A Brief Introduction to Multiple Sequence Analysis through GCG’s SeqLab

The SeqLab Graphical User Interface (GUI) is a ‘front-end’ to the Wisconsin Sequence Analysis Package. It provides an intuitive alternative to the command line by allowing menu-driven access to over 140 different GCG programs and is a great way to develop, refine, and analyze multiple sequence alignments. So what’s so great about a multiple sequence alignment? They are:

- very useful in the development of PCR primers and hybridization probes;
- great for producing annotated, publication quality, graphics and illustrations;
- invaluable in structure/function studies through homology inference;
- essential for building HMMs and profiles for remote homology analyses; and
- required for molecular evolutionary phylogenetic inference techniques such as those in PAUP* (Phylogenetic Analysis Using Parsimony [and other methods]), PHYLIP (PHYLogeny Inference Package), and MrBayes (a Bayesian likelihood method).

This introductory tutorial will illustrate several of SeqLab’s multitude of features, just the ‘tip-of-the-iceberg,’ hopefully whetting your appetite enough to encourage further exploration and learning.

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

Author and Instructor: Steven M. Thompson
Introduction

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 = \( \text{sequence length} \times \text{number of sequences} \)). Mathematically this is an N-dimensional matrix, quite complex indeed. One program, MSA (Gupta, et al., 1995), does attempt to globally solve this equation using a bounding box trick, however, the algorithm's complexity precludes its use in situations with more than a few dozen sequences.

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 (Smith and Smith, 1992). However, 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 the local neighborhood of the full length of just two sequences. The pairwise, progressive solution is implemented in several programs including ClustalW (Thompson, et al., 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.

Given a particular sequence of interest, one can use any text search tool, such as GCG’s LookUp, or tools on the World Wide Web such as NCBI’s Entrez, to find that entry’s name in a sequence database. After the entry has been identified a natural next step is to use some type of a similarity search program, such as FastA and/or BLAST to help prepare a list of sequences to be aligned. One of the more difficult aspects of multiple sequence alignment regards understanding what sequences you should attempt it with. Any list from any program will need to be restricted to only those sequences that actually should be aligned. 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 estimate 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 pairwise 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, and amino acids can be similar, nitrogenous bases are either identical or not. Therefore, the signal to noise ratio is much, much better with protein sequences. 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, unless the sequences are extraordinarily similar. The resulting alignment will probably have to be extensively edited, if it works at all. Nucleotides are that much more difficult to align.

Profiles are a tremendously powerful approach for finding distantly related proteins and discovering structural motifs. Originally described by Gribskov (1987), later refinements have added more statistical rigor (see e.g. Eddy’s Hidden Markov Model Profiles [1996 and 1998]). John Devereux, former president of GCG, wrote an excellent overview essay of the method in the GCG program manual; please take the time to read this (http://www.scs.fsu.edu/gcg/profileanalysis.html) at some point. Sean Eddy’s HMMR package (Eddy, 1996 & 1998), also included in the Wisconsin Package, extends Gribskov’s profile method adding statistical robustness. All 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.

The strategy works best when one has prepared and refined a multiple sequence alignment of significantly similar sequences, or regions within sequences, and involves forming a ‘profile’ from that alignment and then searching the databases with that profile. The profile should usually be refined to only include the more conserved areas of an alignment. This refinement procedure, including repeatedly searching the databases and including or excluding members as the case may be, is known as validating the profile. 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 and generate a profile.

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). 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.

All profile methods enable 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 that may
be only distantly related and that will not be found by any other search algorithm. Even though profile searches require some work to setup and run — a meaningful multiple sequence alignment must be assembled and the search job itself takes quite a long time to run — it is well worth the bother.

We can visualize those areas of an alignment that profile searching puts the most emphasis on. They are the most conserved areas of the 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 also 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 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 in the identified regions 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, various algorithms can estimate the most reasonable evolutionary tree for that alignment. Therefore, you need to devote considerable time and energy toward developing the most satisfying 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. The Ribosomal Database Project at the Center for Microbial Ecology at Michigan State University has extensively used this process to help guide the construction of their rRNA alignments and structures (http://rdp.cme.msu.edu/). 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 this field 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 (Gilbert, 1990), allows for the back and forth conversion between several different formats. The PAUP* interfaces in the GCG system, PAUPSearch and PAUDPDisplay, 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. 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 placeholders for the sequence. Therefore, it is safest to manually edit an alignment to change leading and trailing gap symbols to ‘unknown.’ 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 that 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 relatively easy. A great way to do it 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.

**SeqLab tutorial**

I will illustrate the techniques in this exercise with a dataset containing a protein with representatives in all the branches of cellular life. I use a broad representation across all cellular life, while still keeping within the practical limits of this evening’s computer lab session. In the exercise you will use the GCG GUI SeqLab and the program PileUp to prepare and refine an alignment of this protein. You will also use the programs PlotSimilarity, Motifs, and HelicalWheel to analyze it. And finally, you will learn about some of the tools and tricks available for producing output appropriate as input to phylogenetic inference software.

The Elongation Factors are a vital protein family crucial to protein biosynthesis. They are ubiquitous to all of cellular life and, in concert with the ribosome, must have been one of the very earliest enzymatic factories to evolve. Because of strong evolutionary pressure resulting in very slow divergence and because of its ubiquity, EF-1α is an appropriate gene on which to estimate early life phylogenies and with which to ask early branching order questions in ‘deep’ eukaryotic evolution. In fact, a series of papers in the early-90’s, notably those by Iwabe, et al. (1989), Rivera and Lake (1992), and Hasegawa, et al. (1993) all base ‘universal’ trees of life on this gene. Iwabe, et al. used the trick of aligning the EF-1α gene paralogue EF-1β to their EF-1α dataset to root the tree.
Three distinct subtypes of elongation factors all work together to help perform the vital function of protein biosynthesis. In Bacteria and Eukaryota nuclear genomes they have the following names (the nomenclature in Archaea has not been completely worked out and is often contradictory):

<table>
<thead>
<tr>
<th>Eukaryota</th>
<th>Bacteria</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF-1α</td>
<td>EF-Tu</td>
<td>Binds GTP and an aminoacyl-tRNA; delivers the latter to the A site of ribosomes.</td>
</tr>
<tr>
<td>EF-1β</td>
<td>EF-Ts</td>
<td>Interacts with EF-1α/Tu to displace GDP allowing the regeneration of GTP-EF-1α/Tu</td>
</tr>
<tr>
<td>EF-2</td>
<td>EF-G</td>
<td>Binds GTP and peptidyl-tRNA and translocates the latter from the A site to the P site.</td>
</tr>
</tbody>
</table>

The Elongation Factor subunit 1-Alpha (EF-1α) in Eukaryota and most Archaea (called Elongation Factor Tu in Bacteria and plastids) has guanine nucleotide, ribosome, and aminoacyl-tRNA binding sites, and is essential to the universal process of protein biosynthesis, promoting the GTP-dependent binding of aminoacyl-tRNA to the A-site of the intact ribosome. The hydrolysis of GTP to GDP mediates a conformational change in a specific region of the molecule. This region is conserved in both EF-1α/Tu and EF-2/G and seems to be typical of GTP-dependent proteins which bind non-initiator tRNAs to the ribosome.

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

The three-dimensional structure of Elongation Factor 1α/Tu has been solved in more than fifteen cases. Partial and complete *E. coli* structures have been resolved and deposited in the Protein Data Bank (1EFM, 1ETU, 1DG1, 1EFU, and 1EFC), the complete *Thermus aquaticus* and *Thermus thermophilus* structures have been determined (1TTT, 1EFT, and 1AIP), and even cow EF-1α has had its structure determined (1D2E). Most of the structures show the protein in complex with its nucleotide ligand, some show the ternary complex. The *Thermus aquaticus* structure is shown to the right, as drawn by NCBI’s Cn3D molecular visualization tool:
Notice that half of the protein has well defined alpha helices and the rest is rather unordered coils partly defined by beta strands. GTP fits right down in amongst all the helices in the pocket. The *Thermus aquaticus* structure has six well-defined helices that occur from residue 24 through 38, 86 through 98, 114 through 126, 144 through 161, 175 through 184, and 194 through 207. There are also two short helices at residues 47 to 51 and 54 to 59. The guanine nucleotide-binding site involves residues 18 to 25, residues 81 to 85, and residues 136 to 139. Residue 8 is associated with aminoacyl-tRNA binding.

1) **Log onto your UNIX-based host account using X Windows**

I write tutorials from a ‘lowest-common-denominator’ biologist’s perspective. That is, I assume that you’re relatively inexperienced regarding computers, especially command line computing. As a consequence the tutorials are written quite explicitly, and may even seem remedial. However, if you do exactly what is written, it will work. This requires two things: 1) you must read very carefully and not skim over vital steps, and 2) don’t take offense if you already know what I’m discussing. I’m not trying to insult your intelligence. This also makes the tutorials somewhat longer than otherwise necessary. Sorry.

I use **bold type** in the 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 greater-than symbol, “>” indicates the system prompt and should not be typed as a part of commands. Pay attention — really important statements may be underlined.

The Wisconsin Package only runs on server computers running the UNIX operating system, but it can be accessed from any networked terminal. SeqLab requires X Windows graphics for its display. This can be supplied through genuine X Windowing on UNIX/Linux workstations or through X server emulation on personal computers. MS Windows machines are often set up with either XWin32 or eXceed to provide this function. Macintoshes previously used either MacX or eXodus software, but Mac OS X is UNIX so it supports true X windowing, it just doesn’t get preinstalled by default (but see Apple’s X11 or XDarwin).

Each participant in the session needs to use a different UNIX account. Use the appropriate ssh connection commands on the personal computer or workstation that you are sitting at to launch X and log onto the UNIX host computer that runs GCG at your site. Get the instructor’s assistance for this step if you are unsure of yourself. There are too many variations in method for them all to be described here. Login with your existing Mendel UNIX account and password, or the new one supplied to you today. An X-style terminal window should appear on the desktop after a few moments, if it doesn’t, launch one with the appropriate command. Refer to Appendix II for more information on the UNIX operating system and X windowing. I am also available for individualized personal help in your own laboratories, if you are having difficulties connecting to and using the GCG server from there, just contact me at stevet@bio.fsu.edu. A couple of ‘X’ tips should be mentioned at this point though. Rather than holding mouse buttons down, to activate items, just click on them; and buttons are turned on when they are pushed in and shaded.
Also, do not close windows with the X-server software’s close icon in the upper right- and/or left-hand window corner, rather, always use GCG’s “Close” or “Cancel” or “OK” button, usually at the bottom of the window.

2) Multiple sequence alignment — introducing GCG’s SeqLab

First let’s look at the list file that I have prepared for your use. Use the GCG command “fetch” to pull the file into your account from the GCG public databases:

> fetch EFla-Tu.pep.list

Check out the list file format. Following the “more” command:

> more EFla-Tu.pep.list

!!SEQUENCE_LIST 1.0
This is a list of 25 representative Elongation Factor 1 Alpha (Tu in Bacteria) protein sequences. This list spans all of cellular life and collects sequences from a broad phylogenetic spectrum available in the Swiss-Prot UniProt database. ..

UniProt:EF1A1_XENLA
UniProt:EF1A_DROME
UniProt:EF1A_HUMAN
UniProt:EF1A_ARATH
UniProt:EF1A_DICDI
UniProt:EF1A_ENTHI
UniProt:EF1A_EUGGR
UniProt:EF1A_GIALA
UniProt:EF1A_ONCVO
UniProt:EF1A_PLAFK
UniProt:EF1A_PYRWO
UniProt:EF1A_SULSO
UniProt:EF1A_TETPY
UniProt:EF1A_THEAC
UniProt:EF1A_WHEAT
UniProt:EF1A_YEAST
UniProt:EFTU_SYNP6
UniProt:EFTU_CHLTR
UniProt:EFTU_ECOLI
UniProt:EF1A_HALMA
UniProt:EF1A_METVA
UniProt:EFTU_MYCGA
 UniProt:EFTU_MYCTU
UniProt:EFTU_THEAQ
UniProt:EFTU_THEMA

Realize that had I not already prepared this file for you, you could have easily run the GCG text based search program LookUp to find your own entries and generate your own list file.

OK, now for something completely different. The SeqLab GUI, based on Steve Smith’s (1994) GDE (the Genetic Data Environment) makes running the Wisconsin Package much more intuitive by providing a common editing interface from which GCG programs can be launched. Launch GCG’s GUI by typing “seqlab &” (without the quotes). The ampersand, “&,” is not necessary but it allows you to retain control of the terminal window by running SeqLab in the
background, thereby not tying up the terminal window. This way you can switch back and forth between the terminal and SeqLab windows. After a few moments two windows will open, click in the smaller one and check “OK.” This will put you in the main SeqLab window where all analyses may be performed. All menus that I refer to from this point on in SeqLab will be within the SeqLab display, not anywhere else on the monitor — those menus talk directly to the personal computer or workstation, not the UNIX host. Before beginning the analyses, go to the “Options” menu and select “Preferences . . .” The defaults are usually good; I would just like to point out those settings that allow SeqLab to run in 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 in the future, however, it is appropriate to leave it as is for now. Be sure that the “Start SeqLab in:” choice has “Main List” selected and that “Close the window” is selected under the “After I push the ‘Run’ button:” choice. Next select the “Output Preferences.” Make sure that “Automatically display new output” is turned on. The other choices are also good at their defaults. Take a look at the “Fonts” menu next. We will leave these choices as they are, but I want to point out that if you are dealing with large alignments and/or are using a small monitor, then changing to smaller font point sizes may be desirable to allow you to see more of the alignment at once. Click “OK” to accept any changes. Next under the “Options” menu, take a look at “Graphics Devices . . .” to see your site’s standard SetPlot menu; we don’t need to do anything here so just press the “Cancel” button to close the window.” Now the SeqLab interface is ready to be utilized.

Be sure the “Mode:” “Main List” choice is selected and then go to the “File” menu. Pick “Add sequences from” and select “Sequence Files.” 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 all of the files in your working directory are displayed due to a “*” wild card filter. You may want to use some other term in the “Filter” box to restrict what is displayed, such as, in our case, “*.list.” Press the “Filter” button to display only those files that you ‘filtered.’ Select the file entitled “EF1a-Tu.pep.list” from the “Files” box and then check the “Add” and then “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. You can also directly load sequences from the online GCG databases with the “Databases . . .” choice under the “Add sequences” menu if you know their proper identifier name or accession code. Furthermore, in “Editor Mode” two additional choices are available. You can “Import” sequences from a variety of formats or from ABI style binary trace files. And you can use the “File” menu “New Sequence” choice to create empty slots to hold brand new entries, either “DNA,” “RNA,” “Protein,” or “Text,” where you can either type in data or copy and paste it from a different window.

Be sure the list file name is selected in the main window and switch to “Mode:” “Editor” to load the sequences into the SeqLab editor. The sequences come from the local GCG UniProt database (remember logical_term:ID). They’re named with official UniProt entry names (ID identifier). Notice that all of the sequences now appear in the editor window with color-coded amino acids. The nine color groups are based on a clustering of the BLOSUM62 (Henikoff
and Henikoff, 1992) matrix and approximate the physical properties of the amino acid. Expand the window full-screen. You should be able to see all of your sequences now. The display will look something like the graphic below:

![Graphic](image)

Explore the editor interface for a while. Turning off “Invert” causes the letters to assume the colors and the background to go white. Turning “Wrap” on causes the sequence to wrap vertically in the display. Use whichever combination of settings you prefer. I prefer the default non-wrapped, inverted display. GCG programs are accessible through the “Functions” menu including the powerful similarity search tools FastA and BLAST. (Do not run any similarity searches at this point though.) The scroll bar at the bottom allows you to move through the sequences linearly. You can select any sequence or position 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.

Change the “Display:” box from “Residue Coloring” to “Feature Coloring.” The colors are now based on the information from the database Feature Table for each entry. Change “Display:” to “Graphic Features,” now the features are represented using the same colors as before but in a ‘cartoon’ fashion. Use the mouse to move your cursor to one of the colored areas; quickly double-click it (or use the “Features” selection under the “Windows” menu). This will produce a new window that describes the features located at the cursor. Click on one of the features to get more information on it and to select it in its entirety. All the features are fully editable through the “Edit” check box in this panel and new features can be added with several desired shapes and colors through the “Add” check box. The display will look something like my example below:

![Graphic](image)
Close the “Sequence Features” window and return your display to “1:1.”

3) Structural analysis and annotation

While on the topic of feature annotation, let's briefly explore a protein structural analysis program available in GCG. As most of you no doubt realize, structural prediction is fraught with difficulties. However, using comparative multiple sequence approaches is by far the most reliable strategy. In my opinion, the best predictor of secondary structure around, PredictProtein, on the Web at http://www.predictprotein.org/, developed by the Protein Design Group at the European Molecular Biology Laboratory, and now based at Columbia University, uses multiple sequence alignment profile techniques along with neural net technology. A multiple sequence alignment is performed by a weighted dynamic programming method (MaxHom, Sander and Schneider, 1991) and a secondary structure prediction is produced by the profile network method (PHD). PHD is rated at an expected >72% average accuracy for the three states helix, strand, and loop (Rost and Sander, 1993 and 1994). In fact, even three-dimensional modeling without crystal coordinates is possible. This is “homology modeling.” It will often lead to remarkably accurate representations, if the similarity is great enough between your protein and one in which the structure has been solved through experimental means. Automated homology modeling is even available through the Web at the Expasy server in Switzerland (Guex Peitsch, 1997 and Guex, et al., 1999, http://www.expasy.ch/swissmod/SWISS-MODEL.html).

In the sample dataset that we are using we have a perfect example of being able to use structural inference. This is because several members of the dataset have had their three-dimensional structure solved. Among others, the *E. coli* sequence has been crystallized and, therefore, is completely annotated with secondary structure feature data inferred from its three-dimensional coordinates.

Be sure that your display is still set to “Graphic Features” and then select the entirety of one of the larger helices by double clicking the red zigzag graphic (or use the “Features” selection under the “Windows” menu), and then selecting its name, “HELIX,” in the “Sequence Features” window; close the window afterwards. Now go to the “Functions,” “Protein Analysis,” “Helical Wheel . . .” menu to launch the HelicalWheel program on that particular helix.

If asked about “Which selection” “selected sequences” or “selected region,” choose “selected region” to run the program on just the helix selected in the *E. coli* EF-Tu protein. Press “Run” in the program window and the standard “wheel” representation where you are looking down the axis of the helix will appear momentarily, as is shown below:
The most amphiphilic (sometimes known as amphipathic because of their strong tendency to be highly antigenic for T-cells) helix (i.e. the helix with the highest hydrophobic moment, in other words, the helix with the greatest partitioning of hydrophobic and hydrophilic residues from one face to the other) that I could find was the one from residue 143 through 160. What do you think?

Several other structural programs are available under the “Protein Analysis” menu, but I encourage you to proceed with the tutorial at this point. If you do run some of these other GCG “Protein Analysis” programs in the future, especially PepPlot and PeptideStructure/PlotStructure, be forewarned that they use very old and unreliable secondary structure prediction algorithms. They are NOT to be trusted as secondary structure predictors. However, they do plot several other very worthwhile and reliable attributes such as hydrophobicity and hydrophobic moments.

One of the more important things to realize is many of these types of algorithms are based on soluble, globular proteins; therefore, when dealing with other types of proteins you should alter parameters and interpret the results in this light. Using the same parameters with all types of proteins is not appropriate; you must tailor them appropriately. Since the defaults are often based on soluble type guidelines, you need to be especially careful when working with membrane-associated or membrane-spanning proteins. The simplest parameter to change is often the window size. It should be set approximately to the size of the feature being analyzed (e.g., use a window size of about 21 when trying to find membrane-spanning alpha helices).

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. In many cases this can be a tremendous aid in ascertaining the function of an unknown peptide sequence. This database of catalogued consensus patterns is called PROSITE. The GCG program Motifs performs a search against this database. Motifs searches for recognized structural, regulatory and enzymatic consensus sequences in the PROSITE Database of protein families and domains (Bairoch, 1992). The program can tolerate mismatches with a mismatch option and it displays an abstract with selected references for each motif signature found.

Select the E. coli sequence entry name and then go back to the “Functions,” “Protein Analysis” menu. Pick “Motifs . . .” there to launch the program. Most likely a “Which selection” window will pop up asking if you want to use the “selected sequences” or “selected region;” choose “selected sequences” to run the program on the full length of the E. coli EF-Tu protein. The “Motifs” program window will then display; press “Run” to screen the sequence without any options. Carefully look over the text file that is displayed. Notice the sites that have been characterized in this sequence and the extensive bibliography associated with them:

```
MOTIFS from: /users/thompson/.seqlab-mendel/input_20.rsf{EFTU_ECOLI}
Mismatches: 0     May 3, 2006 20:18 ..
Atp_Gtp_A
   (A,G)x4GK(S,T)
      (G)x(4)GK(T)
   18: NVGTI    GHVDHGKT    TLTAA
*****************************************
```
From sequence comparisons and crystallographic data analysis it has been shown [1,2,3,4,5,6] that an appreciable proportion of proteins that bind ATP or GTP share a number of more or less conserved sequence motifs. The best conserved of these motifs is a glycine-rich region, which typically forms a flexible loop between a beta-strand and an alpha-helix. This loop interacts with one of the phosphate groups of the nucleotide. This sequence motif is generally referred to as the ‘A’ consensus sequence [1] or the ‘P-loop’ [5].

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

- ATP synthase alpha and beta subunits (see <PDOC00137>).
- Myosin heavy chains.
- Kinesin heavy chains and kinesin-like proteins (see <PDOC00343>).
- Dynamins and dynamin-like proteins (see <PDOC00362>).
- Guanylate kinase (see <PDOC00670>).
- Thymidylate kinase (see <PDOC00524>).
- Thymidylate kinase.
- Shikimate kinase (see <PDOC00868>).
- Nitrogenase iron protein family (nifH/frxC) (see <PDOC00580>).
- ATP-binding proteins involved in 'active transport' (ABC transporters) [7] (see <PDOC00185>).
- DNA and RNA helicases [8,9,10].
- GTP-binding elongation factors (EF-Tu, EF-1alpha, EF-G, EF-2, etc.).
- Ras family of GTP-binding proteins (Ras, Rho, Rab, Ral, Ypt1, SEC4, etc.).
- Nuclear protein ran (see <PDOC00859>).
- ADP-ribosylation factors family (see <PDOC00781>).
- Bacterial dnaA protein (see <PDOC00771>).
- Bacterial recA protein (see <PDOC00131>).
- Bacterial recF protein (see <PDOC00539>).
- Guanine nucleotide-binding proteins alpha subunits (Gi, Gs, Gt, G0, etc.).
- DNA mismatch repair proteins mutS family (See <PDOC00388>).
- Bacterial type II secretion system protein E (see <PDOC00567>).

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

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

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

-Last update: November 1997 / Text revised.

  EMBO J. 1:945-951(1982).
Elongation factors [1,2] are proteins catalyzing the elongation of peptide chains in protein biosynthesis. In both prokaryotes and eukaryotes, there are three distinct types of elongation factors, as described in the following table:

<table>
<thead>
<tr>
<th>Eukaryotes</th>
<th>Prokaryotes</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF-lalpha</td>
<td>EF-Tu</td>
<td>Binds GTP and an aminoacyl-tRNA; delivers the latter to the A site of ribosomes.</td>
</tr>
<tr>
<td>EF-lbeta</td>
<td>EF-Ts</td>
<td>Interacts with EF-1a/EF-Tu to displace GDP and thus allows the regeneration of GTP-EF-1a.</td>
</tr>
<tr>
<td>EF-2</td>
<td>EF-G</td>
<td>Binds GTP and peptidyl-tRNA and translocates the latter from the A site to the P site.</td>
</tr>
</tbody>
</table>

The GTP-binding elongation factor family also includes the following proteins:
- Eukaryotic peptide chain release factor GTP-binding subunits [3]. These proteins interact with release factors that bind to ribosomes that have encountered a stop codon at their decoding site and help them to induce release of the nascent polypeptide. The yeast protein was known as SUP2 (and also as SUP35, SUF12 or GST1) and the human homolog as GST1-Hs.
- Prokaryotic peptide chain release factor 3 (RF-3) (gene prfC). RF-3 is a class-II RF, a GTP-binding protein that interacts with class I RFs (see <PDO00607>) and enhance their activity [4].
- Prokaryotic GTP-binding protein lepA and its homolog in yeast (gene GUF1) and in Caenorhabditis elegans (ZK1236.1).
- Yeast HBS1 [5].
- Rat statin S1 [6], a protein of unknown function which is highly similar to EF-lalpha.
- Prokaryotic selenocysteine-specific elongation factor selB [7], which seems
to replace EF-Tu for the insertion of selenocysteine directed by the UGA codon.
- The tetracycline resistance proteins tetM/tetO [8,9] from various bacteria such as Campylobacter jejuni, Enterococcus faecalis, Streptococcus mutans and Ureaplasma urealyticum. Tetracycline binds to the prokaryotic ribosomal 30S subunit and inhibits binding of aminoacyl-tRNAs. These proteins abolish the inhibitory effect of tetracycline on protein synthesis.
- Rhizobium nodulation protein nodQ [10].

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

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


4) Performing the alignment — the PileUp program

Click the “Close” box and return your display to “1:1” and “Residue Coloring.” Take a look at each of the members of the list. Quickly double click on various entries’ names to see the database reference descriptions for them (or click
on the “INFO” button). Next select all of the entries in the list through the “Edit” menu “Select All” command (or by dragging the mouse through all the entries or by shift-clicking the bottom and top entry [select non adjacent entries with Ctrl-right-clicks]). Once all of your sequences are selected, go to the “Functions” menu and select “Multiple comparison.” Click on “PileUp . . .” to align the entries. A new window will 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 program will first compare every sequence with every other one. This is the pairwise nature of the program, and then it will progressively merge them into an alignment in the order of determined similarity, from most to least. 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 BLOSUM62 matrix and gap introduction and extension penalties used by default. In most cases these work just fine though they can be changed if desired (the BLOSUM30 matrix can be very helpful for aligning quite divergent sequences). Scroll through your alignment to check it out and then “Close” the window afterwards. An abridged output file from my example follows below:

```
!!AA_MULTIPLE_ALIGNMENT 1.0
PileUp of: @/users/thompson/.seqlab-mendel/pileup_1.list

Symbol comparison table: GenRunData:blosum62.cmp  CompCheck: 6430
GapWeight: 12
GapLengthWeight: 4
pileup_1.msf  MSF: 483  Type: P  April 29, 1999 17:22  Check: 9074 ..
Name: eftu_anani  Len:  483  Check: 7317  Weight:  1.00
Name: eftu_theaq  Len:  483  Check: 2028  Weight:  1.00
Name: eftu_ecoli  Len:  483  Check: 8483  Weight:  1.00
Name: eftu_myctu  Len:  483  Check: 7189  Weight:  1.00
Name: eftu_myctga  Len:  483  Check: 9514  Weight:  1.00
Name: eftu_thema  Len:  483  Check:  426  Weight:  1.00
Name: eftu_chlitr  Len:  483  Check:  545  Weight:  1.00
Name: eftu_xenla  Len:  483  Check: 7491  Weight:  1.00
Name: eftu_humna  Len:  483  Check: 7963  Weight:  1.00
Name: eftu_drome  Len:  483  Check: 3601  Weight:  1.00
Name: efl1_oncvo  Len:  483  Check: 9453  Weight:  1.00
Name: efl1_yeast  Len:  483  Check: 3241  Weight:  1.00
Name: efl1_arath  Len:  483  Check: 4009  Weight:  1.00
Name: efl1_wheat  Len:  483  Check: 5710  Weight:  1.00
Name: efl1_euggr  Len:  483  Check: 2159  Weight:  1.00
Name: efl1_enthi  Len:  483  Check: 3526  Weight:  1.00
Name: efl1_tetpy  Len:  483  Check: 9876  Weight:  1.00
Name: efl1_dicdi  Len:  483  Check: 1495  Weight:  1.00
Name: efl1_plafk  Len:  483  Check: 2043  Weight:  1.00
Name: efl1_giala  Len:  483  Check: 6078  Weight:  1.00
Name: efl1_theac  Len:  483  Check: 9544  Weight:  1.00
Name: efl1_metva  Len:  483  Check: 4336  Weight:  1.00
Name: efl1_halma  Len:  483  Check: 3230  Weight:  1.00
Name: efl1_pyrwo  Len:  483  Check: 7726  Weight:  1.00
Name: efl1_sulso  Len:  483  Check: 2391  Weight:  1.00

//
1  50
eftu_anani  MARAKFERTK PHANIGTIGH VDHGKTTLTA AITTVLAKAG .MA.KARAYA
eftu_theaq  ~AGEFIRTK PHVNGTIGH VDHGKTTLTA ALTYVAAAIN PHV.EVKDYG
eftu_ecoli  ~SEKFERRTK PHVNGTIGH VDHGKTTLTA AITTVLAKTY G..GAARAFD
eftu_myctu  MARAKFQRTK PHVNGTIGH VDHGKTTLTA AITKVLHDFK PDNGETKAFD
```
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. Or the sequence could be “CUT” from the alignment with the SeqLab Editor. Another important number on the individual checksum lines should be pointed out. The “Weight” designation determines how much importance each sequence contributes to a standard Gribskov-style profile made from the alignment. Sometimes it is worthwhile to adjust these values so that the contribution of a collection of very similar sequences does not overwhelm the signal from a few more divergent sequences. In the SeqLab interface the “Sequence Info . . .” window can be used to accomplish this. 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 is a very important window and will contain all of the output from your current SeqLab session. Files may be displayed, printed, saved in other locations or with other names, and deleted from this window. We need to use an extremely important function at this point. Press the “Add to Editor” button and specify “Overwrite old with new” in the next window when prompted, to take your MSF output and merge it with the RSF (Rich Sequence Format: the sequence data as well as all the reference and feature information) file in the open editor. This will keep all feature information intact, inserting gaps as needed, renumbering all reference locations. “Close” the “Output Manager” after loading your alignment.

The next window shows PileUp’s sequence similarity dendrogram. 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. My example is illustrated here on the right:
This is not an evolutionary or phylogenetic tree and it should never be presented as one. (Although, if the rates of evolution for each lineage were exactly the same, which is seldom the case in nature, it could be the same as one.) No evolutionary models for multiple hits or methods for correction of unequal rates of substitution or divergence are used in its construction. It merely indicates the relative similarity of the sequences. However, the dendrogram can assist in determining sequence weighting factors to even out each sequences’ contribution to a profile.

If desired, you can directly print from this window to a PostScript file by picking “Print . . .” with the “Output Device:” “[Encapsulated] PostScript File.” You can rename the output file to anything you may want in this window; 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 direct access to the UNIX system printer and it is PostScript compatible. (All Macintosh compatible laser printers run PostScript by default. Carefully check any laser printer connected to a Wintel system to be sure that it is PostScript compatible.) “Close” the dendrogram window to return to the editor. Now notice that your residues align by color. My editor display looks like the graphic below after loading the MSF file:

5) Visualizing conservation in multiple sequence alignments

The most conserved portions of an alignment are those most resistant to evolutionary change, often due to some type of structural constraint. The GCG graphics program PlotSimilarity allows us to easily visualize these conserved regions in a multiple sequence alignment. This is also a very nice way to see those areas of an alignment that may need improving by pointing out the most variable regions. The 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 all of the sequence names are selected and then go back to the “Functions” menu and under the “Multiple comparison” section choose “PlotSimilarity . . .” We need to change some of the program defaults 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. The second specification launches the program’s command line -Expand option that ‘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 was used to create your alignment or you can specify an alternative. The default matrix, BLOSUM62, begins its identity value at 4 and ranges up to 11; mismatches go as low as -4. “Close” the 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 on the left:

Regardless of whether you print this plot or not, take notes of where the similarity significantly falls off within and at the beginning and end of the alignment. In my example above, this is a region around 220, around 300, and about the last 25 residues or so. Now go to the “File” menu and click on “Open Color Mask Files.” This will produce another window from which you should select your new “plotsimilarity.cmask” file; click on “Add” and “Close” the window. This will produce a gray scale overlay on your sequences that describes their regional similarity where darker gray corresponds to higher similarity values. My sample alignment, using a zoom factor of 8 to 1, looks like the following. Notice the strong conservation peak centered around residue 100 in the alignment, one of EF-1α/Tu’s GTP binding regions. The graphic from my SeqLab display with its superimposed color mask follows below:

Make a PostScript file of this plot too, if desired. You can directly print from this window to a PostScript file by picking “Print . . .” Just as before, be sure that the “Output Device:” chosen is “[Encapsulated] PostScript File.” You can name the output file anything that you may want in this window; click “Proceed” to create the EPSF output file in your current directory and then “Close” the window.
6) 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 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 “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). Once all of your sequences and the region that you wish to improve are selected, go to the “Functions” menu and again select “Multiple comparison.” Click on “PileUp . . .” to realign all of the sequences within that region. (The “Windows” menu also contains a ‘shortcut’ listing of all of the programs that you have used in the current session; you can launch any of them from there as well.) You will be asked whether you want to use the “Selected sequences” or “Selected region.” It is very important to specify “Selected region.” This will produce a new window with the parameters for running PileUp. Next, be sure to click on “Options . . .” to change the way that PileUp will perform the alignment. In the “Options” window check the gap creation and extension boxes and change their respective values to much less than the default. Changing them to 3 and 1 respectively works well for me in this step. Most importantly, check “Realign a portion of an existing alignment;” this calls up the command line -InSitu option. Otherwise only that portion of your alignment selected will be retained in the output. Furthermore, we really don’t need another similarity dendrogram, so uncheck the “Plot dendrogram” box. “Close” the window and notice the new options in the PileUp “Command Line:” “Run” the program to improve your alignment. The window will go away and your results will return very quickly since you are only realigning a portion of the alignment; new output windows will automatically display. The top window will be the MSF output from your PileUp run. Notice the BLOSUM62 matrix used by default (others can be specified in the options menu) and the lowered gap introduction and extension penalties of 3 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, 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 your alignment efforts in subsequent steps. “Close” the “Output Manager” window after loading your new alignment.

Your alignment should now be a bit better within the specified region. Repeat this process in all areas of low similarity, again, working from the carboxy termini toward the amino end. Notice that all of the options that you last specified are retained by the program so you don’t need to respecify them. You can also save run parameters so that they will come up in subsequent sessions by clicking on the “Save Settings” box in any of the program run windows. You may want to go to the “File” menu periodically to save your work using the “Save as . . .” function in case of a computer or network problem. It’s also probably a good idea to reperform the PlotSimilarity and color mask procedure after going through the entire alignment to see how things have improved after you’ve finished the various InSitu PileUps. If you discover an area that you can not improve through this automated procedure, then it is time to either manually ‘correct’ it or ‘throw it away.’ Again, note those ‘problem’ areas and then switch back to “Residue Coloring.” This will ease manual alignment by allowing your eyes to work with columns of color.
Other things that can help manual alignment are “GROUP”ing and “Protections.” The “GROUP” function allows you to manipulate ‘families’ of sequences as a whole — any change in one will be propagated throughout them all. To “GROUP” sequences, select those that you want to behave collectively and then click on the “GROUP” icon right above your alignment. You can have as many groups as you want. The space bar will introduce a gap into the sequence and the delete key will take a gap away. However, you can not delete a sequence residue without changing that sequence’s (or the entire alignment’s) “Protections.” Click on the padlock icon to produce a “Protections” window. Notice that the default protection allows you to modify “Gap Characters” and “Reversals” only. Check “All other characters” to allow you to “Cut” regions out of your alignment and/or delete individual residues and then click “OK” to close the window. A very powerful manual alignment function can be thought of as the ‘abacus’ function. To take advantage of this function select the region that you want to slide and then press the shift key as you move the region with the right or left arrow key. You can slide residues greater distances by prefacesing the command keystrokes with the number of spaces that you want them to slide.

Make subjective decisions regarding your alignment. Is it good enough; do things line up the way that they should? If, after all else, you decide that you just can’t align some region, or even an entire sequence, then perhaps get rid of it with the “Cut” function. Another alternative is the mask function that I will describe later on. 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 automatically 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 now align perfectly. This is another way to assure that your alignment is as biologically ‘correct’ as possible. Everything you do from this point on, and especially later if you use alignments to ascertain molecular 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. Furthermore, many people prefer to run phylogenetic analyses on DNA rather than protein regardless of how similar they are — the multiple substitution models are much more robust for DNA. The logic to this paired protein and DNA alignment approach follows:

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

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

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, reformating 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 appropriate Reformat option (either -MSF or -RSF) and you must specify all the sequences within the file, i.e. “{*},” for example:

    > reformat -msf your_favorite.msf{*}

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

7) 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 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’ds plurality level (“Percent required for majority”) with a threshold of 5 (“Minimum score that represents a match”); however, these are a very stringent 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. (If making a normal sequence consensus rather than a weight mask, you can generate a gray intermediate similarity color as well as the black and white representation. This is a nice way to prepare alignment figures for publication.) The screenshot below illustrates my example using the BLOSUM62 matrix, a plurality of 15%, and a threshold cutoff value of 3:

Few areas are excluded by the Mask in this alignment because of the large similarity of this group of sequences. This is as it should be for excluding many more columns in this particular alignment would likely just leave all identical sequences and it would be impossible to ascertain how they are related. In fact, as described above, when dealing with sequences this similar, it may be best to align the DNA sequences along with their corresponding proteins and then perform the phylogenetic analyses on the DNA rather than on the proteins. 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.

Once a Mask has been created in SeqLab most of the programs available through the “Functions” menu will use that Mask, if the Mask is selected along with the desired sequences, to weight the columns of the alignment data matrix appropriately. This only occurs through the “Functions” menu.
When you've found a combination that you like, you may wish to go to the “File” “Print . . .” command in order to prepare a PostScript file of your SeqLab display (change “Output Format:” to “PostScript”). You can 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 you would specify “Destination . . . File,” give it an appropriate filename, and then click “OK.” This will result in a PostScript file of the alignment using the displayed coloring and the specified parameters to be created in the directory where you launched SeqLab. This file can then be transferred to another machine for color PostScript printing, or for importing into PostScript savvy programs for further manipulation, or it 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 some 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 several different formats are available for saving a copy of your RSF file. However, do not export any of these formats at this point, and “Cancel” the window. 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 subsequent phylogenetic analyses, we'll use another method to export our alignment. Therefore, after being sure that all of the protein sequences as well as the Mask sequence are selected, go to the “Functions” menu, where choices will be affected by the Mask, and choose “Importing/Exporting” “SeqConv+ . . .” “Set the output format to: FastA” and press “Run” to convert those portions of the alignment that are not masked out into FastA format. FastA is a great intermediate format on our way to PHYLIP’s required format because it is so simple. However, the new file is not automatically displayed by SeqLab and is not listed in the Output Manager. The file will appear in your working directory with the name “seqconv+.fa.” Use your terminal window to look at it. The very first part of my FastA format output file is shown below.

```
> EFTU_ECOLI In situ PileUp of: @/export/home/gcg/stevet/.seqlab-snap/pileup_16.list
SKEKFERTKPHVNVGTIGHVDHGKTTLT-------AATIT---TV–LA–KTYGGA–ARA
F–DQIDNAPIEEKAGRTINTSHVEYDTPTRHYAHVDCPQHADVKMNITGAAQMDGAILV
VAATDG––MP––QTREHILLGRQGVYIVFLNKnCMOV–D–DEELLELVEHEV
RELSQYDFPGDD----TPIV–RGSAKALE----GDAEWEA–KIE-----LAGFLDS
Y–IPEPERAIDKPFFLPEDVFSISGRTVVTGVRGRIIKGEEVEI–VGITQKST–CTGVEMFKLLEGRAGENVVLLRGIKREEIERSQVL----ARPGTIKPHTKFSEQVYL

> EFTU_MYCTU In situ PileUp of: @/export/home/gcg/stevet/.seqlab-snap/pileup_16.list
MAKAKFQRTKPHVNVGTIGHVDHGKTTLT-------AATIT----KV–LHDKFPDLNE–TK
AF–DQIDNAPIEKQGRITINIHAHEYQTDRKHYAHVDAPGHDYIKMNITGAAQMDGAIL
VVAATDG––MP––QTREHILLARRQGVYPILVALNKADAV–D–DEELLELVEHE
```

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. Use your terminal window to save the file under a name that makes sense to you, such as “EFla-Tu.fa.” We’ll next use ReadSeq to convert this FastA format file to PHYLIP compatible format.

Stay in 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 though. 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 current PHYLIP format and then designate the input sequence file name. 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):
EFla-Tu.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:
EFla-Tu.tfa
Sequences in EFla-Tu.tfa (format is 8. Pearson/Fasta)
  1) EFTU_ECOLI_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
  2) EFTU_MYCTU_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
  3) EFTU_THEAQ_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
  4) EFTU_ANANI_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
  5) EFTU_THEMA_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
  6) EFTU_CHLTR_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
  7) EFTU_MYCGA_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
  8) EFTU_MYCGA_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
  9) EFTU_EUGGR_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
 10) EFTU_TETPY_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
 11) EFTU_XENLA_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
 12) EFTU_HUMAN_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
 13) EFTU_DROME_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
 14) EFTU_ONCVO_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
 15) EFTU_ENTHI_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
 16) EFTU_PLAFK_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
 17) EFTU_TETPY_1 PileUp of: @/mole98/thompson/.seqlab-mole1/pileup_1.list
18) EF1A_YEAST_1 PileUp of: @/mole98/thompson/.seqlab/mole1/pileup_1.list
19) EF1A_DICDI_1 PileUp of: @/mole98/thompson/.seqlab/mole1/pileup_1.list
20) EF1A_GIALA_1 PileUp of: @/mole98/thompson/.seqlab/mole1/pileup_1.list
21) EF1A_PYRWO_1 PileUp of: @/mole98/thompson/.seqlab/mole1/pileup_1.list
22) EF1A_THEAC_1 PileUp of: @/mole98/thompson/.seqlab/mole1/pileup_1.list
23) EF1A_SULSO_1 PileUp of: @/mole98/thompson/.seqlab/mole1/pileup_1.list

Choose a sequence (# or All):
all
This format requires equal length sequences.
Sequence truncated or padded to fit.

Never mind if you get the "... padded to fit" error message — 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 before, periods and tildes will not work to represent indels (gaps); they must all be changed to hyphens (dashes). The following, rather strange, UNIX command works very well for this step from the command line, but you should not need to use it in this exercise:

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

Run "more" on your new file to see what PHYLIP format looks like. The top part is shown here:

```
> more EFla-Tu.phy

25 483
EFTU_ECOLI SKEKFRTKP HVNVGTIGHV DHGKTTLT-- -----AAIT-- --TV--LA-- K
EFTU_MYCTU MAKAFQRTK PHVNGTIGHV IDHGKTTLT-- -----AAIT-- --KV--LHD--
EFTU_THEAQ AKGFRTKPHVNVGTIGHVDHGKTTLT-- -----AAIT-- --YV--AAA--
EFTU_THEA MAKERFDRSK PHVNGTIGHV IDHGKTTLT-- -----AAIC-- --TV--LS--
EFTU_ANANIK MARAFKRTK PHVNGTIGHV IDHGKTTLT-- -----AAIT-- --KY--LML--
EFTU_CHLIR SKETFQNRK HHNGTIGHV DHGKTTLT-- -----AAIT-- --RA--LS--
EF10_XENLA M-------GKEK THINIVVIGH VDSGKSTTTG HLIYKCGGID --KRTIEKFK
EF11_HUMAN M-------GKEK THINIVVIGH VDSGKSTTTG HLIYKCGGID --KRTIEKFK
EF11_DROME M-------GKEK THINIVVIGH VDSGKSTTTG HLIYKCGGID --KRTIEKFK
EF1A_ONCVO M-------GKEK THINIVVIGH VDSGKSTTTG HLIYKCGGID --KRTIEKFK
EF1A_YEAST M-------GKEK SHINIVVIGH VDSGKSTTTG HLIYKCGGID --KRTIEKFK
EF1A_ARATH M-------GKEK FHINIVVIGH VDSGKSTTTG HLIYKLGID --KVIEFKE
EF1A_WHEAT M-------GKEK THINIVVIGH VDSGKSTTTG HLIYKLGID --KRVIEFKE
EF1A_EUGGR M-------GKEK VHISLVVIGH VDSGKSTTTG HLIYKCGGID --QRSIEFKE
EF1A_ENTHI M-------PKEK THINIVVIGH VDSGKSTTTG HLIYKCGGID --QRTIEFKE
EF1A_TETPY M-------ARGDK VHINLVVIGH VDSGKSTTTG HLIYKCGGID --KRVIEFKE
EF1A_DICDI M-------EPHESEK THINIVVIGH VDSGKSTTTG HLIYKCGGID --KRVIEFKE
EF1A_PLAFF M-------GKEK THINIVVIGH VDSGKSTTTG HIIYKLGID --RRTIEFKE
EF1A_GIALA ------- ------- ------- ------- STLTGH LIYKCGGID --QRTIEFKE
EF1A_HALMA -------SDEQ HQNLAIGHV DHGKSTLVGR LLYETGSV-P EHVIEQHKE
EF1A_METVA M-------AKTB PILNVAFIGH VDHGKSTTVG RLLDDGAID PQLIV--LRLK
EF1A_THEAC M-------ASQK PHNLITIGH VDHGKSTLVG RLYEHGEI-- PAHIEEYRK
EF1A_PYRWO M-------MPDKD PVHNIVIGH VDHGKSTTTG RLLYDTN-- PEIIKPF--E
EF1A_SULSO M-------SQK PHNLIVIIGH VDHGKSTLVG RLLMDRGFID EK--TVKEAE

TYGGA--ARA F-DQIDNA PEKARGITINT SHVEYDTFTPR HYAIVDCP
```
Notice that the file begins with two numbers; the first shows the number of sequences in the matrix and the second lists the length of the matrix including any gaps and ambiguities. The next section lists the names of the sequences truncated to ten characters, if necessary, along with all the sequences printed in an ‘interleaved’ fashion. Only the first sequence block lists the names, all others just give sequence data.

Return to your SeqLab display to generate a NEXUS file for PAUP* (Swofford, 1989–2006). NEXUS format files are easily and reliably built from GCG alignments with one of GCG’s interface programs to the PAUP* package, PAUPSearch. This program can provide an easy to use access to a subset of PAUP*. However, for serious phylogenetic analysis you’ll probably want to run PAUP* exterior to GCG in native mode, directly at the command line, to take advantage of the complete package’s substantial power. You can get the latest version, 4.0b10, directly from Sinauer Associates, the publishing company that distributes the software, and install it on your system. Sinauer, can be found on the Web at http://www.sinauer.com/, or by e-mail at publish@sinauer.com. Regardless, PAUPSearch is a very handy tool for generating NEXUS format files for PAUP*. The procedure will be shown next.

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. We merely want to generate a NEXUS file, so we will run PAUPSearch in its fastest mode without actually performing a search. The default settings for “Tree Optimality Criterion” “maximum parsimony,” “heuristic tree search (fast)” “Method for Obtaining Best Tree(s),” and do not “perform bootstrap replications....” don’t really matter, since we’re not going to be doing a search anyway. However, the “Options” menu is very important — press the button. In the “PaupSearch Options” menu check in the top box to save the PAUPscript file. The automatic conversion of your alignment to NEXUS format creates this PAUPScript. It contains your alignment in NEXUS format as well as all the default GCG PAUP commands. 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 through the options menu, leaving the rest of the options at their default settings, but do check them out. “Close” the options menu. Be sure that “How:” “Background Job” is specified on the main PAUPSearch menu and then press “Run.” The results will be quickly displayed.

The PAUPscript output file is very important. 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 PAUP*, in its native mode on your server, or by ‘secure copying’ it to some other appropriate machine. Using a pre-OS X capable Macintosh may be desirable in order to take advantage of PAUP**’s very friendly Classic Macintosh graphical user interface. Since GCG automatically creates this file when you run PAUPSearch, correctly encoding all of the required format data, 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 want to comment out or remove 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.

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 your terminal window where you can log off Mendel.

8) Conclusion.

Obviously I have only touched the ‘tip of the iceberg’ regarding SeqLab’s full potential. Please refer to the online GCG documentation available through the command “GenHelp,” through the SeqLab Help menus, or at http://www.scs.fsu.edu/gcg/ to fully explore its many possibilities. It is an incredibly powerful way to run the Wisconsin Sequence Analysis Package.

Gunnar von Heijne in his quite readable, but by now somewhat dated, 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.”

References


Genetics Computer Group (GCG), a part of Accelrys Inc., San Diego, California, U.S.A..


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


Appendix I

The Genetics Computer Group — the Accelrys Wisconsin Package for Sequence Analysis

GCG began in 1982 on a VAX/VMS computer in Oliver Smithies’ Genetics Department laboratory at the University of Wisconsin, Madison; and then starting in 1990 it became a private company; which was acquired by the Oxford Molecular Group, United Kingdom, in 1997; and then by Pharmacopeia Inc., United States, in 2000; and then in 2004 Accelrys Inc. left Pharmacopeia to become an independent entity. The suite is now exclusively a UNIX package and contains around a 150 programs designed to work in a "toolbox" fashion. That is, several simple programs used in succession can lead to sophisticated results. Most importantly, the package has 'internal compatibility,' i.e. once you learn to use one program, all programs can be run similarly, and, the output from many programs can be used as input for other programs. It is used all over the world, for more than 20 years, by more than 30,000 scientists at over 950 institutions worldwide, so learning it here will most likely be useful at any of several places you may end up at.

1) Specifying sequences GCG style and logical terms!

To answer the question, “What sequence(s)?” In order of increasing power and complexity, four methods:

a) The sequence is in a local GCG single sequence format file (SSF) in your UNIX account. This sequence file can be anywhere in your account as long as you supply an appropriate ‘path’ so that the program can find the file. The sequence file can have any name but it is best to use extensions that tell you what type of molecule it is, e.g. .seq and .pep (my.pep or ~user/subdir/my.seg). Use the program ‘reformat' to convert ‘raw' text format files to GCG format.

This is a small example of ‘raw' GCG single sequence format.

```
Always put some documentation on top, so in the future you can figure out what you're dealing with! Two periods separate that documentation from the actual data.

.....

ACTGACGTCACATACTGGGACTGAGTTACCGAGTTATACAGATTTAATAGCATGCGATCCCATGGGA
```

Next the clean GCG format single sequence file after the ‘reformat' command:

```
!!NA_SEQUENCE 1.0
Always put some documentation on top, so in the future you can figure out what you're dealing with! Two periods separate that documentation from the actual data.

example.seq  Length: 77  July 21, 1999 09:30  Type: N  Check: 4099  ..

  1  ACTGACGTCACATACTGGGACTGAGTTACCGAGTTATACAGATTTAATAGCATGCGATCCCATGGGA
  51  GATTTAATAGCATGCGATCCCATGGGA
```
b) The sequence is in a local GCG database in which case you ‘point’ to it by using a GCG database logical name. These names make sense, and are either the name of the database or an abbreviation thereof. Subcategory logical names can be used for nucleotide databases, such as rodent. A colon, “:”, always sets the logical name apart from either an accession code or a proper identifier name or a wildcard expression, and they are case insensitive. Some examples follow: GenBank:EctufBT, gb:x57091, UniProt:EFTu_Ecoli, and sw:p02990 all refer to elongation factor Tu sequences in *Escherichia coli*, the first two nucleotide, the last two protein. If the database uses consistent naming conventions, then you can use a wild card to specify all of a particular type of sequence. This works great in SwissProt; e.g. SW:EFTu_* specifies all of the EFTu sequences in SwissProt. Because sequences are in local GCG databases, it is unnecessary to put individual sequences in your account; that only takes up disk space.

**Logical terms for the GCG Wisconsin Package** *(terms may vary at different installations)*

**Sequence databases, nucleic acids:**

<table>
<thead>
<tr>
<th>Logical name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENBANKPLUS:*</td>
<td>all of GenBank plus EST, HTC, and GSS</td>
</tr>
<tr>
<td>GBP:*</td>
<td>all of GenBank plus EST, HTC, and GSS</td>
</tr>
<tr>
<td>GENBANK:*</td>
<td>all of GenBank except EST, HTC, and GSS</td>
</tr>
<tr>
<td>GB:*</td>
<td>all of GenBank except EST, HTC, and GSS</td>
</tr>
<tr>
<td>BACTERIAL:*</td>
<td>GenBank bacteria and archaea</td>
</tr>
<tr>
<td>BA:*</td>
<td>GenBank bacteria and archaea</td>
</tr>
<tr>
<td>INVERTEBRATE:*</td>
<td>GenBank invertebrate</td>
</tr>
<tr>
<td>IN:*</td>
<td>GenBank invertebrate</td>
</tr>
<tr>
<td>OTHERMAMMAL:*</td>
<td>GenBank other mammal</td>
</tr>
<tr>
<td>OM:*</td>
<td>GenBank other mammal</td>
</tr>
<tr>
<td>OTHERVERTEBRATE:*</td>
<td>GenBank other vertebrate</td>
</tr>
<tr>
<td>OV:*</td>
<td>GenBank other vertebrate</td>
</tr>
<tr>
<td>PHAGE:*</td>
<td>GenBank phage</td>
</tr>
<tr>
<td>PH:*</td>
<td>GenBank phage</td>
</tr>
<tr>
<td>PLANT:*</td>
<td>GenBank plant and fungi</td>
</tr>
<tr>
<td>PL:*</td>
<td>GenBank plant and fungi</td>
</tr>
<tr>
<td>PRIMATE:*</td>
<td>GenBank primate</td>
</tr>
<tr>
<td>PR:*</td>
<td>GenBank primate</td>
</tr>
<tr>
<td>RODENT:*</td>
<td>GenBank rodent</td>
</tr>
<tr>
<td>RO:*</td>
<td>GenBank rodent</td>
</tr>
<tr>
<td>VIRAL:*</td>
<td>GenBank viral</td>
</tr>
<tr>
<td>VI:*</td>
<td>GenBank viral</td>
</tr>
<tr>
<td>TAGS:*</td>
<td>GenBank EST, HTC, and GSS</td>
</tr>
<tr>
<td>EST:*</td>
<td>GenBank EST Expressed Sequence Tags</td>
</tr>
<tr>
<td>GSS:*</td>
<td>GenBank Genome Survey Sequences</td>
</tr>
<tr>
<td>HTC:*</td>
<td>GenBank High Throughput cDNA</td>
</tr>
<tr>
<td>HTG:*</td>
<td>GenBank High Throughput Genomic</td>
</tr>
<tr>
<td>PATENT:*</td>
<td>GenBank patent</td>
</tr>
<tr>
<td>PAT:*</td>
<td>GenBank patent</td>
</tr>
<tr>
<td>STS:*</td>
<td>GenBank Sequence Tagged Sites</td>
</tr>
<tr>
<td>SYNTHETIC:*</td>
<td>GenBank synthetic</td>
</tr>
<tr>
<td>SY:*</td>
<td>GenBank synthetic</td>
</tr>
<tr>
<td>UNANNOTATED:*</td>
<td>GenBank unannotated</td>
</tr>
<tr>
<td>UN:*</td>
<td>GenBank unannotated</td>
</tr>
<tr>
<td>REFSEQNUC:*</td>
<td>NCBI RefSeq transcriptomes</td>
</tr>
<tr>
<td>RS_RNA:*</td>
<td>NCBI RefSeq transcriptomes</td>
</tr>
</tbody>
</table>
| Genome sequence databases, nucleic acids
| HOMO:*             | NCBI human RefSeq working draft                  |
| PAN:*              | NCBI chimpanzee RefSeq working draft             |
| DANIO:*            | Sanger Zebrafish assembly                        |
| CELEGANS:*         | NCBI nematidode RefSeq assembly                  |
| Sequence databases, amino acids:
| UNIPROT:*          | all of Swiss-Prot and all of SPTREMBL            |
| UNI:*              | all of Swiss-Prot and all of SPTREMBL            |
| SWISSPROTPLUS:*    | all of Swiss-Prot and all of SPTREMBL            |
| SWP:*              | all of Swiss-Prot and all of SPTREMBL            |
| SWISSPROT:*        | all of Swiss-Prot (fully annotated)              |
| SWISS:*            | all of Swiss-Prot (fully annotated)              |
| SPTREMBL:*         | Swiss-Prot preliminary EMBL translations         |
| SPT:*              | Swiss-Prot preliminary EMBL translations         |
| GENPEPT:*          | all of GenBank’s CDS translations                |
| GP:*               | all of GenBank’s CDS translations                |
| REFSEQPROT:*       | NCBI RefSeq proteomes                            |
| RS_PROT:*          | NCBI RefSeq proteomes                            |

33
c) The sequence is in a GCG format multiple sequence file, either an MSF (multiple sequence format) file or an RSF (rich sequence format) file. The difference is that MSF files contain only the sequence names and sequence symbol characters, whereas RSF files contain names, annotation, and actual sequence data. As in GCG single sequence format, it is always best to retain the suggested GCG extensions, msf or rsf, in order for you to easily recognize what type of file they are without having to look, though it is not required and they could just as well be named Joe.Blow. To specify sequences contained in a GCG multiple sequence file, supply the file name followed by a pair of braces, “{},” containing the sequence specification. For example, to specify all of the sequences in an alignment of elongation 1α and Tu factors, one may use a naming system such as the following: ef1a-tu.msf{*}. Furthermore, one can point to individual members of the alignment or subgroups by specifying their name within the braces, e.g. EF1a-Tu.rsf{eftu_ecoli} to point just to the E coli sequence or EF1a-Tu.rsf{eftu_—} to point at all of the EfTu’s as long as you use a sequence naming convention that retains this convention.

d) The most powerful method of specifying sequences is in a GCG “list” file. This file can have any name though it is convenient to use the extension “.list” to help identify list files in your account. It is merely a list of other sequence specifications, and can even contain other list files within it. The convention to use a GCG list file in a program is to precede it with an at sign, “@.” Furthermore, you can supply attribute information within list files to specify something special about the sequence. This is especially helpful with length attributes that can restrict an analysis to specific portions of a sequence, and can be seen in the example below:

```plaintext
!!SEQUENCE_LIST 1.0

An example GCG list file of many elongation 1α and Tu factors follows. As with all GCG data files, two periods separate documentation from data. ..

my-special.pep begin:24 end:134
SwissProt:EfTu_Ecoli
Ef1a-Tu.msf{*}
/usr/accounts/test/another.rsf{ef1a_*}
@another.list
```

2) SeqLab — a brief history — Steve Smith’s GDE + GCG’s WPI

While working on bacterial ribosomal RNA phylogenies with Walter Gilbert and Carl Woese, Steve Smith realized the need for a comprehensive multiple sequence editor. Nothing existed at the time that satisfied him, so he invented one. In addition to providing the vital editing function, it also served as a menuing system to external functions such as PHYLIP routines and Clustal alignments. He called it the “Genetic Data Environment” (Smith, et al., 1994). Many people were very impressed, and he made it freely available, though it was designed just for Sun workstations. Coincidentally GCG realized the need for some sort of a ‘point-and-click’ environment for their system. They were losing lots of business, only being able to provide a command line interface. Therefore, they started trying to develop a GUI for the Wisconsin Package. They called it the “Wisconsin Package Interface.” Nobody was impressed — it was a terrible attempt. It only provided a menu to their programs, hardly anything more than the “—check” option they’ve always had. So they did a natural and very smart thing. They hired Steve
Smith away from Millipore, where he had newly moved, into their company, so that he could merge his GDE with their WPI. The offspring was SeqLab, and, thank goodness, they threw away the acronyms. As ‘they’ say “The rest is history” and once more GCG’s customers are (generally) happy.

3) And what’s the deal with the new “+” programs?

Quoting directly from the GCG Program Manual:

“Advantages of Plus “+” Programs:

- Plus programs are enhanced to be able to read sequences in a variety of native formats such as GCG RSF, GCG SSF, GCG MSF, GenBank, EMBL, FastA, SwissProt, PIR, and BSML without conversion.
- Plus programs remove sequence length restriction of 350,000 bp.

If you do not need these features and wish to have more interactivity, you might wish to seek out and run the original program version.”

So, yes, they can cope with huge sequences, but they are much less ‘friendly’ than their old counterparts.

Appendix II

A basic guide to UNIX for neophytes

Because this is all somewhat confusing to newcomers, here’s a basic UNIX tutorial that we won’t take the time to go through today, but I encourage you to do so at some point.

The original UNIX operating system (OS) was developed in the USA, first by Ken Thompson (no relation) and Dennis Ritchie at AT&T’s BELL Labs in the late 1960’s; it is now used in various implementations, on many different types of computers the world over. One of the most popular variations is RedHat Linux. RedHat is a commercial distribution of the free, UNIX derived, Open Source Linux OS. Linux was invented in the early 1990’s by a student at the University of Helsinki in Finland named Linus Torvalds as a part-time ‘hobby.’ FreeBSD (from the U.C. Berkley UNIX implementation) is another popular Open Source UNIX OS.

All UNIX OSs are a line-oriented system similar conceptually to the old MS-DOS OS, though many GUIs exist to help drive them. It is possible to use many UNIX computers without ever-learning command line mode and using a “shell” terminal window. However, becoming familiar with some basic UNIX commands will make your computing experience much less frustrating. The shell program is your command line interface to the UNIX OS. It interprets and executes the commands that you type. Common UNIX shells include bash, the C shell, and a popular C shell derivative called tcsh. tcsh, like bash, enables command history recall using the keyboard arrow keys, accepts tab word completion, and allows command line editing. Among numerous UNIX command line guides available on the Internet, there’s a very good beginning UNIX tutorial at http://www.ee.surrey.ac.uk/Teaching/Unix/, if you would like to see an alternative approach to what I present.
The UNIX command line is often regarded as very unfriendly compared to other OSs. Actually UNIX is quite straightforward, especially its file systems. UNIX is the precursor of most tree structured file systems including those used by MS-DOS, Microsoft Windows, and the Macintosh OS. These file systems all consist of a tree of directories and subdirectories. The OS allows you to move about within and to manipulate this file system. A useful analogy is the file cabinet metaphor — your account is analogous to the entire file cabinet. Your directories are like the drawers of the cabinet, and subdirectories are like hanging folders of files within those drawers. Each hanging folder could have a number of manila folders within it, and so on, on down to individual files. Hopefully all arranged with some sort of logical organizational plan. Your computer account should be similarly arranged.

1) Generalities

In command line mode each command is terminated by the ‘return’ or ‘enter’ key. UNIX uses the ASCII character set and unlike some OSs, it supports both upper and lower case. A disadvantage of using both upper and lower case is that commands and file names must be typed in the correct case. Most UNIX commands and file names are in lower case. Commands and file names should not include spaces nor any punctuation other than periods (.), hyphens (−), or underscores (_). UNIX command options are specified by a required space and the hyphen character (−). UNIX does not use or directly support function keys. Special functions are generally invoked using the ‘Control’ key. For example a running command can be aborted by pressing the “Control” key [sometimes labeled “CTRL” or denoted with the karat symbol (^)] and the letter key “c” (think c for cancel). The short form for this is generally written CTRL-C or ^C. Using control keys instead of special function keys for special commands can be hard to remember, the advantage is that nearly every terminal program supports the control key, allowing UNIX to be used from a wide variety of different platforms that might connect to the server.

The general command syntax for UNIX is a command followed by some options, and then some parameters. If a command reads input, the default input for the command will often come from the interactive terminal window. The output from a system level command (if any) will generally be printed back to your terminal window. General UNIX command syntax follows:

```
  cmd
  cmd -options
  cmd -options parameters
```

The command syntax allows the input and outputs for a program to be redirected into files. To cause a command to read from a file rather than from the terminal, the “<” sign is used on the command line, and the “>” sign causes the program to write its output to a file (for programs that don’t do this by default, also “>>” appends output to the end of an existing file):

```
  cmd -options parameters < input
  cmd -options parameters > output
  cmd -options parameters < input > output
```
To cause the output from one program to be passed to another program as input a vertical bar (|), known as the "pipe," is used. This character is <shift><\> on most USA keyboards:

    cmd1 -options parameters | cmd2 -options parameters

This feature is called "piping" the output of one program into the input of another.

Certain printing (non-control) characters, called "shell metacharacters," have special meanings to the UNIX shell. You rarely type shell metacharacters on the command line because they are punctuation characters. However, if you need to specify a filename accidentally containing one, turn off its special meaning by preceding the metacharacter with a "\" (backslash) character or enclose the filename in "'" (single quotes). The metacharacters "*" (asterisk), "?" (question mark), and "~" (tilde) are used for the shell file name "globbing" facility. When the shell encounters a command line word with a leading "~", or with "*" or "?" anywhere on the command line, it attempts to expand that word to a list of matching file names using the following rules: A leading "~" expands to the home directory of a particular user. Each "*" is interpreted as a specification for zero or more of any character. Each "?" is interpreted as a specification for exactly one of any character, i.e.:

    ~  The tilde specifies the user's home directory (C shell and tcsh only, same as $HOME).
    *  The asterisk matches any string of characters zero or longer,
    ?  The question mark matches any single character.

The latter two globbing shell metacharacters cause 'wild card expansion.' For example, the pattern "dog*" will access any file that begins with the word dog, regardless of what follows. It will find matches for, among others, files named "dog," "doggone," and "doggy." The pattern "d?qg" matches dog, dig, and dug but not ding, dang, or dogs; "dog?" finds files named "dogs" but not "dog" or "doggy." Using an asterisk or question mark in this manner is called using a "wild card." Generally when a UNIX command expects a file name, "cmd filename," it's possible to specify a group of files using a wild card expression.

A couple of examples using wild card characters along with the pipe and output redirection follow:

    cmd *//*.data | cmd2
    cmd my.data? > filename

The first example will access all files ending in "*.data" in all subdirectories one level below the current directory and pass that output on to the second command. The second example will access all files named "my.data" that have any single character after the word data in your current directory and output that result to a file named filename. Wild cards are very flexible in UNIX and this makes them very powerful, but you must be extremely careful when using them with destructive commands like "rm" (remove file).

Four other special symbols should be described before going on to specific UNIX commands:

    /    Specifies the base, root directory of the entire file system.
    .    Specifies the current working directory.
Specifies the parent directory of current working directory.

Execute the specified command in another process.

2) Important UNIX commands and keystroke conventions

Remember to do things in the following sections that are in bold. Do things in the right order, without skipping anything. That way it will work! Some may seem repetitive, but remember, repetition fosters learning.

Getting help in any OS can be very important. UNIX provides a text-based help system called man pages. You use man pages by typing the command “man” followed by the name of the command that you want help on. Most commands have online documentation available through the man pages. Give the command “man tcsh” to see how the man command pages you through the manual pages of the help system, and to read about the T shell:

```bash
> man tcsh
```

Press the space bar to page through man pages; type the letter “q” for quit to return to your command prompt. A helpful option to man is “-k,” which searches through man page titles for specified words:

```bash
> man -k batch
```

Gets you the title lines for every command with the word batch in the title.

Another help system, “info,” may be installed as well. Use it similarly to man, i.e. “info cmd.”

When an account is created, your home directory environment variable, “$HOME,” is created and associated with that account. In any tree structured file system the concept of where you are in the tree is very important. There are two ways of specifying where things are. You can refer to things relative to your current directory or by its complete ‘path’ name. When the complete path name is given by beginning the specification with a slash, the current position in the directory tree is ignored. To find the complete path in the file system to your current directory type the command “pwd” (print working directory). My server’s example follows:

```bash
> pwd
/home/thompson
```

This UNIX command shows you where you are presently located on the server. It displays the complete UNIX path specification (this always starts with a slash) for the directory structure of your account. Also notice that UNIX uses forward slashes (/) to differentiate between subdirectories, not backward slashes (\) like MS-DOS. The pwd command can be used at any point to keep track of your location. Several commands for working with your directory structure follow:

```bash
> pwd
Print working directory. Shows where you are at in the file system. Very useful when you get confused. (Also see “whoami” if you’re really confused!)
> ls
Shows (lists) your files’ names, i.e. the contents of the current directory
> ls -l
Shows files’ names in extended (long) format with size, ownership, and permissions.
> ls -al
Shows all files including dot systems files in your directory in the long format.
> mkdir newdir
Makes a new directory named “newdir” in your current directory.
```
> cd newdir  Move down into a directory named "newdir" from your current directory.
> cd  Move back into your home directory from anywhere (with most shells).
> rmdir newdir  Removes a subdirectory from your current directory. Directory must be empty.

To list the files in your home directory, use the "ls" command. There are many options to the ls command. Check them out by typing "man ls". The most useful options are the "-l," "-t," and "-a" options. These options can be used in any combination, e.g. "ls –alt." The "-l" option will list the files and directories in your current directory in a 'long' form with extended information. The "-t" option displays files ordered by 'time,' with the most recent first. The "-a" option displays 'all' files, even files with a period as the first character in their name, a UNIX convention to hide important system files from normal listing.

This convention has lead to a number of special configuration files with periods as the first character in their name. Some of these are executed automatically when a user logs in, just like "AUTOEXEC.BAT" and "CONFIG.SYS" are by MS-DOS/Windows. Many UNIX systems execute a file called ".login" and another one that sets up the shell environment called ".cshrc" or ".tcshrc" upon every login. Don't mess with these until you are quite comfortable with UNIX. Three examples of the ls command in my account follow:

> ls
bin  EF1a-primitive.DNA.nex  mail  prime.csh  SPDBV
Cn3D_User  EF1a-Tu.DNA.nex  molevol  ribo_files  tutorials
cut.csh  ggc  nsmail  seglab  working
db_info  Latitude  patterns  snap_files

> ls -l
total 228
drwxr-xr-x  4 thompson ggc  4096 Oct 25 12:07 bin
drwxr-xr-x  2 thompson ggc  4096 Oct 25 17:42 Cn3D_User
-rwxr-xr-x  1 thompson ggc  162 Feb 13  2003 cut.csh
drwxr-xr-x  2 thompson ggc  4096 Jan 16  2001 db_info
-rw-r--r--  1 thompson ggc  82710 Apr 28  2005 EF1a-primitive.DNA.nex
-rw-r--r--  1 thompson ggc  63796 Apr 28  2005 EF1a-Tu.DNA.nex
drwxr-xr-x  2 thompson ggc  4096 Oct 11 17:32 ggc
-rw-r--r--  1 thompson ggc  7401 Jan  17  2000 Latitude
drwx-----  2 thompson ggc  4096 Jan 27  2005 mail
drwxr-xr-x  9 thompson ggc  4096 Aug 12  2004 molevol
drwx-----  2 thompson ggc  4096 Oct 18 12:33 nsmail
drwxr-xr-x  4 thompson ggc  4096 Jun  3  1999 patterns
-rw-r--r--  1 thompson ggc  538 Apr  1  2004 prime.csh
drwxr-xr-x 15 thompson ggc  4096 Oct 16  2001 ribo_files
drwxrwxr-x  2 thompson ggc  4096 Oct 25 11:53 seglab
drwxr-xr-x  4 thompson ggc  4096 Oct 25 12:05 snap_files
drwxr-xr-x  7 thompson ggc  4096 Jan 17  2000 SPDBV
drwxr-xr-x  7 thompson ggc  4096 Oct 19 14:05 tutorials
drwxr-xr-x 17 thompson ggc  4096 Nov  8 14:11 working

> ls -a
.
..  .forward  .mailcap  .pauphistory  snap_files
bin  .history  .mime.types  prime.csh  .ssh
Cn3D_User  .java  molevol  ribo_files  tutorials
cut.csh  Latitude  .mozilla  seglab  working
db_info  .Latitude  .ncftp  .seglab-history  .wp
EF1a-primitive.DNA.nex  .login  nsmail  .seglab-mendel  .Xauthority
EF1a-Tu.DNA.nex  mail  patterns  .segmerge
In the output from “ls -l” additional information regarding file permissions, owner, size, and modification date is shown. In the output from “ls -a” all those ‘dot’ systems files are now seen. Nearly all OSs have some way to customize your login environment with editable configuration files; UNIX uses these dot files. An experienced user can put commands in dot files to customize their individual login environment.

Another example of the ls command, along with output redirection is shown below. Issue the following command to generate a file named “program.list” that lists all of the file names in long format located in your server’s “/usr/local/bin/” directory:

```bash
> ls -l /usr/local/bin/ > program.list
```

Rather than scrolling the ls output to the screen, this command redirects it into the file “program.list”

An environment variable, your $PATH, tells your account what directories to look in for programs; /usr/local/bin/ above, is in your path, so you can run any of the programs in “program.list” by just typing its name. You can see your complete path designation by using the command “echo,” along with $PATH, which ‘echoes’ its meaning to the screen. Each path, of the several listed, is separated by a colon:

```bash
> echo $PATH
/usr/local/gcg/License_Pack/Linux_2_Intel_32/exe:/usr/local/gcg/License_Pack/bin:/usr/local/gcg/bin:/usr/local/bin:/bin:/usr/bin:/usr/java/j2sdk1.4.0_01/bin:/usr/local/Cn3D-4.1:/usr/X11R6/bin
```

Subdirectories are generally used to group files associated with one particular project or files of a particular type. For example, you might store all of your memorandums in a directory called “memo.” The “mkdir” command is used to create directories and the “cd” command is used to move into directories. The special placeholder file “..” allows you to move back up the directory tree. Check out its use below with the cd command to go back up to the parent of the current directory:

```bash
> mkdir memo
> ls
bin  gcg  mail  molevol  ribo_files  snap_files  temp.ps  working
db_info  login.bak  memo  patterns  seqlab  temp.epsf  tutorials
> cd memo
> pwd
/home/thompson/memo
> cd ..
> pwd
/home/thompson
```

After the “cd ..” command pwd shows that we are ‘back’ in the home directory. Note that with most shells “cd” all by itself will take you all the way home from anywhere in your account. Next let’s look at several basic commands that affect the file system and access files, rather than directories:
> **cat program.list**

Displays contents of the file “program.list” to screen without pauses; also concatenates files (appends one to another), e.g: “cat file1 file2 > file3” or “cat file1 >> file2.”

> **more program.list**

Shows the contents of the file “program.list” on the terminal one page at a time; press the space bar to continue. Type a “?” when the scrolling stops for viewing options. Type “/pattern” to search for “pattern.” (less is often available; it’s more powerful than more — silly computer systems humor).

> **head program.list**

Shows the first few lines of the file “program.list,” optionally –N lines from the top of the file.

> **tail program.list**

Show the last few lines of the file “program.list,” optionally –N lines from the bottom of the file.

> **wc program.list**

Counts the number of characters, words, and lines in specified file, “program.list.”

> **cp program.list tmp1**

Copies the file “program.list” to the file “tmp1.” Any previous contents of a file named “tmp1” are lost.

> **mv program.list tmp2**

Renames (moves) the file “program.list” to the file “tmp2.” Any previous contents of a file “tmp2” are lost, and “program.list” no longer exists.

> **cp tmp2 memo**

Since “memo” is a directory name not a file name, this command copies the specified file, “tmp2,” into the specified directory, “memo,” keeping the file name intact. Use the “-R” recursive option to copy all files down through a directory structure.

> **rm tmp2**

Deletes (removes) the file “tmp2” in the current directory.

> **rm memo/tmp2**

Deletes (removes) the file “tmp2” in the directory “memo.” It is unrecoverable and permanently gone!

More commands that deal with files:

```bash
rm -r somedir
```

Removes all the files, and subdirectories of a directory and then removes the directory itself — very convenient, very useful, and very dangerous. Be careful!

```bash
chmod somefile
```

Changes the permissions of a file named “somefile.” See “man chmod” and also “man chown” for further (and extensive) details.

```bash
lpr somefile
```

Prints the specified file on a default printer. Specify a particular print queue with the “-p” option to send it elsewhere.

Another example using the `/usr/local/bin/` program list is shown here. This time the ls output is piped to the more command rather than redirected into a file:

```bash
> ls -l /usr/local/bin/ | more
```
A useful command that allows searching through the contents of files for a pattern is called grep. The first parameter to grep is a search pattern; the second is the file or files that you want searched. For example, if you have a bunch of different data files whose file names all end with the word “.data” in several different subdirectories, all one level down, and you wanted to find the one that has the word zebra within it, you could “grep zebra */*.data.” Use the following variation of the grep command to see all the programs in our Mendel program list that have the word “pro” in them:

> grep pro tmpl  
Show the lines in the file “tmpl” that contain the specified pattern, here the word “pro” (these are all PHYLIP protein sequence specific phylogenetics programs).

Another file searching command, “find,” looks not within files’ contents, but rather at their names, to help you find files that are lost in your directory structure. Its syntax is a bit strange, not following the usual rules:

> find . -name ‘*tmp*’  
Finds files from the current directory (.) down containing the word tmp anywhere within its filename. Note that the single quotes (‘’) are necessary for wild card expansion to occur with the find command.

Commands for looking at the system, other users, your sessions and jobs, and command execution follow:

> uptime  
Shows the time since the system was last rebooted. Also shows the “load average”. Load average indicates the number of jobs in the system ready to run. The higher the load average the slower the system will run.

> w (or who)  
Shows who is logged in to the system doing what.

> top  
Shows the most active processes on the entire machine and the portion of CPU cycles assigned to running processes. Press “q” to quit.

> ps  
Shows your current processes and their status, i.e. running, sleeping, idle, terminated, etc. Use “man ps” as options vary widely, see especially the -a, -e, -l, and -f options.

> ps –U user  
Perhaps (user is you) the most useful ps option — show me all of MY processes!

Some more process commands are shown here:

at  
Submit script to the at queue for execution later.

bg  
Resumes a suspended job in background mode.

fg  
Brings a background job back into interactive mode.

And the command to change your password:

passwd  
Change your login password.

Usually it is best to leave programs using a quit or exit command; however, occasionally it’s necessary to terminate a running program. Here are some useful commands for bailing out of programs:

<Ctrl c >  
Abort (cancel) a running process (program); there’s no option for restarting it later.
<Ctrl d > Terminate a UNIX shell, i.e. exit present control level and close the file. Use “logout” or “exit” to exit from your top-level login shell.

<Ctrl z > Do not use this to stop a job! It pauses (suspends) a running process and returns the user to the system prompt. The suspended program can be restarted by typing “fg” (foreground). If you type “bg” (background), the job will also be started again, but in background mode.

> kill –9 psid Kills a process with the given process ID using the “sure kill” option. The PSID number is obtained using some variation of the ps command.

3) Text editing — the good, the bad, and the ugly

Text editing is often a necessary part of computing. This is never that much fun, but it can be very, very important. As mentioned earlier, you can use your own favorite word processor like Microsoft Word, if you insist, but be sure to “Save As” “Text Only” with “Line Breaks,” and specify UNIX line breaks, if you have the choice. Native word processing format contains a whole bunch of binary control data specifying format and fonts and so forth; the UNIX OS can’t read it at all. Saving as text only avoids this problem. Using an ASCII text editor like BBEdit on a Macintosh avoids the binary problem, but you still need to be careful to save with UNIX style line breaks.

Editing files on your own personal computer and then using them on a different computer is a two-step process though. After all the editing is done, the file will need to be transferred with scp or sftp to the UNIX server where it will be used. Therefore, it makes sense to get comfortable with at least one UNIX text editor. That will avoid the file transfer step, saving some hassle and time. There are several around, including many driven through a GUI, but minimally I recommend learning pico (or nano). It’s description, along with two alternatives follow. Launch pico of the tmp1 file with the following command:

> pico tmp1 A simple text editor provided with the pine mailer. It is quite appropriate for general text editing, but is not present on all UNIX systems (it can be installed on any UNIX system). This is a very easy to use editor with a command banner presenting a menu of Ctrl Key command options. Type some sample text into the file, then press <Ctrl x> to exit, reply “y” for yes to save the file, and then accept or modify the file’s name.

Two other command line UNIX editors are emacs and vi:

emacs file This is a very nice alternative text editor available on many UNIX machines. This editor is also quite powerful but not nearly as difficult to learn as vi.
The default UNIX text editor. This comes with all versions of UNIX and is extremely powerful, but it is quite difficult to master. I recommend avoiding it entirely unless you are interested in becoming a true UNIX expert.

4) File transfer — getting stuff from here to there, and there to here

You will often need to move files back and forth between different computers. Remember scp from section II.4. That’s the primary secure way to move files around within the Internet. I never use removable media like floppy or Iomega disks, or CDs, or USB drives anymore. I just copy files between machines over the Internet. The commands in the following table provide simple access to a small subset of UNIX networking capabilities (host refers to a computer’s fully qualified Internet name or number).

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftp host</td>
<td>File transfer protocol. Allows a limited set of commands (dir, cd, put, get, help, etc.) for moving files between machines. Note: insecure method, so often restricted to particular servers that allow “anonymous ftp” only. See sftp and scp as an alternative.</td>
</tr>
<tr>
<td>scp</td>
<td>Secure copy file, syntax: “scp file user@host:path” or “scp user@host:path file.” Good for moving one or a few files at a time.</td>
</tr>
<tr>
<td>sftp</td>
<td>Secure file transfer protocol. Allows same subset of commands as ftp, but through an encrypted connection. Good for moving lots of files.</td>
</tr>
<tr>
<td>telnet host</td>
<td>Provides an insecure terminal connection to another Internet connected host (discouraged and often disabled!). See ssh for a secure alternative.</td>
</tr>
<tr>
<td>ssh user@host</td>
<td>Connect to a host computer using a secure, encrypted protocol. This is often the only allowed way to interactively log onto a remote computer.</td>
</tr>
</tbody>
</table>

I’ll illustrate scp to give you a feel for its syntax. Note the required colon “:” in the command. In its simplest form:

```bash
> scp file user@somemachine.somewhere:
```

You’ll most likely get the same sort of authenticity question as when you first connect to a machine with ssh; answer “yes” and then supply your password. This will put a copy of the file from your current, ‘local’ machine to your home directory on the scp connected, ‘remote’ machine.

Let’s do it the other way ‘round now, that is, from a remote server to a local machine, with a few extra twists:

```bash
> scp user@somemachine.somewhere:somedir/file somedir/somefile
```

OK, what does this command do? It logs you onto a remote machine and looks for a file named “file” in your “somedir” directory there. Then it copies that file into your “somedir” directory on your local machine that you’re already logged onto and it changes its name from “file” to “somefile.” scp also supports a “-r” recursive option so that it can be used to secure copy down through the contents of a directory structure. Simple enough. Got it?
Microsoft Windows machines and Macintoshes often have a GUI form of scp/sftp installed. In the Microsoft Windows world this may be called secure file transfer client, and on OS X Macintoshes a great little free program named Fugu can be used (http://rsug.itd.umich.edu/software/fugu/). Let's get rid of those tmp files now before proceeding. Note that you can remove more than one file specification at a time. Issue the following command:

```
> rm tmp* */tmp*
```

5) Using X between different UNIX computers

These are the bare-minimum instructions necessary for connecting to a UNIX host computer from another UNIX computer using X. Not all commands are necessary in all cases, as they are often set by your account environment; however, I'll supply a complete set. In most cases fully qualified Internet names can be used in these procedures, however, depending on local name servers, you may need to specify IP numbers. A fictitious example host machine, zen.art.motorcycle.com, has the following name and number:

```
zen.art.motorcycle.com 999.999.99.99
```

You will need to know your own machine's name and/or number as well as the host's.

Log on to your UNIX workstation account in the customary manner. Depending on the workstation, you may want to specify an xterm terminal window if your terminal window is not already an xterm. On most systems:

```
Optional: > /usr/bin/X11/xterm &
On Solaris: > /usr/openwin/bin/xterm &
```

Following UNIX X commands with an ampersand, "&," is helpful so that they are run in the background in the new window in order to maintain control of the initial window. Some helpful options supported in most versions of xterm are "-ls" so that your login script is read, "-sb -sl 100" to give you a 100 line scroll back capability, "-tn vt220" to take advantage of vt220 terminal features, and "-fg Bisque -bg MidnightBlue" to give you nice light colored characters on a dark blue background.

Then at your workstation's UNIX prompt, authorize X access to the host with the xhost command:

```
> xhost +zen.art.motorcycle.com (should not be necessary)
```

Next connect to the host with the ssh command; e.g:

```
> ssh -X thompson@zen.art.motorcycle.com (-capital X sets the X environment for you)
```

This should produce a login window. Log in as usual, then, if necessary, issue the following command on the host to setup the X environment (for the c shell and its derivatives), where your_IP_node_name represents the Internet name or number of the workstation that you are sitting at:

```
Host> setenv DISPLAY your_IP_node_name:0 (again, should not be necessary)
```
It is best to run commands from an X terminal window rather than from a default console window as is sometimes created by a remote connection. Therefore, after setting up your environment, an option is to launch xterm by minimally issuing the xterm command to the host (as discussed above, many options are available).

UNIX is not the easiest OS to learn. Have patience, ask questions, and don’t get down on yourself just because it doesn’t seem as easy as other OSs. The power and flexibility of UNIX is worth the extra effort. Plus, UNIX is the de facto standard OS for most scientific computing, so the effort will not be wasted.

Acknowledgement

Special thanks to Charles Severance for providing much of the material on which the basic UNIX guide portion of this tutorial was based (http://www.msu.edu/~systems/ucgdocuments/misc/unixtut.html).