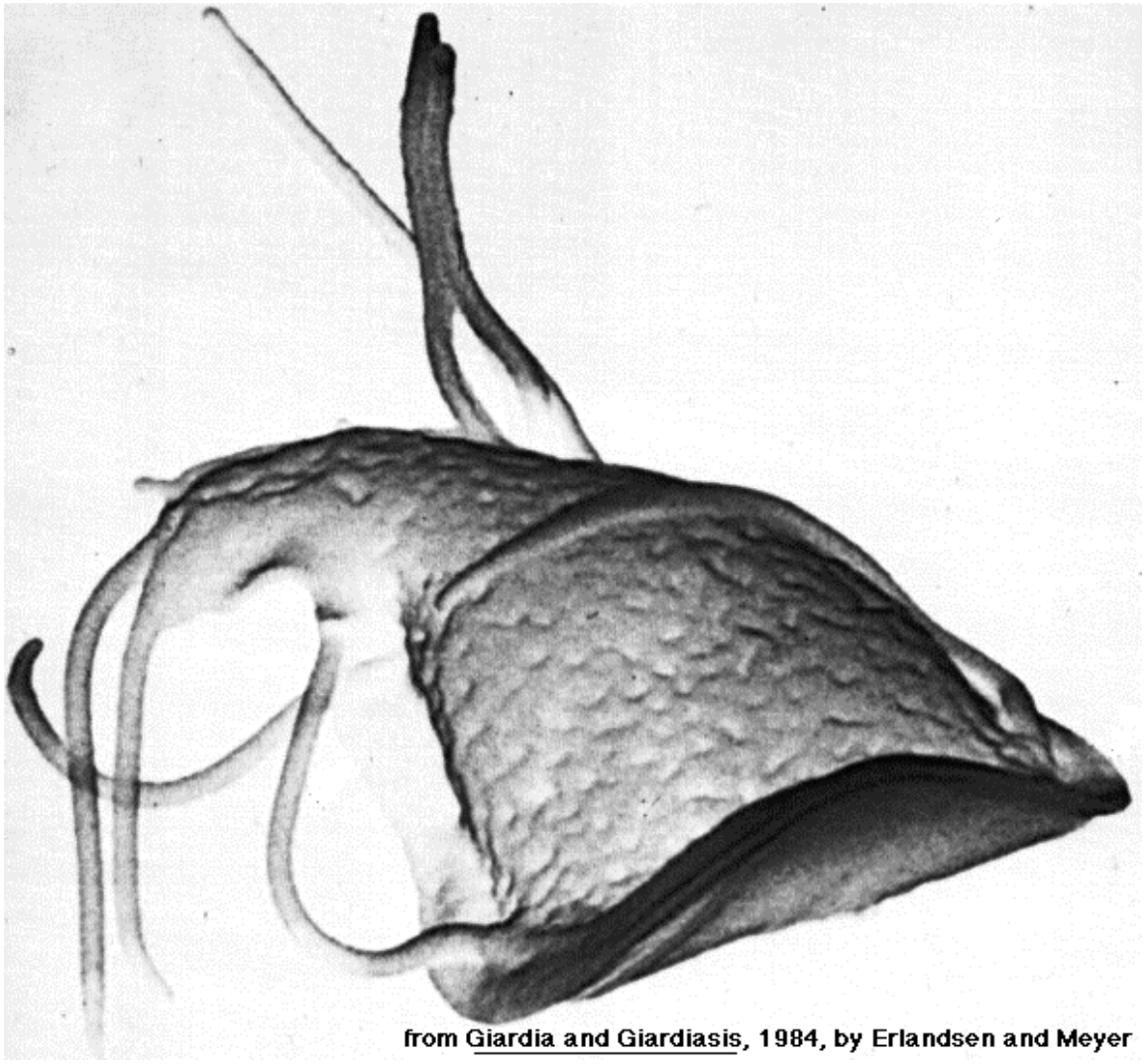
Gene Discovery and Analysis, and Multiple Sequence Alignment: Contig3000, a 5.4 Kb Stretch of the *Giardia lamblia* Genome.



from Giardia and Giardiasis, 1984, by Erlandsen and Meyer

July, 1999; a GCG¥ Wisconsin Package™ SeqLab® Workshop for the Laboratory of Dr. Mitchell L. Sogin, Josephine Bay Paul Center for Comparative Molecular Biology and Evolution; at the Woods Hole Marine Biological Laboratory, by BioInfo 4U.

Author & Instructor: Steven M. Thompson

Steve Thompson
BioInfo 4U (not associated with GCG)
2538 Winnwood Circle
Valdosta, GA, USA 31601-7953
stevet@bio.fsu.edu
912-249-9751 or 912-249-9163

¥GCG is the Genetics Computer Group, an Ofxord Molecular Company, producer of the Wisconsin Package for sequence analysis.

Introduction

Contig3000 is a 5368 base pair consensus sequence from a 35 fragment contig of *Giardia lamblia* genomic DNA. It was generated in the laboratory of Dr. Mitchell L. Sogin of the Josephine Bay Paul Center for Comparative Molecular Biology and Evolution at the Woods Hole Marine Biological Laboratory. See their World Wide Web pages at http://www.mbl.edu/Giardia for a general description, cloning strategies, and preliminary fragment data from their *Giardia* genome sequencing project. Dr. Andrew G. McArthur of that laboratory kindly supplied contig3000 and the following reference regarding it:

Contig3000 comes from the June 9th, 1999 assembly. This contig is an assembly of 35 genomic shotgun sequences from the B, D, F, G, I, and J libraries. However, I is a resized subset of G, and J a resized subset of F. Thus, it is derived from four contributing libraries:

B from LambdaZapII (3-5 Kbp EcoRI digestion inserts)
D from pBluescriptIIKS- (2-6 KBp Tsp509I digestion inserts)
F from pUC18 (2.5-3 Kbp sheared inserts)
G from pUC18 (3-5 Kbp sheared inserts)

Also note that while 35 sequences make up this contig, many are from the use of different primers or multiple attempts on the same DNA. In total the 35 sequences are from 10 independent pieces of genomic DNA. Normally the Sogin lab *Giardia* contigs do not contain such high redundancy, however, it is a good example of a genomic sequence to be used in this tutorial.

Where to begin with a brand new sequence is daunting a problem. The impulse to quickly send it off for a fast-and-dirty network BLAST search is undeniable, and OK, just don't rely too heavily on it. It is merely the beginning of a long analysis process.

One fortuitous observation regarding the *Giardia* genome is introns have never been discovered. This makes the gene discovery mission much less complicated. Introns and exons really make matters difficult. So what is the general plan of attack after generating a big contig such as contig3000? The following outline suggests one good approach:

Gene Finding Strategies, After the Sequencing's Done, What's Next?

Given the nucleotide sequence of a biological molecule, what can we know about that molecule? How are coding sequences recognized in genomic DNA? Understanding the concepts and differentiating between the approaches:

I. Search for signals — transcriptional and translational regulatory sites (and, when relevant, exon/intron splice sites);

FindPatterns and FitConsensus. And simple translation tools.

II. Search by content — 'nonrandomness' and codon usage;

TestCode, Frames, and CodonPreference.

III. Search through the databases for sequence similarity and thereby infer gene location through homology — what's available, the methods and the algorithms — motifs, hashing techniques and heuristics, dot matrix analysis, and significance;

Motifs, the BLAST and FastA families, Compare and DotPlot, and Gap, BestFit, and FrameAlign. How to make sense of them all.

IV. Annotate your sequence in order to see 'how it all comes together.' The combinatorial approach.

V. And finally — multiple sequence alignments. What good are they? They are . . .

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

SeqLab is a part of the Genetics Computer Group's (GCG) Wisconsin Package. This comprehensive package of sequence analysis programs is used world-wide. It has become the global standard in sequence analysis software. The Wisconsin Package provides a comprehensive toolkit of over 130 integrated DNA and protein analysis programs, from database, pattern, and motif searching; fragment assembly; mapping and sequence comparison to gene finding; protein and evolutionary analysis; primer selection; and DNA and RNA secondary structure prediction. The SeqLab X-windows based Graphical User Interface (GUI) is a 'front-end' to the package. It provides an intuitive alternative to the much dreaded UNIX or VMS command line by allowing menu-driven access to most of GCG's programs. It is based on Steve Smith's (1994) GDE (the Genetic Data Environment) and makes running the Wisconsin Package much easier by providing a common editing interface from which most programs can be launched. This introductory workshop will expose you to many of SeqLab's multitude of features, just the 'tip-of-the-iceberg,' hopefully whetting your appetite enough to make you want to learn it even further and to use it for all of your sequence analysis tasks. Once you gain an appreciation for its power and ease of use, I don't think you'll be satisfied with any other sequence analysis system.

Current genome projects are generating billions of base pairs of data at an explosive rate. GenBank has doubled in size about every 14 months since 1982 and now contains more than four million sequences. Given all this unknown DNA, how are encoded genes determined and positioned? Translating from all translational start codons to all 'nonsense' chain terminating, stop codons in every frame provides a list of ORF's (Open Reading Frames), but which of them, if any, actually code for proteins? And if you are dealing with eukaryotic genomic DNA, then exons and introns often considerably complicate the matter. Three general approaches to the gene finding problem can be imagined: 1) all genes have certain regulatory signals positioned in or about them, 2) all genes by definition contain specific code patterns, and 3) many genes have already been sequenced in other organisms so we can infer through homology if our new sequence is similar to an existing sequence. We can utilize all of these facts to help locate the position of genes in DNA. These methods are often known as "searching by signal," "searching by content," and "homology inference" respectively. Although no method is absolutely reliable, one seldom has the luxury of knowing the complete amino acid sequence to the protein of interest and simply translating DNA until the correct pieces fall out. This is the only method that would be 100% positive. Since we are usually forced to discover just where these pieces are, especially with genomic DNA, computerized analysis becomes invaluable.

Signal searches look for transcriptional and translational features. Transcriptional regulatory sites such as promoters and other transcription factor and enhancer binding sequences can help identify the beginnings of genes, however, some of these motifs can be quite distant from the actual start of transcription. The prokaryote Shine-Dalgarno consensus and other eukaryote ribosome binding sites obviously relate to translation initiation, as does the methionine start codon. However, matters can be complicated by alternative

start codons; AUG works in about 90% of cases, but there are exceptions in some prokaryotes and organellar genomes. Transcriptional terminator and attenuator sequences can help identify gene ends as do the termination (stop) codons, but, again, exceptions can be found, especially in some ciliated protists and due to eukaryote suppresser tRNA's. Furthermore, splice site donor-acceptor consensus sequences can point to intron-exon borders. All of these types of signals can help us recognize coding sequences. A major problem is simple consensus pattern type searching is often either overly or insufficiently stringent because of the variable and loosely defined nature of these types of sites. Weight matrix approaches are more powerful, but not nearly as simple to set up in most commercial sequence analysis packages.

There are two general strategies for finding coding regions of DNA based on the content of the DNA itself — one is based on the local 'nonrandomness' of a stretch and the other is based on the known codon usage of a particular life form. The first, the nonrandomness test, does not tell us anything about the particular strand or reading frame; however, it does not require a previously built codon usage table. The codon usage approach is based on the fact that different types of organisms use different frequencies of codons to code for particular amino acids. This approach requires a codon usage table built up from known translations; however, it also tells us the strand and reading frame as opposed to the former.

To use content approaches based on codon usage, you must decide which codon usage table to specify. By default GCG codon usage programs will use a frequency table designed from highly expressed *E. coli* genes. Therefore, if you're working with an *E. coli* gene, the program's default is appropriate. However, if your protein comes from anything else, you will want to use an alternate table. GCG provides alternate data files in a public data library with the GCG logical name GenMoreData. The available tables, in addition to the default codon usage table, ecohigh.cod, are: celegans_high.cod, celegans_low.cod, drosophila_high.cod, human_high.cod, maize_high.cod, and yeast_high.cod. Even more tables are available at various molecular biology data servers such as IUBIO (http://iubio.bio.indiana.edu/soft/molbio/codon/). The TRANSTERM database at the European Bioinformatics Institute (ftp://ftp.ebi.ac.uk/pub/databases/transterm/) also contains several and an especially good selection derived from a recent GenBank version comes from the CUTG database (http://www.dna.affrc.go.jp/~nakamura/codon.html) available in GCG format through various SRS servers (e.g. see http://www.sanger.ac.uk/srs5bin/cgi-bin/wgetz?-fun+pagelibinfo+-info+CUTG). Furthermore, if you are not satisfied with any of the available options, GCG has a program, CodonFrequency, that enables you to create your own codon frequency table from known coding sequences.

Both content approaches, nonrandomness and codon usage, rely on implicit biological constraints imposed on the genetic code, constraints which we can utilize to help discriminate structural genes from all the rest of the, some would incorrectly say, 'junk' DNA found in most genomes. The content approach is often more accurate; it does not generate nearly as many false positives as signal type searches, but its answers aren't concise either. Starting and stopping points are never exactly delineated.

What about comparisons with other sequences? What do database searches and pairwise comparisons tell us and what can we gain from them? Why even bother? Can we learn about one molecule by comparing it to another? Yes, naturally we can; inference through homology is one of the fundamental principles of biology. It can often be the most powerful method, especially now that so many sequences have been collected and analyzed. But it too can be misleading and seldom gives exact start and stop positions. However, if portions of our new sequence fall into a preexisting group then we can gain knowledge of its location within the overall sequence, its function, and possibly even its structure. Database searches can provide valuable insights into enzymatic mechanism and even evolution. Are there any 'families' that parts of your newly discovered sequence fall 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.

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

The mechanics of dynamic programming are beyond the scope of the present exercise; however, I encourage you to read some of the classic papers and modern revisions on the method. Needleman and Wunsch (1970) first described the method pertaining to biological sequences as a global solution. Later refinements (Smith and Waterman, 1981) demonstrated how dynamic programming can also be used to find optimal local alignments. In considering dynamic programming, always remember a very important point. Just because dynamic programming is guaranteed to find an optimal alignment, it is not necessarily the only optimal alignment. Furthermore, the optimal alignment is not necessarily the 'right' or biologically relevant alignment. As always, question the results of any computerized solution based on what you know about the biology of the system.

A further complication occurs in protein sequences. Certain amino acids are very much alike, structurally, chemically and genetically. How can we take advantage of the similarity of amino acids in our alignments? People have been struggling with this problem since the late 1960's. Margaret Dayhoff (Schwartz and Dayhoff, 1979) unambiguously aligned closely related datasets (no more than 15% difference) available at that point in time and noticed that certain residues, if they mutate at all, are prone to change into certain other residues. As it works out, these propensities for change fell into the same categories that chemists had known for years — those same chemical and structural classes mentioned above, conserved through the evolutionary contraints of natural selection. However, Dayhoff's empirical observation quantified these changes. Based on the 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 specify the probabilities at which different residues mutated into other residues through evolutionary history. This is the basis of the famous PAM (corrupted acronym of accepted point mutation) 250 (meaning that the matrix has been multiplied by itself 250 times) table. Since Dayhoff's time other biomathematicians (e.g. see Henikoff and Henikoff's [1992] BLOSUM series of tables) have created newer tables with more or less success than Dayhoff's original but the concept remains the same and Dayhoff's original PAM 250 table remains the classic as historically the most widely used one. Collectively these types of tables are known as symbol comparison tables or scoring matrices and they are fundamental to all sequence comparison techniques.

Once these problems are understood we can screen the database to look for sequences to compare ours to. However, classic dynamic programming techniques take far too long to be used against an entire database with a 'normal' computer. Therefore, most database searching programs use the concepts discussed above; however, they 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 query sequence into small 'words' or 'ktuples' of a set size and creating a 'look-up' table with those words keyed to numbers. Then when any of the words match part of an entry in the database, that match is saved. In general, hashing reduces the complexity of the search problem from N^2 for dynamic programming to N, the length of all the sequences in the database. Approximation techniques are collectively known as 'heuristics.' Webster's defines heuristic as "serving to guide, discover, or reveal; . . . but unproved or incapable of proof." In database searching 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.

Most database searches work best when submitted as a batch or background process. This is because of the size of the databases. In spite of the fast hashing 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 at work comparing your query to that many sequences, therefore, take advantage of batch capablities. All of the GCG database searches accept a really handy automatic batch submission option to make this very easy.

An exception to the standard 'submit the search and wait' style of most database searching is the network BLAST program. This program uses an extremely fast heuristic statistical hashing algorithm on a large parallel computer located at the National Center for Biotechnological Information. Typical searches run in just a few minutes, after you get through the waiting queue; however, realize the limitation of the BLAST algorithm of not being optimized for nucleic acid sequences. It works best on protein queries and is usually not recommended as a tool for comparing DNA queries to DNA databases unless both sequences are 'translated on the fly' as in TBLASTX.

All database searching is far more sensitive at the amino acid level than at the DNA level because proteins have twenty match criteria versus DNA's four. This drastically cuts down the 'noise' level of the search. Therefore, whenever dealing with coding sequence, it is always prudent to search at the protein level. Even though protein searching is more sensitive, the DNA databases are much larger. This drawback has been partly overcome with programs which take a protein query and compare it to translated nucleotide databases, but one still needs to know if the translation is 'real.' So, 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 database you choose.

A big question and a very common mistake that is made in this whole area of searching and alignment is the concept of homology versus similarity: There is a huge difference! Similarity is merely a statistical parameter which describes how much two sequences, or portions of them, are alike according to some set scoring criteria. Homology, by definition, implies an evolutionary relationship — more than just the fact that we've all evolved from the same old 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, morphological, genetic, or fossil, that corroborates your assertion. There is really no such thing as percent homology; something is either homologous or it is not. Dr. 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 are not." Do not make the mistake of calling any old sequence similarity homology, it will get you in trouble with a lot of scientists, especially the evolutionists of the world, though it is a particularly common misnomer.

How do you tell if a similarity is significant or is truly homologous? Many of the programs generate percent similarity scores, however these really don't mean a whole lot. The 'quality' score means a lot more but it is hard to interpret. They are only relevant within the context of a particular comparison or search. At least they take the length of similarity, all of the necessary gaps introduced, and the matching of symbols all into account. 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 are always confusing but the descriptions do help. Some of the programs can generate histograms of score distributions, but again, they can be confusing.

One way of deciding significance is to take advantage of the Monte Carlo randomizations option available in the two dynamic programming comparison programs BestFit and Gap. To utilize this strategy, pull the sequence you wish to evaluate from your search output list and compare it to your query 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 to each, respectively, and specify the Randomizations=100 option. This option jumbles the second sequence of the comparison 100 times and then generates scores and a standard deviation based on the jumbled matches. You can then compare the random scores to the unjumbled score using a "Z score" calculation to help decide significance. The FastA (Pearson and Lipman, 1988), BLAST (Altschul, et al., 1990), and ProfileSearch (Gribskov, et al., 1987) algorithms use a similar approach but base their statistics on the distance from the distribution of all the rest, 'insignificantly similar,' members of the database being searched. They all generate Expectation or Probalitilty statistics based on this distribution in which the closer the number is to zero, the more probable it is that the discovered similarity is not due to chance.

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 system that you specify, a dot is generated. Dot matrix analysis can point out areas of similarity between two sequences that all other methods might miss. This is because most other methods align either the overall length of two sequences or just the 'best' parts of each to achieve one optimal alignment. Dot matrix methods enable the operator to visualize the entirety of both sequences; if you will, it allows the 'Gestalt' of the alignment to be seen. 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. Using this method correctly, you can identify areas within sequences that happen to have significant matches that no other method would ever notice. Again, this is a method that you would perform after running initial searches as it only compares one sequence against another, not the entire database.

One small database that you should screen just as a matter of course is PROSITES. The GCG program Motifs performs this search. Motifs searches for recognized structural, regulatory and enzymatic consensus sequences in the *PROSITE Dictionary of Protein Sites and Patterns* (Bairoch, 1992). This approach is wonderful for trying to ascertain function in an unidentified peptide sequence, but keep in mind the inherent problems of consensus style searches discussed above in signal searching. The program can tolerate mismatches with a mismatch option and it displays an abstract with selected references for each motif signature found. Another small database that should not be ignored is NRL_3D. This database contains all

the sequences from the three-dimensional coordinate database PDB and, thus, can serve as a link between structural and sequence based methods.

Several powerful E-mail and World Wide Web based InterNet servers have also been established that can help with these types of analyses. Many combine several of the methods discussed above. In particular some have been designed to use neural net and artificial intelligence approaches to help in the 'decision' process. A careful application and interpretation of the many resources at one's disposal can go a long way to increasing our understanding of gene structure and function. But, as always, carry a healthy dose of skepticism to, and be extremely wary, at any session with the terminal, as the naive can easily be misled into accepting inappropriate or downright wrong results. Regardless of your approach, all available methods should be used together to help reinforce and/or reject the others' findings. The chore of identifying coding sequences is far from trivial and is a long way from being solved in an unambiguous manner; however, it is extremely important anytime anyone starts sequencing genomic DNA and doesn't have the luxury of an available cDNA library.

Once you have identified where your coding regions lay, then you can progress to the next logical step. That is, prepare a multiple sequence alignment of the region against the homologs discovered by your analysis. The power and sensitivity of sequence based computational methods dramatically increases with the addition of more data. As in pair-wise comparisons, those areas most resistant to change are functionally the most important to the molecule. However, with increased dataset size, the patterns of conservation become evermore clear. But how does one work with more than just two sequences at a time? 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 as computational needs increase with the exponent of the dataset size (complexity=[sequence length]$^{\text{number of sequences}}$). Mathematically this is an N-dimensional matrix, quite complex indeed. One program, MSA (version 2.0, 1995), does attempt to globally solve this equation, however, the algorithm's complexity precludes its use in most situations.

Several heuristics have been employed over the years to simplify the complexity of the problem. 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 (version 1.4, 1995). The most commonly used approach to the problem is known as the pairwise, progressive dynamic programming solution. 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 two sequences. The pairwise, progressive solution is implemented in several programs including Des Higgins' ClustalW (1994) and the GCG program PileUp. Both programs insert gaps to align the full length of a sequence set to produce a multiple sequence alignment.

One of the more difficult aspects of multiple alignment is knowing what sequences you should attempt it with. Be sure that your list of homologs discovered in the previous gene finding analysis is restricted to only those sequences that actually should be aligned. Beware the 'apples and oranges' problem. Make sure that the group of sequences that you align are in fact related, that they actually belong to the same gene family, and that the alignment is meaningful. An alignment is a statement of homology — be sure that it makes sense. Either make paralogous (i.e. evolution via gene duplication) comparisons to ascertain gene phylogenies, or orthologous (within one ancestral loci) comparisons to ascertain organismal phylogenies; try not to mix them up without complete data representation. Lots of confusion and extremely misleading interpretations can

result otherwise. Also be wary of trying to align genomic sequences with cDNA when working with DNA; the introns will cause all sorts of headaches. Similarly, don't align mature and precursor proteins from the same organism and loci. It doesn't 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.

As in pair-wise alignment and sequence database searching, all of this stuff is much easier with protein sequences versus nucleotides. Twenty symbols are just much easier to align then only four; the signal to noise ratio is so much better. If you are forced to align nucleotides the whole process becomes much more difficult. Therefore, as it is in database searching, translate nucleotide sequences to their protein counterparts if you are dealing with coding sequences before performing further analyses including multiple sequence alignment. If one is required to align nucleotides because the region does not code for a protein, then 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.

Another powerful approach that should be utilized if at all possible is the Profile suite (Gribskov, et al., 1987). This strategy works best when one has prepared and refined a multiple sequence alignment of significantly similar sequences or regions within sequences. Profile searching involves forming a 'profile' from an alignment of related sequences and then searching the databases with that profile. Profile searching is tremendously powerful and should be pursued whenever possible. It can provide the most sensitive, albeit extremely computationally intensive, database similarity search possible. A very appropriate strategy is to find similar genes to a newly sequenced gene using traditional database searching techniques and then align all of the significantly similar proteins or protein domains. The aligned sequences can then be run through the Profile package to generate a profile of the family. Often Profile analysis can show features not obvious to individual members. A distinct advantage is in further manipulations and database searches, evolutionary issues are considered by virtue of the Profile algorithms. Gaps are penalized more heavily in conserved areas than they are in variable regions and the more highly conserved a residue is, the more important it becomes. Furthermore, any generated consensus sequences are not based merely on the positional frequency of particular residues but rather utilize the evolutionary conservation of substitutions based on the amino acid substitution 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 in the design of hybridization and PCR probes for unknown sequences where data is available in related species.

We can visualize these areas of an alignment that profile searching puts the most emphasis on. They are the most conserved areas of an alignment, and thus functionally the most important. Realize that in addition to the primary sequence conservation seen in these regions, structure and function is also conserved. We will use SeqLab's built in color functions and the GCG program PlotSimilarity to help visualize these crucial regions within our alignment. PlotSimilarity can be used to ascertain alignment quality by showing which portions of an alignment are conserved, by indicating the overall average similarity, and by noting the changes in these estimates as an alignment is adjusted. Furthermore, PlotSimilarity is a very helpful assistant in probe design by allowing you to visualize the most important, conserved regions of an alignment. It is invaluable for designing phylogenetic specific probes as it clearly localizes areas of high conservation and 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

conservation in phylogenetic category subset datasets to differentiate between universal and specific potential probe sequences. One can then use various primer discovery programs such as the GCG program Prime to further localize and test potential probes for common PCR conditions and problems.

Finally, we can use multiple sequence alignments to infer phylogeny. A multiple sequence alignment is itself a hypothesis about evolutionary history. Based on the explicit assertion of homologous positions in an alignment several algorithms available can estimate the most reasonable evolutionary tree for that alignment. Therefore, devote considerable time and energy toward developing the most satisfying multiple sequence alignment possible. Quality alignments mean everything for obtaining meaningful results from phylogenetic inference algorithms. All of the molecular sequence phylogenetic inference programs make the validity of your input alignment their first and most critical assumption. Be sure that the alignment makes biological sense. Use all available information and understanding to insure that your alignment is as good as it can be. Make sure that known enzymatic, regulatory, and structural elements all align, for the results of your inference are absolutely dependent upon your alignment. To help assure the reliability of any alignment always use comparative approaches. Look for conserved structural and functional sites to help guide your judgment. In ribosomal RNA alignments researchers have successfully used the conservation of covarying sites to assist in this process. 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 formerly at the University of Illinois, Urbana Campus, but now housed at the Center for Microbial Ecology at Michigan State University to help guide the construction of their rRNA alignments and structures (http://www.cme.msu.edu/RDP/). Use everything available to insure that you have prepared a satisfying alignment. Remember the old adage: "garbage in — garbage out!"

One of the biggest problems in biocomputing is that of sequence format. Each suite of programs requires a different sequence format. GCG sequence format exists both as single and Multiple Sequence Format (MSF) and 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. Several different programs are available to allow us 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, allows for the back and forth conversion between several different formats. The PAUP* interfaces in the GCG system, PAUPSearch and PAUPDisplay, automatically generate their required NEXUS format directly from the GCG formatted files, so this is not nearly as much of a hassle. Alignment gaps are another problem. Different program suites may use different symbols to represent them. Hyphens (dashes), "-"'s, are used by most sequence analysis programs to represent gaps, but GCG alignment programs insert periods, "."'s, to represent gaps in the alignment and tildes, "~"'s, to show uneven end lengths. However, periods mean "the same symbol as the above sequence" to PHYLIP and it doesn't regognize the tildes at all. 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 called 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's" to all program packages. This will assure that the programs do not make incorrect assumptions about your sequences.

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 which assuredly do align. If any portions of the alignment are in doubt, throw them out. This usually means trimming down the alignment's terminal ends and may require internal trimming as well. SeqLab makes this process much easier than previous means. Another possibility is to exclude portions with SeqLab's Mask option. This 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.

The Tutorial: A 'Real-Life' Project Oriented Approach.

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 percent symbol, "`%`" indicates the system prompt and should not be typed as a part of commands. Really important statements may be underlined.

The Wisconsin Package only runs on either UNIX or VAX/VMS operating system computers. Specialized "X-server" graphics communications software is required to use GCG's SeqLab interface. This needs to be installed separately on personal style 'Wintel' or Macintosh machines but comes standard with most UNIX operating systems. The details of X and of connecting to your GCG server will not be covered in this exercise. If you are unsure of these procedures ask for assistance in the computer laboratory. I am also available for individualized help; just contact me at stevet@bio.fsu.edu. A couple of tips at this point should be mentioned though. Rather than holding mouse buttons down to activate them, while using X software, just click on items. And 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.

1) Log onto your GCG account and launch SeqLab.

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. Get my assistance for this step if you are unsure of yourself. There are too many variations in method for them all to be described here. A terminal window should appear on the desktop after a few moments.

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

Issue the "**fetch**" command to transfer contig3000 from the GCG public data files where I put it for the purpose of this tutorial. It has already been converted to GCG format. The command line follows:

```
% fetch contig3000.seq
```

Use the UNIX "**more**" utility to scroll through the file a page at a time so that you can see what GCG single sequence format looks like. Press the <**space bar**> not the <return key> to move from one page to the next; press the <**q**> key at any point to quit more.

```
% more contig3000.seq

!!NA_SEQUENCE 1.0
Contig3000


contig3000.seq  Length: 5368  June 24, 1999 15:19  Type: N  Check: 9121  ..

       1  CAGAACATGT GTGCCCGTGT TGTAGACCTT CCTATTGTTC ACTGGGTGGT

      51  GCACTTCGAC TGTCCAGATG GTGTGATCAC CTACGCACAC AGAGCAGGTC

     101  GTGCAGCAAG AATGAACCTC CCTGGCTTCT CACTTCTATT CCTAACAGAT

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

    5251  TTCCAGATTA TCCGTTGCGG ACAGTACATT ATCCATTCGC TCTCTGTGCT

    5301  CGGTAAGGCC CTTCTTGTAG CTTTGAAGAG ACACCTCCAG ACATGCTATC

    5351  TGAGCCTTCA GCATGCAG
```

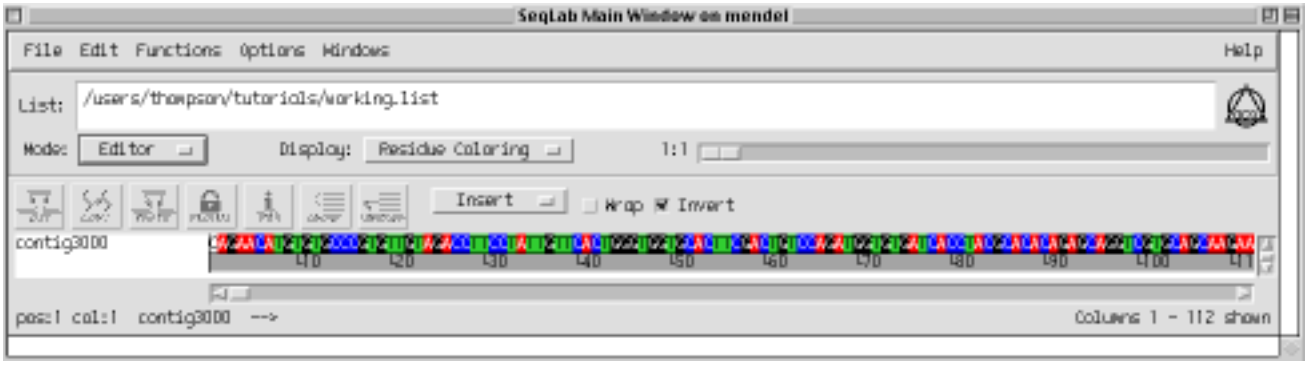
OK, now for something completely different; issue the command "**seqlab &**" in your terminal window to fire up the SeqLab interface. The ampersand, "**&**," is not necessary but really helps out by launching seqlab as a background process so that you can retain control of your initial terminal window. 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 the analyses, go to the "**Options**" menu and select "**Preferences. . .**" We should check a few options there to insure that SeqLab runs its most intuitive manner.

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 probably OK to leave it as is for now. Be sure that the "**Start SeqLab in:**" choice has "**List mode**" selected and that "**Close the window**" is selected under the "**After I push the "Run" button**:" choice. Next select the "**Output**" **"Preference**." Be sure "**Automatically display new output**" is selected. Finally, take a look at the "**Fonts**" menu. We will leave all these choices as is but I want to point out that if you are dealing with very large alignments, then picking a smaller Editor font point size may be desirable in order to see more of your alignment on the screen at once. Click "**OK**" to accept all of your changes.

Be sure the "**Mode:**" "**Main List**" choice is selected in your main window and then go to the "**File**" menu. Pick "**Add sequences from**" and select "**Sequence Files**." (Only GCG format compatible sequences or list files are accessible through this route. Use SeqLab's "Import" function to directly load GenBank format sequences without the need to reformat.) This will produce an "**Add Sequences**" window from which you can select sequences to add to your working.list. The "**Filter**" box is very important here! By default files are filtered such that only those that end with the extension "**.seq**" are displayed. For contig3000.seq that is fine, but it would not be appropriate if we were looking for something else. (For your own information, use the following method to see all the files in your working directory. Delete the ".seq" extension in the "Filter" box; be sure to leave the "*****" wild card. Press the "Filter" button to display all of the files in your working directory.) Select the file entitled "**contig3000.seq**" from the "**Files**" box and then check the "**Add**" and then the "**Close**" buttons at

the bottom of the window to put the file in your working.list. It will appear in the SeqLab "**Main List**" window. Be sure it is selected and then switch to "**Editor**" "**Mode:**" to load the sequence into the SeqLab editor. Notice that the sequence now appears in the editor window with the bases color-coded. Any portion of or all of the sequence is now available for analysis by any of the GCG SeqLab compatibloe programs. Drag the window to an appropriate size by 'grabbing' the bottom-left corner of its 'frame' and 'pulling' it out as far as desired. The display will look something like this:



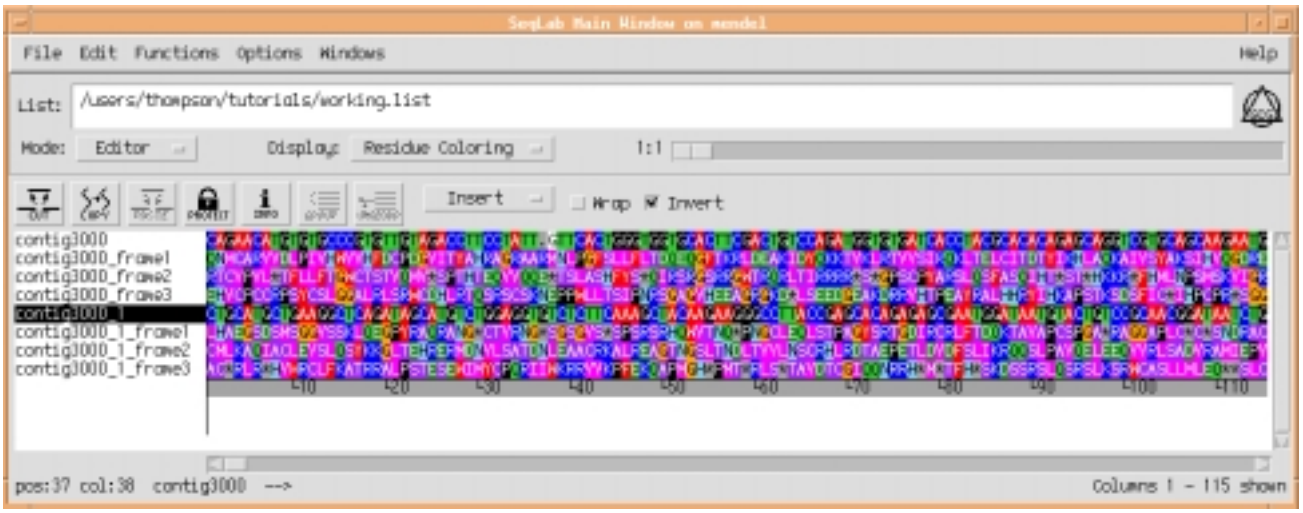
Explore the editor interface for a while. Nearly all GCG programs are accessible through the "Functions" menu including the powerful similarity search tools FastA and BLAST. Do not run a similarity search at this point as we will be running similarity searches later in the exercise on the protein translations from contig3000. The scroll bar at the bottom allows you to move through the sequences linearly. You can select the sequence or any position(s) within it by 'capturing' them with the mouse. The "**pos:**" and "**col:**" indicators show you where the cursor is located in any particular sequence and the overall dataset 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.

2) Map contig3000 with SeqLab's translate function.

URF's and ORF's: what's the difference? Locate all potentially translated reading frames.

The first order of business is to translate all six reading frames within the sequence. We need to tell SeqLab to translate all three forward reading frames and all three reverse reading frames because there is no way of knowing where any genes may lay. It is not uncommon for a stretch of DNA to have genes on opposite reading frames. This will generate all Unidentified Reading Frames (URF's) as opposed to Open Reading Frames (ORF's) which by definition start with a start codon and end with a stop codon. (This can be an especially important consideration when dealing with organisms that have exons and introns, since many exons will not begin with a start codon [only the first will necessarily begin with one], therefore, URF's are the more appropriate choice for most genomic eukaryotic sequences.) Select contig3000 in your SeqLab editor display by clicking on it with your mouse. 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 contig3000. Now go to the "**Edit**" menu and select "**Translate. . ..**" Specify "**Reading Frame:**" "**All Three**" in the "Translate" window that comes up and then press the "**OK**" button. Next, with contig3000 still selected, press the "**COPY**" button and then select the bottom-most sequence, **contig3000_frame3**, and then press the "**PASTE**" button. This will insert a duplicate version of contig3000 as last sequence in the display; select it, only. Now go back to the "**Edit**" menu and select "**Reverse. . ..**" Specify "**Reverse and Complement**" in the "Reverse" window that pops up so that we can analyze contig3000's opposite strand.

Finally, go through the same "**Edit**" "**Translate. . .**" steps that we did on the forward strand so that all six frames will be available. The display should look similar to the following:



Both strands of the DNA sequence can now be seen together with all of the URF's for each; asterisks indicate stop codons. At this point we can see that there are many potentially translated stretches; so what? What can be done with them; how can we turn them into potential genes?

I. Signal Methods: promoters, terminators, repeat regions . . .

FindPatterns and Consensus/FitConsensus; also consider StemLoop & Repeat.

Typical signals to look for are promoter and terminator consensus stretches. GCG provides a searching program named Terminator for looking for the latter in prokaryotic rho-independent cases; however, promoter signals from both prokaryotes and eukaryotes are so varied that they do not have a 'canned' search for them. An impressive eukaryotic transcription factor consensus sequence database has been assembled though, and prokaryotic promoter sequences are fairly well characterized. We can utilize the GCG program FindPatterns to look for these type of sites within our sequence. GCG also provides the ability to find short consensus patterns based on a family of related sequences using weight matrix analysis with the programs Consensus and FitConsensus. These can be used to form and search for specific promoters or other signals based on known sequences. Also, remember that many termination sites are accompanied by inverted repeats, and enhancer sequences are often strong direct repeats; because of these points, the GCG programs StemLoop and Repeat, as well as dotplot procedures, may be helpful.

3) Look for potential regulatory sites by screening with FindPatterns.

One-dimensional signal hunting: simple consensus and pattern matching.

We will use GCG's pattern searching algorithm FindPatterns to locate promoter regions in contig3000. However, let's sidetrack momentarily. I have written and placed a prokaryote promoter consensus pattern based on the *E. coli* data of Hawley and McClure (1983) in the GCG logical directory location GenMoreData. You are welcome to screen contig3000 with this pattern by specifying data=genmoredata:promoter.dat, although it is not required and probably will not be relevant since the pattern encompasses both the -35 and -10 regions (if it was just the -10 portion, some eukaryotic TATA boxes might be found with it). FindPatterns reads its patterns from a distinct type of file known as a pattern.dat file. The Pribnow box pattern file follows so that you can see it's format and content:

```
The standard E. coli RNA polymerase promoter "Pribnow" box file for the program
FINDPATTERNS. This pattern includes both the -35 & the -10 region.  For an
incredibly extensive list of eukaryotic transcription factor recognition sites
see the GCG public datafile tfsites.dat.  To specify one of these files use the
command line option -data=_datafile_name.

Name    Offset    Pattern           Overhang  Documentation  ..

Pribnow     1    TTGACwx{15,21}TAtAaT     0  !Hawley & McClure (1983)
```

Now for the tougher case — contig3000, a eukaryote example. A huge public pattern data file is available. This file (Ghosh, 1990) is called tfsites.dat and is also available in the GCG logical directory location GenMoreData. (You are welcome to take a look at the Transcription Factor Sites database by using the GCG command **name** to resolve the disk location of genmoredata and then by using **more** to display it to your terminal.) Run FindPatterns on contig3000 against tfsites.dat by selecting **contig3000**, only, and then going to SeqLab's "**Function**" menu. From the "**Gene Finding and Pattern Recognition**" submenu select "**FindPatterns**." Next you must specify both the "**Search Set. . .**" and the "**Patterns**" to be used by the program. This gets a bit interesting as you have to navigate through several file chooser boxes to designate your desired input file and the tfsites.dat file. First click the "**Search Set. . .**" button to get a dialog box entitled "**Build FindPattern's Search Set**." Click on "**Add Main List Selection**" to produce the "**List Chooser**;" there select and then "**Add**" the file we've been working on, "**contig3000.seq**." "**Close**" the chooser boxes to return to the FindPatterns program box. Now punch "**Patterns. . .**" to get the "**Pattern Chooser**." Click in the blank box next to "**Pattern Data File. . .**" and insert the words "**genmoredata:tfsites.dat**" and then press the <**return key**>. SeqLab should find the tfsites.dat file and display the patterns in it; "**Close**" the "**Pattern Chooser**." After specifying the search and pattern sets, press the "**Run**" button in the FindPatterns window. The program box will go away and the output will display after a bit. Scroll thorough the file, noticing the incredible quantity of transcription factor patterns found. Also notice that FindPatterns looks on both the forward and reverse strands by default, indicating reverse strand locations with the /Rev designation. "**Close**" the windows when done. An abridged contig3000 tfsites.dat FindPatterns output file follows below:

```
! FINDPATTERNS on /users/thompson/working/Giardia/contig3000.seq allowing 0 mism
atches

! Using patterns from: /gcg/gcgcore/data/moredata/tfsites.dat  June 26, 1999 16:
45 ..


      contig3000.seq  ck: 9121  len: 5,368 ! Contig3000

GCN4-his3-189 /Rev    ATGAGTCAT
         3,305: CTGCA ATGAGTCAT CGTAC

BPV-E2_CS2            ACCNNNNNNGGT
         3,174: ATCGC ACCCCTCATGGT CATCA
         4,453: CCACT ACCTTAACGGGT CCAAC

GCRE                  TGACTC
           990: TATGA TGACTC CTCTT

GCRE /Rev             GAGTCA
           976: AACAA GAGTCA AGCTA
         1,006: TTGAT GAGTCA GAGGA
         3,307: GCAAT GAGTCA TCGTA
```

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

TBP-ADA /Rev           TTTTTTA
         1,471: TTGAC TTTTTTA GACTC
         4,837: CGCAG TTTTTTA ATTCA

REB1-consensus /Rev   CGGGTRRNNR
         4,564: GTTCA CGGGTGGCTA GGAGT

c-Myb-c-myb /Rev      TTCAAT
         1,277: GTAAT TTCAAT CTTGA

CP2-consensus         GCNMNANCMAG
         4,318: GCTCC GCAAGACCCAG CTCCT

CP2-consensus /Rev    CTKGNTNKNGC
            42: GTTCA CTGGGTGGTGC ACTTC

Sp1-gamma-globin_(3) /Rev  GGCCCC
         1,138: TGACC GGCCCC ACACC
         2,016: CTCAA GGCCCC GGCCG

gamma-globin-undefined-site-4 /Rev  GACCCA
         1,691: GGAGC GACCCA GCATA
         4,322: CGCAA GACCCA GCTCC
         5,207: TTAGT GACCCA TTGGT


     Total finds:        691
    Total length:      5,368
 Total sequences:          1
        CPU time:      05.18
```

The output is huge; 691 finds doesn't do us much good. Skim through your output and see if anything stands out as significant. It's not easy to recognize whether anything from this type of search is at all relevant. I do recognize some TATA boxes and a few other well known sites. You can use the system level search utility grep to search the Transcription Factor Sites database by issuing the command **grep string `name -f genmoredata:tfsites.dat`**. It is neccessary to embed the GCG command **name -f genmoredata:tfsites.dat** in single back quotes (`) to expand it's meaning within the UNIX grep utility; otherwise grep would not be able to find the input file. For instance, to search for the second entry above in tfsites.dat issue the following command:

```
% grep BPV-E2_CS2 `name -f genmoredata:tfsites.dat`

BPV-E2_CS2                    0 ACCNNNNNNGGT                                  0
 ! BPV-E2                    Nucleic Acids Res 15: 10267-84 (1987)
```

to get the reference and this may help some, but then you have to go look up the references. Whatever you do, this approach is not at all 'user friendly.' There's got a be a better way.

Other signals that could be looked for in a similar fashion are the Shine-Dalgarno prokaryote translational initiation site, (AGG,GAG,GGA)x{6,9}ATG (Stormo et al., 1982) and eukaryotic ribosome binding sequences. Ribosome binding sequences are based on complementarity to 16s rRNA in prokaryotes; however, in eukaryotes ribosomes seem to initiate translation at the first AUG encountered following the modified

guanosine 5' cap. Kozak (1984) has compiled a consensus at the start codon of **cc(A,g)ccAUGg** which doesn't seem to relate to 18s rRNA complementarity at all, yet seems to hold true in many cases. Try to find this pattern in your sequence, however, we're going to do it in a different, and a bit friendlier way. Select both the forward and reverse **contig3000** sequences by holding down the <**ctrl**> key while you click on the second entry. This allows you to select multiple, nonadjoining entries (To selct a range of adjoining entries, <shift> click the first and last entry.) Next, go to SeqLab's "**Edit**" menu and launch it's "**Find. . .**" utility. If a "**Which selection**" window pops up asking if you want to use the "**selected sequences**" or "**selected region**;" choose "**selected sequences**." Type Kozak's pattern complete with the parenthesis and comma, which mean either an A or a G, into the blank box next to "**Patterns. . .**" and then press the "**Find All**" button. The case of the letters does not matter and T's and U's are treated as identical. Keep increasing the mismatch level until you find at least one occurrence of the pattern. You can tell when you've found something because the residues will all go black. Press the "**Find Next**" button at that point and the display will scroll over to the first occurrence of the pattern. It will be highlighted in red. Keep pressing "**Find Next**" until you've seen all the places where the pattern is found. Take notes of where the patterns are on the sequences.

A main point consensus searches emphasize is "Don't believe everything your computer tells you!" (von Heijne, 1987a). A computer can provide guidance and insight but the limitations can sometimes be overwhelming as is all too evident in these promoter analyses. Oftentimes weight matrix analysis is more appropriate to this type of search. Unfortunately, they are not nearly as simple to set up.

3) Using the weight matrix program FitConsensus to find regulatory sites.

Two-dimensional signal hunting: weight matrices.

GCG has preassembled consensus weight matrices of the donor and acceptor site sequences at exon-intron splice junctions for use with FitConsensus available in their public data files. However, they do not provide any others; therefore, I have reformatted the four weight matrix descriptions of eukaryotic RNA polymerase II promoter elements reported by Bucher (1990) into a form appropriate for GCG's programs. Additionally, McLauchlan et al. (1985) assembled a eukaryotic terminator weight matrix that I have also reformatted for GCG use.

Use the five weight matrices that I reformatted in FitConsensus to help locate the transcription start and termination signals within contig3000 both in the forward and reverse directions. These matrices have the file names **tata.csn**, **cap.csn**, **ccaat.csn**, **gc.csn**, and **terminator.csn**. They have all been placed in the GCG logical location GenMoreData. There is no need to run FitConsensus with the GCG donor and accepter matrices because, as mentioned in the introduction, *Giardia* introns have never been found.

Start the program run in your SeqLab window by selecting just **contig3000** and going to the "**Functions**" "**Gene Finding and Pattern Recognition**" menu; select "**FitConsensus**" there. 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 contig3000. Specify "**genmoredata:tata.csn**" as the "**Consensus matrix file. . .**" by typing it into the box next to the button. Leave the other parameters at their defaults to assure that "**100%**" positions' fit is assured and to generate lists of the 40 best fits to the matrix. Press "**Run**" to launch and then scroll through the output file when it is displayed. Repeat this procedure with each of the five matrices on the forward strand and then select the reverse DNA strand and repeat the same procedure on it. The SeqLab "**Windows**" menu keeps track of the programs run in a session and so provides a handy 'shortcut' to repeated commands.

Notice the output includes the position, frame, and “quality” of each match. The position and quality are tremendously helpful. Position is where in your sequence the specified weight matrix begins, perhaps not where the site that you are most concerned is occurs, rather only where the matrix begins. This is particularly important in the donor and acceptor matrices as these both begin in front of the splice site, not at it. Quality is the percentage fit to the matrix. The higher the percentage, the more probability that the site is an actual signal. The frame designation is troubling — it is misleading as it identifies the frame of the best fit to the matrix not to the coding region — so disregard it. Also note that none of the hits would have been found by a normal FindPatterns type consensus search without mismatches because the scores are all less than 100%.

Those ten files follow below, abridged to include only those sites with a better than 50% fit for each:

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

FITCONSENSUS of: Contig3000

 Using Consensus: cap.csn

CONSENSUS from: Cap signal
Eukaryotic promoter Cap region.  Base freguencies according to
Philipp Bucher (1990) J. Mol. Biol. 212:563-578.
Preferred region:  center between 1 and +5.
Optimized cut-off value:  81.4%.

 List-size: 40  Average quality: 35.27      June 27, 1999 15:34   ..

  position:  1101  2159  2352  2972
     frame:     3     2     3     2
   quality: 51.25 51.63 50.50 50.25

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

FITCONSENSUS of: Contig3000

 Using Consensus: ccaat.csn

CONSENSUS from: CCAAT box
Eukaryotic promoter CCAAT region.  Base freguencies according to
Philipp Bucher (1990) J. Mol. Biol. 212:563-578.
Preferred region:  motif within -212 to -57.
Optimized cut-off value:  87.2%.

 List-size: 40  Average quality: 31.45      June 27, 1999 15:44   ..

  position:   493   727  1155  1589  1791  2199  2267  2749
     frame:     1     1     3     2     3     3     2     1
   quality: 52.25 54.50 50.17 53.33 52.00 50.00 51.25 59.08

  position:  2864  3598  3741  4486  4902  5278
     frame:     2     1     3     1     3     1
   quality: 51.00 52.17 53.08 51.67 54.25 54.08

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

FITCONSENSUS of: Contig3000

 Using Consensus: gc.csn
```

```
CONSENSUS from: GC box
Eukaryotic promoter GC-Box region.  Base freguencies according to
Philipp Bucher (1990) J. Mol. Biol. 212:563-578.
Preferred region:  motif within -164 to +1.
Optimized cut-off value:  88%.

 List-size: 40  Average quality: 30.23      June 27, 1999 15:45   ..

  position:    41   338  3087  4576  4979
     frame:     2     2     3     1     2
   quality: 51.77 50.46 53.85 50.92 51.31

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

FITCONSENSUS of: Contig3000

 Using Consensus: tata.csn

CONSENSUS from: TATA Box
Eukaryotic promoter TATA region.  Base freguencies according to
Philipp Bucher (1990) J. Mol. Biol. 212:563-578.
Preferred region:  center between -36 and -20.
Optimized cut-off value:  79%.

 List-size: 40  Average quality: 25.33      June 27, 1999 15:47   ..

  position:   643  1292
     frame:     1     2
   quality: 50.85 50.46

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

FITCONSENSUS of: Contig3000

 Using Consensus: terminator.csn

CONSENSUS from: Terminator
Possible eukaryotic termination signal region.  Base freguencies
according to McLauchlan et al. (1985) N.A.R. 13:1347-1368.
found in about 2/3's of all eukaryotic gene sequences.

 List-size: 40  Average quality: 24.78      June 27, 1999 15:48   ..

  position:     8    16    70   228   345  1198  1225  1306  1384  1460  1912
     frame:     2     1     1     3     3     1     1     1     1     2     1
   quality: 55.75 59.88 53.75 52.88 62.50 53.50 53.38 56.38 55.38 56.00 54.13


  position:  2048  2411  2630  2705  2733  2856  2947  3048  3459  3624  3626
     frame:     2     2     2     2     3     3     1     3     3     3     2
   quality: 54.75 59.88 61.50 54.88 52.50 56.50 53.00 58.38 57.63 53.00 56.00


  position:  3631  3641  3683  3685  3702  3930  3955  4068  4251  4269  4275
     frame:     1     2     2     1     3     3     1     3     3     3     3
   quality: 56.25 54.38 54.25 56.75 54.00 56.63 55.38 55.13 57.25 53.25 60.38
```

```
  position:  4283  4681  4683  4835  4873  4937  5105
     frame:     2     1     3     2     1     2     2
   quality: 52.75 53.38 66.50 53.63 56.13 55.88 58.00

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

FITCONSENSUS of: Contig3000 reverse strand

 Using Consensus: cap.csn

CONSENSUS from: Cap signal
Eukaryotic promoter Cap region.  Base freguencies according to
Philipp Bucher (1990) J. Mol. Biol. 212:563-578.
Preferred region:  center between 1 and +5.
Optimized cut-off value:  81.4%.

 List-size: 40  Average quality: 35.11      June 27, 1999 15:50   ..

  position:   590   773  2742   5254
     frame:     2     2     3      1
   quality: 51.63 52.00 50.63  51.75

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

FITCONSENSUS of: Contig3000 reverse strand

 Using Consensus: ccaat.csn

CONSENSUS from: CCAAT box
Eukaryotic promoter CCAAT region.  Base freguencies according to
Philipp Bucher (1990) J. Mol. Biol. 212:563-578.
Preferred region:  motif within -212 to -57.
Optimized cut-off value:  87.2%.

 List-size: 40  Average quality: 31.04      June 27, 1999 15:51   ..

  position:    148   163   355   496   692   794   828   984  1541  1593
     frame:      1     1     1     1     2     2     3     3     2     3
   quality:  51.08 61.83 50.67 51.58 53.67 50.42 51.92 51.42 52.33 55.75

  position:  2299  2426  3031  4009  4131  4961  5239  5319 4961  5239  5319
     frame:     1     2     1     1     3     2     1     3    2     1     3
   quality: 50.67 51.50 61.17 53.33 56.83 56.75 50.42 53.5056.75 50.42 53.50

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

FITCONSENSUS of: Contig3000 reverse strand

 Using Consensus: gc.csn

CONSENSUS from: GC box
Eukaryotic promoter GC-Box region.  Base freguencies according to
Philipp Bucher (1990) J. Mol. Biol. 212:563-578.
Preferred region:  motif within -164 to +1.
Optimized cut-off value:  88%.

 List-size: 40  Average quality: 30.38      June 27, 1999 15:52   ..

  position: 300   369  1666  2709  3341
```

```
     frame:   3     3     1     3     2
   quality: .54 50.46 54.38 50.15 50.15

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

FITCONSENSUS of: Contig3000 reverse strand

 Using Consensus: tata.csn

CONSENSUS from: TATA Box
Eukaryotic promoter TATA region.  Base freguencies according to
Philipp Bucher (1990) J. Mol. Biol. 212:563-578.
Preferred region:  center between -36 and -20.
Optimized cut-off value:  79%.

 List-size: 40  Average quality: 25.17      June 27, 1999 15:52   ..

  position:   620  1685  2261  2749  3166  3889
     frame:     2     2     2     1     1     1
   quality: 50.08 52.46 50.08 52.38 50.15 50.31

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

FITCONSENSUS of: Contig3000 reverse strand

 Using Consensus: terminator.csn

CONSENSUS from: Terminator
Possible eukaryotic termination signal region.  Base freguencies
according to McLauchlan et al. (1985) N.A.R. 13:1347-1368.
found in about 2/3's of all eukaryotic gene sequences.

 List-size: 40  Average quality: 25.40      June 27, 1999 15:54   ..

  position:    34    36   123   182   575   952   991  1042  1755  1758  1847
     frame:     1     3     3     2     2     1     1     1     3     3     2
   quality: 58.50 54.00 55.88 52.75 53.25 61.75 61.75 55.75 55.13 53.25 56.50


  position:  1990  2324  2328  2477  2549  2716  2725  2917  2957  3062  3140
     frame:     1     2     3     2     2     1     1     1     2     2     2
   quality: 56.25 62.25 55.38 65.50 57.38 55.00 54.63 55.88 55.88 56.63 56.50


  position:  3242  3299  3420  3464  3466  3471  3632  4066  4103  4443  4470
     frame:     2     2     3     2     1     3     2     1     2     3     3
   quality: 62.50 55.13 52.63 55.00 54.63 54.63 60.13 58.63 66.50 57.00 54.63


  position:  4477  4531  4601  4603  4912  5083  5278
     frame:     1     1     2     1     1     1     1
   quality: 53.00 60.63 59.38 66.50 56.88 56.25 55.50
```

4) Help in locating the ends of genes: Terminator in prokaryotes; StemLoop, Repeat, and finding poly(A) signals in eukaryotes.

The GCG program Terminator will find about 95% of all prokaryotic factor-independent terminators. This is great odds for any computer algorithm; even its namesake Arnold Schwarzenegger would have a hard time

matching this! This program is mentioned for those of you involved with prokaryotic data in your own research programs. However, realize that a main disadvantage of most signal searches, even a sophisticated two-dimensional approach like Terminator, is they find too many false positive sites, in other words they are not discriminatory enough. Just like Schwarzenegger in T2, a few innocents always manage to get in the way.

The GCG programs StemLoop and Repeat may provide some regulatory insight with eukaryotic sequences since many eukaryotic terminators also have hairpin structures associated with them and some enhancer sequences contain strong direct repeats.

The sequence YGTGTTYY has been reported as a eukaryotic terminator consensus (McLauchlan et al., 1985 [this is the consensus from the weight matrix used above]) and the poly(A) adenylation signal **AAUAAA** is well conserved (Proudfoot and Brownlee, 1976); however, realize the inherent problems with consensus searches, as has been previously illustrated. Run the poly(A) search mentioned above on both the forward and reverse strands of contig3000 through the "**Edit**" "**Find**" function as shown previously with the Kozak pattern. Just as with Kozak's pattern, rerun the program increasing the mismatch level until you find at least one poly(A) pattern. Once again, take notes of their locations. You should already have data regarding the termination site from the FitConsensus runs previously made.

II. Content Methods: what the sequence 'looks' like

You have now been exposed to many of the pitfalls of signal type searches. In general, the second type of gene-finding technique, 'searching by content,' is more reliable, at least it seems to be less fraught with false positive problems, however, it can not locate exact positions. Used in concert with the former, the two can be quite powerful tools. Adding in the third, inference through homology, often clinches the story.

Searching by content utilizes the fact that genes necessarily have many constraints placed upon them. This induces certain periodicities and patterns which we can use to help locate coding sequences as opposed to noncoding stretches of DNA. These constraints arise in a number of fashions — the three base genetic code, the 'wobble' hypothesis, an unequal use of synonymous codons, translational factors, the amino acid content of the encoded proteins themselves, and, possibly, because of remnants of an ancient genetic code. All together these factors create distinctly unique coding sequences; non-coding stretches do not exhibit this type of periodic compositional bias. This fact can serve as a gene finding tool in two manners.

5) Using methods based on sequence composition alone.

'Nonrandomness' techniques: TestCode, a gene finding algorithm based exclusively on statistics

The first technique relies solely on the base compositional bias of every third position — nonrandomness. A truly random sequence does not show any type of pattern at all and is not characteristic of any coding sequence. The program can estimate the probability that any stretch of DNA sequence is either coding or noncoding. It will not tell us the strand or the reading frame; however, it does not require any *a priori* assumptions as it relies exclusively on a statistical evaluation of the sequence itself. To run TestCode select **contig3000** and go to the "**Functions**" "**Gene Finding and Pattern Recognition**" menu and pick "**TestCode. . .**." As before, if a "**Which selection**" window pops up asking if you want to use the "**selected sequences**" or "**selected region**;" choose "**selected sequences**." One limitation of this program is it is not designed to detect coding regions shorter than 200 base pairs, hence the 200 bp window size. No claim is made for

significance with windows less than the default 200; therefore, smaller eukaryotic exons may be missed. Run the program at its defaults by pressing the "**Run**" button to produce the following plot:

The plot is divided into three regions. The top and bottom areas predict coding and noncoding regions, respectively, to a confidence level of 95%, while the middle area claims no statistical significance. Diamonds and vertical bars above the graph denote potential stop and start codons respectively.

You may want to press the "**Print. . .**" button to generate a PostScript file of this and all following graphic plots. If you do this, be sure that the "**Output Device:**" in the "**Print**" menu is set to be an Encapsulated PostScript file and that you give it a different filename in the "**Port or File:**" box each time that you create a new PostScript file. Click "**Proceed**" to create the 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 a PostScript print queue on your GCG server. (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 before trying to send it a PostScript file.)

Repeat the TestCode procedure on the reverse strand to yield the following graphic:

6) Running codon usage analysis programs.

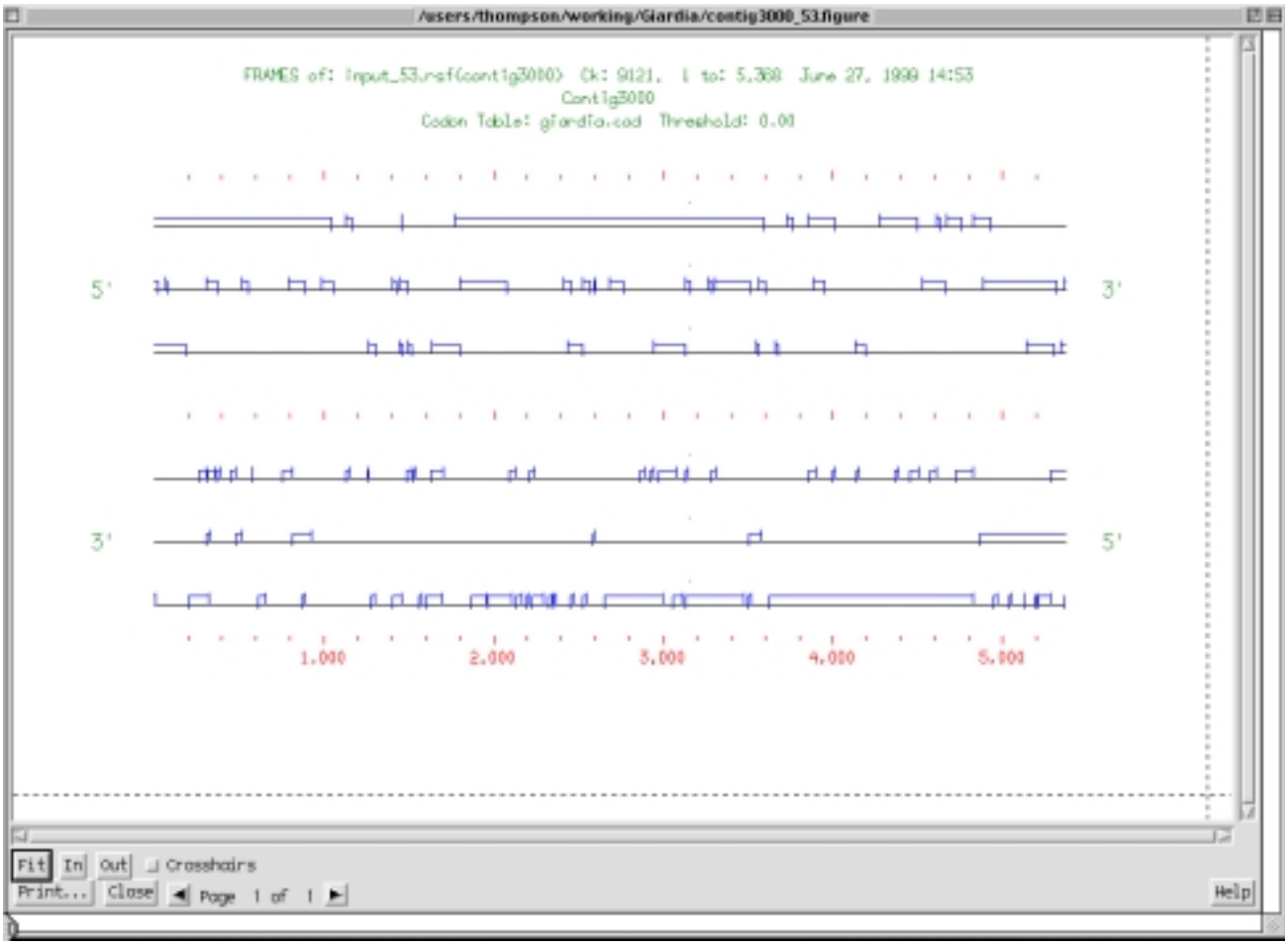
Codon Usage: codon frequency tables and using CodonFrequency.

The second content type of gene finding strategy utilizes the fact that different organisms have different codon usage preferences. In other words, genomes use synonymous codons unequally in a phylogenetic fashion. Codon usage frequency is not the genetic translation code — the genetic code is nearly universal across all phylogenetic lines. However, not all lineages use the same percentage of the various degenerate codons the same amount. The manner in which different types of organisms utilize the available codons is usually tabulated into what is known as a codon usage or frequency table. In order to utilize the codon usage type of gene finding strategy a codon usage table for the particular organism in question must be accessible. The GCG default table for these programs is from highly expressed *E. coli* genes. If your sequence comes from anything else, this table is not appropriate. GCG provides alternate data files in GenMoreData. The available tables, in addition to the default codon usage table, ecohigh.cod, are: celegans_high.cod, celegans_low.cod, drosophila_high.cod, human_high.cod, maize_high.cod, and yeast_high.cod. As mentioned in the Introduction, tables are also available at various molecular biology data servers such as IUBIO (http://iubio.bio.indiana.edu/soft/molbio/codon/), the TRANSTERM database at the European Bioinformatics Institute (ftp://ftp.ebi.ac.uk/pub/databases/transterm/), and the CUTG database (http://www.dna.affrc.go.jp/~nakamura/codon.html) available in GCG format at SRS servers (e.g. see http://www.sanger.ac.uk/srs5bin/cgi-bin/wgetz?-fun+pagelibinfo+-info+CUTG). Furthermore, if you are not satisfied with any of the available options, GCG has a program, CodonFrequency, that enables you to create your own codon frequency table from known coding sequences.

To make your own codon usage table gather all of the known and appropriate coding sequences for your organism and run them through this program. A strategy to achieve this objective, if the need ever arises, follows: To find the sequences in the database you could use the GCG searching program LookUp. The resulting output file is then edited to include only those sequences that refer to structural genes, excluding inappropriate organelle genomes, if that is of concern. The title lines of the sequence files aren't quite what you need though — you need the sequences' references which list CDS regions. Use "fetch -reference -out=filename.ext @your_LookUp_filename" to pull over the references from the database into your own directory. Then use the program CodonFrequency being very careful to only specify the coding regions for each gene from your list. In the case of our *Giardia* genome tutorial, I have placed a copy of a *Giardia* specific codon frequency table in GenMoreData.

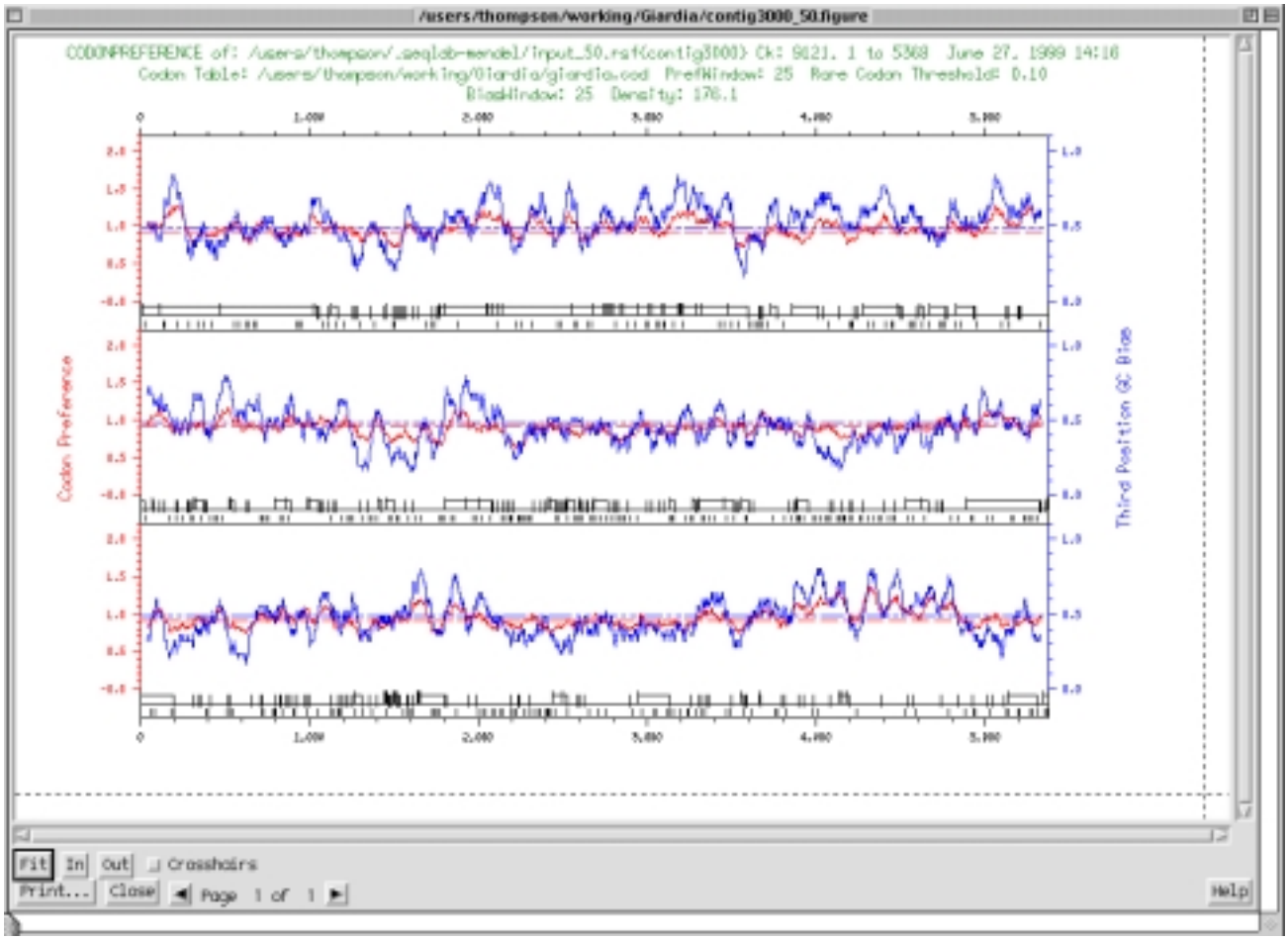
Programs that can utilize codon usage tables to help find genes: Frames & CodonPreference.

The two GCG content analysis programs that can use codon usage tables in this context — Frames, a very simple open reading frame identifier which can utilize codon frequency tables to show rare codon usages, and the quite sophisticated codon frequency analyzer CodonPreference — need to know which codon usage table you want to use. You must determine and specify the codon usage table appropriate for your situation. Do remember, however, that for most eukaryotic genomic sequences, only the first exon will actually have a start codon. Therefore, Frames is generally more appropriate for sequences without exons such as cDNA or prokaryotic data. We just lucked out that *Giardia* appears not to have introns. Perform a Frames analysis with **contig3000**'s forward strand only selected, since Frames automatically runs on both forward and reverse strands. "**Frames**" is located under the "**Functions**" "**Gene Finding and Pattern Recogniton**" menu. Press the "**Options**" button for a chance to change from the default *E. coli* codon frequency table. Type "**genmoredata:giardia.cod**" into the box next to the "**Codon Frequency Table. . .**" button to use that table. Press the "**Run**" button to produce a plot like the following:



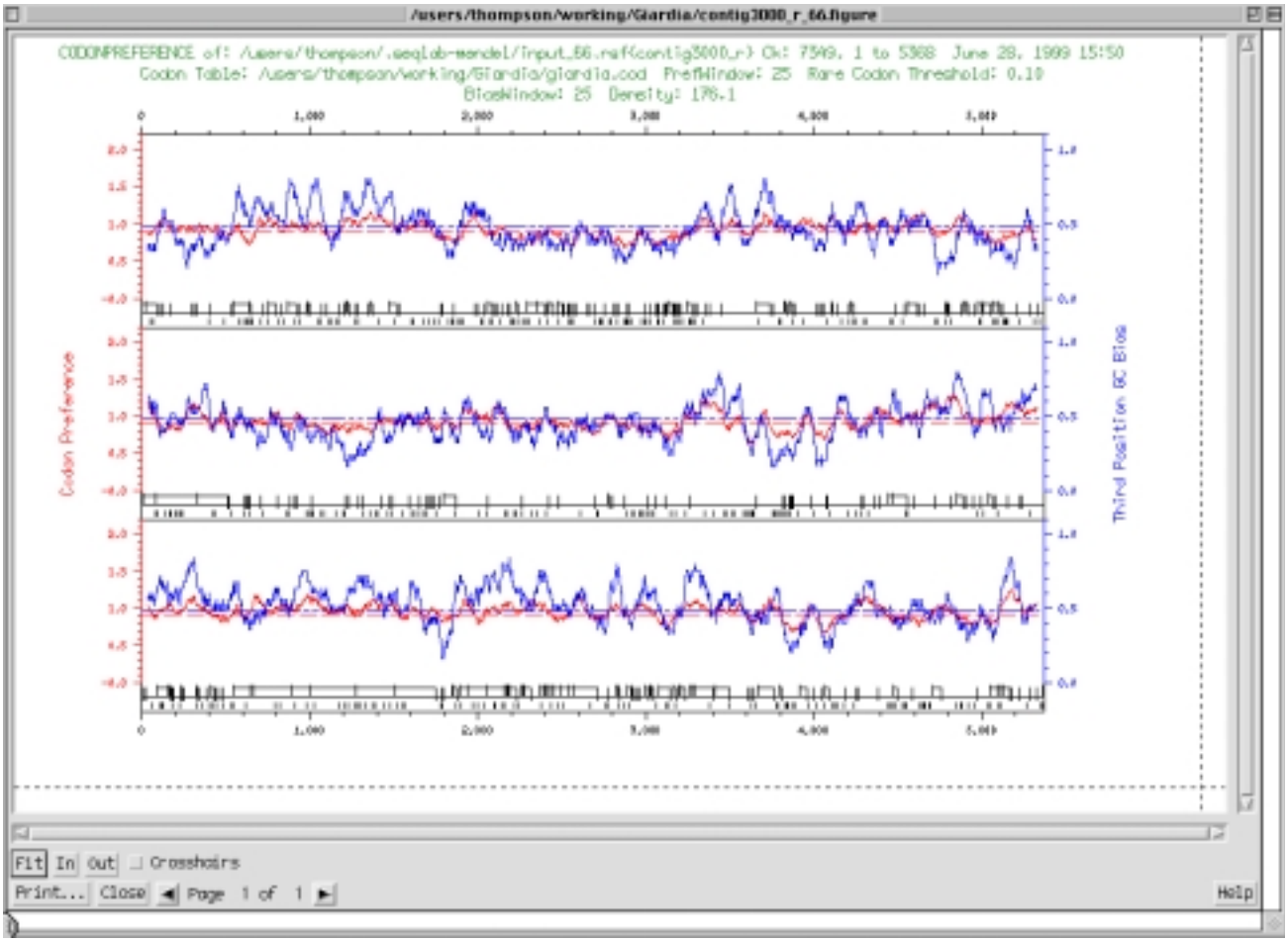
The plot shows all open reading frames and marks rare codon choices with a dot above each.

Next run CodonPreference. It is also located under the "**Functions**" "**Gene Finding and Pattern Recogniton**" menu. First change from the default *E. coli* codon usage table by typing "**genmoredata:giardia.cod**" into the box next to the "**Codon Frequency Table. . .**" button. Then press the "**Options**" button and check "**Show all start and stop signals, not just open frames**." Since this is preliminary genomic data, this option will allow us to see whether sequencing errors may be respossible for the interruption of ORFs. The plot from the CodonPreference run on the forward strand is illustrated below:



The plot shows two color coded curves, a codon preference curve and a third position bias curve, for each reading frame of the sequence in question. The curves rise above background scatter in areas of strong probability of coding potential. The horizontal lines within each plot are the average values of each attribute. CodonPreference moves its window in increments of three recalculating its statistic at each position to generate a continuous function so that each function defines an individual reading frame. An open reading frame display accompanies each panel with start codons represented as vertical lines rising above each box and stop codons shown as lines falling below the reading frame boxes. Rare codon choices are also shown for each frame as hash marks. An optional output from the program can help in the interpretation of significance by calculating the average codon preference for each frame and for the whole sequence randomized, which can be compared to the peaks of the plot. One must realize, however, that not all genes show particularly high codon usage preference. This is especially true of genes which are only weakly expressed. Therefore, you must, as always, interpret results with more than just a few grains of salt. Use as many sources of information as possible!

Repeat the CodonPreference analysis on the reverse strand of contig3000 to produce this plot:



III. Search the databases for sequence similarity, thereby infer gene location through homology.

Enter the world of database searching — background processes, parallel processing, huge numbers, and questions such as, "Is this significant; does it mean anything?" and "This is homologous?" A world often fraught with frustrating results but also exciting discoveries!

As I discussed in the introduction, similarity analysis should usually be done on the protein level versus the DNA level. Therefore, for this section of the tutorial, select all the translations made at the beginning; there should be six of them. Be sure that the DNA is not selected. Many of the programs in this section are cpu 'hogs' so in some cases I do not want you to run the search. I will clearly state the situation for each search discussed.

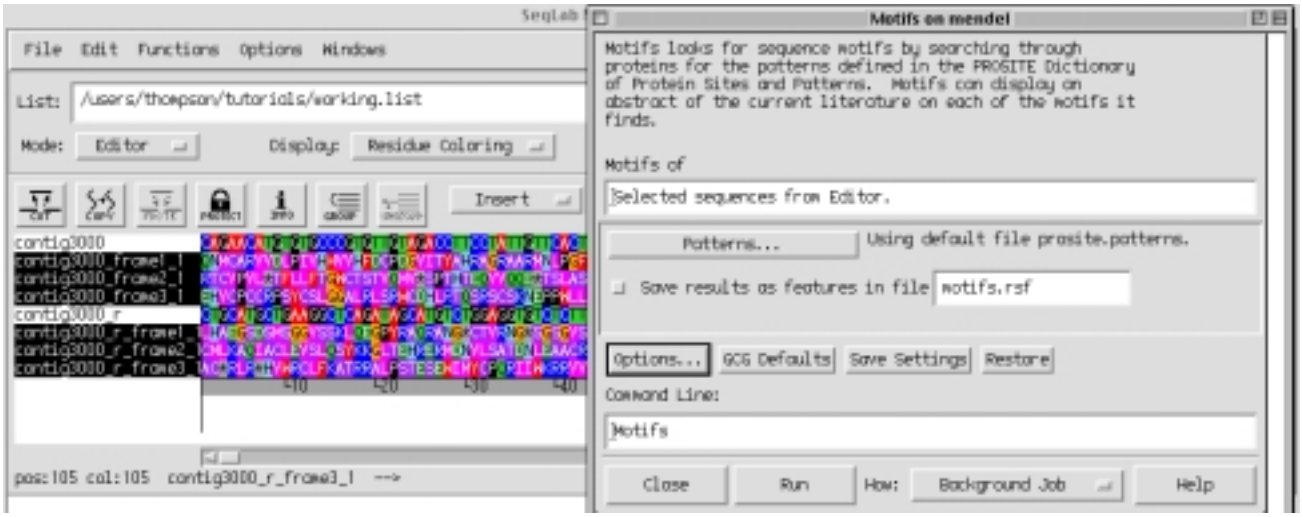
7) Search the PROSITES database.

A Quick and Dirty Method — GCG's Motifs search of PROSITES.

Many, many features have been described and catalogued in sequences over the years. Many of these have recognizable consensus patterns that allow you to screen an unknown sequence for their occurrence. This database of catalogued structural, regulatory and enzymatic consensus patterns is Dr. Amos Bairoch's protein signature database, the *PROSITE Dictionary of Protein Sites and Patterns* (1992). It is one of the quickest and easiest databases to search with a peptide sequence. The GCG program Motifs performs this search. The program can tolerate mismatches with a mismatch option and it displays an abstract with selected references for each motif signature found. In many cases this can be a tremendous aid in

ascertaining the function of an unknown peptide sequence. 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.

Be sure all of your tentative translations are selected, but not the DNA. Start the Motifs program by clicking on the "**Functions**" "**Database Sequence Searching**" "**Motifs. . .**;" button. 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 all the selected sequences. The "**Motifs**" program window will then display and should look similar to the following:



Check the "**Save results as features in file motifs.rsf**" button. We'll use this file later on. We don't need any of the other options so press the "**Run**" button. After a few minutes you should get output. The file displayed, "motifs.rsf," isn't very interesting at this point so "**Close**" it and use the "Output Manager" to display the file with the "**.motifs**" extension. Carefully look over the text file that is displayed. Notice the sites that have been characterized in these sequences and the extensive bibliography associated with them:

```
MOTIFS from: @/users/thompson/.seqlab-mendel/motifs_74.list

 Mismatches: 0                July 14, 1999 21:54  ..


input_74.rsf{CONTIG3000_FRAME1_1}  Check: 311   Length: 1,790 !

_____________________________________________________________________________

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}(V,S){2}(I)(S)C
         1,419: I*GAL                             FAIAIMSVISC
         FQLTS

*********************************************************
* 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).
^^

Thiol_Protease_His (L,I,V,M,G,S,T,A,N)xH(G,S,A,C,E)(L,I,V,M)x(L,I,V,M,A,T)2Gx(G,S,A,D,N,H)
                                                (I)xH(S)(L)x(L,V){2}Gx(A)
         1,760: RCGQY   IIHSLSVLGKA
              LLVAL

****
* Eukaryotic thiol (cysteine) proteases active sites *
****

Eukaryotic thiol proteases (EC 3.4.22.-) [1] are a family of proteolytic enzymes which contain an active site cysteine. Catalysis proceeds through a thioester intermediate and is facilitated by a nearby histidine side chain; an asparagine completes the essential catalytic triad. The proteases which are currently known to belong to this family are listed below (references are only provided for recently determined sequences).

- Vertebrate lysosomal cathepsins B (EC 3.4.22.1), H (EC 3.4.22.16), L (EC 3.4.22.15), and S (EC 3.4.22.27) [2].
- Vertebrate lysosomal dipeptidyl peptidase I (EC 3.4.14.1) (also known as cathepsin C) [2].
- Vertebrate calpains (EC 3.4.22.17). Calpains are intracellular calcium-activated thiol protease that contain both a N-terminal catalytic domain and a C-terminal calcium-binding domain.
- Mammalian cathepsin K, which seems involved in osteoclastic bone resorption [3].
- Human cathepsin O [4].
- Bleomycin hydrolase. An enzyme that catalyzes the inactivation of the antitumor drug BLM (a glycopeptide).
- Plant enzymes: barley aleurain (EC 3.4.22.16), EP-B1/B4; kidney bean EP-C1, rice bean SH-EP; kiwi fruit actinidin (EC 3.4.22.14); papaya latex papain (EC 3.4.22.2), chymopapain (EC 3.4.22.6), caricain (EC 3.4.22.30), and proteinase IV (EC 3.4.22.25); pea turgor-responsive protein 15A; pineapple stem bromelain (EC 3.4.22.32); rape COT44; rice oryzain alpha, beta, and gamma; tomato low-temperature induced, Arabidopsis thaliana A494, RD19A and RD21A.
- House-dust mites allergens DerP1 and EurM1.
- Cathepsin B-like proteinases from the worms Caenorhabditis elegans (genes gcp-1, cpr-3, cpr-4, cpr-5 and cpr-6), Schistosoma mansoni (antigen SM31) and Japonica (antigen SJ31), Haemonchus contortus (genes AC-1 and AC-2), and Ostertagia ostertagi (CP-1 and CP-3).
- Slime mold cysteine proteinases CP1 and CP2.
- Cruzipain from Trypanosoma cruzi and brucei.
- Throphozoite cysteine proteinase (TCP) from various Plasmodium species.
- Proteases from Leishmania mexicana, Theileria annulata and Theileria parva.
- Baculoviruses cathepsin-like enzyme (v-cath).
- Drosophila small optic lobes protein (gene sol), a neuronal protein that contains a calpain-like domain.

- Yeast thiol protease BLH1/YCP1/LAP3.
- Caenorhabditis elegans hypothetical protein C06G4.2, a calpain-like protein.

Two bacterial peptidases are also part of this family:

- Aminopeptidase C from Lactococcus lactis (gene pepC) [5].
- Thiol protease tpr from Porphyromonas gingivalis.

Three other proteins are structurally related to this family, but may have lost their proteolytic activity.

- Soybean oil body protein P34. This protein has its active site cysteine replaced by a glycine.
- Rat testin, a sertoli cell secretory protein highly similar to cathepsin L but with the active site cysteine is replaced by a serine. Rat testin should not be confused with mouse testin which is a LIM-domain protein (see <PDOC00382>).
- Plasmodium falciparum serine-repeat protein (SERA), the major blood stage antigen. This protein of 111 Kd possesses a C-terminal thiol-protease-like domain [6], but the active site cysteine is replaced by a serine.

The sequences around the three active site residues are well conserved and can be used as signature patterns.

-Consensus pattern: Q-x(3)-[GE]-x-C-[YW]-x(2)-[STAGC]-[STAGCV]
[C is the active site residue]
-Sequences known to belong to this class detected by the pattern: ALL, except for P34, testins, SERA antigen, and Theileria annulara protease.
-Other sequence(s) detected in SWISS-PROT: 4.

-Note: the residue in position 4 of the pattern is almost always cysteine; the only exceptions are calpains (Leu), bleomycin hydrolase (Ser) and yeast YCP1 (Ser).
-Note: the residue in position 5 of the pattern is always Gly except in papaya protease IV where it is Glu.

-Consensus pattern: [LIVMGSTAN]-x-H-[GSACE]-[LIVM]-x-[LIVMAT](2)-G-x-[GSADNH]
[H is the active site residue]
-Sequences known to belong to this class detected by the pattern: ALL, except for calpains, P34 and tpr.
-Other sequence(s) detected in SWISS-PROT: 104.

-Consensus pattern: [FYCH]-[WI]-[LIVT]-x-[KRQAG]-N-[ST]-W-x(3)-[FYW]-G-x(2)-G-[LFYW]-[LIVMFYG]-x-[LIVMF]
[N is the active site residue]
-Sequences known to belong to this class detected by the pattern: ALL, except for calpains, bromelain, yeast BLH1, tomato low-temperature induced protease, cathepsin O, pepC and tpr.
-Other sequence(s) detected in SWISS-PROT: NONE.

-Note: these proteins belong to family C1 (papain-type) and C2 (calpains) in the classification of peptidases [7,E1].

-Expert(s) to contact by email: Turk B.
boris.turk@ijs.si

-Last update: November 1997 / Text revised.

[1] Dufour E.
     Biochimie 70:1335-1342(1988).
[2] Kirschke H., Barrett A.J., Rawlings N.D.
     Protein Prof. 2:1587-1643(1995).
[3] Shi G.-P., Chapman H.A., Bhairi S.M., Deleeuw C., Reddy V.Y., Weiss S.J.
     FEBS Lett. 357:129-134(1995).
[4] Velasco G., Ferrando A.A., Puente X.S., Sanchez L.M., Lopez-Otin C.
     J. Biol. Chem. 269:27136-27142(1994).
[5] Chapot-Chartier M.P., Nardi M., Chopin M.C., Chopin A., Gripon J.C.
     Appl. Environ. Microbiol. 59:330-333(1993).
[6] Higgins D.G., McConnell D.J., Sharp P.M.
     Nature 340:604-604(1989).
[7] Rawlings N.D., Barrett A.J.
     Meth. Enzymol. 244:461-486(1994).
[E1] http://www.expasy.ch/cgi-bin/lists?peptidas.txt
^^

```
input_74.rsf{CONTIG3000_FRAME2_1}  Check: 6238  Length: 1,789 !

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,T){2}(T)(S)C
           396: LAVKT                           SVVSVYTLTSC
         LLVVG

                                        ~(D,E,R,K){6}(L,V){2}(V)(G)C
           402: VVSVY                           TLTSCLLVVGC
         SVQKV

Find reference above under sequence: input_74.rsf{CONTIG3000_FRAME1_1}, pattern:
 Prokar_Lipoprotein.


input_74.rsf{CONTIG3000_FRAME3_1}  Check: 5853  Length: 1,789 !

________________________________________________________________________________

Atpase_C              (G,S,T,A)R(N,Q)Px10(L,I,V,M,F,Y,W)2x3(L,I,V,M,F,Y,W)x(D,E)
                                    (S)R(Q)Px{10}(L){2}x{3}(L)x(D)
         1,726: GVDSC                   SRQPLGHWLVTHWCLLLEGLYD
 TPLPD
```

****
* ATP synthase c subunit signature *
****

ATP synthase (proton-translocating ATPase) (EC 3.6.1.34) [1,2] is a component of the cytoplasmic membrane of eubacteria, the inner membrane of mitochondria, and the thylakoid membrane of chloroplasts. The ATPase complex is composed of an oligomeric transmembrane sector, called CF(0), which acts as a proton channel, and a catalytic core, termed coupling factor CF(1).

The CF(0) c subunit (also called protein 9, proteolipid, or subunit III) [3,4] is a highly hydrophobic protein of about 8 Kd which has been implicated in the proton-conducting activity of ATPase. Structurally subunit c consist of two long terminal hydrophobic regions, which probably span the membrane, and a central hydrophilic region. N,N'-dicyclohexylcarbodiimide (DCCD) can bind covalently to subunit c and thereby abolish the ATPase activity. DCCD binds to a specific glutamate or aspartate residue which is located in the middle of

```
the second hydrophobic region near the C-terminus of the protein.

We derived a signature pattern which includes the DCCD-binding residue.

-Consensus pattern: [GSTA]-R-[NQ]-P-x(10)-[LIVMFYW](2)-x(3)-[LIVMFYW]-x-[DE]
                    [D or E binds DCCD]
-Sequences known to belong to this class detected by the pattern: ALL,  except
 for sunflower mitochondrial encoded subunit C which has Trp instead of Arg in
 position 2 of the pattern.
-Other sequence(s) detected in SWISS-PROT: 2.

-Note: the proteolipid subunit of the vacuolar ATPase,  a 16 Kd protein, which
 also binds DCCD, is evolutionary  related to subunit c  and has arisen by the
 duplication of a subunit c type domain. This protein is however too divergent
 to be detected by this pattern.

-Expert(s) to contact by email: Recipon H.
                                recipon@ncbi.nlm.nih.gov

-Last update: December 1992 / Text revised.

[ 1] Futai M., Noumi T., Maeda M.
     Annu. Rev. Biochem. 58:111-136(1989).
[ 2] Senior A.E.
     Physiol. Rev. 68:177-231(1988).
[ 3] Ivaschenko A.T., Karpenyuk T.A., Ponomarenko S.V.
     Biokhimiia 56:406-419(1991).
[ 4] Recipon H., Perasso R., Adoutte A., Quetier F.
     J. Mol. Evol. 34:292-303(1992).
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

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}(F,A){2}(I)(S)C
         1,208: NKELL                             SVVCLPAFISC
         VTMIG

Find reference above under sequence: input_74.rsf{CONTIG3000_FRAME1_1}, pattern:
 Prokar_Lipoprotein.


input_74.rsf{CONTIG3000_R_FRAME1_1}  Check: 7049  Length: 1,790 !

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}(S,A){2}(G)(S)C
           316: C*LRR                             PWGVFAASGSC
         *SMAS


                                          ~(D,E,R,K){6}(T,A){2}(S)(A)C
         1,435: QLTNR                             VSISLSTASAC
         ACSAF

                                          ~(D,E,R,K){6}(S,A){2}(C)(A)C
         1,437: TNRVS                             ISLSTASACAC
         SAFPP

Find reference above under sequence: input_74.rsf{CONTIG3000_FRAME1_1}, pattern:
 Prokar_Lipoprotein.
```

input_74.rsf{CONTIG3000_R_FRAME2_1} Check: 8505 Length: 1,789 !

```
Cytochrome_C          C~(C,P,W,H,F)~(C,P,W,R)CH~(C,F,Y,W)
                      C~(C,P,W,H,F)~(C,P,W,R)CH~(C,F,Y,W)
         1,627: WLADP              CS*CHL                LF*CR
```

```
***************************************************
* Cytochrome c family heme-binding site signature *
***************************************************
```

In proteins belonging to cytochrome c family [1], the heme group is covalently attached by thioether bonds to two conserved cysteine residues. The consensus sequence for this site is Cys-X-X-Cys-His and the histidine residue is one of the two axial ligands of the heme iron. This arrangement is shared by all proteins known to belong to cytochrome c family, which presently includes cytochromes c, c', c1 to c6, c550 to c556, cc3/Hmc, cytochrome f and reaction center cytochrome c.

-Consensus pattern: C-{CPWHF}-{CPWR}-C-H-{CFYW}
-Sequences known to belong to this class detected by the pattern: ALL, except for four cytochrome c's which lack the first thioether bond.
-Other sequence(s) detected in SWISS-PROT: 421.

-Note: some cytochrome c's have more than a single bound heme group: c4 has 2, c7 has 3, c3 has 4, the reaction center has 4, and cc3/Hmc has 16 !

-Last update: June 1992 / Text revised.

[1] Mathews F.S.
     Prog. Biophys. Mol. Biol. 45:1-56(1985).

^^

input_74.rsf{CONTIG3000_R_FRAME3_1} Check: 4540 Length: 1,789 !

```
Pii_Glnb_Ump          Y(K,R)G(A,S)(A,E)Y
                         Y(K)G(S)(A)Y
           562: TGQAH       YKGSAY       HRNAG
```

```
***************************
* P-II protein signatures *
***************************
```

The P-II protein (gene glnB) is a bacterial protein important for the control of glutamine synthetase [1,2,3]. In nitrogen-limiting conditions, when the ratio of glutamine to 2-ketoglutarate decreases, P-II is uridylylated on a tyrosine residue to form P-II-UMP. P-II-UMP allows the deadenylation of glutamine synthetase (GS), thus activating the enzyme. Conversely, in nitrogen excess, P-II-UMP is deuridylated and then promotes the adenylation of GS. P-II also indirectly controls the transcription of the GS gene (glnA) by preventing NR-II (ntrB) to phosphorylate NR-I (ntrC) which is the transcriptional activator of glnA. Once P-II is uridylylated, these events are reversed.

P-II is a protein of about 110 amino acid residues extremely well conserved.

The tyrosine which is urydylated is located in the central part of the protein.

In cyanobacteria, P-II seems to be phosphorylated on a serine residue rather than being urydylated.

In methanogenic archaebacteria, the nitrogenase iron protein gene (nifH) is followed by two open reading frames highly similar to the eubacterial P-II protein [4]. These proteins could be involved in the regulation of nitrogen fixation.

In the red alga, Porphyra purpurea, there is a glnB homolog encoded in the chloroplast genome.

Other proteins highly similar to glnB are:

- Bacillus subtilis protein nrgB [5].
- Escherichia coli hypothetical protein ybaI [6].

We developed two signature patterns for P-II protein. The first one is a conserved stretch (in eubacteria) of six residues which contains the urydylated tyrosine, the other is derived from a conserved region in the C-terminal part of the P-II protein.

-Consensus pattern: Y-[KR]-G-[AS]-[AE]-Y
                    [The second Y is uridylated]
-Sequences known to belong to this class detected by the pattern: ALL glnB's from eubacteria.
-Other sequence(s) detected in SWISS-PROT: 4.

-Consensus pattern: [ST]-x(3)-G-[DY]-G-[KR]-[IV]-[FW]-[LIVM]-x(2)-[LIVM]
-Sequences known to belong to this class detected by the pattern: ALL.
-Other sequence(s) detected in SWISS-PROT: NONE.

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

[1] Magasanik B.
     Biochimie 71:1005-1012(1989).
[2] Holtel A., Merrick M.
     Mol. Gen. Genet. 215:134-138(1988).
[3] Cheah E., Carr P.D., Suffolk P.M., Vasuvedan S.G., Dixon N.E.,
     Ollis D.L.
     Structure 2:981-990(1994).
[4] Sibold L., Henriquet M., Possot O., Aubert J.-P.
     Res. Microbiol. 142:5-12(1991).
[5] Wray L.V. Jr., Atkinson M.R., Fisher S.H.
     J. Bacteriol. 176:108-114(1994).
[6] Allikmets R., Gerrard B.C., Court D., Dean M.C.
     Gene 136:231-236(1993).
^^

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}(S,T){2}(V)(G)C
308: VKVVD NPLLTATSVGC
LRCLR

~(D,E,R,K){6}(T,G){2}(A)(G)C
1,172: GVEIE ANASPGGTAGC

```
         AHSIP

Find reference above under sequence: input_74.rsf{CONTIG3000_FRAME1_1}, pattern:
 Prokar_Lipoprotein.
```

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. More sites will be listed if you specify the frequent option. However, realize that just as in promoter consensus searches, many sites may be false positives. This is most likely the case in the contig3000 example with the prokaryotic lipoprotein site. Whether any of the other sites are biologically relevant should become clearer with the completion of the analysis.

8) Traditional database searching: FastA style approaches — running the algorithms.

Two different symbol matching algorithms have traditionally been utilized in database searching. These two algorithms (see the GCG Program Manual for details) are incorporated into GCG's FastA (Pearson and Lipman, 1988) and WordSearch (Wilbur and Lipman, 1983) programs. WordSearch is rarely used anymore, 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 I will not be illustrating WordSearch. Most of these programs eat cpu; pay attention to the necessity of running them in the background.

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

TFastA is one of the more robust of the searching programs around. It compares your peptide sequence against translations of the DNA database. This way you can take advantage of the multitude of DNA sequences that never make it to the protein databases and yet still retain the sensitivity of protein searches. TFastX makes it even more robust by allowing frame changes that minor sequencing mistakes can cause.

This is one of those cpu intensive programs that I do not want everybody running on every sequence. Therefore, only run TFastX on the first forward-frame translation. I will show the output from the other five searches. Therefore, just select the **contig3000_frame1** sequence and go to the "**Functions**" "**Database Sequence Searching** " menu and select "**TFastX. . .**" to start the Translation FastX program. If a "**Which selection**" window pops up asking if you want to use the "**selected sequences**" or "**selected region**;" choose "**selected sequences**" to run the program on the full length of contig3000_frame1. The default database to search, "**Search Set. . .**" "**Using genembl:***" is fine as are the other parameters in the main TFastX window. (I should point out that the GenEMBL sequence specification does not include the "Tags" division, i.e. all EST's and GSS's; to search all nucleic acid databases use the sequence specification GenEMBLPlus [or GEP].) Press the "**Options. . .**" button to check out and change optional parameters. Scroll down the window and uncheck "**Show sequence alignments in the output file**" to take advantage of the command line option -noalign in order to suppress all the alignments since most are redundant for our purposes and we will be investigating the more interesting ones later anyway. Some of the other options can be very helpful depending on your specific situation and should be explored in your own research. 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 final dynamic programming pass, rather than the initn score, 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. Go on with the rest of the tutorial rather than waiting for results at this point.

Contrast with normal FASTA — protein against protein.

Now run the normal FastA program on the same **contig3000_frame1** sequence searching either the "**PIR:***" and/or "**SwissProt:***" or "**SwissProtPlus:***" logical protein sequence specification. (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. Be sure to suppress alignments again by unchecking "**Show sequence alignments in the output file**" in the "**Options**" menu. "**Close**" the "**Options**" window and "**Run**" the FastA program. Again, proceed with the remainder of the exercise, as these programs will run for a while. Their results will appear as they finish (or be there next time you log on). We will review the results of all of the searches later on.

9) BLAST; Internet and local similarity searching.

The BLAST (Altschul, et al., 1990) server at NCBI can provide the most up to date and quickest database search available. BLAST is a heuristic algorithm for searching sequence databases developed by the National Center for Biotechnology Information at the National Library of Medicine. The acronym stands for Basic Local Alignment Search Tool. NCBI's BLAST by default runs on an eight processor parallel computer system. The original BLAST only looked for ungapped segments; however, the current version (Altschul, et al., 1997) adds a dynamic programming step to produce gapped alignments. BLAST ranks matches statistically and provides probability values for each to help evaluate significance. It is best for identifying shorter regions of high similarity — exactly what you might want with a sequence of unknown function. It is very fast, about an order of magnitude over traditional sequence similarity database searching, yet maintains the sensitivity of older methods for local similarity in protein sequences! BLAST shows you the best alignment for each similar sequence found linked to the next best alignments up to a certain preset cutoff point. This combines the power of dot-matrix type analyses and the interpretative ease of traditional sequence alignments. One can fine-tune BLAST by altering its operating parameters and taking advantage of the many options available in it; however, BLAST is not very appropriate for comparing non-protein-coding nucleotide sequences against the nucleotide database. When you are forced to perform this type of nucleotide-to-nucleotide search it is usually best to use FastA style algorithms instead.

NCBI's BLAST accesses the latest (GenBank updates every night) database by default, nucleotide or protein. The GCG implementation of NCBI's BLAST, called NetBLAST, runs in a remote client-server mode such that NCBI's database and computer perform the analysis. Alternatively you can run GCG's local BLAST program if you have BLAST databases assembled at your site. For help in interpreting BLAST results refer to the GCG BLAST documentation or the BLAST HELP file obtained off the web or 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 can be in valid GCG "list file" format so that it can be fed directly to other GCG programs. Unlike all other GCG programs, the list generated by NetBLAST is <u>not</u> appropriate as input to other GCG analyses. NetBLAST returns files in NCBI's own format and it is not compatible with GCG's. For that reason I will be showing local BLAST here, though the same procedures and logic apply to NetBLAST.

(If you do use NetBLAST, because your site does not maintain local BLAST databases or because you need the very latest sequence data available, then you may have to wait for a few moments in a user waiting queue at NCBI because it tends to get

quite busy off of Web traffic. Furthermore, you will have to modify NCBI's format to make it comply with GCG standards, if you want to be able to read it with GCG programs. For your information some features of BLAST's output format need to be pointed out. This output list is generated by NCBI's computer which doesn't know about GCG format requirements. Therefore, if you want to use the results of a NCBI BLAST search in another GCG program you must manually edit the list changing the database names to reflect the logicals that GCG understands. For example, change sp|P40250|PRIO_CERAE to SW:P40250 PRIO_CERAE (either insert a blank space or ! between the access code and sequence name). The "gi" designation is all the translations from GenBank's CDS references. For instance, "gi|190518 (M81929) prion protein [Homo sapiens]" is the CDS translation from GenBank accession code M81929. This database, GenPept, is installed on some site's systems. The GCG logicals "GP," "GenPep," and/or "GenPept" (case independent) usually point to the GenPept database, if your site maintains it. A number following an underscore indicates the respective CDS region of the entry. For example, this representative line from a BLAST report:

gi|21913 (X62626) vicilin [Theobroma cacao] 99 1.2e-10 4

Tells me that I either have to translate GenBank:X62626 entry's CDS region or I can directly specify the sequence from GenPept that corresponds to GenBank accession code X62626. Unfortunately, BLAST reports list GenBank accession codes and GenPept is based on Locus names; therefore, first run GCG's "typedata -ref" command on the GenBank entry's accession code in order to find the corresponding Locus name. Here that Locus name is TCCSVSV; therefore, specify GenPept:TCCSVSV_1 (or just gp:tccsvsv_1) to use it in any GCG program. This does make life a bit more complicated but is not that difficult to work around.)

To launch GCG's local BLAST program, be sure that only **contig3000_frame1** is selected and then pick "**Blast. . .**" off of the "**Functions**" "**Database Sequence Searching**" menu. As above, if a "**Which selection**" window pops up asking if you want to use the "**selected sequences**" or "**selected region**," choose "**selected sequences**." Accept the program defaults on the main window including "**Search a nucleotide database**" "**Search Set. . .**" "**Using local genembl.**" Using BLAST in this manner, that is a protein query against the nucleotide database, activates TBLASTN and provides maximum sensitivity and database size just as it did with TFastX. Push the "**Options. . .**" button to get a chance to review and use some of them. Notice that "**Filter input sequences for complex / repeat regions**" is checked by default. This activates a very powerful option that should generally be taken advantage of. This option, the -filter=xs switch, causes the troublesome portions of the query sequence to be ignored in the search. This is very powerful for screening out low complexity and repeat sequences from your query to minimize 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.) Change "**Display alignments from how many sequences**" from 100 to 1 (this is the same as specifying -segments=1 at the command line), to suppress segment alignments to only the first one and hence reduce the size of the output file. The standard output file is very long because BLAST automatically aligns the best 100 matches. Check in "**Process the output to be a valid GCG list file**" so that we can directly pass the output back to SeqLab. "**Close**" the "**Options**" window and then press the "**Run**" button in BLAST's window. You should get the following output after a few minutes. You may also get your TFastX and FastA outputs somewhere along now. Use the "**Output Manager**" located under SeqLab's "**Windows**" menu to display and manage these 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. I will show the abridged output files next, all for contig3000_frame1, from local TBLASTN, TFastX, and FastA. They follow below:

```
TBLASTN 2.0.5 [May-5-1998]

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= /users/thompson/.seqlab-
mendel/input_38.rsf{CONTIG3000_FRAME1_1}
         (1790 letters)

Database: genembl
           602,539 sequences; 1,199,477,030 total letters

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

                                                                  Score     E
Sequences producing significant alignments:                      (bits)  Value

GB_PL1:SCU72149 U72149 Saccharomyces cerevisiae putative RNA he...   111  2e-22
GB_PL1:SCYJL033W Z49308 S.cerevisiae chromosome X reading frame...   109  6e-22
GB_PR3:HSU28042 U28042 Human DEAD box RNA helicase-like protein...    98  2e-18
GB_PR1:HSRNAHELC X98743 H.sapiens mRNA for RNA helicase (Myc-re...    78  2e-12
GB_PL1:SPAC1F7 Z67998 S.pombe chromosome I cosmid c1F7. 4/98          78  2e-12
GB_PL1:SC8175 X80836 S.cerevisiae chromosome XIII cosmid 8175. ...    73  8e-11
GB_IN:DMU84552 U84552 Drosophila melanogaster helicase pitchoun...    64  4e-08
GB_PL2:ATH010468 AJ010468 Arabidopsis thaliana mRNA for DEAD bo...    59  1e-06
GB_IN:CEZK512 Z22177 Caenorhabditis elegans cosmid ZK512, compl...    57  4e-06
GB_PL2:ATH010469 AJ010469 Arabidopsis thaliana mRNA for DEAD bo...    55  2e-05
GB_PL1:AB005232 AB005232 Arabidopsis thaliana genomic DNA, chro...    52  2e-04
GB_PL1:SC9346 Z48784 S.cerevisiae chromosome IV cosmid 9346. 8/97     52  2e-04
GB_IN:CELC43H8 AF098499 Caenorhabditis elegans cosmid C43H8. 10/98    50  6e-04
GB_IN:CELB0511 AF067608 Caenorhabditis elegans cosmid B0511. 5/98     50  6e-04
GB_PL2:SPBC21H7 AL023286 S.pombe chromosome II cosmid c21H7. 10/98    47  0.006
GB_IN:DDU78759 U78759 Dictyostelium discoideum IfdA (ifdA) mRNA...    47  0.006
GB_PR1:HSU49082 U49082 Human transporter protein (g17) mRNA, co...    47  0.006
GB_IN:DDHEL2A X81823 D.discoideum Hel2A mRNA for RNA helicase. ...    46  0.007
GB_PL2:SPBC4F6 AL031534 S.pombe chromosome II cosmid c4F6. 9/98       46  0.007
GB_PL2:ATH010475 AJ010475 Arabidopsis thaliana mRNA for DEAD bo...    46  0.009
GB_PL1:YSCH9986 U00027 Saccharomyces cerevisiae chromosome VIII...    45  0.021
GB_BA1:AB001488 AB001488 Bacillus subtilis genome sequence, 148...    43  0.063
GB_PL1:SCE9669 U18795 Saccharomyces cerevisiae chromosome V cos...    43  0.063
GB_BA1:BSUB0003 Z99106 Bacillus subtilis complete genome (secti...    43  0.063
GB_IN:AC004321 AC004321 Drosophila melanogaster DNA sequence (P...    43  0.082
GB_PL1:SCSPB4 X16147 S.cerevisiae spb4 gene for a probable rRNA...    43  0.082
GB_PL1:YSCF4682H D44600 Saccharomyces cerevisiae chromosome VI ...    43  0.082
GB_PL1:YSCCHRVIN D50617 Saccharomyces cerevisiae chromosome VI ...    43  0.082
GB_BA2:AE000458 AE000458 Escherichia coli K-12 MG1655 section 3...    42  0.14
GB_BA1:ECORECQ M30198 E.coli recQ gene complete cds, and pldA g...    42  0.14
GB_IN:AE001395 AE001395 Plasmodium falciparum chromosome 2, sec...    42  0.14
GB_IN:AF017777 AF017777 Drosophila melanogaster tweety (tty), f...    42  0.14
GB_BA1:ECOUW85 M87049 E. coli genomic sequence of the region fr...    42  0.14
GB_PL2:ATH010463 AJ010463 Arabidopsis thaliana mRNA for DEAD bo...    42  0.18
GB_BA2:U39711 U39711 Mycoplasma genitalium section 33 of 51 of ...    42  0.18
GB_PL1:TOBRDB10 D16247 Tobacco mRNA for RNA helicase like prote...    42  0.18
GB_PL1:SCYOR202W Z75110 S.cerevisiae chromosome XV reading fram...    42  0.18
GB_PL1:SCDED1 X57278 S.cerevisiae DED1 (SPP81) gene for putativ...    42  0.18
GB_PL2:ATH010466 AJ010466 Arabidopsis thaliana mRNA for DEAD bo...    42  0.18
GB_PL2:SPBC17D1 AL031322 S.pombe chromosome II cosmid c17D1. 8/98     41  0.24
GB_PL2:SPBC24C6 AL031786 S.pombe chromosome II cosmid c24C6. 9/98     41  0.24
GB_PL1:SPAC13F4 Z69379 S.pombe chromosome I cosmid c13F4. 1/98        41  0.24
GB_OV:DRRNAHELI Y12819 Danio rerio p110a mRNA for putative RNA ...    41  0.24
GB_IN:LBU19888 U19888 Leishmania braziliensis ribosomal DEAD bo...    41  0.32
GB_PL1:SCDB1G X55993 S. cerevisiae DBP1 gene. 2/97                    41  0.32
GB_PR3:HSAF000985 AF000985 Homo sapiens dead box, Y isoform (DB...    41  0.32
GB_PL2:AB010259 AB010259 Arabidopsis thaliana mRNA for DRH1, co...    41  0.32
```

```
GB_PL1:SCIRA1 X78937 S.cerevisiae (S288C) IRA1, YBR1118 and YBR...    41  0.32
GB_PL1:SCU43503 U43503 Saccharomyces cerevisiae chromosome XVI ...    41  0.32
GB_PR3:HSAF000984 AF000984 Homo sapiens dead box, Y isoform (DB...    41  0.32
GB_PL1:SCYBR142W Z36011 S.cerevisiae chromosome II reading fram...    41  0.32
GB_PL1:D89270 D89270 Schizosaccharomyces pombe mRNA, partial cd...    41  0.41
GB_PL2:ATH010457 AJ010457 Arabidopsis thaliana mRNA for DEAD bo...    41  0.41
GB_PL1:SPCC1795 AL022598 S.pombe chromosome III cosmid c1795. 4/98    41  0.41
GB_PL1:SPAC17G6 Z99162 S.pombe chromosome I cosmid c17G6. 9/97        41  0.41
GB_BA1:SPU10405 U10405 Streptomyces purpurascens ATCC 25489 Rdm...    41  0.41
GB_PL2:AF084222 AF084222 Schizosaccharomyces pombe putative DEA...    41  0.41
GB_PL1:AB012389 AB012389 Schizosaccharomyces pombe mRNA for Moc...    41  0.41
GB_PL1:AF025536 AF025536 Schizosaccharomyces pombe suppressor o...    41  0.41
GB_IN:AE001274 AE001274 Leishmania major chromosome 1, complete...    40  0.54
GB_STS:KLAJ9837 AJ229837 Kluyveromyces lactis DNA fragment for ...    40  0.71
GB_BA1:TTHERAGEN X97017 T.thermophilus DNA for RNA dependent AT...    40  0.71

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

GB_OV:XLRNAP54H X92421 X.laevis mRNA for RNA helicase p54. 4/97       37  4.7
GB_PL1:ATTIF4A1 X65052 A.thaliana mRNA for eukaryotic translati...    37  4.7
GB_RO:MUSDVH D14859 Mouse mRNA for drosophila vasa homologue, p...    37  4.7
GB_IN:CELF55F8 U80447 Caenorhabditis elegans cosmid F55F8. 12/96      37  6.2
GB_PL2:ATH010465 AJ010465 Arabidopsis thaliana mRNA for DEAD bo...    37  6.2
GB_HTG:AC005456 AC005456 *** SEQUENCING IN PROGRESS *** DS05130...    36  8.1
GB_PL1:AB011474 AB011474 Arabidopsis thaliana genomic DNA, chro...    36  8.1
GB_IN:DDHEL2B X81824 D.discoideum Hel2B mRNA for RNA helicase. ...    36  8.1

>GB_PL1:SCU72149 U72149 Saccharomyces cerevisiae putative RNA helicase
            (UF1) gene, complete cds. 10/96
            Length = 2977

 Score =  111 bits (274), Expect = 2e-22
 Identities = 68/159 (42%), Positives = 89/159 (55%), Gaps = 1/159 (0%)

Query: 2    NMCARVVDLPIVHWVVHFDCPDGVITYAHRAGRAARMNLPGFSLLFLTDQEQ-GFTKRLD 60
            ++ AR +D P V WVV  DCP+ V TY HR GR AR    G SL+ LT QEQ  F KRL+
Sbjct: 1519 DVVARGIDFPAVDWVVQVDCPEDVDTYIHRVGRCARYGKKGKSLIMLTPQEQEAFLKRLN 1698

Query: 61   EAKIDYQKKTVKLRTVVSIRQKLTELCITDTYIKHLAQKAIVSYAKSIHVQGDREVFPPA 120
              KI+  K  +K     SI+ +L  L   D  +K+L QKA +SY +SI+VQ D+EVF
Sbjct: 1699 ARKIEPGKLNIKQSKKKSIKPQLQSLLFKDPELKYLGQKAFISYVRSIYVQKDKEVF-KF 1875

Query: 121  SELNLTDIALSYGLASNINLSVGKQPGISTQHPASEQQMA 160
             EL   + A S GL     +   K  G+ T   A E++ A
Sbjct: 1876 DELPTEEFAYSLGLPGAPKI---KMKGMKTIEQAKERKNA 1986


  Database: genembl
    Posted date:  Jan 4, 1999 10:04 AM
  Number of letters in database: 1,199,477,030
  Number of sequences in database:  602,539

Lambda     K      H
   0.337    0.144    0.462

Gapped
Lambda     K      H
   0.270   0.0470    0.230
```

```
Matrix: BLOSUM62
Gap Penalties: Existence: 11, Extension: 1
Number of Hits to DB: -2084241564
Number of Sequences: 602539
Number of extensions: 30369647
Number of successful extensions: 230051
Number of sequences better than 10: 266
Number of HSP's better than 10.0 without gapping: 124
Number of HSP's successfully gapped in prelim test: 15
Number of HSP's that attempted gapping in prelim test: 229770
Number of HSP's gapped (non-prelim): 445
length of query: 1790
length of database: 399825676
effective HSP length: 54
effective length of query: 1736
effective length of database: 367288570
effective search space: 637612957520
frameshift window, decay const: 50,  0.1
T: 13
A: 40
X1: 15 ( 7.3 bits)
X2: 38 (14.8 bits)
X3: 64 (24.9 bits)
S1: 39 (21.7 bits)
S2: 81 (36.0 bits)
```

Especially pay attention to BLAST's Poisson distribution E value scores. These are the likelihoods (expectation) that the observed matches could be due to chance. Therefore, the smaller the number, the more significant. You should be able to see somewhat of a demarcation where the scores drop off between the significant hits and background noise.

Next, the output from TFastX; notice the commonallities and differences:

```
!!SEQUENCE_LIST 1.0


(Peptide) TFASTX of: input_75.rsf{contig3000_frame1}  from: 1 to: 1790  July 15,
 1999 11:38


 TO: genembl:*  Sequences:    605,925  Symbols: 1e9 + 198,161,167  Word Size: 2

 Sequences too short to analyze: 27 (123 symbols)
 Databases searched:
   GenBank, Release 112.0, Released on 15Jun1999, Formatted on 1Jul1999
   GenBank, Release 110.0, Released on 14Dec1998, Formatted on 14Dec1998
   EMBL, Release 56.0, Released on 16Sep1998, Formatted on 15Dec1998

 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 996 search set sequences
 Each inset symbol represents 5 search set sequences
 z-scores computed from opt scores

z-score obs    exp
        (=)    (*)

< 20   2081      0:===
  22     56      0:=
  24    140      1:*
  26    302     13:*
  28    631    137:*
  30   1617    834:*=
  32   3739   3225:===*
  34   9373   8747:========*=
  36  21227  17964:==================*===
  38  37029  29689:==============================*========
  40  52538  41413:========================================*===========
  42  57927  50622:==================================================*========
  44  59721  55841:============================================================*===
  46  58101  56876:============================================================*=
  48  53270  54452:==========================================================*
  50  47116  49688:============================================== *
  52  39827  43684:=======================================   *
  54  33829  37313:==================================   *
  56  27238  31168:============================   *
  58  22731  25588:========================  *
  60  17579  20728:==================  *
  62  13629  16618:==============  *
  64  10765  13216:===========  *
  66   8742  10446:========= *
  68   6731   8216:======= *
  70   4826   6439:===== *
  72   3565   5031:==== *
  74   2667   3923:===*
  76   2124   3053:===*
  78   1575   2373:==*
  80   1160   1843:=*
  82    924   1410:=*
  84    734   1117:=*
  86    550    864:*
  88    431    669:*
  90    293    517:*
  92    217    400:*         :=======================================*
  94    192    310:*         :=======================================*
  96    136    240:*         :============================           *
  98    103    185:*         :====================              *
 100     79    143:*         :================           *
 102     77    111:*         :================      *
 104     61     86:*         :=============    *
 106     50     66:*         :==========   *
 108     34     51:*         :=======   *
 110     31     40:*         :=======*
 112     18     31:*         :====  *
 114     20     24:*         :====*
 116     11     18:*         :===*
 118     13     14:*         :==*
>120     95     11:*         :==*================
```

```
Joining threshold: 41, opt. threshold: 29, opt. width:  16, reg.-scaled


The best scores are:                 strand init1 initn   opt    z-sc E(605486).
.

GB_PL1:SCU72149
! U72149 Saccharomyces cerevisiae put...(f)   188   328   386   441.4  5.3e-17
GB_PL1:SCYJL033W
! Z49308 S.cerevisiae chromosome X re...(f)   188   324   379   433.1  1.6e-16
GB_PR3:HSU28042
! U28042 Human DEAD box RNA helicase-...(f)   290   361   331   375.4  2.6e-13
GB_PR1:HSRNAHELC
! X98743 H.sapiens mRNA for RNA helic...(f)   123   149   257   287.3  2.1e-08
GB_PL1:SPAC1F7  Strand: -
! Z67998 S.pombe chromosome I cosmid ...(r)   125   125   257   272.1  1.5e-07
GB_PL1:SC8175  Strand: -
! X80836 S.cerevisiae chromosome XIII...(r)   128   128   246   264.0  4.1e-07
GB_IN:DMU84552
! U84552 Drosophila melanogaster heli...(f)   111   136   220   245.5  4.4e-06
GB_PL2:ATH010469
! AJ010469 Arabidopsis thaliana mRNA ...(f)   124   124   176   196.6  0.0023
GB_IN:CEZK512  Strand: -
! Z22177 Caenorhabditis elegans cosmi...(r)   136   202   179   178.6   0.023
GB_PL1:SCSPB4
! X16147 S.cerevisiae spb4 gene for a...(f)   109   133   164   178.2   0.025
GB_PL2:ATH010468
! AJ010468 Arabidopsis thaliana mRNA ...(f)   153   153   155   172.4   0.052
GB_PR1:HSU12968
! U12968 Human clone S3/1 dinucleotid...(f)    56   105   151   165.7    0.12
GB_PL1:AB005232  Strand: -
! AB005232 Arabidopsis thaliana genom...(r)   152   189   172   165.3    0.13
GB_IN:CELC43H8
! AF098499 Caenorhabditis elegans cos...(f)   112   112   158   162.6    0.18
GB_PL1:SCNSR1
! X57185 Yeast NSR1 gene for nuclear ...(f)    62    90   144   156.4     0.4
GB_PR1:HSU49082
! U49082 Human transporter protein (g...(f)    75   125   145   155.3    0.46
GB_PAT:I15828
! I15828 Sequence 3 from patent US 54...(f)    62    90   144   155.3    0.46
GB_PL1:SC9346  Strand: -
! Z48784 S.cerevisiae chromosome IV c...(r)   150   185   155   154.1    0.54
GB_PL1:SCYGR159C  Strand: -
! Z72944 S.cerevisiae chromosome VII ...(r)    62    90   144   153.7    0.57
GB_PL2:ATH010475
! AJ010475 Arabidopsis thaliana mRNA ...(f)   138   197   144   153.1    0.61
GB_IN:DDU78759
! U78759 Dictyostelium discoideum Ifd...(f)   123   123   140   153.0    0.62
GB_IN:DDHEL2A
! X81823 D.discoideum Hel2A mRNA for ...(f)    92    92   140   151.5    0.76
GB_STS:KLAJ9837  Strand: -
! AJ229837 Kluyveromyces lactis DNA f...(r)    65    65   128   145.6     1.6
GB_BA1:ECORECQ
! M30198 E.coli recQ gene complete cd...(f)   103   186   136   144.0       2
GB_RO:AF103809

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

GB_OV:DRRNAHELI
```

```
! Y12819 Danio rerio p110a mRNA for p...(f)   122   153   128   133.6     7.5
GB_PL1:D89270
! D89270 Schizosaccharomyces pombe mR...(f)    56    56   124   133.4     7.7
GB_PL2:SPBC4F6  Strand: -
! AL031534 S.pombe chromosome II cosm...(r)   140   195   142   133.3     7.8
GB_PL2:ATH010467
! AJ010467 Arabidopsis thaliana mRNA ...(f)    86    86   122   133.1       8
GB_RO:MMU46690
! U46690 Mus musculus ATP-dependent R...(f)   115   115   124   132.7     8.4
\\End of List

! Distributed over 1 thread.
!      Start time: Thu Jul 15 10:34:03 1999
! Completion time: Thu Jul 15 11:38:35 1999

! CPU time used:
!        Database scan:  0:58:49.3
! Post-scan processing:  0:00:00.5
!       Total CPU time:  0:58:49.8
! Output File: /users/thompson/working/Giardia/contig3000_frame1_75.tfastx
```

Had we not chosen to suppress aligning the results, the TFastX output would also show the sequence alignment for as many pairs as we were to specfify, in which case the beginning and ending alignment points could be used to go back to the original nucleotide entries to check whether the match-ups correspond to actually translated areas. Notice that the output file is an acceptable GCG list file that can serve as input to other programs such as their multiple sequence alignment program PileUp. A histogram of the score distribution is also displayed in the FastA outputs. This can be helpful to get a feeling for the statistical significance of the search and in ascertaining whether you ran your search list large enough. The more closely the curve of asteriks follows the actual distribution, the better the statistics. 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 will describe below. 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 changed after the *opt* dynamic programming and normalization pass. A good example is shown in the above output under the GB_PL1:AB005232 entry. It scored a relatively good *initn* score of 189 versus its somewhat mediocore final *z-score* of 165.3. This point underscores the importance of using multiple algorithms.

The Expectation function, *E()*, is the most important column. It is very similar to the Poisson style 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 rule-of-thumb, for a search against a database of about 10,000 sequences, as long as optimization is not turned off, *E()* scores of less than 0.01 are almost certainly homologous, and scores between 1 to 10 may be, although these guidelines can be skewed by compositional biases.

Next, the abridged example FastA output file:

```
!!SEQUENCE_LIST 1.0
```

```
 (Peptide) FASTA of: input_77.rsf{contig3000_frame1}  from: 1 to: 1790  July 15,
1999 14:22

TO: SwissProtPlus:*  Sequences:    254,782  Symbols:   82,009,484  Word Size: 2

 Databases searched:
   SWISS-PROT, Release 36.0, Released on 18Jul1998, Formatted on 18Aug1998
   SPTREMBL, Release 8.0, Released on 21Nov1998, Formatted on 15Dec1998

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

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

The best scores are:                    init1 initn   opt    z-sc E(254390)..

SW:DBP4_YEAST
! P20448 saccharomyces cerevisiae (ba...  188   355   398   428.1  1.2e-16
SW:DDXX_HUMAN
! Q13206 homo sapiens (human). probab...  290   379   340   364.6  4.3e-13
SW:YAK2_SCHPO
! Q09916 schizosaccharomyces pombe (f...  125   207   266   287.1  8.9e-09
SP_HUM:Q92732
! Q92732 homo sapiens (human). rna he...  123   123   266   286.8  9.3e-09
SW:YOQ2_CAEEL
! P34640 caenorhabditis elegans. puta...  136   136   258   278.5  2.7e-08
SP_PL:O49530
! O49530 arabidopsis thaliana (mouse-...  125   165   257   276.8  3.3e-08
SW:HAS1_YEAST
! Q03532 saccharomyces cerevisiae (ba...  128   164   255   276.0  3.7e-08
SP_IN:O77001
! O77001 drosophila melanogaster (fru...  111   111   229   246.3  1.7e-06
SP_PL:O48546
! O48546 arabidopsis thaliana (mouse-...   73   103   202   219.0  5.5e-05
SP_PL:Q42400
! Q42400 arabidopsis thaliana (mouse-...   73   103   199   215.7  8.4e-05
SP_IN:O61815
! O61815 caenorhabditis elegans. b051...  112   146   192   207.2  0.00025
SW:SPB4_YEAST
! P25808 saccharomyces cerevisiae (ba...  109   109   189   203.6  0.0004
SP_HUM:Q99624
! Q99624 homo sapiens (human). transp...   75   106   176   190.6  0.0021
SW:YSPK_CAEEL
! Q19425 caenorhabditis elegans. hypo...   82    82   169   181.8  0.0065
SP_FUN:O74764
! O74764 schizosaccharomyces pombe (f...   36    36   167   179.8  0.0084
SW:YBI9_YEAST
! P38176 saccharomyces cerevisiae (ba...   46    77   166   179.8  0.0085
SW:MS16_YEAST
! P15424 saccharomyces cerevisiae (ba...  150   150   164   176.0   0.014
SW:Y308_MYCGE
! P52271 mycoplasma genitalium. proba...  123   123   148   161.6   0.087
SP_IN:O45198
! O45198 caenorhabditis elegans. w09g...   61    61   146   160.1    0.11
SP_PL:O22719
! O22719 arabidopsis thaliana (mouse-...   53    53   146   159.3    0.12
SP_IN:O17275
```

```
! O17275 caenorhabditis elegans. t27a...   64    64   146   158.9    0.12
SP_FUN:O74393
! O74393 schizosaccharomyces pombe (f...  140   140   145   155.6    0.19
SP_IN:P90529
! P90529 dictyostelium discoideum (sl...  123   123   142   155.4    0.19
SW:VIE3_MCMVS
! P29832 murine cytomegalovirus (stra...   57    86   143   153.8    0.24
SP_IN:Q21205
! Q21205 caenorhabditis elegans. k04c...   55    93   142   153.6    0.24
SP_OV:O13098
! O13098 xenopus laevis (african claw...   41    41   145   153.6    0.24
SP_BA:O68668
! O68668 bacillus megaterium. gas ves...   49    49   138   152.9    0.27
SW:DBP2_YEAST
! P24783 saccharomyces cerevisiae (ba...   90    90   140   151.2    0.33
SP_IN:Q23909
! Q23909 dictyostelium discoideum (sl...   92   121   140   150.8    0.35
SP_PL:O23506
! O23506 arabidopsis thaliana (mouse-...  129   129   140   149.8    0.39
SW:YG1F_YEAST
! P53214 saccharomyces cerevisiae (ba...   56    56   137   147.9     0.5
SW:YAXB_SCHPO
! Q10202 schizosaccharomyces pombe (f...   78    78   136   146.5     0.6
SP_IN:Q21736
! Q21736 caenorhabditis elegans. r05d...  115   115   136   146.5     0.6
SW:RECQ_ECOLI
! P15043 escherichia coli. atp-depend...  103   103   136   146.2    0.62
SP_OM:P79801
! P79801 microcebus murinus. presenil...   51    78   134   146.0    0.64
SW:DB10_NICSY
! P46942 nicotiana sylvestris (wood t...  127   127   134   144.1    0.82
SW:DBP8_YEAST
! P38719 saccharomyces cerevisiae (ba...  103   103   132   144.0    0.83
SP_BA:O83483
! O83483 treponema pallidum. conserve...   65    65   131   143.9    0.84
SP_IN:Q21472
! Q21472 caenorhabditis elegans. simi...   49    49   131   143.4     0.9
SP_RO:O88832
! O88832 mus musculus (mouse). garp34...   45    45   129   142.8    0.96
SP_HUM:Q13061
! Q13061 homo sapiens (human). triadi...   71    98   133   141.9     1.1
SP_HUM:O00580
! O00580 homo sapiens (human). cerebe...   69    69   134   140.5     1.3
SW:NUCL_XENLA
! P20397 xenopus laevis (african claw...   95   124   131   140.4     1.3

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

SP_FUN:O60080
! O60080 schizosaccharomyces pombe (f...  116   116   116   124.7     9.8
\\End of List

! Distributed over 1 thread.
!      Start time: Thu Jul 15 14:19:57 1999
! Completion time: Thu Jul 15 14:22:48 1999

! CPU time used:
!        Database scan:  0:02:48.5
! Post-scan processing:  0:00:00.7
```

```
!       Total CPU time:  0:02:49.2
! Output File: /users/thompson/working/Giardia/contig3000_frame1_77.fasta
```

So that you don't need to run similarity searches on the rest of the frames I will include those results here. I list abridged TBLASTN output first for frames 2 and 3 in the forward direction and then frames 1, 2, and 3 for the reverse strand. Next are the abridged TFastX output files for the same set.

```
TBLASTN 2.0.5 [May-5-1998]


Database: genembl         602,539 sequences; 1,199,477,030 total letters

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= CONTIG3000_FRAME2  (1789 letters)

                                                                  Score     E
Sequences producing significant alignments:                       (bits)  Value

GB_PR3:HS75C10 AJ011931 Homo sapiens chromosome 21q22.3, PAC cl...    38  2.1

>GB_PR3:HS75C10 AJ011931 Homo sapiens chromosome 21q22.3, PAC clone
             75C10 complete sequence bases 1..98443. 11/98
             Length = 98443

 Score = 38.3 bits (87), Expect = 2.1
 Identities = 23/70 (32%), Positives = 32/70 (44%)

Query: 588   FKVWSKGFRKRAMRWGLSAHTLQCRREKRLLRFPHPIGARLPAQHSTPRCLLSTLQMRRL 647
              F++      + A     S+H L C+R +  L   P P     P  H TP C+ +  +MR L
Sbjct: 57415 FRLAPDAHEREACSEASSSHPLPCQRLQHPLPSPLPSLPCGPPAHPTPVCMCTVTEMRVL 57236

Query: 648   EHNSGTFNRR 657
                 SGT  RR
Sbjct: 57235 GRRSGTVVRR 57206

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

Query= CONTIG3000_FRAME3  (1789 letters)

GB_BA1:XCU33548 U33548 Xanthomonas campestris hrpB pathogenicit...    40  0.55
GB_BA1:XANHRPA1A M99173 Xanthomonas campstris HrpA1 gene, compl...    40  0.55

>GB_BA1:XCU33548 U33548 Xanthomonas campestris hrpB pathogenicity
             locus proteins HrpB1, HrpB2, HrpB3, HrpB4, HrpB5, HrpB6,
             HrpB7, HrpB8, HrpA1, and ORF62 genes, complete cds. 9/96
             Length = 8429

 Score = 40.2 bits (92), Expect = 0.55
 Identities = 25/65 (38%), Positives = 34/65 (51%), Gaps = 2/65 (3%)

Query: 976  HPSGDVMRCHRLVSHHPSPCPGL--HNAVDSIQQDERENRHNCILLRWLWCNIAVHGRSH 1033
             HP G+V++  R VS HP  C  L  H+  D +Q+     RH+  L+R L    A H  +
Sbjct: 7521 HPGGEVIKIGRCVSAHPDECLLLVVHDRFDVVQRRNLGRRHDLCLVRRLQRRHARHQVAP 7342
```

```
Query: 1034 VIPQHRR 1040
             I QH R
Sbjct: 7341 RISQHGR 7321

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

Query= CONTIG3000_R_FRAME1  (1790 letters)

GB_PR2:AC004542 AC004542 Homo sapiens PAC clone DJ430N08 from 2...    37  4.8

>GB_PR2:AC004542 AC004542 Homo sapiens PAC clone DJ430N08 from
             22q12.1-qter, complete sequence. 4/98
             Length = 134914

 Score = 37.1 bits (84), Expect = 4.8
 Identities = 19/60 (31%), Positives = 27/60 (44%)

Query: 626   RLWRGPCRYRTRHIATWAAQCLEAVSFQGPCEYVSXDWSVCPXKTTSQARRRLXNYSPCT 685
             R W  PC  +T    +WA++C  AV+ Q PC+        CP       + RL    PC+
Sbjct: 23052 RPWPSPCGLQTSGGRSWASECRHAVASQWPCQ-----GRDCPIPRQ*MRKLRL*EVKPCS 23216

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

Query= CONTIG3000_R_FRAME2  (1789 letters)

 ***** No hits found ******

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

Query= CONTIG3000_R_FRAME3  (1789 letters)

GB_IN:GIACPA1 L49236 Giardia duodenalis multigene unit CPA1 enc...    63  7e-08
GB_IN:GIACPA2 L49298 Giardia duodenalis multigene unit CPA2 enc...    62  1e-07
GB_PR1:HSANKB440 Z26634 H.sapiens mRNA for ankyrin B (440 kDa)....    54  3e-05
GB_RO:RNU65916 U65916 Rattus norvegicus ankyrin mRNA, membrane ...    54  3e-05
GB_PR3:HSBRANK2 X56958 Human mRNA for brain ankyrin (brank-2). ...    54  3e-05
GB_IN:CET28D6 Z81134 Caenorhabditis elegans cosmid T28D6, compl...    48  0.003
GB_HTG:CEY47D3 Z98865 Caenorhabditis elegans DNA *** SEQUENCING...    48  0.003
GB_PL1:SC9916 Z48952 S.cerevisiae chromosome XIII cosmid 9916. ...    46  0.012
GB_PR3:AF082557 AF082557 Homo sapiens TRF1-interacting ankyrin-...    44  0.037
GB_PR3:AF082556 AF082556 Homo sapiens TRF1-interacting ankyrin-...    44  0.037
GB_HTG:CEY43C5 AL021449 Caenorhabditis elegans DNA *** SEQUENCI...    43  0.11
GB_IN:CER10H10 Z70686 Caenorhabditis elegans cosmid R10H10, com...    42  0.19
GB_RO:RNU50185 U50185 Rattus norvegicus kidney protein phosphat...    42  0.19
GB_RO:S74907 S74907 PP1M M110=protein phosphatase 1M 110 kda re...    42  0.19
GB_PR2:AB003062 AB003062 Homo sapiens MYPT2 mRNA, complete cds....    40  0.54
GB_OV:CHK130KDA D37985 Chicken mRNA for 133 kDa myosin-binding ...    40  0.71
GB_OV:CHK130KDB D37986 Chicken mRNA for 130 kDa myosin-binding ...    40  0.71
GB_PL1:VFPOTCHAN Y10579 V.faba mRNA for potassium channel. 8/97       39  0.93
GB_RO:MUS25RNASE L10382 Mus musculus 2-5A-dependent RNase gene,...    39  0.93
GB_RO:MMU010902 AJ010902 Mus musculus MRNA for inversin. 10/98        39  0.93
GB_PR2:D87930 D87930 Homo sapiens mRNA for myosin phosphatase t...    39  0.93
GB_PL1:ATFCA1 Z97336 Arabidopsis thaliana DNA chromosome 4, ESS...    39  1.6

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

GB_IN:CEF02A9 Z19555 Caenorhabditis elegans cosmid F02A9, compl...    36  8.1

>GB_IN:GIACPA1 L49236 Giardia duodenalis multigene unit CPA1 encoding
```

```
            cysteine-rich protein, protein kinase and ankyrin
            homolog. 5/96
            Length = 6822

 Score = 60.1 bits (143), Expect = 6e-07
 Identities = 44/138 (31%), Positives = 71/138 (50%), Gaps = 10/138 (7%)

Query: 190  AAVASGDQPLVAGYSPRFKKSVNENGMTALMVAAQTGNTELAAILLKDEQQIKD----SQ 245
            +A A+G   +V     +     + NG TALM+AA+ G+ E   +LL+ E  +K     S
Sbjct: 6230 SAAANGHAEIVELLLEKEGGMRDRNGKTALMIAAEKGHPECIKLLLEKEGGMKKNDFFSN 6409

Query: 246  GRTALIYAIQSGQSQLCRLLATRELDTSNLKSQSPFSVAIQNDAHDCLEAMLQAVGPVKV 305
            G TAL+ A ++G+ +  RLL  +E         +   +A QN   DC+E +L+  G ++
Sbjct: 6410 GGTALMCAARNGRPECVRLLLDKEGGMKGSNGGTALMIAAQNGHSDCVEILLEKEGGMQE 6589

Query: 306  VD------NPLLTATSVGCLRCLRILLE 327
                      L+ A S   + C R+L E
Sbjct: 6590 GGFFSNGWTALMWAVSCSQIECARLLAE 6673

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

 Score = 51.2 bits (120), Expect = 3e-04
 Identities = 25/86 (29%), Positives = 49/86 (56%)

Query: 212  NENGMTALMVAAQTGNTELAAILLKDEQQIKDSQGRTALIYAIQSGQSQLCRLLATRELD 271
            ++ GMTA M AAQ G+     +L++ E+ +KD  G TAL++A  +G  ++ +++A  E
Sbjct: 4907 DKQGMTAFMHAAQQGHGRPVELLVEKEKGLKDKNGWTALMHAAHNGHPEIVKIIAPHEHG 5086

Query: 272  TSNLKSQSPFSVAIQNDAHDCLEAML 297
              +L   +   +A Q  + + + ++L
Sbjct: 5087 LQDLHGHTALMIAAQQGSLEVVKLLL 5164


 Score = 51.5 bits (121), Expect = 2e-04
 Identities = 55/236 (23%), Positives = 101/236 (42%), Gaps = 11/236 (4%)

Query: 212  NENGMTALMVAAQTGNTELAAILLKDEQQIKDSQGRTALIYAIQSGQSQLCRLLATRELD 271
            ++NG TALM AA  G+ E+  I+   E  ++D  G TAL+ A Q G  ++ +LL   E
Sbjct: 5000 DKNGWTALMHAAHNGHPEIVKIIAPHEHGLQDLHGHTALMIAAQQGSLEVVKLLLDHEKG 5179

Query: 272  TSNLKSQSPFSVAIQNDAHDCLEAMLQAVGPVKVVD-NPLLTATSVGCLRCLRILLEYGQ 330
              + +  +    A++N      E ++    P        L+ A + G    R+L+  +
Sbjct: 5180 LRDKQHHNALYHALENGHLGVAEMIIPYEDPTDGNGVTALMRAAARGDTEMVRLLIPVQK 5359

Query: 331  CFEMSEFDXXXXXXXXXXXXXCTELVSWKHEITDIIAIANKAPQ----------ISKEK 380
               M + D               T +V  KHE +    + + A              K+K
Sbjct: 5360 --GMKDKDGNTAFMHALKNKHIDTGVVLGKHEDSSWTPLMHAAADGGIEAVKKHLSDKDK 5533

Query: 381  FHNTIDTSTRALVESHFVDISEVKHDLNERIAHLLAENKELRAMLKNLEANNKVLRAELA 440
             +NT +T+      +   +I E+    +E         K + A+++  + N+      LA
Sbjct: 5534 KNNTGETALMIAARARHRNIVELLDPTDE---------KGVTALMRAADRNDPAAVKALA 5686

Query: 441  DVRTQQE 447
             ++T Q+
Sbjct: 5687 PLQTGQK 5707


 Score = 57.4 bits (136), Expect = 4e-06
 Identities = 33/116 (28%), Positives = 64/116 (54%), Gaps = 2/116 (1%)
```

```
Query: 212  NENGMTALMVAAQTGNTELAAILLKDEQQIKDSQGRTALIYAIQSGQSQLCRLLATRELD 271
            NE+G TALM+AA+  + +   +LL+ E  ++D+ G+TAL+ A + G +++  +L  +E
Sbjct: 6017 NEDGETALMIAAEGSHADCVKLLLEKEGSMRDNNGQTALVTAAEKGHTKIVEILLEKEGG 6196

Query: 272  TSNLKSQSPFSVAIQNDAHDCLEAMLQAVGPVKVVD--NPLLTATSVGCLRCLRILLE 327
               +    +    A  N   + +E +L+  G ++  +    L+ A   G   C++LLE
Sbjct: 6197 LRDNGGWTALMSAAANGHAEIVELLLEKEGGMRDRNGKTALMIAAEKGHPECIKLLLE 6370


 Score = 63.2 bits (151), Expect = 7e-08
 Identities = 36/116 (31%), Positives = 67/116 (57%), Gaps = 6/116 (5%)

Query: 212  NENGMTALMVAAQTGNTELAAILLKDEQQIKDSQGRTALIYAIQSGQSQLCRLLATRELD 271
            + NG TAL+ AA+ G+T++  ILL+ E  ++D+ G TAL+ A  +G +++  LL  +E
Sbjct: 6110 DNNGQTALVTAAEKGHTKIVEILLEKEGGLRDNGGWTALMSAAANGHAEIVELLLEKEGG 6289

Query: 272  TSNLKSQSPFSVAIQNDAHDCLEAMLQAVGPVKVVD------NPLLTATSVGCLRCLRIL 325
               +   ++   +A +    +C++ +L+  G +K  D        L+ A   G   C+R+L
Sbjct: 6290 MRDRNGKTALMIAAEKGHPECIKLLLEKEGGMKKNDFFSNGGTALMCAARNGRPECVRLL 6469

Query: 326  LE 327
            L+
Sbjct: 6470 LD 6475


 Score = 55.8 bits (132), Expect = 1e-05
 Identities = 36/114 (31%), Positives = 60/114 (52%), Gaps = 2/114 (1%)

Query: 214  NGMTALMVAAQTGNTELAAILLKDEQQIKDSQGRTALIYAIQSGQSQLCRLLATRELDTS 273
            +G TALM A   G+ ++  ILL++E  ++D  GRTAL  A +S  S   RLL  +E
Sbjct: 5744 HGGTALMRAVAYGHAKIVEILLEEEAGMQDIFGRTALHLAAESNNSDCVRLLVKKEGGMQ 5923

Query: 274  NLKSQSPFSVAIQNDAHDCLEAMLQAVGPVKVVD--NPLLTATSVGCLRCLRILLE 327
                 +  ++A Q    +C++ +L+  G ++  D    L+ A       C+R+LLE
Sbjct: 5924 TSYGSTALTIAAQRGHLECVKLLLEKEGGMQNEDGETALMIAAEGSHADCVKLLLE 6091

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

And now the abridged TFastX output files:

```
 (Peptide) TFASTX

of: input_44.rsf{contig3000_frame2}  from: 1 to: 1789  June 27, 1999 02:47

TO: GenEMBLPlus:* Sequences: 3,049,812 Symbols: 2e9 + 170,427,914 Word Size: 2

 Sequences too short to analyze: 27 (120 symbols)
 Databases searched:
   GenBank, Release 110.0, Released on 14Dec1998, Formatted on 14Dec1998
   GenBank_Tags, Release 110.0, Released on 14Dec1998, Formatted on 15Dec1998
   EMBL, Release 56.0, Released on 16Sep1998, Formatted on 15Dec1998
   EMBL_Tags, Release 56.0, Released on 16Sep1998, Formatted on 15Dec1998

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

```
The best scores are:                  strand init1 initn   opt    z-sc E(3049546)..

\\End of List
! No sequences found with E() < 10.00

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

of: input_45.rsf{contig3000_frame3}  from: 1 to: 1789  June 27, 1999 02:51

GB_EST14:AA595274
! AA595274 no35a02.s1 NCI_CGAP_Pr23 H...(f)    90   116   167   168.0    0.46
\\End of List

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

of: input_46.rsf{contig3000_r_frame1}  from: 1 to: 1790  June 27, 1999 03:21

\\End of List
! No sequences found with E() < 10.00

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

of: input_47.rsf{contig3000_r_frame2}  from: 1 to: 1789  June 27, 1999 03:17

GB_VI:IBU30819
! U30819 Infectious bursal disease vi...(f)    46    72   148   152.4     3.4
\\End of List

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

of: input_48.rsf{contig3000_r_frame3}  from: 1 to: 1789  June 27, 1999 02:55

GB_GSS2:AQ048799
! AQ048799 cLM-10a7-u cLM Giardia int...(f)   122   184   190   221.2  0.0005
GB_GSS2:AQ047478
! AQ047478 cLM-1c6-u cLM Giardia inte...(f)   162   162   188   220.0  0.00058
GB_GSS2:AQ047866
! AQ047866 cLM-3g8-u cLM Giardia inte...(f)   147   175   182   211.1  0.0018
GB_IN:GIACPA2
! L49298 Giardia duodenalis multigene...(f)   152   152   193   209.9  0.0021
GB_IN:GIACPA1
! L49236 Giardia duodenalis multigene...(f)   113   180   191   208.7  0.0025
GB_GSS2:AQ048634  Strand: -
! AQ048634 cLM-8h6-t cLM Giardia inte...(r)   145   145   173   200.4  0.0072
GB_GSS2:AQ049321  Strand: -
! AQ049321 cLM-13f8-t cLM Giardia int...(r)    69   107   171   198.4  0.0093
GB_RO:RNU65916
! U65916 Rattus norvegicus ankyrin mR...(f)    80   138   171   190.1   0.027
GB_PR3:HSBRANK2
! X56958 Human mRNA for brain ankyrin...(f)    80   197   171   184.9   0.052
GB_BA2:AB012226  Strand: -
! AB012226 Vibrio alginolyticus gene ...(r)    79   162   167   183.4   0.064
GB_GSS2:AQ047580  Strand: -
! AQ047580 cLM-2a10-t cLM Giardia int...(r)   147   147   158   181.5   0.081
GB_PL1:SCU72149  Strand: -
! U72149 Saccharomyces cerevisiae put...(r)    96   165   164   180.6    0.09
GB_GSS2:AQ048941  Strand: -
! AQ048941 cLM-11a8-t cLM Giardia int...(r)   153   153   156   180.1   0.097
```

```
GB_GSS2:AQ049113
! AQ049113 cLM-12b9-t cLM Giardia int...(f)    91    91   154   178.5    0.12
GB_PL1:SCYJL033W  Strand: -
! Z49308 S.cerevisiae chromosome X re...(r)    84   153   161   177.0    0.14
GB_PR1:HSANKB440
! Z26634 H.sapiens mRNA for ankyrin B...(f)    75   170   166   174.7    0.19
GB_GSS2:AQ047571
! AQ047571 cLM-1h4-u cLM Giardia inte...(f)   114   114   152   174.6     0.2
GB_GSS2:AQ048114
! AQ048114 cLM-5e11-u cLM Giardia int...(f)   131   131   150   172.2    0.27
GB_GSS2:AQ049051
! AQ049051 cLM-11g5-t cLM Giardia int...(f)   129   153   149   170.6    0.33
GB_GSS2:AQ047729
! AQ047729 cLM-2g9-t cLM Giardia inte...(f)   133   133   147   169.3    0.39
GB_GSS2:AQ048414
! AQ048414 cLM-7e6-t cLM Giardia inte...(f)   100   100   147   169.1     0.4
GB_GSS2:AQ049706  Strand: -
! AQ049706 cGR-64c6-t cGR Giardia int...(r)   145   145   145   168.3    0.44
GB_GSS2:AQ049672  Strand: -
! AQ049672 cGR-63d4-t cGR Giardia int...(r)   120   120   140   162.6    0.91
GB_GSS2:AQ048035  Strand: -
! AQ048035 cLM-4h7-u cLM Giardia inte...(r)   119   119   141   161.0     1.1
GB_GSS2:AQ047774  Strand: -
! AQ047774 cLM-3b2-t cLM Giardia inte...(r)   118   118   141   160.8     1.2
GB_RO:AF069525
! AF069525 Rattus norvegicus 190 kDa ...(f)    89   121   151   160.5     1.2
GB_RO:MUSANK3B
! L40632 Mus musculus epithelial anky...(f)    89   169   151   159.7     1.3
GB_RO:AF102552
! AF102552 Rattus norvegicus 270 kDa ...(f)    89   145   151   159.1     1.4
GB_GSS2:AQ049413  Strand: -
! AQ049413 cLM-14c8-t cLM Giardia int...(r)   114   114   137   156.3       2
GB_GSS2:AQ048219
! AQ048219 cLM-6c8-t cLM Giardia inte...(f)   118   118   135   155.7     2.2
GB_EST13:AA576640
! AA576640 nm72g02.s1 NCI_CGAP_Co9 Ho...(f)    87    87   135   155.5     2.3
GB_PR1:HSU13616
! U13616 Human ankyrin G (ANK-3) mRNA...(f)    89   116   151   155.4     2.3
GB_EST20:AI074225
! AI074225 oz85a05.x1 Soares_senescen...(f)    71    96   134   154.7     2.5
GB_GSS2:AQ049513
! AQ049513 cGR-10g11-t cGR Giardia in...(f)   117   117   132   154.3     2.7
GB_GSS2:AQ048456
! AQ048456 cLM-7g5-t cLM Giardia inte...(f)   101   101   134   151.8     3.7
GB_EST19:AI036566
! AI036566 ue17f11.y1 Sugano mouse em...(f)    74    74   132   151.4     3.9
GB_GSS2:AQ047654  Strand: -
! AQ047654 cLM-2d2-u cLM Giardia inte...(r)   122   122   132   151.0     4.1
\\End of List

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

The only obviously significant hits were found on frame 1 in the forward direction and frame 3 in the reverse direction. Although frame 3 forward did have an interesting possibility that should probably be checked out. And, back in the content section of the exercise, I noticed that the mid-section of forward frame 1 had high TestCode and CodonPreference coding potential. It is interesting that the long ORF there doesn't show obvious similarity to anything in the database. Though, perhaps, the higher similarity of the first ORF on

frame 1 to helicases, is 'swamping' out the signal from the second ORF. To test this you should run just that second ORF alone in a similarity search.

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

What Next? Comparisons, interpretations, and further analyses.

Interpreting Results — what is significant.

It should be obvious by now that we have at least two genes in contig3000, one in forward frame 1 and one in reverse frame 3. Where they begin and end needs to be ascertained, and the possiblility of any other genes in the sequence needs to be addressed. How can we make any sense out of all of these individual analyses? The strength lies in a combinatorial approach.

Let's try to get a handle on the two genes that we've tentatively identified. I'll illustrate with the first one, contig3000_frame1. Look at the search results for this sequence — every one identified the same *Saccharomyces cerevisiae* putative RNA helicase (UF1) gene as the most significantly similar sequence to compare ours to. Therefore, pull this sequence, SW:DBP4_YEAST, equivalent to GB:SCU72149, into your SeqLab display by going to the "**File**" "**Add sequences from**" "**Databases. . .**" menu. Merely type "**SW:DBP4_YEAST"** in the "**Database Specification:**" box in the "**SeqLab Database Browser**" and then press the "**Add to Main Window**" button. Let's also try to get a control, "twilight zone," sequence to compare contig3000_frame1 with for comparison purposes. From the FastA output file SW:NUCL_XENLA appears a likely candidate. This sequence scored a FastA expectation value of 1.3; it should provide a good comparison. "**Add**" this sequence also. "**Close**" the browser box after adding the sequence.

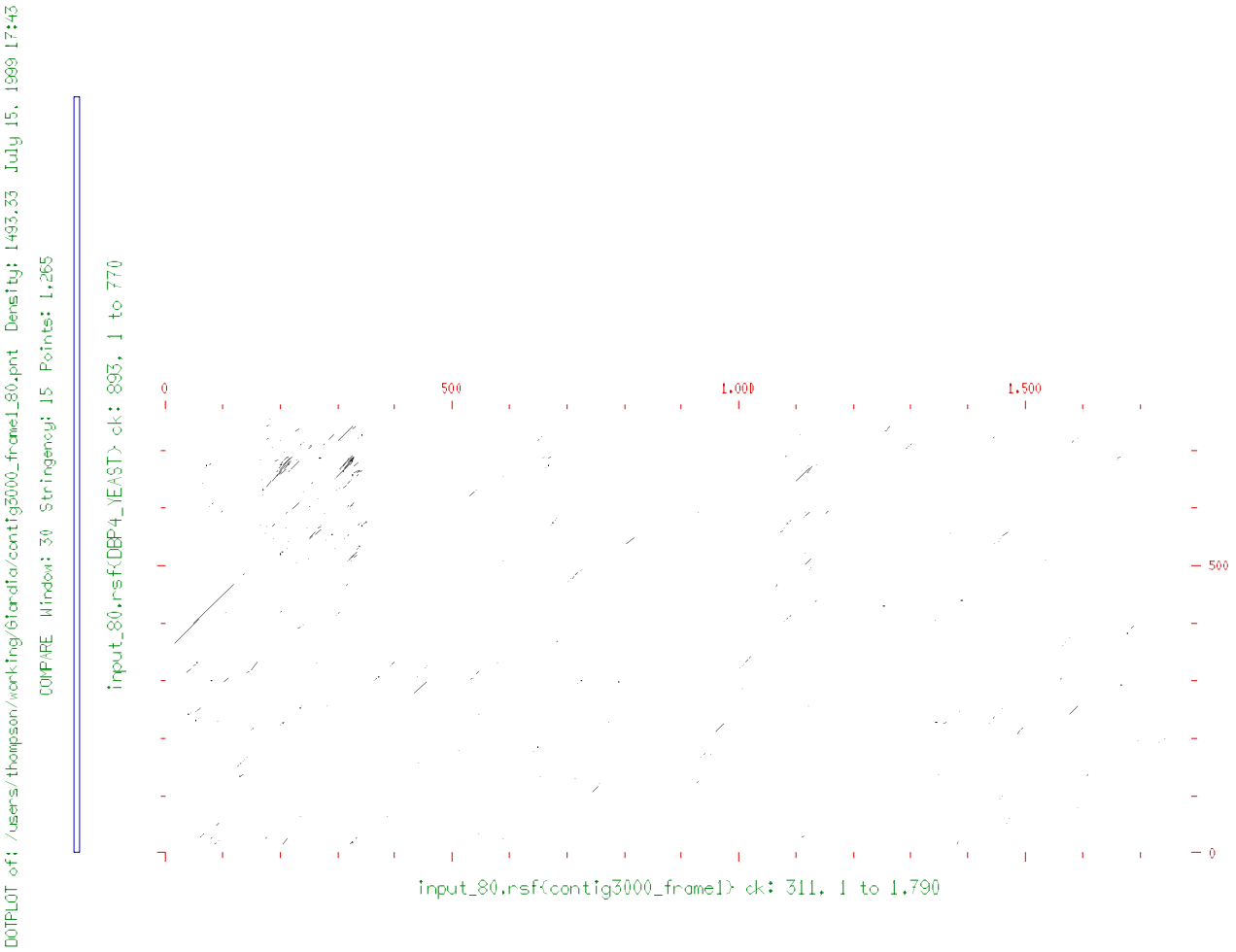
10) Dot matrix methods.

Compare and DotPlot: the GCG implementation of dot matrix analysis.

Dot matrix analysis is one of the few ways to identify other elements beyond what dynamic programming algorithms show to be similar between two sequences. GCG implements dot matrix methods with two programs. Compare generates the data that serves as input to DotPlot which actually draws the matrix. Analyze contig300_frame1 to the two sequences loaded into SeqLab above using these methods. Start the program by selecting "**contig3000_frame1**" and "**DBP4_YEAST**" (remember to select nonadjacent entries with the <**ctrl**> key) in the SeqLab main Editor display. 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 11 points within a window of 30 residues.

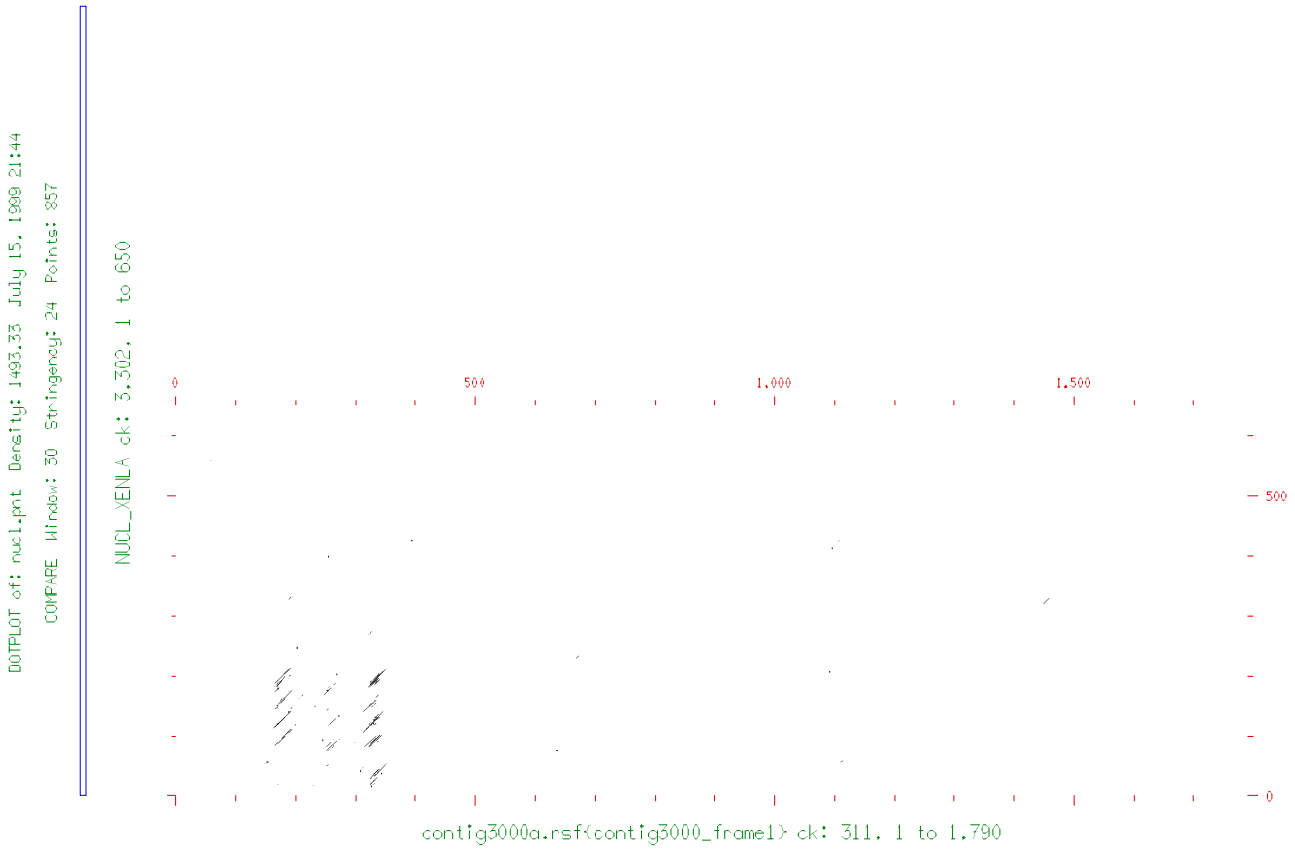
Just as in all windowing algorithms, you want to use a window size of approximately the same size as the feature that you're trying to recognize. Leave the window at its default setting of 30 for these runs, unless one of your sequences is too short for this size of a window to find much, in which case you should reduce the

window size appropriately. (In general, put the longer sequence along the horizontal axis. An advantage to this is you can page through the resultant dotplot, if desired, to see more detail by changing the density function when you run the program.) 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 is done through the "**Options**" menu. Below, I found that a stringency score of 15 within the default window of 30 resulted in 1265 points — right between the two sequences' lengths being compared. In this case, wherever the average of match scores within the window is equal to or exceeds 15, a point will be drawn at the middle of the window, then the window will be slid over one position at which point the process is repeated. When run at this stringency the graphic looks like the following:



Notice that running the comparison at an appropriate stringency, in this case 15 in a window of 30, produces a very clean plot with very little confusing noise. 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. The regions of similarity between contig3000_frame1 and the yeast helicase are very clear in the dotplot. Contig3000_frame1 lines up with DBP4_YEAST where its beginning through about residue 350 corresponds to around 350 to the end of the yeast protein. Also notice the direct repeat region; a sequence located around 700 on the yeast protein occurs at least twice in each sequence. Columns or rows of diagonals always mean direct repeat sequences.

To contrast the extensive similarity seen above, here's the dotplot of contig3000_frame1 to SW:NUCL_XENLA. I ran this comparison at a stringency of 24 within the default window size of 30 to generate 857 points. The plot follows below on the next page:



Here similarity is restricted to two columns of small diagonals near the beginning of the NUCL_XENLA protein — most probably direct repeat regions. When running all the dotplots, take notes of those particular regions in the proteins that appear similar. For example, as noted in the above plot, at least two short regions of the contig3000_frame1 sequence from around residue 150 to 200 and 300 to 350 are repeated several times in the NUCL sequence from about residue 1 through 225. We will need these numbers in the next section.

11) The dynamic programming alignment algorithms: use the right one for the right job — Gap and BestFit and FrameAlign.

You need to understand the difference between these first two algorithms! Gap is a 'global' alignment scheme and BestFit is a 'local' algorithm. 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 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, try specifying a more stringent alternative symbol comparison table, such as pam250.cmp or blosum100.cmp located in the logical directory GenMoreData. Both programs can generate 'gapped' output files in standard sequence formats; this can be handy as direct input to other GCG routines — particularly multiple sequence analysis programs.

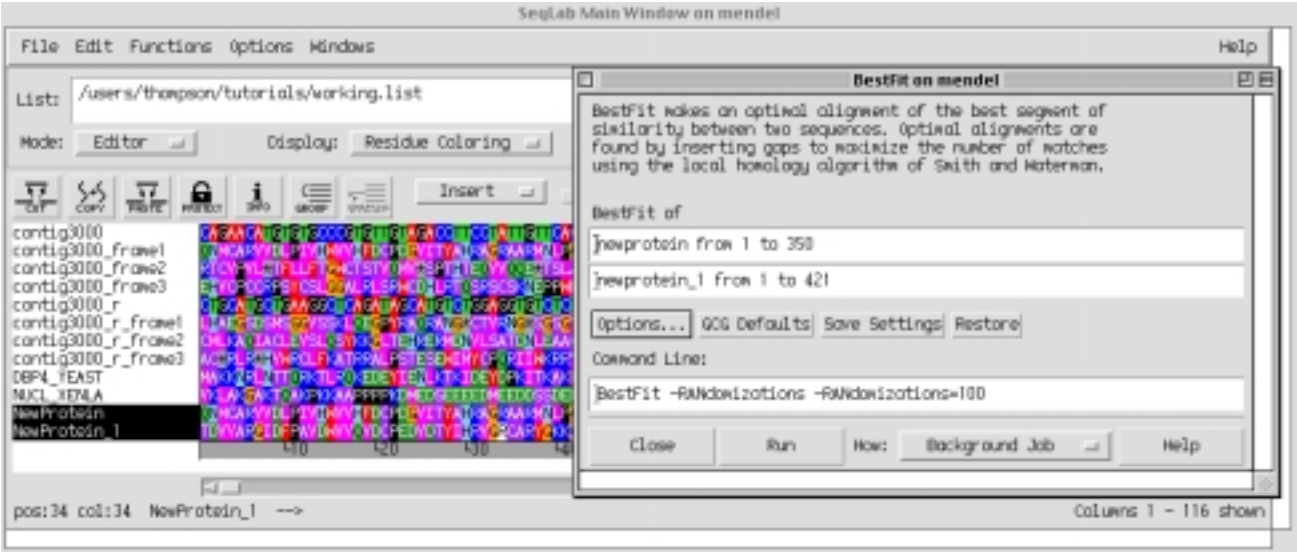
BestFit and Gap: how to use them to estimate significance.

What is significant? GCG provides a way to estimate significance in these two programs. When running them specify the option -randomizations=100 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 can be calculated with the following formula:

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

This type of significance analysis is known as a Monte Carlo approach; it has many implicit statistical problems, however, few practical alternatives exist. I will use randomizations with BestFit and contig3000_frame1 cross the DBP4 and NUCL proteins to illustrate. 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. 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. In lieu of switching to the command line, first create a new space to hold duplicate sequence data by going to the "**File**" "**New Sequence. . .**" menu and then specify "**Protein**." Next, select "**contig3000_frame1**" and then use the "**Edit**" function "**Select Range. . .**" to select just the desired region in contig3000_frame1. For this first comparison with DBP4 that is a region from residue **1** through **350**; type these 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 "**NewProtein**" sequence 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**." Repeat this whole procedure with "**DBP4_YEAST**" so that you now have two new sequences with just those portions of the original that we want to test. Select the two new sequences 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**." "**Close**" the "**Options**" window. The display should look like the following:



Press "**Run**" in the BestFit window and in a few moments your output file will appear. That output file is shown below; notice the extensive similarity over the length analyzed, the high original quality, and the low randomized quality. The Z score calculates out to be 32.5; therefore, the interpretation is that the similarity is extremely significant.

```
BESTFIT of: input_82.rsf{NewProtein}  check: 5464  from: 1  to: 350


 to: input_82.rsf{NewProtein_1}  check: 7330  from: 1  to: 421


 Symbol comparison table: /gcg/gcgcore/data/rundata/blosum62.cmp
 CompCheck: 6430

         Gap Weight:      8      Average Match:  2.912
      Length Weight:      2   Average Mismatch: -2.003

            Quality:    295             Length:    350
              Ratio:  0.875               Gaps:     10
 Percent Similarity: 40.120   Percent Identity: 29.641

 Average quality based on 100 randomizations: 47.7 +/- 7.6

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

 input_82.rsf{NewProtein} x input_82.rsf{NewProtein_1} July 15, 1999 21:18  ..

                 .         .         .         .         .
       2 NMCARVVDLPIVHWVVHFDCPDGVITYAHRAGRAARMNLPGFSLLFLTDQ 51
         .. || :| | | |||  |||: | || || || ||    | ||: || |
       2 DVVARGIDFPAVDWVVQVDCPEDVDTYIHRVGRCARYGKKGKSLIMLTPQ 51
                 .         .         .         .         .
      52 EQ.GFTKRLDEAKIDYQKKTVKLRTVVSIRQKLTELCITDTYIKHLAQKA 100
         ||  | |||.  ||:  |  :|     ||: .|  |   |  :|:| |||
      52 EQEAFLKRLNARKIEPGKLNIKQSKKKSIKPQLQSLLFKDPELKYLGQKA 101
                 .         .         .         .         .
```

```
101 IVSYAKSIHVQGDREVFPPASELNLTDIALSYGL..ASNINLSVGKQPGI 148
     :|| :||:|| |::||    ||   : | | ||  |  | :   |
102 FISYVRSIYVQKDKQVF.KFDELPTEEFAYSLGLPGAPKIKMKGMKTIEQ 150
            .         .         .         .         .
149 STQHPASEQQMASATGIRQPEGEHTDADDEEERDDLLKVTKIVTSVLSSK 198
    . :   . .|:|  .   : :||  :   .: |    |. .     : |.
151 AKERKNAPRQLAFLSKANE.DGEVIEDKSKQPRTKYDKMFERKNQTILSE 199
            .         .         .         .         .
199 EKEELQQEREKQIERKLLKGSIEEAARIAREAGRHKI.LNTSSDEESQST 247
        : . . .: |      |:.       ||    : | ||   : ..
200 HYLNITKAQAQEDEDDDFI.SVKRKDHEINEAELPALTLPTSRRAQKKAL 248
            .         .         .         .         .
248 SGAFSAKHTNNSAQ....DESDESELSSYTSASEEHS.GTTFPNEASHVS 292
    |   |     |...    || :   .       | |  |     .   ..
249 SKKASLASKGNASKLIFDDEGEAHPVYELEDEEEFHKRGDAEVQKTEFLT 298
            .         .         .         .         .
293 RLQQRIAHNDSFDREAHKRKNRRKSKRRAAS...EQESSYDDS.SFDESE 338
    :    .|  |. |::  | | . | ::|  .   | |.. ::  | || |
299 KESAVMADIDNIDKQVAKEKKQEKKRKRLEAMRREMEAAMEEEISGDEEE 348
```

If you suspect a frame shift sequencing error in the sequence frame being considered, a very powerful pairwise alignment program, FrameAlign, is available. This program uses dynamic programming to align a protein to a DNA sequence with the allowance of frame shifts. Let's try aligning all of contig3000 to DBP4 with this program to see what happens. You can try this on your own, without my guidance. Here's the abridged result of my run, where I used the BLOSUM30 scoring matrix:

```
Local FrameAlign alignment of: Contig3000

to: dbp4_yeast

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

            .         .         .         .         .
 13 GCCCGTGTTGTAGACCTTCCTATTGTTCACTGGGTGGTGCACTTCGACTG 62
    ||||||   :::|||:::|||   |||   |||||||||   ...|||||
354 AlaArgGlyIleAspPheProAlaValAspTrpValValGlnValAspCy 370
            .         .         .         .         .
 63 TCCAGATGGTGTGATCACCTACGCACACAGAGCAGGTCGTGCAGCAAGAA 112
    ||||...   |||   ||||||   ||||||...||||||   ||||||
371 sProGluAspValAspThrTyrIleHisArgValGlyArgCysAlaArgT 387
            .         .         .         .         .
113 TGAACCTCCCTGGCTTCTCACTTCTATTCCTAACAGATCAGGAGCAG... 159
           ...|||   ||||||:::   ||||||   |||||||||
388 yrGlyLysLysGlyLysSerLeuIleMetLeuThrProGlnGluGlnGlu 403
            .         .         .         .         .
160 GGGTTCACGAAGAGGCTGGACGAGGCAAAGATTGACTATCAGAAGAAGAC 209
       |||   |||||||||...      ||||||...      |||   ..
404 AlaPheLeuLysArgLeuAsnAlaArgLysIleGluProGlyLysLeuAs 420
            .         .         .         .         .
210 GGTGAAGCTAAGGACCGTCGTGTCCATACGCCAGAAGCTTACAGAGCTTT 259
    .:::|||               ||||||...      |||      |||
421 nIleLysGlnSerLysLysLysSerIleLysProGlnLeuGlnSerLeuL 437
            .         .         .         .         .
260 GCATCACAGATACATACATAAAGCACCTAGCACAAAAAGCGATAGTTTCA 309
           |||      :::|||   |||   |||||||||   :::|||
438 euPheLysAspProGluLeuLysTyrLeuGlyGlnLysAlaPheIleSer 453
            .         .         .         .         .
```


310 TATGCTAAATCCATCCATGTCCAAGGTGATCGGGAGGTGTTTCCCCCGGC 359
454 TyrValArgSerIleTyrValGlnLysAspLysGlnValPhe...LysPh 469
360 ATCTGAACTCAACCTAACAGATATAGCATTGAGCTACGGATTGGCCAGCA 409
470 eAspGluLeuProThrGluGluPheAlaTyrSerLeuGlyLeu.ProGly 485
410 ATATTAATTTATCAGTTGGAAAGCAGCCTGGTATATCTACGCAACACCCT 459
486 AlaProLysIleLysMetLysGlyMetLysThrIleGluGlnAlaLysGl 502
460 GCATCAGAACAGC...AGATGGCATCAGCTACAGGGATCCGCCAACCAGA 506
503 uArgLysAsnAlaProArgGlnLeuAlaPheLeuSerLysAlaAsnGluA 519
507 A.GGCGAGCATACTGATGCAGACGATGAAGAGGAGAGAGACGATCTCTTA 555
520 spGlyGluValIleGluAspLysSerLysGlnProArgThrLysTyrAsp 535
556 AAGGTCACCAAGATTGTCACATCTGTACTCAGCTCCAAGGAAAAAGAGGA 605
536 LysMetPheGluArgLysAsnGlnThrIleLeuSerGluHisTyrLeuAs 552
606 GCTTCAACAAGAACGGGAAAAGCAAATAGAGAGAAAGTTATTAAAGGGGT 655
553 nIleThrLysAlaGlnAlaGlnGluAspGluAspAspAspPheIleSerV 569
656 CCATTGAGGAGGCTGCTCGAATTGCCCGAGAAGCTGGGCGCCACAAGATT 705
570 alLysArgLysAspHisGluIleAsnGluAlaGluLeuProAlaLeuThr 585
706 CTTAATACAAGTAGCGACGAGGAGAGCCAATCTACGTCTGGCGCTTTCAG 755
586 LeuProThrSerArgArgAlaGlnLysLysAlaLeuSerLysLysAlaSe 602
756 TGCGAAACACACCAACAATTCAGCTCAA.....GACGAAAGCG 793
603 rLeuAlaSerLysGlyAsnAlaSerLysLeuIlePheAspAspGluGlyG 619
794 ATGAGAGCGAGCTGTCGTCATATACTTCGGCTAGTGAAGAACACTCC... 840
620 luAlaHisProValTyrGluLeuGluAspGluGluGluPheHisLysArg 635
841 GGTACAACGTTTCCAAATGAAGCCTCTCACGTCTCTAGGCTCCAACAACG 890
636 GlyAspAlaGluValGlnLysThrGluPheLeuThrLysGluSerAlaVa 652
891 CATAGCACATAACGATTCCTTCGATCGGGAAGCACATAAACGAAAGAACA 940
653 lMetAlaAspIleAspAsnIleAspLysGlnValAlaLysGluLysLysG 669
941 GGAGGAAGAGCAAGCGAAGAGCAGCCTCTGAACAAGAGTCAAGCTATGAT 990
670 lnGluLysLysArgLysArgLeuGluAlaMetArgArgGluMetGluAla 685
991 GACTCCTCTTTTGATGAGTCAGAGGAGGAAATGCAGAGCAAG 1032
686 AlaMetGluGluGluIleSerGlyAspGluGluGluGlyLys 699

It appears as if no frame shift errors are in this stretch of DNA since all gaps are multiples of three and the similarity extends nearly to the end of DBP4. Not only that, but we didn't discover any other similarity to any helicase type proteins on any other contig3000 frames except forward frame1.

However, often a seemingly good alignment will not be significant upon further inspection — do not blindly accept the output of any computer program! Always investigate further for similarities can be strictly artifactual. The second comparison that I chose, contig3000_frame1 against the NUCL_XENLA protein, turned out to be entirely insignificant. Repeat this analysis in a manner similar to the above BestFit run with DBP4, if desired. From the previous DotPlot analysis, I saw that at least two short regions of the contig3000_frame1 sequence within the area of residue 150 to 350 are repeated several times in the NUCL sequence from about residue 1 through 225. A Monte Carlo analysis of this comparison is shown below. The Z score is 3.5, right near the bottom of Doolittle's "Twilight Zone:"

```
BESTFIT of: contig3000a.rsf{contig3000_frame1}  check: 311  from: 150  to: 350


 to: NUCL_XENLA  check: 3302  from: 1  to: 225

ID   NUCL_XENLA     STANDARD;      PRT;   650 AA.
AC   P20397;
DT   01-FEB-1991 (REL. 17, CREATED)
DT   01-FEB-1994 (REL. 28, LAST SEQUENCE UPDATE)
DT   01-OCT-1994 (REL. 30, LAST ANNOTATION UPDATE)
DE   NUCLEOLIN (PROTEIN C23). . . .

 Symbol comparison table: /gcg/gcgcore/data/rundata/blosum62.cmp
 CompCheck: 6430

         Gap Weight:      8      Average Match:  2.912
      Length Weight:      2   Average Mismatch: -2.003

            Quality:     92             Length:    111
              Ratio:  0.852               Gaps:      2
 Percent Similarity: 32.710   Percent Identity: 27.103

 Average quality based on 100 randomizations: 64.2 +/- 8.0

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

 contig3000a.rsf{contig3000_frame1} x NUCL_XENLA July 15, 1999 22:16  ..

                  .         .         .         .         .
     242 EESQSTSGAFSAKHTNNSAQDESDESELSSYTSASEEHSGTTFPNEASHV 291
         ||  |.            |.  .  .. .   .|.    | | | .
      34 EEDDSSDEEVEVPVKKTPAKKTATPAKATPGKAATPGKKGAT.PAKNGKQ 82
                  .         .         .         .         .
     292 SRLQQRIAHNDSFDREAHKR...KNRRKSKRRAASEQESSYDDSSFDESE 338
         .: |:     |  | ||  .   ||:  .|:  | .:||  ||   ||||
      83 AKKQESEEEEDDSDEEAEDQKPIKNKPVAKKAVAKKEESEEDDDDEDESE 132
                  .
     339 EEMQSKRKQKP 349
         ||    :|  |
     133 EEKAVAKKPTP 143
```

The extreme aspartate and glutamate richness of both sequences is what the program found, yet the Monte Carlo test suggests that this similarity is not at all significant, it is merely the result of compositional bias. As mentioned previously, the programs Xnu and Seg are now available outside of BLAST for prefiltering 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.

Database searching analysis is one of the most commonly misunderstood areas in computational molecular biology and bioinformatics today. There is a tremendous amount of confusion in this field and anything that can be done to try and clear up some of the mess is entirely worthwhile. 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.

Now that we've seen how powerful homology inference methods are, how can we tie it all together; how can we delineate exactly where our genes are? What else is available to help? We still have one more 'trick up our sleeve.'

12) Combined methods for gene inference available on the Internet.

An additional source of information that should not be ignored are the powerful Internet servers available for these type of analyses. Most of these servers combine many of the methods that we have already explored in this exercise but they consolidate the information and often combine signal and content methods with homology inference in order to ascertain exon locations. Many use powerful neural net or artificial intelligence approaches to assist in this difficult process.

Most gene inference services are available through the World Wide Web. A very nice bibliography on computational methods for gene recognition has been compiled at Rockefeller University (http://linkage.rockefeller.edu/wli/gene/). Four popular gene-finding servers listed there are GRAIL, GeneId, NetGene, and GenMark. GRAIL (Gene Recognition and Analysis Internet Link) is a neural net system for detecting exons (trained on human data). It looks for base composition that 'appear' exon-like. It does not define boundaries. GeneId is an Artificial Intelligence system for analyzing vertebrate genomic DNA and predicting exon and gene structure. NetGene predicts splice sites likelihood using neural net techniques. GenMark is based on a special type of Markov chain model of coding and noncoding sequences; it is optimized for enteric eubacteria. The BCM Gene Finder (http://dot.imgen.bcm.tmc.edu:9331/gene-finder/gf.html) is particularly powerful. You should find all these services to be extremely helpful but I will not demonstrate their use here.

IV. Annotate your sequence to see 'how it all comes together.' The combinatorial approach.

13) Interpretations: data annotation and analysis.

Translation: where to start and stop — Exons and Introns, Splice Junctions, etc.
What about precursor versus mature, signal peptides and processing?
How can we tell just what makes up the mature protein?

About the best way to make any sense out of all of this data is to get it all in one spot. I used to make paper maps of the sequences and annotate the 'heck' out of them with colored markers. Now with SeqLab we can prepare annotations right in the context of the editor. Whereever a preponderance of data suggests a gene, then I believe one is there; where the data is contradictory, decisions can't be made very well; and where lots of data argues against the location of genes, then I believe one is not there. You will want to have all data at your disposal to do this step. Either prepare and print PostScript graphics of the various figure files prepared in the exercise or look back through the figures I have supplied in the tutorial. The mechanics of printing PostScript graphics at your site will vary and will have to be resolved with the assistance of the system administrator.

(Another way to get PostScript files from all the figures in the exercise is to initialize GCG graphics to PostScript at the command line in a terminal window. Either type the command postscript, and then specify epsf for Encapsulated PostScript File as the graphics device, and then specify a temporary filename as the output port, or use the Encapsulated PostScript choice on the SetPlot menu. All graphics output will now be written to your account in PostScript format under the specified temporary filename. A sample screen trace of the PostScript command follows:

```
% postscript

 Use Postscript graphics with what device:

  LaserWriter
  Lzr1200
  LN03-ScriptPrinter
  LPS20
  ColorScript-100
  EPSF (single page encapsulated postscript format)

 Please choose one ( * LASERWRITER * ) epsf

 To what port is your EPSF connected (* term: *) temp.epsf

 Plotting Configuration set to:

       Language: psd
         Device: EPSF
  Port or Queue: temp.epsf
```

Next use the Figure program to draw each plot to a PostScript file. Launch the program by typing figure with the option -plot=(new actual EPSF output filename) in order to redirect the output from temp.epsf to your_plot.epsf. A screen trace illustrates:

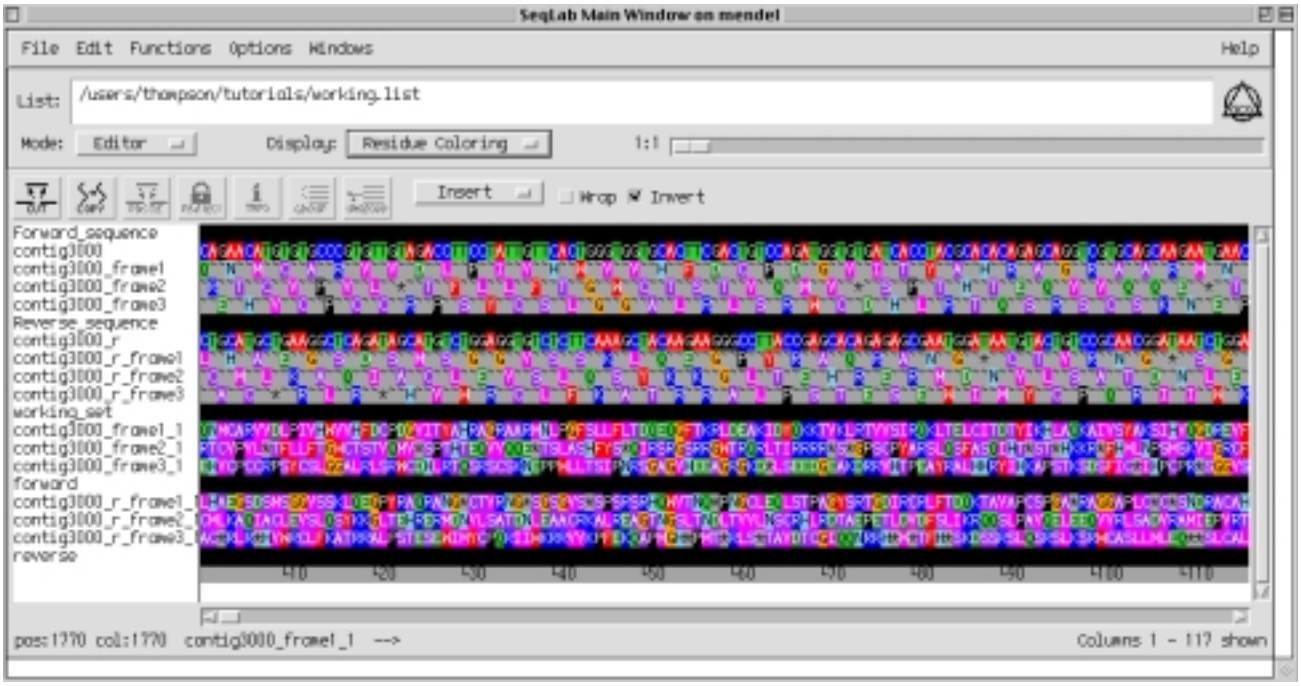
```
% figure -plot=contig3000_testcode.epsf

Figure makes figures and posters by drawing graphics and text
together. You can include output from other GCG graphics programs as
part of a figure.

 FIGURE from what file ?  contig3000_testcode.figure

 PostScript instructions for a EPSF are now being sent to contig3000_testcode.epsf.)
```

Also either make printouts of the text files from the one- and two-dimensional signal searches as well as the similarity searches that you performed earlier or use my screen traces in the tutorial. The point — you want to compare all of these methods of analysis and try to make some educated guesses as to where the genes actually start and stop.

To annotate sequences in the SeqLab editor go to the "**File**" "**New Sequence. . .**" menu and press the "**Text**" button. This will place an empty text line below your sequences. You may want to add several "**NewText**" lines. You may also want to "**CUT**" out the test sequences that we tested above under the pairwise comparison section of the exercise. Finally it may be helpful to have translations of contig3000 generated that are aligned to their respective codons. To do this go through the "**Edit**" "**Translate. . .**" function just like you did at the very beginning of the tutorial, but this time check the "**Align Translation**" button. Repeat this for every frame, forward and backward. Arrange your display any way that makes sense to you with the "**CUT**" and "**PASTE**" buttons. "**PASTE**" inserts sequences below whatever sequence is selected at the time. Change entries' names by quickly double-clicking them (or by pressing the "**INFO**" button) and editing the "**Name:**" box of the "**Sequence Information**" window.

Now would also be a good time to add in the feature annotation that we created automatically back when we ran Motifs. Remember the file motifs.rsf? We're going to load that into the editor now so that the location of the PROSITE signatures will be included in the editor sequence display. To do this use the "**SeqLab Output Manager**." This is a very important window available through the SeqLab "**Windows**" menu 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/or deleted from this window. We need to use an extremely important function at this point; select the file **motifs.rsf**, then press the "**Add to Editor**" button and specify "**Overwrite old with new**" in the next window when prompted, to take the motifs.rsf feature file and merge it with the old RSF (Rich Sequence Format: the sequence data as well as any reference information) file in the open editor. "**Close**" the "**Output Manager**" after loading your new RSF file.

My display looks like the following after doing all of this:

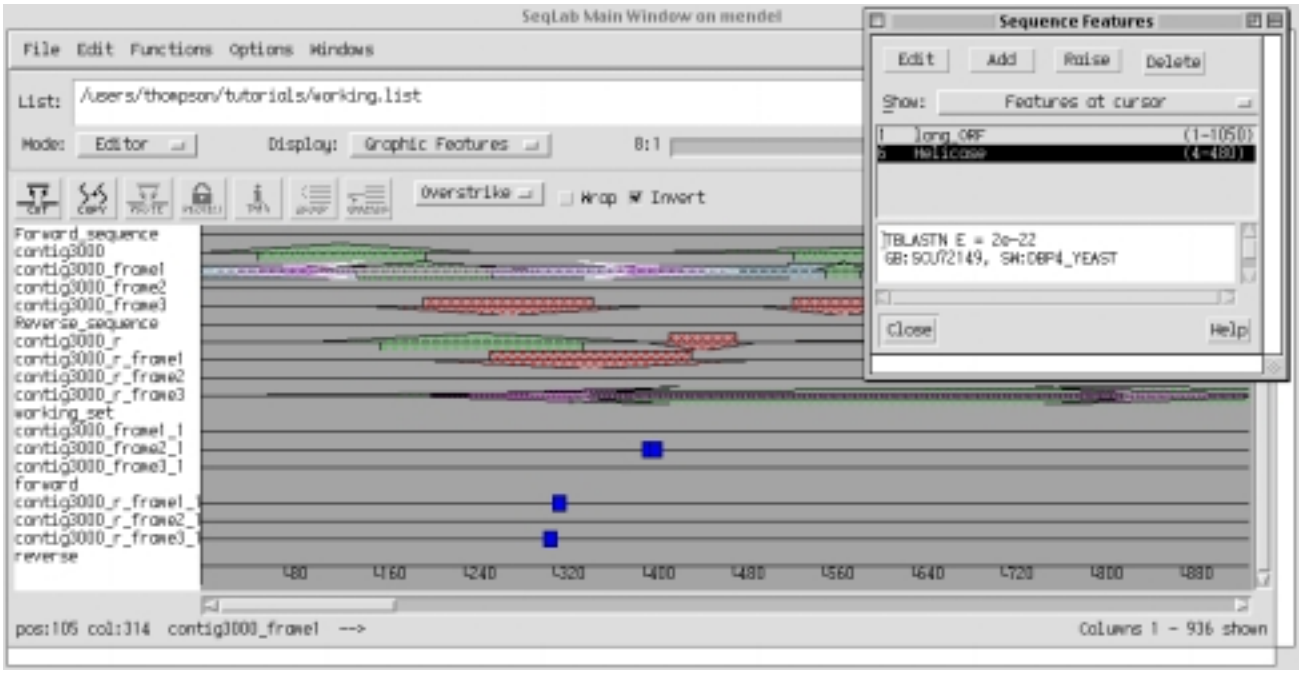


Many matters complicate this process. Eukaryotic exons and introns (in most eukaryota) can be especially confusing, but prokaryotic and organellar DNA have their own problems too. One that concerns all genes is, after you do translate the entire thing, whether it has a signal peptide portion and how to tell which is what. A database of preprotein signal peptides is available through Gunnar von Heijne for just this type of analysis

(1987b). The program SPScan incorporates von Heijne's method and can be run with a prokaryote gram negative or positive switch to change from the default eukaryote search matrix.

Analyze and ponder all your data. It won't be as easy as you would have hoped for. Fortunately, often in a lab situation cDNA data is also available on a given sequence, although with the increased emphasis on genomic sequencing this is becoming less and less true. Go into the new text lines and type in your annotations at the appropriate spots. Changing from "**Insert**" to "**Overstrike**" may help you type in text lines without skewing data downstream. Another way to annotate RSF files is to use the colored feature display. Develop a coding system to represent various attributes and then embed them in your RSF file. Do this by first selecting a region within your sequences with your mouse that you wish to annotate (you can select multiple, nonadjacent regions in the same sequence, if desired) and then go to the "**Windows**" "**Features**" menu. Press "**Add**" in the "**Sequence Features**" window and then type in relevant information under the "**Keyword:**" and "**Comments**" areas; also choose an appropriate shape, fill, and color for your new custom feature. Press "**OK**" after adding your new feature and then "**Close**" the "**Sequence Features**" window. It is probably a good idea to periodically save your work by going to the "**File**" "**Save As. . .**" menu and supplying an appropriate RSF file name. If you are prompted by a "**File exists**" box, then answer "**Overwrite**" to replace your old file with the update. Used in combination text and color annotation can help produce stunning presentations for publication or poster display. Annotate the sequences in the editor at the precise position where you believe relevant features lay. Indicate where the various signals and content biases you found are located. Note where you feel the actual starts and stops of the coding regions are in your sequence. Similarities discovered through database searching will greatly assist your interpretation. This is often very helpful, especially if you are dealing with a system that has much available database information. We need to synthesize all this data to decide what portions of the tentative URF's actually code for proteins. Putative coding regions (CDS's) that the analyses have indicated will then be used in the next portion of the tutorial to prepare multiple sequence alignments of those regions. The validity of your interpretations will relate directly to your understanding of the molecular biology of the system you are dealing with.

Go to the "**Display:**" box and change it from "**Residue Coloring**" to "**Feature Coloring**." Zoom out on the file so that you can see more of it at once. The colors are now based on the information that we coded into the file. There would be even more color feature information were these sequences from the databases as SeqLab reads the Feature Table for each entry and annotates correspondingly. Change the "**Display:**" to "**Graphic Features**;" now the features are represented using the same colors as before but in a 'cartoon' fashion. Use the mouse to move your cursor to one of the colored areas; quickly double-click it. This will produce a new window that describes the features located at the cursor. Click on one of the features to get more information on it. As you've seen, all the features are fully editable through the "Edit" check box in this panel and new features can be added with several desired shapes and colors through the "Add" check box. The display will look something like my example below:



Close the "**Sequence Features**" window and return your display to "**1:1**."

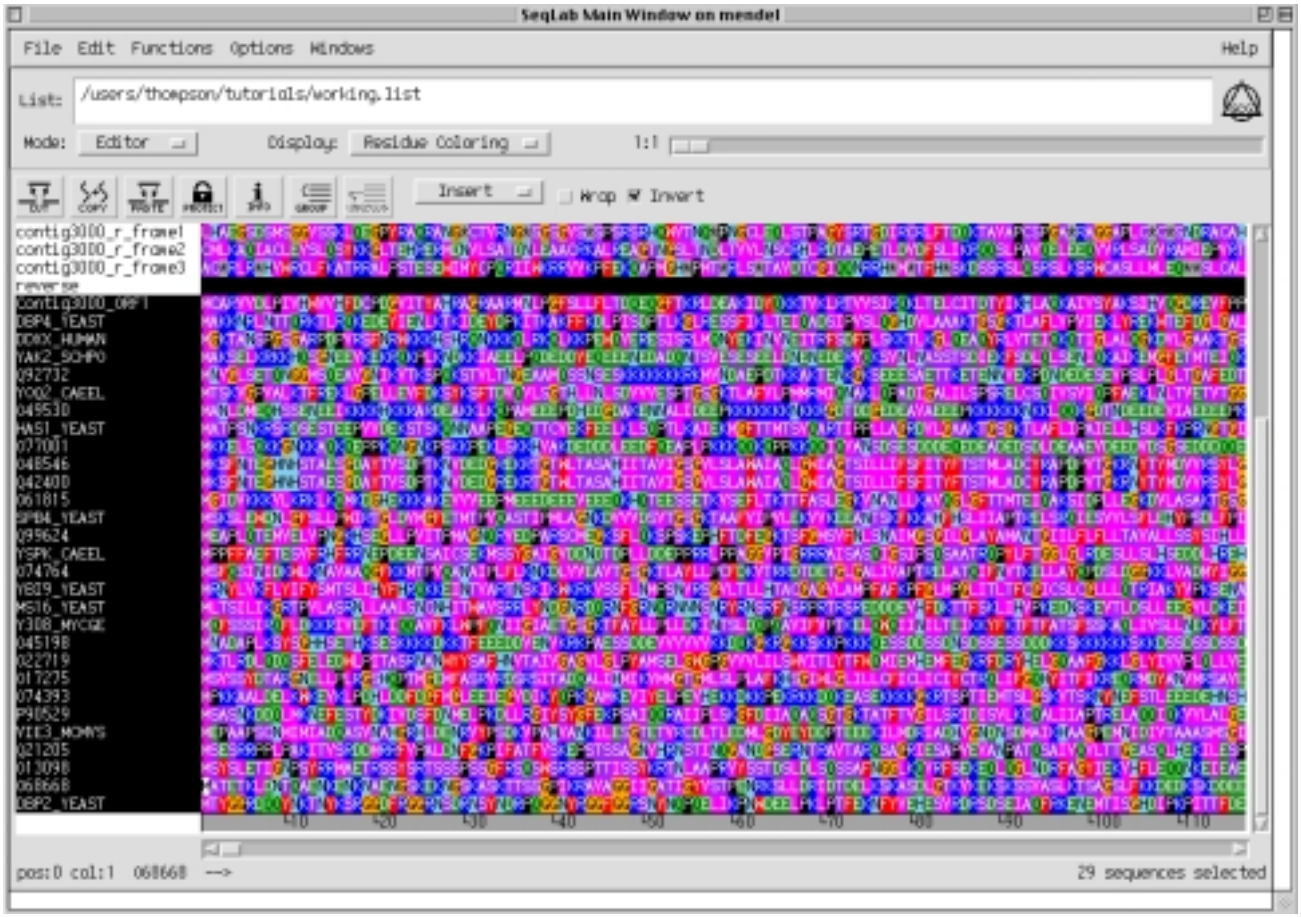
You have been exposed to a perplexing variety of technique for the identification of protein coding regions in genomic DNA. As in all molecular and biological computer analyses, the more you understand the chemical, physical and biological systems involved, the better your chance of success in analyzing them. Certain strategies are inherently more appropriate to others in certain circumstances. Making these types of subjective, discriminatory decisions and utilizing all of the available options so that you can generate the most practical data for evaluation are two of the most important 'take-home' messages that I can offer!

Several general references are available in this field — most provide extensive weight matrices for consensus pattern searches. Naturally each would have to be tailored into the format correct for whichever matrix searching program you might be using. They also all describe many of the factors involved and the constraints used in content type algorithms. *Sequence Analysis Primer*, by Gribskov and Devereux (1992), is a good starting point.

V. And finally — multiple sequence alignments. What good are they?

Now we can prepare a multiple sequence alignment of the regions identified as CDS. I'll illustrate with the helicase sequence found on contig3000_frame1 and the FastA file that we ran against that sequence. Create space for a new protein sequence through the "**File**" "**New Sequence. . .**" "**Protein**" menu and then select the region of the DNA aligned **contig3000_frame1** sequence that you believe is actually translated with the "**Edit**" "**Select Range**" function and "**COPY**" and "**PASTE**" it into the new empty sequence slot. Remember to place your cursor in position one of the new empty sequence space before pasting and to paste from the "**Text clipboard**," not the "Sequence clipboard." Now go to the "**Edit**" "**Remove Gaps. . .**" menu and "**Remove all gaps**" within the sequence to create something we can build an alignment off of. Next, go to the "**File**" menu. Pick "**Add sequences from**" and select "**Sequence Files**." (Reminder: only GCG format compatible sequences or list files are accessible through this route. Use SeqLab's "Import" function to directly load GenBank format sequences without the need to reformat.) This will produce an "**Add Sequences**" window from which you can select sequences to add to the editor. Use the "**Filter**" box to find the proper file. Files

are normally filtered so that only those that end with the extension "**.seq**" are displayed. This won't help us becuase the sequences that we want to add are in the FastA file that we produced a while ago. That file should have the word fasta in it somewhere, unless you changed it to something else. Therefore, delete the "**.seq**" extension in the "**Filter**" box; and insert "***fasta***," a wild card expression that should find all files in your working directory with the word fasta in them. Press the "**Filter**" button to screen your file list. Select the correct file from the "**Files**" box and then check the "**Add**" button at the bottom of the window to load the FastA file into the SeqLab editor window. Press the "**Interrupt Loading**" button after about 15 to 20 sequences to limit the number of sequences in the alignment to not more than 20 so that we can do the rest of the exercise in 'real' time. (In reality you should cut your list off at a point determined by the significance of the hit.) "**Close**" the "**Add Sequences**" window afterward. The sequences will load at the bottom of the display. Take a look at some of the members from the FastA list. Quickly double click on various entries' names to see the database reference descriptions for them. (This is the same sort of information that you can get with the GCG command "typedata -ref" at the command line.) Select the duplicated contig3000 ORF and all the FastA sequences by dragging the mouse through the entries or by shift-clicking the bottom and top entry desired (select non-adjacent entries with Cntrl-clicks). The editor display should look similar to the following now:



14) Performing the alignment — the PileUp program

After your sequences are selected, go to the "**Functions**" menu and select "**Multiple comparison**." Click on "**PileUp . . .**" to align the entries. A new window will be produced with the parameters for running PileUp. Be sure that the "**How:**" box says "**Background Job**." For this first pass accept all of the program defaults by

merely pressing the "**Run**" button and the window will go away. The run may produce an empty output file and a log file. If this happens, go to the "**Windows**" "**Job Manager**" menu and select the "**PileUp**" job listed there to read the error message. If the message is "***** ERROR! More than 2000 gap insertions. *****" then cut some more sequences off the bottom of your list as they are too divergent of a set to align. This happened to me in the above example and I ended up cutting the list down to about 15 entries.

You may also want to try using an alternate symbol comparison matrix. The default BLOSUM62 matrix is very good for 'run-of-the-mill' similarity but other, more appropriate, matrices can be specified in the options menu. The BLOSUM30 matrix is most appropriate for datasets with a high degree of divergence. To specify this matrix, press the "**Options**" button in the PileUp window and then check the box in front of "**Scoring Matrix. . .**" and press the "**Scoring Matrix. . .**" button to select "**blosum30.cmp**" off of the "**Chooser for Scoring Matrix**" window displayed. Click "**OK**" in the "**Chooser**" window, "**Close**" the "**Options**" window, and then press "**Run**" in the PileUp window. Once you do get the program to run, it 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. 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 BLOSUM matrix and gap introduction and extension penalties used. In most cases the default values work fine. Scroll through your alignment to check it out and then "**Close**" the window afterwards. If it is unacceptable, reperform the PileUp with fewer sequences still and/or lower gap penalties. I ended up using 13 sequences along with the **BLOSUM30** matrix with a gap creation penalty of **10** and a gap extension penalty of **3** before I was happy with my initial alignment. An abridged output file from my example follows below. Notice the interleaved character of the sequences, yet they all have unique identities, addressable by using their MSF filename together with their own name in braces, {name}:

```
!!AA_MULTIPLE_ALIGNMENT 1.0
PileUp of: @/users/thompson/.seqlab-mendel/pileup_88.list

 Symbol comparison table: /gcg/gcgcore/data/moredata/blosum30.cmp  CompCheck: 85
99

                  GapWeight: 10
             GapLengthWeight: 3

 pileup_88.msf  MSF: 1378  Type: P  July 16, 1999 16:54  Check: 3960 ..

 Name: o48546           Len:  1378  Check: 7617  Weight:  1.00
 Name: q42400           Len:  1378  Check: 7512  Weight:  1.00
 Name: yak2_schpo       Len:  1378  Check: 3110  Weight:  1.00
 Name: has1_yeast       Len:  1378  Check: 6635  Weight:  1.00
 Name: q92732           Len:  1378  Check: 8484  Weight:  1.00
 Name: o77001           Len:  1378  Check: 1009  Weight:  1.00
 Name: o61815           Len:  1378  Check: 6967  Weight:  1.00
 Name: o49530           Len:  1378  Check: 6662  Weight:  1.00
 Name: dbp4_yeast       Len:  1378  Check: 4306  Weight:  1.00
 Name: ddxx_human       Len:  1378  Check: 5150  Weight:  1.00
 Name: yoq2_caeel       Len:  1378  Check: 1590  Weight:  1.00
 Name: spb4_yeast       Len:  1378  Check: 8219  Weight:  1.00
 Name: contig3000_orf1  Len:  1378  Check: 6699  Weight:  1.00

//
```

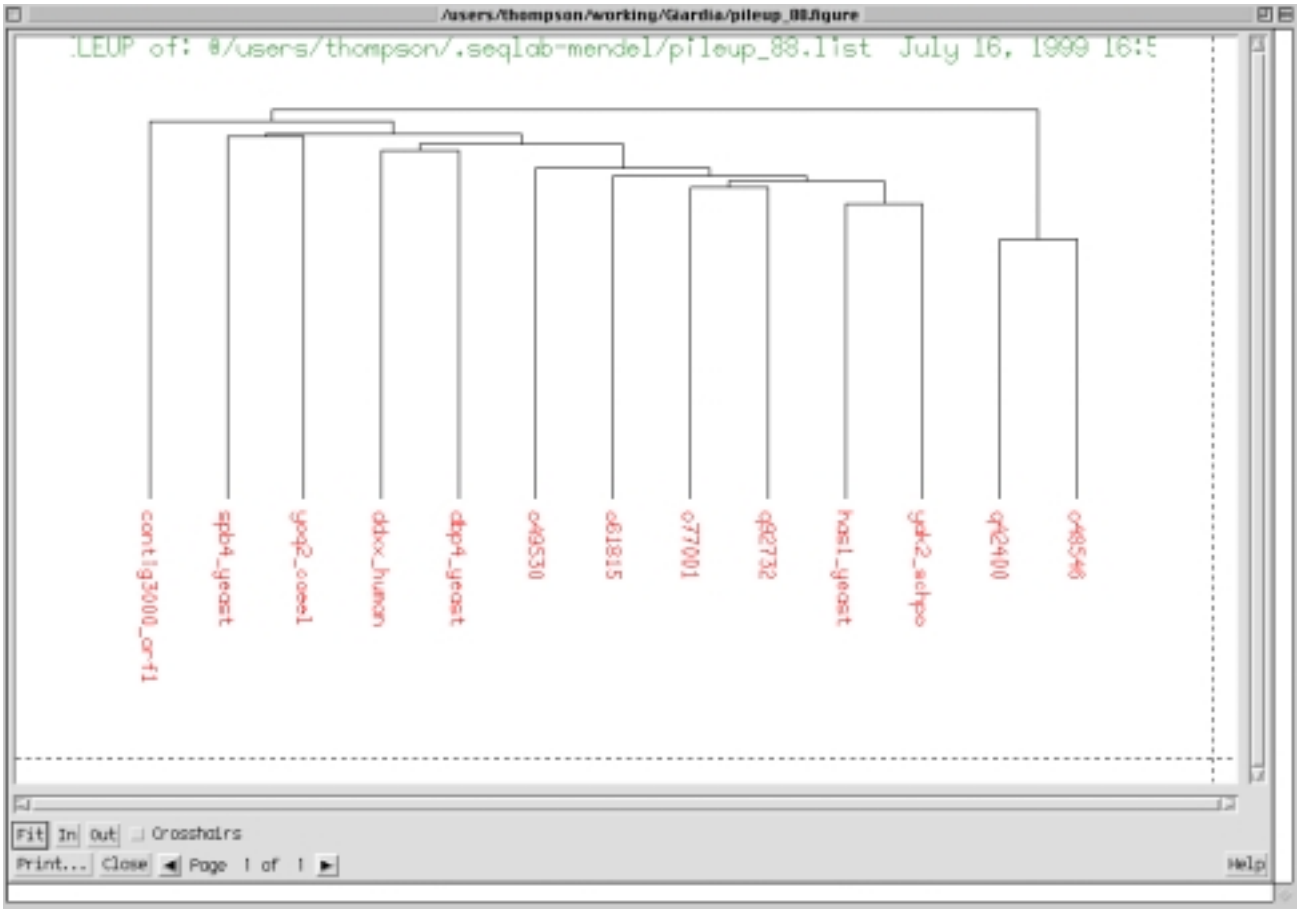
```
                1                                                    50
        o48546  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
        q42400  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     yak2_schpo  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     has1_yeast  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
        q92732  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~MNVG
        o77001  MKKELSQKKG NKKAQKQEPP KQNGNKPSKK PEKLSKKHVA KDEDDDLEED
        o61815  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
        o49530  ~~~~~~~~~~ ~~~~MANLDM EQHSSENEEI KKKKHKKRAR DEAKKLKQPA
     dbp4_yeast  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     ddxx_human  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     yoq2_caeel  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     spb4_yeast  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
contig3000_orf1  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~

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

501                                                  550
        o48546  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
        q42400  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     yak2_schpo  GILLCTNVAA RGLDIPAVDW IVQYDPPDDP RDYIHRVGRT ARGTKGTGKS
     has1_yeast  GILICTDVAA RGLDIPAVDW IIQFDPPDDP RDYIHRVGRT ARGTKGKGKS
        q92732  GTLLCTDVAA RGLDIPEVDW IVQYDPPDDP KEYIHRVGRT ARGLNGRGHA
        o77001  GILLCTDVAA RGLDIPQVDW IVQYDPPGDQ ASIIHRVGRT ARGSGTSGHA
        o61815  GILLCTDVAA RGLDIPAVDW IVQYDPTDEP REYIHRVGRT ARGTNGSGKA
        o49530  GILLCTNVAA RGLDFPHVDW IVQYDPPDNP TDYIHRVGRT ARGEGAKGKA
     dbp4_yeast  VCLFATDVVA RGIDFPAVDW VVQVDCPEDV DTYIHRVGRC AR.YGKKGKS
     ddxx_human  AVLFATDIAA RGLDFPAVNW VLQFDCPEDA NTYIHRAGRT AR.YKEDGEA
     yoq2_caeel  GVMISTDVMA RGIDISDIDW VIQFDLPKHS SWFVHRAGRT AR.CGREGNA
     spb4_yeast  SVLFTTDVAA RGIDIPDVDL VIQLDPPTNT DMFMHRCGRT GR.ANRVGKA
contig3000_orf1  ~~~~~~~MCA RVVDLPIVHW VVHFDCPDGV ITYAHRAGRA AR.MNLPGFS

                551                                                  600
        o48546  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
        q42400  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     yak2_schpo  LMFLAPSEL. GFLRYLKTAK VSLNEFE... ...FPANKVA NVQSQLEKLV
     has1_yeast  LMFLTPNEL. GFLRYLKASK VPLNEYE... ...FPENKIA NVQSQLEKLI
        q92732  LLILRPEEL. GFLRYLKQSK VPLSEFD... ...FSWSKIS DIQSQLEKLI
        o77001  LLLMRPEEL. GFLRYLKAAK VPLNEFE... ...FSWQKIA DIQLQLEKLI
        o61815  LLVLRPEEL. GFLRYLKAAK VTLNEFE... ...FSWSKVA NIQSQLENLI
        o49530  LLVLTPQEL. KFIQYLKAAK IPVEEHE... ...FEEKKLL DVKPFVENLI
     dbp4_yeast  LIMLTPQEQE AFLKRLNARK IEPGKLN... ...IKQSKKK SIKPQLQSLL
     ddxx_human  LLILLPSE.K AMVQQLLQKK VPVKEIK... ...INPEKLI DVQKKLESIL
     yoq2_caeel  L.ILIASEQL AYVNFL..DN HEKVKLDEIK VPTNNSRKSE ELRQKMIKIQ
     spb4_yeast  ITFLNEGREE DFIPFMQVKN VELEELD.LE VKGITTNFYE DFRNWILE..
contig3000_orf1  LLFLTDQEQ. GFTKRLDEAK IDYQK..... .KTVKLRTVV SIRQKLTELC

                601                                                  650
        o48546  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
        q42400  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     yak2_schpo  SKNYYLQQSA KDGYRSYLQA YASYS..... .......... ..........
     has1_yeast  KSNYYLHQTA KDGYRSYLQA YASHS..... .......... ..........
        q92732  EKNYFLHKSA QEAYKSYIRA YDSHS..... .......... ..........
        o77001  AKNYFLNQSA KEAFKSYVRA YDSHQ..... .......... ..........
        o61815  SKNYYLNKSA KEAYKCYLRA YDSHSLKVIH .KVQMMTYET NCHGVSRCPA
        o49530  SENYALKESA KEAYKTYISG YDSHS..... .......... ..........
     dbp4_yeast  FKDPELKYLG QKAFISYVRS IYVQKDKQVF .KFDELPTEE FAYSLGLPGA
     ddxx_human  AQDQDLKERA QRCFVSYVRS VYLMKDKEVF .DVSKLPIPE YALSLGLAVA
```

```
     yoq2_caeel  VSDRAILEAG TRAFVSHVES YAKHDCHLIC .SLDDLNVVG LANSYALLRL
     spb4_yeast  ..DRDRFDKG VKAYVAFIKY YSNHSATSIF .RLQSLDYVG IAKLYGLFRL
contig3000_orf1  ITDTYIKHLA QKAIVSYAKS IHVQGDREVF PPASELNLTD IALSYGLASN

                 651                                                700
         o48546  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         q42400  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     yak2_schpo  .......LKS IFDINKLDLA KVAKSFGFAH PPNVNITIGA SGRTDKKERR
     has1_yeast  .......LKT VYQIDKLDLA KVAKSYGFPV PPKVNITIGA SGKTPNTKRR
         q92732  .......LKQ IFNVNNLNLP QVALSFGFKV PPFVDLNVNS NEG.KQKKRG
         o77001  .......LKQ IFNVNTLDLQ AVAKSFGFLV PPVVDLKVAR PSGSDRKSDV
         o61815  TKY....LKD IFDVTNMDLT AVSKSFGFSV PPFVDLPISN KPKVEIRSKL
         o49530  .......MKD VFNVHQLNLT EVATSFGFSD PPKVALKIDR GGYRSKREPV
     dbp4_yeast  PKI.......  .KMKGMKTIE QAKERKNAPR QLAFLSKANE DGEVIEDKSK
     ddxx_human  PRV....RFL QKMQKQPTKE LVRSQADKVI EPRAPSLTND EVEEFRAYFN
     yoq2_caeel  PKM....RE. ...LSQRKDL DMFDRSAIET SEIKYADVKL EANRETVMKE
     spb4_yeast  PRM....PEI TKYLATEKQE GIFPGNWLVD PPVNMDEYKY KDKKREKERQ
contig3000_orf1  INLSVGKQPG ISTQHPASEQ QMASATGIRQ PEGEHTDADD EEERDDLLKV

                 701                                                750
         o48546  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         q42400  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     yak2_schpo  AGYNKKNHVD VYSKQRSSAI SQDKERGWSR ~~~~~~~~~~ ~~~~~~~~~~
     has1_yeast  KTHK~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         q92732  GGGGFGYQKT KKVEKSKIFK HISKKSSDSR QFSH~~~~~~ ~~~~~~~~~~
         o77001  GGGGFGFYKK MNEGSASKQR HFKQVNRDQA KKFMR~~~~~ ~~~~~~~~~~
         o61815  SGAGYRKKKQ SFTFKAKK~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         o49530  NKFKRGRGGG RPGGKSKFER Y~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     dbp4_yeast  QPRTKYDKMF ERKNQTILSE HY.LNITKAQ AQEDEDDDFI SVKRKDHEIN
     ddxx_human  EKMSILQKGG KRLEGT...E HRQDNDTGNE EQEEEEDDEE EMEEKLAKAK
     yoq2_caeel  KHEKKVETLA AKDKKRREKE ARKLKKMGGR FRNGGGTGRK AEEKKALKRK
     spb4_yeast  ETLKNISLIN DKKKLKSELK KKNLAWSDKT LTKERKLERK ..EKMSLKRK
contig3000_orf1  TKIVTSVLSS KEKEELQQER EKQIERKLLK GSIEEAARIA REAGRHKILN

                 751                                                800
         o48546  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         q42400  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     yak2_schpo  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     has1_yeast  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         q92732  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         o77001  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         o61815  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         o49530  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     dbp4_yeast  EAELPALTLP TSRRAQKKAL SKKASLASKG NASKLIFDDE GEAHPV..YE
     ddxx_human  GSQAP..SLP NTSEAQKIKE VPTQFLDRDE EEEDADF.LK VKRHNVFGLA
     yoq2_caeel  AEEEDDAQND IRLLKRIKRG KLSKKEIKDV L~~~~~~~~~ ~~~~~~~~~~
     spb4_yeast  AIEEELKAEE LDENAEEERI KEDWKEI..V LQNKRKKVSS KAIQGNFDDL
contig3000_orf1  TSSDEESQST SGAFSAKHTN NSAQDESDES ELSSYTSASE EHSGTTFPNE

                 801                                                850
         o48546  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         q42400  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     yak2_schpo  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     has1_yeast  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         q92732  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         o77001  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         o61815  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         o49530  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     dbp4_yeast  LEDEEEFHKR GDAEVQKTEF LTK...ESAV MADIDNIDKQ VAKEKKQEKK
```

```
     ddxx_human  LKDEKTLQKK DPSNSSIKKK MTKVAEAKKV MKRNFKVNKK ITFTDEGELV
     yoq2_caeel  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     spb4_yeast  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
contig3000_orf1  ASHVSRLQQR IAHNDSFDRE AHKRKNRRKS KRRAASEQES SYDDSSFDES

                 851                                                900
         o48546  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~MKSFNTE
         q42400  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~MKSFNTE
     yak2_schpo  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     has1_yeast  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         q92732  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         o77001  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         o61815  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
         o49530  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     dbp4_yeast  RKRLEAMRRE MEAAMEEEIS GDEEEGKTVA YLGTGNLSDD MSDGDMPDSE
     ddxx_human  QQWPQMQKSA IKDAEEDDDT GGINLHKAKE RLQEEDKFDK EEYRKKIKAK
     yoq2_caeel  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
     spb4_yeast  ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~
contig3000_orf1  EEEMQSKRKQ KP~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~ ~~~~~~~~~~

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

Notice the listing of sequence names near the top of the file. This listing contains an important number called the checksum. All GCG sequence programs utilize 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. Another important number on the individual checksum lines needs to be pointed out. The "Weight" designation determines how much importance each sequence contributes to a profile made of the alignment (see the Profile section below). 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. The "Sequence Info . . ." window can be used to accomplish this. However, we will not be bothering with it here.

After scrolling through your alignment and then "**Close**"ing its window, the next window visible will be the "**SeqLab Output Manager**." This important window contains all of the output from your current SeqLab session. Be sure to press the "**Add to Editor**" button and specify "**Overwrite old with new**" in the next window when prompted, to take your new MSF (Multiple Sequence Format) output and merge it with the RSF file in the open editor. This will keep all feature information intact, yet renumber all of its reference locations. "**Close**" the "**Output Manager**" after loading your new alignment. The next window will contain PileUp's cluster dendrogram; in my example's case, the following:



This dendrogram of the similarity clustering relationships between the sequences is automatically created when you run PileUp. It shows the clustering process used to create the alignment. The length of the vertical lines is proportional to the difference in similarity between the sequences. This figure is not an evolutionary or phylogenetic tree and should not be presented as one, although if the rates of evolution for each lineage are exactly the same, which is seldom the case in nature, it can be the same as one. It is basically a UPGMA style dendrogram — no substitution models for multiple hits or methods for correction of unequal rates of divergence are used in its construction. It merely indicates the relative similarity of the sequences. However, as discussed above, the dendrogram can assist in determining sequence weighting factors to even out each sequences' contribution to a profile. It can also help point out sequences that should probably be excluded from the analysis; here O48546 and Q42400 are distinct outliers.

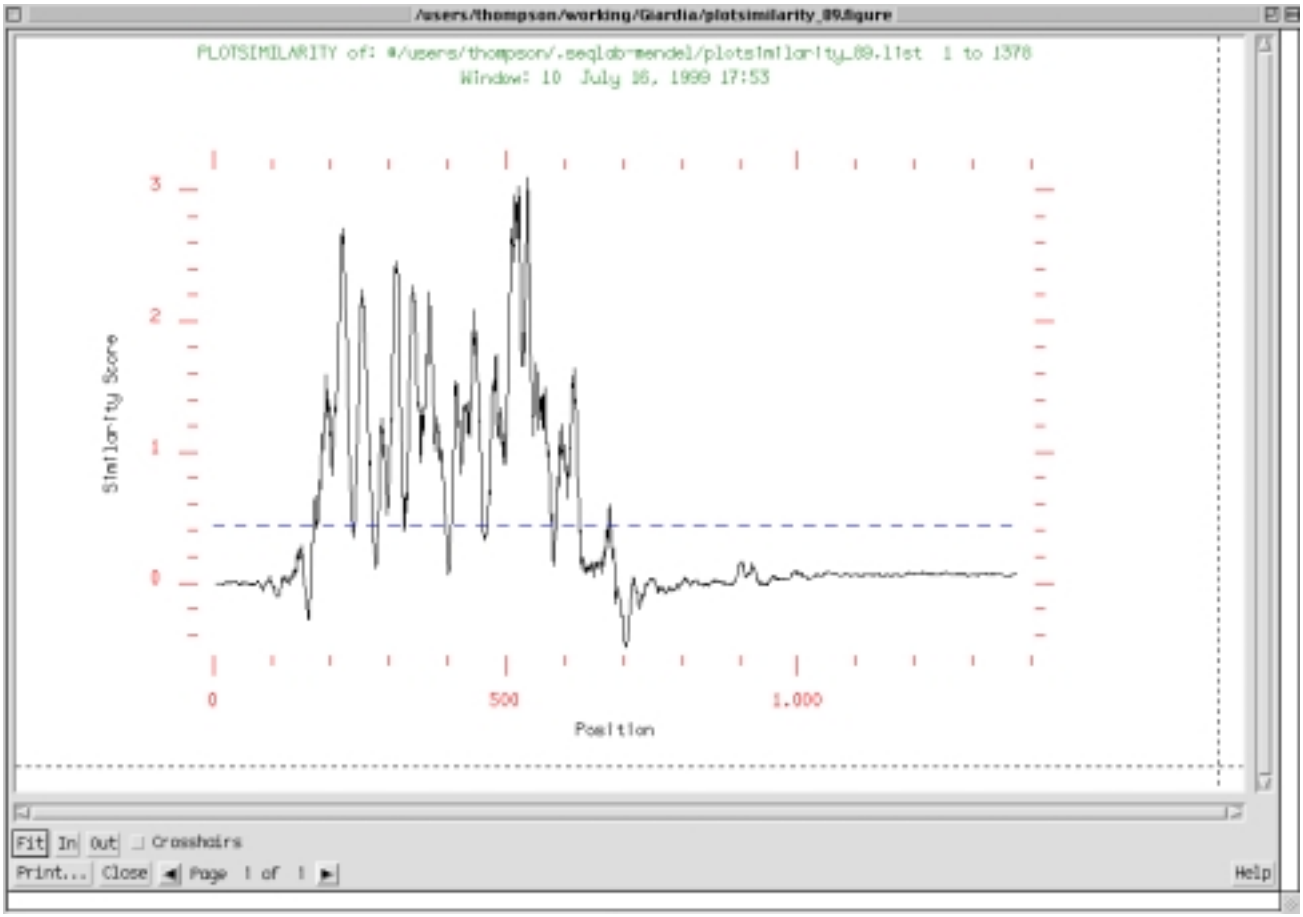
"**Close**" the dendrogram window to return to the editor. Now notice that your residues align by color. My editor display looks like the following after loading the MSF file:

Notice the typical ATP-dependent helicase ‘DEAD’ box signature in most of the sequences visible in this section.

15) Visualizing conservation in multiple sequence alignments.

To easily visualize the most conserved portions of a multiple sequence alignment we can utilize the GCG graphics program PlotSimilarity. Additionally this is a very nice way to see those areas of your alignment that may need improving by pointing out the most variable regions. This program draws a graph of the running average similarity along a group of aligned sequences (or of a profile with the Profile option).

Be sure that only the sequence names within the alignment are selected and then go back to the “**Functions**” menu; under the “**Multiple Comparison**” section choose “**PlotSimilarity. . .**” We need to change some of the program defaults there so choose “**Options. . .**” 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 and the second specification launches the program’s command line -expand option which 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 will use the similarity values from whichever symbol comparison matrix you specify. The default matrix, BLOSUM62, begins its identity value at 4 and ranges up to 11; mismatches go as low as -4. “**Close**” the 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 below:



Make a PostScript file of this plot too, if desired. "**Close**" the window. 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 the overall average similarity indicated by the blue dashed line is pretty poor, nonetheless, it is easy to see that some regions are much better than others, especially that portion before about 150 and after around 600. Now go to the "**File**" menu and click on "**Open Color Mask Files**." This will produce another window from which you can select your new "**plotsimilarity.cmask**" file. Click on "**Add**" and then "**Close**" the window. This will produce a gray scale overlay on your sequences that describes their regional similarity. My sample alignment, using a zoom factor of 8 to 1, looks like the following:

16) Improving alignments within SeqLab.

The beauty of this representation is you can now select only those regions of low similarity to try to improve their alignment automatically. This is possible because of PileUp's InSitu option. Be sure that all of your sequences in the alignment are selected and then zoom back in your alignment to 1:1 so that you can see individual residues and then scroll to the 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, either by using the mouse or by using 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). It may help you recognize similarity by switching back and forth between "**Residue Coloring**" and "**Color Mask**" "**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. (Reminder: The "**Windows**" menu also contains a 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. The BLOSUM30 natrix should still be selected if you had it chosen earlier. I would definitely recommend using it in this dataset because of the high level of sequence divergence. 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 **5** and **1** respectively seems to work pretty well for the BLOSUM30 matrix. You will have to experiment to see what works best for you. 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 "**Options**" 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 matrix used and the lowered gap introduction and extension penalties of 5 and 1 respectively.

Scroll through your alignment to check it out and then "**Close**" the window. The next window will be the "**Output Manager**." Just like before, if you like the alignment, 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 information. This feature information may help guide 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. As before, 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. If needed, 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. I did this with both O48546 and Q42400. 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 FastA or BLAST run, allowing SeqLab to trim the ends based on beginning and ending constraints considerably improves this situation. Overall, things to look for include strongly conserved residues such as tryptophans, cysteines, and histidines, important structural amino acids such as prolines, tyrosines and phenylanines, and the conserved isoleucine, leucine, valine triumvirate; 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 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 evolution, 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.

Sometimes you may want to align DNA sequences along with their corresponding proteins (the "Group" function is very helpful for this) in order to perform phylogenetic analyses 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. That is not a problem here, but many people prefer to run phylogenetic analyses on DNA rather than protein regardless — the multiple substitution models are much more robust for DNA.

The logic to this approach is as follows: 1) The easy case where you can align the DNA. If the DNA sequences are alignable, then merely create your DNA alignment, use the Edit-Translate function to create aligned corresponding protein sequences, and then 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. In this case you need to load the protein sequences first, create their alignment, and then load the corresponding DNA sequences. 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 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. 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 to bad. The Group function keeps everyting 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.

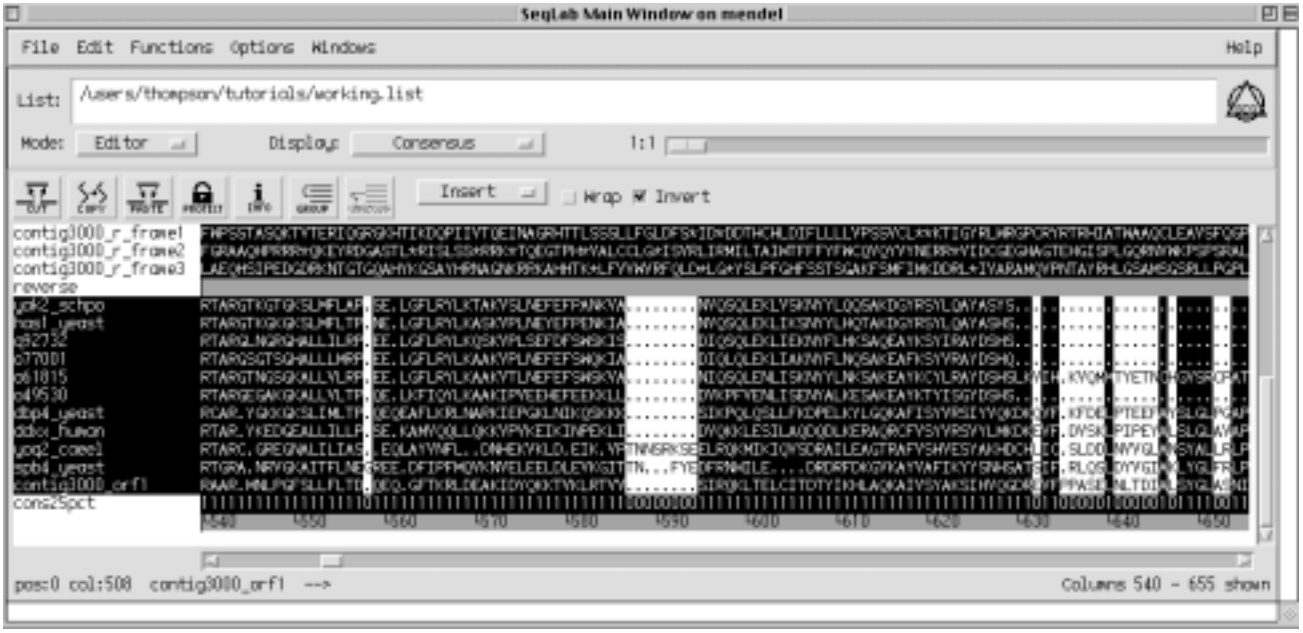
Many other alignment editors are available for cleaning up multiple sequence alignments. However, I think that you will find SeqLab most satisfying, and only using a GCG compatible editor assures that the format will not be corrupted. If you do make any changes to a GCG sequence data file with a non-GCG compatible editor, you must reformat the alignment afterwards. However, reformatting MSF (or RSF with the -rsf option) 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 MSF option of Reformat (or RSF with the -rsf option) and you must specify all the sequences within the file, i.e. "{*}," for example:

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

Here you will <u>not</u> need to perform this step, unless for some perverse reason you decided to edit your alignment with a non-GCG compliant editor such as pico; however, it may prove necessary in other situations. After reformatting, the new MSF or RSF file will follow GCG convention, with updated format, numbering, and checksums.

17) Masking and export format issues.

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. 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. Masks can be created manually also through the "New Sequences" menu and can have values all the way through 9. Masking can be very helpful for phylogenetic analysis by excluding those less reliable columns in your alignment where you are not confident in the positional homology. At this point be sure all of your alignment sequences are selected and then create a Mask style sequence consensus of them by going to the "**Edit**" "**Consensus . . .**" menu and specifying "**Consensus type:**" "**Mask Sequence**." 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**"); however, these are a very high values for phylogenetic analysis and would likely not leave much phylogenetically informative data. Therefore, experiment with 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. (At certain levels you can generate a gray intermediate similarity color as well as the black and white representation, if making a normal sequence consensus rather than a weight mask. This is a nice way to prepare alignment figures for publication.) The following screen illustrates my example using the BLOSUM30 matrix, a plurality of 25%, and a threshold cutoff value of 4:



Areas excluded by the Mask will be ignored by subsequent analyses. Once a Mask has been created in SeqLab any 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. Just like most computational molecular biology techniques, one is always balancing signal against noise — and it can be quite the balancing act! Too much noise or too little signal both degrade the analysis to the point of nonsense.

When you've found a combination that you like, go to the "**File**" "**Print. . .**" command and change "**Output Format:**" to "**PostScript**" in order to prepare a PostScript file of your SeqLab display. Play around with the other parameters as you like — 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 should result in a PostScript file of the alignment using the displayed coloring and the specified parameters to be created in the directory that you launched SeqLab from which can then be transferred to another machine for color PostScript printing, or for importing into PostScript savvy programs for further manipulation, or that can be printed to a black and white laser printer that will simulate the colors with gray tones. Unfortunately the format of this 'raw' PostScript file is different enough from a standard Encapsulated PostScript file that you may have to use a different print queue in many instances. Discuss these matters with your system administrator. It may require some variation of the following type of command:

```
% lpr -PPostScript_que seqlab_alignment.ps
```

Return to the "**SeqLab Main Window**" and go to the "**File**" "**Export**" menu; click "**Format**" in the new window and notice that "MSF," "GenBank," and "GDE2.2" are all available for saving a copy of your RSF file in some alternative formats. At this point do not export any of these formats and "**Cancel**" the window. Be sure to realize that using this export route does not use the Mask data to include or exclude columns from your alignment. Since we want to take advantage of the Mask data for any subsequent phylogenetic analyses, we will export our alignment using another method. Therefore, after being sure that all of your alignment sequences as well as your Mask are selected, go to the "**Functions**" menu, where all choices will be affected by the Mask, and choose "**Importing/Exporting**" "**ToFastA . . .**." No options are required here; just press "**Run**" to convert your alignment into FastA format. We will use FastA as a good intermediate format on our way to PHYLIP's required format. The new file will be displayed by SeqLab; 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 dash 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. You can next use ReadSeq to convert this FastA format file to PHYLIP compatible format.

Temporarily switch to your terminal window to run Don Gilbert's program ReadSeq that 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. Begin the program by typing "**readseq**." It 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 file name (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, and let it do its thing. A sample screen trace is shown below; as usual, responses are shown in bold:

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

Name of output file (?=help, defaults to display):
contig3000.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:
contig3000.tfa
Sequences in contig3000.tfa  (format is  8. Pearson/Fasta)
 1)  YAK2_SCHPO In situ PileUp of: @/users/thompson/.seqlab-mendel/pileup_98.lis
 2)  HAS1_YEAST In situ PileUp of: @/users/thompson/.seqlab-mendel/pileup_98.lis
 3)  Q92732 In situ PileUp of: @/users/thompson/.seqlab-mendel/pileup_98.list
 4)  O77001 In situ PileUp of: @/users/thompson/.seqlab-mendel/pileup_98.list
 5)  O61815 In situ PileUp of: @/users/thompson/.seqlab-mendel/pileup_98.list
 6)  O49530 In situ PileUp of: @/users/thompson/.seqlab-mendel/pileup_98.list
 7)  DBP4_YEAST In situ PileUp of: @/users/thompson/.seqlab-mendel/pileup_98.lis
 8)  DDXX_HUMAN In situ PileUp of: @/users/thompson/.seqlab-mendel/pileup_98.lis
 9)  YOQ2_CAEEL In situ PileUp of: @/users/thompson/.seqlab-mendel/pileup_98.lis
 10)  SPB4_YEAST In situ PileUp of: @/users/thompson/.seqlab-mendel/pileup_98.li
 11)  CONTIG3000_ORF1 In situ PileUp of: @/users/thompson/.seqlab-mendel/pileup.

Choose a sequence (# or All):
all

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

You may get the program notice "`This format requires equal length sequences.  Sequence truncated or padded to fit.`" Don't mind it — the program is just doing what it is supposed to do. Do realize, though, that had we 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 in the Introduction, 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 very well for this step from the command line, but you should not need to use it in this exercise:

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

Return to your SeqLab display to generate a NEXUS file for PAUP*. To build NEXUS format files it is easiest to use GCG's interface to the PAUP* package. The interface is the paired programs PAUPSearch and PAUPDisplay. However, I do not recommend seriously analyzing your dataset with PAUP* from within GCG through their PAUPSearch and PAUPDisplay programs because GCG's version of PAUP* in both GCG version 9.1 and 10 is an old 4.0.0d55 version, rather PAUPSearch can be used as a very handy tool for generating NEXUS format files which can then be fed directly to the most recent version of PAUP*. Naturally, if you do not have access to the latest and greatest version of PAUP*, which contains several bugs fixes since 4.0.0d55 was incorporated into the GCG package, then this is a legal alternative. If you absolutely have to rely on GCG's version of PAUP*, then at least bother to learn how to run the most robust searches possible, before accepting any of its output as possible phylogenetic inferences. However, I would strongly suggest contacting Sinauer Associates to acquire the current version of PAUP*.

Begin the NEXUS conversion process by again being sure all of your alignment sequences, including the weight Mask, are selected and then go to the "**Functions**" "**Evolution**" menu. Select "**PaupSearch. . .**" to launch the dialogue box. Because we merely want to generate a NEXUS file, we will run PAUPSearch in its fastest mode. 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. The PAUPScript output file results from the automatic conversion of your alignment to NEXUS format and contains all the PAUP commands as well as your alignment. (If needed, the PAUP log file keeps track of all that happened during the program run and is a good place to look for any error messages.) You can change or leave the file names as you wish. Uncheck the next box, "**Perform the anaysis**." 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 in 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, since we don't want to run PAUPDisplay, uncheck the "**PaupDisplay . . .**" button in PaupSearch's main window. Be sure that "**How:**" "**Background Job**" is specified on the main PAUPSearch menu and then press "**Run**" there. After a moment the results will be displayed. An abridged version of mine follows:

```
#NEXUS

[! Aligned sequences from GCG file(s) '@/users/thompson/.seqlab-mendel/paupsearc
h_100.list' ]

[Length: 589  Type: P  July 16, 1999 23:59]

[ Name: YAK2_SCHPO       Len:   589  Check: 1130  Weight:  1.00]
[ Name: HAS1_YEAST       Len:   589  Check: 8758  Weight:  1.00]
[ Name: Q92732           Len:   589  Check: 6220  Weight:  1.00]
[ Name: O77001           Len:   589  Check: 8242  Weight:  1.00]
[ Name: O61815           Len:   589  Check: 7881  Weight:  1.00]
[ Name: O49530           Len:   589  Check: 4054  Weight:  1.00]
[ Name: DBP4_YEAST       Len:   589  Check: 7409  Weight:  1.00]
[ Name: DDXX_HUMAN       Len:   589  Check:  310  Weight:  1.00]
[ Name: YOQ2_CAEEL       Len:   589  Check: 4903  Weight:  1.00]
[ Name: SPB4_YEAST       Len:   589  Check: 5093  Weight:  1.00]
[ Name: CONTIG3000_ORF1  Len:   589  Check: 6536  Weight:  1.00]

begin data;
     dimensions ntax=11 nchar=589;
     format datatype=protein interleave gap=.;
     matrix
[                1                                                    50]
     YAK2_SCHPO  M....AKKRK GNEEKKDAEE P........D EDDYEQEEE. AQNTSVEED.
     HAS1_YEAST  M....ATKR. ....S.DTEE P......... VVD.....E. SQNNAAP...
         Q92732  MGLQGM.SQE GNIKSKNASE KKKKKRK.DE PDTKKAKTEK S.EESAEEEE
         O77001  MELQGNKKQE GN.KKKDDQE PPKKKQKQDE SDDDEQEDED SLDEAAEEDE
         O61815  MDV..KKKRK GHEK...AEE PE.......E EDEEEVEEEK T..ESSE...
         O49530  MNLQSENKKK RDEAKADKDE PKKKKKNKDG EDEAVAEEEK KKNKKLQQDE
     DBP4_YEAST  MK......R. .T......TQ ...KTR.Q.K EDEY.IE... ..N.......
     DDXX_HUMAN  MKTPGARVRS KH......SH QKKKQRKQ.K KPEWQVEREI LQN.......
     YOQ2_CAEEL  MKV....... .......... .......... .......... ..........
     SPB4_YEAST  MSL....... .......... .......... .......... ..........
CONTIG3000_ORF1  M......... .......... .......... .......... ..........
```

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

[                501                                                  550]
     YAK2_SCHPO  .......IFD INKL..DLKV ....AKS.FG FAH..PPVNI IASGRK....
     HAS1_YEAST  .......VYQ IDKL..DLKV ....AKS.YG FPV..PPVNI IASGKN....
         Q92732  .......IFN VNNL..NLQV ....ALS.FG FKV..PPVDL VNSNEK....
         O77001  .......IFN VNTL..DLAV ....AKS.FG FLV..PPVDL VARPSD....
         O61815  P...TYDIFD VTNM..DLAV ....SKS.FG FSV..PPVDL INKPKI....
         O49530  ......DVFN VHQL..NLEV ....ATS.FG FSD..PPVAL I.........
     DBP4_YEAST  .KMETLHYLN ITKAQEDFSV HIAN.ASKLI FDDEEPVE.. LDEEE.LDDE
     DDXX_HUMAN  EEMEGQSLPN TSEAQEDFKV HVANVNKKIT FTDEEPQQKS IDAEEDLDDK
     YOQ2_CAEEL  PKMEQKDMF. ......DRAI ....ETSEIK YADVLET... ....MK...E
     SPB4_YEAST  PRMETKGIFP GNWL..D.PV ....NMDEYK YKDKRETKNI LNDKKK...E
CONTIG3000_ORF1  LKVKSKE... ..ELQEARAR HIANAQDE.. .SDEE...SS TASEE.LDD.

[                551                                       589]
     YAK2_SCHPO  ...ERR.... .AG......Y NKKNHV.YSI SQD.RGR..
     HAS1_YEAST  ...KRR.... .......... ..KTH..... ......K..
         Q92732  ...KKR.... .GGGGGF..Y QKTKKVSKIK SSD.RQH..
         O77001  ...KSD.... .VGGGGF..Y KKMNEGSKFV NRD.KKR..
         O61815  ...SKL.... .SGAG....Y RKKKQ.SF.. ....KK...
         O49530  ....DR.... .GG......Y RSKREPKFGR PGG.KFY..
     DBP4_YEAST  KKEMRREAME EGKTVAYDD. D...MP...D SEG.KK..P
     DDXX_HUMAN  KKKARREAKE E....AFDDF DPSTLPDKED SEDIKKGKP
     YOQ2_CAEEL  KAKKRREAKF NGGGTGR... .EEKKALKE. .DDIKRRK.
     SPB4_YEAST  KWKKERE.K. ...SLKR..I EEELKALDEI KEDI.QRK.
CONTIG3000_ORF1  .K.NRR..KE ES...SYDDF D........E SE..EESRP

     ;
endblock;


begin paup;
set errorstop;
set criterion=parsimony;
set increase=no;
pset collapse=no;
hsearch start=stepwise addseq=simple swap=tbr;
savetrees /brlens file='/users/thompson/working/Giardia/paupsearch_100.pauptrees
' replace;
quit;
endblock;
```

The top-most file will be the PAUP* script file with the "**SeqLab Output Manager**" right behind it. This PAUPscript file is very important. As mentioned above, it 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. As I stated in the introduction, file format conversion can be the biggest headache of this whole area and here GCG has done all of that work for you. When using this file as input to native PAUP* you will need to delete, comment out, or modify any inappropriate commands within

the command block with a simple text editor. Likewise, this file can be greatly expanded by encoding any desired commands within its command block.

18) Profile Analysis: How to use ProfileMake to create a weighted matrix of the alignment and ProfileSearch to scan the database with that profile.

The Profile Suite: ProfileMake, ProfileSearch, and ProfileScan. And finally — interpreting Profile analysis — why even bother?

Dr. Gribskov et al. (1987 and 1992; see additional references in GCG program manual) have assembled an elegant package for associating distantly related proteins and discovering structural motifs with the Profile analysis suite. John Devereux of GCG has written an excellent overview essay of the method in the GCG program manual; please take the time to read this section at some point with the GenManual command.

The Profile method enables the researcher to recognize features that may otherwise be invisible. The greatly enhanced information content of a Profile over individual sequences has the potential to find similar motifs in sequences which may be only distantly related and that will not be found by any other search algorithms. Even though ProfileSearches do require some work to setup and run — a meaningful multiple sequence alignment must be assembled, the sequences should be appropriately weighted, ProfileMake needs to be run, and the search job itself takes quite a long time to run — it is well worth the bother.

A profile should usually be refined to only include the most highly conserved area of an alignment and its members should be appropriately weighted. This refinement procedure, including repeatedly searching the databases and including or excluding members as the case may be, is known as validating the profile. If using Profile analysis in your own research, following the validation procedures outlined in the GCG Program Manual in the ProfileScan description is a very prudent idea, but we do not have the time for that now. However, we will restrict the length of our profile to exclude the diverged terminal regions of our alignment. (By the way, ProfileScan is another great 'Motifs'-like program to run on unknown protein sequences to help ascertain function.)

A profile, and its inherent consensus, is created with the GCG program ProfileMake. Be sure that all of your alignment sequences except the consensus mask are selected and then, based on your previous observations, select the longest, most conserved, overall length available. Restrict the length of your profile so that any jagged ends in your alignment are excluded. 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 around the similarity peaks of your alignment. After your range is selected go to "**Functions**" "**Multiple Comparison**" "**ProfileMake**" and reply "**Selected region**" in the "**Which selection**" dialog box. Go to the "**Options . . .**" menu from the "**ProfileMake**" dialog box and 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. "**Close**" the "**Options**" box and then "**Run**" ProfileMake. The top window returned will display your profile consensus sequence. Notice that all positions are filled; there are no gaps. This is because the Profile algorithm will decide on the most conserved residue for each position, regardless. Also notice that the header contains information relating to the sequence's creation through ProfileMake; this can be valuable.

"**Close**" the window. The " **Output Manager** " will also list a .prf file. This is the profile itself. You are welcome to take a look. It is a huge 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 in the Introduction, other programs can read and interpret this alignment customized scoroing matrix to perform very sensitive database searches and alignments by utilizing the information within the matrix that penalizes misalignments in phylogenetically conserved areas more than in variable regions. Whatever you do, "**Save As. . .**" the profile giving it an appropriate name that you can recognize; retain the **.prf** extension. "**Close**" the "**Output Manager**."

Follow the instructions in this next paragraph to run ProfileSearch on your own data at some point in the future. DO NOT RUN PROFILESEARCH WHILE IN THIS WORKSHOP. It would load the cpu too heavily and negatively impact other participants! To run ProfileSearch at some point in the future, go to the "Functions" menu and select "Database Sequence Searching" "ProfileSearch." Specify the "Query profile. . ." in the "File Chooser" and click "OK." Uncheck "ProfileSegments. . ." and then go to "Options. . .." Use the MinList option by changing "Lowest Z score to report in output list" from 2.5 to 3.5 and then "Close" the command window. MinList sets a list Z score cut-off value — a very handy way to limit your output list size. ProfileSearch Z scores are normalized and reflect the significance of the results. Here rather than randomizing sequences to evaluate the Z score as is done in the Monte Carlo approach with the Randomization option of GCG's programs Gap and BestFit, they are calculated based on all of the nonsimilar sequences from the database search similar to the way that BLAST and FastA calculates their Probability/Expectation scores. As with Monte Carlo approaches, Z scores much below 3 are probably not worth considering, from around 4 to 7 may be interesting , and above 7 are most probably significant. Be sure that "How:" "Background Job" is selected and then click on "Run."

Now get out of SeqLab by going to the "**File**" menu and clicking on "**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 and SeqLab will close. This will return you to the xterm window.

19) Conclusions and Profile results.

What Profile analysis can show us.

Obviously, even in such an extensive tutorial, I have only touched the 'tip of the iceberg' regarding SeqLab's full potential. For example, I haven't discussed primer design and analysis or protein or nucleotide structural analysis — three 'hot' fields, readily ammenable to SeqLab study. Please refer to the printed or online GCG documentation on SeqLab available through GenHelp or the Help menus in SeqLab itself to fully explore its many possibilities. Also, freel free to contact me in the future at stevet@bio.fsu.edu. SeqLab is an incredibly powerful way to run the Wisconsin Sequence Analysis Package.

ProfileSearches take a very long time to run, they are incredibly cpu intensive, perhaps the most of any program in the GCG package, so if you ever do perform one, be sure to submit your search as early as possible. When you return to a completed ProfileSearch take a careful look at the output. There is a strong chance that some of the sequences in it were not found by other search algorithms. If launched from SeqLab, it'll be located in the same directory as you were working in and will have a cryptic name of the form profilesearch_some-number.pfs. Pay particular attention to the reported Z scores. Notice that in my example ProfileSearch output file below, the program is finding all of the RNA helicases plus some other interesting nucleotide binding proteins such as initiation factors. The nucleotide binding motifs in our profile are among the most highly conserved portions of the alignment and hence more importance are placed on them by the

search, therefore, other proteins with similar domains are found. An abridged screen trace of my ProfileSearch output follows below:

```
!!SEQUENCE_LIST 1.0
(Peptide) PROFILESEARCH of: /users/thompson/working/Giardia/profilemake_103.prf
Length: 441 to: SwissProt:*

         Scores are not corrected for composition effects

                 Gap Weight: 43.05
          Gap Length Weight: 0.48
         Sequences Examined: 71064
         CPU time (seconds): 1158
*    *    *    *    *    *    *    *    *    *    *    *    *    *    *    *
Profile information:
(Peptide) PROFILEMAKE v4.50 of:
 @/users/thompson/.seqlab-mendel/profilemake_103.list  Length: 441
  Sequences: 11  MaxScore: 1094.30  July 17, 1999 00:41
                          Gap: 1.00              Len: 1.00
                     GapRatio: 0.33         LenRatio: 0.10

*    *    *    *    *    *    *    *    *    *    *    *    *    *    *    *
Normalization:                                    July 17, 1999 01:12

         Curve fit using 47 length pools
         0 of 47 pools were rejected

         Normalization equation:

                 Calc_Score = 553.44 * ( 1.0 - exp(-0.0013*SeqLen - 0.2731) )

         Correlation for curve fit: 0.719

         Z score calculation:
         Average and standard deviation calculated using 70959 scores
         105 of 71064 scores were rejected

                 Z_Score = ( Score/Calc_Score - 0.983 ) / 0.095

          Sequence  Strd ZScore   Orig Length ! Documentation  ..
SW:HAS1_YEAST         +   17.33 892.88    505 ! Q03532 saccharomyces cerevisiae
(baker's yeast). probable atp-dependent rna helicas
SW:YAK2_SCHPO         +   15.65 887.85    578 ! Q09916 schizosaccharomyces pombe
 (fission yeast). putative atp-dependent rna helica
SW:DBP4_YEAST         +    8.22 711.90    770 ! P20448 saccharomyces cerevisiae
(baker's yeast). probable atp-dependent rna helicas
SW:YOQ2_CAEEL         +    7.99 626.14    578 ! P34640 caenorhabditis elegans. p
utative atp-dependent rna helicase zk512.2 in chrom
SW:Y669_METJA         +    7.80 510.18    367 ! Q58083 methanococcus jannaschii.
 putative atp-dependent rna helicase mj0669. 11/97
SW:SPB4_YEAST         +    7.79 631.78    606 ! P25808 saccharomyces cerevisiae
(baker's yeast). atp-dependent rrna helicase spb4.
SW:YN21_CAEEL         +    7.70 574.23    489 ! P34580 caenorhabditis elegans. p
utative atp-dependent rna helicase t26g10.1 in chro
SW:DDXX_HUMAN         +    7.67 724.47    875 ! Q13206 homo sapiens (human). pro
bable atp-dependent rna helicase ddx10 (deah box pr
SW:DBP8_YEAST         +    7.15 527.25    431 ! P38719 saccharomyces cerevisiae
(baker's yeast). probable atp-dependent rna helicas
```

```
SW:RRP3_YEAST         +    7.04 578.31    543 ! P38712 saccharomyces cerevisiae
(baker's yeast). atp-dependent rrna helicase rrp3.
SW:RHLB_ECOLI         +    7.01 517.18    420 ! P24229 escherichia coli. putativ
e atp-dependent rna helicase rhlb. 11/97
SW:IF4A_CAEEL         +    6.93 505.23    402 ! P27639 caenorhabditis elegans. e
ukaryotic initiation factor 4a (eif-4a). 11/97
SW:SRMB_ECOLI         +    6.91 526.79    444 ! P21507 escherichia coli. atp-dep
endent rna helicase srmb. 11/97
SW:IF4N_SCHPO         +    6.88 499.40    394 ! Q10055 schizosaccharomyces pombe
 (fission yeast). eukaryotic initiation factor 4a-l
SW:IF4A_SCHPO         +    6.76 494.87    392 ! P47943 schizosaccharomyces pombe
 (fission yeast). eukaryotic initiation factor 4a (
SW:IF4A_CANAL         +    6.69 495.43    397 ! P87206 candida albicans (yeast).
 eukaryotic initiation factor 4a (eif-4a). 11/97

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

d box protein 3 (dead-box rna helicase dead2). 11/9
SW:Y425_MYCPN         +    4.40 452.51    450 ! P75172 mycoplasma pneumoniae. pr
obable rna helicase mg425 homolog. 11/97
SW:DEAD_HAEIN         +    4.36 514.72    613 ! P44586 haemophilus influenzae. a
tp-dependent rna helicase dead homolog. 11/95
SW:YAJ3_SCHPO         +    4.35 558.67    754 ! Q09903 schizosaccharomyces pombe
 (fission yeast). putative atp-dependent rna helica
SW:AN3_XENLA          +    4.29 539.51    697 ! P24346 xenopus laevis (african c
lawed frog). putative atp-dependent rna helicase an
SW:Y308_MYCPN         +    4.28 430.89    409 ! P75335 mycoplasma pneumoniae. pr
obable rna helicase mg308 homolog. 11/97
SW:DDX4_RAT           +    4.04 535.27    713 ! Q64060 rattus norvegicus (rat).
dead box protein 4 (vasa homolog) (rvlg). 11/97
SW:DEAD_ECOLI         +    3.98 512.18    646 ! P23304 escherichia coli. atp-dep
endent rna helicase dead. 11/97
SW:DBP9_YEAST         +    3.97 494.27    594 ! Q06218 saccharomyces cerevisiae
(baker's yeast). probable atp-dependent rna helicas
SW:Y425_MYCGE         +    3.90 436.86    449 ! P47664 mycoplasma genitalium. pr
obable rna helicase mg425. 11/97
SW:DEAD_KLEPN         +    3.86 512.07    659 ! P33906 klebsiella pneumoniae. at
p-dependent rna helicase dead. 2/95
SW:DBP6_YEAST         +    3.85 502.01    629 ! P53734 saccharomyces cerevisiae
(baker's yeast). probable atp-dependent rna helicas
SW:VASA_DROME         +    3.81 510.81    661 ! P09052 drosophila melanogaster (
fruit fly). vasa protein. 2/96
SW:MAK5_YEAST         +    3.71 539.68    773 ! P38112 saccharomyces cerevisiae
(baker's yeast). atp-dependent rna helicase mak5. 1
SW:GLH1_CAEEL         +    3.67 519.60    707 ! P34689 caenorhabditis elegans. a
tp-dependent rna helicase glh-1. 11/95
```

The program ProfileSegments makes BestFit style alignments of the results of a ProfileSearch. A great option, -msf, in ProfileSegments allows you to prepare a multiple sequence alignment of the ProfileSearch segments. The full information content of the profile is utilized in this alignment procedure. Since you didn't run ProfileSearch here, you will not be able to run ProfileSegments; however, I will illustrate the output. The importance of the conserved portions of an alignment as reflected in the content of a profile is fully utilized in this alignment procedure. When you've checked out a ProfileSearch output, a nice idea is to edit it to exclude most of the sequences that you expected to be found by the search except a few positive controls. If you ever do this, be sure not mess with the header portion of the file! Then alignments can be made off of the modified ProfileSearch output file with ProfileSegments. If you do run ProfileSegments sometime in the

future, be sure to set your list size to include all of the sequences remaining in the ProfileSearch output and accept the rest of the defaults. An abridged example of the results of a ProfileSegments run is shown below. Notice how much different the alignments are, after the obvious ones, from the examples seen with other algorithms. Notice in the following examples how the conserved portions of the profile do not allow the corresponding portion of alignment to gap. 'Clustering' is much more critical to Profile analyses than any other method. This is because of profile analysis' variable gap penalties where conserved areas are not allowed to gap and variable regions are. My abridged output from the ProfileSegments follows:

```
(Local) PROFILESEGMENTS of: HAS1_YEAST  check: 7270  from: 1  to: 505

ID   HAS1_YEAST     STANDARD;      PRT;   505 AA.
DE   PROBABLE ATP-DEPENDENT RNA HELICASE HAS1. . . .

 to: profilemake_103.prf  check: 6788  from: 1  to: 441

 has1_yeast x profilemake_103.prf July 17, 1999 01:12  ..

                 .         .         .         .         .
S     12 SESTEEPVVDEKSTSKQNNAAPEGEQTTCVEKFEELKLSQPTLKAIEKMG 61
         ::               .                       ::::::.:|
P      1 SE..............N.....................TLKAIKEMG 12
                 .         .         .         .         .
S     62 FTTMTSVQARTIPPLLAGRDVLGAAKTGSGKTLAFLIPAIELLHSL.... 107
         |..:| :|:.:|||:: |:||:::::|||||||||::|::|::. :
P     13 FTTMTEIQARSIPPLLQGRDVLGAAKTGSGKTLAFLIPAIEMIYRLEQNT 62
                 .         .         .         .         .
S    108 .KFKPRNGTGIIVITPTRELALQIFGVARELME.FHSQTF..G..IVIGG 151
          :|.|:||:|:::|:|||||:.|:|:| .::::  |.::|  |  :::||
P     63 AKFMPRNGTGVIIISPTRELAMQIFGVLRELMEHYHHHTFSVGCQLVIGG 112
                 .         .         .         .         .
S    152 ..ANRRQEAEKL..MKGVNMLIATPGRLLDHLQNTKGFV...FKNLKALI 194
           .::: |::::   :|:|.|::||||:|||:|:: :|.   :.|:. |:
P    113 QEANRRAEAEKLLRNKGINILVATPGRLLDHLQNTPGFIARKFRNLQCLV 162
                 .         .         .         .         .
S    195 IDEADRILEIGF.EDEMRQIIKILPNEDRQSMLFSATQTTKVEDLA..RI 241
         :|||||::::|| |:::::|:..|| . ||::|||||: :::||:  |:
P    163 IDEADRILDIGFIEDEMRQIIKLLPKQNRQTMLFSATQTQKVEDLAIFRI 212
                 .         .         .         .         .
S    242 SLRPGPLFINVVPET.......DNSTADGLEQGYVV.C...DSDKRFLLL 280
         :|:. |..:. | :        ...| ::|:|:|:: |   .:::|::::
P    213 SLRPNPIYVG.VHDVMDGNQNKDNATPDGLEQGYIVECRVDPSDKRFLLL 261
                 .         .         .         .         .
S    281 .....FSFLKRNQ.KKKIIVFLSSCNSVKYYAELLNY......IDLP... 315
              |:|::::: ::|::||:::|::|:|. ::::|      |::|
P    262 SICNNFTFLKRNRLKKKIIVFFSSCNSVKYHYELFNYCLGKRNIDLPGVP 311
                 .         .         .         .         .
S    316 VLELHGKQKQQKRTNTFFEFCNAER.GILICTDVAARGLDIPAVDWIIQF 364
         ::.:||:|:|::||.||::||::  |:|:||||:|||:|||:|||||||
P    312 ILSIHGKQKQQKRTTTFFQFCNAETNGILFCTDVAARGLDIPAVDWIVQY 361
                 .         .         .         .         .
S    365 DPPDDPRDYIHRVGRTARGTKGKGKSLMFLTP..NELGFLRYLKASKVPL 412
         |||||:..||||:||||||. :.|::|::|.|   |::|::||::.|:::
P    362 DPPDDPRDYIHRVGRTARGTNGKGKALLFLTPGQEELGFLRYLKAAKVPY 411
                 .         .         .
S    413 NEYEFPENK..IANVQSQLEKLIKSNYYLH 440
         :|::|: :    .::|.::|.|:  ||::.
P    412 NEFEFEWNPKITANIQSQLEKLISKNYYLH 441
```

```
 (Local) PROFILESEGMENTS of: YAK2_SCHPO  check: 2426  from: 1  to: 578

ID   YAK2_SCHPO     STANDARD;      PRT;   578 AA.
DE   PUTATIVE ATP-DEPENDENT RNA HELICASE C1F7.02C. . . .

 to: profilemake_103.prf  check: 6788  from: 1  to: 441

yak2_schpo x profilemake_103.prf July 17, 1999 01:12  ..

                 .         .         .         .         .
S      4 SELKRKKHQSGNEEVKEKRQKPLKNDKKIAEELPQDEDDYEQEEENEDAD 53
         ::
P      1 SE..................................................... 2
                 .         .         .         .         .
S     54 QNTSVESESEELDNENEDERVQKSVNLNASSTSDIEKFSDLQLSENIQKA 103
          .:        :                                     ::
P      3 .NT........L.................................KA 7
                 .         .         .         .         .
S    104 IKEMGFETMTEIQKRSIPPLLAGRDVLGAAKTGSGKTLAFLIPTIEMLYA 153
         :.::|| .:|:|| .:|||:: |:||:::::|||||||||::|.:|:::
P      8 IKEMGFTTMTEIQARSIPPLLQGRDVLGAAKTGSGKTLAFLIPAIEMIYR 57
                 .         .         .         .         .
S    154 L.....KFKPRNGTGVIIISPTRELALQIFGVAKELLK.YHHQTF..G.. 193
         :     :|.|:||:|::||:|||||:.|:|:| .::.. :|.::|  |
P     58 LEQNTAKFMPRNGTGVIIISPTRELAMQIFGVLRELMEHYHHHTFSVGCQ 107
                 .         .         .         .         .
S    194 IVIGG..ANRRAEADKLV..KGVNLLVATPGRLLDHLQNTKGFV...FRN 236
         :::||  .:::.|::::   :|:|:||:||||:|||:|:: :|.   ::|
P    108 LVIGGQEANRRAEAEKLLRNKGINILVATPGRLLDHLQNTPGFIARKFRN 157
                 .         .         .         .         .
S    237 LRSLVIDEADRILEIGF.EDEMRQIMKILPSENRQTLLFSATQTTKVEDL 285
         :: ||:|||||::::|| |:::::|...|| . |||.||||||: :::||
P    158 LQCLVIDEADRILDIGFIEDEMRQIIKLLPKQNRQTMLFSATQTQKVEDL 207
                 .         .         .         .         .
S    286 A..RISLKPGPLYVNVDS......GKPTSTVEGLEQGYVVV....DSDKR 323
         :  |::|.. |.:|.|.        .  .| ::|:|:|::     .:::|
P    208 AIFRISLRPNPIYVGVHDVMDGNQNKDNATPDGLEQGYIVECRVDPSDKR 257
                 .         .         .         .         .
S    324 FLLL.....FSFLKRN.LKKKVIVFMSSCASVKYMAELLNY......IDL 361
         ::::     |:|::::  ::|::|| ::| :|:|. ::::|      |::
P    258 FLLLSICNNFTFLKRNRLKKKIIVFFSSCNSVKYHYELFNYCLGKRNIDL 307
                 .         .         .         .         .
S    362 P...VLDLHGKQKQQRRTNTFFEFCNAEK.GILLCTNVAARGLDIPAVDW 407
         |   :: :||:|:|:.||.||::||:::  |:|:||:|:|||:|||:|||
P    308 PGVPILSIHGKQKQQKRTTTFFQFCNAETNGILFCTDVAARGLDIPAVDW 357
                 .         .         .         .         .
S    408 IVQYDPPDDPRDYIHRVGRTARGTKGTGKSLMFLAP..SELGFLRYLKTA 455
         |||||||||:..||||:||||||. : |::|::| |   |::|::||: :
P    358 IVQYDPPDDPRDYIHRVGRTARGTNGKGKALLFLTPGQEELGFLRYLKAA 407
                 .         .
S    456 KVSLNEFEFPANK..VANVQSQLEKLVSKNYYLQ 487
         |: ::|::|: :    .::|.::|.|:.:||::.
P    408 KVPYNEFEFEWNPKITANIQSQLEKLISKNYYLH 441


 (Local) PROFILESEGMENTS of: DBP4_YEAST  check: 893  from: 1  to: 770
```

```
ID   DBP4_YEAST     STANDARD;      PRT;   770 AA.
DE   PROBABLE ATP-DEPENDENT RNA HELICASE DBP4 (HELICASE CA4) (HELICASE . . .

 to: profilemake_103.prf  check: 6788  from: 1  to: 441

dbp4_yeast x profilemake_103.prf July 17, 1999 01:12  ..

                .         .         .         .         .
S      2 AKKNRLNTTQRKTLRQKEDEYIENLKTKIDEYDPKITKAKFFKDLPISDP 51
          .    .:                ::  :.:
P      1 SE....NT.................LKA.IKE.................. 10
                .         .         .         .         .
S     52 TLKGLRESSFIKLTEIQADSI.PVSLQGHDVLAAAKTGSGKTLAFLVPVI 100
           .|     |  :|:||: :| |. :.| ||:.:::|||||||||::|::
P     11 ..MG.....FTTMTEIQARSIPPL.LQGRDVLGAAKTGSGKTLAFLIPAI 52
                .         .         .         .         .
S    101 EKLYRE.....KWTEFDGLGALIISPTRELAMQIYEVLTKIGS..H..TS 141
         | :::      :|.. :|.|.:||:|||||::|:: |: ::    |  :
P     53 EMIYRLEQNTAKFMPRNGTGVIIISPTRELAMQIFGVLRELMEHYHHHT. 101
                .         .         .         .         .
S    142 FSAG..LVIGG..KDVKFELERI..SR.INILIGTPGRILQHLDQAVGLN 184
         |  |  :::||   : . | :.:   . :|||:.||||:|.|: |    :
P    102 FSVGCQLVIGGQEANRRAEAEKLLRNKGINILVATPGRLLDHL.Q....N 146
                .         .         .         .         .
S    185 T........SNLQMLVLDEADRCLDMGF.KKTLDAIVSTLSPS..RQTLL 223
         :         |::.||:|||| :|:|| :  : .|:  | |   |||.|
P    147 TPGFIARKFRNLQCLVIDEADRILDIGFIEDEMRQIIKLL.PKQNRQTML 195
                .         .         .         .         .
S    224 FSATQSQSVADLA..RLSL.TD..YKTVGTHDVMDGSVNKEASTPETLQQ 268
         |||||:. : ||:  |::| ..  :  |:.::      ... .|:: |:|
P    196 FSATQTQKVEDLAIFRISLRPNPIY..VGVHDVMDGNQNKDNATPDGLEQ 243
                .         .         .         .         .
S    269 FYIEV.....PLADK..LDIL.....FSFIK.SHL.KCKMIVFLSSSKQV 304
          |: :     |  ::  : ::     |:|::  .  : | :||::: . |
P    244 GYI.VECRVDP.SDKRFL.LLSICNNFTFLKRNRLKK.KIIVFFSSCNSV 289
                .         .         .         .         .
S    305 HFVYETFRKMQ.......PGISLMHLHGRQKQRARTETLDKFNRAQQV.C 346
          | :: |           |.. .. :||:|:|  || |:  |  :.
P    290 KYHYELFNYCLGKRNIDLPGVPILSIHGKQKQQKRTTTFFQFCNAETNGI 339
                .         .         .         .         .
S    347 LFATDVVARGIDFPAVDWVVQVDCPEDVDTYIHRVGRCAR.YGKKGKSLI 395
         |: |||.|||:|:|:|||||||.|:|.|   ||||:|| ||  : .|::|:
P    340 LFCTDVAARGLDIPAVDWIVQYDPPDDPRDYIHRVGRTARGTNGKGKALL 389
                .         .         .         .         .
S    396 MLTP.QEQEAFLKRLNARKIEPGKLNIKQSKKKSIKPQLQSLLFKDPELK 444
          |.| .|: .|:..| : |::  :. .. . :  .  | :   :    :
P    390 FLTPGQEELGFLRYLKAAKVPYNEFEFEWNPKITANIQSQLEKLISKNYY 439

S    445 Y 445
         :
P    440 L 440

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

(Local) PROFILESEGMENTS of: IF41_ARATH  check: 9135  from: 1  to: 412

ID   IF41_ARATH     STANDARD;      PRT;   412 AA.
DE   EUKARYOTIC INITIATION FACTOR 4A-1 (EIF-4A-1). . . .
```

```
 to: profilemake_103.prf  check: 6788  from: 1  to: 441

 if41_arath x profilemake_103.prf July 17, 1999 01:12  ..

                .         .         .         .         .
S      4 SAPEGTQFDARQFDQKLNEVL.E.GQDEFF...TSYDDVHESFDAMGLQE 48
         :  : :          : . : : |    |   |      :      :|
P      1 S..ENT.........L.KAIKEMG....FTTMT......E......IQA 21
                .         .         .         .         .
S     49 NLLRGIYAYGFEKPSAIQQRGIVPFCKGLDVIQQAQSGTGKT.ATFCSGV 97
             . |       |         |.  | ||:  : :|:||| | |   :
P     22 ...RSI.......P.........PLLQGRDVLGAAKTGSGKTLA.FL..I 49
                .         .         .         .         .
S     98 ..LQ...QLD.....FS....L.IQCQALVLAPTRELAQQIEKVMRALGD 132
           .:    :      |     . :    :::.|||||: |:
P     50 PAIEMIYRLEQNTAKFMPRNGTGV....IIISPTRELAMQI......... 86
                .         .         .         .         .
S    133 YLGV..KV....HA......C..V.GGT.SVR..EDQRILQA.GVHVVVG 163
         : :|  :.    |       .  : ||  . :  | ..:    |: :.|.
P     87 F.GVLRELMEHYHHHTFSVGCQLVIGGQEANRRAEAEKLLRNKGINILVA 135
                .         .         .         .         .
S    164 TPGRVFDMLKRQ.S..L.....RADNIKMFVLDEADEMLSR....GF.KD 200
         ||||.:|::  | .  :     :  |:..:|:||||    |    || ::
P    136 TPGRLLDHL..QNTPGFIARKFR..NLQCLVIDEAD....RILDIGFIED 177
                .         .         .         .         .
S    201 ...QIYDIFQLLPP..KIQVGVFSATMPP..EALEITR.KFMSKPVRILV 242
            :|  :  .||.  . |  .||||     : |   |  :   |..: |
P    178 EMRQI..I.KLLPKQNR.QTMLFSATQTQKVEDLAIFRISLRPNPIYVGV 223
                .         .         .         .         .
S    243 ........KRDELTLEGIKQFYVNVEKEEWKLETLC.....D...LYETL 276
                ..  | ::::| |: :          |     :   :.  :
P    224 HDVMDGNQNKDNATPDGLEQGYI.VE.........CRVDPSDKRFLL..L 261
                .         .         .         .         .
S    277 AI....TQSVIFVNTR.R..KVDWLTDK.MRSRDHTV..S.......... 306
               :    |.  : :  :      | :      |  :
P    262 SICNNFT....FLK.RNRLKK......KII......VFFSSCNSVKYHYE 294
                .         .         .         .         .
S    307 ...................ATHGDMDQNTRDIIMREF....RSGSSRVLI 333
                           . ||   |  |     :|      |   :|:
P    295 LFNYCLGKRNIDLPGVPILSIHGKQKQQKRTTTFFQFCNAETNG...ILF 341
                .         .         .         .         .
S    334 TTDLLARGIDVQQVSLVINFDLPTQPENYLHRIGRSGR.FGRKGVAINFV 382
          ||. |||:|: .|  || || |  :  |:||:||: |  : .| :: :.
P    342 CTDVAARGLDIPAVDWIVQYDPPDDPRDYIHRVGRTARGTNGKGKALLFL 391

S    383 TRDDERMLF 391
         .   | . |
P    392 TPGQEELGF 400


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

The abridged multiple sequence alignment output that I created with the -msf option follows below:

```
!!AA_MULTIPLE_ALIGNMENT 1.0
Multiple alignment of profilemake_103.prf and
          has1_yeast          yak2_schpo          dbp4_yeast          yoq2_caeel
          y669_metja          spb4_yeast          yn21_caeel          ddxx_human
```

```
          dbp8_yeast          rrp3_yeast          rhlb_ecoli          if4a_caeel
          srmb_ecoli          if4n_schpo          if4a_schpo          if4a_canal
          if4a_drome          rhlb_haein          fal1_yeast          if48_tobac
          if4a_maize          rhle_ecoli          if4y_tobac          if41_arath
          srmb_haein

 contig3000_frame1.prf.msf  MSF: 1238  Type: P  July 17, 1999 01:12  Check: 4310
 ..

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

//

                    1                                                  50
         has1_yeast  S......EST EEPVVDEKST SKQ..... ..........  ..........
         yak2_schpo  S......ELK RKKHQSGNEE VKEKRQKPLK NDKKIAEELP QDEDDYEQEE
         dbp4_yeast  A......KKN RL........ ..........  ..........  ..........
         yoq2_caeel  S......KVG ..........  ..........  ..........  ..........
         y669_metja  S......D.. ..........  ..........  ..........  ..........
         spb4_yeast  S......K.. ..........  ..........  ..........  ..........
         yn21_caeel  S......DGE DNQKFLG... ..........  ..........  ..........
         ddxx_human  S......PGS GARPD..... ..........  ..........  ..........
         dbp8_yeast  A......D.. ..........  ..........  ..........  ..........
         rrp3_yeast  SKIVKRKEKK A.........  ..........  ..........  ..........
         rhlb_ecoli  S......K.. ..........  ..........  ..........  ..........
         if4a_caeel  ..........  ..........  ..........  ..........  ..........
         srmb_ecoli  S......ELE LDE.......  ..........  ..........  ..........
         if4n_schpo  .......E..  ..........  ..........  ..........  ..........
         if4a_schpo  SVI....E..  ..........  ..........  ..........  ..........
         if4a_canal  S......EGI TEIDSGLI..  ..........  ..........  ..........
         if4a_drome  G......P..  ..........  ..........  ..........  ..........
         rhlb_haein  S......Q..  ..........  ..........  ..........  ..........
         fal1_yeast  SFDR...EED QKLKF.....  ..........  ..........  ..........
         if48_tobac  .......M..  ..........  ..........  ..........  ..........
         if4a_maize  ..........  ..........  ..........  ..........  ..........
         rhle_ecoli  S......P..  ..........  ..........  ..........  ..........
         if4y_tobac  ..........  ..........  ..........  ..........  ..........
         if41_arath  SAP....E..  ..........  ..........  ..........  ..........
         srmb_haein  S......P..  ..........  ..........  ..........  ..........
profilemake_103.prf  S......E..  ..........  ..........  ..........  ..........

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

                    1101                                               1150
         has1_yeast  H......GKQ KQQKRTNTFF EFCNAER.G. ..I...LICT DVAARGLDIP
         yak2_schpo  H......GKQ KQQRRTNTFF EFCNAEK.G. ..I...LLCT NVAARGLDIP
         dbp4_yeast  H......GRQ KQRARTETLD KFNRAQQV.. ..C...LFAT DVVARGIDFP
         yoq2_caeel  H......GKC SNPHRASQIK AFSDS.TNG. ..V...MIST DVMARGIDIS
         y669_metja  H......GDL SQSQREKVIR LF.KQKKIR. ..I...LIAT DVMSRGIDVN
         spb4_yeast  H......GKL QTSARTKTLT AFTDSLSNS. ..V...LFTT DVAARGIDIP
         yn21_caeel  H......GQM SQEKRLGSLN KF.KSKARE. ..I...LVCT DVAARGLDIP
         ddxx_human  H......GRQ QQMRRMEVYN EFVRKRA.A. ..V...LFAT DIAARGLDFP
         dbp8_yeast  H......SQM PQQERTNSLH RF.RANAAR. ..I...LIAT DVASRGLDIP
         rrp3_yeast  H......GDL NQNQRMGSLD LF.KAGKRS. ..I...LVAT DVAARGLDIP
         rhlb_ecoli  HRVGLLTGDV AQKKRLRILD EF....TRGD LDI...LVAT DVAARGLHIP
         if4a_caeel  H......GDM DQAERDTIMR EF....RSGS SRV...LITT DILARGIDVQ
         srmb_ecoli  E......GEM VQGKRNEAIK RL....TEGR VNV...LVAT DVAARGIDIP
         if4n_schpo  H......GEM PQKERDAIMQ DF....RQGN SRV...LICT DIWARGIDVQ
         if4a_schpo  H......GDM DQAQRDTLMH EF....RTGS SRI...LITT DLLARGIDVQ
```

```
          if4a_canal  H......ADL PQAERDTIMK EF....RSGS SRI...LIST DLLARGIDVQ
          if4a_drome  H......GDM EQRDREVIMK QF....RSGS SRV...LITT DLLARGIDVQ
          rhlb_haein  HRVGLLTGDV AQKKRLSLLK QF....TDGD LDI...LVAT DVAARGLHIS
          fal1_yeast  H......GDM KQEERDKVMN DF....RTGH SRV...LIST DVWARGIDVQ
          if48_tobac  H......GDM DQNTRDIIMR EF....RSGS SRV...LITT DLLARGIDVQ
          if4a_maize  H......GDM DQNTRDIIMR EF....RSGS SRV...LITT DLLARGIDVQ
          rhle_ecoli  H......GNK SQGARTRALA DF....KSGD IRV...LVAT DIAARGLDIE
          if4y_tobac  H......GDM DQNTRDIIMR EF....RSGS SRV...LITT DLLARGIDVQ
          if41_arath  H......GDM DQNTRDIIMR EF....RSGS SRV...LITT DLLARGIDVQ
          srmb_haein  E......GEM AQTQRNNAID KL....KSG. ..IVTVLVAT DVAARGIDID
profilemake_103.prf   H......GKQ KQQKRTTTFF QFCNAETNG. ..I...LFCT DVAARGLDIP

                      1151                                          1200
          has1_yeast  AVDWIIQFDP PDDPRDYIHR VGRTARGTKG KGKSLMFLTP ..NELGFLRY
          yak2_schpo  AVDWIVQYDP PDDPRDYIHR VGRTARGTKG TGKSLMFLAP ..SELGFLRY
          dbp4_yeast  AVDWVVQVDC PEDVDTYIHR VGRCAR.YGK KGKSLIMLTP .QEQEAFLKR
          yoq2_caeel  DIDWVIQFDL PKHSSWFVHR AGRTARCGRE GNALILIASE QLAYVNFLDN
          y669_metja  DLNCVINYHL PQNPESYMHR IGRTGRAGKK GKAISIINRR EYKKLRYIER
          spb4_yeast  DVDLVIQLDP PTNTDMFMHR CGRTGR.ANR VGKAITFLNE GREE.DFIPF
          yn21_caeel  HVDMVINYDM PSQSKDYVHR VGRTARAGRS GIAITVVTQY DVEAYQKIEA
          ddxx_human  AVNWVLQFDC PEDANTYIHR AGRTAR.YKE DGEALLILLP ..SEKAMVQQ
          dbp8_yeast  TVELVVNYDI PSDPDVFIHR SGRTAR.AGR IGDAISFVT. ..........
          rrp3_yeast  SVDIVVNYDI PVDSKSYIHR VGRTAR.AGR SGKSISLVSQ YDLELIL...
          rhlb_ecoli  AVTHVFNYDL PDDCEDYVHR IGRTGRAGAS GHSISLACEE YALNLPAIET
          if4a_caeel  QVSLVINYDL PSNRENYIHR IGRSGR.FGR KGVAINFVTE ..........
          srmb_ecoli  DVSHVFNFDM PRSGDTYLHR IGRTARAGRK GTAISLVEAH DHLLLGKVGR
          if4n_schpo  QVSLVINYDL PANRENYIHR IGRSGR.FGR KGVAINFVT. ..........
          if4a_schpo  QVSLVINYDL PANRENYIHR IGRGGR.FGR KGVSINFVT. ..........
          if4a_canal  QVSLVINYDL PANKENYIHR IGRGGR.FGR KGVAINFVT. ..........
          if4a_drome  QVSLVINYDL PSNRENYIHR IGRGGR.FGR KGVAINFIT. ..........
          rhlb_haein  DVTHVFNYDL PDDREDYVHR IGRTGRAGES GVSISFACEE YAMNLPAIEE
          fal1_yeast  QVSLVINYDL PEIIENYIHR IGRSGR.FGR KGVAINFITK ..........
          if48_tobac  QVSLVINYDL PTQPENYLHR IGRSGR.FGR KGVAINFVTT DDERMLF...
          if4a_maize  QVSLVINYDL PTQPENYLHR IGRSGR.FGR KGVAINFVTR DDERIVF...
          rhle_ecoli  ELPHVVNYEL PNVPEDYVHR IGRTGRAAAT GEALSLVCVD EHKLLRDIEK
          if4y_tobac  QVSLVINYDL PTQPENYLHR IGRSGR.FGR KGVAINFVTK DDERMLF...
          if41_arath  QVSLVINFDL PTQPENYLHR IGRSGR.FGR KGVAINFVTR DDERMLF...
          srmb_haein  DVSHVMNFDL PYSADTYLHR IGRTARAGKK GTAVSFVEAH DYKLLGKIKR
profilemake_103.prf   AVDWIVQYDP PDDPRDYIHR VGRTARGTNG KGKALLFLTP GQEELGFLRY

                      1201                                  1238
          has1_yeast  LKASKVPLNE YEFPENK..I ANVQSQLEKL IKSNYYLH
          yak2_schpo  LKTAKVSLNE FEFPANK..V ANVQSQLEKL VSKNYYLQ
          dbp4_yeast  LNARKIEPGK LNIKQSKKKS IKPQLQSLLF KDPELKY.
          yoq2_caeel  HEKVKLDEIK VPTNNSRKSE ELRQKMIK.. ........
          y669_metja  AMKLKIKKLK F......... .......... ........
          spb4_yeast  MQVKNVELEE LDLEVK.... .......... ........
          yn21_caeel  NLGKKLDEYK CVENEVMVLV ERTQEATEN. ........
          ddxx_human  LLQKKVPVKE ..IKINPEKL IDVQKKLESI LAQDQDLK
          dbp8_yeast  .......... .......... .......... ........
          rrp3_yeast  .......... .......... .......... ........
          rhlb_ecoli  YIGHSIPVSK YN........ .......... ........
          if4a_caeel  .......... .......... .......... ........
          srmb_ecoli  YIEEPIK... .......... .......... ........
          if4n_schpo  .......... .......... .......... ........
          if4a_schpo  .......... .......... .......... ........
          if4a_canal  .......... .......... .......... ........
          if4a_drome  .......... .......... .......... ........
          rhlb_haein  YIGHSIPVSQ YETE...... .......... ........
```

```
         fal1_yeast  ......... ......... ......... ........
         if48_tobac  ......... ......... ......... ........
         if4a_maize  ......... ......... ......... ........
         rhle_ecoli  LLKKEIP... ......... ......... ........
         if4y_tobac  ......... ......... ......... ........
         if41_arath  ......... ......... ......... ........
         srmb_haein  YTEE...... ......... ......... ........
profilemake_103.prf  LKAAKVPYNE FEFEWNPKIT ANIQSQLEKL ISKNYYLH
```

Notice the 'gappiness' of the alignment due to the profile method used. This can be a very handy strategy for pregapping sequences in order to introduce them into existing alignments. This particular alignment may not be biologically meaningful, but you should be able to see the power of the technique illustrated.

Gunnar von Heijne in his quite readable treatise, *Sequence Analysis in Molecular Biology; Treasure Trove or Trivial Pursuit* (1987), provides a very 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."

Supplemental Information for Further Exploration:

Phillipp Bucher's Eukaryotic Promoter Database (EPD) (1995):

Dr. Bucher has assembled an extensive list of eukaryotic promoter regions compiled from the EMBL database. His database includes a user's manual, the sequence information itself, and an independent, journal abstracted data reference section for each entry. In order to be included in EPD an entry must:

1) be recognized by eukaryotic RNA POL II,
2) be active in eukaryotes (excludes phycophytes, fungi, myxomycetes, protists),
3) be experimentally defined or sufficiently similar to one defined as such,
4) be biologically functional,
5) be available in the current EMBL release,
6) be distinct from other promoters in the database.

EPD Release 48 (September 1996) has 1285 total promoter entries (846 independent entries).

References

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

Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research* 25, 3389-3402.

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

Bucher, P. (1990). Weight Matrix Descriptions of Four Eukaryotic RNA Polymerase II Promoter Elements Derived from 502 Unrelated Promoter Sequences. *Journal of Molecular Biology* 212, 563-578.

Bucher, P. (1995). The Eukaryotic Promoter Database EPD. EMBL Nucleotide Sequence Data Library Release 42, Postfach 10.2209, D-6900 Heidelberg.

Edelman, I., Olsen, S., and Devereux, J. (1998) *Program Manual for the Wisconsin Package*, Version 10. Genetics Computer Group (GCG), Madison, Wisconsin, USA 53711.

Felsenstein, J. (1993) PHYLIP (Phylogeny Inference Package) version 3.5c. Distributed by the author. Dept. of Genetics, University of Washington, Seattle, Washington, U.S.A.

Ghosh, D. (1990). A Relational Database of Transcription Factors. *Nucleic Acids Research* 18, 1749-1756.

Gilbert, D.G. (1990) ReadSeq, public domain software, Biology Department, Indiana University, Bloomington, Indiana, U.S.A.

Gribskov M., McLachlan M., Eisenberg D. (1987) Profile analysis: detection of distantly related proteins. *Proc. Natl. Acad. Sci. U.S.A.* 84, 4355-4358.

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

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, *Proc. 6th Annual Combinatorial Pattern Matching conference* (CPM '95).

Hawley, D.K. and McClure, W.R. (1983). Compilation and Analysis of *Escherichia coli* promoter sequences. *Nucleic Acids Research* 11, 2237-2255.

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.

Kozak, M. (1984). Compilation and Analysis of Sequences Upstream from the Translational Start Site in Eukaryotic mRNAs. *Nucleic Acids Research* 12, 857-872.

McLauchen, J., Gaffrey, D., Whitton, J. and Clements, J. (1985). The Consensus Sequences YGTGTTYY Located Downstream from the AATAAA Signal is Required for Efficient Formation of mRNA 3' Termini. *Nucleic Acid Research* 13 , 1347-1368.

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

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

Pearson, W.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.

Proudfoot, N.J. and Brownlee, G.G. (1976). 3' Noncoding Region in Eukaryotic Messenger RNA. *Nature* 263, 211-214.

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

Smith, R.F. and Smith, T.F. (1992). Pattern-Induced Multi-sequence Alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for comparative protein modeling. *Protein Engineering* 5, 35-41.

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. *CABIOS*, 10, 671-675.

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

Stormo, G.D., Schneider, T.D. and Gold, L.M. (1982). Characterization of Translational Initiation Sites in *E. coli*. *Nucleic Acids Research* 10, 2971-2996.

Swofford, D.L., PAUP (Phylogenetic Analysis Using Parsimony) (1989-1993) Illinois Natural History Survey, (1994) personal copyright, and (1997) Smithsonian Institution, Washington D.C., U.S.A.

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. (1987a) *Sequence Analysis in Molecular Biology; Treasure Trove or Trivial Pursuit.* Academic Press, Inc., San Diego, CA.

von Heijne, G. (1987b). SIGPEP: A Sequence Database for Secretory Signal Peptides. *Protein Sequences & Data Analysis* 1, 41-42.

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.