG ITINIALWKF ETKKYIVTII DAPGHRDFIK
             KGSFKYAWVL DQLKDERERG ITINIALWKF ETKKFTVTII DAPGHRDFIK
             KGSFKYAWVL DQLKDERERG ITINIALWKF ETKKFIVTII DAPGHRDFIK
             KGSFKYAWVL DQLKDERERG ITINIALWKF ETKKFIVTII DAPGHRDFIK
             KESFKYAGLL DILKAERARG ITIDIALWKF ESQKYSFTII DAPGHRDFIK
             KGSFKYAWVL DKLKAERERG ITIDIALWQF ETPKYHYTVI DAPGHRDFIK
             KGSFKYAWVL DKLKAERERG ITIDIALWKF ETPRYFFTVI DAPGHKDFIK
             KGSFKYAWVL DKLKAERERG ITIDIALWKF ETPRYFFTVI DAPGHKHFIK
             KGSFKYAWVL DKLKAERERG ITIDIALWKF ETPRYFFTVI DAPGHKDFIK
             KGTFKYAWVL DKLKAERERG ITIDIALWKF ETTNRFYTII DAPGHRDFIK
             KASFKYAWVL DKLKAERERG ITIDIALWKF ETENRHYTII DAPGHRDFIK
             KGSFKYAWVL AKMKAERERG ITIDISLWKF ETRKDFFTII DAPGHRDFIK
             KSSFKYAWVL DKLKAERERG ITIDISLFKF QTDKFYSTII DAPGHRDFIK
             KSSFKYAWVL DKLKAERERG ITIDISLFKF QTDKFYFTII DAPGHRDFIK
             KGSFKYAWVL DNLKAERERG ITIDISLWKF ETSKYYFTII DAPGHRDFIK
             KGSFKYAWVL DKLKAERERG ITIDIALWKF ETGKYYFTII DAPGHRDFIK
             KGSFKYAWVL DKLKAERERG ITIDIALWKF ETNKYYFTII DAPGHRDFIK
             KSSFKYAWVL DKLKAEREPG ITIDIALWKF ESPKSVFTII DAPGHRDFIK
             KASFKYAWVL DKLKAERERG ITIDIALWKF ESPKSVFTII DAPGHRDFIK
             KGSFKYAWVL DKLKAERERG ITIDIALWKF ESPKCVFTII DAPGHRDFIK
             KGSFKYAWVL DKLKAERERC ITIDIALWKF ETAKSVFTII DAPGHRDFIK
             KASFKYAWVL DKLKAERERG ITIDIALWKF ETTKYYFTII DAPGHRDFIK
             KQSFKYAWVM DKLKAERERG ITIDIALWKF ETSKYYFTII DAPGHRDFIK
             KGSFKYAWVL DKLKSERERG ITIDIALWKF ETAKYYITII DAPGHRDFIK
             KGSFKYAWVL DKLKAERERG ITIDIALWKF ETPKYYVTII DAPGHRDFIK
             KTSFKYAWVL DNLKAERERG ITIDIALWKF ESPKYFFTVI DAPGHRDFIK
             KGSFKYAWVL DKLKAERERG ITIDIALWKF ETAKSVFTII DAPGHRDFIK
             KGSFKYAWVL DKLKAERERG ITIDIALWNF ETAKSVFTII DAPGHRDFIK
             KGSFKYAWVL DKLKAERERG ITIDISLWNF ETAKRSYTII DAPGHRDFIK
```

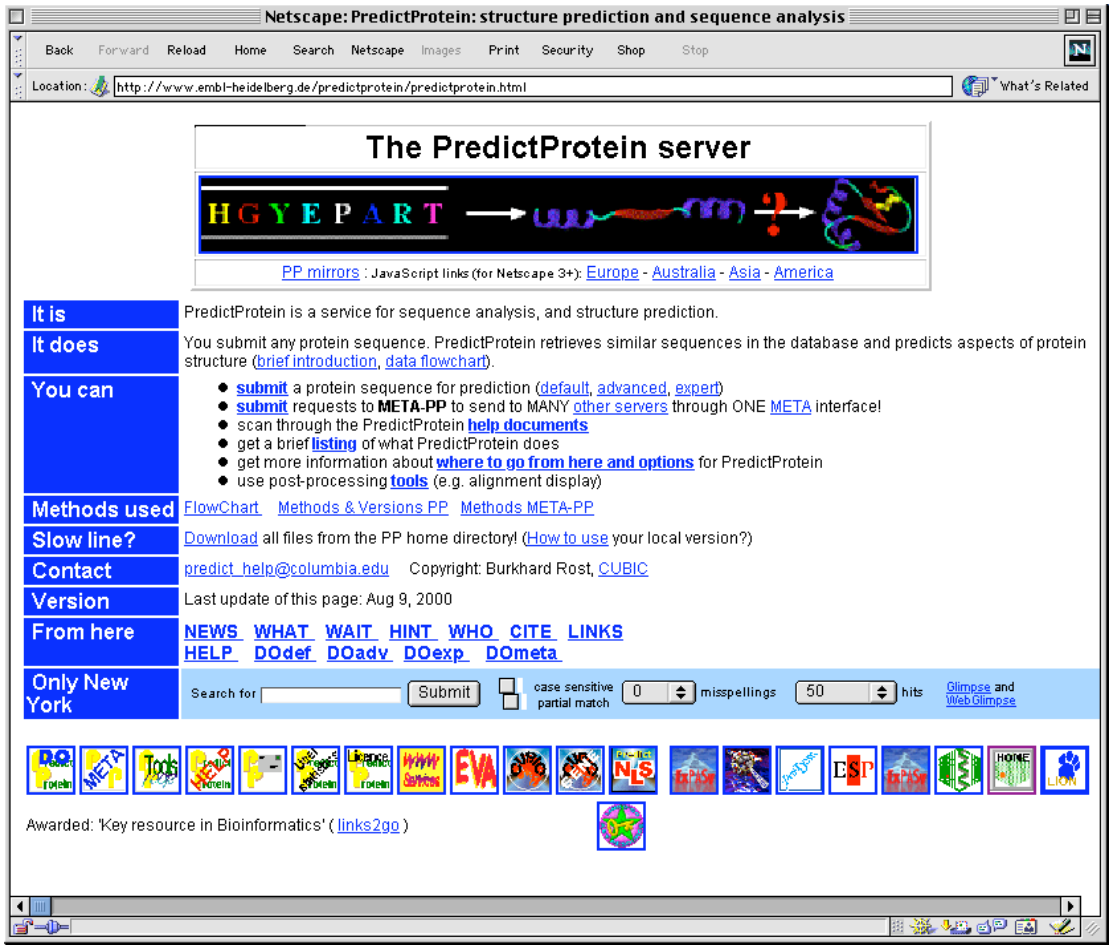
Notice that the file begins with two numbers; the first shows the number of sequences in the matrix and the second lists the length of the matrix including any gaps and ambiguities. The next section lists the names of the sequences truncated to ten characters, if necessary, along with all the sequences printed in an 'interleaved' fashion. Only the first sequence block lists the names, all others just give the sequence data itself.

Regardless of how you go from GCG format to acceptable PHYLIP format, one more technicality requires discussion. As mentioned in the Introduction, you should evaluate the terminal ends of your data matrix. If any of the implied indels are uncertain (especially true if sequence lengths were different), then question marks, "?"'s, are usually more appropriate than hyphens. Leaving them hyphens could be misleading. As discussed earlier, gaps in the data are represented by deletion symbols, "-", which is logically correct in most cases. However, gaps at the ends and beginnings of sequences probably should not have hyphens unless you really know that a deletion/insertion is responsible for the length discrepancy. Therefore, it is a good idea to edit the output from ReadSeq to replace leading and trailing hyphens in your alignment with question marks or the unknowns characters "n" or "x" depending on which is more appropriate, DNA or protein sequence respectively. This is also an excellent point at which to verify that the sequence names are exactly as you wish them to appear in final PHYLIP plots. PHYLIP sequence names can contain very limited punctuation and mixed capitalization, and can be up to ten characters in length. Be very careful with these edits so that the alignment doesn't shift out of phase.

2.18. Multiple Sequence Alignment and Structure Prediction.

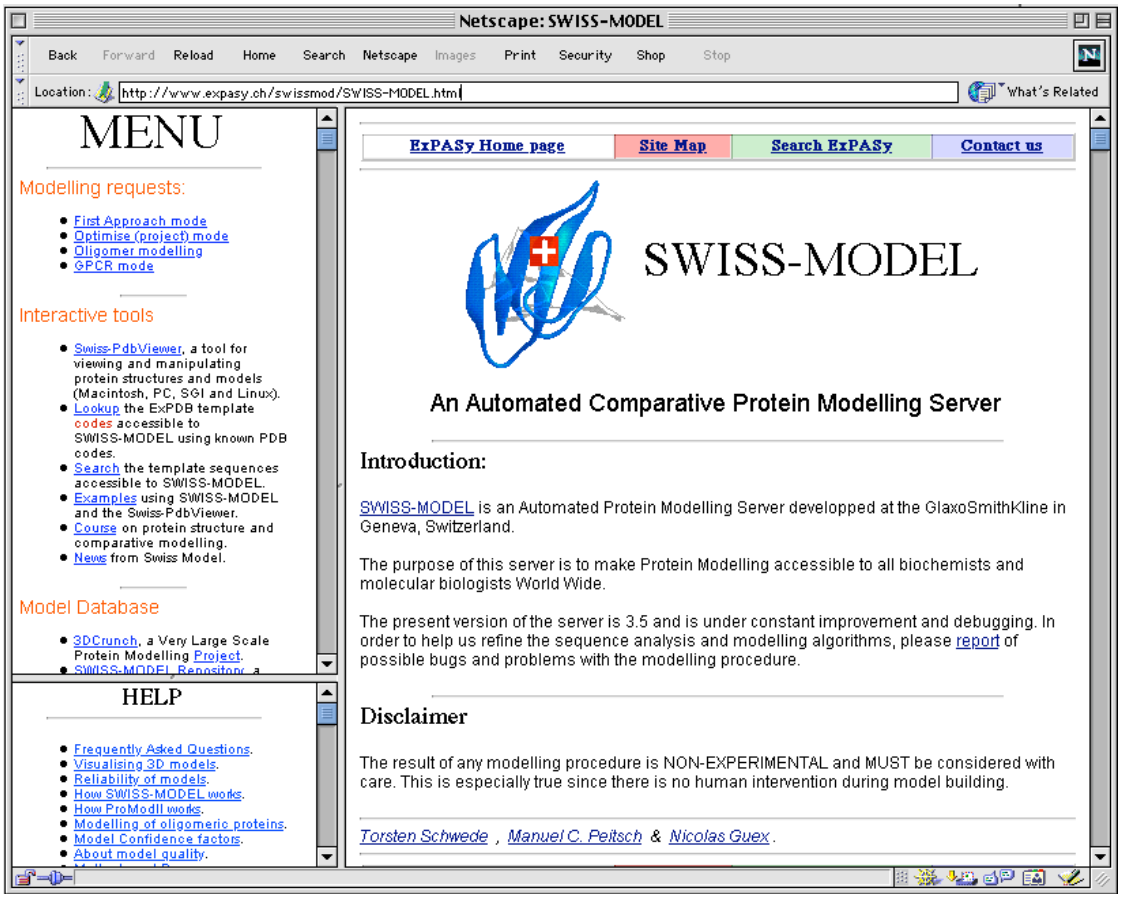
Structural inference is fraught with difficulties, as you no doubt realize. However, using comparative multiple sequence approaches is by far the most reliable strategy. In fact, in my opinion, the best predictor of secondary structure around,

available on the World Wide Web at http://www.embl-heidelberg.de/predictprotein/predictprotein.html, uses multiple sequence alignment profile techniques along with neural net technology. PredictProtein is a service offered by the Protein Design Group at the European Molecular Biology Laboratory, Heidelberg, Germany. A multiple sequence alignment is created with the MaxHom weighted dynamic programming method (Schneider, 1991) and a secondary structure prediction is produced by the profile network method (PHD). PHD is rated at an expected 70.2% average accuracy for the three states helix, strand, and loop (Rost and Sander, 1993 and 1994). Their Web page provides default, advanced, and expert submission forms. One powerful advanced and expert option is to submit your own multiple alignment. Their automated search and alignment procedure is very good, but if you've been working for months on a multiple alignment, and you know it is the best it can be, you may want to force PredictProtein to use that information, rather than it's own automated alignment. The welcome page presents a wealth of informational links:



In fact, three-dimensional modeling without crystal coordinates is even possible. This is "homology modeling." It will often lead to remarkably accurate representations if the similarity is great enough between your protein and one with an experimentally solved structure. Automated homology modeling is available through the Web as GlaxoSmithKline's SWISS-MODEL (see e.g. Guex, et al. [1999] and Guex and Peitsch [1997]) at Bairoch's ExPASy server in Switzerland (http://www.expasy.ch/swissmod/SWISS-MODEL.html). As with PredictProtein, you can submit an individual sequence and the server will perform a database search, in this case against all of the sequences from the three-dimensional Protein Data Bank, and then create a multiple alignment of the significant hits, and then finally provide a structural inference. This is "First Approach mode," or you can submit your own customized and carefully scrutinized multiple sequence alignment using "Optimise (project) mode." There are a couple of tricks to using project mode though. Naturally, your template sequences must have solved structures, however, Swiss-PdbViewer must be used to format and

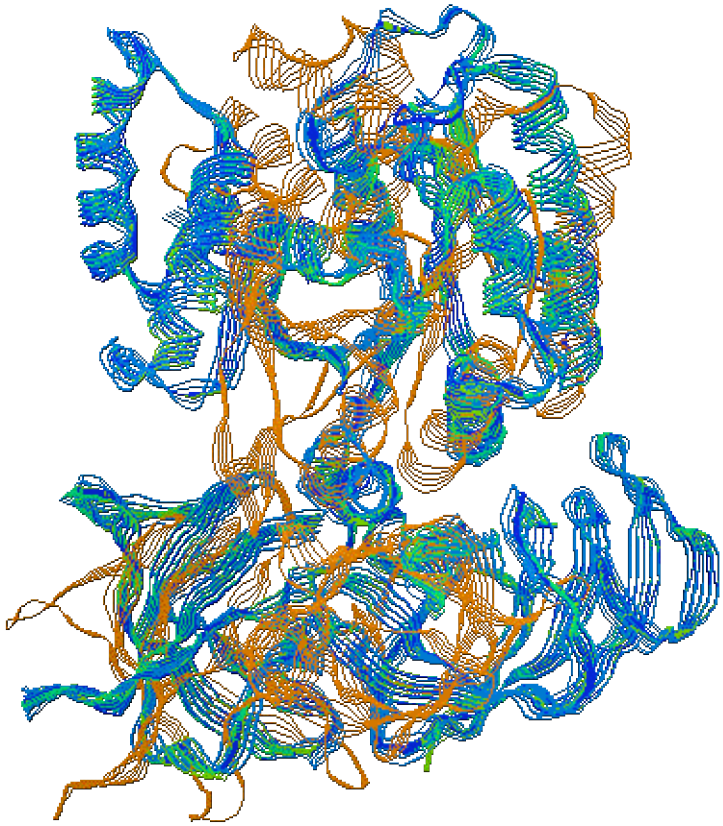
submit your data. Swiss-PdbViewer is an interactive molecular structure viewer and editor, also developed at GlaxoSmithKline, that allows superpositioning of both structures and their corresponding sequences, that you install on your own computer. It has versions for many of the major operating systems. An extensive menu and help system is provided by the SWISS-MODEL home page:



Results are returned via e-mail in one of three modes, Swiss-PdbViewer mode, normal mode, or short mode. Normal mode and short mode both return PDB format coordinates for the model, normal with a complete log file of all the server actions, short without. Swiss-PdbViewer mode returns a project file containing PDB formatted coordinates for the model and all templates superimposed, formatted for Swiss-PdbViewer, and a complete log file.

I submitted the *Giardia lamblia* Elongation Factor 1α sequence to SWISS-MODEL in "First Approach mode." The results were e-mailed back to me in less than five minutes. The figure below displays a RasMac (http://www.umass.edu/microbio/rasmol/ [see e.g. Sayle and Milner-White, 1995]) "Strands" graphic of the *Giardia* EF-1α structural model superimposed over the eight most similar chains with solved structures.

Giardia EF-1 structural model from SWISS-MODEL superimposed over eight other chains:



3. Conclusions — Do You Have Any?

The comparative method is a cornerstone of the biological sciences. Multiple sequence alignment and database searching are the comparative method on a molecular scale, and are a vital prerequisite to some of the most powerful biocomputing techniques available. Many methods are available. Understanding something about the algorithms and the program parameters of each is the only way to rationally know what is appropriate. Several methods are even available on the Internet over WWW servers. Knowing and staying well within the limitations of any particular method will avert much frustration. Oftentimes you'll need to deal with very large datasets, or you may need to manually adjust alignments. A comprehensive multiple sequence editor such as the GCG Wisconsin Package SeqLab graphical user interface can be a lifesaver in these situation.

One point that still needs to be made is that the previous techniques were performed largely using GCG's suggested defaults. This usually will work for you, but it is 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. Another vital point that can't be repeated often enough, is the dramatic importance of your multiple sequence alignments. All subsequent analyses are absolutely dependent upon them, especially phylogenetic inference. Also, if you are building multiple sequence alignments for phylogenetic inference, do not base an organism's phylogeny on just one gene. Many complicating factors can make interpretation difficult. Weird phylogenies can be the result of several things: bad alignments, insufficient data, abjectly incorrect models, saturated positions (homoplasy), compositional biases, and/or horizontal gene transfer. Use several genes — the Ribosomal Database Project (RDP) (http://rdp.cme.msu.edu/html/) provides a good, largely accepted, alignment and phylogenetic framework with which

other phylogenies can be compared. The complete RDP can be installed on a local GCG server in aligned GCG format, given sufficient interest and a cooperative GCG systems manager, which could then be used in the same manner as the sequences explored in this chapter. Otherwise desired data subsets can be downloaded from RDP and loaded into SeqLab. Anytime the orthologous phylogenies of organisms based on two different genes do not agree, there is either some type of problem with the analysis, or you have found a case of lateral transfer of genetic material. Paralogous gene phylogenies are another story altogether and should be based, if at all possible, on sequences all from the same organism.

Furthermore, keep in mind that this tutorial was written using a very similar, quite easily aligned dataset. This was done so that individuals working through the text on-line would be able to proceed in 'real time.' However, most datasets that you will encounter, especially the 'very-interesting!' ones, will not have a bunch of obvious homologues, or you'll be trying to align distantly related domains, or you'll be working on a paralogous system, or . . . . These are the situations that will present vexing alignment problems and difficult editing decisions. These are the times that you'll really have to think.

Gunnar von Heijne in his quite readable but somewhat dated treatise, *Sequence Analysis in Molecular Biology; Treasure Trove or Trivial Pursuit* (1987), provides an appropriate conclusion:

> "Think about what you're doing; use your knowledge of the molecular system involved to guide both your interpretation of results and your direction of inquiry; use as much information as possible; and do not blindly accept everything the computer offers you."

He continues:

> ". . . if any lesson is to be drawn . . . it surely is that to be able to make a useful contribution one must first and foremost be a biologist, and only second a theoretician . . . . We have to develop better algorithms, we have to find ways to cope with the massive amounts of data, and above all we have to become better biologists. But that's all it takes."

This has been a very long workshop; I apologize for that. However, database searching and multiple sequence alignment are two of the more commonly misunderstood areas in computational molecular biology and bioinformatics. There is a tremendous amount of confusion in the field and anything that can be done to try and clear up some of the mess is entirely worthwhile.

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.

Bailey, T.L. and Elkan, C., (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers, in *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, AAAI Press, Menlo Park, California, U.S.A. pp. 28–36.

Bailey, T.L. and Gribskov, M. (1998) Combining evidence using p-values: application to sequence homology searches. *Bioinformatics* **14**, 48–4.

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

Bilofsky, H.S., Burks, C., Fickett, J.W., Goad, W.B., Lewitter, F.I., Rindone, W.P., Swindell, C.D., and Tung, C.S. (1986) The GenBank™ Genetic Sequence Data Bank. *Nucleic Acids Research* **14**, 1-4.

Dayhoff, M. O., Schwartz, R. M., and Orcutt, B. C. (1979) Figure 84 in *Atlas of Protein Sequence and Structure* (Dayhoff, M. O. ed.) **5**, National Biomedical Research Foundation, Washington, D.C., U.S.A. pp. 345-352.

Dobzhansky, T., Ayala, F.J., Stebbins, G.L., and Valentine, J.W. (1977) *Evolution*. W.H. Freeman and Co. San Francisco, California, U.S.A. (The source of the original 1973 quote is obscure though it has been cited as being transcribed from the *American Biology Teacher*, March 1973, **35**, 125-129).

ECDC. The *E. coli* Database Collection. The K12 chromosome http://www.uni-giessen.de/~gx1052/ECDC/ecdc.htm Justus-Liebig-Universitaet, Giessen, Germany.

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.

Felsenstein, J. (1980--2001) PHYLIP (Phylogeny Inference Package) version 3.5+ public domain software distributed by the author. http://evolution.genetics.washington.edu/phylip.html Department of Genetics, University of Washington, Seattle, Washington, U.S.A.

Feng, D.F. and Doolittle, R. F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution* **25**, 351–360 .

Genetics Computer Group (GCG®), (Copyright 1982-2003) *Program Manual for the Wisconsin Package®*, version 10.3, http://www.accelrys.com/products/gcg_wisconsin_package/index.html Accelrys, a wholly owned subsidiary of Pharmacopeia Inc., San Diego, California, U.S.A.

Gilbert, D.G. (1993 [C release] and 1999 [Java release]) ReadSeq, public domain software distributed by the author. http://iubio.bio.indiana.edu/soft/molbio/readseq/ Bioinformatics Group, Biology Department, Indiana University, Bloomington, Indiana, U.S.A.

Gonnet, G.H., Cohen, M.A., and Benner, S.A. (1992) Exhaustive matching of the entire protein sequence database. *Science* **256**, 1443–1145.

Gribskov M., McLachlan M., Eisenberg D. (1987) Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Sciences U.S.A.* **84**, 4355–4358.

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

Gribskov, M., Luethy, R., and Eisenberg, D. (1989). Profile analysis, in *Methods in Enzymology* **183**, Academic Press, San Diego, California, U.S.A. pp. 146–159.

Guex, N. and Peitsch, M.C. (1997) SWISS-MODEL and the Swiss-PdbViewer: an environment for comparative protein modeling. *Electrophoresis* **18**, 2714–2723.

Guex, N., Diemand, A., and Peitsch, M.C. (1999) Protein modelling for all. *Trends in Biochemical Sciences* **24**, 364–367.

Gupta, S.K., Kececioglu, J., and Schaffer, A.A. (1995) Making the Shortest-Paths Approach to Sum-of-Pairs Multiple Sequence Alignment More Space Efficient in Practice, Proceedings of the Sixth Annual Symposium on Combinatorial Pattern Matching (CPM '95).

Gupta, S.K., Kececioglu, J.D., and Schaffer, A.A. (1995) Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *Journal of Computational Biology* **2**, 459–472.

Hasegawa, M., Hashimoto, T., Adachi, J., Iwabe, N., and Miyata, T. (1993) Early branchings in the evolution of Eukaryotes: ancient divergence of Entamoeba that lacks mitochondria revealed by protein sequence data. *Journal of Molecular Evolution* **36**, 380–388.

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.

Higgins, D.G., Bleasby, A.J., and Fuchs, R. (1992) CLUSTALV: improved software for multiple sequence alignment. *Computer Applications in the Biological Sciences* **8**, 189--191.

Iwabe, N., Kuma, E.-I., Hasegawa, M., Osawa, S., and Miyata, T. (1989) Evolutionary relationship of Archaebacteria, Eubacteria, and Eukaryotes inferred from phylogenetic trees of duplicated genes. *Proceedings of the National Academy of Sciences, U.S.A.* **86**, 9355–9359.

Madsen, H.O. Poulsen, K., Dahl, O., Clark, B.F., and Hjorth, J.P. (1990) Retropseudogenes constitute the major part of the human elongation factor 1 alpha gene family. *Nucleic Acids Research* **18**, 1513–1516.

National Center for Biotechnology Information (NCBI) *Entrez*, public domain software distributed by the authors. http://www.ncbi.nlm.nih.gov/Entrez/ National Library of Medicine, National Institutes of Health, Bethesda, Maryland, U.S.A.

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.

Online Mendelian Inheritance in Man, OMIM ™. (1996) http://www.ncbi.nlm.nih.gov/omim/ Center for Medical Genetics, Johns Hopkins University, Baltimore, Maryland, U.S.A. and National Center for Biotechnology Information, National Library of Medicine, Bethesda, Maryland, U.S.A.

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.

Rivera, M.C. and Lake, J.A. (1992) Evidence that Eukaryotes and Eocyte Prokaryotes Are immediate relatives. *Science* **257**, 74–76.

Rost, B. and Sander, C. (1993) Prediction of protein secondary structure at better than 70% accuracy. *Journal of Molecular Biology* **232**, 584–599.

Rost, B. and Sander, C. (1994) Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins* **19**, 55–77.

Sander, C. and Schneider, R. (1991) Database of homology-derived structures and the structural meaning of sequence alignment. *Proteins* **9**, 56–68.

Sayle, R.A. and Milner-White, E.J. (1995) RasMol: biomolecular graphics for all. *Trends in Biochemical Sciences* **20**, 374–376.

Schwartz, R.M. and Dayhoff, M.O. (1979) Matrices for detecting distant relationships, in *Atlas of Protein Sequences and Structure* (Dayhoff, M.O. ed.) **5**, National Biomedical Research Foundation, Washington, D.C., U.S.A. pp. 353–358.

Smith, R.F. and Smith, T.F. (1992) Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for comparative protein modelling. *Protein Engineering* **5**, 35–41.

Smith, R.F., Wiese, B.A., Wojzynski, M.K., Davison, D.B., Worley, K.C. (1996) BCM Search Launcher--an integrated interface to molecular biology data base search and analysis services available on the World Wide Web. *Genome Research* **6**, 454–62.

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.

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

Sogin, M.L., Morrison, H.G., Hinkle, G., and Silberman, J.D. (1996) Ancestral relationships of the major eukaryotic lineages. *Microbiolgia Sem* **12**, 17–28.

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.

Swofford, D.L., PAUP* (Phylogenetic Analysis Using Parsimony and other methods) version 4.0+ (1989–2002) Florida State University, Tallahassee, Florida, U.S.A. http://paup.csit.fsu.edu/ distributed through Sinaeur Associates, Inc. http://www.sinauer.com/ Sunderland, Massachusetts, U.S.A.

Thompson, J.D., Gibson, T.J., Plewniak, F., Jeanmougin, F. and Higgins,D.G. (1997) The ClustalX windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Research* **24**, 4876–4882.

Thompson, J.D., Higgins, D.G. and Gibson, T.J. (1994) CLUSTALW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Research* **22**, 4673–4680.

von Heijne, G. (1987) *Sequence Analysis in Molecular Biology; Treasure Trove or Trivial Pursuit.* Academic Press, Inc., San Diego, California, U.S.A.

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.