An Introduction to Sequence Similarity ("Homology") Searching

Gary D. Stormo¹

¹Washington University, School of Medicine, St. Louis, Missouri

ABSTRACT

Homologous sequences usually have the same, or very similar, functions, so new sequences can be reliably assigned functions if homologous sequences with known functions can be identified. Homology is inferred based on sequence similarity, and many methods have been developed to identify sequences that have statistically significant similarity. This unit provides an overview of some of the basic issues in identifying similarity among sequences and points out other units in this chapter that describe specific programs that are useful for this task. *Curr. Protoc. Bioinform.* 27:3.1.1-3.1.7. © 2009 by John Wiley & Sons, Inc.

Keywords: sequence similarity • homology • dynamic programming • similarity-scoring matrices • sequence alignment • multiple alignment • sequence evolution

AN INTRODUCTION TO IDENTIFYING HOMOLOGOUS SEQUENCES

Upon determining a new sequence, one common objective is to assign a function to that sequence, or perhaps multiple functions to various subsequences. Because homologous sequences, those related by common ancestry, generally have the same or very similar functions, their identification can be used to infer the function of the new sequence. The methods for identifying homologous sequences rely on finding similarities between sequences that are unlikely to happen by chance, thereby allowing the inference that the sequences have evolved from a common ancestor. The units in this Chapter all address aspects of this problem, finding significant similarity between new sequences and those that are currently in the databases of known sequences. This chapter introduction provides a primer on five fundamental issues related to similarity searches: (1) how to find the best alignment between two sequences; (2) methods for scoring an alignment; (3) speeding up database searches; (4) determining statistical significance; and (5) using information in multiple alignments to improve sensitivity.

OPTIMAL SEQUENCE ALIGNMENTS

Given any two sequences, there are an enormous number of ways they can be aligned if one allows for insertions and deletions (collectively referred to as indels or gaps) in the alignment. In order to assess whether the similarity score between two sequences is significant, one must obtain the highest possible score; otherwise one will underestimate the significance. Fortunately, there is a simple method, known as dynamic programming (DP), to determine the score of the best possible alignment between two sequences. Figure 3.1.1 outlines the DP algorithm. The two sequences, A and B, are written across the top and down the side of the DP matrix. Each element of the matrix, S(i,j), represents the score of the best alignment from the beginning of each sequence up to residues aᵢ in sequence A, and bᵢ in sequence B. That element is determined by comparing just three numbers, which represent the three possible end points of those alignments: aᵢ is aligned to bᵢ; aᵢ is aligned to a gap in sequence B; bᵢ is aligned to a gap in sequence A. The three numbers to be compared include the addition of the gap score, δ (which is always...
Figure 3.1.1 Dynamic programming algorithm for optimum sequence alignment. The two sequences are written across the top and along the right side of the matrix. The score of each element is determined by the simple rules shown for the enlarged section and described by the equations below it. (The top row and left column have special rules as described in the text.) The score of the best global alignment is the element $S(n,m)$ and the alignment with that score can be obtained by backtracking through the matrix, determining the path that generated the score at each element.

The other number is the sum of the score of aligning $a_i$ with $b_j$, which depends on what those residues are (see next section) and the score of the best alignment for the preceding subsequences, $S(i-1,j-1)$. The top row and first column require special treatment, but the rule is still simple. Aligning $a_1$ with $b_1$ just gets $\text{score}(a_1,b_1)$, but aligning $a_1$ with $b_j$
requires \(j-1\) gaps in sequence B, so \(S(1,j) = (j-1)\delta + \text{score}(a_1, b_j)\). Once the top row and first column are initialized, the entire matrix can be filled following the simple rule shown in Figure 3.1.1, and the score of the best possible alignment of the two sequences is the element in the lower right corner of the matrix, \(S(n,m)\). The actual alignment that gives that score can be found by tracing backwards through the matrix, starting from the final element, to identify the preceding element that gave rise to each score. Keep in mind that there may be more than one alignment with the highest possible score, but most programs only show one of them.

The result of the DP algorithm described above is a “global alignment” of the two sequences, where each residue of each sequence is included in the alignment and contributes to the score. However, often one is interested in the best “local alignment,” an alignment that may contain only a portion of one, or both, sequences. For example, two proteins may have a common domain that one would like to identify but the remaining parts of the sequences are unrelated. Alternatively, perhaps one is comparing a newly sequenced gene to the entire genome of another species and you only expect it to match the segment corresponding to the homologous gene. A simple modification of the basic DP algorithm can be used to identify the best local alignment, as first deduced by Smith and Waterman (1981). The simple change is that only positive values are kept in the DP matrix, which just means that zero is one of the values to be compared and the maximum taken for \(S(i,j)\). Now the score of the best alignment is not necessarily the element \(S(n,m)\), but may occur anywhere within the matrix. Once the highest score is found, the alignment that generated that score can be found by tracing backwards through the matrix, as before, but stopping when the first zero is encountered.

One additional modification to the basic DP algorithm is often used. It is known from extensive studies of protein sequences that gaps of multiple residues are much more frequent than the same number of separate gaps. For example, a single deletion event might remove three residues at once, and that event is much more likely than for three separate events that each delete one residue. To capture this effect, one can use two gap penalties, one that counts the occurrence of a gap, usually referred to as the gap-opening penalty, and a different, lower one, that is proportional to the length of the gap. This can be easily incorporated into the DP algorithm with only a minor increase in the time required to find the best alignment (Fitch and Smith, 1983). The DP algorithm is guaranteed to give the highest-scoring alignment of the two sequences if the only evolutionary processes allowed are substitutions and indels. While real evolutionary processes also include duplications, transpositions, and inversions, those are not considered, as they cannot be handled by the DP approach and would drastically increase the time required to find the optimal alignment. However, it is generally seen that substitutions and indels are much more common than the other processes, so finding the best aligned considering only those processes will most frequently indicate the true optimum alignment.

**SCORING SEQUENCE SIMILARITY**

*UNIT 3.5* discusses the construction of protein similarity matrices and considerations to take into account when choosing which to use for a specific search. This section briefly describes some of fundamental issues regarding similarity scores. The score of an alignment is the sum of the scores for each position in the alignment, as we saw in the DP algorithm for finding the highest-scoring alignment. For every pair of residues, which might be RNA, DNA, or protein, there is a score defined such that the more similar the sequences are, the higher the score. For RNA and DNA, the scores often just distinguish between matches, where the two residues are the same, and mismatches. However, sometimes it is useful to distinguish between mismatches that are transitions, changes from one purine or pyrimidine to the other, from those that are transversions, changes between a purine and a pyrimidine. For proteins, the situation is more complex and one needs a
score for all possible amino acid pairs and, ideally, one that reflects the probability of one amino acid being replaced by another during the course of evolution. Replacements are governed by two processes. The first is the mutation that changes one amino acid to another, and this will be much more likely for amino acids that have similar codons. For example, a single transition mutation can change a His to a Tyr (CAT to TAT), but three transversions would be necessary to change a His to a Met (CAT to ATG). The second process is that the mutation must survive selection for it to be observed, and this requires that the new amino acid can substitute for the previous one without disrupting the protein’s function. This is more likely if the two amino acids have similar properties, such as size, charge, and hydrophobicity. Which properties are most important may depend on where the amino acid is within the protein sequence and what role it plays in determining the structure and function of the protein. Some positions in a protein may be highly constrained and only a few, perhaps only one, amino acids will work, whereas other positions may tolerate many different amino acids and therefore be highly variable in related protein sequences. This implies that one would really like to have different scoring matrices depending on the constraints at particular positions in the protein, but this would require many more parameters, and often one is lacking the information needed to choose the appropriate matrix. This problem can be partially addressed using multiple alignments of protein families, as described below. However, for general database searches, where one hopes to identify homologous sequences in the collection of known sequences, a general similarity-scoring matrix, which is the average over various types of constrained positions, is often utilized.

Most commonly used similarity matrices are empirically based on reliable alignments of protein families. For instance, the BLOSUM matrices (UNIT 3.5) are based on the alignments in the BLOCKS protein family alignments (UNIT 2.2). These are reliable alignments of protein domains across many different protein families. The score assigned to a specific pair of residues is the log-odds of their co-occurrence in aligned positions compared to what would be expected by chance given their individual frequencies. So for two amino acids, \( a \) and \( b \), the similarity score is:

\[
score(a,b) = \ln \frac{f(a,b)}{f(a)f(b)}
\]

where \( f(a,b) \) is the observed frequency at which those amino acids are aligned and \( f(a) \) and \( f(b) \) are their frequencies in the entire database. If the two amino acids occur together at a frequency expected by chance, which is the product of the independent frequencies, then the score is 0. Amino acids that occur together more often than expected by chance get positive scores and ones that occur less frequently get negative scores. The score for aligning an amino acid with itself is always the highest score for that amino acid, as it is more likely to occur than any substitution. Positive-scoring pairs represent amino acids with similar properties, and they are often observed to substitute for one another, whereas negative scoring pairs are dissimilar amino acids that are less likely to substitute for each other. It is important to note that the expected score of a random alignment is negative. In addition to the amino acid similarity scores, one also needs to have penalties for indels in the alignment, and those values are also based empirically on many reliable alignments over many different protein families.

**FAST SEARCHING METHODS**

The DP method described above is guaranteed to find the highest-scoring alignment between two sequences. The time it takes to complete the alignment is proportional to the number of DP matrix elements to compute, which is the product of the sequence lengths. While this is very fast for comparing any two sequences of reasonable length, it is
not practical for searching the current sequence databases, which contain many millions of sequences and many billions of residues. Therefore methods have been developed that can search entire databases much faster. While these methods do not guarantee finding the absolute best alignments, they have been finely optimized so that they have very high sensitivities and generally do obtain the optimal, or near-optimal, alignments. The most commonly used method is BLAST (Altschul et al., 1990; UNITS 3.3, 3.4 & 3.11). BLAST is convenient to use through the online service (UNITS 3.3 & 3.4), has access to the major DNA and protein databases, and provides convenient tools for displaying and analyzing the output. One can also download BLAST to a local computer for in-house use (UNIT 3.11). But most importantly BLAST has been shown to be very sensitive, giving results nearly equivalent to running the Smith/Waterman DP algorithm on the whole database but in a small fraction of the time.

As a brief and approximate description of how BLAST works, consider the DP matrix of Figure 3.1.1. Most of the time is spent determining the scores of elements that do not contribute to the final, optimum alignment. BLAST achieves its speed by eliminating most of the computations and focusing its effort on the highest-scoring paths through the matrix. It does this by creating an index of the “words,” short subsequences, from the query sequence, and high-scoring matches to those words can be identified very quickly in the database. Those initial matches, or “hits,” are extended using DP but with cutoffs employed so that when any extension becomes unlikely to lead to a significant match it is terminated. The matches that do lead to significant alignments are extended as far as possible to give a result essentially equivalent to that obtained by the full DP algorithm. Since most sequences in the database, and most of the possible alignments even to the most similar sequences, are eliminated quickly from further consideration, BLAST achieves an enormous increase in speed with only a small loss of sensitivity.

THE SIGNIFICANCE OF AN ALIGNMENT SCORE

Obtaining a score for an alignment does not, by itself, tell you whether it is significant. One needs to determine what is the probability of observing such a score by chance, given the scoring system used and the lengths of the sequences being compared. It is important to realize that the distribution of optimum alignment scores is not normal. If one were to consider the scores of all possible alignments, one would expect a normal distribution. However, we are only considering the highest-scoring alignments to each sequence in the database, and the distribution of those maximum, or extreme, values will be quite different from the distribution of scores for all alignments. Consider the following simple example. If one throws two dice, the score distribution follows the binomial distribution, and the probability of getting a 12 is 1/36. But if one throws two pair of dice, and records only the highest score, the distribution is quite different, with the probability of getting a 12 nearly twice as high, 71/362 to be precise. When searching a large database with a query sequence, one is interested only in the highest-scoring alignments, those that are mostly likely to represent homologous sequences. However, to answer the question of how high a score is required to be unlikely to have occurred by chance, one must know the distribution of highest scores expected by chance. Karlin and Altschul (1990) showed that for alignments without gaps, the distribution of highest scores follows a Gumbel distribution, a type of extreme value distribution. Furthermore, they showed how to compute the dependency of that score distribution on the scoring system that is used and the length of the query and database sequences. In addition, while their theory only holds analytically for ungapped alignments, empirical evidence indicates that it holds quite well for gapped alignments too. This allows one to compute a statistical significance for the best scoring alignments in a database search and from that infer whether the best matches are likely to be homologous sequences or could have arisen by chance. The BLAST program computes these values as part of the output, with
most people focusing on the E-value, which is the number matches with equal or greater scores that are expected by chance. When E-values are much less than one, they are essentially the same as p-values, the probability of an equal or greater score by chance, but E-values can exceed one, whereas p-values cannot.

MAKING AND USING MULTIPLE SEQUENCE ALIGNMENTS

The previous sections of this unit have focused on aligning pairs of sequences, or more generally searching an entire database for high-scoring alignments to a query sequence, which requires many pairwise comparisons. However, if one has several members of a protein family, an alignment of those sequences can provide additional information about the sequence constraints at different positions of the proteins. For example, some positions may be critical for the function of those proteins and perhaps only one, or a few, amino acids are allowed at those positions. If that is the case, then a general similarity-scoring matrix is not appropriate and one should utilize the information available from the multiple alignments to constrain the search for homologous proteins. The two main issues are how the multiple alignments are made and how they are used to identify new members of the protein family. Several units in this chapter and the previous one address various aspects of the construction and use of multiple sequence alignments. Unit 3.7 provides an overview of multiple sequence alignment.

One can generate optimal multiple alignments by extending the DP method to multiple sequences, following the same rules as described above but expanded to multiple dimensions, one for each sequence. However, the time required to fill in the DP matrix is proportional to the product of the length of all the sequences, which becomes impractical for more than a few sequences (Lipman et al., 1989). Practical methods for multiple sequence alignment use one of two approaches. One is a progressive alignment strategy where pairs of sequences are aligned and then new sequences are added to that alignment, or alignments are aligned to each other, but all steps only involve pairwise comparisons that can be accomplished efficiently using the DP strategy described. The Pileup program from GCG (Unit 3.6) uses that strategy, as does the ClustalW program (Unit 2.3), which is probably the most popular method for constructing multiple alignments. Psi-BLAST also uses a similar iterative strategy within the BLAST suite of programs (Unit 3.4) to build multiple alignments from the matches identified by BLAST. The other strategy uses an iterative refinement approach, where an initial approximate alignment is built and then sequences are realigned to the resulting model until the method converges to a final multiple alignment. The method is based on an expectation maximization (EM) algorithm, similar to that used in MEME (Unit 2.4) but allowing gaps in the alignment. The EM algorithm is at the heart of the Hidden Markov Model (HMM) approach to protein family models first developed by the Haussler group (Krogh et al., 1994; Eddy, 1996; overview in Unit 3.7 & Appendix 3A). In an HMM each position of the alignment is represented by a probability distribution over all of the amino acids and there are probabilities assigned for insertions and deletions. This model captures, in a probabilistic fashion, the natural variability and constraints that are observed in a protein family. The T-Coffee approach (Unit 3.8) is somewhat different in that it combines, through a consistency-based scoring system, pairwise alignments that can be obtained from a variety of different and independent approaches. For example, if structural information is available and useful for determining the correct alignment that can be incorporated along with other types of information.

Once a multiple alignment is built, it can be used to identify new sequences that are members of the family. Gribskov et al. (1987) first described such an approach where the multiple alignment is converted to a “profile,” and new sequences are scored against it using a DP algorithm and a sum-of-pairs method. This is equivalent to finding the best
alignment of the new sequence to the multiple sequence alignment, where the scoring is the average similarity score to each sequence in the alignment. This is also how ClustalW works internally, as it is building up the multiple alignment (UNIT 2.3). A new sequence can also be aligned to an HMM for a protein family using a version of the DP algorithm, the same method that is used in the iterative realignment for building the HMM. Pfam (UNIT 2.5) is a database of protein families represented as HMMs, and a new sequence can be compared to the database to determine how similar it is to each family and if it can be inferred to belong to any of the families. The InterPro resource of databases and analysis tools (UNIT 2.7) also facilitates the identification of protein families that may be homologous to a new sequence of interest.

**SUMMARY**

When one obtains a new sequence and wants to assign a function to it, identifying homologous sequences with known function is a reliable strategy. Two complementary approaches can be easily employed and are generally very useful. The first is to search the databases of known sequences, using programs like BLAST, to determine if there are any that are sufficiently similar to be homologous, and if those sequences have known functions such that information can be transferred with reasonable reliability. One can also search the databases of known protein families that include the extra information from multiple sequence alignments, such as Pfam and the InterPro databases. While those protein family databases are less comprehensive than the single sequence databases, simply because many sequences do not have known family membership, they can be more sensitive at identifying distantly related sequences that may be missed without knowledge of the specific constraints associated with specific families. Therefore, a combined approach that utilizes both types of resources can be most effective in identifying homologous sequences for a new sequence of interest.

**LITERATURE CITED**


Framesearch (Edelman et al., 1995) is an extension of the classic Smith-Waterman pairwise sequence comparison algorithm. As illustrated in the upper portion of Figure 3.2.1, when the classic Smith-Waterman search algorithm is used to compare a nucleotide query sequence against a database of peptide sequences, it compares the six possible translations of that nucleotide query sequence as peptide sequences against the peptide sequence database. Single-nucleotide indels (INSertion or DELetion errors) are not taken into account because the alignments are between whole codons translated into amino acids. On the other hand, as illustrated in the lower portion of Figure 3.2.1, the Framesearch algorithm includes the possibility of a frameshift error in its alignment algorithm, and therefore can find alignments that span different reading frames. Unfortunately, the great power of Framesearch in finding sequence similarity despite frameshift errors comes at a high cost in computing time. Since Framesearch is an extension of the Smith-Waterman algorithm, it is even more CPU-intensive than a Smith-Waterman search.

Basic Protocol 1 in this unit describes the use of Framesearch in the GCG Wisconsin Package environment to search a protein sequence database for sequences that are similar to a query nucleotide sequence. Basic Protocol 2 describes the use of Framesearch to search a nucleotide sequence database for sequences that are similar to a query protein sequence. Three Alternate Protocols describe ways to improve the speed of Framesearch and thus make it practical for routine use (Alternate Protocols 1, 2, and 3). The Support Protocol describes how to convert FASTA files to GCG format so they are suitable for Framesearch. Framesearch is especially appropriate for low-quality single-read nucleotide sequence data, such as ESTs (expressed sequence tags) or early drafts of genomic sequences; it does not offer any significant advantage over less CPU-intensive algorithms for relatively high-quality nucleotide sequences without many single-nucleotide insertion or deletion errors.

FRAMESEARCH USING A NUCLEIC ACID QUERY SEQUENCE

This protocol describes the use of Framesearch in the GCG Wisconsin Package environment to search a protein sequence database for sequences that are similar to a query nucleotide sequence. Any user familiar with the GCG Package will find using Framesearch in that environment straightforward. Framesearch has recently been added to the algorithms supported by the SeqWeb version of the GCG Package (Accelrys, 2001), so users who prefer a Web-based interface may find it simpler to run Framesearch in the SeqWeb environment (if locally available) rather than at the command line as described in this protocol.

Necessary Resources

Hardware
Framesearch can be run on any Unix or VMS system that has the Wisconsin Package installed; because it is so CPU-intensive, Framesearch should be run on the fastest computer available to the user

Software
GCG Wisconsin Package (v. 8.1 or higher)
Figure 3.2.1  Six-frame-translated search versus Framesearch.
Files

DNA sequence file of interest (this will be the query sequence; maximum length, 350 kb)

Protein database of sequences to which the DNA sequence will be compared

For example, *BA000007.faa* contains the amino acid translations of all putative genes found in this bacterial genome by the lab where it was sequenced, as a single FASTA format text file (APPENDIX 1B).

Both the query sequence and the database files must be converted to the GCG format (Support Protocol).

The files used in this example should be downloaded from NCBI or from the Current Protocols Web site (http://www3.interscience.wiley.com/c_p/cpbi_sampledatafiles.htm) and converted to GCG format, as described in the Support Protocol.

1. Download and convert the sequence files as described in the Support Protocol below. To launch Framesearch, initialize the GCG Wisconsin Package environment and then type:
   ```
   framesearch -check -batch
   ```
   and hit the return key.

   The *-batch* parameter will make the actual search run in the background, which is desirable with such a CPU-intensive program, and the *-check* parameter will make Framesearch interactively prompt the user for the most commonly needed parameters before launching the actual search process in background mode. The documentation that comes with GCG explains all the parameters for Framesearch in exhaustive detail, but the *-check* parameter will prompt the user for the most common ones. As with all GCG programs, at many prompts a default value appears surrounded by parentheses and stars; one can accept that default by simply hitting the return key.

   For example, when the Framesearch program asks *Begin (* 1 *)?*, by hitting the return key one accepts the default value of 1. Similarly, when the Framesearch program later asks *what is the frameshift penalty (* 0 *)?*, by hitting the return key one accepts the default value of 0.

2. Enter the query sequence(s) at the prompt; for this example, type `Nuc_GCG/gi-15829254_01.seq` (the name given to the first part of the nucleotide sequence that was downloaded and converted to GCG format in the Support Protocol). Next, specify the beginning and ending nucleotide positions, defining the portion of the query sequence to use in the search.

   For this example, simply hit the return key after each of these two prompts, since the default values of Begin and End are the beginning and end of the entire query sequence. In Alternate Protocol 1, below, these parameters will be used to run Framesearch using only a portion of the query sequence.

3. Specify what sequence(s) to search. For this example, type `AA_GCG/*pep`, which will make it search for regions of similarity to the query nucleotide sequence in all the peptide sequences that were downloaded in the Support Protocol.

4. Enter the gap creation, gap extension, and frameshift penalty values, and the name for the output file; for this example simply accept default values of all these parameters. (See Critical Parameters for discussion of these parameters).

5. Determine if a graphical histogram of Framesearch quality scores (Fig. 3.2.2) is desired. This plot can be an extremely valuable aid in the interpretation of Frame- search output. The Framesearch program will offer the user a choice of printing the plot on an HP7550 (an old device few sites today are likely to have), generating a FIGURE file, or not generating a plot at all.

Finding Similarities and Inferring Homologies

3.2.3
The suggested option is to generate a FIGURE file. However, to view this FIGURE file, the GCG graphical environment must also be configured. Unfortunately, the details of configuring GCG for graphical output are extremely installation-specific, so one must consult local support staff to learn them.

6. Once all parameters are specified, Framesearch will search in background mode. Even on a relatively powerful computer, this may take as long as a day or more. When the search is done, the output file will appear in the directory where Framesearch was launched.

For the reader’s convenience, a sample copy of this output file can be found on the Current Protocols Web site at the address http://www3.interscience.wiley.com/c_p/cpbi_sample datafiles.htm.

Figures 3.2.3 and 3.2.4 show the most important portions of the Framesearch output generated by this example. For a detailed discussion of these figures, see Guidelines for Understanding Results.
FRAMESEARCH USING A PROTEIN QUERY SEQUENCE

This protocol describes the use of Framesearch in the GCG Wisconsin Package environment to search a nucleotide sequence database for sequences that are similar to a query protein sequence. Any user familiar with the GCG Package will find using Framesearch in that environment straightforward. Framesearch has recently been added to the algorithms supported by the SeqWeb version of the GCG Package (Accelrys, 2001), so users who prefer a Web-based interface may find it simpler to run Framesearch in the SeqWeb environment (if locally available) rather than at the command line as described in this protocol.

**Necessary Resources**

**Hardware**

Framesearch can be run on any Unix or VMS system that has the Wisconsin Package installed; because it is so CPU-intensive, Framesearch should be run on the fastest computer available to the user.

**Software**

GCG Wisconsin Package (v. 8.1 or higher)
Files

Protein sequence file of interest (this will be the query sequence)
Nucleic acid database of sequences to which the protein sequence will be compared

For example, BA000007.fna contains the nucleotide sequence of all putative genes found in this bacterial genome by the laboratory where it was sequenced, as a single FASTA format text file (APPENDIX 1B).

Both the query sequence and the database files must be converted to the GCG format (Support Protocol).

The files used in this example should be downloaded from NCBI or from the Current Protocols Web site (http://www3.interscience.wiley.com/c_p/cpbi_sampledatafiles.htm) and converted to GCG format, as described in the Support Protocol.

1. Download and convert the sequence files as described in the Support Protocol below.
To launch Framesearch, initialize the GCG Wisconsin Package environment, and then type:

```
framesearch -check -batch
```

and hit the return key.
The --batch parameter will make the actual search run in the background, which is desirable with such a CPU-intensive program, and the --check parameter will make Framesearch interactively prompt the user for the most commonly needed parameters before launching the actual search process in background mode. The documentation that comes with GCG explains all the parameters for Framesearch in exhaustive detail, but the --check parameter will prompt the user for the most common ones. As with all GCG programs, at many prompts a default value appears surrounded by parentheses and stars; one can accept that default by simply hitting the return key.

For example, when the Framesearch program asks Begin (* 1 *)?, by hitting the return key one accepts the default value of 1. Similarly, when the Framesearch program later asks what is the frameshift penalty (* 0 *)? by hitting the return key one accepts the default value of 0.

2. Enter the name of the query sequence(s); for this example, type AA_GCG/gi-13361126.pep (the name that FROMFASTA gave to one of the protein sequences that was downloaded and converted to GCG format in the Support Protocol). Next, specify the beginning and ending residue positions, defining the portion of the query sequence to use in the search.

For this example, simply hit the return key after each of these two prompts, since the default values of Begin and End are the beginning and end of the entire query sequence. In Alternate Protocol 2, below, these parameters will be used to run Framesearch using only a portion of the query sequence.

3. Specify what sequence(s) to search against. For this example type Nuc_GCG/*seq, which will search for regions of similarity to the query protein sequence in all the nucleotide sequences that were downloaded in the Support Protocol.

4. Specify the gap creation, gap extension, and frameshift penalty values, and a name for the output file; for this example simply accept default values of all these parameters. (See Critical Parameters for discussion of these parameters).

5. Determine if a graphical histogram of Framesearch quality scores is desired. This plot can be an extremely valuable aid in the interpretation of Framesearch output. The Framesearch program will offer the user a choice of printing the plot on an HP7550 (an old device few sites today are likely to have), generating a FIGURE file, or not generating a plot at all.

The suggested option is to generate a FIGURE file. However, to view this FIGURE file, the GCG graphical environment must also be configured. Unfortunately, the details of configuring GCG for graphical output are extremely installation-specific, so one must consult local support staff to learn them.

6. Once all parameters are specified, Framesearch will search in background mode. Even on a relatively powerful computer, this may take as long as a day or more to run. When the search is complete, the output file will appear in the directory where Framesearch was launched.

For the reader’s convenience, a sample copy of this output file can be found on the Current Protocols Web site at the address http://www3.interscience.wiley.com/c_p/cpbi_sampledatafiles.htm.

Figures 3.2.5 and 3.2.6 show the most important portions of the Framesearch output generated by this example. For a detailed discussion of these figures, see Guidelines for Understanding Results.
ALTERNATE PROTOCOL 1

PREFILTERING WITH A SEARCH ALGORITHM TO IMPROVE THE SPEED OF FRAMESearch WITH A NUCLEIC ACID QUERY SEQUENCE

The major advantage of Framesearch is its facility at aligning protein sequences to nucleotide sequences that contain numerous single-nucleotide insertion or deletion errors. When a nucleotide sequence is compared to a protein sequence, most similarity search algorithms translate all six possible reading frames into protein, then perform a protein sequence comparison (UNIT 3.4). If a region of nucleotide sequence that is similar to a protein sequence is interrupted by a frameshift error, then what should be one match will become two shorter matches in different reading frames.

Figure 3.2.5 The list of hits from a Framesearch run in which an amino acid sequence was used to search a number of nucleotide sequences. The name of the query sequence, the wildcard expression specifying the target sequences, and the name of the nucleotide sequence with the best match have been boldfaced in the sample output.
However, unless the stretches of nucleotide sequence between single-nucleotide insertions or deletions (indels) are very short, a translated search with BLAST (UNIT 3.4) is likely to find at least part of the alignment, if not the whole stretch of similar sequence. BLAST (UNIT 3.4) will do a typical similarity search several orders of magnitude faster than Framesearch. Therefore, one can take advantage of the speed offered by BLAST (UNIT 3.4) to find possible hits, then use Framesearch to improve the alignment near each hit.

**Necessary Resources**

**Hardware**
Framesearch can be run on any Unix or VMS system that has the Wisconsin Package installed; because it is so CPU-intensive, Framesearch should be run on the fastest computer available to the user.

**Software**
- GCG Wisconsin Package (v. 8.1 or higher)
- BLAST program (UNIT 3.4)
**Finding Homologs Using the Framesearch Program**

In the GCG environment assumed for these examples, both BLAST and Framesearch are included.

**Files**

DNA sequence file of interest (this will be the query sequence; maximum length, 350 kb)

Protein database of sequences to which the DNA sequence will be compared

*For example, BA000007.faa contains the amino acid translations of all putative genes found in this bacterial genome by the lab where it was sequenced, as a single FASTA format text file (APPENDIX 1B).*

Both the query sequence and the database files must be converted to the GCG format (Support Protocol).

The files used in this example should be downloaded from NCBI or from the Current Protocols Web site (http://www3.interscience.wiley.com/cpbi_sampledatafiles.htm) and converted to GCG format, as described in the Support Protocol.

1. Download the files, set up the GCG environment, and run FROMFASTA on the data files as described in the Support Protocol below.

2. Use your nucleotide sequence as the query in BLAST searches of the appropriate protein database (see **UNITS 3.3 & 3.4** for more about BLAST).

   *If a BLAST search using the default parameters does not find any matches, it may be necessary to increase the cutoff E-score to a value greater than 1, possibly even as high as 10 or more, in order to obtain at least one match. Note which regions of the nucleotide sequence are similar to which protein sequences.*

3. For each BLAST hit found in step 2, run a Framesearch job as described above in Basic Protocol 1, but modified so it only uses a portion of the nucleotide query sequence to search a single peptide database sequence. In the Framesearch job corresponding to each BLAST hit, use the Begin and End parameters to specify a region of the nucleotide sequence that brackets the region of nucleotide sequence where BLAST found a match. Instead of giving a wildcard expression that makes it search thousands of protein sequences, specify only the name of the protein sequence from this BLAST hit.

   *For example, from BLAST searches one can determine that nucleotides 52500 through 55000 of Nuc_GCG/gi-15829254_55.seq correspond to all of the amino acid sequence AA_GCG/gi-133795.pep, plus portions of two flanking genes. Therefore, one can use Framesearch to compare only this region of Nuc_GCG/gi-15829254_55.seq with AA_GCG/gi-133795.pep by responding to Framesearch prompts as follows:*

   *when asked Framesearch with what query sequence(s) ?, type Nuc_GCG/gi-15829254_55.seq and hit the return key. When asked Begin (*) ?, type 52500 and when asked End (* 98450 *) ?, type 55000. When asked Search for query in what sequence(s) ? type AA_GCG/gi-133795.pep. Accept default values for all remaining prompts by hitting the return key after each prompt.*

   *Because each of these Framesearch runs is comparing a portion of the nucleotide query sequence to a single protein sequence, instead of comparing the entire nucleotide query sequence to all the protein sequences, this method will take dramatically less time than Basic Protocol 1.*

4. The output files from these Framesearch runs will be very similar to the output files generated by Basic Protocol 1 discussed above. The interpretation of Framesearch output is discussed later in this unit (see Guidelines for Understanding Results).
The major advantage of Framesearch is its facility at aligning protein sequences to nucleotide sequences that contain numerous single-nucleotide insertion or deletion errors. When a nucleotide sequence is compared to a protein sequence, most similarity search algorithms translate all six possible reading frames into protein, then perform a protein sequence comparison (e.g., UNIT 3.4). If a region of nucleotide sequence that is similar to a protein sequence is interrupted by a frameshift error, then what should be one match will become two shorter matches in different reading frames.

However, unless the stretches of nucleotide sequence between single-nucleotide insertions or deletions (indels) are very short, a translated search with BLAST (UNIT 3.4) is likely to find at least part of the alignment, if not the whole stretch of similar sequence. BLAST (UNIT 3.4) will do a typical similarity search several orders of magnitude faster than can Framesearch. Therefore, one can take advantage of the speed offered by BLAST (UNIT 3.4) to find possible hits, then use Framesearch to improve the alignment near each hit.

**Necessary Resources**

**Hardware**

Framesearch can be run on any Unix or VMS system that has the Wisconsin Package installed; because it is so CPU-intensive, Framesearch should be run on the fastest computer available to the user.

**Software**

- GCG Wisconsin Package (v. 8.1 or higher)
- BLAST program (UNIT 3.4)

*In the GCG environment assumed for these examples, both BLAST and Framesearch are included.*

**Files**

- Protein sequence file of interest (this will be the query sequence)
- Nucleic acid database of sequences to which the protein sequence will be compared

*For example, BA000007.fna contains the nucleotide sequence of all putative genes found in this bacterial genome by the laboratory where it was sequenced, as a single FASTA format text file (APPENDIX 1B).*

*Both the query sequence and the database files must be converted to the GCG format (Support Protocol).*

*The files used in this example should be downloaded from NCBI or from the Current Protocols Web site (http://www3.interscience.wiley.com/c_p/cpbi_sampledatafiles.htm) and converted to GCG format, as described in the Support Protocol.*

1. Download the files, set up the GCG environment, and run FROMFASTA on the data files as described in the Support Protocol below.

2. Use your protein sequence as the query in BLAST searches of the appropriate nucleotide database (see UNITS 3.3 & 3.4 for more about BLAST).

*If a BLAST search using the default parameters does not find any matches, it may be necessary to increase the cutoff E-score to a value greater than 1, possibly even as high as 10 or more, in order to obtain at least one match. Note which regions of each nucleotide hit sequence are similar to the protein query sequence.*
3. For each BLAST hit found in step 2, run a Framesearch job as described above in Basic Protocol 1, but modified as follows. First, use the nucleotide sequence from each BLAST alignment as the query sequence, and specify the original protein sequence as the sequence to be searched for matches to that query. This reversal is because GCG Framesearch asks the user which portion of the query sequence to use for searching, so by using each database hit found by BLAST as a Framesearch query sequence, one can restrict the search to a particular region of that nucleotide sequence. When prompted for the Begin and End parameters of the nucleotide sequence, use values that surround the region where the alignment found by BLAST is located. Instead of giving a wildcard expression that makes it search thousands of protein sequences, specify only the name of the protein query sequence for the database to search.

For example, from BLAST searches one can determine that nucleotides 53299 through 54102 of Nuc_GCG/gi-15829254_55.seq correspond to the amino acid sequence AA_GCG/gi-133795.pep. Therefore, one can use Framesearch to compare only a region surrounding this portion of Nuc_GCG/gi-15829254_55.seq with AA_GCG/gi-133795.pep by responding to Framesearch prompts as follows:

When asked Framesearch with what query sequence(s)?, type Nuc_GCG/gi-15829254_55.seq and hit the return key. When asked Begin (* 1 *)?, type 52500 and when asked End (* 98450 *)?, type 55000. When asked Search for query in what sequence(s)?, type AA_GCG/gi-133795.pep. Accept default values for all remaining prompts by hitting the return key after each prompt. Note that although the initial query sequence used for the BLAST searches was a protein sequence, for Framesearch the nucleotide sequence is used as the query sequence, so that one has an opportunity to specify the nucleotide region of interest.

Because each of these Framesearch runs is comparing a portion of a single nucleotide database sequence to the protein sequence, instead of comparing the protein query sequence to the entire length of all nucleotide sequences in the database, this method will take dramatically less time than Basic Protocol 2.

4. The output files from these Framesearch runs will be very similar to the output files generated by Basic Protocol 1 discussed above. The interpretation of Framesearch output is discussed later in this unit (see Guidelines for Understanding Results).

**ALTERNATE PROTOCOL 3**

**IMPROVING SPEED OF FRAMESearch BY USING SPECIALIZED HARDWARE**

While Alternate Protocols 1 and 2 make practical the routine use of Framesearch on an ordinary computer, it is not a perfect solution to the Framesearch performance problem. Restricting Framesearch to regions surrounding BLAST hits will permit alignments that are interrupted by frameshift errors to span reading frames, thus converting a short BLAST hit into a longer Framesearch hit. However, there may be hits that BLAST (UNIT 3.3 & 3.4) will not find at all, because no single reading frame has enough sequence identity for BLAST to find.

Several companies sell specialized computing hardware that can run Framesearch several orders of magnitude faster than can a typical general-purpose CPU, thus making it practical to run a full Framesearch on large amounts of sequence data.

Special-purpose genomics computing systems from several companies, including Compugen (http://www.cgen.com), Paracel (http://www.paracel.com), and TimeLogic (http://www.timelogic.com), use massively parallel arrays of special processors to perform such CPU-intensive algorithms as Smith-Waterman, Framesearch, and hidden Markov models at speeds that bring these rigorous dynamic programming algorithms into the realm of practical daily use. For example, while the Framesearch run that generated
Figures 3.2.3 and 3.2.4 took more than 32 hours of CPU time on a large Unix server, the Framesearch runs that generated Figures 3.2.7 through 3.2.9 took only a few minutes each on a TimeLogic machine. Similar machines from Compugen and Paracel also run Framesearch at speeds that make it practical for interactive use.

The basic steps involved in using Framesearch on specialized genomic computing systems are very similar to those described in the Basic and Support Protocols of this unit: download the sequences, copy the database onto the special-purpose computer, select the algorithm and search parameters, and run the search. However, since the details are different for each brand of special-purpose computer, it is not possible to describe them here. Readers who have access to such hardware should refer to the documentation that comes with it.

Fortunately, all three vendors listed above include Web interfaces with their products. Therefore, these devices are quite simple to use for single searches. Paste the query sequence into a Web page, select the various options, and click a submit button.
In order to run Framesearch, the nucleotide query sequence and the protein database files must be in GCG format. This protocol uses the program FROMFASTA to convert FASTA files to GCG files that may subsequently be used in the Basic and Alternate Protocols.

Necessary Resources

Hardware

Any Unix or VMS system that has the Wisconsin Package installed

Software

GCG Wisconsin Package (v. 8.1 or higher; includes FROMFASTA)

Files

The files used in this example can be downloaded from the NCBI FTP server as described below, or from the Current Protocols Web site (http://www3.interscience.wiley.com/c_p/cpbi_sampledatafiles.htm)
1. Using any FTP client or Web browser, connect to the following directory on the NCBI server:


This directory contains downloadable data from the recently-sequenced genome of the O157:H7 strain of E. coli. Readers who may have previously used the old NCBI FTP server address ftp.ncbi.nlm.nih.gov should be aware that the official address of the NCBI FTP server is now ftp.ncbi.nih.gov although the old address (including the nlm subdomain) still works.

2. From this directory, download, into any convenient working directory where the user has write access, the two files needed for this example:

BA000007.faa

This contains the amino acid translations of all putative genes found in this bacterial genome by the laboratory where it was sequenced, as a single FASTA format text file (APPENDIX 1B).
BA000007.fna

This file contains the nucleotide sequence of this bacterial genome, again as a single FASTA text file (APPENDIX 1B).

3. In the directory where the two downloaded files have been placed, create two subdirectories called “AA_GCG” and “Nuc_GCG.” While this step is not essential, it will be much more convenient to put these files into their own subdirectories before running FROMFASTA, since the FROMFASTA steps below will create thousands of individual GCG-format sequence files.

4. Initialize the GCG environment. The exact commands may vary. For example:

```
source /gcg/gcgstartup
gcg
```

5. Copy or move the FASTA file BA000007.faa to the “AA_GCG” subdirectory and put a copy of the FASTA file BA000007.fna into the “Nuc_GCG” subdirectory (APPENDIX IC).

6. Type the following commands to generate the GCG-format files needed for later steps:

```
    cd AA_GCG
    FROMFASTA
```

and the FROMFASTA program will ask FROMFASTA of what FASTA sequence file(s)? Type the name of the amino acid file to be converted:

```
    BA000007.faa
```

and hit the return key.

Then wait while FROMFASTA converts this FASTA file into 5361 individual GCG-format peptide files with names ending in “.pep” (FROMFASTA will correctly determine that these sequences are peptides).

7. To convert the nucleotide sequence file, type the following commands:

```
    cd ../Nuc_GCG/
    FROMFASTA
```

the FROMFASTA program will ask FROMFASTA of what FASTA sequence file(s)? Type the name of the nucleotide file to be converted:

```
    BA000007.fna
```

and hit the return key.

Wait while FROMFASTA converts this FASTA file into GCG sequence format; FROMFASTA will correctly recognize that the single sequence in this FASTA file is a nucleotide sequence. Because this nucleotide sequence is fairly long, being an entire genome, FROMFASTA will convert it into 55 overlapping chunks of 110000 bp, with an overlap of 10000 bp. (The filenames will be gi-15829254_1.seq, gi-15829254_2.seq, gi-15829254_3.seq, and so on through gi-15829254_54.seq and gi-15829254_55.seq.).

8. Change the working directory back to the starting directory by typing:

```
    cd ..
```

Now that the files have been converted into GCG format, the Protocols above can be performed on these sequences.
GUIDELINES FOR UNDERSTANDING RESULTS

The output from Framesearch begins with a list of hits similar to that shown in Figure 3.2.3, sorted by the quality score. For each of the top hits there is then an alignment section as shown in Figure 3.2.4, in which the alignment is preceded by some general statistics that summarize this match. The TimeLogic implementation of Framesearch adds “P-Scores,” statistical estimates of the probability that a given match could have occurred by chance.

By default, the GCG version of Framesearch will return the top 40 matches, ranked by the quality score. This can be changed using the parameters –LISTsize which controls the number of hits appearing in the list and –ALIgn which controls the number of complete alignments displayed. Other Framesearch implementations have different ways of specifying how many hits to show; for example on a TimeLogic DeCypher system one can specify a cutoff quality score or statistical probability score as well as the maximum numbers of scores and alignments to present.

There are a number of things one should consider in judging whether the degree of sequence similarity presented by a given pairwise alignment does or does not support the hypothesis that these two sequences are in fact homologous. These include the distribution of quality scores in the list, the P-Score value if one was computed, the distribution of scores in the histogram if one was printed, and the percentages of similarity and identity computed by the program. In the author’s experience, when a Framesearch alignment is generated on a TimeLogic DeCypher system the best overall indicator of biological significance is the P-Score. Hits with P-Score values of less than about 1.0E-05 are highly likely to represent valid matches, while hits with P-Score values of more than about 0.05 are questionable. When the P-Score falls in between these limits, assessing the validity of a hit is a judgment call; there is no hard-and-fast rule.

Since the GCG version of Framesearch does not compute a statistical probability estimate, determining the validity of a Framesearch hit generated by the GCG implementation is somewhat more difficult. One must therefore look through the hits to judge whether there is a point after which the hits are of dramatically lower quality than those with higher scores, rather as a teacher might look over a list of test scores to judge which are close enough to the top to merit the grade of “A.” For instance, in the example given for Basic Protocol 2 above, shown in Figure 3.2.5, an E. coli protein sequence is compared with all nucleotide sequences from the same genome. The top hit has a quality score of 2597, and the alignment shown in Figure 3.2.6 shows that this hit is 100% identical. However, the second hit in this search has a quality score of just 66. Therefore, it would seem likely that only the first hit in this example is valid.

On the other hand, in the example given for Basic Protocol 1 above, where a relatively large stretch of E. coli nucleotide sequence is compared with all proteins identified in that genome, the highest-scoring hit (as shown in Figure 3.2.3) has a score of 5450, but a number of other hits have scores of over 1800. In this case, probably most or even all of the 40 hits given by Framesearch are biologically valid. In biological terms, Framesearch has found the genes contained in the genomic region used, plus, possibly, genes that are homologous to those genes.

Keep in mind however that comparing finished genomic sequence data against a database of protein sequences identified in that very genome is a somewhat artificial example. Frequently, in practice, one does not see such a dramatic difference between the top hit and the second hit as was seen in Figure 3.2.5. More commonly, one sees a small number of hits with high-quality scores and percentage identity values, followed by a more hits
with somewhat lower-quality scores, followed by a large number of hits with much lower-quality scores. Figure 3.2.2 shows such an example.

Figure 3.2.2 depicts the score distribution generated by using Framesearch to compare nucleotides 52500 through 55000 of Nuc_GCG/gi-15829254_55.seq against all the example peptide sequences. That is, in step 3 of Alternate Protocols 1 and 2 above, instead of specifying a single sequence as described there in response to the prompt Search for query in what sequence(s)? one can generate this score distribution by typing AA_GCG/*.pep thus doing the search against 5,301 amino acid sequences instead of against only one amino acid sequence. Since the selected region comprises all of one gene and parts of two flanking genes, there are three very strong hits with scores of 978, 1154, and 1391, highlighted by arrows in Figure 3.2.2. There are also many lower-quality hits with scores below 400. From this histogram, it seems likely that hits with scores above 200 might represent genes related to the three genes contained in this region, while hits with scores below 100 probably do not represent biologically significant matches. The validity of hits with scores between 100 and 200 would need to be assessed on a case-by-case basis.

Numbers computed by programs do not replace human judgment. With experience, one develops an intuitive feeling for the overall “shape” of pairwise alignments. It is also important to look at the annotations of the database entries that match the query sequence. Often one gains more insight from reading the descriptions of multiple hits than one can gain from reading the descriptions of the top one or two hits. For instance, if the top few hits are described in words that relate to very similar biological concepts, but the next few hits have very different descriptions, then one should be somewhat more skeptical of the latter alignments. On the other hand, if there are several strong matches whose descriptions do sound plausible and consistent, but among them is one match with a very different description, then one might suspect that description could be incorrect. One may also have an expectation based on the source of the query sequence. Unfortunately, given the variable quality of annotations found in most large genetic sequence databases, interpreting the hits from similarity search algorithms is more of an art than an exact science.

COMMENTARY

Background Information

Advantages of Framesearch

A nucleotide sequence can be compared at the nucleotide level with nucleotide databases, using the BLAST, FASTA, or SSEARCH algorithms (UNITS 3.3 & 3.4). However, because protein sequences are usually of greater biological significance, a translated search against a protein database is often much more informative. If the nucleotide sequence is of high quality, then good results can be obtained by translating the entire nucleotide sequence in all six reading frames and using a standard protein versus protein search algorithm (e.g., UNIT 3.4). But if the nucleotide sequence is of low quality, particularly if it contains many single-nucleotide insertion or deletion errors, then a six-frame-translated search may not work very well. Comparing low-quality nucleotide sequences to protein sequences is the forte of Framesearch.

When six-frame-translated searches are used with nucleotide sequences that contain many indels, every single-nucleotide indel creates a frameshift. If a region of nucleotide sequence that is similar to a protein sequence contains one or more frameshift errors, then what should be a single alignment becomes multiple shorter alignments with any six-frame-translated search algorithm. Figure 3.2.7 shows how indels can prevent the six-frame-translated Smith-Waterman algorithm from finding more than a small part of the region of sequence similarity because none of the six translations has very long regions of good alignment.

Framesearch does not work by generating the six possible translations of each nucleotide sequence. Instead, it translates the nucleotide sequences on the fly, taking into account all possible frameshifts as well as the amino acid insertions, deletions, and mismatches consid-
ered by the Smith-Waterman algorithm. It assumes the protein sequence is accurate, and dynamically inserts or deletes single nucleotides as needed to obtain the best possible alignment between the peptide sequence and the protein sequence. This often means that Framesearch can obtain much longer and better alignments between low-quality nucleotide sequences and protein databases than is possible with a six-frame-translated algorithm. Figures 3.2.8 and 3.2.9 show how, given the same input data as used with the Smith-Waterman algorithm to generate Figure 3.2.7, Framesearch can still align nearly the entire length of the protein sequence with the corresponding nucleotide sequence, even though five frameshift errors have been introduced into the nucleotide sequence.

The author of this unit has found Framesearch to be especially valuable when searching for possible homologs of known genes in two types of nucleotide sequence databases: human EST sequences and unfinished microbial genomic sequences. Unfortunately, this power comes at a considerable cost in CPU time, because Framesearch is very slow when run on a general purpose computer.

**Motivation behind Framesearch**

The Framesearch algorithm was first presented in a poster (Edelman et al., 1995) describing the joint work of researchers at GCG in Wisconsin and Compugen in Israel. Its development was motivated by the observations that: (1) EST sequences had roughly 10 times as many indels as did the rest of GenBank at that time; (2) about half of all sequence errors in GenBank at that time were indels; and (3) nucleotide versus nucleotide sequence comparisons tended to be much less biologically meaningful than protein versus protein sequence comparisons (GCG, 1995). Furthermore, by 1995 it was clear that in the very near future large amounts of genomic sequence data would soon become available, and “searching a protein database with a translated nucleotide query sequence offers important insights in nucleotide sequencing projects, even in the early stages, when sequence data may be particularly error prone.” However, the authors noted, six-frame translated searches “have difficulty dealing with sequencing errors, particularly insertion and deletion errors in nucleotide sequences. Thus, significant database similarities may go undetected” (GCG, 1995).

**The algorithm**

The theory of Framesearch is very similar to that of the classic Smith-Waterman algorithm. The translated Smith-Waterman search generates the six possible translations of the entire nucleotide query sequence, then attempts to align each translation with each peptide sequence in the database. Thus, if there is a frameshift error in the middle of an alignment, a six-frame-translated search algorithm generally cannot find the entire alignment because it considers only one reading frame at a time. With both a Smith-Waterman search and Framesearch, amino acid substitutions are scored using a scoring matrix (UNIT 3.5), and gaps of one or more amino acids in the query or database sequence are scored using affine gap penalty parameters. Framesearch scores amino acid substitutions and whole-codon gaps much as the Smith-Waterman algorithm does, and uses a similar dynamic programming matrix (UNIT 3.1) to search the entire space of possible alignments. However, instead of translating the entire nucleotide query sequence in each of six reading frames, Framesearch inserts or deletes individual nucleotides in order to generate the best possible alignment even when that alignment crosses a reading frame shift. This represents a dramatic expansion of the search space that must be considered, which is why Framesearch is even more CPU-intensive than a Smith-Waterman search.

Like the Smith-Waterman algorithm and heuristic algorithms based on Smith-Waterman such as BLAST (UNITS 3.3 & 3.4) and FASTA, Framesearch assumes that the probability of a given amino acid being found at a given position is independent of its position along the sequence. By contrast, profile-based search algorithms, such as hidden Markov models, profile searches, and PSI-BLAST, take into account position-specific probability distributions.

BLAST (UNITS 3.3 & 3.4) and FASTA are known as “heuristic” algorithms because they use methods that work well in the vast majority of cases to find regions of possible alignment, which are then improved and possibly extended by exact dynamic programming searches (UNIT 3.1) that are restricted to those regions. This makes BLAST (UNITS 3.3 & 3.4) and FASTA much less CPU-intensive than the Smith-Waterman and Framesearch algorithms, at some cost in sensitivity.

All search algorithms that are based on Smith-Waterman, including BLAST (UNITS 3.3 & 3.4) and FASTA, and profile searches, use methods that work well in the vast majority of cases to find regions of possible alignment, which are then improved and possibly extended by exact dynamic programming searches (UNIT 3.1) that are restricted to those regions.
Finding Homologs Using the Framesearch Program

3.2.20

& 3.4), FASTA, and Framesearch, share the limitation that they compute similarity scores based only on the probability that, for instance, a typical cysteine residue would mutate to a phenylalanine residue in the course of evolution. That is, the only source of biological knowledge considered by such algorithms is encoded in the scoring matrix, so any Cys-Phe substitution will be given the same weight in scoring of potential alignments.

However, when groups of related proteins are compared, normally one finds that some residues are much more highly conserved. This generally indicates that these residues are for some reason important to the biological function of these proteins. Scoring matrices of the type used by Smith-Waterman and related algorithms consider only the overall frequency with which a given amino acid is replaced by another, not the fact that, for instance, one cysteine might be in a binding pocket while another cysteine is in a position of less functional importance.

In a profile-based method, a multiple alignment of related sequences is used to build a profile, a statistical summary of the multiple alignment that is then compared against a database in a search for additional sequences that might also be related to the input sequences. Since this profile is based on multiple sequences, it can take into account the fact that certain positions are more highly conserved among the input sequences summarized by that profile, and thus are possibly of greater functional importance than other positions where there is more variation among the input sequences. Some gene-finding programs combine HMM-based methods with Framesearch-like dynamic programming to search nucleotide-sequence databases for potential exons; as one might imagine, such programs are capable of consuming remarkable amounts of CPU time.

Other options for similar analysis

In this unit, we present as an alternative to a full Framesearch the option of first doing a BLAST search and then using Framesearch to improve and extend the alignments in a region surrounding each BLAST hit (Alternate Protocol 1 and 2). Yet another way to search for protein sequences that are similar to a nucleotide sequence that contains many single-nucleotide indels is to use FASTX or FASTY (Zhang et al., 1997). Note that FASTX and FASTY take nucleotide query sequences and search amino acid databases. To compare an amino acid query sequence against a nucleotide database, use the TFASTX and TFASTY programs. These recent additions to the FASTA family of heuristic search programs are to classic FASTA what Framesearch is to classic Smith-Waterman. While FASTX and FASTY are a good deal slower than six-frame-translated BLAST (UNIT 3.4) or FASTA, they are considerably faster than Framesearch.

Both the FASTX program and the FASTY program allow for frameshifts in the nucleotide query sequence. However, the FASTX program only considers frameshifts between codons while the FASTY program also considers frameshifts within codons. Thus, FASTX is significantly more CPU-intensive than is FASTY, but produces better alignments with poor quality nucleotide sequences. In some cases, FASTX and FASTY may find hits that BLAST does not detect. Thus, if a BLAST search (UNIT 3.4) does not find any good hits, and dedicated hardware for accelerated Framesearch is not available, one might try FASTX or FASTY to find possible hits and then attempt to improve any alignments found with Framesearch as in Alternate Protocol 1 described above.

Recently (Halperin et al., 1999), Compugen has invented a new algorithm called Frameplus, a further extension of Framesearch, which they claim offers improved sensitivity when the nucleotide sequence data may contain both single-nucleotide indels and longer deletions. Code for a software implementation of this new algorithm is available on the Compugen Web site, and the company’s bioinformatics products now include a hardware-accelerated version of Frameplus. The author of this unit has no experience with Frameplus, but users who have access to Frameplus may wish to try it on their data.

TimeLogic bioinformatics computers offer two proprietary algorithms that may be of particular interest to users who work with EST or unfinished genomic sequence data, known as “Semi-global Smith-Waterman” and “Symmetric Frame Independent” (TimeLogic, 2001). Both of these algorithms are designed for comparing a low-quality nucleotide query sequence against a database of low-quality nucleotide sequences.

Semi-global Smith-Waterman is a modification of classic Smith-Waterman, representing a compromise between global (Needleman-Wunsch) and local (Smith-Waterman) pairwise sequence alignment. The semi-global algorithm does a local alignment, but it will prefer alignments that fall near one end of either of
the two nucleotide sequences being aligned. This can help to assemble fragments of low-quality sequence into contigs.

Symmetric Frame Independent search is an extension of Framesearch that compares two nucleotide sequences, dynamically translated into protein, with allowance for frameshifts in either sequence. The reader should be aware that SFI takes almost as long on accelerated hardware as Framesearch does on conventional hardware.

**Critical Parameters and Troubleshooting**

This section should be read in conjunction with the documentation of the specific Framesearch implementation being used. While the GCG version of Framesearch offers more optional algorithmic parameters than most others, any version of Framesearch should support the critical parameters reviewed here.

As with any bioinformatics algorithm, the choice of Framesearch options is something of an art. While most implementations supply reasonable defaults thought suitable for general use, these should be considered a starting point to be adjusted as needed. A major benefit from using special-purpose genomic computing hardware, if available, is that when Framesearch takes minutes instead of hours it is practical to experiment, trying different combinations of parameters to see their effect on results.

**Scoring matrix selection**

Probably the single most important parameter for getting good results with the Framesearch algorithm is the choice of scoring matrix (UNIT 3.5). A scoring matrix is based on specific assumptions about how amino acids change in the evolutionary process. Typically, such matrices will come in a series, such as the PAM series, the BLOSUM series, or the less-widely-used GONNET series and the other specialized matrices. Each member of such a series of matrices is given a number related to the degree of evolutionary distance between sequences for which that matrix is designed. A search for close relatives of the query sequence should use a matrix designed for finding close relatives, while a search for more-distant relatives of the query sequence should use a matrix designed for sequences with a greater amount of divergence. Note that with some matrices, a larger number indicates greater assumed sequence similarity, while with other matrices a larger number indicates greater assumed evolutionary distance, so one must read the documentation for the matrix series being used with close attention.

UNIT 3.5 provides a detailed discussion of how to select an appropriate scoring matrix. In many cases, it is a very good idea to try a search several times, specifying a different scoring matrix each time, and then compare the results.

**Selecting the coding strand**

If the nucleotide query sequence is known to represent the coding strand, the search will take half as long if Framesearch is told to use only the forward strand of the query sequence, and not the reverse complement of the query sequence as well. If the coding direction is not known a priori, translated BLAST searches may provide a good indication.

For the GCG version of Framesearch, the -ONEstrand parameter causes Framesearch to assume the given nucleotide sequence represents the coding strand. In the GCG implementation of Framesearch, one cannot specify a similar restriction when using a protein query sequence to search a nucleotide database. Therefore, when using a protein query sequence, if one wishes to search only one strand of the nucleotide database, one must resort to a procedure similar to that of Alternate Protocol 2, above. By contrast, the Timelogic implementation of Framesearch found on their DeCypher systems has a much more flexible provision for specifying which strand to search.

**Setting the frameshift penalty**

The default frameshift penalty for the GCG version of Framesearch is zero. According to the GCG manual, selecting a nonzero frameshift penalty makes the search take significantly longer, but rarely yields significant benefits in exchange for that cost in CPU time. When special-purpose computing hardware is available, it may be appropriate to try a range of values for the frameshift penalty.

**Setting the gap penalty**

As with the classic Smith-Waterman algorithm of which it is an extension, Framesearch includes options for gap creation and extension penalties. To discourage gaps, use values for both gap parameters larger than the program defaults. To find regions of high similarity that are separated by long gaps, use a smaller value for the gap extension penalty. In addition to the classic affine gap penalty parameters, the GCG version of Framesearch offers an optional
maximum gap penalty. If this optional parameter is set, then very long gaps will never be penalized by more than this value. The GCG manual says this option can be used to find matches that are interrupted by introns, but cautions that setting a maximal gap penalty increases the time required to run Framesearch.

**Other parameters**

Most Framesearch implementations provide numerous other options as described in their manuals. The GCG version of Framesearch includes many parameters that control internal details of how the algorithm generates pairwise alignments; for general use these should probably be left at their default values. While the TimeLogic version of Framesearch offers somewhat less scope for fine-tuning the internal behavior of its alignment algorithm than does the GCG version, the TimeLogic version offers far more scope for changing the format of its output than does the GCG version. The Compugen extension of Framesearch, Frameplus, has additional gap penalty parameters because it accounts for two different classes of gaps in the nucleotide sequence, single-nucleotide indels and longer deletions.

Before experimenting with the more exotic parameters of Framesearch, it is best to gain experience with the effects of changing the basic options such as matrix choice and gap penalties.

**Suggestions for Further Analysis**

**Nucleic acid query sequence**

After using Framesearch to find one or more nucleotide sequences with some degree of similarity to a protein query sequence, the logical next step is to perform a multiple sequence alignment of the protein query sequence with translations of the top nucleotide hits found by Framesearch, because a multiple alignment is often far more informative than a single pairwise alignment. When a TimeLogic system is used to perform a Framesearch query, one can readily obtain translations of the nucleotide hits as determined by Framesearch, taking into account all the indels identified by comparison with protein sequence. If the user has access to such a system, then these translations, plus the protein query sequence, can be run through ClustalW (UNIT 2.3) or GCG PileUp UNIT 3.6 to create a multiple sequence alignment.

**Amino acid query sequence**

After using Framesearch to find one or more nucleotide sequences with some degree of similarity to a protein query sequence, the logical next step is to perform a multiple sequence alignment of the protein query sequence with translations of the top nucleotide hits found by Framesearch, because a multiple alignment is often far more informative than a single pairwise alignment. When a TimeLogic system is used to perform a Framesearch query, one can readily obtain translations of the nucleotide hits as determined by Framesearch, taking into account all the indels identified by comparison with protein sequence. If the user has access to such a system, then these translations, plus the protein query sequence, can be run through ClustalW (UNIT 2.3) or GCG PileUp UNIT 3.6 to create a multiple sequence alignment.

**Either type of query sequence**

Unfortunately, the GCG version of Framesearch does not provide a translation of the nucleotide sequence in a Framesearch alignment. Therefore the GCG Framesearch user will need to use Seqedit to extract the relevant portion of the nucleotide sequence from each alignment, then feed that to GCG Pepdata to generate its six-frame translations, and run these translations plus the protein sequence through a multiple alignment program to get an idea what the best translations may be.

Comparing this multiple alignment with the pairwise alignments generated by Framesearch can often give the user an excellent idea where the nucleotide sequence is likely to contain indels, where the nucleotide sequence is of relatively high quality, and where there may be problems in the nucleotide sequence data. In sum, neither Framesearch nor any other analysis can magically convert low-quality nucleotide sequence data into high-quality data. But, when used with care, and especially when used in conjunction with multiple alignments, the Framesearch algorithm can sometimes extract value from poor sequence data where other methods fail to deliver.

**Literature Cited**


Sequence and analysis Conference, Hilton Head, South Carolina, 1995.

NOTE: The text of this poster can be found at http://sulu.gcg.com/company/posters/framesearch.html.


NOTE: The GCG Transcript, subtitled “Bio-Computing News for Users of the Wisconsin Package,” was published by the company for a number of years. The text of this issue, which features a discussion of the newly-added Framesearch program, can be found at http://sulu.gcg.com/pub/newsletter/vol3_no2_nov95.html.


Key References

Edelman et al., 1995. See above.

The key reference for the Framesearch algorithm is the poster by Edelman. The key reference for a particular implementation of Framesearch is the documentation supplied with that implementation.

Internet Resources

http://www.accelerys.com/
Web site of Accelerys, the corporate parent of GCG.

http://www.cgen.com/
Web site of the Compugen company.

http://www.paracel.com/
Web site of the Paracel company.

http://www.timelogic.com
Web site of the TimeLogic company.

Contributed by Matthew Healy
Bristol-Myers Squibb Pharmaceutical Research Institute
Wallingford, Connecticut
UNIT 3.3

Finding Similar Nucleotide Sequences Using Network BLAST Searches

Istvan Ladunga

1University of Nebraska-Lincoln, Lincoln, Nebraska

ABSTRACT

The Basic Local Alignment Search Tool (BLAST) is a keystone of bioinformatics due to its performance and user-friendliness. Beginner and intermediate users will learn how to design and submit blastn and Megablast searches on the Web pages at the National Center for Biotechnology Information. We map nucleic acid sequences to genomes, find identical or similar mRNA, expressed sequence tag, and noncoding RNA sequences, and run Megablast searches, which are much faster than blastn. Understanding results is assisted by taxonomy reports, genomic views, and multiple alignments. We interpret expected frequency thresholds, biological significance, and statistical significance. Weak hits provide no evidence, but hints for further analyses. We find genes that may code for homologous proteins by translated BLAST. We reduce false positives by filtering out low-complexity regions. Parsed BLAST results can be integrated into analysis pipelines. Links in the output connect to Entrez, PUBMED, structural, sequence, interaction, and expression databases. This facilitates integration with a wide spectrum of biological knowledge. Curr. Protoc. Bioinform. 26:3.3.1-3.3.26. © 2009 by John Wiley & Sons, Inc.

Keywords: BLAST • sequence alignment • database search • homology search • mapping • nucleic acid • DNA • RNA • genome • blastn • Megablast

INTRODUCTION

The Basic Local Alignment Search Tool (BLAST; Altschul et al., 1990; 1997) is the most widely used and productive tool in bioinformatics. BLAST can identify similar sequences at an amazing speed, allowing us to infer the biochemical function, exon boundaries, and domain architecture. We recommend good BLAST practices for beginners and intermediate users, address the over- or under-interpretation of results, and demonstrate how to add further information from diverse databases. Just running a BLAST search is easy and can still be of great service to researchers, but fine-tuning the parameters of the program allows us to detect more distant relatives and reduce the vast amount of output, saving us time spent on visual analyses. For diverged sequences, we use the slower blastn program, while closely similar matches are identified much faster by Megablast. Both can map expressed sequence tags (ESTs) or other sequencing reads to a genome or the transcriptome. blastn and Megablast can also identify short, similar sequences in other mRNAs during primer design, or target selection in RNA interference. BLAST has been integrated into network applications and comprehensive bioinformatics packages. Among these, network BLAST creates visual displays of similar sequences linked to literature, structural, gene expression, pathology databases, and ontologies. These links help us to resolve issues like missing or contradicting annotations, low-scoring alignments, or sequences obfuscated by frameshift mutations or sequencing errors. Here, we discuss nucleotide BLAST in six protocols. In Basic Protocol 1, we submit blastn searches on remote servers (output results shown in Basic Protocol 2). Since it would be naive to believe that the default parameter values are acceptable for all the specific applications and gene families, in Support Protocol 1, we select a subset of the database, set filters to avoid a mass of partly meaningless results, and set the expected frequency of hits to be
BASIC PROTOCOL 1

USING THE WEB-INTERFACE BLAST FROM THE NCBI BLAST SERVER FOR NUCLEOTIDE SEQUENCES

NCBI’s user-friendly, high-performance BLAST server site provides the easiest way to run a BLAST search (http://blast.ncbi.nlm.nih.gov; Fig. 3.3.1—please note the new URL). NCBI has been offering this service since 1992 (Johnson et al., 2008).

Necessary Resources

Hardware

Any Internet-connected computer

Software

Web browser (Mozilla Firefox, Internet Explorer, Safari, or Opera)

Files

If an accession number or a gi (Gene Identification) number is available or a DNA query sequence in the FASTA format (see APPENDIX 1B) can be copied from another window, no file is necessary.

NOTE: The example shown here uses a FASTA file of the human let 7c microRNA. The sequence is available from the microRNA database at http://microrna.sanger.ac.uk/cgi-bin/sequences/mirna_entry.pl?acc=MI0000064. Click on “Get sequence.”

Figure 3.3.1 The home page of the NCBI BLAST server (http://blast.ncbi.nlm.nih.gov).
Prepare for a basic BLAST search

1. Point the browser to the address http://blast.ncbi.nlm.nih.gov and select “nucleotide blast.” Once this is done, the title should read Basic Local Alignment Search Tool (Fig. 3.3.2). This page is organized into five sections: Enter Query Sequence, Choose Search Set, Program Selection, (submit) BLAST, and Algorithm parameters (as of April, 2009).

   In addition to nucleotide searches, protein and translated searches (UNIT 3.4) can also be performed starting from the BLAST page.

2. Copy and paste the query sequence—for now, the human let 7c microRNA above—into the Search text area in FASTA format (see Fig. 3.3.2).

   In the FASTA format (see APPENDIX 1B), the first line starts with a greater than sign “>”, which is immediately followed by an identifier, then by optional comments separated by one or more spaces. The second and further lines contain the sequence. Only the standard IUPAC abbreviations can be entered (APPENDIX 1A). Within the sequence, the NCBI server allows using spaces and numbers in contrast to some other servers; however,

Figure 3.3.2  The basic search screen for nucleic acid BLAST at NCBI. The title line and the sequence of the human let-7c microRNA in FASTA-format were pasted into the search field.
Table 3.3.1 Nucleotide Sequence Databases at the NCBI as of April 2009

<table>
<thead>
<tr>
<th>Database Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human genomic plus transcript (Human G+T)</td>
</tr>
<tr>
<td>Mouse genomic plus transcript (Mouse G+T)</td>
</tr>
<tr>
<td>Nucleotide collection (nr/nt)</td>
</tr>
<tr>
<td>Reference mRNA sequences (refseq_rna)</td>
</tr>
<tr>
<td>Reference genomic sequences (refseq_genomic)</td>
</tr>
<tr>
<td>NCBI Genomes (chromosome)</td>
</tr>
<tr>
<td>Expressed Sequence Tags (est)</td>
</tr>
<tr>
<td>Non-human, non-mouse ESTs (est_others)</td>
</tr>
<tr>
<td>Genomic survey sequences (gss)</td>
</tr>
<tr>
<td>High-throughput genomic sequences (HTGS)</td>
</tr>
<tr>
<td>Patent sequences (pat)</td>
</tr>
<tr>
<td>Protein Data Bank (PDB)</td>
</tr>
<tr>
<td>Human ALU repeat elements (human_alu)</td>
</tr>
<tr>
<td>Sequence tagged sites (dbsts)</td>
</tr>
<tr>
<td>Whole-genome shotgun reads (wgs)</td>
</tr>
<tr>
<td>Environmental samples (env_nt)</td>
</tr>
</tbody>
</table>

nonalphanumeric characters—with the exception of white space, (-) for gaps, and asterisks (*) for stop codons—may cause problems.

When using NCBI's Entrez system (http://www.ncbi.nlm.nih.gov/Entrez) to retrieve the sequence in the FASTA format, one has to remove additional information. After searching GenBank for the appropriate record, mark only one sequence by checking the box next to the sequence identifiers. From the Display menu, select FASTA and click on Text and again on Display. Now the frame will display only the definition line and the sequence, which makes it easier to copy and paste the FASTA sequence into the BLAST search window.

3. Set the “Query subrange” if necessary. For now, leave these two fields empty.

4. Change the machine-assigned Job Title, if necessary, to obtain a unique identifier for each search.

5. Select the database to be searched (Table 3.3.1). First check the “Others (nr etc.):” button, then from the drop-down menu, choose NCBI Genomes (chromosomes).

Multiple databases cannot be queried in a single run on the NCBI server, in contrast to command-line BLAST. Also, note that this server does not host a number of publicly accessible databases. To search such databases, these need to be installed on a local computer along with the standalone BLAST version.

6. In the Program Selection menu, choose “Somewhat similar sequences (blastn).” This will provide more sensitive searches at a price of slower computing speed.

7. Select Show results in a new window.

8. At this point, the user can click on the BLAST button; however, this is the bare minimal protocol, which is only acceptable on the first day of learning BLAST. It would be extremely naive to think that the BLAST default parameters provide maximum sensitivity and selectivity for all applications. To improve performance, one has to select from the options available for advanced BLAST (see Support Protocol 1) and Formatting (see Support Protocol 2).

See Basic Protocol 2 for information on the default blastn result output.
THE DEFAULT BLASTN RESULT OUTPUT

The NCBI servers typically return the results in a few minutes, but at peak hours, users have to wait longer. The content of basic blastn output (Fig. 3.3.3) is enriched with information on conserved domains, the distribution of hits along the query sequence, color-coded cross-references to other databases, and anchors to facilitate jumping from one-line descriptions to the actual pairwise alignments. Here, we describe the designs as of April 2009, with the caveat that the organization of the HTML output is constantly evolving. In order to accommodate the needs of specific applications, users can specify their format requirements by clicking on the link “Reformat These Results” close to the top center of the screen (see Support Protocol 2).

The default output is organized into the following sections: Job identifier, reference to the current BLAST publication, database parameters, when searching a single genome, a clickable icon to display the Genome View (Fig. 3.3.14), graphical representation of the alignments, distance tree of results, related structures one-line descriptions, alignments, and statistics.

1. Administrative section (self-explanatory) contains the following information:
   a. The Request ID number (advanced) can be used to retrieve results within a 24-hour period. Go to the BLAST page (http://blast.ncbi.nlm.nih.gov), click on Recent Results for a Request ID link just below the BLAST logo (Fig. 3.3.2, top) in order to retrieve results sorted by either submission time, request ID, status, program, title, query length, database, or expiration (Fig. 3.3.3A). Having a free NCBI account, allows, among many other benefits, storing search strategies for easy modification and resubmission.
   b. Search summary (advanced; not shown) displays the query and database summary, the BLAST version and citation, the Karlin-Altschul statistics, and the effective size of the search space (Karlin and Altschul, 1990). See also the Baxevanis (2005) and Korf et al. (2003) tutorials on BLAST.
   c. Taxonomy reports (advanced; not shown). If hits are found to multiple species, the user can display their distribution according to taxonomical categories, a service that can ease finding similar proteins in diverse organisms.

2. Distribution of <number> Blast Hits on the Query Sequence (Fig. 3.3.3.A) is an interactive graphical display of the alignments color-coded by score. Positioning the cursor over a colored line displays the brief one-line description of the database sequence. Partial alignments may indicate alternative splicing, exons coding for protein domains, or mosaic (chimeric) genes.

3. Distance tree of results (advanced; not shown) generates a tree based on sequence similarity, unlike the Taxonomy Reports discussed above. Since these trees are generated by pairwise alignments and no phylogenetic corrections are performed, we cannot call them bona fide phylogenetic trees. For the construction of phylogenetic trees, see Chapter 6.

4. Related Structures (if available, advanced) displays the hits to NCBI’s Molecular Modeling Database (Wang et al., 2007). Thanks to the rapid development in structural biology, a significant percentage of proteins have some distant relatives with known three-dimensional structures that can provide inference for the structure or function of the query protein. Links are supplied to the structures in the Protein Data Bank (Berman et al., 2000) where structures can be displayed interactively by a repertoire of structure viewer software (see UNIT 1.3).
Finding Similar Nucleotide Sequences Using Network BLAST Searches

3.3.6

Figure 3.3.3  (legend on following page)
5. Legend for links to other resources: “U” UniGene, “E” GEO, “G” Gene, “S” Structure, and “M” Map Viewer (Fig. 3.3.3B). Here, UniGene is a nonredundant database of EST assemblies. GEO (Gene Expression Omnibus) is a database of gene expression experiments. Gene is a highly annotated reference database of genes and proteins. Structure is the Protein Data Bank of three-dimensional structures; and MAP Viewer is a tool to display sequences on chromosomal maps. All of these resources are discussed in Wheeler et al. (2008).

6. One-line descriptions: “Sequences producing significant alignments” (Fig. 3.3.3B). Each line describes the identifiers of the matching database sequence, its short description, the score and the E-value (expected frequency value) of getting this score or higher purely by chance calculated using a semi-random model (see Interpreting the Results below). In the last columns, color-coded one-letter symbols indicate the additional database(s) where the search sequence was found. The sequence identifiers are linked to the respective database entries, and clicking on a score will take us to the pairwise alignment of the query and this database sequence.

7. Pairwise local alignments (Fig. 3.3.4), in contrast to global alignments, do not necessarily cover the whole length of the query and the database sequences. Instead, the High-Scoring Segment Pair(s) (HSPs) are reported. The BLAST algorithm initially looks for “words,” or seeds, 7 to 29 bases long, gapless segments that match identically between the query and the database sequence. BLAST attempts to extend the words in both directions, now allowing mismatches, insertions, and deletions. Words are extended to the point where the cumulative score reaches its maximum based on the scoring matrix and gap penalties selected. Small drops in the cumulative score are allowed, as long as further positively scoring residue pairs allow for a higher cumulative score. Shorter words increase sensitivity at the expense of execution time.

Figure 3.3.4 Pairwise local alignment of the query and the mouse BAC clone RP24-270A10 from chromosome 13. Note that the query matches this clone at two distant locations.

Figure 3.3.3 (on previous page) The results of a blastn search using Basic Protocol 1. (A) Administrative section and the color-coded graphical display of the best hits to the query sequence. (B) One-line descriptions of the database sequences similar to the query with maximal and total scores, total coverage, E-value, maximal percent identity, and links to other databases.
Finding Similar Nucleotide Sequences Using Network BLAST Searches

3.3.8

Supplement 26 Current Protocols in Bioinformatics

Note that a query and a database sequence may have multiple HSPs (alignments). BLAST reports only nonoverlapping alignments. In this search, a number of identical hits are found that cover the entire 84 base length of the query. These are clearly biologically and statistically significant similarities with an E-value of 1e-34 (10^{-34}). For a more challenging example, find on the results page the alignment to AC147162.2 (use Control-F on Firefox or Internet Explorer). This is the mouse BAC clone RP24-270A10 from chromosome 13, which has two matches to the query. Note that the results are grouped by database sequences and sorted by their E-values. When a database sequence has multiple HSPs to the query, sorting is performed according to the lowest (most significant) E-value. As an example, AC147162.2 is sorted by the 1e-08 E-value, not by the E-value of 2.0 for the second HSP. This example will be discussed in the Guidelines for Interpreting the Results section.

a. A checkbox to the left of the identifiers can be checked for sequence retrieval. The identifiers are followed by the short description of the database sequence.

b. Score is the cumulative score for the HSP, in raw and normalized (bitwise) forms.

c. Expect is the expected frequency of the random occurrence of the above score or higher between a sequence, as long as the query and a random database of the same size (see Interpretation of Results).

d. The number of Identities is followed by the total number of letters (nucleic acid bases) in the alignment, and by the percentage of identities.

e. Gaps is the number and percentage of gaps introduced during sequence alignment.

f. The coordinates of the Query and Sbjct (database) sequences for each line.

g. Alignment lines for the Query and the Sbjct sequences, and in the middle, the match between them. Identities are marked by vertical bars (|), mismatches by spaces, and gaps by hyphens (–).

h. Strands (of DNA): one of the following combinations of the query and the search sequence: Plus/Plus, Plus/Minus, and Minus/Plus.

8. Statistics (advanced) at the end of the BLAST report include the number of residues in the database, its effective size, the Karlin-Altschul parameters (Karlin and Altschul, 1990), the length of the query, etc. For the description of this section, we recommend Korf et al.’s (2003) tutorial on BLAST.

SETTING OPTIONAL PARAMETERS

This protocol walks the user through setting the advanced search parameters that are available from the NCBI BLAST Web page (Fig. 3.3.5). Taking the time to set these parameters greatly enhances the sensitivity and selectivity of BLAST and shortens the time necessary to analyze the results.

Necessary Resources

Hardware

Any Internet-connected computer

Software

Web browser (Mozilla Firefox, Internet Explorer, Safari, or Opera)

Files

A file with the DNA query sequence in FASTA format (APPENDIX 1B)

NOTE: The example shown here uses a FASTA file of the human let 7c microRNA. The sequence is available from the microRNA Database at http://microrna.sanger.ac.uk/cgi-bin/sequences/mirna_entry.pl?acc=MI0000064. Click on “Get sequence.”
1. Prepare the search as described in Basic Protocol 1 (steps 1 to 7).

2. If necessary, limit the search to a subset of the chosen database by applying an Entrez query (see UNIT 1.3). For example, to eliminate all mRNAs and limit the search to multicellular animals or (any) fungi or plants, use the Entrez query:

   (metazoa [orgn] OR fungi [orgn] OR plants [orgn]) NOT "chromosome 15"

   The Boolean operators “AND,” “OR,” and “NOT” should be capitalized, otherwise they are interpreted as search terms. Search field abbreviations have to be enclosed by square brackets for the same reason. Note that without the parentheses, the “NOT ‘chromosome 15’” would refer to plants only and would include all metazoan and fungal chromosomes. Before designing complex queries, refer to UNIT 1.3 and http://www.ncbi.nlm.nih.gov/books/bv.fcgi?rid=helpentrez.chapter.EntrezHelp.

3. Click on Algorithm parameters so as to display their options. When a value other than the default is selected, the surroundings of the text field turn yellow (Fig. 3.3.5).

4. Select Max target sequences to 500 since the number of expected target sequences is higher than the default 100.

5. Enter an Expect threshold. This reflects the expected frequency—i.e., the number of database hits with a score equal to or higher than the match between the query and the subject database sequence.

   For further discussion of this parameter, see Critical Parameters below.

   An expected (absolute) frequency of 10 (the default value) means that for the reported score or higher, 10 merely random matches can be expected in the database of given size. This extremely nonconservative level of statistical significance may be necessary for finding short exons or frameshifted translations. In our experience, in order to achieve reliable statistical inference for the common evolutionary origin or function of two sequences, the expectation value should not exceed $1 \times 10^{-10}$ (entered as $1e-10$) so as to compromise between false-positive and false-negative matches. Note that this is considerably higher than the threshold of $10^{-20}$ for amino acid BLAST (UNIT 3.4).
6. Set Word size to the smallest allowed value, 7. To improve speed, the BLAST algorithm initially looks for identical words between two sequences. Once such a word is found, BLAST attempts to extend it to the High-Scoring Segment Pair (HSP). For short and degenerate sequences, the default word length of 14 may result in missing a number of truly similar HSPs. For sequence assembly or novelty checking, increase the word size to 15 in order to gain speed.

7. **Scoring parameters (advanced):** The quasi-optimal selection of these parameters depends on the target frequency of identical nucleotide pairs in an alignment. The default $+2/-3$ match/mismatch scores, with gap opening and extension costs of 5 and 2, work best for a target frequency of 90%. For a target frequency of 75%, the $+1/-1$ scores are shown to be the best. For a detailed overview, see APPENDIX B in Korf et al. (2003). For the general theory of scoring matrices and gap penalty values, see Altschul (1991).

8. In order to avoid masking low-complexity regions in the query and the database sequences, uncheck the boxes for all filtering parameters. Multiple filters may be applied simultaneously. For mapping a sequence shorter than a few most applications, filtering may lead to skipping some good hits. However, when the query sequence is either a long mRNA, EST, or genomic sequence, leave the “Low-complexity regions” filtering checked. In addition, unless assembling long sequences, turn on the Species-specific repeats filter and select the appropriate species from the pull-down menu. By default, filtering is only applied to the query sequence (or its translation products), not to database sequences. The user can override that by unchecking the Mask for lookup table only parameter. Mask lowercase letters can be used to ignore certain regions in the query sequence, like abundant domains that are present in unrelated sequences. In order to lowercase those segments that need to be ignored during the search, use a text editor that produces unformatted text, like Emacs or Notepad.

9. At this point, it is possible to submit the search by clicking on the BLAST icon (see Basic Protocol 2 for information on a blastn result output).

The results can also be recovered for a period of 24 hr following the execution. Go to the BLAST page (http://blast.ncbi.nlm.nih.gov), click on Recent Results, and retrieve results sorted by either submission time, request ID, status, program, title, query length, database, or expiration. Having a free NCBI account is helpful, and it allows, among many other benefits, storing search strategies for easy modification and resubmission.

**FORMATTING RESULTS OF A BLAST SEARCH**

This protocol provides an overview of the format options that are available for a BLAST output (Fig. 3.3.4). Clicking on the Reformat these results link (top-center of Fig. 3.3.3A) displays the choices for diverse alignment formats (Fig. 3.3.6).

**Necessary Resources**

**Hardware**

Any Internet-connected computer
Software

Web browser (Mozilla Firefox, Internet Explorer, Safari, or Opera)

Files

A file with the DNA query sequence in FASTA format (see APPENDIX 1B)

NOTE: The example shown here uses a FASTA file of the human let 7c microRNA. The sequence is available from the microRNA Database at http://microrna.sanger.ac.uk/cgi-bin/sequences/mirna_entry.pl?acc=MI0000064. Click on “Get sequence.”

1. Prepare the search as described (see Basic Protocol 1, steps 1 to 7 or Support Protocol 1, steps 1 to 8).

Select format options

2. Click on the link: “Reformat these Results” close to the top center of the BLAST result in Fig 3.3.3A.

3. Show options include:

   a. Pull-down menu: Show the only choice is Alignment (as of July 2008).
   b. Pull-down menu: “as” HTML (default) with rich links to external resources. The alternative, which is Plain text is practical when the results will be viewed by Notepad in Microsoft Windows or the UNIX/LINUX commands “less” or “more” (without quotes), or in a text editor. Either Plain text or XML (see below) is also more suitable than HTML for automatic parsing of the results for processing in BioPERL (see below) or storing in a relational database. ASN.1 is the format for the NCBI Toolkit, a topic that is outside of the scope of this unit.
   c. Advanced view is recommended over the Traditional BLAST output.

Specify how the alignment is to be shown

4. Select an Alignment View:

   a. Pairwise (default): Standard BLAST alignment in pairs of query sequence and database match (Fig. 3.3.4). This is the best display to evaluate the similarity between two sequences and is therefore recommended for most searches. However,
Finding Similar Nucleotide Sequences Using Network BLAST Searches

3.3.12

when individual positions are analyzed across a family of sequences, select one of the query-anchored multiple alignment displays.

b. **Pairwise with dots for identities** is best for displaying the differences between two highly similar sequences (not shown).

c. The format “Query-anchored letters for identities” (not shown) displays identities with single letter nucleic acid codes. This is similar to the standard multiple alignment displays. Pairwise alignment with dots for identities, Flat query-anchored with dots for identities, and Flat query-anchored with letters for identities are less practical than the default and the Hit Table (see below) formats.

d. **Hit table** (Fig. 3.3.7; for Megablast, see Alternate Protocols 1 and 2): This output format is sometimes missing from the drop-down menu. It is one of the most useful formats specifically designed for automatic parsing of vast outputs that are difficult to process by eye. The alignments are not displayed, just their critical parameters in tab-delimited format in the following order: query identifiers, subject (database) sequence identifiers, percent identity, alignment length, the number of mismatches, the number of gaps, query start position, query end, subject start, subject end, E-value, and (normalized) bit score. This output is best viewed in Excel. For novelty checks, quality assurance, and some other tasks, the information provided by this format is completely satisfactory. However, for the detailed biological analysis, the actual alignments are necessary. Since the alignments can easily be reproduced by pairwise blast (UNIT 3.4), saving just the tabular output can save a significant amount of disk storage space compared to storing full alignments.

This search demonstrates the amazing conservation of ribosomal proteins. Even alignment number 1000 (as of July, 2008; Fig. 3.3.10), to the 18S ribosomal RNA of Pisidium nitidum, a bivalvic mollusc, indicates an 88% identity over 153 of the query’s 156 nucleotides. Searching for such highly conserved sequences was accelerated by at least an order of magnitude by using discontiguous Megablast as opposed to the traditional blastn.

**Display panel**

5. Select a display panel:

a. The Graphical Overview box (checked by default) tells BLAST to provide a graphical overview of the top 100 BLAST hits relative to the query sequence (see Basic Protocol 1).

b. Leaving the Linkout box checked instructs BLAST to create a gateway via cross-links from the BLAST results to Unigene, Gene Expression Omnibus, Gene, Structure databases, and the Map Viewer. These resources are most useful in function or structural annotations, and further links, in particular from the Gene Database, can help revealing physiological or pathological connections.

c. Checking Sequence Retrieval causes to display checkboxes next to each alignment that allows downloading the database sequence if selected by the user.
d. Check the NCBI-gi (gene identifier number) box if you want to show the NCBI-gi identifiers in addition to the accession number and locus name. Note that gi numbers disappear by sequence revisions. For projects spanning several months, the stable accession numbers provide a better choice.

e. CDS feature, if checked, causes annotated sequence features including coding regions and translations to be shown. Mismatches in the database sequence that result in coding for a different amino acid residue than in the query, are displayed in pink.

f. Masking Character allows a choice between Lowercase (default) or N for nucleotide.

g. Masking Color let us to choose between Grey (default), Black, or Red.

Limit results to be displayed

6. Choose the limit results to be displayed:

a. Descriptions: select a number from the drop-down menu.

By decreasing the value for this parameter, searches can be accelerated and the output will be shorter. Keep in mind however, that the top BLAST hit is not necessarily the closest homolog or any homolog at all in the sense of common evolutionary origin or biological function (see Guidelines for Understanding Results). Another reason to display several hits is the potential lack of meaningful annotations, or to reveal potential discrepancies between annotations of different data sequences, a surprisingly frequent issue.

b. Graphical Overview, see the remarks above.

c. Alignments, see the remarks above.

d. Organism shows only sequences from the selected species or higher taxonomic unit. The page will try to auto-complement your choice.

e. Entrez query (see Support Protocol 1, step 2) can be entered to narrow down the search range. Boolean queries can also be used.

f. Specify Expect(ed) frequency value range (see Support Protocol 1, step 4; also see Critical Parameters).

**MEGABLAST SEARCH FOR RIBOSOMAL RNA**

NCBI has developed a version of BLAST that is orders of magnitude faster than the traditional *blastn* search. This improvement is partly due to longer words (28 bases) and partly due to indexing some genome databases. *Megablast* (Zhang et al., 2000; Morgulis et al., 2008) is a powerful tool for cross-genome comparisons, eliminating vector contamination, gene predictions, copy number studies, analyzing single-nucleotide polymorphisms, and some other tasks. It can be accessed from NCBI's top BLAST page (http://blast.ncbi.nlm.nih.gov) by clicking on either one of the listed genomes or on “nucleotide blast.” In the latter case, at the “Program selection” part choose either “Highly similar sequences (Megablast)” or “More dissimilar sequences (discontinuous Megablast),” depending on your search. This will take us to a screen like the one shown in Figure 3.3.9.

When word sizes (W) of 16 bases or longer are used, this tool can be (much) more than 10 times faster than *blastn*. *Megablast* will find and extend matches of word size W + 3. The default non-affine gap parameters (i.e., zero gap-opening penalty; *UNIT 3.1*) decrease the memory requirement and improve speed; however, the user can select affine gap parameters (preferably with larger word sizes) or word sizes as low as 8 bases. Generally speaking, non-affine gap parameters produce a higher number but shorter gaps. Such gap structures can be adequate for sequences with errors or for comparing polymorphisms within the same species. The genome-specific BLAST searches use *Megablast* by default.
Finding Similar Nucleotide Sequences Using Network BLAST Searches

3.3.14

Supplement 26 Current Protocols in Bioinformatics

Figure 3.3.8 The query, database and limit selection page for Megablast for the human 5.8S ribosomal RNA (NR_003285).

As an example, let us find the 1000 most similar sequences to the human 5.8S ribosomal RNA (NR_003285), a ribosomal sequence highly conserved during the evolution of eukaryotes (Fig. 3.3.8).

Necessary Resources

Hardware

Any Internet-connected computer

Software

Web browser (Mozilla Firefox, Internet Explorer, Safari, or Opera)

1. Point the browser to the address http://blast.ncbi.nlm.nih.gov and select “nucleotide blast.” On the top of the page, type NR_003285 into the “Enter accession number, gi, or FASTA sequence” text area.

2. In the Database section, select the button near “Others (nr, etc.)” to search the nonredundant genome/transcriptome collection.

3. In the Program Selection section, next to Optimize for, click on the button next to “More dissimilar sequences (discontinuous megablast).”

   The reason for this selection is that long segments of ribosomal RNAs are conserved, so it is safe to use word sizes of 11 or even more. However, a word length of 28 (the default for Megablast) would be too restrictive.

Algorithm parameters (Fig. 3.3.9)

4. Underneath the BLAST link, there is a link for Algorithm parameters. Click this link to expand the parameters options. Set the Max target sequences parameter to 1200.

   Leaving the Short queries box checked does not affect us.

5. To increase speed, set the Expect threshold value to 0.001 or even lower values.
6. Uncheck all filter and mask boxes since the sequence is highly conserved.

7. Use the default values of Template length and Template type.

8. At this point, it is possible to submit the search by clicking on the BLAST icon (see Basic Protocol 2 for information on a blastn result output).

The 1000 almost full-length, highly significant hits show the extreme evolutionary conservation of ribosomal RNAs. Even hit number 1000 to a bivalve mollusc, *Pisidium nitidum* strain L1 5.8S RNA, is significant at the 3e-46 level (Fig. 3.3.10).
FINDING TRANSCRIBED GENE COPIES AND SPLICE VARIANTS USING MEGABLAST

Go to NCBI's top BLAST page (http://blast.ncbi.nlm.nih.gov; Fig. 3.3.1) and select “nucleotide blast.” This will take us to the screen shown in Figure 3.3.11.

**Necessary Resources**

**Hardware**

Any Internet-connected computer

**Software**

Web browser (Mozilla Firefox, Internet Explorer, Safari, or Opera)

1. On the top of the page, type the Refseq identifier for the Duchenne muscular dystrophy reference mRNA: NM_000109 into the “Enter accession number, gi, or FASTA sequence” text area. This mRNA maps to the largest gene in the human genome due to a high number of unusually long introns.

2. In the Database section, select the button near “Human genomic + transcripts.”

3. Since now we want to search for the transcriptome only, enter the Entrez Query **NOT genomic**. Take care of capitalizing the Boolean operator “NOT,” otherwise it would be interpreted as a search term.

4. In the Program Selection section, next to Optimize for, click on the button in front of “Highly similar sequences (megablast).”

**Algorithm parameters (Fig. 3.3.12)**

5. Underneath the BLAST link, there is a link for Algorithm parameters. Click this link to expand the parameters options. Set the Max target sequences parameter to 1000.

**Figure 3.3.11** The query, database and limit selection page for Megablast for the human Duchenne muscular dystrophy gene (NM_000109).
6. Leave the Short queries box checked to be able to find medium-size exons.

7. To increase speed, set the Expect threshold value to 1e-10.

8. Set Word size to 16 to increase sensitivity.

9. The dystrophin gene contains Alu and other repeats, as well as low-complexity segments, even in the exons. These repeats and segments match to a large number of sites in the transcriptome. Therefore, check the boxes for Low-complexity regions and “species-specific repeats for Human.”

10. Use the default values of Template length and Template type.
Finding Similar Nucleotide Sequences Using Network BLAST Searches

3.3.18

Finding Similar Nucleotide Sequences Using Network BLAST Searches

Figure 3.3.14  The Genomic View of the localizations of the sequences similar to the human Duchenne muscular dystrophy gene (NM_000109).

11. At this point, it is possible to submit the search by clicking on the BLAST icon (see Basic Protocol 2 for information on a blastn result output).

The graphical display of the 280 HSPs matching to the query (Fig. 3.3.13) clearly indicates an unusually high number of splice variants. By clicking on the Genome view button, we can display the map of these exons on the chromosomes (Fig. 3.3.14). Clearly, this gene is present in multiple copies in the human genome. Note that the number of ticks on the chromosomes is the number of database sequences (transcripts), not the number of HSPs as in Figure 3.3.4.

GUIDELINES FOR UNDERSTANDING RESULTS

Marginally similar matches

If blastn does not find statistically significant matches, researchers can opt for a blastx search against a protein database, where the query nucleotide sequence is translated in six frames, a tblastn search of a protein query against a nucleotide database translated in six frames, or a tblastx search where both the query and the database are translated (UNIT 3.4). However, there are many cases when translated BLAST cannot find significantly similar database sequences either. When interpreting or further analyzing marginal similarities, extra caution is advised. The graphical displays need special consideration, primarily regarding gene predictions, short exons or long introns, detecting splice variants, and interpreting marginally significant matches between the query and multiple database sequences. The major difficulty in data interpretations is the inference from statistical significance to biological function, structure, or evolutionary relationship. Even when the inference is highly uncertain, a marginally similar match could be a useful hint for further analysis. Unfortunately, sometimes these marginally or not significant matches are treated as hard-core evidence for unjustified homology. This approach is responsible for many inaccurate annotations.

Instead, marginal hits can be a part of a well-integrated body of a diverse spectrum of supporting evidence. Among others, encoded protein domains, multiple alignments,
structural similarity, resembling patterns of gene expression in identical tissue, mutant or knockdown experiments, ontology classifications, and genomic localization, may form such a body of evidence.

**Multiple weak hits**
We have met several researchers who believed that the multiplicity of hits in itself was a valuable indicator of common ancestry. As an easy example, a high number of database sequences may match the same segment of the query with expected frequencies of $10^{-5}$ or higher, which is not considered significant in BLAST searches. It is well established that such matches occur quite frequently between unrelated sequences (Altschul et al., 1994).

A corollary of this claim would be that “biological significance” could be generated purely by sequencing members of the matching sequences in several species. In addition, the number of marginal hits may be increased purely by the high level of polymorphism of the family or by the interest in sequencing the members of that family.

**Transitive similarity**
Transitive similarity can also serve as a useful hint but not evidence that a set of three or more sequences may be related. There have been numerous successful tools (Gerstein, 1998; Baker et al., 2000; Bolten et al., 2001) using transitive similarities as hints. One should be extremely cautious in handling such clues to avoid false positives (see below), as transitive similarity does not qualify as hard-core evidence. If sequence A is similar to B and B is similar to C, then would A be similar to C, in spite of the fact that the direct comparison of A to C fails to confirm that? Similarity per se is not transitive. On an everyday example, transitive similarity would lead to the conclusion that white is similar to black as white is similar to light gray, light gray is similar to medium gray, etc. Try to confirm such similarities using some other techniques as discussed below briefly.

**The real use of marginal and transitive similarities**
Fortunately, the ultimate objective for most researchers is not to establish the statistically significant similarity between two or more sequences. Instead, the final goal is the prediction of biological function, the three-dimensional structure of the encoded protein or the exon structure of the gene. In this quest, sequence similarity may be a sufficient indicator, but not a necessary one. Profile-like methods (UNIT 2.3), analysis of domains (UNIT 2.2), structural predictions and comparisons, gene expression, or pathway data may provide real evidence for the function or the structure. If blastn, blastx, and tblastx (UNIT 3.4) cannot find good pairwise matches, translate the query in the best matching frame and search it against a protein database using Position-Specific Iterated BLAST (PSI-BLAST; Altschul et al., 1997). For the detection of conserved domains, search against a database of hidden Markov models (HMMs) like PFAM (Finn et al., 2008, UNIT 2.5) or SMART (Schultz et al., 1998; Letunic et al., 2006). HMM databases can be queried by hmmpfam (Eddy, 1998), SAM (Barrett et al., 1997), or other specific programs.

Basically, all of these methods may provide some both biologically and mathematically justified approaches to assign higher scores to matches and more negative scores for mismatches at conserved positions, indicating their importance in maintaining or disrupting the structure or the function of the protein.

**Gene predictions**
Mapping EST sequences to a genomic sequence is one of the best methods to predict genes on the condition that ESTs cover all exons in the gene. Fortunately, with the completion of the human genome, and with the unraveling of most of the human genes, chances are significantly higher for finding reliable matches between ESTs, proteins, and...
Finding Similar Nucleotide Sequences Using Network BLAST Searches

3.3.20

Supplement 26

Current Protocols in Bioinformatics

the genome than before. Now, specialized tools are available (Wu and Watanabe, 2005) for the optimal assembly of ESTs over the genomic sequence. blastn and Megablast can be used for the manual curation of these assemblies. Depending on the rate of sequencing errors for the ESTs, we require ∼95% or higher identity between the genomic sequence (query) and the ESTs (database sequences) of the same organism. This can be performed conveniently and rapidly using the Megablast tool described in Alternate Protocol 1.

For mapping (e.g., mouse ESTs to human genomic sequences or vice versa), a more permissive threshold can be chosen for the identity (i.e., ≥80% between human and rodent). Unfortunately, interspecies comparisons pose the risk of comparing paralogs (divergent homologs, like hemoglobin and myoglobin) in the absence of orthologs—the closest relatives in the two organisms with identical functions (e.g., like the genes encoding the hemoglobin α chain in human and mouse). Although BLAST controls the output by E-value, not by percent identity, sufficiently long exons can be detected by using an appropriately chosen E-value threshold. Both the “sufficient length” of an exon and the expected value threshold are functions of the search space, the latter being defined as the product of the query length and the size of the database.

We recommend using the “Query-anchored with letters for identities” alignment view. In this view, only those segments of the genomic sequence are shown that match the mRNA. The introns appear as gaps in the alignment.

**Splice variants**

Graphical representations are excellent tools for the visual discovery of splice variants (Fig. 3.3.13). Select a full-length mRNA or an EST sequence as query. Search it against a database of EST sequences and/or full-length mRNA sequences of the same organism. Specify an expected value that corresponds to a 95% or higher level of identity over 25 bases. Select the “Query-Anchored with dots for identities” alignment view. If the query has a nonterminal deletion or insertion relative to all matching EST sequences in the database, that indicates a new splice variant. Another interesting case occurs when multiple insertions/deletions occur in the query. Although the individual insertions or deletions may be present in some database sequences, their combination may produce a unique transcript. As the query EST sequence may contain incompletely spliced nuclear RNA, we recommend identifying the corresponding genomic sequence by a blastn search against the genome of the organism. Then, compare the EST against the underlying genomic segment using the ESTwise tool (Birney and Durbin, 1997). If the intronic regions next to the ends of the insertion or deletion do not contain exon boundaries (i.e., acceptor and donor sites), then the transcript may have been incorrectly spliced in the nucleus. Such transcripts are unlikely to code for a protein.

**Mapping sequencing reads**

One can map hundreds of millions of sequencing reads generated by ultra-high-throughput sequencing to the genome. Although such large-scale mappings require specialized algorithms like ELAND Genome Analyzer (Illumina), BLAT (Kent, 2002), or SSAHA (Ning et al., 2001), blastn is indispensible for assessing the performance of these algorithms.

**COMMENTARY**

**Background Information**

The efficiency of interpreting BLAST results is enhanced significantly by providing single-click access to annotations, literature, expression or structural information, other searches, and predictions available over the Internet or by intranet connections. This is true regardless of whether the researcher is using BLAST for just a few single searches or managing a high-performance pipeline generating and analyzing millions of searches. The raw BLAST output can be converted into an
information gateway leading to all the above data plus those that are specific to the project or pipeline.

Types, advantages, and disadvantages of network searches

There are several different alternatives to running BLAST searches locally. Servers with Web interfaces allow users to fine tune the parameters (also called options or arguments), select the database, and launch queries. The strict-sense network BLAST application provides a full UNIX/LINUX command-line interface between the user and the remote server. Specialized servers provide another convenient interface to BLAST.

For beginners, the National Center for Biotechnology Information’s (NCBI) user-friendly Web-server (http://blast.ncbi.nlm.nih.gov), or a similar host, provides the best way to learn BLAST and get hands-on experience with it. The application of the UNIX/LINUX command-line based network BLAST requires a minimal knowledge of the UNIX or LINUX operating system (see APPENDIX 1C). Large-scale submissions of several sequences against multiple databases can be automated by PERL (Tisdall, 2001).

Choose a public network BLAST service over local applications when all of the conditions below apply to you:

1. The CPU requirement of the submitted jobs does not exceed the limitations of the remote servers.
2. The user is not worried that others may monitor the query sequence.
3. No internal databases are used.

Some private servers guarantee the security of searches from hackers. Such services do not limit the number and the frequency of the jobs submitted since jobs are queued by sophisticated job scheduler systems on compute farms.

Network BLAST is the only choice when the BLAST package and the databases are not installed on the server of the user’s institution. Users may want to consider network BLAST when the locally stored databases are not updated from a File Transfer Protocol (FTP) server on a regular basis or when queuing of a large number of requests locally is an open problem. Local BLAST also requires a rudimentary knowledge of some basic UNIX commands, although an intranet BLAST server can easily be installed on a local Common Gateway Interface (Stein, 1998) or Java server (Liang, 2006).

Nucleic acid versus protein searches

Nucleic-acid-versus-nucleic-acid comparisons are generally much less sensitive in finding distant relatives than alignment methods involving at least one protein sequence. This decreased sensitivity comes from the observation that at the nucleic acid level, mutations causing amino acid substitutions cannot be differentiated from silent mutations that do not affect the protein. In addition, nucleic-acid-versus-nucleic-acid comparisons cannot differentiate between amino acid substitutions conserving the structure of the protein and substitutions that disrupt it. For most amino acid positions, the replacement of an alanine residue has minor effects, while substituting a cysteine participating in a disulfide bridge may disrupt the tertiary or quaternary structure. Consequently, orthologous genes accumulate a higher number of silent than nonsilent mutations. Conservative amino acid substitutions are more common than substitutions of residues with vastly different characteristics (e.g., structure and hydrophobicity).

While refined statistical measures (Dayhoff and Eck, 1968; Henikoff and Henikoff, 1992; Baxevanis, 2005) are available for amino acid comparisons to score these substitutions based on their biological relevance, nucleic acid substitutions cannot be treated in such a precise manner.

Consider also that nucleic acid sequences are by far more random than amino acid sequences. One reason for this is that while the frequencies of A, C, G, and T are close to 25% in coding sequences of higher eukaryotes, amino acid residues have markedly uneven distributions. For example, leucine is ~10× more abundant in soluble proteins than tryptophan or methionine (Karlin and Bucher, 1992). From the amino acid sequence, the conservation of the critical structural and functional features can be analyzed, while the same would be extremely (and unnecessarily) difficult at the nucleotide level. For all the above reasons, when examining common function or ancestry of two proteins, amino acid comparisons provide higher levels of sensitivity and selectivity than nucleic acid comparisons.

Other applications can only be done at the nucleic acid level. Assembling genomic sequences or expressed sequence tags (ESTs) or mapping ESTs to genomic sequences from a single organism require a high level of identity (95% or more) at the DNA or RNA level. These tasks definitely require Megablast, blastn, or another nucleic-acid-to-nucleic-acid
comparison tool (UNIT 3.2). Such tools can handle frameshift errors in a more robust way than amino acid comparison programs as the translation changes drastically downstream of the frameshift. Promoter, enhancer sequences, and tRNAs that are not translated cannot be analyzed by blastp and its derivatives.

### Critical Parameters and Troubleshooting

**Why is it necessary to fine-tune the search parameters?**

BLAST is one of the most user-friendly tools in computational biology, particularly when accessed over the network. However, the ease of this application may be somewhat deceptive, as sometimes the default values will not achieve the desired results. For example, primer design may require that a decanucleotide would not appear in the transcriptome of the organism in other genes. As the default word size for blastn is 11 nucleotides, a shorter word size (10) must be specified (see Support Protocol 1).

Relying on the default values of the search parameters, disregarding the specific circumstances of the search, frequently generates prohibitively long outputs. Keep in mind that while submitting a BLAST search is very easy and fast, visual inspection or even (semi-)automatic querying of relational databases created from the parsed results may be a daunting task. Fortunately, BLAST is a highly responsive instrument that can be finely tuned for specific query/database combinations. Careful selection of the parameters is a rewarding investment of the researcher’s time, with hundred-fold savings on the size of the output, and decreasing noise. Formatting results (e.g., for displaying a large number of EST matches to a genomic sequence or to a full-length mRNA by showing mismatched bases only in a multiple alignment) is much more visual than the traditional pairwise hit format. Fine-tuning of the expected number of hits, the number of hits to be displayed, word size, and other parameters can accelerate BLAST searches significantly.

### A word of caution about nonredundant databases

Nonredundant (nr) databases are generated from multiple databases by the NCBI’s nrdb program available by anonymous FTP. Sequences that are 100% identical or one that is fully contained in the other are merged into a single entry. The definition line of the new entry will contain all the definition lines of the merged sequences, the individual “parent” definition lines are separated by control-A characters. Note that the databases created by the nrdb program are still highly redundant in the sense that if two sequences differ at a single position, they will remain listed as separate entries. Taking into account the error rate of sequencing, as well as the natural variations, the redundancy of databases causes unjustified reduction in the expected values, and the number of hits displayed can be much higher than necessary. One possible remedy for this problem is offered by nrdb90 (Holm and Sander, 1998), a program to eliminate sequences that are identical in over 90% to 99% of the nucleotides or amino acids. The actual percentage can be specified by the user. Note that nrdb90 may require several orders of magnitude more CPU time than nrdb.

### Filter parameters

**Low-complexity filters.** For most applications, leave the Low-Complexity filtering parameter checked. Poly-A tails, collagen helix motifs, short repeats, and a number of other segments can match thousands of similar sequences that may not share common evolutionary origin. To avoid a large number of such possibly false “homologs,” the query nucleic acid sequence can be filtered for these low-complexity regions using the DUST program (Morgulis et al., 2006) for blastn and Megablast and SEG (Wootton and Federhen, 1996) for other programs. However, when one wishes to find out whether or not sequences identical or highly similar to the query have been published in diverse databases (novelty checking), filtering should be turned off at the expense of decreasing the signal-to-noise ratio of the output. Filtering is only applied to the query sequence (or its translation products), not to database sequences. For database filtering, see the “Mask for lookup table only” parameter below.

**Filtering human repeats.** Except for when assembling long sequences, turn on the Filtering Human repeats parameter. Alu, LINE, SINE, and other human sequences (Jurka et al., 2007) occurring in a large number of repeats are masked when this parameter is selected. Repeat filtering decreases noise in the output and increases search speed. These improvements are particularly significant during the analysis of large genomic sequences with hundreds of repeats. Fine-tuned masking of Alu, SINE, LINE, and other human repeats can be performed at the RepeatMasker Web site.
As of July, 2008, NCBI offers this option exclusively for the blastn program.

**Mask for Lookup Table Only.** As an alternative to filtering the query for low-complexity regions, one may want to mask the database lookup table.

**Mask Lower Case.** Nucleotide sequences coding for epidermal growth factor or other frequent domains may have too many database hits that may or may not be helpful in the inference for the biological function. To mask these regions manually, set the Mask Lower Case parameter to turn off searching for these regions. This is a practical feature for analyzing splice variants or for excluding potentially mispredicted exons or Alu repeats.

**Expect range**

The expected frequency is the number of database hits with a score equal to or higher than the match between the query and the subject (database) sequence. This most important, advanced parameter may save considerable time when analyzing marginally significant hits. It blocks reporting those matches that can be expected to occur in the database by the given absolute frequency. In contrast to relative frequencies, absolute frequencies (the expected numbers of occurrences of an event) can exceed 1. An expected (absolute) frequency of 10, which is the default value, means that for the given score or higher, 10 merely random matches can be expected in the database of given size. This extremely non-conservative level of statistical significance may be necessary for finding short exons or frameshifted translations. In the author’s experience, in order to achieve reliable statistical inference for the common evolutionary origin or the function of two sequences, the expectation value should not exceed $10^{-10}$ $(1e-10)$ so as to minimize both false-positive and false-negative matches; however, note that such expected frequencies can occur for gene pairs coding for proteins that diverged in function. On the other hand, at the nucleotide level a number of true homologs will be missed due to their divergence in sequence. All these factors mandate the time-consuming visual inspection of the hits (see the section Guidelines for Understanding Results). Database sequences matching with high expected frequencies (i.e., with marginal levels of statistical significance) should be carefully analyzed by hidden Markov models, PSI-BLAST, GeneWise, or other tools, as well as by human eye.

**Suggestions for Further Analysis**

**Integrating different predictions and searches over diverse databases**

The interpretation of BLAST results is hindered by hits to unannotated sequences, alternative splicing, the absence of close similarities, or the opposite, misleading similarities to unrelated sequences or to functionally diverged relatives, and uncertain gene models. These issues warrant careful interpretation of results, integration of searches over diverse databases, and application of domain searches (UNITS 2.4 & 2.5), gene predictions (Chapter 4), EST analysis, structural predictions, knockout mutants or knockdowns by short interfering RNAs (Elbashir et al., 2002), and gene expression information to locate a number of yet unidentified genes and to assign biological functions to thousands of others.

Using the hyperlinks to annotations, PUBMED references, and other resources provided by the BLAST Web servers is just the beginning. The real challenge is to create textual and graphical summary pages optimized for a computer screen showing all the matching nucleotide and protein sequences, and relevant predictions aligned to a gene or genomic sequence. To this end, the author recommends parsing the sequence identifiers, functional, taxonomic, tissue, disease, polymorphism, and other data, alignment quality measures, and the coordinates of the local alignments from the BLAST output by the BLAST module of the BioPerl package (http://www.bioperl.org) or similar programs. These can be stored in relational databases (UNIT 9.1), tab-delimited flat files, XML files, and others, possibly stored on a (remote) server. This way annotations or evaluations can be performed on inexpensive client computers of any reasonable operational system and hardware platform. Installing BLAST and Web-server software, relational databases (UNIT 9.1), and so forth is necessary for the server or servers only, not for the clients. Most importantly, central servers facilitate community (company- or department-wide) annotations and sharing the results. This centralization also eliminates the necessity for multiple copies of the databases, and minimizes duplicate runs of searches and their redundant storage on different servers.
computers. Regular (nightly) backups of data and programs demand considerably less efforts and resources on a few servers than on an army of hosts.

**Integrating information from numerous sources**

The information necessary to annotate a single gene can be dispersed on hundreds of pages of original output from BLAST, gene predictions, etc. This necessitates the automatic extraction of useful data, decreasing redundancy when annotating large amounts of DNA sequences. Integrated, nonredundant data representation and analysis can dramatically save on the human annotation time required for the genome and proteome projects and other gene discovery efforts. Such tools also decrease the chances for overlooking important information buried in the haystack of output and numerous other resources into a dynamic interface that facilitates easy navigation.

The University of California Santa Cruz Genome Browser discussed in UNIT 1.4 (Karolchik et al., 2008; Zweig et al., 2008) organizes a wealth of different analyses into a single zoomable frame anchored to a user-selected segment of the selected genome. The essential pieces of information from hundreds of pages from the original outputs are condensed into a single-screen graphical representation. The results of the searches of the genomic segment against the nr protein database, the Gene Database’s mRNA collection, human, mouse and other ESTs are superposed over the genomic segment. The individual searches or predictions can be expanded or collapsed at the user’s discretion. Detailed information can be displayed in a separate frame by clicking on the lines representing search results or predictions.

**Parsing BLAST results for automatic processing and/or for input to relational databases**

NCBI BLAST results can be produced in XML format by selecting XML format from the Formatting screen (Fig. 3.3.6; also see Support Protocol 2). Such output allows researchers to use readily available XML parsing tools (Møller and Schwartzbach, 2006) on the BLAST results. The XML output is more stable than the flat BLAST reports, which are subject to change; however, XML outputs may take up 100× more disk space than plain result files, which can also be parsed, e.g., by BioPerl’s (Stajich, 2007) BLAST::Parse module (http://bioperl.org).

The parsed results can be easily loaded into MySQL (Ullman, 2006), ORACLE, or other relational databases. Relational databases allow extremely fast retrieval of data, running sophisticated queries, and a high level of integration of data for user-friendly interfaces (UNIT 9.1). The queries may perform, for example, automatic novelty analysis using several sequence databases; however, these topics are far beyond the scope of this introductory unit.

**A word of caution**

Relatively simple tasks like novelty checking can and should be performed by fully automated tools; however, exon predictions, assessments of biological function, and many other analyses, while receiving vital support from automated tools, require careful human evaluation in order to avoid the numerous problems unforeseen by the designers of the automated tools.

**Acknowledgments**

This work was supported by NSF grant EPS-0701892. The author is grateful to Drs. Jean-Jack M. Riehovhen and Sridhar A. Malkaram for their help and comments.

**Literature Cited**


Tisdall, J.D. 2001 Beginning PERL for Bioinformatics: An Introduction to PERL for Biologists. O'Reilly, Sebastopol, Calif.


Finding Similar Nucleotide Sequences Using Network BLAST Searches

3.3.26


Key References

Altschul et al., 1994. See above.
An excellent review on the application of pairwise BLAST tools for the identification of possible coding regions, for the elucidation of gene structure and protein function. This review discusses significance sequence filtering, database issues, alignment statistics, gap costs, scoring systems, and others.

Altschul et al., 1997. See above.
This is the original research paper on gapped alignment blast and position specific iterative BLAST. A series of algorithmic and performance improvements, gap penalty, and statistical considerations, as well as biological examples with marginal similarities are covered.

A widely taught, clearly written textbook that introduces pairwise sequence similarity searches, biological databases, and many other areas of bioinformatics. Reviews the general concepts of alignments, scoring matrices, and BLAST with practical applications and guidelines for interpretation.

Korf et al., 2003. See above.
An excellent overview of theory and practice of the BLAST tools as of 2003. This most comprehensive and easy-to-understand textbook is highly recommended to everyone in bioinformatics or computational biology.

Internet Resources

The NCBI BLAST Web site.

The Entrez Documentation at NCBI.

The Entrez site for nucleic acid searches at NCBI.

http://www.bioperl.org
The BioPerl site.

The full documentation for BLAST at NCBI.

http://www.ebi.ac.uk/blast2/nucleotide.html
The European Bioinformatics Institute Server for the Washington University BLAST.

http://repeatmasker.genome.washington.edu
The RepeatMasker Website.

http://www.girinst.org/Censor_Server.html
The Genetic Research Institute Website.
Finding Homologs in Amino Acid Sequences Using Network BLAST Searches

Istvan (Steve) Ladunga¹

¹University of Nebraska–Lincoln, Lincoln, Nebraska

ABSTRACT

The Basic Local Alignment Search Tool (BLAST) is the most fundamental (and most misused) algorithm and software in bioinformatics/computational biology for functional assessment of unknown proteins or discovery of similar proteins with potentially common evolutionary origins. We show how to balance sensitivity with selectivity (without generating massive output) by selecting and demonstrating proper database, algorithm, and alignment display options of the user-friendly Web sites of the National Center for Biotechnology Information (NCBI). We discuss protein query searches against protein databases and submission of all combinations of translated searches. Careful biological and statistical inferences are drawn to possible functions, taking into account the highly nonrandom nature of proteins. Guidelines for such inferences, using real-life biological examples (e.g., protein kinases with widely distributed structural and functional domains), are provided. We show how to avoid incorrect functional inference from misleading similarities, using the divergent evolution of a serine protease domain that erodes the protease function in haptoglobins. Curr. Protoc. Bioinform. 25:3.4.1-3.4.34. © 2009 by John Wiley & Sons, Inc.

Keywords: BLAST • bioinformatics • computational biology • database search • functional assessment • statistical inference • local alignment • translated database search

INTRODUCTION

BLAST, the Basic Local Alignment Search Tool (Altschul et al., 1990, 1997; UNIT 3.3), is the single most fundamental tool of bioinformatics and computational biology. It is used more frequently and more productively than any other biological sequence database search program. Its broad range of applications includes, among many others, rapid and highly reliable identification of similar or identical sequences in large databases of proteins or nucleic acids, including whole genomes. From sequence similarity, one can, with proper care, infer biological function, structure, polymorphisms, or evolutionary relationships.

It is critical to understand that BLAST reports similarity, not homology. Homology in biology is defined as common ancestry (Reeck et al., 1987), which is neither a simple nor unique function of similarity. However, homology and common function can be inferred from similarity, and, as is the case with any statistical inference, it comes with a considerable degree of uncertainty. Identical matches between a query and a database sequence may indicate matching of a predicted peptide from a proteomics experiment to a protein database. BLAST is the first technique to use when looking for homologs of a new protein, genomic DNA sequences, or mRNA, including expressed sequence tags (EST; see APPENDIX 2). If the BLAST search cannot safely indicate relatives, other approaches, such as dynamic programming (Smith and Waterman, 1981), hidden Markov models (Krogh et al., 1994), and structural comparisons (Wang et al., 2007) can be applied.
The unprecedented popularity of BLAST is due to a number of major advantages over its competitors. BLAST is several times faster than any other currently available general search tool. BLAST contributes to the integration of biology via links and interfaces to a diverse spectrum of external databases. BLAST has been incorporated into most comprehensive bioinformatics systems, including gene discovery pipelines. To serve this purpose, BLAST output is available in several special formats. HyperText Markup Language (HTML) links to sequence, gene expression, structure, polymorphism, and other databases, as well as literature, annotations, chemoinformatics, and other resources. Alternatively, one can generate BLAST outputs in eXtensible Markup Language (XML; Møller and Schwartzbach, 2006) format for standardized storage and retrieval of the results, or with tab-delimited fields as input for Microsoft Excel or relational databases like MySQL (Ullman, 2006).

Although some tools exceed the performance of pairwise BLAST in detecting remote homologs (see Guidelines for Understanding Results), the search algorithm, along with sophisticated calculations of statistical significance and gapped alignments, provide high levels of sensitivity and selectivity. Sensitivity is the ability to recognize true positive phenomena (in this case, finding proteins that are related to the query protein in functional or evolutionary terms). Sensitivity is the ratio of true positive predictions to the sum of true positive and false negative predictions. Selectivity is the ability to recognize true negative phenomena, in this case, unrelated proteins. Selectivity is the ratio of true negative predictions to the sum of true negative and false positive predictions. One of the primary quests of bioinformatics is the highest balance of sensitivity and selectivity, i.e., simultaneously generating the highest number of true positive and true negative predictions.

The ease of running user-friendly BLAST searches over the Web (or even on command-line windows) may lead novice users to believe that the biological interpretation of the observed level of similarity is similarly easy. This is often not the case. For example, inflated scores are caused by widely spread protein domains. Therefore, in this unit, we try to assist both novice users and more advanced researchers in the interpretation of the results, handling statistical significance and biological relevance, selection of complementary analyses, and utilization of different integrative approaches. To facilitate learning, advanced topics have been marked as such; beginners may leave these parts for later review.

This unit covers three classes of BLAST programs (Altschul et al., 1990; 1997): standard protein-to-protein searches (see Basic Protocol 1 and Support Protocols 1 and 2), translated searches where the query and/or the database consist of nucleotide sequences translated into proteins (see Basic Protocol 2; Gish and States, 1993), and, finally, programs for comparing two sequences, as opposed to searching one sequence against a database of sequences (see Basic Protocol 3). Note that translated searches differ considerably from the blastn searches discussed in UNIT 3.3. We do not discuss the Washington University BLAST (WU-BLAST) programs (Gish and States, 1993), which use different methods and statistics. Please refer to Korf et al. (2003) for a tutorial on both BLAST packages and the Baxevanis (2005) chapter on assessing pairwise sequence.

**BASIC PROTOCOL 1**

**USING THE BLAST WEB INTERFACE TO PERFORM A PROTEIN-TO-PROTEIN SEARCH (blastp)**

Using the NCBI Web site (Wheeler et al., 2008) is the best way to get hands-on experience with blastp. This user-friendly site provides a tutorial and well organized documentation that facilitate experimenting with different search arguments.

**Necessary Resources**

**Hardware**

- Computer with Internet access
Software

Up-to-date Internet browser, e.g., Internet Explorer (http://www.microsoft.com/ie), Netscape (http://browser.netscape.com), Firefox (http://www.mozilla.org/firefox), or Safari (http://www.apple.com/safari)

Files

Protein sequence identifier or protein query sequence in FASTA format (APPENDIX 1B)

Navigate to the BLAST Web site

1. Point the Web browser to the URL http://blast.ncbi.nlm.nih.gov. Bookmark this page (recommended; Fig. 3.4.1).

2. Click “protein blast” in the Basic BLAST section. As of November 2008, the resulting page (Fig. 3.4.2), is organized into four major sections: Enter Query Sequence, Choose Search Set, Program Selection, and Algorithm Parameters (Fig. 3.4.3).

   Help is accessible by clicking on the small blue circles to the right of the sections and subsections.

Populate the Enter Sequence Query section

3. Under the heading “Enter accession number, gi, or FASTA sequence,” type an accession number (e.g., NP_005913). Alternatively, copy and paste the protein query sequence in FASTA format into the search text area. Note that the “Job Title” field changes to “NP_005913:mitogen-activated protein kinase ...” For the Database pull-down menu, leave this set to “Non-redundant protein sequences (nr).”

   Bare sequences with no header line are also accepted, but this is not a recommended practice because it makes it difficult to identify the query sequence later.
Finding Homologs in Amino Acid Sequences Using Network BLAST Searches

3.4.4

Supplement 25 Current Protocols in Bioinformatics

Only the standard IUPAC abbreviations for amino acids can be entered (see APPENDIX 1A), including the letter U for selenocysteine or X for any residue. In addition, hyphens (-) denote gaps of indeterminate length, and asterisks (*) stand for stop codons. Numbers are ignored without warning. The sequence should not contain empty lines, but lines with both sequence and whitespace are acceptable. For more information, see http://www.ncbi.nlm.nih.gov/BLAST/blastcgihelp.shtml.

4. (Advanced) If necessary, search only a part of the query sequence by entering a range in the “From and/or To” fields in the “Query subrange” subsection.

This is practical when focusing the search on a region or domain of the query sequence or for those regions that do not contain some uncharacteristic, widely distributed sequence element. Such elements include the epidermal growth factor domain, which has been dispersed into several, otherwise unrelated proteins (see Guidelines for Understanding Results). If no Start or End is entered, blastp takes the complete sequence as query.

Populate the Choose Search Set section

5. Move to the “Choose Search Set” section and select the database to be searched from the Database pull-down menu (see choices below). In this example, leave the default “Nonredundant protein sequences (nr)” database.

Baxevanis and Ouellette (2005) provide an excellent detailed overview of molecular sequence databases.

Nonredundant protein sequences (nr). This default database serves best for most purposes, particularly the prediction of biological function. This is the largest collection of protein sequences available (Wheeler et al., 2008), compiled from the following resources: translations of GenBank (Benson et al., 2008) coding sequences, Swiss-Prot protein sequences (see below), Protein Data Bank of three-dimensional structures (Berman et al., 2000), the Universal Protein Information Resource (Bairoch et al., 2005; Wu et al., 2006), and the Protein Research Foundation sequences. The nr database does not include any sequences from environmental samples (see below).
Figure 3.4.3  Algorithm parameters for advanced BLAST searches on the NCBI Web server can be displayed by clicking on “Algorithm parameter” (shown at the bottom of Fig. 3.4.1).

The term nonredundant means that there are no two completely identical sequences or any sequence that is fully included in some other sequence, but entries that differ even at only a single residue are not eliminated. For a more detailed discussion on redundancy, see the Critical Parameters section in UNIT 3.3.

Reference proteins (refseq_proteins). This database (Pruitt et al., 2007) at NCBI stores highly curated and validated reference sequences with extensive automatic and manual analyses.

Swiss-Prot protein sequences (swissprot). The Swiss-Prot database (Boutet et al., 2007) has achieved the highest quality of annotations, which comes at the price of lower coverage and longer delays with annotation. Many users have found it helpful to create links to the detailed, well organized pages of Swiss-Prot annotations (http://expasy.org/sprot for Europe, or several mirror sites like http://us.expasy.org/sprot) using a simple script. Swiss-Prot is a part of the Universal Protein Resource (UniProt) and is fully included in the Nonredundant protein sequences (nr) database as well.

Patented protein sequences (pat). This database contains sequences for which patent applications have been published for evaluating the novelty of a query protein, a
Finding Homologs in Amino Acid Sequences Using Network BLAST Searches

3.4.6

candidate for patenting. Note that patent applications are published only after an 18-month waiting period, a circumstance that causes a significant risk when filing an application for a presumably novel protein. Novelty searches should also be performed on the patented nucleic acid sequences database because full-length or partial mRNAs or expressed sequence tags (ESTs) were also patented in the earlier years. Before filing a patent application, consider the rapidly changing legal definitions for the novelty of a protein. As a rule, publication in any database, journal, or publicly accessible source of information terminates the novel status.

Environmental samples proteins (env_nr). Unlike almost every other database, the source organisms for the proteins are unknown because the heterogeneous samples taken from the ocean depth or mine shafts may contain hundreds of species from diverse phyla. This collection represents the broadest spectrum of biological diversity, including species yet unknown to science. On the downside, users have to consider that some sequences are mere artifacts assembled from multiple, unrelated genomes.

Protein Data Bank proteins (pdb). Searching against PDB (Westbrook et al., 2002) may identify similar proteins with known three-dimensional structure.

6. If desired, check “Organism (optional)” to limit the species or other taxonomic units searched. If unchecked, the BLAST search will be performed against the complete database selected. For example, type mamm into the text field, which will be auto-complemented to mammals (taxid: 40674). Alternatively, to find the mouse homolog(s) of the human protein kinase selected above, type mouse into the text field, and a menu with the names of several organisms will appear. Select house mouse (taxid: 10090).

7. (Optional) To limit the search database by a query referring to specific field codes and/or any words in the annotation, use Entrez Query (optional). Enter field codes in square brackets as in the examples shown in Table 3.4.1. Combine terms using Boolean operators such as AND, OR, or NOT (all capitalized). For example, enter the query:

<table>
<thead>
<tr>
<th>Field Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[fkey]</td>
<td>Feature key</td>
</tr>
<tr>
<td>[acccn]</td>
<td>Accession</td>
</tr>
<tr>
<td>[ecno]</td>
<td>Enzyme catalog number</td>
</tr>
<tr>
<td>[titl]</td>
<td>Title word</td>
</tr>
<tr>
<td>[word]</td>
<td>Text word*</td>
</tr>
</tbody>
</table>

*For example, classes of proteins, pathways, Gene Ontology terms (see UNIT 1.3).

Note that omitting the parentheses around activating OR mutant OR putative will exclude all proteins that have the word “activating” in any part of the annotations, but would include all mutants and all putative proteins (not only kinases), contrary to our intent.

Entrez is a powerful search engine that performs federated database searches over PubMed, nucleic acid/protein sequence, domain and structure databases, genomes, disease, polymorphism and expression observations, and taxonomy. For more information, see UNIT 1.3 and http://www.ncbi.nlm.nih.gov/books/bv.fcgi?rid=helpentrez.chap3.4.6

Table 3.4.1 Examples of Field Codes Used to “Entrez Query”
Populate the Program Selection section and search

8. Move to the “Program Selection” section and select from “blastp (protein-protein BLAST),” “PSI-BLAST (Position-Specific Iterated BLAST)”, and “PHI-BLAST (Pattern Hit Initiated BLAST).”

   This unit will discuss only the pairwise protein-to-protein blastp search and translated BLAST searches.

9. Check the “Show results in a new window” button and submit the search by clicking on the BLAST icon in the bottom section.

   For now, we can skip fine-tuning the algorithm parameters, and this may be acceptable for the first days of learning.

10. (Optional, for more ambitious researchers) Click on the “Algorithm parameters” link, and follow the details of this advanced approach, as described in Support Protocol 1.

   NCBI’s QBLAST queuing system facilitates the retrieval of results at the user’s convenience within 24 hr of the execution. QBLAST also allows reformatting the display of results several times with different format arguments. With QBLAST, all searches submitted to the servers are assigned a request identifier (RID; see Fig. 3.4.4A, next to “Formatting Results”).

Interpret the default blastp result output

The NCBI servers usually return the results in a few minutes, but at peak hours, users have to wait longer. The content of BLAST output is enriched with information on conserved domains, the distribution of hits along the query sequence, color-coded cross-references to other databases, and anchors to facilitate jumping from one-line descriptions to the actual pairwise alignments (Fig. 3.4.4C). Here we describe the design as of November 2008, but the organization of the HTML output is constantly evolving. However, since a single output design cannot fit all the possible applications, users can specify their format requirements by clicking on the link “Reformat These Results,” close to the top center of the screen (see below).

The default output is organized into the following sections: job identifier, references, search parameters, graphical representation of the alignments, distance tree of results, related structures one-line descriptions, alignments and statistics. More advanced output formats for specific purposes (e.g., automatic parsing) are discussed in Support Protocol 2.

11. The information in the “Administrative section” is mostly self-explanatory.

12. Use the “RID number” to retrieve results within a 24-hour period. Go to the BLAST page (http://blast.ncbi.nlm.nih.gov), select “protein-to-protein BLAST” and then click on “Retrieve Results” for an RID link just below the BLAST logo (Fig. 3.4.4A).

13. (Advanced) Click on “Taxonomy reports” (Fig. 3.4.4A) to display the hit sequences according to taxonomical categories, a service that can ease finding similar proteins in diverse organisms (Fig. 3.4.5).

14. Use “Distribution of <number> Blast Hits on the Query Sequence” (Fig. 3.4.4B) to align search sequences to the query sequence, using the interactive, color-coded, graphical display describing the where and how of the process. For example, on this particular search result page, red lines indicates BLAST hits with an alignment score of 200 or higher, and magenta-colored lines show hits that score of 80 to 200. Numbers under the “Query” line indicate the number of positions in the query sequence.
Finding Homologs in Amino Acid Sequences Using Network BLAST Searches

3.4.8

Supplement 25 Current Protocols in Bioinformatics

Figure 3.4.4 (continues on next page) The blastp result page displayed in three panels. (A) The top section of the results with links to the BLAST home page, recent results, saved search strategies, help, reformatting, and resubmission, and for saving the search strategies. Database and query sequence information is also shown. (B) The graphical summary of the alignments and their one-line descriptions, with Link Out icons representing linked external databases (see Table 3.4.4 in Support Protocol 2). (C) Sequence retrieval links and detailed pairwise alignments between the query sequence and the search sequences.
Position the cursor over a colored line to display the brief one-line description of the database sequence.

Partial alignments may indicate alternative splicing, protein domains or mosaic (chimeric) proteins.

15. (Advanced) Click on “Distance tree of the results” (Fig. 3.4.4B) to generate a tree based on protein similarity (unlike the “Taxonomy Reports” in step 13) to obtain the display shown in Figure 3.4.6.

Clearly, serine/threonine kinases in organisms as diverse as human and fungi are clustered at lower distances than diverse kinases in the same species.

Since these trees are generated by pairwise alignments, and no phylogenetic corrections are performed, we cannot call them bona fide phylogenetic trees. For the construction of phylogenetic trees, see Chapter 6.
Figure 3.4.5  The taxonomy report displays found sequences, sorted by organism.

Figure 3.4.6  The distance tree of results shows results based on protein similarity.
Finding Similarities and Inferring Homologies

3.4.11

Figure 3.4.7 The related structures display shows related proteins with known three-dimensional structures.

16. (Advanced) Display the hits to NCBI’s Molecular Modeling Database using “Related Structures” by clicking on the button shown in Figure 3.4.4B (if available; see Fig. 3.4.7; Wang et al., 2007).

Thanks to the rapid development in structural biology, most novel proteins have some distant relatives with known three-dimensional structures that can provide inference for the structure or function of the query protein. Links are supplied to the structures in the Protein Data Bank (Berman et al., 2000) where structures can be displayed interactively by a repertoire of structure viewer software (see Chapter 5).

17. Select “Sequences producing significant alignments” to provide one-line descriptions, including the identifiers of the matching database sequence, a short description, the score, and the expected frequency value (E-value) of getting this score or higher purely by chance calculated using a semi-random model (see Interpreting the Results below). In the last columns, color-coded one-letter symbols indicate the additional database(s) where the search sequence was found. The sequence identifiers are linked to the respective database entries, and clicking on a score will take us to the pairwise alignment of the query and this database sequence. See more complete descriptions of the items below.

Pairwise local alignments (Fig. 3.4.4C), in contrast to global alignments, do not necessarily cover the whole length of the query and the database sequences. Instead, high-scoring segment pair(s) (HSP) is (are) reported. The identically matching words (seeds) are extended in both directions so as to maximize the cumulative score based on the scoring matrix and gap penalties selected. Small drops in the cumulative score are allowed as long as further positively scoring residue pairs allow for a higher cumulative score. Note that a query and a database sequence may have more than one HSP (alignment); blastp reports only nonoverlapping alignments.

Please note that Figure 3.4.4C shows only one of the possible HSPs between the query and subject sequences. The definition line is not repeated for the subsequent HSPs, but scores and other statistics are calculated for each and every HSP separately.
18. Identifiers and the short description of the database sequence are reported first.

19. “Score” is the cumulative score for the HSP, in raw and normalized (bitwise) forms.

20. “Expect” (E-val) is the expected frequency of the random occurrence of the above “Score” or higher, between a sequence as long as the query and a random database of the same size (see Guidelines for Understanding Results).

21. The number of “Identities” is followed by the total number of letters (amino acid residues) in the alignment and by the percentage of identities.

22. “Positives” are the number of aligned residue pairs with positive scores (including identities). These indicate conservative substitutions like between isoleucine and valine residues that are not likely to affect the function or the structure of the protein.

23. “Gaps” are the number of gaps introduced during sequence alignment.

24. The coordinates of the “Query” and “Sbjct” (database) sequences are provided for each line.

25. For alignment lines for the “Query” and “Sbjct” sequences, and in the middle, the match between them, identities are marked by the sequence letter, with positively scoring but nonidentical residue pairs by a plus sign (+), negatively scoring pairs by spaces, and gaps by hyphens (–).

26. (Advanced) Use “Statistics” at the end of the BLAST report to include, e.g., the number of residues in the database, its effective size, the Karlin-Altschul parameters (Karlin and Altschul, 1990), and the length of the query. For a description of this section, we recommend the Korf et al. (2003) tutorial on BLAST next to “Formatting Results”.

NCBI’s QBLAST queuing system facilitates the retrieval of results at the user’s convenience within 24 hr of the execution. QBLAST also allows reformating the display of results several times with different format arguments. With QBLAST, all searches submitted to the servers are assigned a request identifier (RID; see Fig. 3.4.4A).

SETTING ALGORITHM PARAMETERS FOR ADVANCED BLAST

It is important to realize that the default set of parameters cannot be optimal or even functional for the diverse purposes and applications of BLAST. As an example, filtering out low-complexity regions is necessary for eliminating potentially thousands of spurious hits with no biological relevance. However, when searching for identical or highly similar sequences, this filter should be turned off. Setting certain parameters (Fig. 3.4.3) increases either the sensitivity or selectivity (but rarely both) for the specific query sequence–database combination. It also allows the scientist to specify the retrieval of closely similar sequences or to include marginally similar sequences, as well. Different filters can be turned on or off (see Table 3.4.2). In addition nonstandard searches can be launched when the pairwise BLAST runs do not find sequences that are similar to the query as estimated by statistical measures.

Necessary Resources

Hardware

Computer with Internet access

Software

Up-to-date Internet browser, e.g., Internet Explorer (http://www.microsoft.com/ie), Netscape (http://browser.netscape.com), Firefox (http://www.mozilla.org/firefox), or Safari (http://www.apple.com/safari)
Table 3.4.2  Advanced Parameters for BLAST Searches

<table>
<thead>
<tr>
<th>Short name on Web site</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General parameters</strong></td>
<td></td>
</tr>
<tr>
<td>Max target sequences</td>
<td>The highest number of target (database) sequences to be displayed. Default: 500.</td>
</tr>
<tr>
<td>Short queries</td>
<td>Automatically adjust scores for short sequences. On by default.</td>
</tr>
<tr>
<td>Expect threshold</td>
<td>The highest expected frequency to be displayed. Default: 10.</td>
</tr>
<tr>
<td>Word size</td>
<td>Word size (integer value). Default for proteins is 3 amino acid residues, minimum is 2.</td>
</tr>
<tr>
<td><strong>Scoring parameters</strong></td>
<td></td>
</tr>
<tr>
<td>Matrix</td>
<td>The amino acid similarity scoring matrix (see text). Default: BLOSUM62.</td>
</tr>
<tr>
<td>Gap Costs</td>
<td>Penalties for gaps to be subtracted from the score: for existence (opening), default: 11, for extension, default: 1.</td>
</tr>
<tr>
<td>Compositional adjustment</td>
<td>Default: Conditional compositional adjustment.</td>
</tr>
<tr>
<td><strong>Filtering and masking</strong></td>
<td></td>
</tr>
<tr>
<td>Filter</td>
<td>For low-complexity regions, it is recommended to turn it on except when searching for identical and/or short sequences. Default is off.</td>
</tr>
<tr>
<td>Mask</td>
<td>Mask for lookup table only (no hit extension will be performed from masked regions).</td>
</tr>
<tr>
<td>Mask lower-case letters</td>
<td>Lower-case letters not aligned.</td>
</tr>
<tr>
<td>PS/PHI BLAST (position-specific iterative/pattern hit induced BLAST, see UNIT 3.11)</td>
<td></td>
</tr>
<tr>
<td>Upload PSSM (optional)</td>
<td>Browse for and upload a position-specific scoring matrix from your computer.</td>
</tr>
<tr>
<td>PSI-BLAST threshold</td>
<td>The upper expected frequency (E-value) threshold for including a new sequence into the next iteration, for the calculation of the new PSSM</td>
</tr>
</tbody>
</table>

**Files**

Sequence identifier or protein query sequence in FASTA format (see APPENDIX 1B)

1. Prepare the protein-to-protein search as described in Basic Protocol 1, steps 1 to 6.

**Set the Algorithm parameters section**

2. Click on “Algorithm parameters” (Fig. 3.4.2) to open this section (Fig. 3.4.3). When a parameter value is changed from its default, it will be highlighted in yellow.

3. Use “Maximal number of target sequences” when searching for distant relatives to display higher number of target sequences. Set this value for 250 and see how this field changes to yellow. However, for finding the closest relatives, it is sufficient to select a number between 20 and 50.

*Smaller numbers allow for faster execution and shorter outputs that are much easier to view. Note that the actual alignments are displayed following the one-line descriptions (Fig. 3.4.4C), although clicking on the one-line descriptions will position the browser to the alignments. For large outputs, automatic parsing becomes more practical than manual inspection; we recommend the BLAST package of the BioPerl Project (Stajich, 2007) for this purpose.*
4. Check “Short queries” to automatically adjust the word size (see below) for short queries. We recommend leaving this box checked.

5. Set “Expect threshold” to establish the expected frequency (number) of database hits with a score equal to or higher than the match between the query and the database sequence. An expected (absolute) frequency of 10 (the default value) means that with the given score or higher, 10 merely random matches can be expected in the database of given size (also see UNIT 3.3). This parameter causes the BLAST program not to report matches that are expected to occur more frequently on a purely random basis than the value specified here. By appropriately setting this argument, the user can delicately balance between sensitivity and the frequently daunting flood of output (see Critical Parameters).

Result size is a minor problem when only a handful of searches are involved, as one can just stop reading the results after seeing the first couple of irrelevant matches; however, for large-scale projects, such results can easily fill up the available disk space and downgrade the performance of the possible relational database. Care should be taken with XML files (Møller and Schwartzbach, 2006), which can be orders of magnitude larger than plain files.

6. (Advanced) Set the “Word size” (the length of words or seeds, identically matching segments between the query and the database sequence for extending an alignment in BLAST to two (the only alternative) when no significant matches are found running BLAST with a default Word Size of three (recommended).

Shorter words increase sensitivity at the expense of execution time. The Web-based version of BLAST does not allow specifying words longer than three residues. When using command-line BLAST in the quest of nearly exact matches, increasing the word size can notably accelerate searches.

Set the Scoring Parameters section

7. Select a protein-scoring “Matrix” (see Critical Parameters and UNIT 3.5). For most purposes, the default BLOSUM62 matrix (Henikoff and Henikoff, 1992) is the best solution.

8. “Gap costs” are combinations of gap opening and gap extension penalties and can be specified from the pull-down menu. Only a few combinations of scoring matrices and gap costs are allowed. NCBI’s recommendations for the selection of the scoring matrix and the gap penalty values (see Critical Parameters and UNIT 3.5) as a function of query length are shown in Table 3.4.3.

9. Set “Compositional adjustments.” The standard scoring matrices, including the BLOSUM series (Henikoff and Henikoff, 1992) reflect only a limited sample of the wide spectrum of amino acid compositions of proteins.

Table 3.4.3 Optimal Combinations of Scoring Matrices and Gap Opening and Extension Cost as a Function of Query Length

<table>
<thead>
<tr>
<th>Query length (residues)</th>
<th>Scoring matrix</th>
<th>Gap opening cost</th>
<th>Gap extension cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;35</td>
<td>PAM30</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>35-50</td>
<td>PAM70</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>50-85</td>
<td>BLOSUM80</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>&gt;85</td>
<td>BLOSUM62</td>
<td>11</td>
<td>1</td>
</tr>
</tbody>
</table>

aValues as recommended by NCBI. See Critical Parameters for further discussion.
Several proteins are rich in a few types of residues. For example, in most proteins, cysteines form disulfide bridges and, hence, are highly conserved during evolution. This conservation is reflected in the high score for cysteine identities in the BLOSUM (Henikoff and Henikoff, 1992) and other scoring matrices. Such high scores would produce low expected frequencies for a hit between two unrelated cysteine-rich proteins.

DNA-associated proteins are often rich in arginines and lysines. Thus, matching arginines may occur quite frequently, even between two unrelated proteins. These biologically less relevant matches would produce inflated scores and low expected frequencies.

Composition-based statistics reduce this effect and provide biologically more meaningful results.

When aligning two unrelated example proteins, both rich in arginine and lysine, many such residues may escape low-complexity filtering by the SEG program (Wootton and Federhen, 1996) as discussed below. blastp and position-specific iterative PSI-BLAST can apply a scaling procedure with a modified scoring system for each database sequence. The most important difference between this advanced scoring and the earlier procedure resides in assigning scores to identities reflecting the compositional bias in the protein database (Altschul et al., 1997, 2005; Schaffer et al., 2001).

**Apply desired parameters in the Filters and Masking section**

10. Turn on the “Filter low complexity regions” filter (recommended for most purposes) to avoid false positive matches. Several programs have been introduced to mask low-complexity regions (see annotation) by “X” characters that stand for unknown residues. Consequently, BLAST ignores masked regions from the estimation of expected frequencies. The default tool for masking amino acid sequences is Wootton and Federhen’s (1996) SEG program. As the extent of the low-complexity regions varies widely from protein family to family, a considerable proportion of sequences are not affected by filtering but some other proteins may be masked in their entirety.

Low-complexity means local or sequence-wide enrichment in one or more residues, e.g., glycine and proline in collagens, arginine and lysine in histones and other proteins that bind to nucleic acids, glutamine in huntingtin, and long runs of glutamine or other repeats that deform other proteins. Proline and serine are also extremely abundant in a variety of unrelated protein families. As the calculation of the expected frequency is based on much more random models, the match between two sequences overabundant in the same residues would produce extremely significant expected frequencies.

11. *(Advanced)* Use the “Mask for lookup table only” as a compromise between applying and not applying the low-complexity filter for queries with extended low-complexity regions. The lookup table is a special representation of the query sequence that accelerates searching for words (seeds) of matching regions in the database sequences. Eliminating computationally intensive lookup procedures with low-complexity words (seeds) like “AAA” or “QQQ” speeds up the searches. Once a matching word is found in a database sequence, the extension process will be performed on the unmasked query sequence.

12. *(Advanced)* Identify any regions that produce a mass of spurious and misleading hits by visually inspecting the pairwise (Fig. 3.4.4C) or multiple alignments (Fig. 3.4.9) and make them lower case, using a text editor. Then copy and paste the modified sequence in FASTA-format (*APPENDIX 1B*) into the “Enter accession number, gi, or FASTA sequence” field (Fig. 3.4.2), and resubmit the BLAST search.”

A region in the query sequence may produce matches to unrelated proteins, e.g., coiled coils, leucine-rich repeats, as well as ATP-binding, epidermal growth factor and myosin domains.

**Submit the search**

13. At this point, submit the search by clicking on the BLAST button.
REFORMATTING RESULTS FROM A BLAST SEARCH

This protocol provides an overview of select format options that are available for BLAST HTML output. Specifications for formatting can be entered after the results have been received (see Fig. 3.4.4A-C).

**Necessary Resources**

**Hardware**

Computer with Internet access

**Software**

Up-to-date Internet browser, e.g., Internet Explorer (http://www.microsoft.com/ie), Netscape (http://browser.netscape.com), Firefox (http://www.mozilla.org/firefox), or Safari (http://www.apple.com/safari)

**Files**

Protein query sequence in FASTA format (APPENDIX 1B)

1. Prepare the protein-to-protein search as described in Basic Protocol 1, steps 1 to 5, and Support Protocols 1, steps 1 to 6.

**Select format options**

2. When BLAST returns the search results, click on the link “Reformat these results” at the top center of the page (Fig. 3.4.4A) to move to the “Format Request” page (Fig. 3.4.8).

3. Use “Request ID” to enter a different RID number of a previously executed search or format operation. Check the box “Show results in a new window.”

**Select features in the Format section**

4. (Advanced) If the default alignment is kept, select HTML, Text, ASN.1, or XML formats from the Show menu for display and/or download. Alternatively, save Bioseq-formatted results to the user’s local computer or display as an input for the NCBI toolbox, or save in the perhaps the most readable multiple alignment

![Figure 3.4.8](image) The Format Request page. Note the Request ID number.
format, “Query-anchored letters for identities” (Fig. 3.4.9). Format as a “Hit Table” (Fig. 3.4.10) to provide a tabular display of the one-line descriptions and report the query sequence coverage, as well.

5. Select an “Alignment View” from the pull-down menu:

Pairwise (default). Standard BLAST alignment in pairs of query sequence and database match (Fig. 3.4.4C). This is the best display to evaluate the similarity between two sequences and is therefore recommended for most searches; however, when the conservation of individual positions is being analyzed across a family of sequences, select one of the query-anchored multiple alignment displays. For automated analysis, the results in tab-delimited format may serve as input to relational databases.

Query-anchored letters for identities. Figure 3.4.9 shows identities with single letter amino acid or nucleic acid codes. This is similar to the standard multiple alignment displays.

Pairwise alignment with dots for identities, Flat query-anchored with dots for identities, and Flat query-anchored with letters for identities. These are less practical than the default and “Hit Table” (see below) formats.

Hit Table. This tab-delimited format can easily be parsed to serve as an input for relational databases or programs making automatic selections on the basis of novelty, similarity and other features (Fig. 3.4.10). Note that the definition lines of the database sequences are not displayed in such tables, but these definitions can easily be retrieved by a manual or automated Entrez queries using the “gi” numbers or accessions.

Select features in the Display section

6. “Graphical Overview” represents the alignments by lines color-coded according to score (Fig. 3.4.4B).

7. “Link Out” is a color-coded, one-letter representation of the databases that include the matched database sequence (Table 3.4.4 and Fig. 3.4.4C).
Finding Homologs in Amino Acid Sequences Using Network BLAST Searches

### Figure 3.4.10
View of a “Hit table.” This view displays separate rows for each hit, with tab-delimited fields display the high-scoring segment pairs for each database sequence.

<table>
<thead>
<tr>
<th>Table 3.4.4 Color and Letter Coding for the Link Out Database Icons</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single letter representation</strong></td>
</tr>
<tr>
<td>U</td>
</tr>
<tr>
<td>G</td>
</tr>
<tr>
<td>S</td>
</tr>
<tr>
<td>E</td>
</tr>
<tr>
<td>M</td>
</tr>
</tbody>
</table>

8. “Sequence retrieval” checkboxes on left of the identifiers allow fast retrieval of the selected database sequences.

9. “NCBI-gi” causes the display of NCBI-gi identifiers in addition to the accession number.

**Select arguments**

10. “Masking character” and “Masking color” are self-explanatory.

11. Use “Limit results” to help reduce the tedium of manual analyses. Decrease the value for this argument to accelerate searches and make the output shorter.

   However, keep in mind that the top BLAST hit is not necessarily the closest homolog or a homolog at all in the sense of common evolutionary origin or biological function (see Guidelines for Understanding Results).

   Another reason to display several hits is the potential lack of meaningful annotations.

12. Customize the numbers of (one-line) “Descriptions,” “Graphical overview” entries, and “Alignments.”

13. Narrow the searched sequences using the “Organism” field.

   This is useful in the quest for close or distant relatives of the query sequence (see above).

14. *(Optional)* To limit the search database by a query referring to specific field codes and/or any words in the annotation, use “Entrez query” as described in Basic Protocol 1, step 7.
15. Use “Expect min” and “Expect max” to set the range for the expected frequencies of hits (see Support Protocol 1, step 5).

*Step 5 in Support Protocol 1 describes the usage of the expected frequency thresholds.*

16. Use “Format for” to generate input for PSI-BLAST (position-specific iterative BLAST; Altschul et al., 1997). From the database sequences that match the query with an E-value (expected frequency) specified in this text field, a multiple alignment is generated. In turn, the alignment is converted to a position-specific scoring matrix (profile), input for the PSI-BLAST search.

*PSI-BLAST is an extremely sensitive tool for the discovery of distant relatives that may escape detection by traditional, pairwise BLAST. Note that PSI-BLAST works on amino acid sequences only.*

**TRANSLATED BLAST SEARCHES**

It may be desirable to find homologs of a known protein in a different organism where many proteins and genes are either not known or not confidently predicted. If the genome, expressed sequence tags (EST; see APPENDIX 2), or partial genomic sequences are available, we can conceptually translate these nucleic acid sequences into conceptual amino acid sequences. The translation is performed in both directions and for each direction, in all the three reading frames. Then, the alignments are generated at the amino acid level. Since proteins are, as a rule, more conserved than exons (due to synonymous codons), translated BLAST searches are more sensitive and selective than pure nucleic acid BLAST (blastn) searches.

Translated BLAST searches (Gish and States, 1993), now implemented in all BLAST versions, come in three flavors. In blastx, a nucleotide sequence is searched against a protein database. blastx is the best tool for finding genes in a genomic sequence, characterizing a single EST, or for the assembly of several ESTs that are believed to be reverse-transcribed from the same mRNA. In tblastn, a protein query is searched against a nucleotide database translated in six frames. A major use of tblastn is finding homologs of an already known protein in a genome or in a collection of ESTs. tblastx translates both the query and the database sequences. This extremely CPU-intensive application can compare an uncharacterized gene candidate of one organism (e.g., human) to the genome of another species (e.g., mouse). The DNA should be derived from closely or moderately distantly related species in order to produce biologically meaningful comparisons. As a rule, genomic sequences of species within the same class (e.g., mammals) can lead to reliable gene predictions. Because these searches are submitted in an identical way, this protocol describes the specifics for blastx searches only. Interpretations of results for all types of searches are discussed in the Guidelines for Understanding Results.

**Necessary Resources**

**Hardware**

Computer with Internet access

**Software**

Up-to-date Internet browser, e.g., Internet Explorer (http://www.microsoft.com/ie), Netscape (http://browser.netscape.com), Firefox (http://www.mozilla.org/firefox), or Safari (http://www.apple.com/safari)

**Files**

Nucleotide or protein query sequence in FASTA format (see APPENDIX 1B)
Finding Homologs in Amino Acid Sequences Using Network BLAST Searches

3.4.20

3.4.20

Figure 3.4.11  The top page for translated BLAST searches at the WebBLAST server at NCBI. This screen appears when the blastx program is selected.

1. Navigate to http://blast.ncbi.nlm.nih.gov (Fig. 3.4.1). Choose from one of the three different kinds of translated BLAST searches displayed, where either or both the query and the database sequences are translated in all six frames:

   blastx. Search a protein database using a translated nucleotide query.
   tblastn. Search a translated nucleotide database using a protein query.
   tblastx. Search a translated nucleotide database using a translated nucleotide query.

   Technically, these searches are submitted in an identical way, therefore we show protocol for blastx searches only. The specifics of interpretations will be discussed in the Guidelines for Understanding of Results.

   No matter which program is selected, the user will be directed to a page (Fig. 3.4.11) very similar to the blastp home page. The major differences are that the query is a nucleic acid sequence and the drop-down menu for the selection of the genetic code.

2. Select “blastx” to find new genes in a genomic region or in mRNA or EST sequences (query) that code for proteins similar to those in the search database.

3. Either enter an identifier (accession or gi number), copy and paste the nucleotide or protein query sequence into the Search text area in FASTA format (see Appendix 1b), or upload a sequence from the disk on your computer.

   As an example, let us experiment with a segment of the largest human gene, the mutants of which are associated with Duchenne muscular dystrophy. This gene occupies as many as 2.4 million bases on human chromosome X (identifier in the NCBI Genome Database is NC_000023.9, indicating Version 9 of the chromosome); see Fig. 3.4.12.
4. Type or paste NC_000023.9 into the “Enter accession number, gi, or FASTA sequence” text area.

5. Limit the search to base pairs 33100000 through 33267647 by entering these numbers into the Start and End text fields, since a blastx search with such an extensive query sequence would likely time out on the NCBI server.

Also consider that very long queries (over 10,000 nucleotides) produce noisy results as discussed in Guidelines for Understanding Results. Since a number of mammalian genes extend to hundreds of thousands of base pairs, their analyses require automated or semi-automated pipelines.

6. Select the “Genetic code.” The default is the best choice for eukaryotic nonmitochondrial genes. For mitochondrial, macronuclear, and certain bacterial genes or genomes, make the appropriate selection.

7. Make further selections as described in Basic Protocol 1, steps 3 and 7, and set “Algorithm parameters” as in Support Protocol 1. Note that low-complexity masking will be applied not to the nucleotide query but to its six-frame translations.

In the refinement process, when running BLAST again, we recommend setting the subsequence to a single gene region, if any was found. This will save the user a lot of frustration and wasted time when evaluating the results.
8. In order to avoid the sometimes hundreds of pages of useless output, leave the Low-Complexity filter turned on.

When probing genomic sequences, noncoding segments may produce an extremely noisy output obscuring biologically interesting matches to real genes (see Guidelines for Understanding Results; Fig. 3.4.13).

It may be necessary to carefully mask the Alu, SINE, LINE, and other elements of the human genome that occur in very high numbers, using RepeatMasker (A.F. A. Smit and P. Green, unpubl. observ.). This can be performed on the Web site at the Institute for Systems Biology (http://www.repeatmasker.org) or by downloading the software for use on a local machine.

An alternative method is the Censor tool, accessible via the Web server and download site of the Genetic Information Research Institute at http://www.girinst.org/Censor_Server.html (Jurka et al., 2005).

9. Submit the search by clicking on BLAST. The search indicates only a few sequences, primarily proteins of unknown functions and noncoding or frameshifted regions indicated by the STOP codons, denoted by asterisks (*; Fig. 3.4.12).

This is in sharp contrast to the numerous and highly relevant hits that one can obtain by running a blastp search with the dystrophin protein (NP_000100.2) as query (Fig. 3.4.13). Fortunately, translated BLAST searches with shorter and fewer introns are more productive than this extreme example.
b2seq FOR COMPARING TWO SEQUENCES

Frequently, only two sequences are to be aligned, e.g., two suspected homologs, a gene and its transcript, a gene and the protein for which it codes, the transcript and the protein. In such a case, performing a full database search would be a waste of time. Instead, the b2seq program (Tatusova and Madden, 1999) can generate the local alignment or alignments much faster and with less noise. b2seq can be run either on the command line or on the Web page http://blast.ncbi.nlm.nih.gov. Selecting “Align two sequences using BLAST (b2seq)” close to the bottom of the screen (Specialized BLAST section) will redirect the user to the b2seq home page (Fig. 3.4.14).

Necessary Resources

Hardware

Computer with Internet access

Software

Up-to-date Internet browser, e.g., Internet Explorer (http://www.microsoft.com/ie), Netscape (http://browser.netscape.com), Firefox (http://www.mozilla.org/firefox), or Safari (http://www.apple.com/safari)

Files

Two nucleotide or protein (in any combination) query sequences in FASTA format (see APPENDIX 1B) or (alternatively) the accession or gi numbers

1. Choose the appropriate program from Table 3.4.5. For this exercise, compare two protein sequences by selecting “blastp” from the BLAST Web site (http://blast.ncbi.nlm.nih.gov).

Figure 3.4.14  Launching b2seq to perform BLAST comparisons of two sequences.
Finding Homologs in Amino Acid Sequences Using Network BLAST Searches

2. Copy and paste both sequences into a text area, select the sequences by specifying an accession code or a gi number, or upload the sequences from a file residing in the user’s computer. For this example, enter the accession numbers for the human haptoglobin (AAA88080) and complement C1r-B subcomponent precursor (gi number: 76364095).

3. Leave all settings at the default values and click on “Align.” The alignment (Fig. 3.4.15) covers 357 amino acid residues of the 406 residue haptoglobin (the protein with the fewer residues). A high score of 189 bits and an E-value of 4e-46.
Table 3.4.5  BLAST Programs for Comparing Two Sequences

<table>
<thead>
<tr>
<th>Program</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>blastp</td>
<td>Search protein database using a protein sequence as the query</td>
</tr>
<tr>
<td>blastn*</td>
<td>Search nucleotide database using a nucleotide sequence as the query</td>
</tr>
<tr>
<td>blastx*</td>
<td>Search protein database using a translated nucleotide sequence as the query</td>
</tr>
<tr>
<td>tblastn</td>
<td>Search translated nucleotide database using a protein sequence as the query</td>
</tr>
<tr>
<td>tblastx*</td>
<td>Search translated nucleotide database using a translated nucleotide sequence as the query</td>
</tr>
</tbody>
</table>

*If analyzing nucleotide sequences, specify whether both, forward- (only), or reverse- (only) strand translations should be applied.

are reported, with 32% of the residue pairs being identical and 50% of them having positive scores.

It is very likely that the two proteins descended from common ancestors (in other words, they are homologous). This is also indicated by the shared trypsin-like serine protease domains.

However, even such a statistically significant E-value does not indicate common function. While the complement Clr-B subcomponent is a bona fide serine protease, haptoglobin probably diverged from the ancestral protease and lost its catalytic function completely. Haptoglobin has evolved into a blood plasma glycoprotein that binds free hemoglobin, thus preventing oxidative damage (Carter and Worwood, 2007).

GUIDELINES FOR UNDERSTANDING RESULTS

While launching BLAST searches on the user-friendly NCBI Web servers may be a relatively easy task, interpretation of the results may require some experience and often a number of additional tools. It should be clear that unrelated proteins may match with an expected value of $1 \times 10^{-20}$ or even better, although establishing universally valid thresholds remains an elusive task. Similar cutoff values must balance between false positive and false negative matches. Note that amino acid sequences are even less random than nucleotides, partly due to the ambiguity of the genetic code. Sequences of amino acids evolve under considerable biological constraints. Some examples include the dominance of glycine-X-proline motifs in collagens (where X stands for any residue); overabundance of basic residues in DNA-binding proteins; clustering of hydrophobic residues in transmembrane segments and signal peptides; periodic alternation of hydrophobic and hydrophilic residues in amphiphilic $\alpha$-helices, and specific amino acid preferences in different structural elements, to mention just a few. All these factors decrease randomness and increase the chances that a subsequence of amino acids or its variation occurs more frequently in real proteins than expected on a purely random basis. This clearly illustrates that one should not equate sequence similarity with homology in the sense of common evolutionary origin of the two genes or proteins (Reeck et al., 1987).

The results of BLAST and several other bioinformatics tools are used as statistical inference from the observed degree of similarity to similarity of function or to the possible common evolutionary origin of the two molecules. Statistical inference, by its very nature, carries some uncertainty, which sometimes can be practically negligible, but in other frequent cases may warrant further examination. Statistical inference needs to be supplemented by a diverse arsenal of biological analysis, for example it is a good practice to check whether the top significant hits have similar functions. Also, because similar proteins may be annotated in the databases under different names, the user may need to follow the hyperlinks to the hits to read the detailed annotations. When the top proteins have genuinely diverse functions, this may indicate either that some of the matches may provide poor or ambiguous indications for the function, or that some of the
Finding Homologs in Amino Acid Sequences Using Network BLAST Searches

3.4.26

Supplement 25

Current Protocols in Bioinformatics

annotations of the database sequences could be incorrect. As an example of the first, the epidermal growth factor domain matches 889 proteins in the SMART database (Letunic et al., 2006). These molecules are as diverse on the functional spectrum as urokinase, thrombospondin, transforming growth factors, ookinete surface proteins, and others.

The manual analysis of raw BLAST output of hundreds of vast genomic regions could be extremely and unnecessarily challenging, as shown again by the example of the human dystrophin gene (Fig. 3.4.12). Such a large-scale analysis can and should be automated up to a specific point. The numerous semi-automatic approaches include reformattting the BLAST output in XML or tabular format or parsing the raw output using the BioPerl package (http://www.bioperl.org) or some other home-made program. The parsed fields can be input to relational databases like MySQL (Ullman, 2006), where hits can be retrieved using sophisticated queries over multiple database search and gene prediction results. For smaller projects, flatfiles (if they are not prohibitively large) can be used to store data for queries written in the Perl or Java languages.

When no significant hits are found

BLAST, like any other sequence analysis tool, may miss very distant relatives. Such distant relatives include proteins that share similar three-dimensional structures, like families in the Structural Classification of Proteins (SCOP) database (Andreeva et al., 2008). More than half of the distant relatives as defined by SCOP were undetected by pairwise BLAST (Hubbard et al., 1998). Unfortunately, a significant proportion of newly predicted proteins do not have known closer relatives either. In such cases, before applying any other method, double-check the quality of the sequence, as frameshift errors or inaccurate exon boundaries frequently corrupt the amino acid translations.

If correcting for such errors fails, probing the databases by PSI-BLAST can be the next step. This program can also be launched from the NCBI Web server by checking the Format for PSI-BLAST field in the “Reformat results” screen. The caveat is the query-specific selection of the inclusion threshold as values that are too high can cause PSI-BLAST to include sequences for which functional or evolutionary relationships are not established into the multiple alignment underlying the position-specific scoring matrix (PSSM) calculations (UNIT 3.11). Another possibility is to run conserved domain searches (selectable from the top screen), which may reveal diagnostic domains. Searching against databases of hidden Markov models (HMMs) (Krogh et al., 1994) of protein domains—such as Pfam (Finn et al., 2008) and SMART (Letunic et al., 2006)—can provide higher levels of sensitivity.

COMMENTARY

Background Information

Optimizing BLAST results

The simplicity of running BLAST using the default search arguments values frequently misleads scientists new to this field. By using defaults for all tasks, one may get either sensitivity that is too low or selectivity that is too low, leading to more output than necessary. Much more effort and time will be needed for interpreting results which have not been optimized than for learning how to fine-tune the sensitivity, output format, and number of descriptions and alignments to be displayed. Consider that chances are high that scientists working with protein and nucleic acid sequences will use BLAST quite frequently. The time invested into learning how to optimize this powerful tool is amply rewarded.

Even if the BLAST results are produced by a co-worker, the interpretation may remain for the biologist, whether he or she is a bench scientist or a computational biologist. This interpretation means that, among other things, the researcher will have to decide if it is possible to make an informed decision on whether the results support accepting the matching sequence as a functional or structural relative, marginally significant results might warrant further analyses by other tools, or the match is so poor that spending resources on further
A model pipeline

To discover new genes coding for a specified class of proteins (e.g., G-protein coupled receptors) in a database of genomic DNA, the following simple pipeline can be constructed. Compile a representative set of the protein class or import it by anonymous FTP from Swiss-Prot (ftp://ftp.expasy.org/databases/swiss-prot/special_selections). First, select a representative sample of the protein class. One-by-one, probe these proteins against a genome’s database using the tblastn program in order to find similar conceptual translations from the target genome. We expect that the underlying gene candidates may code for proteins of similar function. Select the boundaries of a gene candidate that significantly match one of the probe proteins. Note that there may exist some other proteins that are more similar to the gene candidate than the probe sequence; therefore, it is mandatory to find the best match for the candidate gene by searching it against the nr database of proteins using the blastx program. It is necessary to confirm that the gene candidate is the highest scoring match to the protein probed, in other words, these may be each other’s closest homologs, provided that homology can be inferred from similarity (see Interpretations of Results). If closest similarity is confirmed, the gene can be compared to the highest scoring protein using the GeneWise program (Birney et al., 2004). For larger scale projects, the above process should be automated by means of a pipeline, including a parser, e.g., the BLAST-parser in the BioPerl package at http://www.bioperl.org (Stajich, 2007).

Translated BLAST searches

While the complete sequence of the human (Lander et al., 2001; Venter et al., 2001) and a number of other genomes are known, only fractions of the proteome have been identified. The reason for this discrepancy is that sequencing DNA or complementary DNA (cDNA) is far more productive and much more economically attractive than sequencing proteins. Until the proteome is determined, protein and gene discovery, as a rule, are confined to starting from nucleic acid sequences. The challenge is to find genes in vast regions of genomic sequences. Unfortunately, improving the accuracy (Bajic et al., 2006) of gene predictions (Uberbacher et al., 1996; Burge and Karlin, 1997; Solovyev et al., 2006; Stanke et al., 2006) from genomic DNA depends on the availability of closely related sequences. Not surprisingly, these relatives are identified by BLAST searches; however, as discussed in the introduction of UNIT 3.3, nucleic-acid-to-nucleic-acid comparisons are significantly less sensitive than comparisons of two amino acid sequences.

One may hope to resolve these problems by translating the query nucleotide sequence (or the database) in all the six reading frames, then running six independent searches. Unfortunately, sequencing errors like missing or added bases can shift the frame, corrupting the translation downstream and potentially missing true hits. A much better and easier solution is to perform a single BLAST run where either the query or the database sequences are translated on the fly and, for each database sequence, the significant matches for all the six frames are reported together. This approach facilitates reporting the match upstream and downstream of the frameshift error, although in different reading frames. Naturally, these matches are shorter and their expected frequencies are higher (less significant) than for uncorrupted sequences. The special considerations for evaluating such noisy data are discussed below (see Critical Parameters).

When using translated BLAST searches, two considerations should be kept in mind. First, translated searches produce much more noise than protein-to-protein searches; therefore, it is preferable to use blastp (protein-to-protein) when learning BLAST. This is followed by nucleotide-to-nucleotide (blastn), and finally translated searches (tblastn, blastx, and tblastx). The other issue is that a search in six frames increases the CPU requirement as compared to a blastp comparison of the two (known) translations. Although a single lookup table is created, the increase remains significant. The optimal alignment of two nucleotide sequences by tblastx takes even more CPU time. The largest increase, however, results from the circumstance that generally, translated BLAST is not confined to coding regions but applied to vast genomic sequences or complete genomes.

BLAST servers

Fortunately, there are several network servers for performing BLAST searches and retrieving the results. Most comprehensive
bioinformatics packages include BLAST network applications. We highly recommend learning BLAST on the National Center for Biotechnology Information’s (NCBI) WebBLAST server at http://blast.ncbi.nlm.nih.gov, which also provides hyperlinks to nucleotide and protein databases including the Protein Data Bank of three-dimensional structures (Berman et al., 2000), as well as to PubMed (UNIT 1.3). Different Web applications have been developed to integrate BLAST with, e.g., domain predictions, EST information, gene predictions, and structural data.

If the BLAST package and the databases are not installed in the user’s institution, network BLAST remains the only choice. If these resources are available locally, an intranet BLAST server can easily be installed on a local common gateway interface (CGI; Stein, 1998) or Java (Liang, 2006) server.

**Software and hardware requirements**

The NCBI BLAST Web server can be accessed using a browser like Microsoft Internet Explorer, Firefox, Safari, or Opera and an Internet connection. Another option is to install a local client-server application (netblast) that may be obtained by anonymous FTP from http://www.ncbi.nlm.nih.gov/blast/download.shtml. Clients for Macintosh OS X, Microsoft Windows, LINUX, and Solaris (for the latter three operating systems, both 32- and 64-bit versions) are available. At the time of this writing, the Standalone Web BLAST Server is not available for Windows servers.

**Other CPBI units concerning BLAST**

This unit is based on knowledge of Standalone BLAST for nucleotide and protein sequences. It also frequently cites UNIT 3.3 (Network BLAST Searches for Nucleotide Sequences). We recommend reading the Background Information section of UNIT 3.3 regarding the types, advantages, and disadvantages of network searches, as well as when it is reasonable to choose a public network BLAST service over local applications. We also refer the reader to an excellent textbook that covers pairwise alignments, BLAST, and a wider spectrum of bioinformatic tools and approaches (Baxevanis and Ouellette, 2005).

**Critical Parameters**

**Basic parameters**

**Expect frequencies**

Controlling the output exclusively by the Number of Descriptions and the Number of Alignments arguments at the “Algorithm parameters” screen would cause BLAST to report biologically uninteresting matches when the specified number of alignments exceeds the number of interesting hits. The default expected frequency threshold is 10, an overly nonconservative value that instructs the program to report any hit of potential interest.

At the other extreme, always relying on the top hit would be a risky proposition. The top hit may be an open frame with no known function or, worse, may have an incorrectly predicted function. When the function of the best hit is radically different from those of the second, third, . . . nth best matching database sequences, chances are that the top hit was assigned an incorrect function. Always inspect a minimum of ten hits. If necessary, increase this number so as to obtain a three-quarter consensus on the function. If most of them have no clear function or the annotations contradict each other, analyze as many hits that facilitate a safe consensus of the function, whenever possible.

Expected frequencies should play a key role in the interpretation of the search results. These issues are discussed elsewhere (see Guidelines for Understanding Results; Baxevanis, 2005).

The sensitivity of the blastp program for changing the arguments is shown in Figure 3.4.3. Note that most arguments change the number of matches to be reported or the format of the output. We do not recommend changing the gap penalties or the drop-off values.

**Selecting a scoring matrix**

BLAST, like most other tools that involve sequence alignments, assigns differential scores to various amino acid matches or mismatches based on their biological relevance (UNIT 3.5). For example, substituting an isoleucine for a valine residue within a hydrophobic segment usually has minor effects on the proteins; however, a glycine residue (the smallest amino acid) can rarely be substituted for the bulky tryptophan in β-turns, and cysteines forming disulfide bridges are also highly conserved. Matches of identical residues are not equal either; the coincidence of rare events, e.g., matching tryptophan residues (which would be deleterious in most sequence positions), carries more information than coincidences of frequent events like matching two alanine residues (which are well tolerated in most sequence positions (Dayhoff and Eck, 1968). These fine distinctions account for the sensitivity of protein-to-protein
BLAST searches. The scores are arranged to matrices input to the BLAST program.

According to the experiments conducted by Altschul et al. (1994) and many others, the best balance between sensitivity and selectivity can be achieved by using the BLOSUM62 scoring matrix (Henikoff and Henikoff, 1992), provided that the query sequence extends to more than 85 residues. For shorter sequences, matches need to be assigned higher scores and mismatches lower scores as compared to BLOSUM62 so as to eliminate false positives. The NCBI guidelines for selecting a scoring matrix are presented in Table 3.4.3. Additional information regarding selecting a protein-scoring matrix can be found in UNIT 3.5.

**Gap penalties**

Introducing gaps into the sequences may be necessary so as to achieve the best alignment (UNIT 3.1). However, if gaps were not penalized, unrelated sequences would produce high-scoring alignments with abundant long gaps and short matches. To avoid such biologically meaningless alignments, the opening and the extension of a gap must come at a cost. The cost for a gap is calculated as the gap opening cost plus the gap extension cost multiplied by the length of the gap. As this cost is subtracted from the score of the alignment, each matrix must have a specific gap cost combination (Table 3.4.3).

**Improving the signal-to-noise ratio in translated BLAST searches**

Translated BLAST searches (Gish and States, 1993) have intrinsically worse signal-to-noise ratios than protein-to-protein searches. When analyzing large genomic sequences, it may save annotation time and effort to run BLAST again with a query DNA sequence to a segment pruned to a single gene, decreasing both the noise and the size of the output. In this second run, another way to save effort is to limit the search to the already known direction of the strand. However, we do not recommend specifying the frame because of possible frameshift errors in the sequence. Such errors can cause the matches between an exon and the amino acid sequence to be split into two high-scoring pairs (HSPs), each with possibly much less significant expected frequency than the correct full-length exon. Note that short exons may frequently escape detection even in the absence of frameshifts. Other sequencing errors may produce false stop codons that can also split or shorten the matches between the exon and the corresponding protein segment.

Another case requiring extra care occurs when the database contains only fragments of the protein coded by the gene in the query genomic segment. Even if all matches are 100% identical between the nucleotide and the amino acid sequences, some of them, particularly the short ones, may produce higher expected frequencies than matches to more distant homologs for which the complete protein is known. Consequently, matches to short fragments may be found at the bottom of the output (see Suggestions for Further Analysis).

**Advanced parameters**

**Invoking the mask lower-case letters filter option**

Biological sequences frequently include segments that match unrelated or very distantly related sequences with a higher score than relatives with similar biochemical function. Consequently, such high-scoring hits are displayed closer to the top of the output, rendering its evaluation more tedious. The proteome is abundant in elements highly dispersed during protein evolution—e.g., coiled-coil segments, leucine-rich repeats, ancient domains (e.g., EF-hand), myosin, ankyrin, and many others—hence eliminating the elements from the output and from the calculation of scores may be necessary. For example, it is possible to predict coiled-coil regions of the web site http://www.ch.embnet.org/software/COILS_form.html. The predicted coiled segments may be rewritten in lower-case letters with a Perl script or with any text editor (APPENDIX 1C). Repeating the BLAST search using the “Mask Lowercase Letters Filter” argument may provide better results than without such masking.

Another situation for lower-case filtering is found in moderately cysteine-rich proteins. Such proteins are rich in unconserved cysteine residues that may produce deceptively significant expected frequencies, as cysteine to cysteine matches are assigned the second highest score both in the BLOSUM62 (Henikoff and Henikoff, 1992) and the PAM120 (Dayhoff and Eck, 1968) matrices (see Scoring matrix, above), while mismatches to cysteines are harshly penalized. These scores reflect extracellular proteins where most of the cysteines form disulfide bridges that are crucial in maintaining the tertiary or quaternary structure of the multisubunit proteins. This is not
the case for cysteine-rich proteins, and high match scores may generate a number of false positives. One can easily avoid getting such false positives by using the lower-case filter to mask such cysteine residues.

**Position-specific scoring matrices (PSSM)**

These matrices are specific representations of multiple alignments, differing from the BLOSUM and PAM matrices discussed above in having a separate column for each position in a multiple alignment. Each column has 20 elements, one for each of the residue types. If, for example, a histidine residue is conserved at a given position of the sequence, the element in the intersection of the histidine row and the column for the particular position will be assigned a high positive value. Other elements of the same column are assigned negative scores. In less conserved positions, multiple residues may be assigned positive scores. For a detailed discussion of scoring matrices and gap penalties, see Baxevanis (2005).

**Suggestions for Further Analysis**

**User interfaces for manual annotations**

While certain automated analyses (e.g., assessing novelty) are tremendously helpful, in our opinion, the fully automated prediction of gene structures and biological functions is not realistic, given the current state of the art. There seems to be no way to escape time-consuming visual annotations, but, fortunately, several user interfaces have been constructed to increase the productivity and accuracy, as well as the aesthetic quality of the sometimes daunting work of annotations. A highly efficient strategy for gene discovery has been to integrate searches of a segment of genomic DNA against the NCBI nr protein database, the RefSeq human mRNA collection, and databases of human, mouse, and other ESTs. In order to reveal the exon structure of a new gene, ab initio gene predictions (Uberbacher et al., 1996; Burge and Karlin, 1997; Solovyev et al., 2006; Stanke et al., 2006) and GeneWise (Birney et al., 2004) than by BLAST. sim4 (Florea et al., 1998) and lap (Huang and Zhang, 1996) can generate either nonoverlapping local or global alignments, provided that the sequences to be aligned match with a high level of identity. Frameshift errors can be corrected by translated dynamic programming searches (Smith and Waterman, 1981), preferably accelerated on a specialized, highly parallel architecture.

**Taxonomy reports**

It can be challenging to review the complex taxonomic distribution of hits for families of proteins conserved over eons of evolution, e.g., copper-zinc superoxide dismutase. The taxonomic hierarchies are presented in a user-friendlier format on the Taxonomy links at the top of the BLAST result pages. Clicking there will display the scientific binomens, the common names, and part of the classification of the organisms (Fig. 3.4.5). Note the taxonomy codes used by NCBI, which can easily be used for retrieving hits to proteins or nucleic acids from a specific taxon (see http://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html).
Acknowledgements
I am grateful to Dr. Jean-Jack M. Riethoven for comments. This study was supported by NSF Grant EPS-0701892.

Literature Cited


Finding Homologs in Amino Acid Sequences Using Network BLAST Searches

3.4.32


**Key References**

Altschul et al., 1994. See above.  
*Probably the best description of the BLAST program that produced nongapped alignments at that time. This review discusses the underlying statistics and their biological interpretation, the scoring schemes, the search, the sensitivity, and selectivity on biological examples.*

Altschul et al., 1997. See above.  
*The original research paper on gapped and PSI-BLAST. Both are significant improvements over earlier BLAST versions. Computational speed, increased sensitivity, and decreased selectivity are analyzed.*

Baxevanis and Ouellette, 2005. See above.  
*A widely taught textbook that introduces pairwise sequence similarity searches, biological databases and many other areas of bioinformatics. Reviews the general concepts of alignments, scoring matrices and BLAST with practical applications and guidelines for interpretation.*

Gish and States, 1993. See above.  
*Another original research paper, this one about translated BLAST. The authors evaluate the advantages and pitfalls of this application when processing introns, frameshifts, and similar issues. Besides the theory, implications on statistical significance are illustrated on examples.*

Korf et al., 2003. See above.  
*An excellent overview of theory and practice of the BLAST tools as of 2003. This most comprehensive and easy-to-understand textbook is highly recommended to everyone in bioinformatics or computational biology.*

**Internet Resources**

The NCBI BLAST Web site.

The full documentation for BLAST at NCBI.

http://www.ebi.ac.uk/blast2  
The European Bioinformatics Institute Server for the Washington University BLAST.

http://repeatmasker.genome.washington.edu/cgi-bin/RepeatMasker  
The RepeatMasker Web site.

http://www.girinst.org/Censor_Server.html  
The Genetic Research Institute Web site.

http://www.ch.embnet.org/software/COILS_form.html  
Coiled coil predictions.
Selecting the Right Protein-Scoring Matrix

OVERVIEW

Every program for searching protein sequences against a database includes a choice of a “protein-scoring matrix,” also called a “weight matrix.” Weight matrices add sensitivity to the search, while statistical significance adds selectivity (see UNIT 4.1). Virtually every user chooses the default, typically PAM 250 or BLOSUM62. Despite the fact that the choice of matrix can strongly influence the outcome of the analysis, most users do not know why a particular matrix should be used. In general, scoring matrices implicitly represent a particular theory of protein sequence evolution. This unit provides guidance in the choice of a scoring matrix, as understanding the assumptions underlying the PAM and BLOSUM scoring matrices can aid in making the proper choice. The selection of PAM matrices is covered first, after which the selection of BLOSUM matrices is discussed, and finally a brief overview of the wide variety of specialized scoring matrices is provided.

PAM MATRICES

PAM, a rearranged acronym derived from Accepted Point Mutation (Dayhoff, 1978) is a probabilistic model for amino acid replacement derived by comparing the frequencies of replacement in closely related sequences to the frequency expected from the completely random replacement of amino acids. The basis of this scoring system is the observation that the evolution of protein sequences is a nonrandom process—i.e., some amino acid replacements occur much more frequently than others, especially in related sequences. Amino acid substitutions tend to conserve charge, size, and hydrophobicity among other characteristics. One would expect that the substitution of glycine for alanine (CH$_3$ versus H) would have less of an effect on a protein’s structure and function than the substitution of alanine for threonine (CH$_3$ versus substituted indole ring). The inference is that if two aligned sequences manifest a higher than expected prevalence of these characteristic replacements, the sequences are related. An excellent discussion of the derivation and use of the PAM matrices is given in George et al. (1990).

PAM matrices are the result of computing the probability of one substitution per 100 amino acids, called the PAM 1 matrix. Higher PAM matrices are derived by multiplying the PAM 1 matrix by itself a defined number of times. Thus, a PAM 160 matrix is the result of performing 160 matrix multiplications of the PAM 1 matrix against itself. Similarly, the PAM 250 matrix is derived by multiplying the PAM 1 matrix against itself 250 times.

Biologically, the PAM 50 matrix means that in 100 amino acids there have been 50 substitutions, while the PAM 250 matrix means there have been 2.5 amino acid replacements at each site (see UNIT 3.1 regarding insertions and deletions). This sounds unusual, but remember that over evolutionary time, it is possible that an alanine was changed to a glycine, then to a valine, and then back to an alanine. These silent substitutions are derived from observed amino acid frequency data in protein families and superfamilies.

Choosing a PAM Matrix

It is extremely important to note that PAM matrices are derived from protein sequence data available in the late 1960s and early 1970s. Most proteins known at that time were small, globular, and hydrophilic. If the researcher believes their protein contains substantial hydrophobic regions, such as membrane-spanning helices or sheets, the PAM matrices are less useful than others described in this unit. Dayhoff et al. (1978) were the first to define the terms protein family and superfamily. A protein family is defined as sequences 85% identical or greater to each other. A protein superfamily is defined as sequences related from 30% identical or greater to each other. A protein superfamily may contain many protein families. The user should be aware that while the terms “family” and “superfamily” are widely used in biology, most of the time the original definition of Dayhoff and collaborators is not being used (see below).

Locating all potential similarities: PAM 250

The most widely used PAM matrix is PAM 250 (Fig. 3.5.1). It has been chosen because it is capable of accurately detecting similarities in the 30% range (i.e., superfamilies), that is, when the two proteins are up to 70% different from each other (George et al., 1990). Another way to think about this is that the PAM 250...
matrix provides the best “look-back” in evolutionary time in a protein sequence comparison. If the goal is to know the widest possible range of proteins similar to the protein of interest, PAM 250 has been shown to be the most effective (Altschul, 1991). It is the best to use when the protein is unknown or may be a fragment of a larger protein. It would also be used when building a phylogenetic tree (Chapter 6) of the protein and exploring its relationship to other proteins.

Determining if a protein sequence is a member of a particular protein family: PAM 160

Assume a protein is a known member of the serine protease family. Using the protein as a query against protein databases with PAM 250 will detect virtually all serine proteases, but it is also likely that a sizable number of other matches irrelevant to the researcher’s purposes will be located. In this case, the PAM 160 matrix should be used. It detects similarities in the 50% to 60% range (Altschul, 1991). In effect, it reduces potentially irrelevant matches.

Determining the most highly similar proteins to the query protein sequence: PAM 40

To reduce irrelevant matches even further, using a low-numbered PAM matrix (e.g., PAM 40) will find only those proteins most similar (70% to 90%) to the query protein sequence.

Figure 3.5.1 PAM 250 as a log-odds matrix. In a log-odds matrix, positive scores indicate a given pairing is more likely to occur in related sequences than in random ones—e.g., the score for matching a glutamate (E) and aspartate (D) is 3. Zero scores indicate the pairings are as likely to occur in related sequences as in random sequences. Negative scores indicate the pairings are more likely to occur in random sequences than in related ones. The observation that amino acids with similar physical chemical properties cluster together with positive values underscores the success of the approach to amino acid scoring.

Critique of the Dayhoff Model

Assumptions

Any mathematical model has assumptions built into it by the properties of the equations used to describe the system. For the PAM matrix the following assumptions exist:

1. Replacement at any site depends only on the amino acid at that site. More explicitly, it is a first-order Markov process (see Feller, 1968). All amino acid pairings were tallied independently of the rest.

2. Sequences that are being compared have average amino acid composition.

3. The position-dependent amino acid frequencies depend on the amino acid frequencies of the data set used to calculate the model. If a given protein has a significantly different composition, the probabilities of chance alignment will be different, invalidating the model.

Possible sources of error

The following is a list of reasons a PAM matrix may not properly predict similarity as expected:

1. The Markov process is an imperfect representation of evolution (violation of assumption 1). A Markov process assumes knowledge of the immediately preceding change. Evolution does not “know” what a previous sequence change has been, consequently, the assumption is violated.

2. Distantly related sequences usually have islands (blocks) of conserved residues. This...
implies that replacement is not equally probable over the entire sequence (violation of assumption 1).

3. Many sequences depart from average composition (violation of assumption 2).

4. Rare replacements were observed too infrequently to resolve relative probabilities accurately (e.g., for 36 pairs no replacements were observed).

5. Any errors in PAM 1 are magnified in the extrapolation to PAM 250, due to repeated matrix multiplications. A small error in PAM 1 will become a large error in PAM 250.

For the data available at the time, the Dayhoff model was quite successful. It was the standard matrix for scoring protein sequence alignments throughout the 1980s. This method was subsequently improved upon by the BLOSUM matrices discussed below.

**BLOSUM MATRICES**

BLOSUM is an acronym derived from Blocks Substitution Matrix by Henikoff and Henikoff (1992). They began with data from their Blocks database (Fig. 3.5.2; see UNIT 2.2 for a full description of the Blocks database). Blocks are conserved ungapped protein-sequence alignments. Different blocks may have widely varying similarity. In the derivation of PAM matrices, sequences that were represented many times were not excluded from the calculation. During the construction of BLOSUM matrices, measures were taken to avoid biasing the matrices by removing frequently occurring and highly related sequences. As the calculation was done in the early 1990s, the matrices also have a much greater representation of hydrophobic and nonglobular proteins.

A great many sequences in the blocks are identical or nearly identical. This creates a bias for evolutionarily close sequences in the matrix. To cope with this problem, similar sequences in a block above a specified threshold percent similarity are clustered, and members of the cluster count fractionally toward the final tally. This reduces the relative influence of closely related sequences and increases the influence of distantly related sequences in the final scoring matrix.

If the clustering threshold is 80%, the final matrix is BLOSUM80. Clustering at 62% (BLOSUM62) reduces the number of blocks contributing to the table by 25%. Even with this reduction in data, there are still $1.25 \times 10^6$ amino acid pairs contributing to the calculation of the scoring matrix. As the clustering threshold is decreased, the distantly related sequences have an even more increased contribution to the final matrix scores.

It is important to note, therefore, as the BLOSUM number decreases (i.e., BLOSUM80, BLOSUM60, BLOSUM50, BLOSUM30...), the ability to detect more distantly related sequences increases in a manner that parallels the effect of increasing the PAM distance (i.e., PAM 40, PAM160, PAM250...).

BLOSUM62 appears to be superior to PAM 250 in detecting distant relationships even if the PAM method is updated with current data sets (Altschul, 1991).

**Choosing a BLOSUM Matrix**

**Locating all potential similarities:**

**BLOSUM62**

The most widely used BLOSUM matrix is BLOSUM62. Analogous to the choice of PAM 250, it has been selected because it is capable of accurately detecting similarities down to the 30% range (i.e., superfamily), that is, when the two proteins are up to about 70% different from each other. The BLOSUM62 matrix provides the best look-back in evolutionary time in a protein sequence comparison. If the goal is to know the widest possible range of proteins similar to the protein of interest, BLOSUM62 has been shown to be the most effective, superior to PAM 250 (Altschul, 1991). It is the best to use when the protein is unknown or may be a fragment of a larger protein. It would also be used when building a phylogenetic tree of the protein and examining its relationship to other proteins.

**Determining if a protein sequence is a member of a particular protein family:**

**BLOSUM80**

Assume a protein is a known member of the serine protease family. Using the protein as a query against protein databases with BLOSUM62 will detect virtually all serine proteases, but it is also likely that a sizable number of other matches irrelevant to the researcher’s purpose will be located. In this case, the BLOSUM80 matrix should be used, as it detects identities at the 50% level. In effect, it reduces potentially irrelevant matches.

**Determining the most highly similar proteins to the query protein sequence:**

**BLOSUM90**

To reduce irrelevant matches even further, using a high-numbered BLOSUM matrix will find only those proteins most similar to the query protein sequence.
There has been considerable discussion about which is the “best” protein weight matrix to use. It should be apparent to the user that “best” depends on for what the user is looking. For hydrophobic proteins, hydrophilic proteins, very distantly related proteins, and very closely related proteins there are matrices that work well and others that work poorly.

A user may be presented with a program that only offers BLOSUM matrices when they would prefer to use a PAM matrix, or one that offers PAM matrices when they would prefer to use a BLOSUM matrix. Careful mathemati-
ical analysis based on an information theory measure called relative entropy (Altschul, 1991; Henikoff and Henikoff, 1992), showed that the following matrices are equivalent:

1. PAM 250 is equivalent to BLOSUM45.
2. PAM 160 is equivalent to BLOSUM62.
3. PAM 120 is equivalent to BLOSUM80.

Relative to the PAM 160 matrix, BLOSUM62 is less tolerant to substitutions involving hydrophilic amino acids, while BLOSUM62 is more tolerant to substitutions involving hydrophobic amino acids. Biologically, this is a reflection of the underlying data. In the derivation of PAM matrices in the mid-1970s, small, globular, hydrophilic proteins were used because it was possible to isolate that type of protein. In the 1980s, technology to isolate more hydrophobic proteins became available. Thus, BLOSUM matrices reflect the proportional change in the protein sequence database. What this in effect means is that although both PAM 250 and BLOSUM62 detect similarities at the 30% level, since BLOSUM uses much more recent data, and therefore a much wider range of proteins, PAM 250 is actually equivalent to BLOSUM45 when considering all proteins, not just those that are hydrophilic.

The user may note that some programs refer to the programs using “log-odds” versions of PAM matrices. This is a mathematical transformation performed because it is easier for computers to deal with very large or very small numbers in log form rather than as the plain number. For the PAM matrices, the conversion was done by dividing each entry in the matrix by the probability that two amino acids could be aligned by chance. One problem with the PAM model of evolution becomes apparent here. For six proteins, ≤35% similar pairwise, there would not be one amino acid conserved across all six sequences if all sites were equally mutable.

Biologically speaking, note that the essential features of protein sequence alignments are those regions that are conserved. In the process of developing the Blocks database (UNIT 2.2), Henikoff and Henikoff (1992) had this key insight. Instead of focusing on the mutations, they decided to focus on what is conserved and produce matrices from those regions.

SPECIALIZED SCORING MATRICES

The PAM and BLOSUM matrices described above are designed for the general case of comparing any protein sequence against any other protein sequence. In many cases, however, one wishes to use a protein from a specific family of proteins to detect other members of that family. Alternately, one may wish to detect specific features of a protein family—e.g., the 7-transmembrane helix family rather than any transmembrane protein. While BLOSUM and PAM matrices will work well in this case, they will also return irrelevant true positives—i.e., genuinely similar protein sequences, but not the specific family required.

In these cases, investigators have developed family-specific or superfAMILY-specific scoring matrices. There are a very large number (>600) of them in the literature. There are a number of situations in which one would prefer to use a protein superfamily or protein family-specific matrix. This discussion highlights a very small number of such matrices. If the specific matrix you would like to use has not been discussed, a PubMed (http://www.ncbi.nih.gov/pubmed) search of “amino acid substitution matrices” will be a good start. A recent review by Henikoff and Henikoff (2000) is also a good resource.

Updated PAM Matrices

An update to the PAM matrix using the method of Dayhoff et al. (1978) was created by David Jones and co-workers (1992). They examined 59,190 accepted mutations in 16,130 sequences. About 10 times the original data volume used.

The Gonnet Matrix

The Gonnet matrix is a scoring matrix based on alignment of the entire 1991 SwissProt database against itself (Gonnet et al., 1992). A total of 1.7 × 10^6 matches were used from sequences differing by 6.4 to 100.0 PAM units. This matrix has broad but selective coverage of protein sequences, because SwissProt covers only selected families. This matrix is very useful because of the excellent annotation of proteins included in SwissProt.

Matrix of Structural Substitutions

This scoring matrix was created based on observed substitutions of residues found in similar structural environments in crystallographic structures (Bordo and Argos, 1991).

Matrices for Detecting Frameshift Mutations

A group of matrices is available for detecting frameshift mutations, creating “new” coding sequences. These are also useful for data prone to sequencing errors (Claverie, 1993).
GPCR-Specific Matrix

This matrix is based solely on aligned blocks of G-protein-coupled receptors in which the elements of the matrix are proportional to the rarity of the substitution. This approach was used to construct a coherent phylogeny of widely diverged G-protein receptors (Kowlakowski and Rice, 1994).

Hydrophobicity-Based Matrices

There have been many scoring matrices proposed based on hydrophobicity of free amino acids. Discussing these is well beyond the scope of this unit. Briefly, each hydrophobicity matrix derives its hydrophobicity values in a different manner. Some use partitioning in octanol and water, or in other aqueous and nonaqueous solutions.

The standard reference is Kyte and Doolittle (1982). In such matrices, the “probability” of any amino acid becoming any other amino acid is considered to be proportional to the degree of change of hydrophobicity. Such scales are arbitrary and do not represent any formal mathematical basis for probability.

Literature Cited


Contributed by David Wheeler
Human Genome Center
Baylor College of Medicine
Houston, Texas
Constructing and Refining Multiple Sequence Alignments with PileUp, SeqLab, and the GCG Suite

The comparative approach is a keystone of biology and is tremendously powerful for discovering similarities between sequences in a dataset. Perhaps the best way to recognize these similarities is to prepare and analyze a multiple sequence alignment.

This unit uses a well characterized protein family, elongation factor 1α, to teach multiple sequence alignment and analysis using the Genetics Computer Group’s (GCG) Wisconsin Package (see Internet Resources) SeqLab graphical user interface (GUI). The progressive pairwise alignment program PileUp and several other GCG programs will be used to create the alignment and analyze the results.

The first protocol (see Basic Protocol 1) describes the use of the SeqLab GUI and of PileUp itself to construct and view a multiple sequence alignment. The next two (see Support Protocols 1 and 2) discuss how to compile and refine the initial list of sequences to be aligned. The initial list is compiled by the reference searching program LookUp, a Sequence Retrieval System (SRS) derivative (Etzold and Argos, 1993; see Support Protocol 1). Support Protocol 2 illustrates the use of GCG’s implementation of FASTA (Pearson and Lipman, 1988; Pearson, 1998) for increasing or decreasing the number of sequences to be aligned. Several refinement and annotation techniques, and the mask concept are also presented (see Support Protocols 3 and 4). Support Protocols 5 and 6 discuss methods for exporting multiple sequence alignments into formats compatible with two popular phylogenetic inference software packages, namely, the public domain package PHYLIP (see Support Protocol 6; also see Internet Resources and http://evolution.genetics.washington.edu/phylip.html; UNIT 6.3) and the commercial software suite PAUP* (see Support Protocol 5; also see Internet Resources; http://paup.csit.fsu.edu/index.html; UNIT 6.4), available through Sinauer Associates (http://www.sinauer.com) or as provided within GCG.

Two additional basic protocols relate to motif discovery and dataset annotation. The first (see Basic Protocol 2) describes how to use Motifs, GCG’s Prosite (Bairoch, 1992; http://www.expasy.org/prosite) scanning tool, to discover cataloged motifs within the dataset and add that annotation to the dataset. The second (see Basic Protocol 3) shows the use of MEME (Bailey and Elkan, 1994; UNIT 2.4) within GCG for discovering and annotating common sequence elements within a dataset.

A final protocol (see Basic Protocol 4) explains the use and creation of multiple sequence alignment profiles, both traditional Gribskov-style profiles (Gribskov et al., 1987 and 1989) and statistically-based HMMER profiles (e.g., Eddy, 1996 and 1998).

NOTE: The author’s complete sample data RSF file with all annotation, protein sequences, and DNA sequences is available at the Current Protocols Web site (http://www3.interscience.wiley.com/c_p/cpbi_sampleddatafiles.htm), in case the reader wishes to work with the EF1a-primitive.rsf dataset.
MULTIPLE SEQUENCE ALIGNMENT USING PILEUP WITHIN SEQLAB

Given interest in a particular biological molecular sequence, several text string searching tools are available for finding the names of those entries in sequence databases, and for preparing and analyzing a multiple sequence alignment from them. This protocol instructs the user on how to launch and set up the Wisconsin Package SeqLab interface, access sequences in GCG’s databases based on taxonomy using LookUp, align those sequences with PileUp, visualize that alignment based on similarity, and in turn, how to refine the alignment in regions of lower similarity. Furthermore, regions common to the dataset will be annotated and various consensus and mask techniques will be explored.

Necessary Resources

Hardware
Terminal or personal workstation with access to a Unix server running commercial GCG software

Software
SeqLab (GCG Wisconsin Package; see Internet Resources)
X-server graphics communications software (APPENDIX 1D)

X-server emulation software needs to be installed separately on personal-style Microsoft Windows or Macintosh machines, but genuine X-Windowing comes standard with most Unix operating systems. Microsoft Windows machines are often set up with either XWin32 or eXceed to provide this function, while Macintoshes are often loaded with either MacX or eXodus software. The details of X are provided in APPENDIX 1D. If the user is unsure of these procedures assistance from local computer support personnel should be sought.

Files
Protein or DNA sequences of interest in GCG format (e.g., from LookUp in the GCG package and/or FASTA; see Support Protocols 1 and 2)

Log onto the GCG account and launch SeqLab

1. Install the GCG Wisconsin Package onto a Unix server according to the user manual (see Internet Resources).

   Normally the user does not need to perform this step as the software is server-based and thus installed by a system administrator.

2. Use the appropriate connection commands on the personal computer or terminal to launch X and log onto the Unix server with an existing GCG account and password. If an X-style terminal window does not appear on the desktop after a few moments, launch one with the appropriate command. If the Wisconsin Package does not initialize automatically as soon as the terminal window launches, issue the command gcg to initialize the software suite.

   This initialization process activates all of the programs within the package and displays the current version of both the software and all of its accompanying databases.

   The details of X (APPENDIX 1D) and of connecting to a GCG server are not covered in this unit. There are just too many variations in method for them all to be described. Obtain assistance from the institution’s computer support personnel if there is any uncertainty about how to perform this step.

3. Issue the command seqlab & in the terminal window to initialize the SeqLab interface.

   The ampersand (&) is not necessary but helps by launching SeqLab as a background process so that control of the initial terminal window can be maintained.
4. Check OK in the first of the two windows that appear, which should activate SeqLab’s List mode. Check the Mode: button and switch to Main List mode from Editor mode if necessary.

5. Before beginning any analysis, go to the Options menu and select Preferences.

   *The defaults are usually fine, but it is useful to see what options are available. Remember, buttons are turned on when they are pushed in and shaded.*

6. First notice that there are three different Preferences settings that can be changed: General, Output, and Fonts.

   a. **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 if desired; however, it is appropriate to leave it as is for now. Be sure that the Start SeqLab In: choice has Main List selected and that Close The Window is selected under the After I Push the Run Button: choice.

   b. **Output.** Be sure Automatically Display New Output is selected.

   c. **Fonts.** If dealing with very large alignments, then picking a smaller Editor Font Point Size may be desirable in order to see more of the alignment on the screen at once.

   Click OK to accept any changes.

**Load data and explore the GCG Editor Window**

7. Assemble the dataset using GCG’s LookUp and/or FASTA program (see Support Protocols 1 and 2). Select the LookUp output file in the SeqLab Output Manager. Press the Add to Main List button in the SeqLab Output Manager, and Close the window afterwards.

   *The SeqLab Output Manager is a very important window and will contain all of the output from the current SeqLab session. Files may be displayed, printed, saved in other locations and/or with other names, and deleted from this window.*

   Another way to get sequences into SeqLab is to use the Add Sequences from Sequence Files choice under the File menu. Only GCG-format-compatible sequences or list files are accessible through this route. Use SeqLab’s Editor File menu Import function to directly load GenBank format sequences or ABI binary trace files without the need to reformat. It is also possible to directly load sequences from the online GCG databases with the Databases choice under the Add Sequences menu if the proper identifier names or accession codes are known. The Filter box in the Add Sequences window is very important! By default, files are filtered such that only those that end with the extension .seq are displayed. Often the sequences to be added have other extensions. Therefore, delete the .seq extension in the Filter box (including the period) if necessary, but be sure to leave the asterisk (*) wild card. Press the Filter button to display all of the files in the working directory. Select the desired file from the Files box, and then check the Add, and then Close buttons at the bottom of the window to put the desired file into the current list if in List mode, or directly into the editor if in Editor mode. Note that SeqLab will accept any mix of valid GCG sequence specifications—i.e., individual GCG format sequences, database calls, multiple GCG sequence files (MSF or RSF), or list files in any combination.

8. Go to the File menu and press Save List. Next, be sure that the LookUp output file is selected in the SeqLab Main Window and then switch Mode: to Editor. This will load the file into the SeqLab Editor and allow further analyses on those entries to be performed.

   *Notice that all of the sequences now appear in the Editor window with the amino acid residues color-coded. The nine color groups are based on a UPGMA clustering of the*
BLOSUM62 amino acid scoring matrix (UNIT 3.5), and approximate physical property categories for the different amino acids.

9. Expand the window to an appropriate size by “grabbing” the bottom-left corner of its “frame” and “pulling” it out as far as desired. The loaded alignment is now available for analysis by any of the GCG programs. The display should look similar to Figure 3.6.1 after loading the dataset.

10. While the sequences are loaded in the Editor, explore the interface for a bit. Each protein sequence is listed by its official SwissProt or SPTREMBL entry name (ID identifier). Use both scroll bars to move around within the sequences. The scroll bar at the bottom allows movement through the sequences linearly, while the one at the side allows movement through all of the entries vertically. Quickly double click on various entry names (or single click the INFO icon with the sequence entry name selected) to see the database reference documentation on them in a Sequence Information window. (This is the same information that can be accessed with the GCG command typedata -ref at the command line.) It is also possible to change the sequence names and add any desired documentation in the Sequence Information window.

11. Change the Display: box from Residue Coloring to Feature Coloring and then Graphic Features. Now the display shows a schematic of the feature information for each entry with colors based on the information from the database Feature Table for the entry. Graphic Features represents features using the same colors but in a “cartoon” fashion. Quickly double-click on one of the various colored regions of the sequences (or use the Features choice under the Windows menu). This will produce a new window that describes the features located at the cursor. Select the feature to show more details and to select that feature in its entirety.

All the features are fully editable through the Edit check box in this panel and new features can be added with several desired shapes and colors through the Add check box.

12. Nearly all GCG programs are accessible through the Functions menu. Select the various entry names and then go to the Functions menu to perform different analyses. Sequences can be selected in their entirety by clicking on their names or at any position(s) within the sequences by “capturing” them with the mouse. Several methods are available for selecting multiple sequence entry names—i.e., either drag the mouse through them all (if they are all visible at once in the display), press the Shift button while clicking on the top- and bottom-most entries with the mouse (select nonadjacent entries using the Control rather than Shift button), or choose Select All from the Edit menu to select the entire dataset. The col: and pos: indicators at the bottom of the window show where the cursor is located on a sequence with and without including gaps, respectively. The 1:1 scroll bar near the upper right-hand corner allows the user to zoom in or out on the sequences—(e.g., move it to 2:1 and beyond and notice the difference in the display).

13. It is a good idea to save the sequences in the display at this point and multiple times down the road as work continues on a dataset. Do this periodically while using SeqLab just in case there’s an interruption of service for any reason. Go to the File menu and choose Save As. Accept the default .rsf extension, but give it an appropriate file name and directory specification.

The rich sequence format (RSF) contains all of the aligned sequence data and the name of each sequence, as well as all the reference and feature annotation associated with each entry. It is richer than most other multiple sequence formats and is SeqLab’s default format.
Performing the alignment: the PileUp program

14. To align all of these protein sequences, select all of the entries in the Editor window using one of the methods described above (step 12). Once all of the sequences are selected, go to the Functions menu and select Multiple Comparison. Click on PileUp to align the entries. A new window will appear with the parameters for running PileUp.

15. Often all of the program defaults will be accepted on a first run by pressing the Run button; however, in this example the scoring matrix for the alignment will be changed from the default BLOSUM62 to the alternate BLOSUM30 matrix (UNIT 3.5). Click on the Options button. To specify the BLOSUM30 matrix, select the check button next to the Scoring Matrix box and click on the Scoring Matrix box itself in the PileUp Options window. This will launch a Chooser for Scoring Matrix window from which the BLOSUM30 matrix file, blosum30.cmp, can be selected. Double-click the matrix’s name to see what it looks like, then click OK to close both windows. Scroll through the rest of the PileUp Options window to see all those available. Close it when finished. Notice the V command PileUp line shows the -Matrix option. Be sure that the How: box says Background Job and then press Run in the PileUp window to launch the program.

Depending on the level of divergence in a data set, better multiple sequence alignments can often be generated with alternate scoring matrices (UNIT 3.5), by using the -Matrix option and specifying the desired matrix from the GCG logical directory GenMoreData, and/or assigning different gap penalties. Beginning with GCG version 9.0, the BLOSUM62 (Henikoff and Henikoff, 1992) matrix file, blosum62.cmp, is used as the default symbol comparison table in most programs. Furthermore, appropriate gap creation and extension penalties are now coded directly into the matrix, though they can still be adjusted within the program if desired. This is a greatly improved situation over the normalized Dayhoff PAM 250 table (Schwartz and Dayhoff, 1979) and the program encoded penalty values that GCG formerly used. The BLOSUM series are much more robust at handling a wider range of sequence divergence than the PAM table ever was, with the BLOSUM30 table being most appropriate for the most divergent datasets and the BLOSUM100 table for the most conserved datasets. Since these sequences are from quite a wide spectrum of organisms, it is more appropriate to use the BLOSUM30 matrix in this case.

The program will first compare every sequence with every other one. This is the pairwise nature of the program, and then it will progressively merge them into an alignment in the order of determined similarity, from most to least similar (Feng and Doolittle, 1987).

16. The window will go away and 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 the PileUp run. Notice the BLOSUM30 matrix specification and the default gap introduction and extension penalties associated with that matrix: 15 and 5, respectively. Scroll down to examine the alignment and then close the window.

As mentioned above, in most cases the default gap penalties will work well with their respective matrixes, though they can be changed if desired. In fact, see below on improving regions within alignments, where it is absolutely required.

The author’s abridged output file example is shown in Figure 3.6.2. Notice the interleaved character of the sequences, yet they all have unique identities, addressable through their MSF filename together with their own name in braces—i.e., {name}.

17. Return to the list of sequence names near the top of the file. This list contains an important number called the checksum. All GCG sequence programs use this number as a unique sequence identifier. There is a checksum line for the whole alignment (Check: 2476 in Fig. 3.6.2) as well as individual checksum lines for each member
of the alignment (e.g., Check: 8631). 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. Alterna-
tively, the sequence can be cut from the alignment with the SeqLab Editor.

The checksum is calculated based on the sequence content and has no biological meaning.
It merely safeguards against file corruption.

18. Another important number on the individual checksum lines also needs to be pointed
out. The weight designation determines how much each sequence contributes to a
profile made of the alignment. Sometimes it is worthwhile to adjust these values so
that the contribution of a collection of very similar sequences does not overwhelm
the signal from a few more divergent sequences. In the SeqLab interface the Sequence
Info... window can be used to accomplish this, or a simple text editor (APPENDIX 1C)
can be used.

19. To merge the LookUp output with the aligned sequence file, an extremely important
Output Manager function must be used. Select the PileUp output file in the Output
Manager window and then press the Add to Editor button. When prompted, Specify
Overwrite Old with New in the Reloading Same Sequence window to take the PileUp
output and merge it with sequences already in the open RSF file in the editor. This
will keep all of the database feature annotation intact, yet renumber all of its reference
locations based on the inclusion of gaps in the alignment. Close the Output Manager
after loading the new alignment.

View and print the dendrogram that guided the progressive alignment

20. The next window will contain PileUp’s cluster dendrogram, in the EF-1α example,
the graphic shown in Figure 3.6.3. If this window was accidentally closed earlier, use
the Output Manager to redisplay the file that ends with the extension .figure.

PileUp automatically creates this dendrogram of the similarity clustering relationships
between the sequences. It can be very helpful for adjusting sequence weight values, which
even out the contributions of each sequence to a profile. The lengths of the vertical lines
are proportional to the differences in similarity between the sequences; however, realize
that this is not an evolutionary tree and should never be presented as one. No phylogenetic
inference optimality criteria algorithm (e.g., maximum likelihood, least-squares fit, or
parsimony), molecular-substitution multiple-hit correction models—e.g., Jukes-Cantor
(UNIT 6.3), Kimura (UNIT 6.3), or any other subset of the General Time Reversible (GTR)
model—or any site-rate heterogeneity models (e.g., Gamma correction) are used in its
construction. PileUp’s dendrogram merely indicates the relative similarity of the sequences
based on the scoring matrix used, by default BLOSUM62, but BLOSUM30 in the author’s
example, and therefore, the clustering order used to create the alignment. It is roughly an
uncorrected UPGMA tree, prone to all the same errors seen in UPGMA; therefore, if the
rates of evolution for each lineage were exactly the same, then it could represent a “true”
phylogenetic tree, but this is seldom the case in nature.

21. If desired, it is possible to print directly from SeqLab graphics Figure windows to
PostScript files by selecting Print... [Encapsulated] PostScript File Output Device:.
Name the output file appropriately and click Proceed to create an EPSF output in the
current directory.

To actually print this file it may be necessary to transfer it to a local machine attached to
a PostScript savvy printer via FTP or SCP unless the user has 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.
22. Close the dendrogram window.

Options for viewing the alignment

23. Notice that the residues align by color. The author’s editor display is shown in Figure 3.6.4 after loading the MSF file using Residue Coloring and a 1:1 zoom ratio. Notice the nice columns of color representing columns of aligned residues.

24. Change the Display: box from Residue Coloring to Graphic Features. Now the display shows a schematic of the feature information from each entry, as well as all of the motifs discovered by the programs Motifs (see Basic Protocol 2) and Motif-Search (see Basic Protocol 3), and will look something like Figure 3.6.5, at a 4:1 zoom ratio.

25. Remember, double clicking quickly on any of the color coded feature regions in the Editor display will open a Features window where more information about that particular feature can be accessed. Clicking once in the colored region and then using the Features option from the Windows menu will also produce the Features window.

26. At this point, it would be wise to save the data as an updated RSF file. Select overwrite in the File Exists box if the same name has been used for this file earlier. This is the route suggested by the author, as RSF files are quite large and there is no need to save all the various versions of the data.

27. At this point, the user may further analyze the alignment (see Basic Protocol 4 or Support Protocols 3 and 4) or convert it to a different format for phylogenetic analysis (see Support Protocols 5 and 6).

USING LOOKUP TO ASSEMBLE A DATASET

The collection of sequences used throughout this unit contains representative EF-1α sequences from many lower eukaryotes, chosen based on taxonomy. To access entries of interest in GCG sequence databases, their proper database names or accession codes need to be known. Database text searching programs are often the easiest way to do this. In this protocol, the dataset will be assembled using GCG’s LookUp program; however, it could be collected using Entrez at NCBI (UNIT 1.3; also see Internet resources and http://www.ncbi.nlm.nih.gov/Entrez) as well, either through the Web or installed as their client/server NetEntrez application. The advantage of LookUp is that it creates an output file that can be used as an input list file to other GCG programs. The author uses it here to find a representative set of elongation factor entries from the so-called primitive eukaryotes (i.e., those eukaryotes that exclude the Fungi, Metazoans, and true Plants).

Necessary Resources

Hardware

Terminal or personal workstation with access to a Unix server running commercial GCG software

Software

LookUp (GCG Wisconsin Package; see Internet Resources)
X-server graphics communications software (APPENDIX 1D)

X-server emulation software needs to be installed separately on personal-style Microsoft Windows or Macintosh machines, but genuine X-Windowing comes standard with most Unix operating systems. Microsoft Windows machines are often set up with either XWin32 or eXceed to provide this function, while Macintoshes are often loaded with either MacX.
Files
None

1. Initialize the GCG environment and launch SeqLab. Be sure that the Mode: Main List choice is selected in the main window and then launch LookUp through the Functions Database Reference Searching menu. In the new LookUp window (Fig. 3.6.6), be sure that Search The Chosen Sequence Libraries is checked and then select SwissProt as well as SPTRREMBL for the libraries to search. Under the main query section of the window, type elongation & factor & alpha following the category Definition and eukaryota ! (fungi | metazoa | viridiplantae) in the Organism category. Next press the Run button.

The Boolean operator symbols are needed to connect the individual query strings because the databases are indexed using individual words for most fields. The Organism field is an exception; it will accept Genus species designations as well as any other single word supported level of taxonomy (e.g., fungi.) The Boolean operators supported by LookUp are the ampersand (&) meaning AND, the pipe symbol (|) to denote the logical OR, and the exclamation point (!) to specify NOT. Other LookUp query construction rules are case insensitivity, parenthesis nesting, wildcard (* and ?) support, and automatic wildcard extension.

This query should find most of the elongation factor alpha’s from the lower eukaryotes in the SwissProt and SPTRREMBL databases and will provide a reasonable and interesting starting dataset. The LookUp window should look similar to Figure 3.6.6.

2. The program will display the results of the search; scroll through the output and then Close the window. The beginning of the LookUp output file from the example is shown in Figure 3.6.7. Examine the output to ensure that all of the proteins included from any text-searching program are appropriate.

In this case, the elongation factors found all look correct, but improper nomenclature and other database inconsistencies can always cause problems. If inappropriate proteins are found upon reading the output, either edit the output file (APPENDIX 1C) to remove them, or CUT them from the SeqLab Editor display after loading the list (see Basic Protocol 1, step 17). Another option, if using an editor, is to comment out the undesired sequences by placing an exclamation point (!) in front of the unwanted lines.

3. If the query returns an insufficient (or overwhelming) number of sequences, run a similarity search to augment (or guide judicious editing of) the dataset (see Support Protocol 2).

4. If the LookUp results, which are automatically saved by SeqLab (.lookup extension) do not have the desired filename, use the Output Manager to save it with a more appropriate name.

SIMILARITY SEARCHING TO INCREASE (OR DECREASE) DATASET SIZE
After an entry has been identified, a logical next step in preparing a multiple sequence alignment might be to use a sequence similarity searching program such as FASTA (Pearson and Lipman, 1988; Pearson, 1998) and/or BLAST (Altschul et. al, 1990, 1997; UNITS 3.3 & 3.4) to help prepare a list of sequences to be aligned. Using these programs within the context of GCG is incredibly helpful as the output from all of the GCG search programs is a valid GCG list file, appropriate as input to many of the other GCG programs, PileUp in particular. The results are immediately available for further analyses without the need for any sequence downloading or reformattting because of the GCG list file format.
and the fact that all of the databases are mounted locally. Therefore, the sorted, most similar sequences can be loaded directly from the local GCG databases to the dataset without any intermediate steps and the dataset can be built as large as desired.

But what about the opposite situation, when there are too many homologs? Database similarity searching can also be very helpful for sorting collections of GCG sequence specifications into the order of alignment significance. This data mining function allows the user to easily screen undesired sequences from the bottom of any list or combination of lists. However, be warned, none of the GCG-BLAST programs can be used to search against a sequence set that has not been preformatted into a BLAST compatible database. Because of this, BLAST is not an appropriate program to use for this type of list-file-sorting, data-mining function. However, the GCG implementation of the FASTA family of programs supports all GCG sequence specifications, so it works great for this purpose.

In this protocol, GCG’s version of FASTA is used to sort an existing LookUp output list file (see Support Protocol 1) along with NRL_3D to narrow down the data to a more manageable size. The Giardia sequence will be used as a query for this search because many researchers consider Giardia’s most ancient ancestor to be rooted near the base of the universal tree of life on the eukaryote lineage (e.g., see Sogin et al., 1996). Since the target dataset is all lower eukaryotes, this seems like an appropriate choice.

**NOTE:** FASTA in GCG version 10.2 cannot be run on too small of a dataset without causing core dumps on some systems! A trick is to add another small database such as NRL_3D, or the results of any other database search to the Search List Set. This provides the necessary background randomization to allow proper normalization.

**Necessary Resources**

**Hardware**

Terminal or personal workstation with access to a Unix server running commercial GCG software

**Software**

SeqLab (GCG Wisconsin Package; see Internet Resources)

X-server graphics communications software (**APPENDIX 1D**)

* X-server emulation software needs to be installed separately on personal-style Microsoft Windows or Macintosh machines, but genuine X-Windowing comes standard with most Unix operating systems. Microsoft Windows machines are often set up with either XWin32 or eXceed to provide this function, while Macintoshes are often loaded with either MacX or eXodus software. The details of X are given in **APPENDIX 1D**. If the user is unsure of these procedures ask for assistance from local computer support personnel.

**Files**

Protein or DNA sequences of interest in GCG format (e.g., from LookUp in the GCG package; see Support Protocol 1)

1. Begin by selecting all of the sequences from the previous LookUp output list file in the open SeqLab Editor, as described (see Basic Protocol 1, steps 7 to 8). Deselect the EF1A_GIALA sequence entry name in the Editor display by pressing the Control button and clicking its entry name. Cut all of the other sequences from the display. Go back and select the remaining EF1A_GIALA sequence and launch FASTA off the Functions Database Sequence Searching menu. If a Which Selection window pops up asking if the user wants to use the Selected Sequences or Selected Region,
choose Selected Sequences to run the program on the full length of the selected protein.

2. At most sites, the default protein database to search, Search Set, will be Using pir:*; hence, in this example, it will be set to the LookUp output list file and NRL_3D. Therefore, push the Search Set button, select pir:* in the Build FastA's Search Set box that pops up, and then Remove from Search Set. Next, press the Add Main List Selection button, select the previous LookUp output list file from the List Chooser window that pops up, and press Add To Search Set. Repeat this process using the Add Database Sequences button in the Build FastA's Search Set window, and specify NRL_3D:* as described above.

Alternatively, the results of other database searches can be added using Add Sequence Files and the Filter function correctly to identify and load any other relevant output files. If done with BLAST or FASTA, they will have names that end in some variation of .blast or .fasta, respectively.

3. Close the List Chooser and the Build Search Set windows. Decrease the Cutoff Expectation value in the main FASTA window List scores until it reaches something quite stringent (e.g., 0.001) to reduce the output list size. Be sure that the FASTA program window shows How: Background Job, and then press the Run button. The output will quickly return since it is a very small search set. An abridged example of that output is shown in Figure 3.6.8.

Load results into SeqLab

4. Whatever method is used to get the dataset to the appropriate size, it is still necessary to load it into the SeqLab Editor using the Output Manager window, which is always available through the SeqLab Windows menu. Use the Output Manager’s Add to Editor function and then Overwrite Old with New, to take the new FASTA output and merge it with the sequence in the open Editor window. Click Interrupt Loading in the Loading Sequences window after as many sequences have loaded as can be comfortably managed. FASTA files are loaded in order of similarity to the query. In the case of this example, the author restricted analysis to about the top 50 entries of the final FASTA file.

5. The next prompt requires some thought if loading the results of a similarity search. The user will be asked whether to Modify the Sequences or Ignore All Attributes in a List File Attributes Set window. The answer will depend on the type of alignment being created and the biological questions being asked. In many cases, especially if phylogenetic questions are being asked, the user will not want to modify the sequences. Load their full length to maximize available signal. However, if dealing with extremely diverse sequences and/or just domains of sequences, then trimming the sequences down to the most conserved portions identified by FASTA can be very helpful.

In this case, sequences will not be trimmed down, so the Ignore All Attributes button is pressed.

6. Close the Output Manager after loading the new FASTA list file and return the display to 1:1 and Residue Coloring. Take a look at the new members in the display. As before, quickly double click on the names of various entries to see the database reference descriptions for them (or click on the INFO button). Figure 3.6.9 shows the Editor display after loading the top part of the author’s FASTA file.

Now would also be a good time to go back to the File menu and Save the current RSF file.

The sequence set is now ready to align (see Basic Protocol 1).
USING PLOTSIMILARITY AND SEQLAB TO IMPROVE AND EDIT THE MULTIPLE SEQUENCE ALIGNMENT

The most conserved portions of an alignment are those most resistant to evolutionary change, often due to some type of structural constraint. To easily visualize the positional conservation of a multiple sequence alignment use the graphics program PlotSimilarity. The program draws a graph of the running average similarity along a group of aligned sequences (or of a profile with the -Profile option). The PlotSimilarity peaks of a protein alignment represent the most conserved areas of the alignment, but even more so, those areas most resistant to evolutionary change due to the algorithm’s use of the BLOSUM matrix (UNIT 3.5) in its calculations. PlotSimilarity is also a nice way to see those areas of an alignment that may need improving by pointing out the most variable regions. Furthermore, PlotSimilarity can be helpful for ascertaining alignment quality by noting changes in the overall average alignment similarity and changes in those regions of conservation within the alignment, as it is adjusted and refined.

**Necessary Resources**

**Hardware**
- Terminal or personal workstation with access to a Unix server running commercial GCG Software

**Software**
- PlotSimilarity and SeqLab (GCG Wisconsin Package; see Internet Resources)
- X-server graphics communications software (APPENDIX ID)

* X-server emulation software needs to be installed separately on personal-style Microsoft Windows or Macintosh machines, but genuine X-Windowing comes standard with most Unix operating systems. Microsoft Windows machines are often set up with either XWin32 or eXceed to provide this function, while Macintoshes are often loaded with either MacX or eXodus software. The details of X are provided in APPENDIX ID. If the user is unsure of these procedures ask for assistance from local computer support personnel.

**Files**
- Multiple sequence alignment in GCG format (see Support Protocol 1)

**Run PlotSimilarity to identify conserved regions**

1. Select all of the sequence names and then go to the Functions menu and under the Multiple comparison section choose PlotSimilarity. It is advisable to change some of the program defaults, so choose Options in the program window. Check Save SeqLab Colormask To and Scale the plot between the minimum and maximum values calculated from the alignment.

   The first option’s output file will be used in step 4. The second specification launches the program’s command line -Expand option. This blows up the plot, scaling it between the maximum and minimum similarity values observed, so that the entire graph is used, rather than just the portion of the Y-axis that the alignment happens to occupy. The Y-axis of the resulting plot uses the similarity values from whichever scoring matrix is used to create the alignment unless an alternative is specified. The default matrix, BLOSUM62, begins its identity value at 4 and ranges up to 11; mismatches go as low as −4.

2. Close the Options window. Notice that the Command Line: box now reflects the 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. The author’s example is shown in Figure 3.6.10.
This example shows a great deal of sequence similarity. Strong peaks are seen centered around positions 30, 100, and 375. The ordinate scale is dependent on the scoring matrix used by the program, here the BLOSUM30 table, which ranges in score from $-7$ to $+20$. The dashed line across the middle shows the average similarity value for the entire alignment, here $\sim 4.4$.

3. Make a PostScript file of this plot, if desired. As before, to print a SeqLab graphics Figure to a PostScript file: select Print off the Figure window, choose Output Device: [Encapsulated] PostScript File, and click Proceed to create EPSF output. Regardless of whether this plot is printed, take notes of where the similarity significantly falls off at the beginning and end as well as within the alignment. In the example above, this is the first 25 residues or so, a region around 190 and 220, around 390, and about the last 25 residues. Close the PlotSimilarity window after noting where these deepest valleys, the least similar regions of the alignment, lie.

4. Go to the File menu and click on Open Color Mask Files. This will produce another window from which the new plotsimilarity.cmask file should be selected; click on Add and Close the window. This will produce a gray scale overlay on the sequences which describes their regional similarity where darker gray corresponds to higher similarity values. The sample alignment, using a zoom factor of 4 to 1, looks like Figure 3.6.11. Notice the strong conservation peak centered just before residue 100 in the alignment, one of EF-1α’s GTP binding regions.

**Improve alignments within SeqLab**

5. The beauty of the masked representation created in step 4 is that those regions of low similarity can be easily selected to try to improve their alignment automatically. This is possible because of PileUp’s incredibly effective -InSitu option that can realign regions within an alignment. Be sure that all of the sequences are selected and then zoom back in the alignment to 1:1 so that individual residues can be seen, and then scroll to the carboxy end.

   It is best to start at the carboxy termini in this process so that the upstream positions of the low-similarity regions do not become skewed as the procedure is followed.

6. Now select a region of low similarity across the complete sequence set. This can be done using the mouse if it is all on the screen, which is not the case here. Otherwise, use the Edit Select Range function (determine the positions by placing the cursor at the beginning and end of the range to be selected and noting the column number in the lower left-hand of the Editor display). Once all of the sequences and the region to be improved 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 have been used in the current session; any of them can be launched from there as well.

7. The user will be asked whether to use the selected sequences or 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 specify the BLOSUM30 matrix (also see Basic Protocol 1), and check the Gap Creation and Extension boxes and change their respective values to much less than the default. Changing them to about a third the default value works well for a start. Thus for the BLOSUM30 matrix, change the values to 5 and 2 respectively. Most importantly, check Realign a Portion of an Existing Alignment, which calls up the command-line -InSitu option. Otherwise, only that portion of the alignment selected will be retained in the output. Furthermore, as there really is no need for another similarity dendrogram,
uncheck the Plot Dendrogram box. Close the window and notice the new options in the PileUp Command Line window. Run the program to improve the alignment.

The window will go away and the results will return very quickly since only a portion of the alignment is being realigned; new output windows will automatically display. The top window will be the MSF output from the PileUp run. Notice the BLOSUM30 matrix specified (others available through the options menu) and the lowered gap introduction and extension penalties of 5 and 2.

8. Scroll through the new alignment to examine it and then Close the window. The next window will be the Output Manager. Just like before, click on Add to Editor and then specify Overwrite Old with New in the new Reloading Same Sequences window to merge the new alignment with the old one and retain all feature annotation. This feature information may help guide the alignment efforts in subsequent steps. Close the Output Manager window after loading the new alignment.

9. The 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 terminus toward the amino end. Notice that all of the options that were last specified are retained by the program so there is no need to respecify them. These run parameters can also be saved so that they will come up in subsequent sessions, by clicking on the Save Settings box in any of the program run windows. A good suggestion is to go to the File menu periodically to save the work in progress using the Save As function in case of a computer or network problem.

10. It is also probably a good idea to perform the PlotSimilarity and color mask procedure again after going through the entire alignment to see how things have improved after the various InSitu PileUps are finished. If an area is discovered that cannot be improved through this automated procedure, then it is time to either manually correct it (step 11) or throw it away (or mask it as will be shown later; see Support Protocol 4). Again, note those problem areas and then switch back to Residue Coloring. This will ease manual alignment by allowing the eyes to work with columns of color.

Manually align and edit problem areas in the alignment

11. GROUPing and Protections can help manual alignment. The GROUP function allows manipulations of families of sequences as a whole, as any change in one will be propagated throughout them all. To GROUP sequences, select those that are to be treated collectively and then click on the GROUP icon right above the alignment. There can be as many groups as desired. The Space Bar will introduce a gap into the sequence and the Delete key will take a gap away. However, a sequence residue cannot be deleted 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 the modification of Gap Characters and Reversals only. Check All other characters to allow regions to be cut out of the 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 to slide and then press the Shift key while using the right or left arrow keys to move the region. Residues can slide greater distances by prefacing the command keystrokes with the desired number of spaces.

12. Make subjective decisions regarding the alignment. Is it good enough? Do things line up the way that they should? If, after all else, it is decided that some region or even an entire sequence just cannot be aligned, then get rid of it with the Cut function. Another alternative is the Mask function (see Support Protocol 4). Cutting out an entire sequence may leave some columns of gaps in the alignment. If this is the case,
then reselect all of the sequences, go to the Edit menu, and select Remove Gaps Columns of Gaps. Notice the extreme amino and carboxy ends of the alignment. Amino and carboxy termini seldom align properly and are often jagged and uncertain. This is fairly common in multiple sequence alignments and subsequent analyses should probably not include these regions. If loading sequences from a database similarity search, allowing SeqLab to trim the ends automatically based on beginning and ending constraints considerably improves this situation. Overall, things to look for include columns of strongly conserved residues (e.g., tryptophans, cysteines, and histidines), important structural amino acids (e.g., prolines, tyrosines, and phenylalanines), and conserved substitutions (e.g., isoleucine, leucine, and valine); make sure they all align.

13. After having finished tweaking, evaluating, and readjusting the 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 the alignment is as biologically correct as possible. Everything done from this point on, especially if later the alignments are used to ascertain molecular phylogenies, is absolutely dependent on the quality of the alignment! It needs to be a very clean, unambiguous alignment that inspires high confidence—i.e., a truly biologically meaningful alignment. Each column of symbols must actually contain homologous characters. Many other alignment editors are available for cleaning up multiple sequence alignments; however, in the author’s opinion, SeqLab is the most satisfying, and only using a GCG compatible editor assures that the format will not be corrupted. If any changes are made to a GCG sequence data file with a non-GCG compatible editor, the alignment must be reformatted afterwards. However, reformattting GCG, MSF, or RSF files requires a couple of tricks. If this step is not done correctly, very weird results will occur. If this must be done for any reason, the appropriate Reformat option (either -MSF or -RSF respectively) must be used and all the sequences within the file must be specified using the brace specifier—i.e., {^}. For example, > reformat -rsf your_favorite.rsf{*}.

There should rarely be a need to do this, unless for some unusual reason there is a decision to edit an alignment with a non-GCG compliant editor; however, it may prove necessary in some situations. After reformattting, the new MSF or RSF file will follow GCG conventions with updated format, numbering, and checksums.

**SeqLab editor on-screen annotation**

14. Something that may be desirable after the alignment is all cleaned up, is to add text, color, and shape annotation to the display. Changing the names of the entries might also be helpful for presentation purposes. All are easy to do in the SeqLab Editor. Double-click on an entry’s name to get its Sequence Information window and directly edit the name and annotation there. Selecting the entry name and then pressing the INFO icon does the same thing.

15. To put text lines directly into the display, go to the SeqLab File menu New Sequence entry and select the Text button to the What Type of Sequence? question. This will put a NewText line at the bottom of the Editor display in which the annotation can be directly typed. It is also possible to add customized Graphic Features and Features Coloring annotation with the Windows Features window. Select a desired region across an alignment and launch the Features window. Press Add to get a Feature Editor window where the feature’s Shape:, Color:, and Fill: can be designated and the region’s Keyword: and Comments: can be annotated. IMPORTANT NOTE: A feature annotation can be added to a region across an entire alignment, but the annotation cannot be deleted or edited from the whole region collectively.
afterwards. It is possible only to edit or delete a feature annotation from an RSF file with the SeqLab Editor one sequence feature at a time!

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

**CONSENSUS AND MASKING ISSUE: GCG’S MASK OPERATION**

Consensus methods are another powerful way to visualize similarity within an alignment. A SeqLab mask allows the user to differentially weight different parts of their alignment to reflect their confidence in it. It can be a handy trick with some data sets, especially those with both highly conserved and highly variable regions. Masks can also be modified by hand or they can be created manually through the New Sequences menu. They can have position values all the way up to 9, though it is doubtful anyone would want a column of an alignment to be nine times as important as some other column. Masking is especially helpful for phylogenetic analysis by excluding those less reliable columns in the alignment where the user is not confident in the positional homology without actually getting rid of the data.

**Necessary Resources**

**Hardware**

Terminal or personal workstation with access to a Unix server running commercial GCG software

**Software**

SeqLab (GCG Wisconsin Package; see Internet Resources)

X-server graphics communications software (*APPENDIX 1D*)

*X*-server emulation software needs to be installed separately on personal-style Microsoft Windows or Macintosh machines, but genuine *X*-Windowing comes standard with most Unix operating systems. Microsoft Windows machines are often set up with either XWin32 or eXceed to provide this function, while Macintoshes are often loaded with either MacX or eXodus software. The details of *X* are discussed in *APPENDIX 1D*. If the user is unsure of these procedures, ask for assistance from local computer support personnel.

**Files**

Multiple sequence alignment in GCG format (see Support Protocol 1)

1. The SeqLab Edit menu allows the user to easily create several types of consensus representations. To create a standard protein sequence consensus, select all the sequences, use the Edit Consensus menu and specify Consensus type: Protein Sequence.

   *When making a normal sequence consensus of a protein alignment, figures can be generated with highly similar residues in black, intermediate similarities in gray, and nonsimilar amino acids in white. This is a nice way to prepare alignment figures for publication. The default mode is to create an identity consensus at the 2/3 plurality level (percent required for majority) with a threshold of 5 (minimum score that represents a match).*

2. Try different lower plurality and threshold values as well as different scoring comparison matrices to see the difference that it can make in the appearance of the 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.
Figure 3.6.12 illustrates a region near the carboxy termini in the author’s example using the BLOSUM30 matrix, a Percent required for majority (plurality) of 33%, and a Minimum score that represents a match (threshold) cutoff value of 4.

3. When an acceptable plurality combination has been found, go to the File Print command; change the Output Format: to PostScript in order to prepare a PostScript file of the SeqLab display. Whatever color scheme is being displayed by the Editor at the time will be captured by the PostScript file. Play around with the other parameters. Notice that as the font size is changed, the number of pages to be printed varies. In the Print Alignment menu, specify Destination File, give it an appropriate filename, and then click OK.

This command will produce a PostScript language graphics file in the directory from which SeqLab was launched and is a great way to prepare research presentations. The PostScript file can be sent to a color PostScript printer, a black and white laser printer that will simulate the colors with gray tones, or it can be imported into a PostScript savvy graphics program for further manipulation. Unfortunately, if it is longer than one page, the raw PostScript format is so different from standard Encapsulated PostScript that it may be necessary to use a different Unix print queue. Discuss these matters with the system administrator. It may require some variation of the following type of command: > lpr -PPostScript_queue seqlab_alignment.ps.

Create and apply a mask
4. In addition to standard consensus sequences using various similarity schemes, SeqLab also allows the creation of consensus masks that screen specified areas of the alignment from further analyses by specifying 0 or 1 weights for each column.

5. To create a mask-style sequence consensus select all the sequences, then use the Edit Consensus menu, and specify Consensus Type: Mask Sequence. As above, the default mode uses an identity consensus at the 2/3 plurality level with a threshold of 5. However, these are very high values for phylogenetic analysis and would likely not leave much phylogenetically informative data. Therefore, again experiment with different lower pluralities, threshold values, and scoring comparison matrices. Be sure that Shade Based on Similarity to Consensus is still checked.

Figure 3.6.13 illustrates the carboxy-terminal end of the author’s example using a weight mask generated from the BLOSUM30 matrix, a plurality of 15%, and a threshold of 4. Few areas are excluded by the mask in this alignment because of the high similarity of this group of sequences. This is as it should be, for excluding many more columns in this particular alignment would likely leave nearly identical sequences and it would be impossible to ascertain how they are related.

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

CONVERT A MULTIPLE SEQUENCE ALIGNMENT TO PAUP* FORMAT FOR PHYLOGENETIC ANALYSIS

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

However, one of the biggest problems in bioinformatics is that of molecular sequence
data format. There are so many of them! The PAUP* interfaces in the GCG Wisconsin
Package, PAUPSearch and PAUPDisplay, automatically generate their required NEXUS
format directly from the GCG formatted files. Most other systems are not nearly so
helpful. Several different programs are available to convert formats back and forth
between the required standards (APPENDIX 1E), but it can all get quite confusing. One public
domain program available, ReadSeq by Don Gilbert at Indiana University (see Internet
Resources and http://iubio.bio.indiana.edu/soft/molbio/readseq; APPENDIX 1E), allows for
the back and forth conversion between several different formats. The author heartily
recommends installing it on all computers used for these purposes. It comes as a
“tried-and-true” C version or a new JAVA version with a graphical interface. This support
protocol will outline a method proven reliable for dataset format conversion to NEXUS.

**Necessary Resources**

**Hardware**

Terminal or personal workstation with access to a Unix server running commercial
GCG software

**Software**

SeqLab (GCG Wisconsin Package; see Internet Resources)

X-server graphics communications software (APPENDIX 1D)

X-server emulation software needs to be installed separately on personal-style Microsoft
Windows or Macintosh machines, but genuine X-Windowing comes standard with most
Unix operating systems. Microsoft Windows machines are often set up with either XWin32
or eXceed to provide this function, while Macintoshes are often loaded with either MacX
or eXodus software. The details of X are provided in APPENDIX 1D. If the user is unsure of
these procedures, ask for assistance from local computer support personnel.

**Files**

Multiple sequence alignment loaded into SeqLab (see Basic Protocol 1)

**GCG’s interface to PAUP* and NEXUS format**

1. GCG implements David Swoford’s PAUP* (usually pronounced “pop star”; UNIT 6.4)
phylogenetic analysis package with the paired programs PAUPSearch and PAUPDis-
play. These interface programs provide an easy to use access to much of PAUP* (UNIT
6.4) within GCG. Within the context of GCG, NEXUS format files are most easily
and reliably built from alignments with GCG’s own interface to the PAUP* package.
PAUPSearch within SeqLab can be used to generate NEXUS format files which can
then be fed directly to any version of PAUP* (UNIT 6.4).

The use of these programs for evolutionary inference will not be covered here. For serious
phylogenetic analysis the user may want to consider running PAUP* exterior to GCG by
getting the latest version directly from Sinauer Associates, the publishing company that
distributes the software (see Internet Resources; http://www.sinauer.com), and installing
it on a local machine. PAUP*, included in GCG version 10.3 and earlier, runs either in
native mode or through the PAUPSearch and PAUPDisplay programs, and is an old
4.0.0d55 version. If the user does not have access to the latest version of PAUP*, which
contains many bugs fixes and enhancements since 4.0.0d55, then using it within GCG is a
legal alternative. Use the following command in a terminal window to read the license
agreement with GCG if interested: &gt; typedata paup-license.txt.
2. Begin the NEXUS conversion process by selecting all relevant sequences and any desired weight mask, in the Main Editor Window display. Select PAUPSearch from the Functions Evolution menu to launch the dialogue box. To only generate a NEXUS file, this example will run PAUPSearch in its fastest mode without actually performing a search. Accept the default Tree Optimality Criterion Maximum Parsimony and the Heuristic Tree Search (fast) Method for Obtaining Best Tree(s). Be sure that the Perform Bootstrap Replications button is not pressed and then launch the Options menu by pressing the appropriate button. In the PAUPSearch Options menu check the top box to save the PAUPscript file. This is not required for running the programs, but since in this example only a NEXUS format is being generated, it is essential. The file name can be changed or left as desired. The PAUPscript output file results from the automatic conversion of the alignment to NEXUS format and contains all the PAUP* commands as well as the alignment.

If needed, the PAUPlog file keeps track of all that has happened during the program run and is a good place to look for any error messages. Saving this avoids otherwise frustrating troubleshooting.

3. Uncheck the next box, Perform the Analysis. This makes the program generate the NEXUS script, but prevents it from performing the heuristic search for the best tree (equivalent to the GCG PAUPSearch command-line option –NoRun). Scroll down through the options menu, leaving the rest of the options at their default settings, but check them out. Close the Options menu.

4. Normally PAUPSearch and PAUPDisplay are linked to each other when run from the SeqLab interface. Therefore, uncheck the PAUPDisplay button in PAUPSearch’s main window to turn PAUPDisplay off. Be sure that How: Background Job is specified on the main PAUPSearch menu and then press Run in that menu. After a moment the output PAUPscript file will be displayed. Figure 3.6.14 shows the author’s abridged PAUPscript output example, the elongation factor protein dataset after being converted to NEXUS format with masked columns removed.

The PAUPscript file contains the NEXUS format file that was generated by GCG to run PAUP*. Notice that columns of the alignment with zeroes in their mask are excluded from the NEXUS alignment. If using this file as input to native PAUP*, comment out—i.e., enclose in square brackets ([ ])—or remove any inappropriate commands within the command block near the end of the file with a simple text editor. Likewise, this file can be greatly expanded by encoding any desired commands and rate matrices within its command block.

To run this file in the latest version of PAUP*, FTP or SCP it to an appropriate machine. Using a Macintosh may be desirable in order to take advantage of PAUP*’s very friendly Macintosh GUI. Since GCG automatically creates this file, correctly encoding all of the required format data when running PAUPSearch, there is no need to hassle with a later conversion of the alignment to NEXUS. File format conversion can be a huge headache and here GCG has done all of the work.

The author recommends running the latest version of PAUP* available, but whatever version is used, learn how to run the most robust searches possible before accepting any output as valid phylogenetic inference. The techniques of molecular phylogenetics are beyond the scope of this unit, but are covered elsewhere in this book (See Chapter 6).
distance matrix, likelihood, bootstrapping, and consensus techniques, as well as tree
drawing and manipulation programs. Data is read into the program from a text file in
PHYLIP format. Output is written into a text outfile and a treefile in Newick format (UNIT
6.2), an informal standard agreed to in 1986 by the authors of a number of major phylogeny
packages. To reliably generate PHYLIP format from GCG alignments in SeqLab a
combination approach of GCG’s ToFastA and Don Gilbert’s ReadSeq (see Internet
Resources) will be used.

**Necessary Resources**

**Hardware**

Terminal or personal workstation with access to a Unix server running commercial
GCG software

**Software**

SeqLab (GCG Wisconsin Package; see Internet Resources)
ReadSeq (D.G. Gilbert; see Internet Resources; APPENDIX 1E)

X-server graphics communications software (APPENDIX 1D)

X-server emulation software needs to be installed separately on personal-style Microsoft
Windows or Macintosh machines, but genuine X-Windowing comes standard with most
Unix operating systems. Microsoft Windows machines are often set up with either XWin32
or eXceed to provide this function, while Macintoshes are often loaded with either MacX
or eXodus software. The details of X are provided in APPENDIX 1D. If the user is unsure of
these procedures, ask for assistance from local computer support personnel.

**Files**

Multiple sequence alignment loaded into SeqLab (see Basic Protocol 1).

1. Go to the SeqLab Main Window File Export menu; click Format and notice that MSF,
GenBank, and GDE2.2 are all available for saving a copy of an RSF file in a few
alternative formats. At this point Cancel the window and do not export any of these
formats. Realize that using this export route does not use the mask data to include or
exclude columns from the alignment. To take advantage of the mask data for
subsequent phylogenetic analyses, export the alignment using the method below.

2. Select all of the relevant sequences, as well as any desired mask sequence. Next, go
to the Functions menu, where all choices will be affected by the mask that has been
chosen, and choose Importing/Exporting ToFastA No options are required here; just
press Run to convert the portion of the alignment that is not masked into FASTA
format (APPENDIX 1B). FASTA is a good intermediate format on the way to PHYLIP’s
required format (UNIT 6.3). The new file will be displayed by SeqLab. The first part of
the author’s protein dataset FASTA format output file is shown in Figure 3.6.15.

*Notice that it excludes those positions that were masked with zero and that it now follows
all FASTA format conventions including the automatic conversion of all GCG style gap
periods and tildes to the more universal gap hyphen representation. This step, therefore,
circumvents the common “dot-to-dash” problem often encountered when converting GCG
sequence format to anything else.*

3. Close the ToFastA output window. It may be desirable to use the Output Manager to
save the file under a name that makes more sense to the individual user using the Save
As menu item.

4. Next, ReadSeq (APPENDIX 1E) is used to convert this FASTA format file to PHYLIP
compatible format. To do this, either exit SeqLab with the File menu Exit choice, or
temporarily switch to the background terminal window. If exiting, the user will
probably be asked if they want to save the RSF file and any changes in the list. If interested in saving the data, accept the suggested changes giving appropriate names, and SeqLab will close. This will return the user to the terminal window, formerly behind the SeqLab display, where ReadSeq can be run. This program can be used to change the FASTA format file into something acceptable to PHYLIP.

A limitation of ReadSeq is that it does not allow one to only choose a portion of an alignment, nor does it automatically convert dots and tildes to hyphens. However, since these points have been taken care of while in SeqLab, it will work just fine here.

5. Begin the program by typing readseq at the prompt in the terminal window. ReadSeq first prompts for an appropriate output file name, not an input file.

NOTE: Do not make a mistake in this step by giving the name of the input file first. If this happens, the input file will be overwritten while running the program. Then, when the program tries to read the input file, there will be nothing left!

6. Next choose 12 off of the ReadSeq menu for current PHYLIP format and then designate the input sequence filename. Do not use the GCG {*} designator; this is not a GCG program. Finally, after the program has read all of the input sequences, specify all the sequences by typing the word all. When the program again asks for an input sequence, press Enter to inform it that all of the sequences were input. When ReadSeq is finished, a terminal session screen trace, such as the one shown in Figure 3.6.16, is returned. Never mind if a padded to fit error message occurs; the program is just doing what it is supposed to do.

Do realize, though, that had ReadSeq not been used on the output from ToFastA to convert to PHYLIP, and instead a GCG MSF file was used as input, then an essential change would have to be made before it would be correct for PHYLIP. Periods and tildes will not work to represent indels (gaps) in any program other than GCG; they must all be changed to hyphens (dashes). The following, rather unusual, Unix command works well for this step from the command line, but there should be no need to use it, if the suggested procedure using GCG’s ToFastA program has been used:

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

7. Review the output. The first part of the example PHYLIP output file is displayed in Figure 3.6.17. Notice that the file begins with two numbers; the first shows the number of sequences in the matrix and the second lists the length of the matrix including any gaps and ambiguities. The next section lists the names of the sequences truncated to ten characters, if necessary, along with all the sequences printed in an interleaved fashion. Only the first sequence block lists the names, all others just give the sequence data itself.

Regardless of how files are taken from GCG format to acceptable PHYLIP format, one more technicality requires discussion. As mentioned elsewhere, the terminal ends of the data matrix should be evaluated. If any of the implied indels are uncertain (especially true if sequence lengths were different), then question marks (?), are usually more appropriate than hyphens (-). Leaving them hyphens could be misleading. Gaps in the data are represented by deletion symbols (-), which is logically correct in most cases. However, gaps at the ends and beginnings of sequences probably should not have hyphens unless it is really known that a deletion/insertion is responsible for the length discrepancy. Therefore, it is a good idea to edit the output from ReadSeq to replace leading and trailing hyphens in the alignment with question marks or the unknown characters n or x for DNA or protein sequences, respectively. This is also an excellent point at which to verify that the sequence names are exactly as the user wants them to appear in the final PHYLIP plots. PHYLIP sequence names can contain very limited punctuation and mixed capitalization, and can be up to ten characters in length. Be very careful with these edits so that the alignment does not shift out of phase.
SEARCHING PROSITE: GCG’S MOTIFS—A QUICK AND DIRTY METHOD

Many, many features have been described and cataloged in biological sequences over the years. Most of these have recognizable consensus patterns that allow screening of an unknown sequence for their occurrence. The PROSITE Dictionary of Protein Sites and Patterns (Bairoch, 1992; http://www.expasy.org/prosite) is one of the largest collections of such patterns in proteins. PROSITE is one of the quickest and easiest databases to search with a peptide sequence, just be careful of over-interpreting the results, as false positives can be quite common.

The GCG program Motifs performs this search. It displays an abstract with selected references for each motif signature found. The program can tolerate mismatches with a -MisMatch option and can show routine post-translational modification sites with a -Frequent option. Especially be wary of false positives if using either of these options. However, when used with care, Motifs can be a tremendous aid in ascertaining the function of an unknown peptide sequence. It can often lead to immediate answers and routes of investigation. It should always be utilized as it is just too fast and simple to ignore. As with the previous protocols in this unit, the only resource necessary is X-server access to a Unix GCG account (see Basic Protocol 1) and the unaligned sequences of interest loaded into SeqLab (see Support Protocol 1).

Start the Motifs program by selecting the names of all of the protein entries in SeqLab, as previously explained, and then going to the Functions Protein Analysis menu and picking Motifs. The Motifs program window will be displayed. Check the Save Results as Features in file Motifs.rsf button in the program window. This file contains RSF style annotation discovered by the Motifs program and it will be used below. None of the other options are required for this run so press the Run button. After a few moments the file, motifs.rsf, will be displayed. It is not interesting to read, so Close it and use the Output Manager to display the file with the .motifs extension. Carefully look over the text file that is displayed; the author’s abridged Motifs output file is shown in Figure 3.6.18. Notice the sites that have been characterized in these sequences and the extensive bibliography associated with them.

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

Among the other motifs discovered, the characteristic P-Loop, shown starting on the third panel of Figure 3.6.18, is defined as \((A, G)x4GR(S, T)\)—i.e., either an alanine or a glycine, followed by four of anything, followed by an invariant glycine-lysine pair, followed by either a serine or threonine. Exceptions are noted in the documentation. This particular site has been very well researched and many three-dimensional structures are available for it. It always has a \(\beta/\alpha/\beta\) secondary structure conformation and is sometimes known as the Rossman Fold.

Post-translational modification sites, such as glycosylation, phosphorylation, amidation, and myristylation, are commonly found in many proteins, but will only be listed if the -Frequent option is specified. However, realize that sites may be false positives, especially if the -Frequent option is used. This is always a danger with simple consensus-style searches. The GCG programs ProfileScan and HmmerPfam use a much more sensitive profile matrix approach to search the sequence with profiles, including most of PROSITE, and will be discussed later. Notice in the example above that Motifs discovered the truly positive GTP-binding elongation factor signature and the ATP/GTP-binding P-loop site, yet it also found two probable false positives, the prokaryotic membrane lipoprotein lipid attachment site and the FGGY family of carbohydrate kinases signature.
Close the Motifs output window after looking at it, and then load the `motifs.rsf` file into SeqLab. This will add the feature annotation created with the `-RSF` option. The location of the PROSITE signatures will now be included in the Editor sequence display. Again use the SeqLab Output Manager to do this, as discussed previously. Select the file `motifs.rsf`, then press the Add to Editor button and specify Overwrite Old with New to take the new `motifs.rsf` feature file and merge it with the old RSF file in the open editor. Close the Output Manager after loading the new RSF file. Look at the display using Features Coloring or Graphic Features to see the new annotation and determine if any differences are recognizable. The author’s unaligned dataset is illustrated in Figure 3.6.19 using Features Coloring, now annotated with its original database features as well as the new Motifs patterns and MEME discoveries (see Basic Protocol 3).

**SEARCHING MEME WITHIN GCG TO IDENTIFY MOTIFS**

Before performing multiple sequence alignment on any dataset, a powerful de novo motif discovery algorithm can be run. The algorithm is called Expectation Maximization; it uses Bayesian probabilities and unsupervised learning to find conserved motifs among a group of unaligned, ungapped sequences (Bailey and Elkan, 1994). The motifs do not have to be in congruent order among the different sequences (i.e., it has the power to discover “unalignable” motifs between sequences). This characteristic differentiates MEME (UNIT 2.4) from most other profile-building techniques. It is implemented in the Wisconsin Package as the MEME program and it produces output containing a multiple profile file as well as a readable report file. Its profile output serves as input to MotifSearch (Bailey and Gribskov, 1998). The author strongly suggests reading the MEME and MotifSearch chapters in the GCG Program Manual (`genmanual` at the command line or the Help buttons in the SeqLab program), as they explain the details of the algorithms quite well. As with the rest of this unit, X-server access to a GCG account is required to perform this protocol.

**Necessary Resources**

**Hardware**

Terminal or personal workstation with access to a Unix server running commercial GCG software

**Software**

MotifSearch and SeqLab (GCG Wisconsin Package; see Internet Resources)

X-server graphics communications software (`APPENDIX ID`)

*X-server emulation software needs to be installed separately on personal-style Microsoft Windows or Macintosh machines, but genuine X-Windowing comes standard with most Unix operating systems. Microsoft Windows machines are often set up with either XWin32 or eXceed to provide this function, while Macintoshes are often loaded with either MacX or eXodus software. The details of X are provided in `APPENDIX ID`. If the user is unsure of these procedures, ask for assistance from local computer support personnel.*

**Files**

Protein or DNA sequences of interest in GCG format (e.g., from LookUp in the GCG package; see Support Protocol 1; also see Internet Resources)

1. Open GCG and load desired sequences into SeqLab.
2. Select all of the sequences in the Editor window so that MEME runs on them all. Launch MEME off of the Functions Multiple Comparisons menu. A Which Selection window may pop up asking whether to use the Selected Sequences or Selected
In most cases the default parameters will work fine, but the algorithm can be sped up at the cost of sensitivity by decreasing the number of motifs to be found, by restricting the number of motifs found to exactly one in each sequence, and/or by decreasing the allowable motif window size. Again, reading UNIT 2.4 and the relevant GCG Program Manual chapters is suggested.

3. MEME output consists of two files: a .meme readable text file and a .prf multiple profile text file. MotifSearch will scan any dataset specified with the multiple profile file that MEME produced. A helpful thing to do is scan the original “training” dataset with which the profiles were created. This will annotate those regions that MEME discovered in the SeqLab Editor RSF file. After alignment, the MEME motifs that are alignable will all line up. Go to the Database Sequence Searching menu and select MotifSearch. Specify the query profile or profiles (i.e., the one that was just made) and change the Search set to the RSF dataset that has now been loaded into the Editor. Be sure to activate Save motif features to the RSF file.

4. The output will return with the .rsf file on top. Do not bother trying to read it; just close it. It contains the SeqLab rich sequence format for all the feature data discovered by MEME in the dataset. The .ms file contains the readable results of the search in list file format with $E$-value statistics and the number of motif hits for each fit. After the list file portion, a position diagram schematically describes the hits in each sequence. Take a moment to look it over by pressing the Display button in the Output Manager and then close it.

5. Use the Output Manager to merge the motifsearch.rsf feature file with the existing data already in the open SeqLab Editor. This will add the feature annotation created when MotifSearch -RSF option was activated. The location of each motif will be included in the Editor sequence display. To do this, again use the important Add to Editor Output Manager function. As before, specify Overwrite Old with New in the next window when prompted. Close the Output Manager after loading the new RSF file. Change Display: to Graphic Features and check out the additional annotation. Figure 3.6.20 illustrates the Graphic Features display at a 4:1 zoom ratio. The MEME motifs are clearly delineated in the unaligned dataset.

PROFILE-ANALYSIS: POSITION-SPECIFIC WEIGHTED SCORE MATRICES OF MULTIPLE SEQUENCE ALIGNMENTS

One-dimensional motifs are a way to capture the information of an important portion of an alignment; however, motifs cannot convey any degree of residue importance. For instance, in the GTP-binding P-Loop seen in previous protocols (see Basic Protocols 2 and 3), is it better to have an alanine or a glycine in the first position or does it matter? This lack of sense of importance causes a loss of sensitivity. More robust methods can convey the importance of each residue in the region.

Profile methods (Gribskov et al., 1987, 1989) and statistically based HMMER profiles (e.g., Eddy, 1996, 1998) enable the researcher to recognize features that may otherwise be invisible to individual sequence members. Both Gribskov and HMMER profiles are built from a set of prealigned sequences; however, HMMER profiles do not require that the alignment be as comprehensive and accurate. Profile analysis uses the full information content of an alignment. The greatly enhanced information content, over that of individual sequences, has the potential to find similar motifs in sequences that are only distantly related, more so than any other class of search algorithm. All other methods of describing
an alignment such as consensus or pattern description either throw away too much information or become too ambiguous. Profiles achieve additional sensitivity with a two-dimensional weight matrix approach versus a simple one-dimensional string technique. Furthermore, profiles are a special type of two-dimensional weight matrix in which conserved areas of the alignment receive the most importance and variable regions hardly matter.

Even though ProfileSearch requires some work to setup and run—i.e., a meaningful multiple sequence alignment must be assembled and refined, ProfileMake needs to be run, and the search job itself takes quite a long time—it is well worth the bother. ProfileSearch and HmmerSearch are incredibly CPU intensive, so be sure to submit them as early as possible (if launched from the command line, use the -Batch option).

**Necessary Resources**

**Hardware**
Terminal or personal workstation with access to a Unix server running commercial GCG software

**Software**
SeqLab (GCG Wisconsin Package; see Internet Resources)
X-server graphics communications software (*APPENDIX 1D*)

X-server emulation software needs to be installed separately on personal-style Microsoft Windows or Macintosh machines, but genuine X-Windowing comes standard with most Unix operating systems. Microsoft Windows machines are often set up with either XWin32 or eXceed to provide this function, while Macintoshes are often loaded with either MacX or eXodus software. The details of X are provided in *APPENDIX 1D*. If the user is unsure of these procedures, ask for assistance from local computer support personnel.

**Files**
Multiple sequence alignment loaded into SeqLab (see Basic Protocol 1).

**Stepwise protocol procedure**

Create a traditional Gribskov style profile

1a. To run ProfileMake, be sure that all of the alignment sequences are selected. Do not select a mask sequence, as profiles need to include all of the ambiguity of the alignment within the region being used, and would be the wrong length if any alignment columns were excluded. Then, based on previous observations and the experimental objectives, select the longest most conserved overall sequence length available. Restrict the length of the profile so that jagged ends in the alignment are excluded. In SeqLab do this through the Edit Select Range menu. Select and then Close the box.

Another effective strategy is to develop multiple shorter profiles centered about the similarity peaks of the alignment. These will most likely correspond to functional or structural domains in the protein.

2a. After the range is selected, use the Functions Multiple Comparison ProfileMake menu and choose Selected region in the Which selection dialog box. The Options menu from the ProfileMake dialog box can be used to specify the -SeqOut command option by checking Write the Consensus into a Sequence File and giving it an appropriate name. This will generate a normal sequence file of the consensus in addition to the profile file. Play with any of the other desired options, such as the scoring matrix (*UNIT 3.5*), and then Close the Options box and Run ProfileMake.
3a. After running ProfileMake, the top window returned will display the profile consensus sequence. All positions will be filled and there will be no gaps. This is because the profile algorithm will decide on the most conserved residue for each position, regardless. The header contains information relating to the creation of the sequence through ProfileMake. Close the consensus window. The Output Manager will also list a .prf file. This is the profile itself.

4a. Use the Save As function on the profile in the Output Manager, giving it an appropriate name that is easily recognizable and retaining the .prf extension. Close the Output Manager.

**Query databases with the profile**

5a. ProfileSearch is launched through SeqLab with the Functions menu. Select Database Sequence Searching ProfileSearch, specify the Query Profile in the File Chooser, and click OK. Search whichever protein database is preferred, though to reduce CPU load the author suggests just using NRL_3D for now. (The author prefers to run ProfileSegments separately after the ProfileSearch is done.) Therefore, uncheck ProfileSegments to prevent ProfileSearch’s output from automatically being passed to ProfileSegments. This way the ProfileSearch output file can be edited so that ProfileSegments only makes pairwise or multiple alignments of sequences onto the profile. Also, under Options use the -MinList option by changing Lowest Z-score to report in output list from 2.5 to 3.5 or higher. MinList sets a list Z-score cut-off value, a handy way to limit the output list size. Close the Options window, making sure that How: Background Job is selected, and then click Run.

As in BLAST and FASTA searches, ProfileSearch estimates a realistic significance parameter. In the case of profile searching, it is a Z-score based on the distance, in the number of standard deviations, from the rest of the insignificant database matches. ProfileSearch Z-scores are normalized and reflect the significance of the results. Here, rather than randomizing sequences to evaluate a Z-score, as is done in Monte Carlo approaches, it is calculated based on all of the nonsimilar sequences from the database search, similar to the way that FASTA calculates its Expectation values. Pay particular attention to the reported Z-scores in the output. As with Monte Carlo approaches, Z-scores <3 are probably not worth considering, from around 4 to 7 may be interesting, and >7 are most probably significant and should definitely be evaluated further. Remote similarities can be found that all other methods will miss using profile analysis properly—it is extremely powerful.

6a. Look at ProfileSearch output carefully. There is a good chance that other search algorithms will have missed some of the sequences listed as significant matches. If launched from SeqLab, the output will be located in the working directory and it will have a cryptic name of the form profilesearch_some-number.pfs. A greatly abridged screen trace of a sample ProfileSearch output with a profile built from most of the length of the author’s elongation factor 1α alignment is shown in Figure 3.6.21. Many of the entries that would normally be expected have been excluded and some of the surprises have been left.

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

Notice the very clean Z-score demarcation in Figure 3.6.21 between the EF-1α orthologs, with Z-scores above 100, and all the GTP-binding proteins below that with Z-scores from below 100 to almost 5, and what is most likely just noise, with Z-scores of 5. This search has entries in common with previous searches, but there are also substantial differences.
This is another reason why it is always a good idea to run as many different types of analyses as is practical.

7a. The program ProfileSegments makes BestFit style alignments of the results of a ProfileSearch. A great option in ProfileSegments, -MSF, allows the user to prepare a multiple sequence alignment of the ProfileSearch segments.

This can be very helpful for merging ever-increasingly distant sequences into an alignment (but also see HmmerAlign below). The full information content of the profile including the importance of the conserved portions of the alignment is used in this alignment procedure. When checking out a ProfileSearch output, the author often edits it to exclude (or comment out by placing an exclamation point at the beginning of the entry’s line) most of the sequences that are expected to be found by the search, except a few positive controls (i.e., in this example most of the EF-1α’s). In such a case, be sure not to change the header portion of the file, as it specifies the profile’s directory location.

8a. Alignments are made from the modified ProfileSearch output file with ProfileSegments. When running ProfileSegments be sure to set the list size big enough to include all of the relevant sequences remaining in the ProfileSearch output. Another handy option is -Global versus the -Local default; this will force full-length alignments, which might be what is wanted, especially if trying to build up a multiple sequence alignment. A screen snapshot centered about the tRNA binding region of a ProfileSegments -MSF -Global alignment made from many of the entries from the above ProfileSearch example aligned against the example EF-1α profile is shown in Figure 3.6.22.

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

Construct a HMMER profile

1b. Build a HMMER profile of an alignment in SeqLab similar to what would be done for a traditional profile. Select all of the relevant sequences (except a mask) and perhaps a region within them to exclude jagged unalignable ends. Go to the Functions HMMER menu and pick HmmerBuild. Specify Selected Region rather than Selected Sequences if restricting the profile’s length. Accept the default Create a New HMM and specify some Internal name for the profile HMM. Also specify the Type of HMM to be Built (Multiple Global is the default).

There is a big difference between HmmerBuild and other profile building programs. When the profile is built, it is necessary to specify the type of eventual alignment it will be used with, rather than when the alignment is run. The HMMER profile will either be used for global or local alignment, and will occur multiply or singly on a given sequence. Weighting is also handled differently in HMMER than it is with Gribskov profiles. To use a custom weighting scheme (e.g., if the RSF file weight values have been modified for ProfileBuild), it is necessary to tell HmmerBuild not to use one of its built-in weighting schemes with the -Weighting=N option. Otherwise HmmerBuild’s internal weighting algorithm will calculate the best weights automatically, based on the sequence similarities using a cluster-analysis approach. It again becomes important to understand the types of biological questions being asked to rationally set many of the program parameters.

2b. Notice HmmerCalibrate is checked by default. The completion of HmmerBuild automatically launches a calibration procedure that increases the speed and accuracy of subsequent analyses with the resultant profile. The other HmmerBuild options can
be explored, but read the Program Manual first. For now, accept the default HmmerBuild parameters and press Run. The output is an ASCII text profile representation of a statistical model, a hidden Markov model, of the consensus of a sequence family, deduced from a multiple sequence alignment.

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

3b. To use a HMMER profile as a search probe, go to the Functions menu and pick HMMER HmmerSearch. Specify the new HMMER profile by clicking Profile HMM to Use as Query and using the File Chooser window to select the correct HMMER profile. Either accept the default Sequence Search Set PIR:* specification or choose other sequences to search. Again, NRL_3D is a nice small database to test. HmmerSearch has similar cutoff parameters as other GCG database searches, that is, the size of the output can be restricted based on significance scores and the number of pairwise alignments displayed can be limited.

HmmerSearch is very slow because it is a true dynamic-programming implementation (UNIT 3.1)—i.e., a HMMER profile matrix against a whole database. So definitely run it in the background when using SeqLab or, if at a terminal session, use the -Batch command line option. If the server has multiple processors, HmmerSearch supports the multithreading -Processors = x option to speed things up. Run the program when the options are set as desired. The output is huge but very informative. Everything is based on significance expectation value scores. The top portion is a list of best hits based on all domains, the second section is the GCG list file portion of the best domain hits, next pairwise alignments are given, and finally a score distribution is plotted. Since it is a GCG list file, it can be read by other GCG programs, in particular HmmerAlign.

4b. To align new sequences with the HMM profile, launch HmmerAlign from the Functions HMMER menu by picking HammerAlign. Specify the correct HMMER profile with the Profile HMM to Use button and pick the desired sequences to align to the profile with the Sequences to align button.

1EFT is one of the most similar elongation factor 1α homologues to the author’s lower eukaryote EF-1α profile example that has a solved structure. Therefore, an alignment of its primary sequence with structural annotation against the sample dataset should allow a decent inference of secondary structure across the entire alignment. This is the basis of homology modeling. Figure 3.6.23 shows the results of a HmmerAlign run on NRL_3D:1EFT, the EF-Tu structure from Thermus aquaticus, against the example EF-1α HMMER profile and its associated alignment. The inferred secondary structure is illustrated in the Features Coloring graphic of Figure 3.6.23 by highlighting the α-helices in red. These appear medium gray in the figure, but are identified with the structural annotation line comment alpha helix.

HmmerAlign can be an incredible help to people working with very large multiple alignments and for adding newly found sequences to an existing alignment regardless of size. Somewhat similar in concept to the -MSF option of ProfileSegments, it takes a specified profile, in this case a HMMER profile, and aligns a specified set of sequences to it, to produce a multiple sequence alignment based on that profile. Unlike ProfileSegments, HmmerAlign takes any GCG sequence specification as input, not just the output from its own database-searching program. It is much faster to create very large multiple alignments this way, versus using PileUp, on an entire large dataset. The rationale being to take the time to make a good small alignment and HMMER profile and then use that to build up the original larger and larger. The alignment procedure used by HmmerAlign is a full-blown, recursive, dynamic programming implementation. HmmerAlign can also use its profile to align one multiple alignment to another and produce a merged result of the two. Using the original alignment with which the profile was made against another sequence
set is very fast; it is the -MapAlignment = some.rsf{*} option and it provides an exact, nonheuristic alignment. A heuristic (optimality is not guaranteed) solution is provided when using any other alignments. To use this option, choose Combine Output Alignment and Another Alignment in the SeqLab HmmerAlign Options window. This will launch the command line -Heuristic=some.msf{*} option.

5b. As with Motifs and MotifSearch, HmmerPfam can help build up the annotation of an RSF file. This program scans sequences against a library of HMMER profiles, by default the Pfam library (UNIT 2.5), a database of protein domain family alignments and HMMs (the Pfam library is under copyright to the The Pfam Consortium, 1996 to 2000). Select all of the protein sequences (do not select annotation or mask lines) and launch the program through the Functions HMMER HmmerPfam menu. Save the best scoring profile HMMs as an RSF file and give it an appropriate name. It is possible to check out the options if desired. It may be useful to reduce the Expectation cutoff values. Run the program.

It can take quite a while to run, so do not wait for it to finish. Proceed with the protocol instead.

6b. When it has finished, add its RSF output file to the editor display as before, with the Output Manager’s Add to Editor and Overwrite Old with New functions.

The output .hmmerpfam file lists Pfam domain matches ranked by E-values and with the -RSF option writes the domain identification and E-value as a feature in an RSF file. The screen snapshot shown in Figure 3.6.24 shows the author’s sample alignment.

COMMENTARY

Background Information

The power and sensitivity of sequence-based computational methods for discovering similarities dramatically increases with the addition of more data. More data yields stronger analyses, if done carefully! Otherwise, it can confound the issue. The patterns of conservation become clearer by comparing the conserved portions of sequences amongst a larger and larger dataset. Those areas most resistant to change are functionally the most important to the molecule. The basic assumption is that those portions of sequence of crucial functional value are most constrained against evolutionary change. They will not tolerate many mutations. Not that mutations do not occur in these portions, just that most mutations in the region are lethal so they are never seen. Other areas of sequence are able to drift more readily, being less subject to evolutionary pressure. Therefore, sequences end up a mosaic of quickly and slowly changing regions over evolutionary time. It is those slowly changing regions that the researcher is interested in recognizing through multiple sequence comparisons. However, in order to learn anything by comparing multiple sequences, it is necessary to know how to compare them.

As with pairwise dynamic programming, looking at every possible position by sliding one sequence along every other sequence just will not work for alignment. Therefore, dynamic programming reduces the problem down to $N^2$, where $N$ is the length of the longer sequence. But how is it possible to work with more than just two sequences at a time? It becomes a much harder problem. The researcher could painstakingly manually align all of the sequences using some type of editor, and many people do just that, but some type of an automated solution is desirable, at least as a starting point to manual alignment. However, solving the dynamic programming algorithm for more than just two sequences rapidly becomes intractable. Dynamic programming’s complexity, and hence its computational requirements, increases exponentially with the number of sequences in the dataset being compared (complexity = $[\text{sequence length}]^{\text{sequence}}$). Pairwise dynamic programming solves a two-dimensional matrix, and the complexity of the solution is equal to the length of the longest sequence squared. A three-member, standard, dynamic programming sequence comparison would be a matrix with three axes, the length of the longest sequence cubed, and so forth. It is possible to at least draw a three-dimensional matrix, but more than that becomes impossible to even visualize. This is an $N$-dimensional matrix, with $N$ equal to the num-
How the alignment algorithm works

The most common implementations of automated multiple alignment modify dynamic programming by establishing a pairwise order in which to build the alignment. This modification is known as pairwise, progressive, dynamic programming. Originally attributed to Feng and Doolittle (1987), this variation of the dynamic programming algorithm generates a global alignment, but restricts its search space at any one time to a local neighborhood of the full length of only two sequences. Consider a group of sequences. First, all are compared to each other pairwise, using normal dynamic programming. This establishes an order for the set, most to least similar. Subgroups are clustered together similarly. Then the top two most similar sequences are aligned using normal dynamic programming. Next a consensus of the two is created and that consensus is aligned to the third sequence using standard dynamic programming. A consensus of the first three sequences is created and aligned to the fourth most similar. This process continues until it has worked its way through all sequences and/or sets of clusters. The pairwise progressive solution is implemented in several programs. ClustalW (UNIT 2.5; Thompson, et al., 1994) and its multiplatform graphical user interface ClustalX (UNIT 2.5; Thompson, et al., 1997) are commonly used worldwide. The Wisconsin Package program PileUp also implements this method and is the centerpiece of this unit.

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

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

Motifs: Regular expression descriptions of consensus elements

After having created and refined a multiple-sequence alignment, and after having recognized consensus elements within that alignment, how can the biological information of that constrained sequence region be captured? One very simplistic approach is to look at the alignment, see that certain regions are conserved, and create a simple, one-dimensional consensus of that region. A multiple-sequence alignment of elongation factor Tu/1α from several different organisms in Figure 3.6.25 illustrates the conservation of the first of several GTP-binding domains in these proteins—i.e., the region around position twenty.

Based on experimental evidence, it is known that the indicated region bounded by the glycine and serine in Figure 3.6.25 is essential (e.g., see Saraste et al., 1990). So the various residues in those locations are merely counted and the most common one assigned to the consensus. Simple, but what about the fact that the middle histidine is not always a histidine. In this data set, just as often it is a serine and sometimes it is an alanine. Other positions are also seen not be invariant. And there are many other members of this gene family not being represented here at all. A consensus is not necessarily the biologically correct combination. How is this other information included? A simple consensus throws much of it away; therefore, it is necessary to adopt some sort of standardized ambiguity notation. The trick is to
define a motif such that it minimizes false positives and maximizes true positives (i.e., it needs to be just discriminatory enough). The development of the exact motif is largely empirical. A pattern is made, tested against the database, then refined, over and over; however, when experimental evidence is available, it is always incorporated. This approach is known as motif definition and fortunately Amos Bairoch has done it for a huge number of sequences!

His database of cataloged structural, regulatory, and enzymatic consensus patterns is a protein signature database, the PROSITE Dictionary of Protein Sites and Patterns (Bairoch, 1992). Bairoch’s compilation, now named the PROSITE Database of Protein Families and domains, contains 1079 documentation entries that describe 1459 different patterns, rules, and profile matrices (Release 16.33, January 25, 2001). Pattern descriptions for these characteristic local sequence areas are variously and confusingly known as motifs, templates, signatures, patterns, and even fingerprints, but do not let the terminology be bewildering. Those that GCG’s Motifs program can access are one-dimensional regular-expression descriptions that encode ambiguity of some sort of functional or otherwise constrained consensus region of a sequence alignment (e.g., glycosylation and phosphorylation sites, SH3-binding sites, nuclear localization sequence, and enzymatic active sites). See the FindPatterns GenManual entry for pattern syntax rules. Motifs may or may not represent sequence homology and may or may not encompass an entire structural domain, they do not necessarily signify known function, nor common origin, but it is a very quick and easy search to accomplish so it is always recommended.

Profile algorithms

Given a multiple sequence alignment, how is the extra information contained within it used to find ever more remotely similar sequences? What method is used to search and explore into and past Russell Doolittle’s (1986) Twilight Zone—i.e., those similarities below ~25% identity, those Z-scores below ~4, or those BLAST/Fast E-values above ~1 × 10⁻³ or so? Just because a similarity score between two sequences is quite low, it is not automatically known that the two structures do not fold in a similar manner or perform a similar function. In fact, there is no idea of homology at all!

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

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

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

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

**Traditional profiles**

The Gribskov et al., (1987 and 1989) method is implemented in the Wisconsin Package with a series of five programs:

1. ProfileMake: Creates the profile from a multiple sequence alignment.
2. ProfileSearch: Searches other sequences (the database) with a profile.
3. ProfileSegments: Aligns the output list of a ProfileSearch.
4. ProfileGap: Aligns individual sequences to a profile.
5. ProfileScan: Searches validated PROSITE-based profile library built by Gribskov.

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

The author has created a small profile of just the P-Loop region to show how to interpret a profile matrix. The greatest amount of conservation of the P-Loop region is centered about absolute residue position twenty or so. What happens if the profile is prepared around just this region? What does it look like? It is a big table of numbers that does not make a whole lot of sense to many people, but it is a tremendously powerful tool in subsequent analyses. Other programs can read and interpret this alignment-customized scoring matrix to perform very sensitive database searches and further alignments by utilizing the information within the matrix that penalizes misalignments in phylogenetically conserved areas more than in variable regions. Check out the P-Loop profile in Figure 3.6.26.

On closer inspection, the matrix begins to make some sense. Across the top are all possible residues. The first column is that residue that received the highest score in the program, the consensus. But notice the interior of the matrix. Numbers bounce all over the place, from 150 to −87. What is that all about? Without going into the mathematics involved, based on the alignment that was fed into it and the initial scoring matrix used (by default BLOSUM62, but others can be specified) the program has scaled those positions which are most important up and those positions least important down. For instance the threonine at position 27 in the alignment is the only residue absolutely conserved throughout. Therefore, it gets the highest score. The aspartate at position 22 substituted with a tryptophan would never happen, hence the −87 score. Tryptophan is the most conserved residue on both the PAM and BLO-
SUM matrix series and the aspartate is conserved at all positions in the alignment that have residues at that position. The negative matrix score of any substitution to tryptophan multiplied by the high conservation at that position for aspartate equals the most negative score in the profile. How about those positions where the conservation is not as striking? Position 16 is a good one to pick. Valine is the assigned consensus residue because it has the highest score (37), but glycine also occurs several times, a score of 20. However, other residues are ranked in the substitution matrices as being quite similar to valine; therefore, isoleucine and leucine also get similar scores, 24 and 14, and alanine occurs some of the time in the alignment so it gets a comparable score, 15. But realize that all of these numbers are much less than the highest numbers in the matrix. Because the position is not well conserved, all the values are fairly mediocre at that position.

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

**HMMER: Hidden Markov modeling and profiles**

As powerful as Gribskov style profiles are, they require a lot of time and skill to prepare and validate, and they are heuristics based. An excess of subjectivity and a lack of formal statistical rigor also contribute as drawbacks. In collaboration with Sean Eddy (1996 and 1998), GCG has incorporated the HMMER (pronounced hammer) package into the Wisconsin Package, version 10.2. HMMER uses hidden Markov modeling, with a formal probabilistic basis and consistent gap insertion theory, to build and manipulate HMMER profiles and profile databases, to search sequences against HMMER profile databases and visa versa, and to easily create multiple sequence alignments using HMMER profiles as a seed. Again, GCG has taken the time to write an excellent essay in the Program Manual on HMMER, what hidden Markov models are, and how the algorithms work. It is strongly suggested that the user reads it, as well as each individual HMMER program description, at some point. There are nine different HMMER programs in GCG: HmmerAlign, HmmerBuild, HmmerCalibrate, HmmerEmit, HmmerFetch, HmmerIndex, HmmerPfam, and HmmerSearch. The take-home message is that HMMER profiles are much easier to build than traditional profiles and they do not need to have nearly as many sequences in their alignments in order to be effective. Furthermore, they offer a statistical rigor not available in Gribskov style profiles, and they have all the sensitivity of any profile technique.

**Coding DNA issues**

When dealing with very similar sequences, it is often best to align DNA sequences along with their corresponding proteins (SeqLab’s GROUP function is very helpful for this). Phylogenetic analyses are then performed on the DNA rather than on the proteins. This is especially important when dealing with datasets that are quite similar since the proteins may not reflect many differences hidden in the DNA. Furthermore, many people prefer to run phylogenetic analyses on DNA rather than protein regardless of how similar they are, as the multiple substitution models are much more robust for DNA. In fact, many phylogenetic inference algorithms do not even take advantage of amino acid similarity when dealing with protein sequences; they only count identities! However, the more diverged a dataset becomes, the more random third and eventually first codon positions become, which introduces noise (error) into the analysis. Therefore, often third positions and sometimes first positions are masked out of datasets. Just like in most of computational molecular biology, one is always balancing signal against noise. Too much noise or too little signal both degrade the analysis to the point of nonsense.

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

1. The easy case is where the DNA can be aligned directly. If the DNA sequences are directly alignable because they are quite simi-
lar, then merely create the DNA alignment. Next, use the Edit menu Translate function and the Align Translations option to create aligned corresponding protein sequences. Select the region to translate based on the CDS reference in each DNA sequence’s annotation. Be careful of CDS entries that do not begin at position 1. The GenBank CDS feature annotation /co-don_start = identifies at which position the translation begins within the first codon listed. It may also be necessary to trim sequences down to just the relevant gene, especially if they are genomic. Their protections will have to be changed 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 much more difficult case is where the protein sequence is needed to create the alignment because the DNA is not directly alignable. In this case it is necessary to load the protein sequences first, create their alignment, and then load their corresponding DNA sequences. The DNA sequence accession codes can be found in the annotation of the protein sequence entries. Next, translate the unaligned DNA sequences into new protein sequences with the Edit-Translate function using the align translations option and GROUP these to their corresponding DNA sequences, just as above. However, this time the DNA along with their translated sequences are not aligned as a set, only 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 the 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 are aligned to the rest of the set. Merge the newly aligned sequences into the existing alignment, GROUPing as it progresses, and then start on the next one. It sounds difficult, but since two identical protein sequences are being matched, the DNA translation and the original aligned protein, it is really not too bad. The GROUP function keeps everything together the way it should be, so that the original alignment is not lost as the residues are spaced apart to match them up with their respective codons. Some codons may become spaced apart in this process and will have to be adjusted afterwards. As usual, save work often.

Figure 3.6.27 shows a screen dump of the author’s sample dataset part way through this process.

**The protein system example**

The elongation factors are a vital protein family crucial to protein biosynthesis. They are ubiquitous to all cellular life and, in concert with the ribosome, must have been one of the very earliest enzymatic factories to evolve. The author uses the elongation factor subunit known as 1-alpha (EF-1α) in Eukaryota and Archaea and called elongation factor Tu in (Eu)Bacteria (and Euk’ and Arch’ plastids) as an example throughout this unit. It is essential in the universal process of protein biosynthesis and promotes the GTP-dependent binding of aminoacyl-tRNA to the A-site of the intact ribosome. GTP is hydrolyzed to GDP in the process. Elongation factor 1α/Tu has guanine nucleotide, ribosome, and aminoacyl-tRNA binding sites. As shown in Table 3.6.1, there are three distinct types of elongation factors that work together to help perform the vital function of protein biosynthesis. In EF-1α, a specific region is involved in a conformational change mediated by the hydrolysis of GTP to GDP. This region is conserved in both EF-1α/Tu and EF-2/G and seems to be typical of GTP-dependent proteins which bind noninitiator tRNAs to the ribosome.

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

The three-dimensional structure of elongation factor 1α/Tu has been solved in about 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 *thermophilus* structures have been determined (1TTT, 1EFT, and 1AIP), and even the cow EF-1α has been determined.
Most of the structures show the protein in complex with its nucleotide ligand, but some show the ternary complex. The *Thermus aquaticus* structure as drawn by RasMol (Sayle and Milner-White, 1995) is shown in Figure 3.6.28 (Kjeldgaard, 1993). Notice that half of the protein has well defined $\alpha$-helices and the rest is rather unordered coils. GTP fits right down in amongst all the helices in the pocket.

The *T. aquaticus* structure has six well-defined $\alpha$-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 $\alpha$-helices at residues 47 to 51 and 54 to 59. The guanine nucleotide–binding site involves the following regions: residues 18 to 25, residues 81 to 85, and residues 136 to 139. Residue 8 is associated with aminoacyl-tRNA binding.

Multiple sequence analysis with this protein clearly delineates these functional and structural regions as well as other conserved sites. Furthermore, because of strong evolutionary pressure resulting in very slow divergence and its ubiquity, EF-1$\alpha$/Tu is an appropriate gene on which to estimate early life questions. In fact, a series of papers in the early 1990s, 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 $\alpha$-gene paralogue EF-1$\beta$ to their $\alpha$ dataset to root the tree. Therefore, to further investigate the phylogenetic potential of this system, the author has restricted the example in this unit to a subset of lower eukaryotic EF-1$\alpha$ sequences. These include many protists and algae, but exclude much of the Crown group, including all of the higher plants, true fungi, and metazoans. As such, it may be an appropriate dataset with which to ask early branching order questions in deep eukaryotic evolution.

---

**Critical Parameters and Troubleshooting**

**The initial dataset**

One of the more difficult aspects of multiple sequence alignment and analysis is knowing with what sequences it should be attempted. Any list from any program will need to be restricted to only those sequences that actually should be aligned. Make sure that the group of sequences that are aligned are in fact related, that they actually belong to the same gene family, and that the alignment will be meaningful. In these days of huge genome projects and massive databases, one important slant is a data mining question—i.e., figuring out just which sequences to align from the huge number available that are similar to the query. This question is particularly appropriate here since there are an enormous number of elongation factors present in the databases. So often it depends on the type of scientific question being asked in the user’s research. Is the interest in predicting the structure or the function of a particular molecule? How about in ascertaining the evolution of a paralogous gene family within a species as the result of gene duplications? What about the evolution of several species based on an analysis of the orthologues present in several different species? Clearly the dataset to be used is directly molded by the question being asked.

In the reader’s research settings, and depending on the type of questions that are being asked, it may be desirable to create very large alignments by screening all available databases for sequences of significant similarity to the query. Thus, a discussion about size is appropriate. The Wisconsin Package’s restrictions, as of version 10.2, allow individual sequences to be a maximum of 350 Kb in length (longer entries are cut into overlaps in database creation steps), though SeqLab can display longer sequences. It may be useful to load a longer sequence into SeqLab if one is working on the

---

### Table 3.6.1 Elongation Factor Nomenclature in Eukaryota and (Eu)Bacteria

<table>
<thead>
<tr>
<th>Eukaryota</th>
<th>(Eu)Bacteria</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF-1$\alpha$</td>
<td>EF-Tu</td>
<td>Binds GTP and an aminoacyl-tRNA; delivers the latter to the A site of ribosome.</td>
</tr>
<tr>
<td>EF-1$\beta$</td>
<td>EF-Ts</td>
<td>Interacts with EF-1$\alpha$/Tu to displace GDP allowing the regeneration of GTP-EF-1$\alpha$/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>

---

*aThe nomenclature in Archaea has not been completely worked out and is often contradictory.*
genome scale, and wants to extract subranges from that entry. The MSF file format can hold up to 500 sequences; however, the RSF format can hold much more, limited only by system memory. This allows programs such as HmmerAlign to produce multiple sequence alignment output larger than 500 sequences. PileUp itself can handle a sequence alignment up to 7,000 characters long, including gaps. Input sequences are restricted to a length of 5,000 characters by default. The overall surface-of-comparison is restricted to 2,250,000 with the default program, a bit more than all the residues or bases plus all the gaps in the alignment. Alternative executables are provided with the package for allowing 10,000, 15,000, and 20,000 character input, though these executables are usually not scripted into SeqLab. Launch them from the command line withpileup_10000, pileup_15000, and pileup_20000, respectively. The take home message is that it is possible to make really huge alignments if desired. It is all up to what is really necessary to answer the biological questions being asked.

A major complication is molecular sequence data format conventions. Each suite of programs to come along seems to require its own unique format. The major databases all have their own. For example Clustal has its own, and even the database similarity searching program FASTA has a sequence format associated with it (Appendix IB). GCG Wisconsin Package sequence format exists both as single and multiple sequence format (MSF), and GCG’s SeqLab has its own format called rich sequence format (RSF) that contains both sequence data and reference and feature annotation. PAUP* has a required format called the NEXUS file and PHYLIP has its own unique input data format requirements.

Alignment gaps are another problem. Different program suites may use different symbols to represent them. Most programs use hyphens (‐), while the Wisconsin package uses periods (·). Furthermore, not all gaps in sequences should be interpreted as deletions. Interior gaps are probably okay to represent this way, as regardless of whether a deletion, insertion, or duplication event created the gap, logically they will be treated the same by the algorithms. These are known as 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 simply place holders for the sequence. Therefore, it is safest to manually edit an alignment to change leading and trailing gap symbols to exes (x) which mean unknown amino acids, ens (n) which mean unknown bases, or question marks (?). The last of these (?) is supported by many programs, but not all, including GCG, and means unknown residue or indel. This will assure that those programs where it matters do not make incorrect assumptions about the sequences.

Reliability

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

Researchers have successfully used the conservation of covarying sites in ribosomal and other structural RNA alignments to assist in alignment refinement. That is, as one base in a stem structure changes, the corresponding Watson-Crick paired base will change in a corresponding manner. This process has been used extensively by the Ribosomal Database Project at the Center for Microbial Ecology at Michigan State University to help guide the construction of their rRNA alignments and structures. The Web site is http://rdp.cme.msu.edu/html.

Be sure an alignment makes biological sense—i.e., align things that make sense to align! Remember the old adage “garbage in, garbage out.” Beware of comparing apples and
oranges. If creating alignments for phylogenetic inference, either make paralogous comparisons (i.e., evolution via gene duplication) to ascertain gene phylogenies within one organism, or orthologous (within one ancestral loci) comparisons to ascertain gene phylogenies between organisms which should imply organismal phylogenies. Try not to mix them up without complete data representation. Lots of confusion can arise, especially if all of the data are not available and/or if the nomenclature is contradictory, as extremely misleading interpretations can result. Be wary of trying to align genomic sequences with cDNA when working with DNA; the introns will cause all sorts of headaches. Similarly, do not align mature and precursor proteins from the same organism and loci. It does not make evolutionary sense, as one is not evolved from the other, rather one is the other. These are all easy mistakes to make, so try as much as possible to avoid them. Biocomputing is always a delicate balance of signal against noise and sometimes it can be quite the balancing act!

Conclusions
The comparative method is a cornerstone of the biological sciences. Multiple sequence alignment is the comparative method on a molecular scale, and is a vital prerequisite to some of the most powerful biocomputing techniques available. Understanding the algorithm and program parameters for multiple alignment is the only way to rationally know what is appropriate. Oftentimes it will be necessary to deal with very large datasets, or alignments may have to be adjusted manually. A comprehensive multiple sequence editor such as the GCG Wisconsin Package SeqLab graphical user interface can be a lifesaver in these situations.

One point that needs to be made is that the previous techniques were performed largely using GCG’s suggested defaults. This usually will work, but it is a good idea to think about what these default values imply and adjust them accordingly, especially if the results seem inappropriate after running through a first pass with the default parameters intact. Another vital point that cannot be repeated often enough, is the dramatic importance of multiple sequence alignments. All subsequent analyses are absolutely dependent upon them, especially phylogenetic inference. Also, if building multiple sequence alignments for phylogenetic inference, do not base an organism’s phylogeny on just one gene. Many complicating factors can make interpretation difficult. Weird phylogenies can be the result of several things: bad alignments, insufficient data, abjectly incorrect models, saturated positions (homoplasy), compositional biases, and/or horizontal gene transfer, use several genes instead. The Ribosomal Database Project (RDP; http://rdp.cme.msu.edu/html) provides a good and largely accepted alignment and phylogenetic framework with which other phylogenies can be compared. The complete RDP can be installed on a local GCG server in aligned GCG format, given sufficient interest and a cooperative GCG systems manager, which could then be used in the same manner as the sequences explored in this chapter. Otherwise, desired data subsets can be downloaded from RDP and loaded into SeqLab. Anytime the orthologous phylogenies of organisms based on two different genes do not agree, there is either some type of problem with the analysis, or a case of lateral transfer of genetic material has been found. Paralogous gene phylogenies are another story altogether and should be based, if at all possible, on sequences from the same organism.

Furthermore, keep in mind that this unit was written using a very similar, quite easily aligned dataset. This was done so that individuals working through the text on-line would be able to proceed in real time. However, many datasets that will be encountered, especially the very-interesting ones, will be composed of much more divergent sequences, such as distantly related protein domains, paralogous sequences all within one organism, or orthologous sequences between widely separated taxa in the tree of life. These are the situations that will present vexing alignment problems and difficult editing decisions, and are when serious thought is required.

Gunnar von Heijne (1987) in his quite readable but somewhat dated treatise, Sequence Analysis in Molecular Biology: Treasure Trove or Trivial Pursuit, provides an appropriate conclusion: “Think about what you are 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 is all it takes.”

**Suggestions for Further Analysis**

As mentioned previously, a primary application of multiple sequence alignment is their use in molecular phylogenetics inference applications. Please refer to Chapter 6 of this book for the theory and methodology of evolutionary tree estimation based on multiple molecular sequence alignments.

**Literature Cited**


**Internet Resources**


The Phylogeny Inference Package (PHYLIP), version 3.5+, is public domain software distributed by the author, J. Felsenstein. It is available online from the Department of Genomes Sciences, University of Washington, Seattle.

http://www.uni-giessen.de/~gx1052/ECDC/ecdc.htm

The E. coli Database Collection (ECDC). The K12 chromosome. Available online from Justus-Liebig-Universitaet, Giessen, Germany.


The Wisconsin package, version 10.3, is available from the Genetics Computer Group (GCG), a part of Accelrys, which is in turn a subsidiary of Pharmacopeia, and is copyright protected (1982–2002). The home page includes a copy of the program manual.

The Wisconsin Package provides a comprehensive toolkit of almost 150 integrated DNA and protein analysis programs, from database, pattern and motif searching, fragment assembly, mapping, and sequence comparison, to gene finding, protein and evolutionary analysis, primer selection, and DNA and RNA secondary structure prediction. The powerful SeqLab X-windows based graphical user interface (GUI) is a front end to the package. It provides an intuitive alternative to the Unix command line by allowing menu-driven access to most of GCG’s programs. SeqLab is based on Steve Smith and collaborators’ (1994) genetic data environment (GDE) and makes running the Wisconsin Package much easier by providing a common editing interface from which most programs can be launched and alignments manipulated.

http://ubio.bio.indiana.edu/soft/molbio/readseq

ReadSeq is public domain software distributed by the author, D.G. Gilbert, and is available from the Bioinformatics Group at the Biology Department of Indiana University, Bloomington.


Entrez is public domain software distributed by the authors and available from the National Center for Biotechnology Information (NCBI) at the National Library of Medicine, National Institutes of Health (NIH), Bethesda, Maryland.


Online Mendelian Inheritance in Man (OMIM). Available from the Center for Medical Genetics, Johns Hopkins University, Baltimore, Maryland, and the NCBI at the National Library of Medicine, NIH, Bethesda, Maryland. Also see Pearson et al. (1994).

http://www.sinauer.com

Phylogenetic Analysis Using Parsimony (PAUP*) was developed by D.L. Swofford (copyright, 1989–2002). The official homepage is located at Florida State University (see below). A 4.0 beta version is available at the time of this writing, and is distributed by Sinauer Associates.

http://paup.csit.fsu.edu

The official homepage for PAUP*, located at Florida State University, Tallahassee.

Contributed by Steven M. Thompson
Florida State University
Tallahassee, Florida
This appendix contains the following figures: Figure 3.6.1, Figure 3.6.2, Figure 3.6.3, Figure 3.6.4, Figure 3.6.5, Figure 3.6.6, Figure 3.6.7, Figure 3.6.8, Figure 3.6.9, Figure 3.6.10, Figure 3.6.11, Figure 3.6.12, Figure 3.6.13, Figure 3.6.14, Figure 3.6.15, Figure 3.6.16, Figure 3.6.17, Figure 3.6.18, Figure 3.6.19, Figure 3.6.20, Figure 3.6.21, Figure 3.6.22, Figure 3.6.23, Figure 3.6.24, Figure 3.6.25, Figure 3.6.26, Figure 3.6.27, and Figure 3.6.28.

Figure 3.6.1 The SeqLab Editor window with a LookUp dataset loaded and ready to analyze.
Pile-Up of: /users1/thompson/.seqlab-mendel/pileup_28.list
Symbol comparison table: /usr/gcg/gcgcore/data/moredata/blosum30.cmp CompCheck: 8599
GapWeight: 15
GapLengthWeight: 5
pileup_28.msf MSF: 472 Type: P May 14, 2001 14:35 Check: 2476 ..

Name: ef1a_giala Len: 472 Check: 8631 Weight: 1.00
Name: q25166 Len: 472 Check: 6209 Weight: 1.00
Name: q25073 Len: 472 Check: 2914 Weight: 1.00
Name: o36039 Len: 472 Check: 7560 Weight: 1.00
Name: o96981 Len: 472 Check: 3858 Weight: 1.00
Name: o96980 Len: 472 Check: 3082 Weight: 1.00
Name: o44031 Len: 472 Check: 851 Weight: 1.00
Name: ef1a_crypv Len: 472 Check: 2406 Weight: 1.00
Name: o77447 Len: 472 Check: 9210 Weight: 1.00
Name: o77478 Len: 472 Check: 1123 Weight: 1.00
Name: ef1a_plafk Len: 472 Check: 1436 Weight: 1.00

Name: o96981 Len: 472 Check: 2476 Weight: 1.00
Name: eucic_porpu Len: 472 Check: 6199 Weight: 1.00
Name: o46335 Len: 472 Check: 7668 Weight: 1.00
Name: o97018 Len: 472 Check: 5669 Weight: 1.00
Name: o97109 Len: 472 Check: 6457 Weight: 1.00

//            1                                                   50
ef1a_giala  ~~~~~~~~~~ ~~~~~~~~~~ ~~~STLTGHL IYKCGGIDQR TIDEYEKRAT
q25166  ~~~~~~~~~~ ~~~~~~~~~~ NGKSTLTGHL IYKCGGIDQR TLDEYEKRAN
q25073  ~~~~~~~~~~ ~~~~~~~~~~ NGKSTLTGHL IYKCGGIDQR TLDEYEKKAN
o36039  ~~~~~~~~~~ ~~~~~~~~~~ NGKSTLTGHL IYKCGGIDQR TLDEYEKKAN
o96981  ~~~~~~~~~~ ~~~~~~~~~~ NGKSTLTGHL IYKCGGIDQR TLDEYEKKAN
o96980  ~~~~~~~~~~ ~~~~~~~~~~ NGKSTLTGHL IYKCGGIDQR TLDEYEKKAN
o44031  ~~~MGKEKTH INLVVIGHVD SGKSTTTGHL IYKLGGIDKR TIEKFEKESS
ef1a_crypv  ~~~MGKEKTH INLVVIGHVD SGKSTTTGHL IYKLGGIDKR TIEKFEKESS
ef1a_plafk  ~~~MGKEKTH INLVVIGHVD SGKSTTTGHL IYKLGGIDKR TIEKFEKESS
o77447  ~~~MGKEKTH INLVVIGHVD SGKSTTTGHL IYKLGGIDKR TIEKFEKESS
ef1a_plafk  ~~~MGKEKTH INLVVIGHVD SGKSTTTGHL IYKLGGIDKR TIEKFEKESS
o77478  ~~~MGKEKTH INLVVIGHVD SGKSTTTGHL IYKLGGIDKR TIEKFEKESS
o97109 Len: 472 Check: 6457 Weight: 1.00

Figure 3.6.2 (continues on next page) An abridged GCG PileUp output MSF file. The format holds the file name, type, date, and checksum, as well as sequence names, checksums, lengths, and weights, and the aligned sequence data in an interleaved fashion.
Figure 3.6.2  (continued)
Figure 3.6.3  PileUp's similarity dendrogram. The PileUp program automatically plots a cluster dendrogram of the similarities between the sequences of the dataset. The lengths of the vertical lines are proportional to those similarities. This is not an evolutionary tree and should never be presented as one.

Figure 3.6.4  The PileUp alignment of elongation factor, loaded into the SeqLab Editor, displayed using Residue Coloring.
Figure 3.6.5 SeqLab can use “cartoons” to graphically display the feature annotation contained in sequence database entries and produced by programs such as Motifs (see Basic Protocol 2). SeqLab merges this annotation with existing datasets using the Add to Editor and Overwrite Old with New function. It also allows the user to zoom in or out on a dataset to see its entire length. This figure shows the PileUp aligned dataset visualized with SeqLab’s Graphic Features annotation and a 4:1 zoom ratio. Aligned annotation now includes original database Feature Table sites, plus output from the program Motifs, and from the program pair MEME/MotifSearch.
Figure 3.6.6  The Wisconsin Package SeqLab LookUp window. LookUp is an SRS derivative that allows for the construction of complex, text-based sequence-database queries. It produces GCG list file format output.
Figure 3.6.7 Abridged screen trace of GCG’s LookUp output file. Notice the “list file” format that can be read by Wisconsin Package interfaces and programs, such as SeqLab and PileUp.
!!SEQUENCE_LIST 1.0

(Peptide) FASTA of: input_25.rsf{ef1a_giala} from: 1 to: 396 May 14, 2001 
12:43

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

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

ID   EF1A_GIALA     STANDARD;      PRT;   396 AA.

TO: @/users1/thompson/.seqlab-mendel/fasta_ssl_25.list Sequences:     23,370
Symbols:     4,555,867  Word Size: 2

Databases searched:
   SWISS-PROT, Release 39.0, Released on 15Jun2000, Formatted on 18Sep2000
   SPTREMBL, Release 14.0, Released on 15Jun2000, Formatted on 20Sep2000
   NRL_3D, Release 27.0, Released on 30Mar2000, Formatted on 2Oct2000

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

Histogram Key:
Each histogram symbol represents 39 search set sequences
Each inset symbol represents 3 search set sequences
z-scores computed from opt scores

<table>
<thead>
<tr>
<th>z-score obs</th>
<th>exp</th>
<th>(=)</th>
<th>(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 20</td>
<td>745</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>5:*</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>6</td>
<td>31:*</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>92</td>
<td>120:==*=</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>371</td>
<td>325:=====*</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>571</td>
<td>668:=============*</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>1711</td>
<td>1104:==================================</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>1537</td>
<td>1540:==================================*</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>1997</td>
<td>1883:==================================*===</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>1775</td>
<td>2077:====================================*</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>1782</td>
<td>2116:====================================*</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>2290</td>
<td>2025:====================================*===''</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.6.8 (continues on next three pages) An abridged output list file from GCG’s implementation of FASTA. A histogram of score distributions is plotted preceding the list portion of the file where hits are ranked statistically by E-value. Normally a pairwise alignment section would follow the list, but that was turned off in this run with the -NoAlign option.

Constructing and Refining Multiple Sequence Alignments

3.6.46
Joining threshold: 37, opt. threshold: 25, opt. width: 16, reg.-scaled

The best scores are:

| SWISSPROT:EF1A_GIALA Begin: 1 End: 396 |
| : Q08046 giardia lamblia (giardia int... 2696 2696 2696 3151.0 2.4e-169 |
| : SP_INVERTEBRATE:Q25166 Begin: 4 End: 399 |
| : Q25166 diplomonad atcc50330. elonga... 2318 2318 2318 2709.5 9.3e-145 |
| : SP_INVERTEBRATE:Q25073 Begin: 4 End: 399 |
| : Q25073 hexamita inflata. elongation... 2125 2125 2125 2484.1 3.3e-132 |

Figure 3.6.8 (continued)
### Figure 3.6.8 (continued)

Constructing and Refining Multiple Sequence Alignments

3.6.48
Figure 3.6.8 (continued)
Figure 3.6.9 The SeqLab editor loaded with sorted FASTA output. FASTA can be used as a tool to sort a list into ranked order based on similarity to a particular query. All or any desired portion of this output can then be loaded into the SeqLab editor for further analysis.

Figure 3.6.10 GCG PlotSimilarity draws a graph of the running similarity along the length of a multiple sequence alignment using a sliding window averaging approach. Peaks are conserved regions, while valleys are dissimilar areas. The ordinate scale comes from the similarity matrix used (by default the BLOSUM62 table).
Figure 3.6.11  PlotSimilarity can produce a color mask that can be superimposed over an open alignment in the editor. Dark regions now correspond to conserved peaks, whereas valleys are represented by white areas.

Figure 3.6.12  SeqLab Consensus display of a region near the carboxy termini of the author’s EF-1α example using the BLOSUM30 matrix, 33% required for majority (plurality), and a cutoff value of 4 for the minimum score that represents a match (threshold).
Figure 3.6.13  SeqLab Consensus mask display of the carboxy terminal region of the author’s EF-1α example using a weight mask generated from the BLOSUM30 matrix, a plurality of 15%, and a threshold of 4.
#NEXUS

[! Aligned sequences from GCG file(s) '@/users1/thompson/.seqlab-mendel/paupsearch_51.list' ]

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Len</th>
<th>Check</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIARDIA_L</td>
<td>441</td>
<td>2966</td>
<td>1.00</td>
</tr>
<tr>
<td>DIPLOMONAD_SP</td>
<td>441</td>
<td>4608</td>
<td>1.00</td>
</tr>
<tr>
<td>HEXAMITA_I</td>
<td>441</td>
<td>9530</td>
<td>1.00</td>
</tr>
<tr>
<td>SPIRONUCLEUS_V</td>
<td>441</td>
<td>2245</td>
<td>1.00</td>
</tr>
<tr>
<td>SPATHIDIUM_SP</td>
<td>441</td>
<td>2937</td>
<td>1.00</td>
</tr>
<tr>
<td>CRYPTOSPORIDIUM_P</td>
<td>441</td>
<td>7665</td>
<td>1.00</td>
</tr>
<tr>
<td>PLASMODIUM_K</td>
<td>441</td>
<td>9956</td>
<td>1.00</td>
</tr>
<tr>
<td>PLASMODIUM_B</td>
<td>441</td>
<td>9937</td>
<td>1.00</td>
</tr>
<tr>
<td>PLASMODIUM_P</td>
<td>441</td>
<td>796</td>
<td>1.00</td>
</tr>
<tr>
<td>EUPLOTES_A</td>
<td>441</td>
<td>8831</td>
<td>1.00</td>
</tr>
<tr>
<td>EUPLOTES_C</td>
<td>441</td>
<td>8653</td>
<td>1.00</td>
</tr>
<tr>
<td>BLASTOCYSTIS_H</td>
<td>441</td>
<td>9014</td>
<td>1.00</td>
</tr>
<tr>
<td>STENTOR_C</td>
<td>441</td>
<td>5386</td>
<td>1.00</td>
</tr>
<tr>
<td>BLEPHARISMA_J</td>
<td>441</td>
<td>7915</td>
<td>1.00</td>
</tr>
<tr>
<td>ENTAMOEBA_H</td>
<td>441</td>
<td>8365</td>
<td>1.00</td>
</tr>
<tr>
<td>OXYMONADIDA_SP</td>
<td>441</td>
<td>8531</td>
<td>1.00</td>
</tr>
<tr>
<td>DINENYMPHA_E</td>
<td>441</td>
<td>5471</td>
<td>1.00</td>
</tr>
<tr>
<td>TRYPANOSOMA_C</td>
<td>441</td>
<td>9945</td>
<td>1.00</td>
</tr>
<tr>
<td>TRYPANOSOMA_B</td>
<td>441</td>
<td>960</td>
<td>1.00</td>
</tr>
<tr>
<td>KENTROPHOROS_SP</td>
<td>441</td>
<td>1567</td>
<td>1.00</td>
</tr>
<tr>
<td>EUGLENA_G</td>
<td>441</td>
<td>492</td>
<td>1.00</td>
</tr>
<tr>
<td>PLANO PROTOSTELIUM_A</td>
<td>441</td>
<td>8843</td>
<td>1.00</td>
</tr>
<tr>
<td>DICTYOSTELIUM_D</td>
<td>441</td>
<td>6233</td>
<td>1.00</td>
</tr>
<tr>
<td>PHYSARUM_P</td>
<td>441</td>
<td>320</td>
<td>1.00</td>
</tr>
<tr>
<td>CYANOPHORA_P</td>
<td>441</td>
<td>4176</td>
<td>1.00</td>
</tr>
<tr>
<td>PHYTOPHTHORA_I</td>
<td>441</td>
<td>804</td>
<td>1.00</td>
</tr>
<tr>
<td>STYLO NYCHIA_M</td>
<td>441</td>
<td>2825</td>
<td>1.00</td>
</tr>
<tr>
<td>STYLO NYCHIA_L</td>
<td>441</td>
<td>1254</td>
<td>1.00</td>
</tr>
<tr>
<td>PARANOPHYRS_C</td>
<td>441</td>
<td>196</td>
<td>1.00</td>
</tr>
<tr>
<td>TET RHYMENA_P</td>
<td>441</td>
<td>4061</td>
<td>1.00</td>
</tr>
<tr>
<td>TELOTROCHIDIIUM_H</td>
<td>441</td>
<td>1239</td>
<td>1.00</td>
</tr>
<tr>
<td>PARAMECIUM_T</td>
<td>441</td>
<td>3452</td>
<td>1.00</td>
</tr>
<tr>
<td>COLPODA_I</td>
<td>441</td>
<td>8135</td>
<td>1.00</td>
</tr>
<tr>
<td>NAXELLA_SP</td>
<td>441</td>
<td>6970</td>
<td>1.00</td>
</tr>
<tr>
<td>PORPHYRA_P</td>
<td>441</td>
<td>1559</td>
<td>1.00</td>
</tr>
<tr>
<td>TRICHOMONAS_T</td>
<td>441</td>
<td>6212</td>
<td>1.00</td>
</tr>
<tr>
<td>TRICHOMONAS_V</td>
<td>441</td>
<td>6532</td>
<td>1.00</td>
</tr>
<tr>
<td>NAEGLERIA_A</td>
<td>441</td>
<td>7736</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Figure 3.6.14  (continues on next two pages)  The PAUPSearch program can reliably and quickly extract NEXUS format from GCG multiple sequence alignments using the -NoRun option. Zero mask weighted columns are excluded from the file.
begin data;
dimensions ntax=38 nchar=441;
format datatype=protein interleave gap=.;
matrix

[ 1 50 ]

GIARDIA_L .......... .......... STLTGHLIYK CGGIDQRTID EYEKRATEMG
DIPLOMONAD_SP .......... .......... NGK STLTGHLIYK CGGIDQRTLD EYEKRENEMG
HEXAMITA_I .......... .......... NGK STLTGHLIYK CGGIDQRTLE DYEEKANE1G
SPIRONUCLEUS_V .......... .......... NGK STLTGHLIFK CGGIDQRTLD EYEKANELG
SPATHIDIUM_SP .......... .......... VDSGK STTSTGHLIYK CGGIDERTIE KFEKKEQ1G
CRYPTOSPORIDIUM_P MGKEDTHINL VVIGHVDSGK STTTGHLIYK LGGIDKRTIE KFEKESSEM1G
PLASMODIUM_K MGKEDTHINL VVIGHVDSGK STTTGHLIYK LGGIDKRTIE KFEKESAM1G
PLASMODIUM_B MGKEDTHINL VVIGHVDSGK STTTGHLIYK LGGIDKRTIE KFEKESAM1G
PLASMODIUM_F MGKEDTHINL VVIGHVDSGK STTTGHLIYK LGGIDKRTIE KFEKESAM1G
EUPLOTES_A .......... .......... VDSGK STTSTGHLIYK LGGIDRTIE KFEKEAE1G
EUPLOTES_C MGKEKHEINL VVIGHVDSGK STTTGHLIYK LGGIDKRTIE KFEKESAM1G
BLASTOCYSTIS_H MGKEKPHINL VVIGHVDSGK STTTGHLIYK LGGIDQRTIE KFEKESAEM1G
STENTOR_C .......... .......... VDSGK STTSTGHLIYK LGGIDKRTIE KFEKESAM1G
BLEPHARISMA_J .......... .......... VDSGK STTSTGHLIYK LGGIDKRTIE KFEKESAM1G
ENTAMOEBA_H MGKEKTHINL VVIGHVDSGK STTTGHLIYK CGGIDQRTIE KFEKESAEM1G
OXMONADIDA_SP .......... .......... STTTRHLIYK CGGIDQRTL1D KFDKDAEMG
DINENYMPHA_E .......... .......... STTSTGHLIYK CGGIDERTIE KFEQESAM1G
TRYPANOSOMA_C MGKEKTMNL VVIGHVDSGK STTSTGHLIYK CGGIDQRTIE KFEKESAM1G
TRYPANOSOMA_B MGKEKVHNL VVIGHVDSGK STTSTGHLIYK CGGIDQRTIE KFEKESAM1G
KENTROPHOROS_SP .......... .......... VDSGK STTSTGHLIYK CGGIDKRTIE KFDKDAE1G
EUGLENA_G MGKEKPHINL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKESAM1G
PLANOPROTOSTELIUM_A .......... .......... AGK STTTGHLIYK CGGIDKRTIE KFEKESAM1G
DICTYOSTELIUM_D MEKEKTHINI VVIGHVDSGK STTTGHLIYK CGGIDKRIVIE KFEKESAM1G
PHYSARUM_P .......... .......... AGK STTTGHLIYK CGGIDKRTIE KFEKESAM1G
CYANOPHORA_P MGKQKTHINL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKESAM1G
PHYTOPHTHORA_I .......... .......... VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKESAM1G
STYLONYCHIA_M .......... .......... VDSGK STTSTGHLIYK CGGIDKRTIE KFEKESAM1G
STYLONYCHIA_L MGKEKMNHL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKESAM1G
PARANOPHYRE_C .......... .......... VDSGK STTSTGHLIYK CGGIDKRTIE KFEKESAM1G
TETRHYMENA_P M.GDKVHINL VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKESAM1G
TELOTROCHIDIUM_H .......... .......... GHDSTGK STTSTGHLIYK CCGIDQRTIE KFEKESAM1G
PARAMECIUM_T G.KDKHNVNL VVIGHVDSGK STTTGHLIYK LGGIDKRTIE KFEKESAM1G
COLPODA_I .......... .......... VDSGK STTSTGHLIYK CGGIDKRTIE KFEKESAM1G
NAXELLA_SP .......... .......... VDSGK STTSTGHLIYK CGGIDKRTIE KFEKESAM1G
PORPHYRA_P MGKEKQHVS1 VVIGHVDSGK STTTGHLIYK CGGIDKRTIE KFEKESAM1G
TRICHOMONAS_T .......... .......... STTSTGHLIYK CGGIDKRTIE KFEKESAM1G
TRICHOMONAS_V .......... .......... VDAQK STTSTGHLIYK CGGIDKRTIE KFEKESAM1G
NAEGLERIA_A .......... .......... AGK STTTGHLIYK CGGIDKRTIE KFEKESAM1G

//

Figure 3.6.14 (continued)
endblock;

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

Figure 3.6.14  (continued)
Figure 3.6.15  The GCG ToFastA program reliably converts GCG multiple sequence alignments into Pearson FASTA format (APPENDIX 1B). This conversion takes advantage of the mask sequence to exclude columns with zero weights and changes gap periods and tildes to hyphens.
> readseq
readSeq (1Feb93), multi-format molbio sequence reader.

Name of output file (?=help, defaults to display):

**EF1A.phy**

1. IG/Stanford
2. GenBank/GB
3. NBRF
4. EMBL
5. GCG
6. DNAStrider
7. Fitch
8. Pearson/Fasta
9. Zuker (in-only)
10. Olsen (in-only)
11. Phylip3.2
12. Phylip
13. Plain/Raw
14. PIR/CODATA
15. MSP
16. ASN.1
17. PAUP/NEXUS
18. Pretty (out-only)

Choose an output format (name or #): 12

Name an input sequence or -option:

**EF1A.tfa**

Sequences in EF1A.tfa (format is 8. Pearson/Fasta)

1) GIARDIA_L In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
2) DIPLOMONAD_SP In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
3) HEXAMITA_I In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
4) SPIRINUCLEUS_V In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
5) SPATHIDIUM_SP In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
6) CRYPTOSPORIDIUM_P In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
7) EUPLOTES_A In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
8) EUPLOTES_C In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
9) BLASTOCYSTIS_H In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list
10) KENTROPHOROS_SP In situ PileUp of: @/users1/thompson/.seqlab-mendel/pileup_36.list

Choose a sequence (# or All):

**all**

Name an input sequence or -option:

**rtn**

---

Figure 3.6.16  A ReadSeq sample screen trace with user responses highlighted in bold.
<table>
<thead>
<tr>
<th>Organism</th>
<th>Alignment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIARDIA_L</td>
<td>--------------------------</td>
<td>EYEKRATEMG</td>
</tr>
<tr>
<td>DIPLOMONAD</td>
<td>---NGK</td>
<td>EYEKRANEMG</td>
</tr>
<tr>
<td>HEXAMITA_I</td>
<td>---NGK</td>
<td>DYEKKANEIG</td>
</tr>
<tr>
<td>SPIRONUCLE</td>
<td>---NGK</td>
<td>EYEKANELG</td>
</tr>
<tr>
<td>SPATTHIDIUM</td>
<td>---VDSGK</td>
<td>KFEKEAKQIG</td>
</tr>
<tr>
<td>CRYPTOSPOR</td>
<td>MGKEKTHINL VVIGHVDSGK</td>
<td>KPEKESSMEG</td>
</tr>
<tr>
<td>PLASMODIUM</td>
<td>MGKEKTHINL VVIGHVDSGK</td>
<td>KPEKESESMEG</td>
</tr>
<tr>
<td>PLASMODIUM</td>
<td>MGKEKTHINL VVIGHVDSGK</td>
<td>KPEKESESMEG</td>
</tr>
<tr>
<td>PLASMODIUM</td>
<td>MGKEKTHINL VVIGHVDSGK</td>
<td>KPEKESESMEG</td>
</tr>
<tr>
<td>EUPLOTES_A</td>
<td>---VDSGK</td>
<td>KPEKESESMEG</td>
</tr>
<tr>
<td>EUPLOTES_C</td>
<td>MGKEKEHLNL VVIGHVAGK</td>
<td>KPEKEAEEIG</td>
</tr>
<tr>
<td>BLASTOCYST</td>
<td>MGKEKPHINL VVIGHVAGK</td>
<td>KPEKEAEEIG</td>
</tr>
<tr>
<td>STENTOR_C</td>
<td>---VDSGK</td>
<td>KPEKEAEEIG</td>
</tr>
<tr>
<td>BLEPHARISM</td>
<td>---VDSGK</td>
<td>KYEKEAKEMG</td>
</tr>
<tr>
<td>ENTAMOeba_</td>
<td>MPKEKTHINI VVIGHVDSGK</td>
<td>KPEKESESMEG</td>
</tr>
<tr>
<td>OXYMONADID</td>
<td>--------------------------</td>
<td>RFQKESEAMG</td>
</tr>
<tr>
<td>DINENYMPHA</td>
<td>--------------------------</td>
<td>KPEKESESAMG</td>
</tr>
<tr>
<td>TRYPSANOSOM</td>
<td>MGKEKTHINL VVIGHVAGK</td>
<td>KPEKEAEEIG</td>
</tr>
<tr>
<td>TRYPSANOSOM</td>
<td>MGKEKTHINL VVIGHVAGK</td>
<td>KPEKEAEEIG</td>
</tr>
<tr>
<td>KENTROPHOR</td>
<td>--------------------------</td>
<td>KPEKEAEEIG</td>
</tr>
<tr>
<td>EUGLENA_G</td>
<td>MGKEKTHINL VVIGHVAGK</td>
<td>KPEKEAEEIG</td>
</tr>
<tr>
<td>PLANOPROTO</td>
<td>--------------------------</td>
<td>KPEKEAEEIG</td>
</tr>
<tr>
<td>DICTYOSTEL</td>
<td>MEEKKTHINL VVIGHVAGK</td>
<td>KPEKEAEEIG</td>
</tr>
<tr>
<td>PHYSARUM_P</td>
<td>---AGK</td>
<td>KPEKEAEEIG</td>
</tr>
<tr>
<td>CYANOPHORA</td>
<td>MGKEKTHINL VVIGHVAGK</td>
<td>KPEKEAEEIG</td>
</tr>
<tr>
<td>PHYTOPHOTO</td>
<td>---VGHVAGK</td>
<td>KPEKEAEEIG</td>
</tr>
<tr>
<td>STYLOWYCHI</td>
<td>---VDSGK</td>
<td>KPEKEAEEIG</td>
</tr>
<tr>
<td>STYLOWYCHI</td>
<td>MPKEKHLNL VVIGHVAGK</td>
<td>KPEKEAEEIG</td>
</tr>
<tr>
<td>PARANOPHYR</td>
<td>---VDSGK</td>
<td>KPEKEAEEIUG</td>
</tr>
<tr>
<td>TETRHYMENA</td>
<td>M-GDKVHINL VVIGHVAGK</td>
<td>KPEKEAEEIUG</td>
</tr>
<tr>
<td>TELOTROCHI</td>
<td>---GDBSGK</td>
<td>KPEKEAEEIUG</td>
</tr>
<tr>
<td>PARAMECIUM</td>
<td>G-KDKHVINL VVIGHVAGK</td>
<td>KPEKEAEEIUG</td>
</tr>
<tr>
<td>COLPODA_I</td>
<td>---VDSGK</td>
<td>KPEKEAEEIUG</td>
</tr>
<tr>
<td>NAXELLA_SP</td>
<td>---VDSGK</td>
<td>KPEKEAEEIUG</td>
</tr>
<tr>
<td>PORPHYRA_P</td>
<td>MGKEKQHISI VVIGHVAGK</td>
<td>KPEKEAEEIUG</td>
</tr>
<tr>
<td>TRICHOSONA</td>
<td>--------------------------</td>
<td>KPEKEAEEIUG</td>
</tr>
<tr>
<td>TRICHOSONA</td>
<td>--------------------------</td>
<td>KPEKEAEEIUG</td>
</tr>
<tr>
<td>NAGGLERIA_</td>
<td>---AGK</td>
<td>KPEKEAEEIUG</td>
</tr>
</tbody>
</table>

Figure 3.6.17 (continues on next page) The beginning of the author’s sample dataset in PHYLIP format produced by ReadSeq from a FASTA format file (APPENDIX 1B). ToFastA stripped zero weight columns and changed gap periods and tildes to hyphens; the PHYLIP file reflects this.
<table>
<thead>
<tr>
<th>Sequence</th>
<th>Match</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>KGSFKYAWVL DKLKAERERG ITIDIALWKF ETPRYYFTVI DAPGHRDFIK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KGSFKYAWVL DKLKAERERG ITIDIALWKF ETPRYYFTVI DAPGHRDFIK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KGSFKYAWVL DKLKAERERG ITIDIALWKF ETNKYYFTII DAPGHRDFIK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KGSFKYAWVL DKLKAERERG ITIDIALWKF ETTNRFYTI DAPGHRDFIK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KGSFKYAWVL DKLKAERERG ITIDIALWKF ETRKDFFTII DAPGHRDFIK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KGSFKYAWVL DKLKAERERG ITIDIALWKF ETRKDFFTII DAPGHRDFIK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KGSFKYAWVL DKLKAERERG ITIDIALWKF ETENRHYTI DAPGHRDFIK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KGSFKYAWVL DKLKAERERG ITIDIALWKF ETENRHYTI DAPGHRDFIK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KGSFKYAWVL DKLKAERERG ITIDIALWKF ETSKYYFTII DAPGHRDFIK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KGSFKYAWVL DKLKAERERG ITIDIALWKF ETSKYYFTII DAPGHRDFIK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KGSFKYAWVL DKLKAERERG ITIDIALWKF ETGKYYFTII DAPGHRDFIK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KGSFKYAWVL DKLKAERERG ITIDIALWKF ETGKYYFTII DAPGHRDFIK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KGSFKYAWVL DKLKAERERG ITIDIALWKF ETSKYYFTII DAPGHRDFIK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Figure 3.6.17 (continued)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finding Similarities and Inferring Homologies</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Current Protocols in Bioinformatics

3.6.59
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-1alpha</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-1beta</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, SUP12 or GST1) and the human homolog as GST1-Hs.
- Prokaryotic peptide chain release factor 3 (RF-3) (gene prfC). RF-3 is a class-II RF, a GTP-binding protein that interacts with class I RFs (see <PDOC00607>) and enhance their activity [4].
- Prokaryotic GTP-binding protein lepA and its homolog in yeast (gene GUF1) and in Caenorhabditis elegans (ZK1236.1).
- Yeast HBS1 [5].
- Rat statin S1 [6], a protein of unknown function which is highly similar to EF-1alpha.

**Figure 3.6.18** (continues on next six pages) PROSITE patterns found by the GCG Motifs program in the example elongation factor dataset. Notice the extensive reference discussion for each PROSITE pattern found.
In EF-1-alpha, a specific region has been shown to be involved in a conformational change mediated by the hydrolysis of GTP to GDP. This region is conserved in both EF-1alpha/EF-Tu as well as EF-2/EF-G and thus seems typical for GTP-dependent proteins which bind non-initiator tRNAs to the ribosome. The pattern we developed for this family of proteins include that conserved region.

- Consensus pattern: D-[KRSTGANQFYW]-x(3)-E-[KRAQ]-x-[RKQD]-[GC]-[IVMK]-[ST]-[IV]-x(2)-[GSTACKRNQ]

- Sequences known to belong to this class detected by the pattern: ALL, except for 11 sequences.
- Other sequence(s) detected in SWISS-PROT: NONE.
- Last update: November 1997 / Text revised.

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

- ATP synthase alpha and beta subunits (see <PDOC00137>).
- Myosin heavy chains.
- Kinesin heavy chains and kinesin-like proteins (see <PDOC00343>).
- Dynamins and dynamin-like proteins (see <PDOC00362>).
- Guanylate kinase (see <PDOC00670>).
- Thymidine kinase (see <PDOC00524>).
- Thymidylylate kinase (see <PDOC001034>).
- 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 <PDOC00085>).
- 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, Go, 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...

Figure 3.6.18 (continued)
proteins the flexible loop exists in a slightly different form; this is the case for tubulins or protein kinases. A special mention must be reserved for adenylate kinase, in which there is a single deviation from the P-loop pattern: in the last position Gly is found instead of Ser or Thr.

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

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

-Last update: July 1999 / Text revised.

EMBO J. 1:945-951(1982).
[5] Saraste M., Sibald P.R., Wittinghofer A.
Gallagher M.P.
[8] Hodgman T.C.
[9] Linder P., Lasko P., Ashburner M., Gileadi U., Gill D.R.,
Gallagher M.P.

                      STTGGHLIYAC

24: VVAGK
GGIDK
* Prokaryotic membrane lipoprotein lipid attachment site *

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

- Major outer membrane lipoprotein (murein-lipoproteins) (gene lpp).
- Escherichia coli lipoprotein-28 (gene nlpA).
- Escherichia coli lipoprotein-34 (gene nlpB).
- Escherichia coli lipoprotein nlpC.
- Escherichia coli lipoprotein nlpD.
- Escherichia coli osmotically inducible lipoprotein B (gene osmB).
- Escherichia coli osmotically inducible lipoprotein E (gene osmE).
- Escherichia coli peptidoglycan-associated lipoprotein (gene pal).
- Escherichia coli rare lipoproteins A and B (genes rplA and rplB).
- Escherichia coli copper homeostasis protein cutF (or nlpE).
- Escherichia coli plasmids traT proteins.
- Escherichia coli Col plasmids lysis proteins.
- A number of Bacillus beta-lactamases.
- Bacillus subtilis periplasmic oligopeptide-binding protein (gene oppA).
- Borrelia burgdorferi outer surface proteins A and B (genes ospA and ospB).
- Borrelia hermsii variable major protein 21 (gene vmp21) and 7 (gene vmp7).
- Chlamydia trachomatis outer membrane protein 3 (gene omp3).
- Fibrobacter succinogenes endoglucanase cel-3.
- Haemophilus influenzae proteins Pal and Pcp.
- Klebsiella pullulunase (gene pulA).
- Klebsiella pullulunase secretion protein pulS.
- Mycoplasma hyorhinis protein p37.
- Mycoplasma hyorhinis variant surface antigens A, B, and C (genes vlpABC).
- Neisseria outer membrane protein H.8.
- Pseudomonas aeruginosa lipopeptide (gene lppL).
- Pseudomonas solanacearum endoglucanase egI.
- Rhodopseudomonas viridis reaction center cytochrome subunit (gene cytC).
- Rickettsia 17 Kd antigen.
- Shigella flexneri invasion plasmid proteins mxiJ and mxiM.
- Streptococcus pneumoniae oligopeptide transport protein A (gene amiA).
- Treponema pallidium 34 Kd antigen.
- Treponema pallidium membrane protein A (gene tmpA).
- Vibrio harveyi chitobiase (gene chb).
- Yersinia virulence plasmid protein yscJ.
- Halocyanin from Natrobacterium pharaonis [4], a membrane associated copper-binding protein. This is the first archaeabacterial protein known to be modified in such a fashion.

Figure 3.6.18 (continued)
From the precursor sequences of all these proteins, we derived a consensus pattern and a set of rules to identify this type of post-translational modification.

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

[1] Hayashi S., Wu H.C.
[3] von Heijne G.

It has been shown [1] that four different type of carbohydrate kinases seem to be evolutionary related. These enzymes are:
- L-fucolokinase (EC 2.7.1.51) (gene fucK).
- Gluconokinase (EC 2.7.1.12) (gene gntK).
- Glycerokinase (EC 2.7.1.30) (gene glpK).
- Xylulokinase (EC 2.7.1.17) (gene xylB).
- L-xylulose kinase (EC 2.7.1.53) (gene lyxK).
These enzymes are proteins of from 480 to 520 amino acid residues.
As consensus patterns for this family of kinases we selected two conserved regions, one in the central section, the other in the C-terminal section.

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

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

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

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


Figure 3.6.18 (continued)
Figure 3.6.19  Motifs can create an RSF file with the location of PROSITE patterns annotated by color and shape. The display now shows annotation from the database, from Motifs, and from MEME/MotifSearch, using Features Coloring.

Figure 3.6.20  The unaligned dataset shown using Graphic Features and a 4:1 zoom ratio. Annotations now include the original database Feature Table entries as well as conserved elements discovered by MEME/MotifSearch.
(Peptide) PROFILESEARCH of: /users1/thompson/seqlab/primitive.prf Length: 428 to pir:

Scores are not corrected for composition effects

- Gap Weight: 60.00
- Gap Length Weight: 0.67
- Sequences Examined: 188196
- CPU time (seconds): 2713

Profile information:
(Peptide) PROFILEMAKE v4.50 of:
@/users1/thompson/.seqlab-mendel/profilemake_63.list Length: 428
Sequences: 38 MaxScore: 1798.78 July 11, 2001 20:11
- Gap: 1.00 Len: 1.00
- GapRatio: 0.33 LenRatio: 0.10

Normalization: July 11, 2001 21:21

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

Normalization equation:

- Calc_Score = 66.96 * (1.0 - exp(-0.0023*SeqLen - 0.6191))
- Correlation for curve fit: 0.973

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

- Z_Score = (Score/Calc_Score - 1.010) / 0.164

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Strd</th>
<th>ZScore</th>
<th>Orig Length</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIR2:A49171</td>
<td>+</td>
<td>158.30 1454.17</td>
<td>435</td>
<td>translation elongation factor e</td>
</tr>
<tr>
<td>PIR2:A54760</td>
<td>+</td>
<td>157.48 1458.18</td>
<td>449</td>
<td>translation elongation factor e</td>
</tr>
<tr>
<td>PIR2:S11665</td>
<td>+</td>
<td>156.90 1458.53</td>
<td>456</td>
<td>translation elongation factor e</td>
</tr>
<tr>
<td>PIR2:S16308</td>
<td>+</td>
<td>156.81 1449.85</td>
<td>446</td>
<td>translation elongation factor e</td>
</tr>
<tr>
<td>PIR2:JCS5117</td>
<td>+</td>
<td>155.73 1442.59</td>
<td>449</td>
<td>translation elongation factor e</td>
</tr>
</tbody>
</table>

Figure 3.6.21 (continues on next page) A greatly abridged GCG ProfileSearch output list file. Most of the known elongation factors have been edited from the file, although several distant homologues are left intact.
**Finding Similarities and Inferring Homologies**

---

Figure 3.6.21 (continued)

---

Current Protocols in Bioinformatics
Figure 3.6.22  The t-RNA binding region of a ProfileSegments -MSF -Global alignment of selected near and distant EF-1α homologues aligned against the author's example EF-1α profile.

Figure 3.6.23  EF-1α primitive dataset aligned to the *Thermus aquaticus* EF-Tu sequence by HmmerAlign. Inferred α-helices based on the *Thermus* structure are displayed by Features Coloring in (medium grey here). Text annotation lines have also been added to the display where the location of the helices is noted.
Figure 3.6.24  Screen snapshot of the author's sample alignment showing the same region as Figure 3.6.23, but now including additional HmmerPfam annotation and displayed with Graphic Features. Inferred $\alpha$-helices are now seen as transparent red coils (seen here as open box zigzags).

Figure 3.6.25  A multiple sequence alignment and simple motif consensus of elongation factor Tu/1$\alpha$ from several different organisms illustrates the conservation of the first of several GTP-binding domains, that region around position twenty here, the P-Loop.
| Cons | A | B | C | D | E | F | G | H | I | K | L | M | N | P | Q | R | S | T | V | Y | Z | Gap | Len |
| E    | 11| 20| -11| 27 | 33| -21| 10 | -9 | 16 | 18 | 0  | 8  | 17 | -3 | -29| -15| 26 | 12 | 12 |
| K    | 0 | 27| -40| 21 | 22| -47| -6 | 7  | -11| 100| 20 | 13 | 27 | 7  | 27 | 53 | 14 | 13 | -3 | -13| 5  | -40| 28 | 12 |

**Figure 3.6.26** A traditional Gribskov-style profile of the elongation factor 1α/Tu P-Loop region. The horizontal axis contains all possible amino acid residues and the two gap penalties. The vertical axis lists the consensus positions along the profile. Noted positions from the text are highlighted with gray boxes.

**Figure 3.6.27** SeqLab can be used to align DNA sequences against an already aligned dataset of its translational products. This is sometimes very helpful, especially when phylogenetic inference is the eventual goal.

**Constructing and Refining Multiple Sequence Alignments**

3.672
Figure 3.6.28  Protein Data Bank 1EFT. The *Thermus aquaticus* elongation factor Tu structure in its GTP conformation (Kjeldgaard et al., 1993). Structural visualization by RasMol (Sayle and Milner-White, 1995).
Multiple sequence alignment (MSA) is the central technique for inferring biological information from a set of sequences. It involves the alignment of more than two sequences and aims to find equivalent positions across the aligned query set of sequences. An MSA can be viewed as a 2-dimensional table in which the sequences are the rows, and where the columns of equivalent amino acids have been arranged by placing gap characters in appropriate positions, such that the biological relationship of the sequences is best represented. An MSA can provide a wealth of information about structure-function relationships within a set of protein sequences, e.g., the evolutionary conservation of functionally or structurally important amino acids at certain sequence positions or conserved hydrophobicity patterns in particular regions. It is also often useful as a starting point for site-directed mutagenesis experiments. In addition to being an important means for gleaning the above biological clues by visual inspection, MSAs are an essential prerequisite to many computational modes of analysis of protein families, such as homology modeling, secondary structure prediction, and phylogenetic reconstruction. They may further be used to derive profiles (Gribskov et al., 1987) or hidden Markov models (Haussler et al., 1993; Bucher et al., 1996) that can be used to scour databases for distantly related members of the family.

An MSA is an attempt to represent evolutionarily related sequences in the most consistent way. Finding the maximal alignment score according to a given evolutionary model is equivalent to maximizing the probability that the sequences evolved as given by the MSA. Despite the considerable history of MSA (Needleman and Wunsch, 1970; Smith and Waterman, 1981; Hogeweg and Hesper, 1984; Barton and Sternberg, 1987; Corpet, 1988; Higgins and Sharp, 1988; Taylor, 1988; Altschul et al., 1989; Gotoh, 1993, 1996; Thompson et al., 1994a,b; Heringa, 1999, 2002; Notredame et al., 2000; Lee et al., 2002), the methodology is still under continuous development.

However, because a complicated relationship often exists between homologous sequences, combined with a lack of information about their true evolutionary history, absolute certainty about the correctness of MSAs is often hard to achieve. Therefore, it is instructive to bear in mind that a computational biologist is dealing with “truth” or “correctness” at two levels: the first level being the biological reality (current and past), and the second the investigator’s model of this reality in terms of scoring schemes, graphs, or alignments. Thus, it is appropriate to quote: “All models are wrong but some are useful” (George E.P. Box). The fundamental model for all types of sequence comparisons is the (generalized) model of evolution. Such models have been derived from trusted sequence alignments by assuming a Markov model of evolution (Dayhoff et al., 1978; Müller and Vingron, 2000) or by empirical derivation (Benner et al., 1993; Henikoff and Henikoff, 1994). The information about sequence evolution derived from such models is usually stored in amino acid substitution matrices. In the following sections, the focus is on higher-level aspects of modeling, concerning the complex relationships within sequence families, such as tree construction and progressive alignment strategies. These strategies are discussed along with practical considerations about the construction of meaningful MSAs.

With the completion of the first draft of the human genome and well over one hundred genomes of other species, the accurate alignment of biological sequences has become more important than ever. This is due to the fact that the direct prediction of a protein’s structure and function is still a major unsolved problem. To increase the knowledge of the function and interaction of protein sequences obtained by sequencing techniques, many initiatives are underway for large-scale proteomics and structure elucidation of novel genomic proteins. However, roughly 50% of the proteins in most sequenced species do not have an assigned function, and consequently an important target of bioinformatics method development is aimed at gathering the function of an increased fraction of translated proteins by enhancing comparative sequence techniques and threading protocols. In the quest for knowledge about the role of a certain unknown protein in the cellular molecular network, comparing the query sequence with the many sequences in annotated protein sequence databases often leads to useful suggestions regarding the protein’s 3-dimensional (3-D) structure or molecular function. These suggestions are obtained by extrapolating the properties of sequences, residing in annotated public databases, which are identi-
An Overview of Multiple Sequence Alignment

3.7.2

An Overview of Supplement 3 Current Protocols in Bioinformatics

Alignment

Sequence

Multiple

3.7.2

the sequences turn out to be highly similar. Problems; the only exceptions are those cases where the biological complexity of most MSA probably including different strategies, to approach individual query set of sequences. The user should therefor attempt a number of methods, preferably including different strategies, to approach the biological complexity of most MSA problems; the only exceptions are those cases where the sequences turn out to be highly similar.

Global or Local Methods

Many MSA techniques perform global alignment (Needleman and Wunsch, 1970) and match sequences over their full lengths. Problems with this approach can arise when sequences that are only homologous over local regions are compared. In such cases, global alignment techniques might fail to recognize highly similar internal regions because these may be overshadowed by dissimilar stretches and high gap penalties normally required to achieve proper global matching. Moreover, many biological sequences are modular and show shuffled domains (Heringa and Taylor, 1997), which can render a global alignment of two complete sequences meaningless. The occurrence of varying numbers of internal sequence repeats (Heringa, 1998) can also severely limit the applicability of global methods. In general, when there is a large difference in the lengths of two sequences to be compared, it is advisable to include local alignment techniques in the analysis. To address these problems, Smith and Waterman (1981) early on developed a so-called local alignment technique in which the most similar regions in two sequences are selected and aligned. The algorithm has been extended in various techniques to compute a list of top-scoring pairwise local alignments (Waterman and Eggert, 1987; Huang et al., 1990; Huang and Miller, 1991). Alignments produced by the latter techniques are nonintersecting, i.e., they have no matched pair of amino acids in common. For multiple sequences, the main automatic methods include the Gibbs sampler (Lawrence et al., 1993), MEME (Bailey and Elkan, 1994) and Dialign2 (Morgenstern, 1999). These local MSA programs often perform well when there is a clear block of ungap alignment shared by all of the sequences, but perform poorly under moderate gap requirements and show inferior results over general sets of test cases when compared with global methods (Thompson et al., 1999b; Notredame et al., 2000).

Performing MSA

Carrying out an MSA of a given protein sequence set and extracting maximum information from the alignment involves a number of critical steps:

1. The selection of sequences.
2. The choice of the scoring function used to compare sequences or sequence blocks.
3. The application and optimization of this scoring function in compiling the alignment.

Selecting the Sequences

An MSA can be misleading when a sequence set contains sequences that are not homologous. Ideally, the sequences should all be orthologs, but in practice it is often difficult to ensure that this is the case. It should be stressed that most MSA routines will produce an alignment even in the case of biologically unrelated sequences, which can give rise to spurious suggestions regarding the proteins’ structure or function (“garbage in, garbage out”). A widely used way to create a sequence set around a given query sequence of interest is to employ a homology searching technique (e.g., Altschul et al., 1997; Taylor, 1988; Karplus et al., 1998; Eddy, 1998; Karplus and Hu, 2001; http://hmmer.wustl.edu) to scour sequences in public sequence databases. Although the development of P- and E-values to estimate the statistical significance of putative homologs
found by these programs limits the chance of false positives, it is entirely possible that essentially nonhomologous sequences will enter the alignment set, which might confuse the alignment method used.

The Scoring Function

The scoring function is the formalization of the biological knowledge used in aligning the sequences. Ideally, it should contain all available knowledge about evolutionary, structural, and functional aspects of the compared sequences, so that the scoring function approximates the biological reality. In practice, however, this information is often not available or cannot be formalized mathematically. Although each cross-comparison of a residue between two sequences should in reality be evaluated individually based on its structural and functional context, the most widely used scheme to compare sequences is based on generalized averages for scoring each pair of residue types, given in the form of a symmetric 20×20 amino acid exchange matrix. The scheme models the alignment of two sequences as a Markov process, where the amino acid matches are considered independent, so that the product of the probabilities for each match within an alignment can be taken. Since many of the generally applied 20×20 scoring matrices contain propensities converted to logarithmic values (log-odds), the alignment score can normally be calculated by summing the log-odds values corresponding to matched residues minus appropriate gap penalties:

\[ S_{a,b} = \sum_l s(a_l, b_j) - \sum_k N_k \cdot gp(k) \]

where the first summation is over the exchange values associated with \( l \) matched residues and the second over each group of gaps of length \( k \), with \( N_k \) being the number of gaps of length \( k \) and \( gp(k) \) the associated gap penalty. In case affine gap penalties are used, \( gp(k) = pi + k \cdot pe \), where \( pi \) and \( pe \) are the penalties for gap initialization and extension, respectively. A consequence of the widely used affine gap penalty scheme is that long gaps that are required, for example, to span a domain \( B \) in aligning a two-domain sequence \( AC \) (where \( A \) and \( C \) represent domains) with a three-domain sequence \( ABC \), are often too costly, so that such sequences become misaligned. A complication with gap penalties is that there exists no formal model to set their values according to the evolutionary distances suggested by the exchange values within scoring matrices, so that one has to resort to empirical tuning of the gap penalties.

Applying the Scoring Function

Apart from being a fundamental biological challenge, MSA is also a computationally intense problem, which means that for all but the smallest data sets of less than 10 sequences, an exact solution is not feasible. Algorithms that perform simultaneous alignment over a multidimensional search matrix, where each sequence in the MSA represents an extra dimension (Stoye et al., 1998; Lipman et al., 1989), come closest to an exact solution.

The most populated class of algorithms is that of progressive MSA methods. The progressive strategy implies that an algorithm for pairwise sequence alignment is repeatedly used in a stepwise fashion until all sequences are aligned. In the vast majority of progressive methods the Dynamic Programming (DP) strategy is adopted. The DP strategy guarantees that, given an amino acid exchange matrix and gap penalty values, the highest scoring or optimal pairwise alignment is calculated. The progressive alignment strategy reuses the pairwise DP algorithm in a "greedy" manner, i.e., alignments formed during the progression towards the final MSA cannot be changed anymore. The main difference between the available DP-based methods is the way in which the information of aligned blocks of sequences is represented. While early methods used consensus sequences to represent alignment blocks, current methods all use a profile formalism to represent the information in an MSA (Gribskov et al., 1987). Recent developments in multiple alignment techniques have mainly focused on sensitive and optimal models to represent MSA information.

A class of techniques that are able to revisit and optimize is that of iterative multiple alignment techniques. Pioneered by Hogeweg and Hesper (1984), iterative techniques attempt to enhance the alignment quality by gleaning information from a multiple alignment constructed in an earlier round, which is then applied in a next round to improve the alignment according to a given scoring scheme. Another classes of alignments covered in this unit is stochastic alignments, where probabilistic frameworks such as hidden Markov models and Bayesian networks have been attempted. Other techniques based on fast computational techniques such as suffix trees and fast Fourier transforms (FFT) will be discussed below.
In the remainder of this unit, general methodological issues will be covered (see next section, MSA Methodology), after which an overview of current state-of-the-art methods will be presented. The overview also includes some older methods that, although no longer widely used, have been important for developments in the field. Finally, the last section offers some considerations on evaluating MSAs.

**MSA METHODOLOGY**

In this section the focus will be on higher-level aspects of modeling, concerning the complex relationships within sequence families, such as tree construction and progressive alignment strategies. These strategies are discussed along with practical considerations about the construction of meaningful MSAs.

**Progressive Alignment Strategies**

The most populated class of algorithms is that of progressive MSA methods. The progressive strategy implies that an algorithm for pairwise sequence alignment is repeatedly used in a stepwise fashion until all sequences are aligned. In the vast majority of progressive methods the Dynamic Programming (DP) strategy is adopted. The main difference between the DP-based methods is the way the information of aligned blocks of sequences is represented (see Profiles, below). To increase the chance of correct alignment, many methods calculate an appropriate order in which the sequences should be progressively aligned (Fig. 3.7.1). In many cases, this order is derived from all-against-all pairwise alignment of the sequences and the calculation of a dendrogram, often referred to as the “guide tree,” using the pairwise alignment scores (see discussion of Trees, below). The resulting branch order of the dendrogram is then followed to align the sequences, such that the most similar sequences are aligned first, and gradually the more distant sequences are included in the growing MSA. Although an efficient and stable strategy, the progressive alignment protocol suffers from its greediness and is not able to revise any of the alignments made earlier, such that any alignment errors made during the construction of the MSA cannot be repaired. This drawback is particularly significant for distant sequence sets because the alignments of sequences at early steps during progressive alignments cannot make use of information from other sequences, so that proper positional information required for correct matching is not available at early stages. Only later during the alignment progression does more information from other sequences (e.g., through profile representation) become employed in the alignment steps, but this happens quite possibly after misalignment has already taken place.

MSA programs like ClustalW (Thompson et al., 1994a,b; UNIT 2.3), T-Coffee (Notredame et al., 2000), and PRALINE (Heringa, 1999; Heringa, 2002) are based upon the progressive alignment strategy (Feng and Doolittle, 1987) and are all able to produce high-quality align-
ments. This was demonstrated in a recent benchmark (Heringa, 2002) with over 144 alignments in the BAliBASE repository (Thompson et al., 1999a), although the results produced by the various programs are not necessarily identical, particularly with more divergent sequence sets.

Trees

Reconstruction of the evolutionary history of proteins (Chapter 6) is one of the central aims of sequence comparison. An evolutionary model of a particular sequence family consists of an evolutionary tree depicting the sequence relationships within the family, and an MSA (Fig. 3.7.2), which shows the detailed local relation between the individual sequences. The following conventions and terminology are adopted: (a) trees consist of edges (lines) and nodes (crossings), where edge lengths define the distance between sequences and nodes represent the actual sequences; (b) trees are binary, with one incoming edge and two outgoing edges; (c) the ultimate ancestor is called the “root” and the terminal nodes are called “leaves.” Some tree construction algorithms (like parsimony; UNIT 6.4) give no indication about the position of the root, leading to “un-
rooted” trees. For the sake of completeness (and to correct some erroneous formulas in the standard literature), the equations for the number of possible unrooted and rooted trees for \( n \) sequences (Fig. 3.7.3) are presented here:

\[
\text{number of unrooted trees} = \frac{(2n-5)!}{2^{n-3}(n-3)!}
\]

\[
\text{number of rooted trees} = \frac{(2n-3)!}{2^{n-2}(n-2)!}
\]

Rooting of a tree can be achieved by adding a very distant member, also called an out-group, to the family, which defines the root as the point where its branch meets the tree.

The construction of “real” phylogenetic trees is a computer-intensive procedure that requires probabilistic modeling. A widely used approach is the application of maximum-likelihood methods (UNIT 6.6), which calculate the most probable phylogenetic tree associated with a given MSA and evolutionary model (Saitou, 1990). For the purpose of constructing a guide tree for an MSA (see above), more amenable ad hoc strategies are often adopted to reduce the computations. A frequently used approach is to estimate sequence distances from the pairwise alignment scores. Using these heuristic distances, a phylogenetic tree can be constructed to guide the construction of the MSA (see Progressive Alignment Strategies, above).

Regardless of whether the phylogenetic tree is calculated using maximum-likelihood (UNIT 6.6) or distance methods (UNIT 6.3), the significance of the branching of the tree can be estimated using the bootstrap (Felsenstein, 1985; UNIT 6.1). For bootstrapping, the columns in the multiple alignment are re-sampled a significant number of times (100 to 1000) with replacement, such that a single alignment column can occur multiple times in a resampled MSA, after which the significance for each branching in the original tree is taken from the frequency of the occurrence of the branch over all resampled trees (Fig. 3.7.4).

The transformation from pairwise alignments to a phylogenetic tree is performed by clustering algorithms, which fall into two conceptually different categories: distance-based (UPGMA, Neighbor-Joining; UNIT 6.3) and parsimony (UNIT 6.4). Distances can be obtained from sequence identities (Fitch and Margoliash, 1967) or pairwise sequence alignment scores (Hogeweg and Hesper, 1984).

Figure 3.7.4  The phylogenetic tree of the flavodoxin family. The numbers at the ancestral nodes are bootstrap values.
UPGMA (Unweighted Pair Group Method using arithmetic Averages; Sokal and Michener, 1958) joins sequences to clusters and progressively to larger clusters until a single cluster (tree) is accomplished. The order in which sequences or clusters are joined is simple: always the closest pair of sequence/sequence, sequence/cluster, or cluster/cluster is joined. Each joining operation creates an ancestral node, and the distance between the joined sequences (which are the leaves of the tree) is expressed as:

\[
\text{distance between joined sequences} = \frac{1}{n_i \cdot n_j} \sum d_{ij}
\]

where \(n_i, n_j\) are the number of sequences in the two joined clusters, and \(d_{ij}\) are the distances of all possible pairwise sequence combinations between the joined clusters. For example, if cluster “A” consists of sequences (1,2,3) and cluster “B” consists of sequences (4,5):

\[
\text{distance between joined sequences} = \frac{1}{3 \cdot 2} \left( d_{14} + d_{15} + d_{24} + d_{25} + d_{34} + d_{35} \right)
\]

The ancestral node is exactly in the middle between the joined sequences, so that the edge length between the joined sequences and their ancestral node is 1/2 of the above distance.

Neighbor-Joining (NJ; Saitou and Nei, 1987; UNIT 6.3) should be performed instead of UPGMA if the distances between sequences are not additive, which is equivalent to an unscaled evolutionary clock resulting from unequal evolutionary speeds in the various branches. The algorithm starts from a distance table containing all pairwise sequence distances by joining the closest pair of sequences and placing their ancestral node at half distance. The distance table is updated: the two columns/rows of the joined sequences are fused together and the average distance of the two joined sequences to every other sequence is computed. For example, for a sequence set A, B, C, D, after joining the closest sequence pair A and B to AB:

average distance of AB to C = \( \frac{1}{2} (d_{AC} + d_{BC}) \)

and

The closest pair, assuming (AB) and C, is joined to (ABC) and their ancestral node is placed at half distance. The procedure of average distance computation and joining is repeated until all sequences are included. The resulting tree is unrooted.

The Maximum Parsimony (MP; UNIT 6.4) method is an algorithm to find the tree that minimizes residue substitutions summed up over all sites of the whole tree (Eck and Dayhoff, 1966; Kluge and Farris, 1969). Therefore, a sufficient number of different tree topologies is generated in a first phase, and a cost for residue substitutions is assigned to each tree in a second phase. This cost function can be simply the total number of residue substitutions or the sum of weights \( W_{AB} \) for each substitution from residue A to residue B (weighted parsimony). The algorithm of Fitch (1971) is usually used for counting the number of residue changes. Sequences on ancestral nodes can be inferred if pointers between residues on ancestral and daughter nodes are stored. The number of possible tree topologies increases drastically with the number of sequences (see above), but stochastic approaches have been developed (Felsenstein, 1981).

Profiles

An MSA profile is a comprehensive representation of an MSA, stressing the composition of the alignment positions (columns) rather than the composition of the constituting sequences. In general form, a profile is a vector composed of 20 components (amino acids) at each MSA position. The vector components describe the contribution (score, weight, probability etc.) of each amino acid at this position to the MSA. Profiles are used to align an MSA to a single sequence or to another MSA by means of dynamic programming. Scoring of such multiple-to-single or multiple-to-multiple alignments is, in a theoretical sense, far from trivial, because the substitution matrices are strictly derived from probabilities of pairwise residue alignments.

However, classical pairwise substitution scores are commonly averaged or condensed by some other linear transformation over the representing amino acids at each alignment position, yielding a “position-specific scoring matrix” (PSSM). The equation for the average profile score \( s \) at alignment position \( x \) reads:

\[
\text{average distance of AB to D} = \frac{1}{2} (d_{AD} + d_{BD})
\]
An Overview of Multiple Sequence Alignment

3.7.8

A consensus sequence represents the most reduced form of a profile, with each position having one component set to one (the consensus amino acid) and all others to zero. A straightforward way to construct a consensus sequence is to choose the most frequent or most likely residue at each alignment position. Although appealing because of its simplicity, a consensus sequence carries less information than an MSA (thus it is a degenerated representation), which may lead to misinterpretations in comparisons of the consensus sequence with related sequences (Schneider, 2002). A related but more sensitive way to compress the consensus information given by an MSA is through “partial order graphs” (Lee et al., 2002), which can be viewed as a formalism that provides multiple alternative consensus sequences for nonconserved MSA regions. A partial order graph of similar sequences contains a main “consensus” branch for conserved MSA segments and loops where sequences diverge from each other. Despite this condensed representation, the entire information of the MSA is retained. A new sequence to be aligned against the MSA will then be aligned to such nonconserved regions through the most similar sequence within the alignment.

Positional Conservation

Positional conservation is an important measure for detecting homology. Conserved alignment blocks are often described as “motifs,” implicating structurally and/or functionally important parts of proteins. Frequently, even highly divergent sequence families share common motifs; sometimes such motifs are the only indication for sequence relatedness. Some databases are derived from grouping sequences with common alignment blocks or motifs into families, e.g., Blocks (UNIT 2.2) and FSSP.

A problematic aspect is the relation between positional conservation and sequence conservation. When sequences of high pairwise sequence identity are aligned, positional conservation scores are accordingly high, but mostly due to redundancy rather than true evolutionary conservation. In other words, sequences that are close in evolutionary time yield little information about true conservation patterns. This consideration has led to the usage of various weighting schemes: tree-based (Altschul et al., 1989; Thompson et al., 1994a,b), pairwise distance–based (Vingron and Argos, 1989; Sibbald and Argos, 1990; Vingron and Sibbald, 1993), and position-based (Henikoff and Henikoff, 1994).

\[
S_x = \frac{\sum_{i=1}^{20} f_i S_{ij}}{N}
\]

where \( i \) denotes amino acids, \( f_i \) is the frequency of amino acid \( i \) in alignment position \( x \), \( S_{ij} \) is the substitution score of amino acids \( i \) and \( j \), and \( N \) is the number of sequences in the alignment (Gribskov et al., 1987). The average profile score is appropriate for alignments with large \( N \), but gives poor results for small \( N \) when \( f_i/N \) deviates from the expected probability \( p_i \), of finding residue \( i \) at position \( x \). Therefore, it is advantageous to add a quantity proportional to the background probability of each amino acid to the real frequency \( f_i \), yielding:

\[
S_x = \frac{\sum_{i=1}^{20} (f_i + A q_i) S_{ij}}{N + \sum_{i=1}^{20} A q_i}
\]

where the term \( A q_i \) is called “pseudo-counts,” with constant \( A \) as weight of the pseudo-counts (relative to \( f_i \)) and \( q_i \) as background frequency of residue \( i \). There is a good theoretical justification for the use of pseudo-counts within the framework of Bayesian statistics, where they represent the prior information about the data (Durbin et al., 1998).

It must be stressed, however, that in the context of progressive multiple sequence alignment the application of pseudo-counts, and thus the incorporation of background amino acids frequencies, can well decrease proper alignment, notably during early steps of progressive alignment when sequence blocks to be aligned only contain a single or few sequences (J. Heringa, unpub. observ.). Strict Bayesian modeling treats model parameters for prior information as distributions rather than single values. Such distributions can be described by Dirichlet densities or mixtures. A Dirichlet mixture is a probability density over a set of probability vectors, in this case, vectors containing the probabilities of 20 amino acids as components, so that each vector describes a different probability distribution of the amino acids (Sjölander et al., 1996).

Another flexible motif search technique introduced by Bucher et al. (1996) uses “generalized profiles,” which are similar to hidden Markov Models (HMMs). A profile is represented by the sequence alphabet and the possible states of an alignment that are defined as begin, match, insert, deletion, and end.

A consensus sequence represents the most reduced form of a profile, with each position...
Simultaneous Alignment

The dynamic programming algorithm for pairwise sequence alignment can be extended to multiple sequences (Murata et al., 1985; Gotoh, 1986) by using a multidimensional search matrix. However, the dimensionality of the search matrix is equal to the product of all sequence lengths, \(O(L^N)\), where \(L\) is the average sequence length and \(N\) the number of sequences, rendering the search unfeasible even for moderately sized alignments. Approaches to reduce the computational load comprise reduction of the search space to near-diagonal paths (Carillo and Lipman, 1988; Wang and Jiang, 1994; Stoye et al., 1997), preselecting similar segments (Johnson and Doolittle, 1986), or word matching (Sobel and Martinez, 1986; Waterman, 1986; Vingron and Argos, 1989; Waterman and Jones, 1990).

A more recent development is mimicking simultaneous alignment by using precompiled profiles or libraries in progressive alignment. For each sequence, such a library contains information from other sequences and thus extends the sequence information used for each pairwise alignment. The idea behind multiple sequence profiles or libraries is to accumulate information from global and local alignments as well as from various sequence groupings. The accumulated information for each sequence is considered to be more reliable than single sequence alignments alone, so that match errors during progressive alignment are reduced.

The information for each sequence can be gathered by using pairwise alignments to construct a master-slave alignment. In the program PRALINE (Heringa, 1999, 2002), \(N\) master-slave alignments are constructed for \(N\) sequences, where each sequence in turn is the master sequence. The inclusion of slave sequences can be adjusted by a score threshold, so that sequences deemed too divergent are excluded and perturbation of the conservation pattern is avoided. The master-slave alignments are then converted into pre-profiles and used for the progressive construction of a final alignment. The advantage of this method is the ability to combine local and global alignment information. Moreover, sequences contained in multiple pre-profiles can be used to derive position-specific consistency scores, which effectively measure the agreement between the multiple alignment and pairwise alignments (see below).

A combination of local and global alignment is also achieved by the program T-Coffee (Notredame et al., 2000). Information from local and global pairwise alignments is complemented with information from triplet alignments that provide an alignment for each considered pair of sequences through each possible third sequence. For each pair of sequences, the contributions from the direct pairwise alignment and the triplet alignments are combined in a position-specific weight library (library extension), yielding a weight for each aligned residue pair. This library is then used to construct a final alignment by dynamic programming following the progressive alignment strategy.

Alignment Iteration

As mentioned in the above sections, the central aim of MSA methodology is to capture the complex evolutionary relationship between sequences and to convert the biological reality into a sensible scoring scheme. The ultimate step is to find the optimal mutual sequence arrangement by maximizing the alignment score. While strict (multidimensional) dynamic programming guarantees that the optimal (multiple) alignment score will be found, heuristic procedures such as progressive alignment do not necessarily yield the optimal score.

A class of techniques that are able to revisit and optimize an MSA is that of iterative multiple alignment techniques. Pioneered by Hogeweg and Hesper (1984), iterative techniques attempt to enhance the alignment quality by gleaning information from a multiple alignment constructed in an earlier round, which is then applied in a next round to improve the alignment according to a given scoring scheme. Iteration can be employed to further increase the alignment score, the incentive being to reach the optimum by “hill climbing,” i.e., stepwise increase of the target function (alignment score) until convergence is reached. During iteration, the order at which the sequences are progressively aligned can be altered (Hogeweg and Hesper, 1984; Gotoh, 1996), or other criteria derived from a multiple alignment produced in the preceding round can be applied as an iterative scoring scheme (Heringa, 2002). This means that the target function of the iteration process can be different from the alignment score. In some cases it is desirable to maximize consistency, conservation, or some other function specific to the alignment problem.

Iteration is a reasonably efficient and robust technique that alleviates the greediness of the
probabilistic strategy. Results are critically dependent on the scoring scheme used, and often there is no certainty that convergence will be reached, and if so, that the converged multiple alignment will be biologically more optimal than earlier ones. The other two possible scenarios in addition to convergence are divergence, in which the program enters a route through a virtually infinite number of states, and limit cycle, in which the program recursively visits a finite number of states. In cases where a different target function than the alignment score has been used to guide iteration, a decision has to be made whether the last scoring alignment (with the maximal target function value) or the highest scoring alignment will be taken as the result after convergence has been reached. A choice between several solutions also exists in the outcome of the limit cycle and divergence scenarios. It is the task of the investigator to perform alignment iteration with intuition and knowledge, in order to choose the right combination of target functions, alignment strategies, and iterations, to gain information about the sequence set under consideration.

Probabilistic MSA

Generation of an MSA can be performed entirely in a probabilistic framework. The probability of observing a certain sequence can be inferred using a hidden Markov model (HMM). An HMM consists of character-emitting states and transitions between these states. Character emissions and state transitions are connected to probabilities (the probabilistic model) that determine the behavior of the HMM. To create a sequence, an HMM generates a Markovian path through states, i.e., each step, including character emission and transition, is independent of the previous step. Given a sequence, the probability of observing this specific sequence can be derived using the Viterbi algorithm (Viterbi, 1967). Pairwise alignment algorithms for HMMs have been described (Durbin et al., 1998). For progressive MSA, the pairwise approach is extended to include the phylogenetic inter-dependence of sequences (http://evol-linux1.ulb.ac.be/ueg/ProAlign/Loytynoja_and_Milinkovitch_2003.pdf). Probabilistic modeling of sequence alignments is theoretically and computationally involved, and was in the past largely restricted to specialists. However, recent improvements in program development and computer speed have made this approach more accessible and the quality of probabilistic alignments for nucleotide sequences is now comparable to that of standard alignment methods.

Another probabilistic modelling approach is Bayesian inference (Liu and Lawrence, 1999), where all unknown variables are treated as probability distributions. The advantage of Bayesian modeling is that it allows for inclusion of prior knowledge about the system. However, the computational burden can be prohibitive, and specific sampling techniques such as “general Markov chain Monte Carlo” may be required (Tanner and Wong, 1987). A collection of algorithms for Bayesian alignment has been described by Zhu et al. (1998).

Parallelization of MSA

With the increasing availability of computer clusters in computationally oriented laboratories, parallelization of the most time-consuming computational tasks is worth considering. Highly repetitive procedures, such as the pairwise sequence or profile alignment phase in progressive alignment, are favorable targets for parallelized (or distributed) computing. Parallelized programs are designed to split the total computational task into subtasks that are processed on separate CPUs (nodes). Implementation of parallelized code typically requires one to identify the most CPU-intensive task (frequently a loop structure) and to split it into subtasks for independent execution. For example, an MSA of 4 sequences requires 6 pairwise alignments (subtasks), which can be performed, for example, in blocks of 3 on 2 nodes (CPUs), or in blocks of 2 on 3 nodes.

The technical details of parallelization are dealt with by high-level routines provided by a parallelization interface such as the “message passing interface” package MPICH, available at http://www-unix.mcs.anl.gov/mpi/mpich/ (Gropp et al., 1996; Pacheco, 1997). If all nodes execute the same operations but perform these on different subsets of distributed data, the parallelization technology is called single-instruction multiple-data (SIMD). Parallel code is most efficient with a minimum amount of communication between the nodes and with optimal balancing of the computational load over the CPUs.

The MSA program PRALINE (Heringa, 1999, 2002) has been parallelized in the form of SIMD technology (Kleinjung et al., 2002). The scaling of computational times versus the number of employed nodes is plotted in Figure 3.7.5 for three differently sized sets of sequences. Parallelized PRALINE generated an MSA up to ten times more quickly than the
single-processor version, when tested on a set of 200 random sequences of 200 residues length.

**MSA METHODS**

MSA is an intricate problem and over the past 30 years an increasing number of approaches have been developed that try to solve it, each with their own strengths and weaknesses. Unfortunately, the diversity of the methodologies makes it difficult for nonspecialists to know which method is the best to use for a particular problem. A sensible decision can be made with a clear and thorough understanding of how the methods work, where they perform well, and what their limitations are. The methods described below are a compilation of the best characterized methods to date, and useful guidelines are proposed based on published assessments, where available. A summary of the properties and availability of the various methods, along with literature references, can be found in Table 3.7.1.

**BioPat**

BioPat is a mathematical package that includes the first-ever integrated MSA method, which is a global progressive algorithm with iteration capabilities (Hogeweg and Hesper, 1984). Initially, a coalescence tree (dendrogram) is constructed based on all pairwise similarities of the sequences to be aligned, matched by dynamic programming (Needleman and Wunsch, 1970). The method provides choices among many of the commonly used clustering techniques to build the dendrogram, such as Unweighted Pair-Group Mean Average (UPGMA; Sneath and Sokal, 1973), the present-day ancestor method (Blanken et al., 1982), or the Neighbor-Joining (NJ) method (Saitou and Nei, 1987). Once the dendrogram is completed, the sequences are progressively aligned following the branch order of the dendrogram. The resulting alignment is then used to infer the associated new pairwise similarities and the initial dendrogram is updated to produce a new alignment. A new dendrogram is then constructed iteratively, from which a succeeding alignment is created based on the increased information.

**MultAlin**

The method MultAlin (Corpet, 1988) follows the Hogeweg and Hesper (1984) approach in that it also uses hierarchical clustering for constructing a guide dendrogram and iteration. It is different in that for the alignment of two sets of sequences it uses the average similarity score between a pair of alignment columns $i$ and $j$, one from each set, which is the average over the amino acid exchange values associated with all pairwise intercolumn residue comparisions.
<table>
<thead>
<tr>
<th>Method</th>
<th>Algorithm</th>
<th>Online availability</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clustal</td>
<td>Progressive</td>
<td><a href="http://www.ebi.ac.uk/clustalw/">http://www.ebi.ac.uk/clustalw/</a></td>
<td>Thompson et al. (1994a, 1997); UNIT 2.3</td>
</tr>
<tr>
<td>DCA</td>
<td>Exact (MSA based)</td>
<td><a href="http://bibiserv.techfak.uni-bielefeld.de/cgi-bin/dca_submit">http://bibiserv.techfak.uni-bielefeld.de/cgi-bin/dca_submit</a></td>
<td>Stoye (1998)</td>
</tr>
<tr>
<td>Dialign2</td>
<td>Consistency-based</td>
<td><a href="http://bibiserv.techfak.uni-bielefeld.de/cgi-bin/dialign_submit">http://bibiserv.techfak.uni-bielefeld.de/cgi-bin/dialign_submit</a></td>
<td>Morgenstern et al. (1999)</td>
</tr>
<tr>
<td>MACAW</td>
<td>Iterative/Progressive</td>
<td><a href="http://www.bris.ac.uk/Depts/PathAndMicro/services/CGR/Common%20Files/MACAWinstructions.htm">http://www.bris.ac.uk/Depts/PathAndMicro/services/CGR/Common%20Files/MACAWinstructions.htm</a></td>
<td>Schuler et al. (1991)</td>
</tr>
<tr>
<td>MEME</td>
<td>Consistency-based</td>
<td><a href="http://meme.sdsc.edu/meme/website/meme.html">http://meme.sdsc.edu/meme/website/meme.html</a></td>
<td>Bailey and Elkan (1994); UNIT 2.4</td>
</tr>
<tr>
<td>MSA</td>
<td>Exact</td>
<td><a href="http://xylian.igh.cnrs.fr/msa/msa.html">http://xylian.igh.cnrs.fr/msa/msa.html</a></td>
<td>Lipman et al. (1989)</td>
</tr>
<tr>
<td>MUMmer</td>
<td>Suffix Tree</td>
<td><a href="http://www.tigr.org/software/mummer/">http://www.tigr.org/software/mummer/</a></td>
<td>Delcher et al. (2002); UNIT 10.3</td>
</tr>
<tr>
<td>OMA</td>
<td>Iterative DCA</td>
<td><a href="http://bibiserv.techfak.uni-bielefeld.de/oma/">http://bibiserv.techfak.uni-bielefeld.de/oma/</a></td>
<td>Reinert et al. (2000)</td>
</tr>
<tr>
<td>PileUp</td>
<td>Progressive</td>
<td>GCG software package: <a href="http://www.accelrys.com">http://www.accelrys.com</a></td>
<td>GCG (1993); UNIT 3.6</td>
</tr>
<tr>
<td>POA</td>
<td>Partial order</td>
<td><a href="http://www.bioinformatics.ucla.edu/poa/">http://www.bioinformatics.ucla.edu/poa/</a></td>
<td>Lee et al. (2002)</td>
</tr>
<tr>
<td>TNB</td>
<td>Consistency-based</td>
<td><a href="http://globin.cse.psu.edu/ftp/dist/TNB/">http://globin.cse.psu.edu/ftp/dist/TNB/</a></td>
<td>Boguski et al. (1992)</td>
</tr>
</tbody>
</table>
MULTAL

The early method MULTAL (Taylor, 1988) is very fast and constructs a dendrogram during the progressive alignment, as in the method of Feng and Doolittle (1987). It uses a fast sequential branching method to align the closest pairs of sequences first and then subsequently align the next closest sequences to those already aligned. The order in which the sequences are aligned is largely based on the global amino acid composition of the sequences, which saves the fixed cost of performing all-against-all pairwise alignments. Blocks of aligned sequences are scored by dynamic programming similar to the method MultAlin (see above), but the similarity of two alignment columns is additionally normalized by the minimum number of sequences in either of two compared alignment blocks.

MultAlign

The global progressive method MultAlign (Barton and Sternberg, 1987) establishes a simple chain order in which the individual sequences are aligned one by one. Initially, all pairwise alignment scores are determined and the two most similar sequences are matched first. During further iterations, the sequence showing the highest alignment score when matched with the prealigned sequence block is added to it. To determine the alignment score, each sequence position $i$, of the $k$th sequence matched with position $j$ of a pre-aligned block of $k-1$ sequences receives a score per matched position averaged over the corresponding residue substitution values. The PAM-250 substitution matrix (Dayhoff et al., 1983) is utilized, with a constant of 8 added to remove all negative matrix elements. Matched gaps are evaluated by the lowest exchange weight of zero.

The MultAlign method incorporates iteration capabilities in that the resulting MSA can be progressively refined by realigning each sequence with the previous alignment from which that sequence is deleted: i.e., sequence A1 is matched with aligned sequences A2...AN; sequence A2 is then realigned with the alignment of A1, A3...AN, and so forth. This process is repeated until all $N$ sequences are realigned. Barton and Sternberg (1987) recommend two such complete refinement cycles.

The quality of an MSA is assessed by comparisons with alignment scores over randomized sequences if the sequence groups are not too large; otherwise, normalized alignment scores (NAS) are used, which is the alignment score divided by either the length of the shorter matched sequence or the number of residues not aligned with gaps.

ClustalW, ClustalX

ClustalW (Thompson et al., 1994a; UNIT 2.3) and the later window graphic user interface (GUI) version ClustalX (Thompson et al., 1997) are the newest versions of the global progressive alignment algorithm Clustal (Higgins and Sharp, 1988), and are generally considered as the standard method for MSA. The progressive strategy used is a simplification of the original Feng and Doolittle (1987) scheme.

The alignment is constructed by first building a guide dendrogram using Neighbor-Joining (NJ; UNIT 6.5; previous versions used the UPGMA strategy), based on sequence similarity, which is subsequently used to order successive pairwise alignments. The already aligned sequences are reduced to a profile for the subsequent pairwise alignment (previous versions used position consistencies). However, during the progressive alignment process, highly specialized heuristics are applied to try and optimize how the sequence information is processed. When the sequences are ordered for alignment according to the precomputed dendrogram, the alignment of distantly related sequences is delayed, thus overriding the dendrogram. This is implemented to correct for the limitation of progressive alignment, which does not allow alterations of the alignment once a sequence has been aligned even if later-added sequences may require it—"once a gap always a gap" (Feng and Doolittle, 1987). Also the pairwise alignments are performed using local gap penalties and there is automatic selection and adjustment of the residue substitution matrix and gap penalties, respectively.

ClustalW and ClustalX (UNIT 2.3) perform best when the sequences to be aligned are global cases and have no obvious outlier. The evolutionary distance between the sequences must be relatively low, thus producing a dense dendrogram. Also, the penalty scheme used by ClustalW/X discriminates against long insertions and deletions (indels) and will, therefore, exhibit reduced accuracy in such cases. In multidomain-alignment cases its accuracy is greatly reduced compared to more recent global or local algorithms (Lassmann and Sonnhammer, 2002) such as POA (Lee et al., 2002), Dialign2 (Morgenstern, 1999), and T-Coffee (Notredame et al., 2000).
The algorithm is reasonably fast and can handle sizeable sets of sequences. However, its speed decreases when it is given very large sets of data, such as genomic data, and the overall performance is less accurate when compared to the other available methods (Lassmann and Sonnhammer, 2002) such as POA (Lee et al., 2002).

MSA

The simultaneous alignment algorithm MSA (Lipman et al., 1989) employs multidimensional dynamic programming. Note that MSA here denotes the name of the Lipman et al. algorithm rather than the abbreviation for multiple sequence alignment used throughout this unit. To reduce computations, the MSA method employs the Carillo and Lipman (1988) approach, which estimates, using pairwise alignments, how much around the $N$-dimensional search matrix diagonal needs to be searched, where $N$ is the number of sequences to be aligned. The Carillo and Lipman method generalizes the earlier pairwise diagonal strip method of Ficket (1984) to $N$ dimensions. Although this approximation of simultaneous alignment optimizes the sum-of-pairs score, which in principle is much more accurate and error-free than progressive methods using the same optimization, it has huge limitations in how many sequences it can simultaneously align due to its excessive memory and computational requirements. Up to 10 sequences of 200 to 300 residues in length can be aligned with the MSA method. The method addresses an additional problem in the comparison of multiple sequences, which is the weighting of the aligned sequences, as similar sequences should not dominate the final alignment. Lipman et al. (1989) used the weighting scheme suggested by Altschul et al. (1989) based on phylogenetic trees. More recently, the MSA method was extended to larger data sets using a divide-and-conquer strategy (Stoye et al., 1997) implemented in the method DCA (see below).

DCA, OMA

The DCA (Divide-and-Conquer MSA) method (Stoye, 1998) is an exact divide-and-conquer (Stoye et al., 1997) alignment algorithm. DCA follows the same strategy as the MSA algorithm by Lipman et al. (1989) and performs simultaneous MSA instead of the progressive approach (Feng and Doolittle, 1987). The DCA approach is an attempt to overcome the computational complexity of the MSA method (Lipman et al., 1989). The divide-and-conquer strategy first selects the longest sequence in the set to be aligned and cuts it near its midpoint. The rest of the sequences are also cut at suitable positions, which are calculated through a heuristic method to reduce computational time, and consequently two new subsequence sets arise. This can then be repeated on the subsequence sets until a certain predefined minimum threshold for subsequence length is reached. The smaller the threshold value setting, the faster, but less optimal, the alignment becomes. The now shorter sets of subsequences can then be separately aligned using the MSA algorithm, thus decreasing the time and memory requirements. At the end, all the subalignments are concatenated to produce the full final alignment.

DCA represents an improvement in accuracy and speed with respect to MSA, but computational time is still extremely sensitive to sequence distance and length (Stoye, 1998) so that the number of sequences that can be aligned still remains very low. The DCA algorithm has also been implemented as an iterative scheme called OMA (Optimal MSA; Reinert et al., 2000). The OMA method represents an improvement with respect to speed and accuracy by adding face bounding, gray code enumeration, sequence weighting, realignment of cut positions, and parallelization techniques. The OMA protocol initiates a DCA alignment using a very low sequence length threshold that is only calculated once. Using this calculated threshold, a new larger threshold is produced at each iteration. This means that the alignment becomes slower at each subsequent iteration but also more optimal. The user can set the number of iterations to get a compromise between alignment speed and quality. Although OMA shows many improvements in memory usage and accuracy (Reinert et al., 2000), it is still very computationally demanding for average systems and cannot handle large data sets.

Dialign

Dialign (Morgenstern et al., 1996) is a local consistency-based alignment algorithm, which, instead of aligning single residues, aligns whole sequence segments. These segments can be envisaged as diagonals, as they would appear on a dot plot of a dot matrix analysis (Fitch, 1966; Gibbs and McIntyre, 1970). The most recent version, Dialign2 (Morgenstern, 1999) initially performs all pairwise
alignments of the sequences to be aligned, after which all ungapped segments (diagonals) are identified. Consistent sets of diagonals are then determined and added sequentially to the alignment using an iterative mathematical procedure that determines the optimal order of addition. Only sequence fragments for which matched segments are found are aligned; regions in-between blocks of similar segments are left unaligned. The improvement of Dialign2 compared to Dialign1 is the alteration of the original weighting of diagonals, which was previously based on Altschul and Erickson (1986). The Dialign2 algorithm is both an accuracy and computational time improvement over the original method.

Morgenstern (1999) reported that Dialign2 outperforms many local and global algorithms in identifying related motifs, such as ITERALIGN (Brocchieri and Karlin, 1998), ClustalW (Thompson et al., 1994a,b), MultAlin (Corpet, 1988), DCA (Stoye, 1998) and MatchBox (Deperieux and Feytmans, 1992). Dialign2 has also been shown to perform well in both local and global cases of varying evolutionary distance and, more recently, in multidomain cases (Lassmann and Sonnhammer, 2002) against T-Coffee (Notredame et al., 2000), POA (Lee et al., 2002) and ClustalW (Thompson et al., 1994a,b; UNIT 2.3).

MEME

The program MEME (Bailey and Elkan, 1994, 1995; UNIT 2.4) is a tool for unsupervised motif searching within DNA and protein sequences, which operates using an expectation maximization (EM) algorithm. It finds occurrences of motifs by comparing the residue composition at each position of a putative motif against the general composition of background sequence regions that do not display the motif. Regions showing the most discriminating compositions are then selected as motifs. A limitation of the MEME motifs is that they are ungapped, but the program can find multiple occurrences in individual sequences, which on the other hand do not need to be encountered within each input sequence. Another useful feature of the MEME method (also see UNIT 2.4) is that it is geared towards finding DNA palindrome sequences, which are often implicated as DNA-binding sites for proteins. To increase the chance of finding palindrome sets, the nucleotide probabilities of corresponding motif columns (column 1 and W, 2 and W−1, and so forth, with W the width of the motif) are constrained to be the same.

ITERALIGN

The program ITERALIGN (Brocchieri and Karlin, 1998) is a local iterative algorithm that optimizes the consistency between local pairwise alignments and their embedding in an MSA across all input sequences. It first aligns all ungapped regions of significant local similarity. The high-scoring regions are then iteratively replaced by a consensus, based on the distance between them. The existing consensus set is used as input to the next round, until convergence is reached. Core blocks are extracted and optimized using local dynamic programming to further enhance the result. Finally, these blocks are linked to produce the final alignment. Each of these aligned blocks can then be studied independently as a potential functional/structural unit. The authors went so far as to edit individual sequences by replacing amino acids with those that are preponderant at a corresponding position in an MSA, to achieve better recognition of crucial alignment regions.

MACAW

Schuler et al. (1991) introduced the iterative local progressive alignment algorithm MACAW (MSA Construction and Analysis Workbench), which allows the user to lock or shift regions in an alignment, while nonlocked subsequences are aligned automatically. The method is semiautomatic and produces blocks of alignments shared by all or a subset of the sequences. It is possible to iteratively define conserved regions such that the fraction of poorly defined segments which must be aligned automatically become fewer at each iteration cycle. The GIBBS method of Lawrence et al. (1993) has been incorporated in the MACAW procedure to detect the local fragments.

Match-Box

The method Match-Box (Deperieux and Feytmans, 1992) aims to find ungapped sequence regions with a high degree of similarity across a set of input sequences. This is achieved by comparing the frequency distribution of all pairwise aligned sequence fragments (which are gathered from global alignments), with that derived from shuffled sequences. Using a set of the most similar nine-residue fragments, local alignments are created for each fragment, if similarity beyond a threshold is found with segments across all other sequences. Boxes of ungapped regions are then delineated from these local alignments and assembled in a final alignment with unaligned amino acids and gaps in between the boxes. The method also gener-
An Overview of Multiple Sequence Alignment

3.7.16

iates a reliability index for the aligned positions within the boxes, which relies on statistics derived from analyzing a relatively small number of known family alignments.

**PileUp**

The GCG package alignment program PileUp (GCG, 1993; unit 3.6) is a global progressive alignment algorithm. It creates an MSA using a simplification of the progressive alignment method of Feng and Doolittle (1987). It generates an UPGMA-based dendrogram and, for the alignment of two sets of matched sequences, uses the average alignment similarity score of Corpet (1988).

PileUp is limited to 500 sequences, with any single sequence in the final alignment restricted to a maximum length of 7000 characters. If longer sequences are included in the alignment, the number of sequences PileUp can align decreases.

**Prp**

Prp (Gotoh, 1996) is a global iterative stochastic alignment algorithm. This algorithm is a double-nested strategy for MSA optimization. In the inner iteration, the sequences are divided into two groups and subsequently realigned using a global group-to-group alignment algorithm. When the inner iteration converges, new pairwise sequence weights are derived from a dendrogram constructed with the UPGMA cluster criterion and used to calculate the alignment scores when sequence blocks are matched. When these weights converge, the outer iteration stops.

Gotoh (1996) reported improved accuracy when compared to ClustalW (Thompson et al., 1994a,b; unit 3.3). These results were confirmed using JOY, a database of structural alignment (Thompson et al., 1999b) and later on BALiBASE (Thompson et al., 1999a) in the assessment of T-Coffee (Notredame et al., 2000).

**POA**

The Partial Order Alignment (POA; Lee et al., 2002) is an extension of the conventional dynamic programming approach. Instead of performing pairwise alignments following a specific order (from a guide tree), sequences are aligned in the order in which they are given. The growing MSA is represented by a “partial order graph,” in which identical residues within a column are fused and the information of the sequence origin is stored. Thus, despite the condensed representation, all of the information of the MSA is retained. A typical PO-MSA of similar sequences contains a main “consensus” branch and loops where sequences diverge from each other. The POA dynamic programming matrix reflects this structure by adopting the bifurcation points, so that the matrix consists of multiple two-dimensional layers that part and rejoin according to the PO-MSA graph. The best alignment is found by a conventional trace-back operation. The POA algorithm guarantees that each sequence is aligned to the closest sequence in the growing MSA.

POA is a novel local progressive algorithm. The novel feature of this method is that it employs partially ordered graphs to represent aligned sequences instead of profiles (see Profiles, above). The progressive strategy for this method does not follow a guide dendrogram to determine the order in which the sequences will be aligned, but aligns the input sequences in the order in which they are given. Each time a new sequence is added to the growing alignment, it is aligned with the most closely related hybrid sequence within the MSA as given by the partial order graph. Pairwise alignments are performed using the Smith-Waterman algorithm (Smith and Waterman, 1981), which is extended to accommodate the partial order graph representation. The partial order graph is constructed in two main steps: first, the sequences are converted to PO-MSA (Partial Order-MSA) data structures. Next, the most closely related pair of PO-MSAs are aligned and the identical residues are fused into nodes (like the knots on two ropes tied together at points along their length), while the remaining residue origins and positions are recorded and considered as incoming and outgoing (directed) edges from each node (the rope “bubble” before and after each knot; see Fig. 3.7.6). When the partial order graph is then aligned to the next PO-MSA, aligned identical residues are fused regardless of whether they are nodes or edges, and aligned nonidentical residues are recorded as aligned. Finally, any edges connecting the same pair of nodes are removed.

POA combines accuracy and speed (Lassmann and Sonnhammer, 2002). It performs marginally worse than the global method T-Coffee (Notredame et al., 2000) and local method Dialign2 (Morgenstern, 1999) in sequences of varying evolutionary distance. In multidomain alignment cases it performs comparably to Dialign2 and better than the global methods T-Coffee and ClustalW (Thompson et al., 1994a,b; unit 3.3). Finally, it is by far the fastest method to date and can handle very large data sets, although its accuracy in these cases,
particularly since the order in which the sequences are treated is not optimized, has not yet been determined.

**PRALINE**

PRALINE (PRofile ALIgNmEnt; Heringa, 1999) is a global progressive algorithm. Its distinctive feature is that it integrates many strategies for alignment optimization, which will be described below. The progressive alignment strategy does not use a precalculated search dendrogram but performs, at each alignment step, a full profile search with the most recently aligned sequence block. It, therefore, re-evaluates, at each alignment step, which sequences or blocks of sequences should be aligned, and hence determines the alignment order during progressive alignment. The pairwise alignments are performed using dynamic programming. PRALINE offers a number of strategies to optimize the quality of MSA, such as local global alignment and global and local profile preprocessing, and has weighted iteration capabilities. In addition, it can integrate secondary structure to guide the resulting alignment.

PRALINE is more of a tool kit than a one-step MSA method. It allows the user to apply or combine different strategies to a given problem and find the best solution, rather than applying a single approach to everything. The local global optimization strategy performs a local alignment of the sequences first, and then, using that information, performs a final global alignment. The profile preprocessing strategy is aimed at incorporating into each sequence the most trusted information from other sequences, either through global or local alignments. For each sequence, a preprocessed alignment is created including only those sequences that score beyond a user-specified threshold when aligned pairwise with the sequence considered. A low threshold would result in a preprocessed alignment for each sequence comprising all other sequences, while higher thresholds would allow fewer and fewer sequences into the alignment. For each of the formed preprocessed alignments, a profile is constructed. PRALINE then performs progressive MSA using the preprocessed profiles to represent each of the original sequences. Because the preprocessed profiles for each of the sequences incorporate knowledge about other sequences (in particular similar sequences) and comprise position-specific gap penalties, they enable increased matching of distant sequences and likely placement of gaps outside ungapped core regions during progressive alignment (Fig. 3.7.7). The MSA of the preprocessed profiles can also be used to derive consistency scores for each amino acid in the alignment, which for each sequence reflect the consistency among the pairwise alignments used that include the sequence. The consistency of preprocessed profiles can also be used to optimize the alignment through iteration. Since each of the preprocessed profiles can contain information about all the sequences, each sequence in the final alignment can be assessed in terms of the degree of consistency reached across the profiles, which is translated into a consistency score for each amino acid in the MSA. Iteration is then guided by these obtained scores, which are used as weights in the construction of align-

---

**Figure 3.7.6** The Partial Order Graph (POA) alignment representation of the C-termini of a pair of flavodoxin proteins. (A) The alignment in standard format; (B) The alignment in POA representation (adapted from Lee et al., 2002).
An MSA of the flavodoxin family members (13 proteins) created by PRALINE (Heringa, 1999) using local preprocessing with a threshold of 300. The bottom sequence is the cheY sequence (PDB code 3chy), which is an outlier.

The secondary structure incorporation proceeds by initially constructing an MSA without any information about the corresponding secondary structure. Then, for each sequence, the secondary structure is predicted by the PREDATOR (Frishman and Argos, 1996, 1997) or the PHD method (Rost and Sander, 1993), although in principle this could be done by any available method, and iteratively a new alignment is constructed, now using the predicted secondary structure. PRALINE employs dynamic programming to progressively construct an MSA for a query set of sequences (and/or profiles representing already aligned sequence blocks) are matched using three secondary structure-specific residue exchange matrices (Lüthy et al., 1994) and associated gap penalties. The residual exchange weights for matched sequence positions with identical secondary structure states are taken from the corresponding residue exchange matrix; matched sequence positions with nonidentical secondary-structure states are assigned the corresponding value from the default exchange matrix—e.g., the BLOSUM62 matrix (UNIT 3.5). The secondary-structure information is used in a conservative manner, based upon the assumption that consistent secondary structure predictions are indicative of their reliability. Secondary-structure prediction can also be used to optimize the alignments in an iterative fashion. Most reliable secondary-structure prediction methods utilize sequence information in MSAs, and their prediction accuracy relies on the quality of the MSA used. In the PRALINE approach, the MSA is guided by predicted secondary structure, so that an iterative scheme arises that optimizes both the quality of the MSA and that of the secondary-structure prediction. In this iterative scenario, each iteration proceeds in the same way as described above,
producing an MSA that is passed on to the next iteration and guides secondary-structure prediction, which in turn guides alignment and so on.

A future development for incorporating the secondary structure into the creation of an MSA is the use of an optimally segmented secondary-structure consensus for each sequence to be aligned. The optimization step is performed using weighted dynamic programming and can involve the production of a consensus from a single or multiple prediction methods. The optimal segmentation works based on the fact that current state-of-the-art predictors process alignments prior to prediction, causing inaccurate predictions in regions where alignment blocks have been merged. Each consensus produces a structure-based score giving the prediction a quality measure, which can be used in combination with an equivalent residue-based score to give the whole alignment an overall quality measure. In the PRALINE iterative scenario, the structure-based score can be used to select the best-scoring secondary structure predictions for each sequence, from all iteration cycles, which can then be used to perform a separate final structure-guided alignment. Also, the alignment produced at each iteration can be assessed according to combined structure- and residue-based scores, enabling improved quality evaluation.

PRALINE is a very diverse package. When used as a tool kit it can overall outperform both of the currently popular methods ClustalW (Thompson et al., 1994a,b; \textit{UNIT 2.3}) and T-Coffee (Notredame et al., 2000) in varying evolutionary distance and multidomain cases. However, this requires time, because many strategies can be tried until the best solution is found. In addition, the use of these strategies makes PRALINE relatively slow. However, PRALINE has been parallelized (Kleinjung et al., 2002), yielding a ten-fold acceleration compared to single-processor execution (see Parallelization of MSA, above).

**SAGA**

The program SAGA (Sequence Alignment by Genetic Algorithm; Notredame and Higgins, 1996) is an iterative stochastic alignment method that uses a genetic algorithm (GA; Goldberg, 1989) to select the alignment from an evolving alignment population and which optimizes, as an Objective Function (OF), the weighted sum of pairs as used in the MSA program. The algorithm initially generates a random population of alignments of the sequences, called generation zero (G0). Offspring alignments are then generated from the parent alignments in G0 that are evaluated for fitness based on alignment quality. The better the alignment, the more offspring alignments it creates. The operators for offspring alignment creation can be either the mixing of the contents of the parent alignments (crossovers) or the alteration of a single parent (mutation). This process is iterated through successive generations, allowing only the fittest (best-quality) offspring alignments to proceed to the next generation and produce their own offspring alignments. The iteration process stops when no more improvement can be achieved.

SAGA was found to produce overall better scoring alignments when compared to MSA (Lipman et al., 1989) and ClustalW (Thompson et al., 1994a,b; \textit{UNIT 2.3}) and T-Coffee (Notredame et al., 2000), where a single global alignment is generated, and once with the local alignment method Lalign (Huang and Miller, 1991) where 10 top-scoring nonintersecting local alignments are generated. The results are pooled into a primary library of combined weights for each nonredundant residue pair. The combined weight for each residue pair $(x, y)$ corresponds to the sum $(\Sigma)$ of scores $(S)$ of the global and local alignments containing that residue pair.

Each alignment score $(S)$ is the percentage sequence identity of that alignment. A library extension step is then performed using a procedure called matrix extension (Notredame et al., 2000) to measure how residue pairs align with respect to other residues in the library, producing triplet weights. These triplets are then used to assess how well sequences are aligned compared to the other sequences in the data set, rather than looking at pairs of sequences in isolation. The final alignment is built by performing the library extension step to
produce a guide dendrogram, which then orders how the sequences are aligned.

The assessment of Lassmann and Sonnhammer (2002) show that T-Coffee is one of the most reliable MSA methods to date in cases of low to moderate evolutionary distance, with higher accuracy compared to ClustalW (Thompson et al., 1994a,b; UNIT 2.3), Prr (Gotoh, 1996), Dialign2 (Morgenstern, 1999), and POA (Lee et al., 2002). In multidomain cases, it shows good performance as a global alignment method and is only outperformed by local alignment strategies.

However, T-Coffee has speed and computational demand limitations when alignments (>30 sequences) of large sequences (>10,000 residues) are performed and may even fail to complete them on average-powered systems.

**TNB**

The Boguski et al. (1992) semimanual program suite TNB combines the space-efficient local alignment routine SIM (Huang et al. 1990) and the method MSA (Lipman et al., 1989). The pairwise alignment strategy starts by identifying the highest-scoring gap-containing local alignments. From these, the non-gapped regions occurring in each of the sequences are extracted. The adjacent blocks of such motifs are then aligned with the intervening sequence fragments using the MSA method, thus allowing gaps. This method also provides a user interface through which parts of the alignment can be manually edited.

**MUMmer**

Genome-wide sequence alignments require very fast algorithms that can handle millions of nucleotides. The alignment system MUMmer (Delcher et al., 1999, 2002; UNIT 10.3) uses “suffix trees,” which allow for an alignment of two entire genomes in linear time and space. The program finds “maximal unique matches” (MUMs) between two input sequences. A suffix tree is a unique character string where the sequences are identical. A new branch is created where they differ. MUMmer creates a suffix tree based on one (reference) sequence and streams the second (query) sequence against it. MUMmer has been used to assemble contigs from shotgun-sequencing to construct the complete genome.

**MAFFT**

The MAFFT program (Katoh et al., 2002) is based on the fast Fourier transform (FFT) for rapid detection of homologous segments. Amino acids are represented by volume and polarity values, yielding signal peaks if homologous segments are aligned. The recorded segments are joined to a final alignment by dynamic programming.

**ASSESSMENT OF MSA**

In this section, MSA benchmarking issues, as well as the scoring schemes currently in use to evaluate MSA quality using reference alignments, are discussed. As high alignment scores do not necessarily entail a good biological quality, MSA score optimization is also briefly discussed.

**Scoring MSAs**

The score of an MSA is the target function that is optimized by the (progressive) alignment algorithm. Most programs use a Dayhoff-type substitution matrix as evolutionary model and compute the sum-of-pair (SP) score of all $n(n - 1)/2$ combinations of aligned residue pairs (Lipman et al., 1989). Since pairwise alignment algorithms optimize the alignment score based on residue exchange scores and gap penalties, an obvious way of scoring MSAs is to extend the pairwise sequence scores to MSAs. This is referred to as the SP score for alignment: for each amino acid $a_{i,j}$ in sequence $i$ and at position $j$ in the MSA, the SP score is $S(j) = \sum_{k<l} s(a_{k,l}, a_{i,j})$, where $s(a_{k,l}, a_{i,j})$ is the amino acid exchange value. Using the SP alignment column scores, alignments are scored by taking the total sum of the SP scores: $S = \sum_{j=1}^{N} S(j)$, where $N$ is the number of aligned positions. In some applications, the occurrence of each gap in the MSA is penalized using typical gap penalty values. However, the SP score is problematic in several regards. It overweights the evolutionary events close to the root of the alignment tree relative to the leaves. Therefore, a “circular sum” (CS) score derived from the “Traveling Salesman Problem” has been proposed to substitute the SP score (Gonnet et al., 2000).

**Evaluating Alignment Methods**

Evaluating MSA programs is a complex issue. First of all, there is no general agreement as to what the standard of truth should be. For instance, should an alignment be evaluated using evolutionary, structural, or functional criteria? Although in closely related familial sequences these criteria are expected to lead to the same alignment, in more distant cases they can result in very different answers. Moreover, benchmarks are usually carried out using a set of reference alignments, so that the evaluation
becomes crucially dependent on the quality of such a reference alignment database. A few recent attempts to alleviate this database problem are based on using protein 3-D structures directly in assessing the alignments (Notredame et al., 2000). Furthermore, different ways have been proposed to quantify the agreement between a proposed and a reference alignment, such as unweighted or weighted SP scores, or the column score. The unweighted SP score implies checking, for each aligned amino acid pair in the reference MSA, whether this pair has also been aligned in the alignment produced by the method considered. The final score usually is the percentage of the total number of aligned pairs in the reference alignment that have also been matched in the query alignment. The weighted SP score follows basically the same protocol but weights each pair with the corresponding value from an amino acid exchange matrix (e.g., Blossum62). Finally, the column score checks, for each column in the reference alignment, whether the amino acids found aligned here have been reproduced exactly in the query alignment. If only one sequence is misaligned at the column considered, the whole column is taken to be incorrectly reproduced. Compared to the unweighted and weighted SP scores, the column score is a more stringent measure for alignment evaluation. For example, an outlier sequence that is distant from all other sequences in the query set has a relatively high chance of becoming misaligned, and this will be reflected much more dramatically in the column score than in either SP scores.

Optimizing Alignment Scores

The SP scores of “incorrect” sequence alignments are often higher than those of “true” reference alignments derived from structural superpositioning of proteins. This is caused by a lack of structural information in the substitution models. Heringa (2002) calculated SP scores (for single alignments) for each of the BALiBASE benchmark alignments (Thompson et al., 1999a). These scores were then compared to corresponding SP scores of the alignments calculated using nonoptimized PRALINE conditions. More than a quarter of the PRALINE alignments turned out to have higher SP scores than the corresponding reference alignments, while for the largest BALiBASE alignments, more than half attain larger SP scores for the default PRALINE alignments than those of the corresponding reference alignments. This might be referred to as the “Charlie Chaplin” problem—at the peak of his fame, Charlie Chaplin allegedly entered a Charlie Chaplin contest incognito and came in second (Heringa, 2002). It is clear that trying to optimize the SP score for alignments that already score higher than their corresponding reference alignments is not likely to lead to convergence to the latter alignments.

New avenues to novel scoring schemes, as well as benchmarking methods, are being actively researched. Lin et al. (2003) introduced a new alignment-scoring scheme CAO (Contact Accepted mutatiOn) based on the amino acid interactions in tertiary structures. The scheme is based on a new evolutionary model expressed in 400×400 residue contact mutation matrices, and can be used to evaluate alignments whenever there is a tertiary structure at hand for one or more of the sequences, from which the pairwise residue contacts can be derived. Since the contact-based evolutionary model combines sequence and structure information, it yields biologically more meaningful alignment scores (Lin et al., 2003).

CONCLUSION

An MSA can be viewed as an inflexible representation that provides a unified picture of sequence similarity by averaging out matched residues that possibly cannot be consistently matched over the entire lengths of the sequences. This is because evolution, through mutations, insertions, and deletions of sequence fragments, works on spatially and temporally decoupled molecules, so that sequence alignment incompatibilities can well arise under divergent evolution.

Given these complications, building a reliable MSA for a query set of sequences is a daunting task. In this unit it has been made clear that the increased attention to multiple sequence alignment methodology has resulted in recent developments regarding most of its facets. Computational issues have been addressed by both adapting methods to high-throughput computing by code parallelization and by new speed-optimized alignment formalisms, such as the recent method POA (Lee et al., 2002). Sensitivity has been increased by the development of enhanced techniques for carrying out simultaneous alignment, by devising new profile formalisms, by combining local and global alignment, by new iterative schemes, and by the emergence of new schemes to score and exploit the consistency of alignments.

The increased focus has also led to the construction of new benchmark databases and
novel evaluation protocols. Further developments will be crucially dependent on the integration and representation of biological knowledge in new quality criteria. There are now a multitude of high-quality MSA techniques, each with particular strengths and weaknesses. Increased sensitivity could abound as a result of new consensus protocols to utilize the combined power of the techniques, or new techniques to determine the kind of alignment problem at hand and then invoke the most appropriate method or combination of methods available. In the meantime, however, it remains important for the end user to run a combination of different MSA methods to optimize the biological information derived from a set of sequences, either through visual inspection of the resulting MSAs or by the application of other bioinformatics techniques that use these MSAs as input.

**LITERATURE CITED**


An Overview of Multiple Sequence Alignment

3.7.24

Supplement 3

Current Protocols in Bioinformatics


KEY REFERENCES
Dayhoff et al., 1978. See above.
This atlas represents a seminal approach to sequence alignment. The evolutionary model that is still used today in most methods is introduced here, together with the now classical PAM series of amino acid exchange matrices. The evolutionary model is often referred to as the Dayhoff model, while the most widely used early matrix in the PAM series, the PAM250 matrix, is commonly known as the Dayhoff matrix.

Felsenstein, 1981. See above.
In this paper, the important evolutionary method of maximum likelihood is introduced. The method, which attempts to find the tree that maximizes the probability that the observed data will fit the tree under a given evolutionary model, is now generally accepted as the most accurate strategy.

Hogeweg and Hesper, 1984. See above.
This paper introduces the progressive multiple alignment strategy, which is still the most widely used multiple alignment technique. In this early paper, alignment iteration is already addressed. Another interesting feature of the paper is the use of so-called internode sequences, which are additionally inferred sequences ancestral to subgroups of sequences in the phylogenetic tree calculated for the query sequence set.

Needleman and Wunsch, 1970. See above.
This is one of the most quoted early papers on sequence alignment. In this paper, the global dynamic programming algorithm is introduced to the biological community and applied to pairwise amino acid sequences. The basic dynamic programming algorithm had been conceived before by the physicist Richard Belman, who published a large series of papers and books on the topic during the 1950s and 60s.

Smith and Waterman, 1981. See above.
Following the approach by Needleman and Wunsch (1970), Smith and Waterman derived the now classical algorithm for local pair-wise sequence alignment. Most widely used homology search engines such as BLAST are fast approximations of the Smith and Waterman algorithm and perform local alignment.

INTERNET RESOURCES

BioPat software package
http://www.biopat.de

Clustal
http://www.ebi.ac.uk/clustalw/

DCA
http://bibiserv.techfak.uni-bielefeld.de/cgi-bin/dca_submit

Dialign2
http://bibiserv.techfak.uni-bielefeld.de/cgi-bin/dialign_submit

ITERALIGN
http://giotto.stanford.edu/~luciano/iteralign.html

MACAW
http://www.bris.ac.uk/Depts/PathAndMicro/services/CGR/Common%20Files/MACA
Winstructions.htm

MAFFT
http://www.biophys.kyoto-u.ac.jp/~katoh/programs/align/mafft/

Match-Box

MEME
http://meme.sdsc.edu/meme/website/meme.html

MSA
http://xylian.igh.cnrs.fr/msa/msa.html

MULTAL
http://mathbio.nimr.mrc.ac.uk/ftp/wtaylor/multal/

MultAlign
http://cbrg.inf.ethz.ch/Server/MultAlign.html

MultAlin
http://prodes.toulouse.inra.fr/multalin/multalin.html

MUMmer
http://www.tigr.org/software/mummer/

OMA
http://bibiserv.techfak.uni-bielefeld.de/oma/

PileUp (GCG software package)
http://www.accelrys.com

POA
http://www.bioinformatics.ucla.edu/poa/

PRLINE
http://www.cs.vu.nl/~ibivu/programs/pralinewww/

Prrp

SAGA
http://igs-server.cnrs-mrs.fr/~cnotred
**An Overview of Multiple Sequence Alignment**

3.7.26

---

**T-Coffee**
http://igs-server.cnrs-mrs.fr/Tcoffee/

**TNB**
http://globin.cse.psu.edu/ftp/dist/TNB/

**Literature on the Web**
http://evol-linux1.ulb.ac.be/ueg/ProAlign/
Loytynoja_and_Milinkovitch_2003.pdf


---

http://hmmer.wustl.edu


Contributed by Victor Simossis, Jens Kleinjung, and Jaap Heringa

Integrative Bioinformatics Institute (IBIVU)
Free University
Amsterdam, The Netherlands
Computing Multiple Sequence/Structure Alignments with the T-Coffee Package

This unit describes how to assemble a multiple sequence alignment using the T-Coffee multiple sequence alignment package (Notredame et al., 2000). Although T-Coffee is slower than its close relative ClustalW (UNIT 2.3; Thompson et al., 1994), it is better suited for aligning distantly related sequences and mixing sequences and structures. T-Coffee is also much more flexible than most methods because it makes it possible to combine many alternative alignments into a single one, based on an estimate of consistency between the alignments. The protocols below show how such a combination can be done and how alternative alignment methods can be added as separate modules to the T-Coffee process.

This unit assumes that the user wants to align a set of sequences that are more or less homologous over their entire length (see Basic Protocol 1). These sequences may have been gathered using any appropriate database search strategy. Given such a data set, the user can assemble a multiple alignment in order to carry out family modeling (profile; see Basic Protocol 2), structural modeling (see Basic Protocol 3), or phylogenetic modeling (tree computation). This multiple sequence alignment may also be used to analyze the potential effect of nsSNPs (Non-Synonymous Single Nucleotide Polymorphisms) or simply to check whether a specific sequence is a true member of the family.

Most of this unit assumes that the user is familiar with the Unix environment (without being a specialist or a programmer; see APPENDIX 1C & APPENDIX 1D). The last section (see the Appendix at the end of the unit) is a bit more demanding in terms of computer skills, but it should be relatively straightforward to anyone with a basic knowledge of the scripting language Perl.

T-Coffee is very appropriate for generating high-quality alignments, but it is more demanding in terms of resources than other, similar programs. Given a standard 2-GHz PC with 500 Mb of memory, one should not hope to align more than 100 sequences that are up to 2000 residues long when using the default mode. This figure is simply an indication, since the memory requirement depends on the relatedness of the sequences being considered (close sequences require less time and less memory). Given these limitations, it is often a good strategy to start with a rapid multiple sequence alignment method (such as ClustalW; UNIT 2.3) in order to quickly identify potential problems within the data set, before refining the results with T-Coffee. Nevertheless, the authors provide two alternative strategies that make it possible to bypass some of the limitations of T-Coffee regarding memory usage.

In this unit, protocols are presented on how to use T-Coffee in a Unix/Linux environment, taking advantage of the rich command-line options of this program. Yet, for those who prefer clickable Web interfaces, much of what is presented here can also be executed by using the Web tool Tcoffee@igs (Poirot et al., 2003), available at http://igs-server.cnrs-mrs.fr/Tcoffee/. This service is provided to the community by the CNRS and HP servers. Other online versions of this software exist, and an exhaustive list is maintained on the T-Coffee home page (accessible from the Tcoffee@igs server at the abovementioned URL).

NOTE: Investigators who are unfamiliar with the Unix environment are encouraged to read APPENDIX 1C & APPENDIX 1D.
NOTE: Throughout the unit individual commands are given over multiple lines for the sake of clarity. However, each command should be typed, in its entirety, on a single line at the Unix prompt.

**COMPUTING MULTIPLE SEQUENCE ALIGNMENTS**

This protocol is meant to show a user how to run the T-Coffee package in the simplest possible manner. Here, one will find out how to align a set of sequences, control the output of T-Coffee, and evaluate the resulting multiple alignment. Basic Protocol 2 addresses aligning a single sequence with an existing alignment (sequence-to-profile alignment).

T-Coffee is a noninteractive program and thus does not prompt the user for any input (as Clustal does, for instance). After installation of the package on a personal computer (preferably on a Unix or Linux system, although unsupported Windows and Macintosh executables are also available) the user can access the T-Coffee functionalities by typing a complete set of instructions on the command line, from a terminal window. In its simplest form, the syntax is straightforward and very much resembles ClustalW. This makes it possible to recycle most of the scripts based on ClustalW. The default parameters are set to values that are tuned to produce a meaningful result in the most common situations. Such use of T-Coffee does not require an in-depth understanding of the underlying algorithm. Yet, those interested in behind-the-scenes details can find an exhaustive description of the T-Coffee algorithm in the original publication (Notredame et al., 2000) and in the associated online documentation.

In this protocol, it is assumed that the user already has a set of homologous sequences. These sequences may have been obtained using any adequate database search strategy (e.g., UNITS 3.3 & 3.4). Given these sequences, it is assumed that the user intends to produce a multiple sequence alignment.

**Necessary Resources**

**Hardware**

T-Coffee has been especially designed to run on Unix and Linux platforms. The program is distributed as C source code and Perl scripts, and needs to be compiled. Experience shows that it is easy to install on a wide range of platforms (e.g., Irix, Sun, or HP) by following the simple three-step procedure provided in the documentation. For Windows and Macintosh users, precompiled binaries are available from the authors, although these platforms are not officially supported.

A multithreaded version of T-Coffee that can take advantage of multiprocessor machines is under development, but not yet available. This means that one does not get much benefit from running the program on a multiprocessor machine. Linux farms are not very useful either if one wishes to increase the size of the data set that T-Coffee can handle. Given the choice, users should select well endowed single-processor machines.

**Software**

*Environment and compilation:* The user is expected to have access to a common Unix/Linux environment. Standard programs, such as wget (designed for automated download of Web pages) should also be installed. For the compilation of the package, a C compiler is required. The best cross-platform reproducibility of the results is assured using the Gnu C Compiler gcc, which is freely available (http://www.gnu.org) and is the default C compiler on most Linux distributions.
**T-Coffee package:** The T-Coffee package and its source code can be downloaded from [http://igs-server.cnrs-mrs.fr/Tcoffee/](http://igs-server.cnrs-mrs.fr/Tcoffee/). Read the license before installation. This package is free of charge for academic and other nonprofit users provided that they agree not to redistribute or modify the code. Commercial users are required to take a special license when using versions higher than 1.37. The Web server Tcoffee@igs is freely accessible for all academic and commercial users.

**ClustalW package:** To follow the steps below, the user also needs to have access to the ClustalW multiple sequence alignment package ([ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalW/](ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalW/; UNIT 2.3)).

**Files**

T-Coffee is able to read and write the most common file formats (see Support Protocol 1). Sequences may be provided using the FASTA, MSF, or ALN sequence format, and an output can be produced in any of these formats. By default, the nature of the files that T-Coffee outputs is the same as that of ClustalW, including similar file extensions (.aln, .dnd).

Recognition of the file format is carried out automatically by T-Coffee. Note that file extensions (e.g., .pep or .fasta) are not used, so that users are free to use any file extension that they choose. Support Protocol 1 offers advice on reformatting sequences and existing alignments for use with T-Coffee.

1. Download, install, and compile the T-Coffee package by following the directions that come with the software. Change to the directory that contains the sequence files of interest.

   *T-Coffee online documentation including installation instructions is available at [http://igs-server.cnrs-mrs.fr/Tcoffee/](http://igs-server.cnrs-mrs.fr/Tcoffee/).*

2a. **If the sequences are in a single file:** Type, e.g., the command:

   ```
t_coffee sample4.pep
   ```

   Given a set of sequences, computing a multiple sequence alignment with T-Coffee is a one-step procedure that involves providing the program with the proper command line. The example in this step shows the simplest means of computing a multiple sequence alignment. All of the sequences in this example are contained in the file `sample4.pep`.

2b. **If the sequences are spread over several files:** Compile a unified data set before making the alignment by typing, e.g., the command:

   ```
t_coffee sample1.pep sample2.pep Ssample4.aln
   ```

   In this example, `sample1.pep` and `sample2.pep` are sequences in FASTA format, while `sample4.aln` is a multiple sequence alignment in ALN format (the default output format of ClustalW).

   When computing the multiple alignment, T-Coffee considers the sequences in the file `sample4.aln` as a collection of unaligned sequence. The program is instructed to do so by the means of the letter "S" that precedes the file name, e.g., `Ssample4.aln`. This way, the gaps within `sample4.aln` are reset, and this precomputed alignment has no influence on the final T-Coffee alignment. "S" is called a “converter” in T-Coffee jargon. Several converters exist in T-Coffee. Their effects are summarized in Tables 3.8.1 and 3.8.2, and detailed in the Commentary (see Critical Parameters and Troubleshooting).

   When combining several sets of sequences from different files, one should remember the following points: (1) T-Coffee pulls together the sequences coming from the various sets and turns them into a single unified set without duplicates (i.e., a set of sequences that does not contain two sequences with the same name but that can contain identical sequences with different names). (2) None of the individual files should contain a duplicated sequence (this makes T-Coffee data sets compliant with those of ClustalW). (3) When two sequences in two different files have the same name, they are assumed to be the same sequence, but...
if these sequences are not entirely identical, T-Coffee attempts to combine the information they contain into a new reconciled sequence (see Support Protocol 1). (4) As with most multiple sequence alignment procedures, the incoming order of the sequences can have a slight (albeit usually modest) influence on the final alignment.

3. Examine the output. When the computation of the multiple sequence alignment is finished, T-Coffee outputs the two following files (Fig. 3.8.1):

`sample1.dnd`

`sample1.aln`

The file `sample1.dnd` is the guide tree that T-Coffee has constructed and used to build its progressive alignment. This guide tree is in Newick format (UNITS 6.2 & 6.3). It is NOT a phylogenetic tree and should NEVER be used as such. The file `sample1.aln` is the multiple alignment. It comes in a format similar to the one of ClustalW, so most programs that parse ClustalW files should not have difficulty recognizing the ones from T-Coffee. In this file, perfectly conserved positions are indicated with an asterisk (*), highly conserved ones with a colon (:), and partly degenerated with a period (.).

### Table 3.8.1 Output Formats Available in T-Coffee

<table>
<thead>
<tr>
<th>Format name</th>
<th>Description</th>
<th>Input converter (see Table 3.8.2)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>clustalw</td>
<td>Clustal alignment</td>
<td>S, A, R</td>
<td>Yes</td>
</tr>
<tr>
<td>t_coffee_aln</td>
<td>Clustal-like</td>
<td>S, A, R</td>
<td>Yes</td>
</tr>
<tr>
<td>fasta_aln</td>
<td>FASTA alignment</td>
<td>S, A, R</td>
<td>Yes</td>
</tr>
<tr>
<td>fasta_seq</td>
<td>FASTA sequence</td>
<td>S</td>
<td>Yes</td>
</tr>
<tr>
<td>Swissprot_seq</td>
<td>SwissProt sequence</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>pir_aln</td>
<td>PIR alignment</td>
<td>S, A, R</td>
<td>Yes</td>
</tr>
<tr>
<td>pir_seq</td>
<td>PIR sequence</td>
<td>S, A, R</td>
<td>Yes</td>
</tr>
<tr>
<td>Msf_aln</td>
<td>MSF alignment</td>
<td>S, A, R</td>
<td>Yes</td>
</tr>
<tr>
<td>tc_lib</td>
<td>T-Coffee library</td>
<td>L</td>
<td>Yes</td>
</tr>
<tr>
<td>pdb</td>
<td>PDB</td>
<td>P</td>
<td>No</td>
</tr>
<tr>
<td>matrix</td>
<td>Substitution matrix</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>score_html</td>
<td>Evaluated alignment</td>
<td>None</td>
<td>Yes</td>
</tr>
<tr>
<td>score_pdf</td>
<td>Evaluated alignment</td>
<td>None</td>
<td>Yes</td>
</tr>
<tr>
<td>score_ascii</td>
<td>Evaluated alignment</td>
<td>None</td>
<td>Yes</td>
</tr>
<tr>
<td>dnd</td>
<td>Dendrogram</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Table 3.8.2 Format Converters

<table>
<thead>
<tr>
<th>Converter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Sequences within the file must be treated as unaligned sequences</td>
</tr>
<tr>
<td>A</td>
<td>Aligned sequences</td>
</tr>
<tr>
<td>R</td>
<td>The alignment within the file must be treated as a profile</td>
</tr>
<tr>
<td>L</td>
<td>T-Coffee library file</td>
</tr>
<tr>
<td>P</td>
<td>PDB structure</td>
</tr>
<tr>
<td>X</td>
<td>Substitution matrix</td>
</tr>
</tbody>
</table>
4. Change the output format if necessary using the `-output` flag.

   The `-output` flag can be used when it is necessary to output a format other than the default. For instance, assuming that the user wishes T-Coffee to align the sequences in `sample4.pep` and requires output of the multiple alignment in three additional formats (e.g., FASTA, MSF and ALN), the following command may be used:

   ```
   t_coffee sample4.pep
   -output=clustalw,msf_aln,fasta_aln,fasta_seq
   ```

   With this command, T-Coffee outputs the following five files (Fig. 3.8.2):

   ```
   sample1.dnd
   sample1.aln
   sample1.msf_aln
   sample1.fasta_aln
   sample1.fasta_seq.
   ```

   Table 3.8.1 lists some of the output/input formats available in T-Coffee. Multiple alignments come in two different types of formats: interleaved (ALN or MSF) and noninterleaved (e.g., FASTA or PIR). If some of the residues in the input sequences have been uppercased/lowercased, the user needs to use a format that preserves this casing, such as `pir_aln` (noninterleaved), `fasta_aln` (noninterleaved) or `t_coffee_aln` (an interleaved format very similar to the ClustalW format).

5. One can change the output filename using the `-run_name` flag.

   In order to avoid overwriting pre-existing files, the user can change the prefix of the output files. By default, T-Coffee uses the prefix of the first data file that appears on the command line (`sample4` in this case), but the user can alter this behavior using the `-run_name` flag:

   ```
   t_coffee sample4.pep
   -output=clustalw,msf_aln,fasta_aln
   -run_name=yourprefix
   ```

   Given this command line, T-Coffee generates the following four files, where the prefix takes the value specified via the `-run_name` flag:

   ```
   yourprefix.dnd
   yourprefix.aln
   yourprefix.msf_aln
   yourprefix.fasta_aln
   ```
6. Consider evaluating the alignment on the fly with `-output=score_pdf` (see Support Protocol 2).

7. When running multiple analyses on the same data set, consider exporting and reusing the library file by adding the flag `-out_lib=<your_library_name>` to the command line (see Support Protocol 3 and Commentary).

**CREATING PROFILE ALIGNMENTS FROM LARGE DATA SETS**

The default mode of T-Coffee is limited to small data sets (less than 100 sequences). This protocol shows how T-Coffee can be used to align larger data sets by first assembling small alignments, then making a multiple alignment of the small multiple alignments (the so-called profile alignment).

**Necessary Resources**

**Hardware**

T-Coffee has been especially designed to run on Unix and Linux platforms. The program is distributed as C source code and Perl scripts, and needs to be compiled. Experience shows that it is easy to install on a wide range of platforms (e.g., Irix, Sun, or HP) by following the simple three-step procedure provided in the documentation. For Windows and Macintosh users, precompiled binaries are available from the authors, although these platforms are not officially supported.

A multithreaded version of T-Coffee that can take advantage of multiprocessor machines is under development, but not yet available. This means that one does not get much benefit from running the program on a multiprocessor machine. Linux farms are not very useful either if one wishes to increase the size of the data set that T-Coffee can handle. Given the choice, users should select well endowed single-processor machines.

**Software**

*Environment and compilation:* The user is expected to have access to a common Unix/Linux environment. Standard programs, such as wget (designed for automated download of Web pages) should also be installed. For the compilation of the package, a C compiler is required. The best cross-platform reproducibility of the results is assured using the Gnu C Compiler gcc, which is freely available ([http://www.gnu.org](http://www.gnu.org)) and is the default C compiler on most Linux distributions.

*T-Coffee package:* The T-Coffee package and its source code can be downloaded from [http://igs-server.cnrs-mrs.fr/Tcoffee/](http://igs-server.cnrs-mrs.fr/Tcoffee/). Read the license before installation. This package is free of charge for academic and other nonprofit users provided that they agree not to redistribute or modify the code. Commercial users are required to take a special license when using versions higher than 1.37. The Web server Tcoffee@igs is freely accessible for all academic and commercial users.

*ClustalW package:* To follow the steps below, the user also needs to have access to the ClustalW multiple sequence alignment package ([ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalW/](ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalW/); **UNIT 2.3**).

**Files**

T-Coffee is able to read and write the most common file formats (see Support Protocol 1). Sequences may be provided using the FASTA (**APPENDIX IB**), MSF, or ALN sequence format, and an output can be produced in any of these formats. By default, the nature of the files that T-Coffee outputs is the same as that of ClustalW (**UNIT 2.3**), including similar file extensions ( .aln, .dnd).
Recognition of the file format is carried out automatically by T-Coffee. Note that file extensions (e.g., .pep or .FASTA) are not used, so that users are free to use any file extension that they choose.

**Split data set into smaller data sets**

1. Make a fast alignment using ClustalW (*UNIT 2.3*).

2. Use the .dnd file that is output when the alignment is finished to split the data set.

*Users can run Jalview (http://www.es.embnet.org) or TreeView (*UNIT 6.2*) to visualize the tree associated with a multiple sequence alignment (Fig. 3.8.3) and can divide the data set into smaller ones using this information. In the current example, sample4.pep is split into the three data sets sample1.pep (two sequences), sample2.pep (two sequences), and sample3.pep (one sequence). In real life, these files would, of course, contain many more sequences.*

*When splitting the data, note that the sequences in the ClustalW alignment come in the order indicated by the tree. To make a subset, reformat the ClustalW alignment in FASTA and edit it with a word processor.*

---

**Figure 3.8.3** Choosing subgroups with the help of a phylogenetic tree.
**Make the individual sub-multiple alignments**

3. Make default T-Coffee alignments:

```bash
t_coffee sample1.pep
t_coffee sample2.pep
```

**Make a sequence-to-profile alignment**

4. Run the profile alignment command line:

```bash
t_coffee sample3.pep
-profile=sample1.aln,sample2.aln
```

T-Coffee can align one or more sequence(s) to one or more pre-existing multiple sequence alignment(s). When T-Coffee is used this way, the input multiple alignments (sample1.aln, sample2.aln) are not recomputed, but rather treated as simple sequences and aligned to one another.

Up to 200 multiple alignments can be provided with the -profile option. In the current implementation, T-Coffee turns every profile into a consensus sequence, treats these consensus sequences, as ordinary sequences and aligns them using the standard procedure. Once the alignment is finished, the program threads the original multiple sequence alignment onto the aligned consensus sequences. Replacing a multiple alignment with its consensus requires a high level of similarity within the alignment (50% average identity or more) so that it yields a meaningful consensus. This strategy makes it possible to mix sequences and profiles in any proportion. For consistency with ClustalW, T-Coffee also supports the -profile1 and -profile2 flags of ClustalW.

Work is presently underway so that T-Coffee can make genuine profile-to-profile rather than the consensus-based comparison. This future version of T-Coffee should also provide an automated procedure for the prealignment clustering step.

5. Consider evaluating the alignment on the fly with --output=score_pdf (see Support Protocol 2).

6. When running multiple analyses on the same data set, consider exporting and reusing the library file by adding the flag --out_lib=<your_library_name> to the command line (see Support Protocol 3 and Commentary).

---

**Support Protocol 1**

**Reformatting Sequences, Alignments, Structures, and Libraries**

A major source of confusion when working on a multiple sequence alignment project is the variety of formats. The T-Coffee package provides several useful facilities that render data gathering less tedious than normal. When it comes to reformatting, T-Coffee mirrors all the functionalities of ClustalW with a few extra ones, such as structure fetching/reformatting utilities, file merging, and sequence reconciliation.

**Converting an alignment from MSF to FASTA**

Run a command using the -convert flag

```bash
t_coffee -in=Asample1.msf
-output= fasta_aln
-quiet=stdout
-convert
```

This command turns an MSF alignment into a FASTA alignment. Note that the extensions of the files have no influence on their parsing. The format recognition is automatic (supported formats and their identifiers are indicated in Table 3.8.1). If it fails, the authors recommend that a third-party software package (such as FMTseq or Readseq) be used to reformat the data set to FASTA format (FASTA is the most robust format and the easiest to parse for most packages).
Gather the content of several files into a FASTA file

t_coffee -in=Asample1.msf,Pstruc7.pdb,Llib4.tc_lib
-output= fasta_seq
-quiet=stdout
-convert

This command prompts T-Coffee to concatenate several files from different formats into
a single file without duplicate sequences (i.e., all the sequences have different names). The
underlying process is the following.

T-Coffee extracts the sequences from each file. The converter letter (L, A, S, P) makes it
possible for the program to properly parse the associated file (see Table 3.8.1 and 3.8.2
for a complete list).

1. If needed, T-Coffee fetches the PDB structures (provided that wget is installed on the
system and there is an Internet connection).
2. The sequences are pulled together into a file.
3. If two sequences have the same name, but different sequences, T-Coffee attempts a
reconciliation. Duplicates (sequences with the same name) are removed or reconciled
(see below).
4. The complete set is used for making the multiple sequence alignment.

The following section shows how T-Coffee can automatically fetch the structures in which
one is interested. This facility makes it possible to simulate a local installation of PDB.

Converting a PDB structure into a sequence
To fetch the portion of structure in which one is interested, type a command such as the
following:

t_coffee -in=P1bbt2_25_50 P1ayml
-convert
-output= FASTA_seq
-quiet=stdout

This command instructs T-Coffee to start looking for the file 1bbt_25_50 (or the file
1bbt_25_50.pdb) that should contain a PDB structure (as indicated by the converter
P). If this file is not in the current directory, T-Coffee uses extract_from_pdb (a
script that comes with the default T-Coffee distribution) in order to fetch the chain number
2 of the structure 1bbt directly from the PDB database (http://www.rcsb.org/pdb). T-Cof-
fee proceeds by extracting the segment 25 to 50 of this PDB chain, writing a file named
1bbt2_25_50.pdb in the process. If this file remains in the current directory, T-Coffee
will automatically reuse its content when appropriate.

When fetching structures, T-Coffee never overwrites existing files in the home directory.
This way, important data cannot be lost. While aligning or converting, sequences and
structures can be mixed in any suitable proportion.

Reconciling inconsistent sequences
Combine the content of two files by typing a command such as the following:

t_coffee
-in=sample5.pep,sample6.pep
-output=Fasta_seq
-quiet=stdout
-run_name=combined
-convert
Reconciliate sequence A in sample5.pep and sample6.pep

When combining sequence data sets, T-Coffee considers two sequences with the same name as representing the same sequence. In order to check this assumption, T-Coffee compares the two sequences to verify consistency. If an inconsistency occurs, T-Coffee attempts a reconciliation process by aligning the two sequences and determining a consensus. For instance, sequence A varies a little between sample5.pep and sample6.pep:

```
sample5.pep:A   ———AAAAAACAAA-EAAAAAAAAACCCCC
sample6.pep:A   DDDDDAAAAAAA-AAAAA-AAAAAAAA——-
```

After reconciliation, sequence A looks like this:

```
A   DDDDDAAAAAAAACAAAAEAAAAAAAAACCCCC
```

The rules governing the computation of the reconciliation are fairly straightforward: the first sequence is the master and it is used to resolve ambiguities arising from a mismatch. The consensus is derived by replacing every gap with the associated residues.

This strategy is useful when one deals with data sets that have been obtained through database searches (where the N and the C terminus of various sequences may not be in perfect agreement with one another); it is also useful when dealing with structures where special residues may disappear or be replaced by various reformatting programs. Of course, this strategy is prone to fail when the two sequences with the same name only have a very short overlap, a large number of insertions/deletions, or different numbers of repeated elements.

EVALUATING THE LOCAL SCORE OF AN ALIGNMENT

Assessing the biological relevance of a multiple sequence alignment is an important and difficult question. Unfortunately, \( E \)-values and other statistical means of evaluation are not yet as developed for multiple sequence alignments as they are for pairwise local alignment analysis. As a consequence, the estimation of the biological quality of an alignment still relies heavily on visual interpretation. This interpretation is mostly a matter of identifying well conserved blocks and meaningful patterns of hydrophobic/hydrophilic residues. To assist this tedious work, sophisticated color schemes based on sequence conservation have been developed and are part of most multiple sequence alignment editors such as Jalview or Seaview.

T-Coffee provides a new evaluation scheme that is not directly based on sequence conservation but rather on the local consistency between the aligned residues and the T-Coffee default library (made of local and global pairwise alignments). A high score indicates a good agreement with the library. This consistency score can be computed for the entire alignment or for individual residues, thus yielding a local measure named the CORE (Consistency of Overall Residue Evaluation) index.

The CORE index measures the overall agreement between the way a residue is aligned within the multiple sequence alignment and the way this residue is aligned in the pairwise alignments that constitute the library. The CORE index computation is extensively described in Notredame and Abergel (2003), and its biological relevance was analyzed using the BaliBase reference alignment collection (Thompson et al., 1999). This work indicates that residues with a CORE index higher than 5 (on a scale of 0 to 9) are generally correctly aligned (as long as the multiple sequence alignment contains four or more sequences).
The most interesting property of the CORE index is the fact that it does not depend directly on sequence conservation. This makes it an ideal tool to distinguish between a poorly conserved region correctly aligned and a poorly conserved region incorrectly aligned.

**Necessary Resources**

**Hardware**

T-Coffee has been especially designed to run on Unix and Linux platforms. The program is distributed as C source code and Perl scripts, and needs to be compiled. Experience shows that it is easy to install on a wide range of platforms (e.g., Irix, Sun, or HP) by following the simple three-step procedure provided in the documentation. For Windows and Macintosh users, precompiled binaries are available from the authors, although these platforms are not officially supported.

A multithreaded version of T-Coffee that can take advantage of multiprocessor machines is under development but not yet available. This means that one does not get much benefit from running the program on a multi-processor machine. Linux farms are not very useful either if one wishes to increase the size of the data set that T-Coffee can handle. Given the choice, users should select well endowed single processor machines.

**Software**

*Environment and compilation:* The user is expected to have access to a common Unix/Linux environment. Standard programs, such as wget (designed for automated download of Web pages) should also be installed. For the compilation of the package a C compiler is required. The best cross-platform reproducibility of the results is assured using the Gnu C Compiler gcc, which is freely available ([http://www.gnu.org](http://www.gnu.org)) and is the default C compiler on most Linux distributions.

*T-Coffee package:* The T-Coffee package and its source code can be downloaded from [http://igs-server.cnrs-mrs.fr/Tcoffee/](http://igs-server.cnrs-mrs.fr/Tcoffee/). Read the license before installation. This package is free of charge for academic and other nonprofit users provided that they agree not to redistribute or modify the code. Commercial users are required to take a special license when using versions higher than 1.37. The Web server Tcoffee@igs is freely accessible for all academic and commercial users.

*ClustalW package:* To follow the steps below, the user also needs to have access to the ClustalW multiple sequence alignment package ([ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalW/; UNIT 2.3](ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalW/)).

**Files**

T-Coffee is able to read and write the most common file formats (see Support Protocol 1). Sequences may be provided using the FASTA ([APPENDIX 1B](APPENDIX 1B)), the MSF or the ALN sequence format, and an output can be produced in any of these formats. By default, the nature of the files that T-Coffee outputs is the same as that of ClustalW ([UNIT 2.3](UNIT 2.3)), including similar file extensions (.aln, .dnd).

Recognition of the file format is carried out automatically by T-Coffee. Note that file extensions (e.g., .pep, .FASTA) are not used, so that users are free to use any file extension that they choose.

**Requesting an evaluated multiple alignment**

1a. Use the -output flag to request an evaluated alignment.

The local evaluation of T-Coffee multiple alignments can easily be obtained by requesting evaluated alignments with the -output flag, using the following command:
Evaluate an existing alignment

1b. Evaluate the existing alignment, sample4.cw_aln, using the following command:

```bash
t_coffee -infile=sample4.cw_aln -output=score_html,score_ascii -quiet=stdout -out_lib=sample4.tc_lib -score
```

*Note the -score flag. It indicates that the alignment provided via -infile must not be recomputed, but simply evaluated, using a standard library that needs to be generated for this purpose. When using -score, T-Coffee runs quietly without any output, yet for the purpose of this protocol, the authors have set -quiet=stdout, so that the usual output appears on the screen.*
The -score flag makes it possible to evaluate an alignment generated with any multiple sequence alignment package. This computation can be expensive since it requires the compilation of a complete T-Coffee library (all the pairwise global alignments and all the local ones).

The -out_lib flag prompts T-Coffee to output its library so that it can be reused for further evaluation, thus saving time. For instance, in order to evaluate a second alignment of the same sequences, one can reuse the previous library (see Support Protocol 3).

2. Examine the output files.

sample4.aln is the standard ALN-formatted multiple sequence alignment that the user can view with any text editor.

sample4.score_html is a colored alignment in HTML format. It can be viewed with any Internet browser. On this alignment, residues appear with a background indicative of their CORE index. Blue residues have a CORE index of 0, while those with a red score have an index of 9. Evaluation is only reported for columns that contain at least two aligned residues (Fig. 3.8.4).

sample4.score_ps, a similar alignment in PostScript format, can be printed directly or be viewed using GhostView (http://www.cs.wisc.edu/~ghost/).

Figure 3.8.5 An ASCII representation of the CORE index: every residue is replaced with its CORE index.
Sample4.score_pdf is a similar alignment in PDF format that can be viewed on most platforms using Acrobat Reader. The production of this format requires the converter ps2pdf to be installed on the local system (this is standard on most Linux distributions).

Sample4.score_ascii is an ASCII representation in which the evaluated residues are replaced with their CORE index. In conjunction with third-party software (not yet available), this file makes it possible to automatically filter out some portions of one’s alignment (Fig. 3.8.5).

**Generating and Using T-Coffee Libraries**

This support protocol shows how it is possible to generate a user-defined library in order to impose specific constraints to a multiple sequence alignment, and how to use an existing library to evaluate the example alignment sample4.aln.

**Outputting the T-Coffee library during alignment computation**

Use the `-out_lib` flag:

```
t_coffee sample4.pep -out_lib
```

The flag `-out_lib` causes T-Coffee to output the default library in the file sample4.tc_lib. The user can change the name of the library file by complementing the flag with a file name:

```
-out_lib=<your_library_name>
```

Systematically saving libraries makes it easier to reproduce results. In addition, a library can be reused for further evaluation of a data set, thus saving time. For example, in order to evaluate a second alignment of the same sequences, one can reuse the previous library (see below).

**Turn a pre-existing alignment into a library**

Use the `-convert` flag:

```
t_coffee -in=Asample4.aln
-out_lib
-weight=1000
-convert
```

When turning an existing alignment into a library, the weight associated with every pair of residues is by default the average percent identity shared by the two aligned sequences they come from (the pairwise alignment considered for this measure is the one within the multiple sequence alignment). The `-weight` flag allows the user to change this behavior when building a library. The normal range of value for these weights is 0 to 100, but imposing a higher weight (typically 1000) makes it possible to overweight a specific alignment. It is also possible to request position specific weights. For instance,

```
-weight=winsim10
```

causes each pair of residues to be assigned a weight equal to the local percent identity between the two aligned sequences from which they come. This local percent identity is measured on a segment centered on the two aligned residues and spanning 10 residues on each side.

**Design a library with one’s own data**

T-Coffee libraries provide a convenient way to pass prespecified constraints to the alignment program. The best way to design such a library is to start from a template file such as the one provided in sample4.tc_lib. When working on a large project that involves many handmade alignments, saving these alignments as libraries is a convenient way to insure that this costly data can easily be injected into new projects.
The exact format specification can be found in the T-Coffee documentation (available from http://igs-server.cnrs-mrs.fr/Tcoffee/). Note that on the lines describing a constraint, the two last fields can be omitted. Constraint lines only need to contain:

\[ \text{<offset of residue 1> < offset of residue 2> <Weight of the pair>} \]

**Use an existing library to evaluate sample4.aln**

One can provide an existing library with the `-in` option (see Critical Parameters and Troubleshooting):

```
t_coffee -infile=sample4.aln
-in=sample4.tc_lib
```

The `-in` (not to be confused with `-infile`) option controls the library compilation in T-Coffee. It is one of the most important flags (see Critical Parameters and Troubleshooting). The letter `L` that precedes the library filename is a converter whose purpose is to inform T-Coffee that the corresponding file is a T-Coffee library (see Table 3.8.2).

**MIXING AND COMPARING ALIGNMENTS**

The variety of options when assembling a multiple sequence alignment is a major curse: there is always a choice to be made between several alternative methods. To make things worse, it is usually very hard to assess the relative biological merits of two alternative alignments. As a consequence, the user brave enough to evaluate several methods or parameter settings is often left with a puzzling collection of alternative alignments.

If one assumes that the most reliable portions of an alignment are less sensitive than the others to various protocol changes, one may overcome the complicated problem of choosing the correct alignment by computing a consensus alignment based on the combination of a several alternative alignments. T-Coffee provides all the tools needed for such a combination. For instance, in this alternate procedure, the authors show how it is possible to use ClustalW to generate several alignments using various gap penalties, and how to combine these alignments into a single one. This protocol also shows how using a slightly different approach makes it possible to compare two multiple sequence alignments.

The first example addresses the problem of finding the best set of gap penalties for computing a multiple sequence alignment. Parameterizing gap penalties is a complicated problem because it is not possible to determine in advance the optimal for a given family. When faced with this problem, one can generate several alignments with ClustalW, using different gap penalties, and combine these alignments into a consensus alignment with T-Coffee. In this protocol, the authors will also show how this consensus alignment can be visually compared to one of the original ClustalW alignments.

**Necessary Resources**

**Hardware**

T-Coffee has been especially designed to run on Unix and Linux platforms. The program is distributed as C source code and Perl scripts, and needs to be compiled. Experience shows that it is easy to install on a wide range of platforms (e.g., Irix, Sun, or HP) by following the simple three-step procedure provided in the documentation. For Windows and Macintosh users, precompiled binaries are available from the authors, although these platforms are not officially supported.
A multithreaded version of T-Coffee that can take advantage of multiprocessor machines is under development but not yet available. This means that one does not get much benefit from running the program on a multiprocessor machine. Linux farms are not very useful either if one wishes to increase the size of the data set that T-Coffee can handle. Given the choice, users should select well endowed single-processor machines.

Software

Environment and compilation: The user is expected to have access to a common Unix/Linux environment. Standard programs, such as wget (designed for automated download of Web pages) should also be installed. For the compilation of the package, a C compiler is required. The best cross-platform reproducibility of the results is assured using the Gnu C Compiler gcc, which is freely available (http://www.gnu.org) and is the default C compiler on most Linux distributions.

T-Coffee package: The T-Coffee package and its source code can be downloaded from: http://igs-server.cnrs-mrs.fr/Tcoffee/. Read the license before installation. This package is free of charge for academic and other nonprofit users provided that they agree not to redistribute or modify the code. Commercial users are required to take a special license when using versions higher than 1.37. The Web server Tcoffee@igs is freely accessible for all academic and commercial users.

ClustalW package: For the purpose of this tutorial, the user also needs to have access to the ClustalW multiple sequence alignment package (ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalW/; UNIT 2.3)

Files

T-Coffee is able to read and write the most common file formats (see Support Protocol 1). Sequences may be provided using the FASTA (APPENDIX 1B), the MSF, or the ALN sequence format, and an output can be produced in any of these formats. By default, the nature of the files that T-Coffee outputs is the same as that of ClustalW (UNIT 2.3), including similar file extensions (.aln, .dnd).

Recognition of the file format is carried out automatically by T-Coffee. Note that file extensions (e.g., .pep, or .FASTA) are not used, so that users are free to use any file extension that they choose.

Compute a consensus alignment

1. Generate a collection of alignments by varying the gap-opening penalty in ClustalW (UNIT 2.3). For example, run ClustalW three times with the following three commands:

   clustalw -infile=sample4.pep -gapopen=10 -outfile=s4.gop10.aln
   clustalw -infile=sample4.pep -gapopen=15 -outfile=s4.gop15.aln
   clustalw -infile=sample4.pep -gapopen=20 -outfile=s4.gop20.aln

   This yields three multiple sequence alignments that can now be combined into a single one by T-Coffee.

2. Combine the three alignments using T-Coffee.

   t_coffee -in=As4.gop10.aln,As4.gop15.aln,As4.gop20.aln
             -run_name=consensus
             -output=clustalw,score_html,score_pdf
             -evaluate=slow_tcoffee

   This command instructs T-Coffee to turn every alignment into a library, to combine these three libraries into a single one, and to use this library for the assembly of a new multiple alignment named consensus.aln. The -evaluate=slow_tcoffee parameter ensures that the color code of the score_html and score_pdf alignments reflect...
exactly the level of consistency between the consensus alignment and the alignments used to build the library.

3. Examine the three output files generated by the command in step 2:

   consensus.aln: a ClustalW-type alignment
   consensus.score_html: a colored output
   consensus.score_pdf: another colored output

In the colored output, residues are colored according to the consistency of their alignment with the collection of alignments provided with the -in flag. Red residues are aligned in a consistent fashion across all the provided alignments, while blue and green bits correspond to unstable regions where the alignment is very sensitive to the chosen gap penalties. These unstable portions are less likely to be correctly aligned.

**Compare two alignments**

4. Compare the consensus alignment with another alignment.

   t_coffee -infile=consensus.aln
   -in=As4.gop15.aln
   -evaluate_mode=slow_tcoffee
   -output=score_html,score_pdf
   -run_name=comparison
   -quiet=stdout
   -score

The library strategy is used here to compare two alignments and graphically display their regions of similarity, using the color-coded representation. For this purpose, one of the alignments is turned into a library (-in=As4.gop15.aln) and used to evaluate the alignment passed by the -infile argument. The -score flag ensures that the alignment is not re-computed but simply evaluated.

5. Examine the file comparison.score_pdf.

The color code indicates for each residue the proportion of the column in consensus.aln that is similar to the column containing this same residue in cw.go15.aln. Red columns are entirely identical in the two alignments.

**COMBINING SEQUENCES AND STRUCTURES**

For the last 20 years, the number of sequences kept in public databases has increased at an exponential pace. Structure databases have not yet experienced such an expansion and the gap between the number of known sequences and the number of known structures is rapidly widening. The effect of this structural backlog could be considerably lessened if efficient means of extrapolating structural knowledge onto sequences were available. The development of such methods is a major goal of structural bioinformatics.

In this protocol, the authors show how the quality of a multiple sequence alignment can be largely improved when structural information (e.g., in the form of PDB files) is available for one or more sequences. It is assumed that the main motivation of users is to learn more about the putative structure and function of a single target sequence, and to use this alignment as a starting point for the generation of a homology model.

The protocol below concerns yecP from *Escherichia coli*, a putative enzyme-coding gene. This sequence is used as a starting point to identify related sequences and related structures (structural targets). T-Coffee is then used for checking whether the putative targets can be aligned to the sequences with enough accuracy. Once a proper set of structures has been put together, T-Coffee is used to assemble a multiple sequence-struc-
ture alignment, using a mixture of sequence-sequence, sequence-structure, and structure-structure pairwise alignment methods.

**Necessary Resources**

**Hardware**

Structure comparison are expensive, therefore a powerful system (e.g., a PC with 2 GHz or higher processor) may come in handy for carrying out structure comparison analysis like the one presented here.

**Software**

To run the examples, the user needs the same packages as those required for Basic Protocol 1, as well as the SAP package (Taylor and Orengo, 1989; http://mathbio.nimr.mrc.ac.uk) and fugue_client, a small script that renders T-Coffee able to interact with the FUGUE server (Shi et al., 2001), a service dedicated to threading (http://www-cryst.bioc.cam.ac.uk/~fugue). fugue_client is bundled in the T-Coffee distribution.

**Collect the sequences and the structures**

1. Collect homologous sequences with BLAST (UNITS 3.3 & 3.4).

   The query sequence is in the file sample8.pep. Users can gather homologous sequences by doing a BLASTP search (UNIT 3.4) of this target against SwissProt, TrEMBL, or NR. Users should then select some of the sequences that have a bit-score higher than 25% of the bit-score of the perfect match of the target against itself (669 in the example below; Fig. 3.8.6). The sequences chosen are in the file sample9.pep.

2. Collect the structures with the FUGUE server.

   The user can identify structures potentially related to the sequence of interest using the FUGUE prfsearch server (http://www-cryst.bioc.cam.ac.uk/~fugue/prfsearch.html). This server threads the target sequence onto every known 3-D structure in PDB. It returns the identifier of the structures whose folds are the most likely to resemble the fold of the target (Fig. 3.8.7). Other threading servers can be used in place of FUGUE, in particular Metaserver (http://bioinfo.pl/Meta), which simultaneously queries a large number of servers (including FUGUE). The sequences of the chosen structures are in the file sample10.pep.

3. Gather the structures of interest from the PDB structure repository at RCSB (http://www.rcsb.org).

4. Verify the quality of the structures.

   The sequences read from the PDB entries can be significantly different from the original sequence from which these structures were derived: segments may be missing and residues may have been removed to help the crystallization process. These things will be better checked by the means of an alignment between each structure and its corresponding SwissProt/TREMBL sequence (see the DBREF entry in the PDB file for the corresponding accession number). For instance, to compare the structure 1kpi with its associated SwissProt entry, use the following command:

   t_coffee Q11196.seq P1kpi.pdb

   The resulting alignment appears on Figure 3.8.8. It shows that in the structure, some residues have been lost on the N-terminus and that a few cysteines have been mutated. This visual inspection is very important when working with structures, in order to insure the quality of the data, in particular when the user intends to use the alignment for homology-based model building.
Figure 3.8.6  Selecting putative homologs from a BLAST output.

<table>
<thead>
<tr>
<th>Profile Hit</th>
<th>PLEN</th>
<th>RAWS</th>
<th>RVN</th>
<th>ZSCORE</th>
<th>ZORI</th>
<th>AL</th>
</tr>
</thead>
<tbody>
<tr>
<td>hs1kpha</td>
<td>283</td>
<td>-141</td>
<td>282</td>
<td>17.70</td>
<td>20.67</td>
<td>00</td>
</tr>
<tr>
<td>hs1jsxa</td>
<td>193</td>
<td>-94</td>
<td>235</td>
<td>17.65</td>
<td>20.58</td>
<td>02</td>
</tr>
<tr>
<td>hs1lim8a</td>
<td>219</td>
<td>-143</td>
<td>214</td>
<td>17.25</td>
<td>20.21</td>
<td>00</td>
</tr>
<tr>
<td>hs1nkva</td>
<td>239</td>
<td>-103</td>
<td>216</td>
<td>17.06</td>
<td>19.99</td>
<td>00</td>
</tr>
<tr>
<td>hs1khha</td>
<td>193</td>
<td>-116</td>
<td>219</td>
<td>16.95</td>
<td>19.84</td>
<td>02</td>
</tr>
<tr>
<td>hs1338a</td>
<td>178</td>
<td>-132</td>
<td>218</td>
<td>16.86</td>
<td>19.79</td>
<td>02</td>
</tr>
<tr>
<td>CMAS</td>
<td>291</td>
<td>-160</td>
<td>267</td>
<td>15.92</td>
<td>18.90</td>
<td>00</td>
</tr>
<tr>
<td>hsdxvaa</td>
<td>292</td>
<td>-175</td>
<td>239</td>
<td>15.52</td>
<td>18.50</td>
<td>00</td>
</tr>
<tr>
<td>hs1dusa</td>
<td>192</td>
<td>-138</td>
<td>199</td>
<td>15.35</td>
<td>18.32</td>
<td>02</td>
</tr>
<tr>
<td>hs1jg1a</td>
<td>215</td>
<td>-164</td>
<td>205</td>
<td>14.98</td>
<td>17.97</td>
<td>02</td>
</tr>
</tbody>
</table>

Figure 3.8.7  Identifying potential structural homologs with the fugue server.
Identify the suitable structures

5. Build a multiple alignment with each structure and the homologous sequences.

The user must now decide which structures (gathered in step 2) are suitable for inclusion in the multiple sequence alignment. The best way to do this is to build an alignment that includes one of the structures and the other homologous sequences gathered in step 1. For instance, it is possible to check the suitability of 1kpi by running the following command:

```
t_coffee -in=sample10.pep,P1kpi.pdb Mslow_pair,Mfugue_pair
-output=clustalw,score_pdf
```

Since this involves aligning one structure and many sequences, T-Coffee must be run with the following mixture of methods:

- slow_pair for globally aligning pairs of sequences.
- fugue_pair, which uses fugue_client to align every possible sequence-structure couple (in this case, it aligns every sequence with the 1kpi structure).

6. Determine suitable structures by visual inspection of the CORE index.

Users should only keep structures that contain a few stretches of residues with a CORE index higher than 5 (orange or red stretches).

7. Rerun steps 5 and 6 and keep the best structures.

In this example, 1kpi and 1jsx.

Align sequences and structures

8. Make a multiple sequence-structure alignment.

The main specificity of the data set considered here is that it contains a mixture of sequences and structures. In order to take advantage of this, add to the T-Coffee library compilation a structure-structure comparison method (SAP; Taylor and Orengo, 1989), that delivers an alignment based on the structural information rather than the sequence information.

```
t_coffee -in=P1kpi.pdb,P1jsx.pdb,Ssample8.pep,Ssample9.pep
Msap_pair,Mfugue_pair,Mslow_pair
-output=score_pdf,clustal_aln
-outorder=input
```
The command above instructs T-Coffee to compile a library that contains a pairwise sequence alignment for every pair of sequence (including the sequences with a known structure such as 1kpi), a sequence-structure alignment computed by FUGUE for every possible pair of sequence-structure, and a pairwise sequence alignment based on a structure-structure comparison carried out by SAP on every potential pair of structures (1kpi versus 1jsx in this case).

While the SAP alignments are carried out on one’s local machine, the FUGUE alignments result from a submission to the FUGUE server. For this reason, users that are behind a firewall may have difficulties using this service. One may also want to refrain using it if the sequences are confidential. Finally, the fugue_pair method only works if the structures are part of the PDB database (i.e., a user cannot use unpublished structures). If this mode is not suitable one may replace the fugue_pair method with any suitable local method (see the Appendix at the end of this unit), or consider acquiring a FUGUE license in order to run the program locally.

9. Visualize the multiple sequence alignment.

Figure 3.8.9 shows a portion of the multiple alignment obtained when following this protocol. Among other things, this alignment informs investigators on the regions of their structures that are suitable for modeling and those that are not. For instance, the long green stretch in 1jsx is not suitable.

GUIDELINES FOR UNDERSTANDING RESULTS

Users of multiple sequence alignment methods must realize that the assembly of a good multiple sequence alignment remains one of the most challenging tasks in bioinformatics. For instance, the high-quality multiple sequence alignments on which protein profile domain collections are based (such as Pfam, Prosite, or Smart) all rely on the input of skilled biologists who are able to select sequences, align them, trim them, realign them, manually optimize their alignment, and so on.
There is no absolute way to estimate the quality of a multiple sequence alignment, since the equivalent of an $E$-value is not yet available for these models. This is why users will have to rely very much on their experience in order to decide whether a pattern of conservation is meaningful or not. Generally speaking, a good multiple sequence alignment is characterized by a few ungapped blocks separated by regions enriched in indels that normally correspond to the loops. In each of the blocks (which typically span 10 to 30 residues), one should expect to find a few highly conserved aromatic positions (W, F, Y) and possibly a few more degenerate positions.

Of course, the use of pre-established knowledge regarding the studied family can help a lot when looking at a multiple sequence alignment. For instance, an indication of an active site or a disulfide bridge should correlate with highly conserved positions within the alignment. The user should also pay attention to the positions that are highly conserved in one subgroup while being degenerate in another group. These kind of positions make good candidate for being the ones that support substrate specificity.

Although most methods make it possible to modify the parameters in order to change the aspect of an alignment, the authors’ experience is that the major source of tuning is the choice of the sequences one wishes to include in a data set: this is where the user will find maximum flexibility when building a family model. On this issue, it is interesting to point out that many of the methods for multiple alignments were initially developed in an era where the main question was how to align the few sequences available. In most cases, this question is no longer the right one. Over the last years, data have improved faster than methods, and in most situations, given the current database sizes and the wealth of complete genomes, the most relevant question has become how to choose the sequences to align among all the potential homologs. In this context, it is often possible to assemble a data set that perfectly suits the method one wishes to use. This is especially true in the case of a progressive strategy like the one in ClustalW or T-Coffee. During a progressive strategy, sequences are incorporated into the multiple sequence alignment one after the other. As a consequence, outlier sequences distantly related to the entire set can cause trouble and degrade the quality of the entire alignment. This problem can be elegantly solved by providing more intermediate sequences. These intermediates help integrate the outliers within the final alignment.

Along the same lines, a careful choice of the sequences can help in alleviating a complicated problem: the modeling of long insertions/deletions. These long gaps pose problems for most multiple sequence alignment strategies, and, in a data set, the sequences whose proper alignment require very long gaps should be removed. This may not be practical if these sequences also contribute important information. When this situation arises, one should keep in mind that multiple alignment methods which are able to use local information, such as Dialign (Morgenstern et al., 1998) or T-Coffee, have a decisive advantage over other more conventional methods such as ClustalW (Katoh et al., 2002; Lassmann and Sonnhammer, 2002).

Last but not least, the integration of structural information within multiple sequence alignments and the ability to seamlessly mix sequence and structure information has become a key issue. Structural data is widely available, and it makes little sense not to use this information when building multiple sequence alignment models. While T-Coffee is not the only multiple structure alignment program around, it is one of the few that are able to seamlessly combine sequences and structures, and most importantly, it is one of the most flexible.
COMMENTARY

Background Information
This unit presents the Unix-based multiple sequence alignment program T-Coffee. This program can be used as standard multiple sequence alignment program, in a fashion very similar to ClustalW. T-Coffee is also able to carry out new, original tasks such as combining multiple sequence alignments, comparing them visually, evaluating the quality of a multiple sequence alignment, and mixing sequences and structures.

In general, T-Coffee provides a very effective way to deal with heterogeneous collections of sequence alignments. As such, it is an ideal tool for the management of long-term projects that involve the coexistence of hand-made (or at least hand-refined) multiple sequence alignments along with automatically generated alignments. T-Coffee is also an ongoing project and is rapidly evolving. Version 2, currently under development, will include important new features, such as: better handling of profile-profile alignments (including the possibility to do local profile to profile alignments); an improved ability to deal with structures; and an improved library-based evaluation algorithm, meant to decrease the complexity of the algorithm.

Over the years, multiple sequence alignment methods have been established as a key component in biological sequence analysis techniques. Few procedures remain in bioinformatics that do not require, at one point or another, the assembly of a high-quality multiple sequence alignment. One could cite the identification of a protein signature such as a Prosite pattern (Bairoch et al., 1997); the building of a domain profile (or HMM) needed for identifying the most remote members of a protein family (Mulder et al., 2003); structure prediction (Jones, 1999); and phylogenetic analysis (Phillips et al., 2000). More recently, multiple sequence alignments have also proven useful to the characterization of nsSNPs (non-synonymous Single Nucleotide Polymorphisms; Ng and Henikoff, 2002; Ramensky et al., 2002).

Despite the importance of these applications, the design of an efficient and accurate algorithm for the assembly of multiple sequence alignments remains a difficult problem that has not yet been entirely solved. As a consequence, most of the available packages merely provide approximate solutions. For recent reviews on this problem, see Notredame (2002) and Duret and Abdeldaim (2000). Furthermore, none of these methods is consistently better than the others. For instance, systematic benchmarking experiments carried out with established collections of reference alignments have shown that each available package is better suited than the others to certain types of problems, but that none is always the best (Lassmann and Sonnhammer, 2002). This situation explains why, from one bioinformatics project to the next, the authors often use a different multiple sequence alignment package. Unfortunately it is usually difficult to determine which software or algorithm will work best on a given set of sequences, and the only way to address this problem is through a tedious trial-and-error process.

Library
The library is a collection of pairs of residues (e.g., Residue 25 of sequence 1 versus Residue 29 of sequence 50). These pairs all come associated with a weight and are not necessarily compatible with one another. The purpose of T-Coffee is to assemble a multiple sequence alignment composed of a high-scoring collection of pairs. The algorithm does this by re-evaluating the score of each pair according to its compatibility with the rest of the library. It then uses this new score as a position-specific substitution matrix in order to compute a ClustalW-like progressive alignment of the sequences.
When running T-Coffee, the user can either provide the program with a pre-computed library or let T-Coffee generate the library itself. This automatic library is obtained by computing the best global alignment associated with each pair of sequences and the ten best nonoverlapping local alignments associated with the same pairs. However, it is also a simple matter to use another method (e.g., SAP) within the T-Coffee framework.

**Critical Parameters and Troubleshooting**

*Understanding the -in flag*

- in is one of the most important flags in T-Coffee. The package uses the information provided via this flag to compile its library. -in introduces the data that one wishes to combine within the alignment as well as the methods used to combine these elements of data. The data consist either of sequences, structures, or constraints (i.e., precomputed libraries that give information about the sequences alignments).

The methods can either be pairwise or multiple alignment methods, but they can also be structural alignment methods (see Support Protocol 3). Each element that the -in flag passes to T-Coffee is preceded with a converter letter (A, L, P, etc.). This converter indicates how the element will be used by T-Coffee. Tables 3.8.1 and 3.8.2 give a complete list of the converters and of the way they can be used. For instance, the following command,

```
t_coffee -in=Ssample2.pep,Asample4.aln,mlalign_id_pair,mclustalw_aln
```

indicates that T-Coffee will pull together the sequences coming from the files sample2.pep and sample4.aln. The A converter before sample4.aln indicates that this alignment will also be turned into a library and integrated in the final library. Mlalign_id_pair is a pairwise comparison method that will be used to compare every pair of sequences, turn each of these comparisons into a small library, and add these libraries to the final T-Coffee library. Mclustalw_aln is a multiple alignment method that will be used to generate a multiple alignment of the data set. T-Coffee will turn this multiple alignment into a library and add this library to the final one. When the compilation of the library is finished, T-Coffee will then use it to assemble a multiple sequence alignment. Lalign_id_pair and clustalw_aln are two methods installed in T-Coffee. A list of all installed method is available in the T-Coffee documentation. If needed, users can also install their own methods (see the Appendix at the end of this unit).

When -in is omitted from the command line, T-Coffee uses a default setting:

```
-in= Lalign_id_pair,fast_pair
```

If -in appears on the command line, these default settings are reset and the only information used to compile the library is the one provided by the user.

**CPU and memory requirement**

In contrast with other sequence alignment methods, T-Coffee is not extremely sensitive to the choice of parameters. In fact the program has very few parameters, the main one being the way a cocktail of methods is assembled to create the T-Coffee library, via the -in flag.

The main limitation of T-Coffee is its important computational requirements: T-Coffee requires more memory and more time than ClustalW, its close relative. On average, this program is about N times slower than ClustalW. In practice, this means that T-Coffee should be used with relatively small test sets (less than 100 sequences). With larger data sets, the most efficient strategy involves “exploring” the data with ClustalW or with T-Coffee used in the fast approximate mode, in order to add or remove sequences. The next step would then be to produce a refined alignment with T-Coffee, using in the slow/accurate default mode.
**Structural analysis**

In terms of resource requirements, things get worse when using structure comparison methods. Structures are computationally more expensive to analyze than sequences and the results may not be entirely reproducible, especially when they depend on Web based resources (such as the fugue_pair method). This is the reason why users are encouraged to take full advantage of the T-Coffee buffering facilities that save any Web based result in an appropriately named file that T-Coffee can automatically reuse.

**Advanced Parameters**

**Accelerating T-Coffee**

*Changing the alignment mode.* In contrast with ClustalW, T-Coffee does not depend heavily on alignment parameters such as substitution matrix, gap opening, and extension penalties. The reason is that T-Coffee assembles its alignment using other methods whose parameters are set by default. Therefore, the best way to tune T-Coffee is to tune the constituting methods or to modify the cocktail of methods that it uses. Doing so is an advanced procedure described in the Appendix at the end of this unit.

Users should also note that although the -matrix, -gapopen, and -gapext flags are available, their effect is very different from what one would obtain using ClustalW. For instance, the following command,

```
t_coffee sample4.pep -matrix=pam250mt -gapopopen=10 -gapext=1
```

causes T-Coffee to replace its original algorithm with a simpler progressive algorithm similar to the one in ClustalV or PileUp. When used this way, T-Coffee runs faster but is much less accurate. The -matrix flag is responsible for this change of behavior. Under this protocol, the T-Coffee library is replaced with a standard substitution matrix. This makes it unsuitable for sequences that share less than 50% identity. If -gapopen and -gapext are omitted, the program uses appropriate default values.

**Optimizing memory usage**

*Parameterize T-Coffee for saving memory.* In its default mode, T-Coffee does a few trade-offs between speed and memory usage. If memory is the limiting factor in a project, the user can switch off time saving parameters, using the following command line.

```
t_coffee sample1.aln
   -in=Mlalign_id_pair,Mslow_pair
   -dp_mode=myers_miller_pair_wise
   -tree_mode=slow
```

These parameters specify the most memory-efficient mode for T-Coffee. The -in flag introduces the parameters for the compilation of the T-Coffee library, while the -dp_mode indicates the algorithm that T-Coffee uses for assembling its alignment. -tree_mode=slow indicates that the pairwise distance one needs to compute the tree are measured on pairwise alignments produced with the -dp_mode algorithm.

With these parameters T-Coffee requires an amount of memory that is roughly $O(N^2L)$, $N$ being the number of sequences and $L$ their length. This means that, in this mode, T-Coffee needs an amount of memory that increases proportionally with the length of the sequences, and proportionally with the square of the number of sequences. If this strategy fails, two alternatives remain: (1) decrease the data set size (remove sequences); or (2) split the data set into smaller subsets and proceed as in Basic Protocol 2.
Suggestions for Further Analysis

With a high-quality multiple sequence alignment in hands, users will be able to carry out a wide variety of tasks that include protein domain analysis (Chapter 2) and domain-based database searches (UNIT 2.5). The users will also be able to carry out phylogenetic reconstruction, using the resources introduced in Chapter 6.

Literature Cited


Key References

Notredame et al., 2000. See above.

The original paper describing the T-Coffee algorithm and the one that should be cited as a reference for T-Coffee.

Thompson et al., 1994. See above.

The publication describing ClustalW, the most widely used multiple sequence alignment package and a close relative of T-Coffee.
Plugging New Methods into T-Coffee

Basic Protocol 3 shows how desirable it can be to use special methods with T-Coffee when dealing with structures. Yet, if SAP or FUGUE are not the user’s methods of choice, or if the problem requires a different approach, it is possible to add new methods to the T-Coffee strategy.

The following section shows how to adapt the method one is interested in so that it can be used by T-Coffee. It then show hows to customize the method configuration file so that T-Coffee can use this new alignment method.

Preparing the Method

Users must make sure that their method is able to read FASTA sequences and that it can output either an alignment in ClustalW (ALN) format, an alignment in FASTA format, or a library in T-Coffee format. The best strategy is to encapsulate the pre-existing method inside a Perl script so that T-Coffee can run it with a command that would use the following syntax:

```
aln_script -INFILE FASTA_file -OUTFILE name
```

The script `aln_script` is provided along with the material of this unit. It is nothing more than a short set of Perl instructions that call ClustalW with a special set of parameters. When calling `aln_script`, T-Coffee passes the sequences via a FASTA file. These sequences retain their original names so that `aln_script` can use them to gather non–sequence based information stored in other accessory files (e.g., structure or annotation).

The next step shows how to write a configuration file that “describes” `aln_script` to T-Coffee.

Writing the Method Configuration File

The easiest procedure is to adapt the method file that comes along with the T-Coffee distribution (`doc/reference_method.method`). It contains the following settings.
EXECUTABLE S aln_script
ALN_MODE S multiple
OUT_MODE S aln
IN_FLAG S -INFILE&
OUT_FLAG S -OUTFILE&

"*" Indicates a comment line to ignore
"&" Indicates spaces and the
"S" Indicates that the argument is a string

This method file indicates that in order to run (EXECUTABLE S aln_script), T-Coffee should provide all the sequences at once as indicated by (ALN_MODE S multiple). It should then expect an alignment in return, as indicated with (OUT_MODE S aln). It is possible to adapt this configuration file to many specific needs. For instance, if aln_script outputs a T-Coffee library rather than an alignment, the method file must indicate:

OUT_MODE S list

or if one wants aln_script to run on every possible pair of sequences in the data set:

ALN_MODE S pairwise

Many other possibilities are listed in the documentation. aln_script must be an executable file that is either on the Unix path variable or specified with its full path name in reference_method.method (EXECUTABLE S pathname/executable).

reference_method.method must either be in the directory where T-Coffee runs, or in the directory indicated by the environment variable METHODS_4_TCOFFEE. With this configuration file ready, it is possible to add this method to the list of methods that T-Coffee uses to compile its library.

Using the New Method

The following command:

    t_coffee -in=Ssample.pep,Mreference_method.method

instructs T-Coffee to compile its library using the new method:

    reference_method.method
Finding Protein and Nucleotide Similarities with FASTA

The FASTA package of sequence comparison programs provides a comprehensive set of programs for protein and DNA sequence comparison. Sequence similarity searching is most effective when protein sequences or proteins and translated DNA sequences are compared. While the FASTA programs are not as fast as the BLAST programs (UNITS 3.3 & 3.4), they can be equally sensitive, and, because they calculate statistical parameters from the distribution of similarity scores calculated during the search, they can provide more accurate statistical estimates. Although DNA:DNA sequence comparison is far less sensitive than protein:protein or protein:translated DNA comparison, FASTA is more sensitive, with more accurate statistics, than BLASTN (UNIT 3.3). FASTA can be run interactively (see Basic Protocol 1) or from the command line (see Alternate Protocol). Using the network parallel programs in the FASTA package, it is possible to compare complete bacterial proteomes using the rigorous Smith-Waterman program on a network of 16 to 32 processors in less than a day (see Basic Protocol 2).

STRATEGIC PLANNING

In planning a FASTA or BLAST search—which entails choosing the program, the database, and the search parameters—it is important to remember the central goal of a sequence similarity search, which is to identify homologous sequences. Homologous sequences share a common ancestor, have similar three-dimensional structures, and often (but not always) have similar functions. When two sequences share statistically significant similarity—i.e., much more similarity than would be expected by chance—it is then inferred that they are homologous. Similarity searches are most sensitive when: (1) protein or translated protein sequences are compared, and (2) small, comprehensive databases are searched.

USING THE FASTA PROGRAMS INTERACTIVELY

The FASTA programs can be run in two ways: (1) interactively, where the program prompts for the name of the query sequence file, the library, and some other search parameters; or (2) in a noninteractive “command line” mode (see Alternate Protocol). Identical searches can be performed in either mode; the interactive mode is often useful as one becomes familiar with the program; once running the program has become routine, the “command line” mode is often more convenient, because one need not wait for the results of the search.

In interactive mode, the FASTA programs prompt for a limited amount of information (e.g. query sequence file, library, output file name), but there are many more options that can modify the standard FASTA behavior (e.g. setting scoring matrices and gap penalties). These options can be changed by providing additional options on the command line (see Alternate Protocol).

Necessary Resources

Hardware

A Windows 32-bit (Windows 95, 98, NT, 2000, XP), Macintosh (PowerPC), or Unix/Linux computer with at least 5 Mb of free disk space for the programs and 100 to 600 Mb of disk space for protein sequence databases. The FASTA programs require very little memory over that required by the computer’s operating system.
Software

The FASTA programs, installed and configured as described in Support Protocol 1.

Files


A FASTLIBS file that indicates where the sequence databases are located and their corresponding abbreviations (see Support Protocol 1 and see Critical Parameters).

A query protein sequence in FASTA format (APPENDIX IB); this example uses the SwissProt sequence C972_SOYBN and the PIR sequence CRHU2. These sequences can be downloaded from the NCBI Entrez Internet site (http://www.ncbi.nlm.nih.gov/Entrez/) by searching the protein database for C972_SOYBN or CRHU2. Copy the FASTA-formatted sequences into a file for the searches.

1. Having downloaded, compiled, and installed the FASTA programs (see Support Protocol 1), run the program by typing:

```
wrplab 11% fasta34
```

The following response is returned:

```
FASTA searches a protein or DNA sequence data bank
version 3.4t20 Aug 21, 2002
Please cite:
test sequence file name:
```

2. Enter the name of a file containing the query sequence:

```
test sequence file name: c972_soybn
```

3. Select the protein sequence database (see Guidelines for Understanding Results):

```
Choose sequence library:
A: NBRF PIR1 Annotated Protein Database (rel 59)
P: NBRF Protein database (complete)
D: NCBI/Blast PDB structure database
N: NCBI/Blast non-redundant (nr) proteins
Q: NCBI/Blast Swissprot
Y: Yeast proteins
W: C. elegans (Wormpep)
P: Drosophila proteome
E: E. coli proteome
H: Human IPI proteins
Enter library filename (e.g., prot.lib), letter (e.g., P) or a % followed by a list of letters (e.g., %PN): a
```

For this example, a is entered on the last line above.

Figure 3.9.1 (at right) A simple fasta34 search using a soybean cytochrome P450 (SwissProt C972_SOYBN) as a query sequence in a search of the PIR1 (annotated section of the PIR database). (A) Search summary and statistical output. The name and version of the program, and the name and length of the query sequence, are reported, as well as the name of the database searched. The histogram shows the distribution of similarity scores calculated by the program. The left column of numbers indicates a normalized similarity score; the center column reports the number of sequences obtaining that score, and the right column reports the number of sequences expected to obtain the score, based on the database size. (B) The list of top scoring sequences, with their raw similarity scores (opt), the normalized bit score, and the expectation value.
A

FASTA searches a protein or DNA sequence data bank
version 3.4.20 Jan 14, 2003
Please cite:
gli59185851:seq00921:2972_BoVKN, 576 aa
vs NRNPقول Annotated Protein Database (rel 59) Library

< 20 W 0 := nnnn 25 library sequences
24 1 0 :=
26 1 0 :=
30 20 20 :=
32 71 71 :=
34 192 210 :=
38 413 413 :=
38 736 712 :=
40 1009 91 :=
42 1101 1114 :=
44 1507 2233 :=
46 1349 1364 :=
48 1228 2028 :=
50 1209 1191 :=
52 1147 2077 :=
54 852 895 :=
57 751 814 :=
60 558 613 :=
62 475 497 :=
64 350 398 :=
66 332 313 :=
66 217 250 :=
68 199 199 :=
70 171 154 :=
72 157 151 :=
74 85 56 :=
76 58 70 :=
78 57 57 :=
80 41 41 :=
82 35 34 :=
84 29 27 :=
86 24 21 :=
88 17 16 :=
inset = represents 1 library sequences
90 16 13 :=
92 13 10 :=
94 10 7 :=
96 7 3 :=
98 4 4 :=
100 2 3 :=
102 4 3 :=
104 2 2 :=
106 2 2 :=
108 0 1 :=
110 0 1 :=
112 1 1 :=
114 1 1 :=
116 2 0 :=
118 2 2 :=
120 3 0 :=

5569001 residues in 14548 sequences

B

The best scores are:

<table>
<thead>
<tr>
<th>Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12472</td>
<td>Laurate omega-nyroxylase (EC 1.14.12.2)</td>
</tr>
<tr>
<td>12475</td>
<td>Cytochrome P450 2A6 - human</td>
</tr>
<tr>
<td>12476</td>
<td>Steroid 17a-hydroxylase (EC 1.14.99.9)</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 1A2 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Vitamin D 25-hydroxylase (EC 1.14.99.9)</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 1A2 - human</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2B1 - rat</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C8 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 1A2 - human</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C9 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C11 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C19 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C2 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C3 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C5 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C6 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C8 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C9 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C11 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C19 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C2 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C3 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C5 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C6 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C8 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C9 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C11 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C19 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C2 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C3 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C5 - rabbit</td>
</tr>
<tr>
<td>12472</td>
<td>Cytochrome P450 2C6 - rabbit</td>
</tr>
</tbody>
</table>

Finding Similarities and Inferring Homologies

3.9.3
4. Select the “ktup” (word size):

\texttt{ktup? (1 to 2)} [2]

Both FASTA and BLAST are “heuristic” algorithms; they use shortcuts that avoid examining all of the possible alignments between two protein or DNA sequences, focusing on those regions of the two sequences that are likely to produce a good alignment. FASTA and BLAST focus on good-quality regions by using a rapid computer method for finding exact matches. For protein sequences, FASTA uses a default “ktup,” or word size, of 2 to find regions in the two sequences where two residues in a row in the query sequence identically match two residues in the library sequence. With \texttt{ktup=1}, FASTA will look for single aligned identical residues. Thus, an alignment region shared by two sequences that was 50% identical, with every other residue matching, would be missed by FASTA with \texttt{ktup=2}, but not with \texttt{ktup=1}. In practice, homologous proteins typically share several regions with pairs of identical residues, so FASTA with \texttt{ktup=2} has no difficulty identifying homologs that share less than 25% identity. FASTA with \texttt{ktup=1} is more sensitive, but slower. For DNA sequences, FASTA uses \texttt{ktup=6} by default (6 identical nucleotides); searches with \texttt{ktup=3} are more sensitive. When searching with short oligopeptides or oligonucleotides, \texttt{ktup=1} should be used.

5. Examine the results (see Guidelines for Understanding Results).

Figure 3.9.1 shows the results of a “typical” FASTA sequence similarity search, using a soybean cytochrome P450 (SwissProt \texttt{c972_soybn}) as a query sequence in a search of the PIR1 (annotated) section of the PIR database. For a more complete discussion of the output, see Guidelines for Understanding Results. The PIR1 database was chosen because it is a small, comprehensive database. It is very incomplete, and should never be used for “real” similarity searches. It is used for the example here because the small set of results can be examined thoroughly.

\section*{USING THE COMMAND-LINE VERSION OF FASTA}

While many researchers run the FASTA programs in interactive mode, it is often more convenient to start a FASTA search (or a set of FASTA searches), save the results to a file, and come back when the search is complete. The FASTA programs can be run from the Unix, MacOSX, or MS-DOS/Windows command line with all the flexibility available in interactive mode.

\section*{Necessary Resources}

Hardware

A Windows 32-bit (Windows 95, 98, NT, 2000, XP), Macintosh (PowerPC), or Unix/Linux computer with at least 5 Mb of free disk space for the programs and 100 to 600 Mb of disk space for protein sequence databases. The FASTA programs require very little memory over that required by the computer’s operating system.

Software

The FASTA programs, installed and configured as described in Support Protocol 1

Files


A FASTLIBS file that indicates where the sequence databases are located and their corresponding abbreviations (see Support Protocol 1 and see Critical Parameters).

A query protein sequence in FASTA format (APPENDIX IB); in this example, the SwissProt sequence \texttt{C972_SOYBN} and the PIR sequence \texttt{CRHU2} will be used. These sequences can be downloaded from the NCBI Entrez Internet site (http://www.ncbi.nlm.nih.gov/Entrez/) by searching the protein database for \texttt{C972_SOYBN} or \texttt{CRHU2}. Copy the FASTA formatted sequences into a file for the searches.
To start this procedure, invoke the FASTA command line:

```
fasta34 -q c972_soybn a > c972_soybn.results
```

which performs exactly the same search as outlined Basic Protocol 1 above. More specifically, this query compares the sequence c972_soybn to the PIR1 library, where the FASTLIBS file assigns the abbreviation a to the PIR1 library (see Support Protocol 1 and see Critical Parameters).

The general form of a FASTA command-line run is:

```
fasta34 -opt1 -opt2 -opt3 query library ktup
```

Every FASTA command-line search will include the query and library information, and, in general, the option -q, which tells FASTA not to prompt for any additional information. The ktup argument is often left out (see step 4 of Basic Protocol 1). Other options are discussed in Critical Parameters.

On Unix, MacOSX, and Windows/MS-DOS systems, the results of a FASTA search are saved to a file by using “output redirection” by including > results.file at the end of the command. This is not part of the FASTA program, but can be used with any Unix command-line program.

Figure 3.9.1 shows the results of a “typical” FASTA sequence similarity search, using a soybean cytochrome P450 (SwissProt c972_soybn) as a query sequence in a search of the PIR1 (annotated) section of the PIR database. The PIR1 database was chosen because it is a small, comprehensive database. It is very incomplete, and should never be used for “real” similarity searches. It is used for the example here because the small set of results can be examined thoroughly.

**DOWNLOADING AND INSTALLING THE FASTA PROGRAMS**

Although the programs in the FASTA package are available on several Internet sites (see http://restools.sdsc.edu/biotools/biotools1.html for a list of sites), the Internet versions often do not provide the latest programs, and often lack specialized databases (e.g., human or yeast proteins) that can increase sensitivity. Some of the FASTA programs are also available in the GCG package, but the current GCG versions of these programs are more than 2 years out of date. Versions of the FASTA programs are available for Windows, Macintosh, and most Unix/Linux systems. Users who already have access to a local copy of the FASTA programs at their institution should skip to Basic Protocol 1.

**Necessary Resources**

**Hardware**

A Windows 32-bit (Windows 95, 98, NT, 2000, XP), Macintosh (PowerPC), or Unix/Linux computer with at least 5 Mb of free disk space for the programs and 100 to 600 Mb of disk space for protein sequence databases. The FASTA programs require very little memory over that required by the computer’s operating system.

**Software**

Current versions of the FASTA programs can be downloaded from ftp://ftp.virginia.edu/pub/fasta. Unix/Linux versions of the programs are provided as compressed “shell archives,” e.g., fasta3.shar.Z. Be sure to transfer the fasta3.shar.Z file in binary format. Windows and Macintosh versions of the programs are available in the win32_fasta and mac_fasta directories.
The latest versions of the similarity searching programs are in fasta3.zip (for Windows) or fasta3.sea.bin (for Macintosh OS 8.5 to 9.x). The programs come with complete source code, but recompiling should not be necessary on Windows and Macintosh machines, except those running Macintosh OSX, which is really a variant of Unix. The FASTA distribution file should be copied to a new directory for installation.

**Files**

To verify that the program is installed correctly, this protocol uses the mgstm1.aa and prot_test.lseg files included in the FASTA3 distribution file.

**Unpack files and compile if necessary**

**For Unix/Linux**

1a. Unpack the file using the shell command:

    zcat fasta3.shar.Z | sh

    or

    uncompress -c fasta3.shar.Z | sh

2a. Once the files have been unpacked, compile the FASTA programs.

    Makefiles are available for many standard Unix distributions, including DEC/Compaq/HP Tru64 (Makefile.alpha), Linux (Makefile.linux), Macintosh OSX (Makefile.os_x) Sun (Makefile.sun), SGI (Makefile.sgi) and IBM/AIX (Makefile.ibm). For example:

    make -f Makefile.linux all

    will compile all of the programs in the FASTA3 package.

**For Windows and Macintosh OS 8.5 to 9.x**

1b. Decompress the program files.

    The Windows (fasta3.zip) and Macintosh (for OS 9.x and lower) versions of the FASTA package use standard archive compression programs (WinZip, PKZIP for Windows, Stuffit for Macintosh). On those computers, opening the file will usually uncompress it. Macintosh users running MacOSX can use either the fasta.sea.bin (Mac OS9, OSX, Carbon) or the Unix command line fasta3.shar.Z version (see the instructions for unpacking and compiling Unix FASTA, above). If one does most of one’s work under MacOSX using the traditional Macintosh graphical interface, and transfer databases using an FTP program like Fetch, one will want to use the fasta3.sea.bin programs. If one uses the MacOSX terminal program and Unix command line interface, the fasta3.shar.Z programs will be more appropriate. The fasta3.sea.bin programs assume that query files and databases use the Macintosh OS 9 <carriage return> end-of-line character, while the fasta3.shar.Z programs use the Unix <line feed> character.

2b. If running Macintosh OSX (which is a variant of Unix running within a Macintosh shell), see step 2a regarding compilation.

**Install the programs**

On Windows and Macintosh computers, it is generally easiest to run the programs from the directory (folder) where they were unpacked. On Unix/Linux computers, particularly when several people use the computer, one usually copies the FASTA programs to a common directory for programs, e.g., /usr/local/bin/ or perhaps /seqprg/bin/. This directory should be in the executable search path.
3a. For Unix/Linux: Once the programs have been unpacked, test by typing (from a Unix/Linux shell prompt):

```bash
fasta34 -q mgstm1.aa prot_test.lseg
```

This is an example of running the program from the command line, with mgstm1.aa as the query sequence and prot_test.lseg as the database. The mgstm1.aa and prot_test.lseg files are included in the FASTA3 distribution file.

Every FASTA command-line search will include the query and library information, and, in general, the option `-q`, which tells FASTA not to prompt for any additional information.

3b. For Windows: First open a DOS command prompt window, and then type the same command as in step 3a.

3c. For MacOS 8.x or 9.x: Double click the FASTA3 program and enter the following command in the Arguments text box:

```bash
mgstm1.aa prot_test.lseg
```

A prompt will appear for the name of the output file (try mgstm1.out), and some additional information about the number of scores and alignments. To produce the mgstm1.out file without any prompting, follow steps 4 to 6.

**Configure FASTA to use different sequence databases (optional)**

4. Interactive mode (see Basic Protocol 1) prompts for the name of the query filename (mgstm1.aa in this example). The library can be entered as a file name (as in command-line mode), or by entering a letter to specify the library (e.g., A specifies the PIR1 annotated protein database; see Basic Protocol 1, step 3). Finally, the “ktup,” or word size to be used during the FASTA search, can be entered.

The fastlibs file specifies the list of libraries available in interactive mode. It also specifies the abbreviation for each library. A sample fastlibs file is included with the FASTA distribution (see Critical Parameters). Past experience suggests that configuring the fastlibs file is the most challenging step in installing the FASTA package. For details on configuring the fastlibs file, see Critical Parameters. The fastlibs file is optional, but it makes it much easier to use FASTA in the interactive mode, because the list of available databases is displayed (the fastlibs letter abbreviations can also be used on the command line).

5. Once the fastlibs file has been edited to reflect the correct library file names and library types, it is presented to the FASTA programs by setting the FASTLIBS environment variable (in Unix, environment variables are traditionally uppercase words). To set the FASTLIBS environment variable under Unix, use the command:

```bash
setenv FASTLIBS /seqdata/fastlibs (csh, tcsh)
```

or

```
FASTLIBS=/seqdata/fastlibs; export FASTLIBS (sh, ksh)
```

With Windows, one can either type:

```bash
set FASTLIBS=/seqdata/fastlibs
```

from the MS-DOS terminal window, or edit the login environment variables.

6. Once the FASTLIBS environment has been set and the fastlibs file edited to correctly point to the sequence library files (see Critical Parameters), it should be possible to type the command:

```bash
fasta34 -q mgstm1.aa a
```

where a is the abbreviation for one of the sequence libraries specified in the FASTLIBS file. The program will then search the specified database.

*The option `-q` tells FASTA not to prompt for any additional information.*
DOWNLOADING AND PREPARING SEQUENCE DATABASES

The FASTA programs currently work with seven different “flatfile” sequence database formats, NCBI BLAST binary formats, and a MySQL SQL query format (UNIT 9.2). The default database format is FASTA format (APPENDIX 1B); FASTA format sequence databases can be downloaded from the National Center for Biotechnology Information (ftp://ftp.ncbi.nih.gov/blast/db/) and the European Bioinformatics Institute (ftp://ftp.ebi.ac.uk/pub/databases).

If one is installing the FASTA packages for the first time, the SwissProt database can be downloaded in FASTA format:

uncompress the file (on unix):
uncompress swissprot.Z

and this database can be used for further testing.

Sequence database formats

Because many sequence databases are quite large, most researchers prefer not to keep multiple copies of a database. The current version of FASTA supports the BLAST/formatdb databases, the default format used by BLAST.

Removing low-complexity regions with PSEG

While FASTA statistical estimates are, in general, very accurate, they can be confused by query sequences that contain runs with reduced amino acid complexity, for example, proline-rich regions. The SEG and PSEG programs (Wootton and Federhen, 1993) can be used to remove these low-complexity regions, and the PSEG program can be used to convert the low-complexity regions to lowercase, so that, with the FASTA -S option (see Critical Parameters), they are ignored during the initial similarity scan.


Copy all the files into a new pseg directory, compile the program with make, and move it to the program directory (/seqprg/bin).

Finally, convert low-complexity regions to lowercase with the command:

pseg .swissprot -z 1 -q > swissprot.pseg

Here, .swissprot indicates the name of the file to be processed by PSEG and -z 1 -q indicates that the results should be written in FASTA format, with lowercase letters for low-complexity regions, to the file swissprot.1seg.

LARGE-SCALE SEQUENCE ANALYSIS

Most researchers search protein and DNA sequence databases one sequence at a time, searching for homologs of a newly isolated cDNA clone or newly identified protein sequence. However, for large-scale genome analysis, thousands or tens of thousands of sequences are compared to other genomes or sequence databases. For large-scale sequence comparison, the programs in the FASTA package can be configured to run in parallel on networks of Unix (Linux, MacOSX) workstations or “Beowulf” clusters of Linux computers. In addition, the FASTA programs can read sequences out of mySQL databases (UNIT 9.2), and write similarity scores and alignments in a very compact form for subsequent analysis.
The network parallel implementations of the FASTA programs are designed to compare libraries with hundreds of thousands of query sequences against another genome or more comprehensive sequence database. A “manager” program distributes the second database among a set of “worker” programs running on different machines on the network, and then sends each of the query sequences to the workers, collects the results, calculates the statistics, displays the results, and moves on to the next query. For modest-size networks, a good scale of performance is seen; a search that uses 16 “worker” processors will run almost 16 times as fast as a search running on one processor.

Analysis strategies change when analyzing the results from thousands of searches. It is no longer possible to evaluate, or even examine, each of the most tantalizing matches. When thousands of searches are done, results that are expected one time in one thousand happen regularly, so significance thresholds must be adjusted. Unfortunately, more conservative significance thresholds reduce the sensitivity of the search, and more homologs can be missed. Most importantly, it becomes necessary to use database and data mining tools to focus on the scientifically interesting matches.

**Necessary Resources**

**Hardware**

Clusters of Linux workstations, or Beowulf clusters, can provide a very cost-effective computing platform. Systems costing less than $50,000 (16 dual-processor 2 GHz Intel/Athlon machines with 1 Gb memory) are capable of meeting large-scale sequence comparison needs of all but the largest genome centers. To run the PVM/MPI parallel versions of the programs in the FASTA package, one will need accounts on several Unix/Linux/MacOSX computers that share access to the same directories.

**Software**

*PVM or MPI parallel environment*: There are two widely used environments for network parallel computing, PVM (Parallel Virtual Machine; [http://www.epm.ornl.gov/pvm/pvm_home.html](http://www.epm.ornl.gov/pvm/pvm_home.html)), and MPI (Message Passing Interface; implementations are available from [http://www-unix.mcs.anl.gov/mpi/mpich/](http://www-unix.mcs.anl.gov/mpi/mpich/) and [http://www.lam-mpi.org/](http://www.lam-mpi.org/)). Both environments have their proponents; FASTA supports both environments. Installing and testing PVM or MPI is more difficult than installing FASTA, so one should probably use the implementation that is best supported at one’s institution.

*PVM/MPI parallel versions of FASTA*: The PVM/MPI parallel versions of the FASTA programs are included with the standard FASTA distribution, and use identical code for the comparison, statistics, and alignment display functions. However, the program names are different, and the compilation process uses a different Makefile.

**Compile and install the programs**

1. Compile the programs.
   a. To compile the MPI versions of the programs, type:

   ```bash
   make -f Makefile.mpi4 mp34compfa
   ```

   or

   ```bash
   make -f Makefile.mpi4 all
   ```
b. The PVM versions of the programs come in two parts, the “manager” program (pv34compfa, pv34compfx, etc) and the worker program (c34.workfa, c34.workfx, etc). To build a PVM version of the fasta34 program, type:

```
make -f Makefile.pvm4 pv34compfa c34.workfa
```
or
```
make -f Makefile.pvm4 all
```

2. Install the programs.

The MPI and PVM versions of the programs must be located in a directory that is visible to all the machines in the network cluster, preferably with the same name. This is easy to do if one has the same home directory on all the machines, or if there is a common program directory. The most common reason why the MPI/PVM parallel programs do not run is that they cannot be loaded on one of the worker nodes. On the authors’ Beowulf cluster, there is a directory /seqprg/pvm3/bin/LINUX/ which contains both the PVM and MPI versions of the programs. The PVM or MPI programs must be copied to this directory, and this directory should be included in the execution search path.

Perform parallel sequence comparison with MPI and PVM

The MPI/PVM sequence comparison programs are designed to compare a query database with dozens to thousands of query sequences to a second database with thousands to millions of sequences (searches are more sensitive when the second database is smaller). For example, to find the proteins that are most highly conserved between humans and E. coli, one could compare the E. coli proteome against the human IPI protein library. Likewise, to reduce the amount of output, the expectation \( E() \) threshold should be reduced; by default it is 10.0 for proteins for both the FASTA and MPI/PVM versions; \(-E 0.01\) should ensure that all statistically significant alignments are displayed.

3a. For comparisons using MPI FASTA programs: In the MPICH environment, one can run MPI programs with the `mpirun` command (see Critical Parameters for a discussion of options):

```
mpirun -np 17 mp34compsw -S -E 0.01 ecoli.pep hum_ipi.lseg > ecoli_hum.sw
```

The MPI and PVM comparison programs typically use 1 processor to manage the search, and \( N \) workers to actually do the comparisons. Thus, to run on 16 worker processors, specify `-np 17`.

In some MPI configurations, one must specify the computers that will be used for the search in a “machine file” which simply lists the names of the machines one wishes to use, and, optionally, the number of processors per machine. To use a machine file, type:

```
mpirun -np 17 -machinefile my.machines mp34compfa -S -E 0.01 ...
```

3b. For comparisons using PVM FASTA programs: Running PVM parallel programs is a two-step process. First, the PVM control `pvmd` program must be started, which sets up initial communications among the computers to be used. Like the MPI machine file, `pvmd` needs a list of computers to use:

```
pvmd /seqprg/pvm3/my_machines.pvm
```

After the `pvmd` program has been started, test that the machines are available with the `pvmd` command, as illustrated in Figure 3.9.2. Once the PVM environment is initialized, the `pv34compsw` program is run with the following command (see Critical Parameters for a discussion of options):

```
pv34compsw -S -E 0.01 -d 0 -m 9c ecoli.pep hum_ipi.lseg > ecoli_hum.sw
```
The PVM FASTA programs read the last two digits of the “speed” information in the PVM configuration file to determine the number of processors available in the PVM environment. Thus, a speed of 1002 indicates that two processors are available, and, thus, that two worker jobs should be run on that machine. In heterogeneous environments, the processor number value can be increased to place more jobs on faster processors, and thus balance the load, but the value must be an integer.

GUIDELINES FOR UNDERSTANDING RESULTS

The natural question that every similarity search is designed to answer is: “Which proteins are homologous to my query sequence?” The answer can be deduced from the scores and statistical estimates in Figure 3.9.1B, the summary of the high-scoring alignments from the database search. Both the FASTA and BLAST programs provide several numbers for evaluating the quality of an alignment; in Figure 3.9.1B, FASTA provides the raw BLOSUM50 gapped similarity score (in the opt column), the bit score (which is comparable to a BLAST bit score and can be used to calculate the probability of a score using the equation above), and the expectation, or $E()$-value. In addition, the alignment information provides another measure of alignment quality, the percent identity (gapped or ungapped) and the alignment length.

**Using the $E()$-value to identify homologs**

In evaluating the search results, the expectation or $E()$ value is the most reliable and sensitive indicator of likely sequence homology. For protein:protein alignments, if the $E()$-value is less than $10^{-6}$, the sequences are almost certainly homologous. Sequences with $E()$-values $< 10^{-3}$ are almost always homologous as well, but in these cases, one must ensure that the statistical estimates are accurate (see below). Indeed, in most cases, sequences with $E() < 0.01$ are homologous. It is important to remember that the $E()$ value simply reports the number of times a similarity score is expected by chance. Since there will be a highest-scoring unrelated sequence in every search of a comprehensive database, the $E()$ value for the highest-scoring unrelated sequence (the highest-scoring potential false positive) will be approximately equal to 1 (see Critical Parameters, Selecting the Database). Of course, distantly related homologous sequences may also have $E() ~ 1$, or even higher; however, if the similarity is this weak, it cannot be distinguished from a similarity produced by chance. A similarity score with $E() < 0.01$ or $E() < 0.001$ simply says that this score should occur by chance once in 100 or once in 1000 database searches.
As noted in Critical Parameters, the $E$ value depends on the database size; thus, in Figure 3.9.1, $E(14548)$ is shown, because 14,548 sequence alignment scores were examined to find the best alignments.

**Evaluating statistical estimates**

The simple observation that the highest-scoring unrelated sequence should have an $E$ value near 1.0 provides a simple strategy for evaluating the quality of the statistical estimates provided by FASTA (or BLAST)—examining the $E$-value of the highest-scoring candidate unrelated sequence. Of course, it is impossible to know for certain whether a sequence is unrelated (significant similarity implies homology, but not the inverse), but it is possible to do additional searches to test whether a sequence is likely to be unrelated. It is clear that the query sequence is related to a variety of cytochrome P450s, which are also mono-oxygenases; there are more than a dozen family members with statistically significant similarity in this small database [the more comprehensive SwissProt database has almost 400 statistically significant homologs, i.e., $E() < 0.001$]. To confirm the accuracy of the statistical estimates, however, it is the highest-scoring nonhomolog that is sought.

In Figure 3.9.1B, the strongest candidate nonhomologs are CRHU2, carbonate dehydratase, and HMNZHA, measles virus hemagglutinin. To test whether either of these is a cytochrome P450 homolog, another search can be performed, perhaps against a larger database, to see whether CRHU2 or HMNZHA share significant similarity with any cytochrome P450. Figure 3.9.3 shows a partial list of the high scores from CRHU2 search of the SwissProt protein database; all of the statistically significant alignments match carbonic anhydrase or some protein tyrosine kinases that contain a carbonic anhydrase domain; none of the more than 600 cytochrome P450s in the SwissProt database share significant similarity with CRHU2. Thus, CRHU2 is almost certainly nonhomologous, and the highest scoring nonhomolog for the original soybean cytochrome P450 search has $E() < 0.66$, as expected for accurate statistical estimates.

**Evaluating statistics using shuffled sequences**

The FASTA programs estimate the statistical significance of an alignment by examining the distribution of “unrelated” sequence alignment scores calculated during the search. While the FASTA program does not know a priori which sequences are “unrelated,” it assumes that, in a comprehensive database search, fewer than 10% of the sequences in the database will be related and has several strategies to exclude those sequences from the statistics calculation. Thus, one can interpret the $E$-value as a measure of how often the query sequence would match a sequence like those in the database by chance.

Sometimes, however, the query sequence is different from most of the sequences in the database due to sequence composition, or some other sequence ordering peculiarity, and the search finds a sequence in the database that also shares this peculiar property, but is not really homologous. For example, one could imagine that a membrane protein with strongly biased amino acid composition might find other membrane proteins with marginally significant sequence similarity, not because they are homologous, but because they have a high fraction of hydrophobic amino acids (in practice, statistically significant matches to nonhomologous membrane proteins never occur, but the high-scoring nonsignificant matches are often other membrane proteins).

For these cases, the FASTA program provides two programs for evaluating sequence similarity significance using shuffled protein sequences. prss compares a query sequence to a single “library” sequence, calculating an optimal Smith-Waterman alignment score, and then shuffles the “library” sequence 200 to 1000 times, producing 200 to 1000...
new random sequences with the same length and amino acid composition. prss can also use a “window” shuffling mode, which preserves the local amino acid composition within a 10- to 20-residue window when producing the random sequences. Smith-Waterman alignment scores are then calculated for each of the shuffled sequences, and the distribution of these scores is used to estimate the statistical significance of the original unshuffled sequence. prss can be used for either protein:protein or DNA:DNA comparison, though the shuffling strategy does not preserve the higher-order statistical properties of DNA sequences, and is thus less reliable. For example, to test whether the apparent similarity between the soybean cytochrome P450 C972_SOYBN and JH0659, cholesterol monooxygenase, is supported by a shuffled sequence analysis, one would extract the JH0659 sequence from the database as jh0659 and type:

```
prss34 -q c972_soybn jh0659 1000
```

prss34 would then calculate the Smith-Waterman score for an alignment of c972_soybn with jh0659, and then shuffle the jh0659 sequence 1000 times and calculate the statistical significance of the unshuffled score from the distribution of scores from the shuffled sequence comparisons.

In addition, the prfx program can be used to estimate the significance of DNA:protein alignments using the FASTX program. DNA:protein alignment statistics can be misleading because out-of-frame translations often produce low-complexity regions that match low-complexity domains in the sequence databases. While low-complexity regions can often be masked by pseg, occasionally a domain slips through, producing an apparently significant match. prfx can identify these pathological cases; it substantially improves statistical reliability some cases.

For “normal” soluble proteins, the statistical estimates from the original similarity search will match quite closely those from prss or prfx. But if there is some concern about the reliability of the significance estimate, the shuffling programs can provide an alternative estimate.
COMMENTS

Background Information

There are two widely used sets of programs for searching protein and DNA sequences, the BLAST package (UNITS 3.3 & 3.4) and the FASTA package. Both sets of programs do similar things; they compare a query sequence to a library of sequences, calculating tens of thousands to millions of similarity scores, and report back the library sequences that are most similar to the query. Most importantly, both BLAST and FASTA calculate the statistical significance of the alignment scores, so that investigators can judge whether an alignment score is likely to have occurred by chance. Without accurate statistical significance estimates, it is impossible to evaluate the scientific importance of an alignment score. Because there are so many sequences, matches that seem intuitively unlikely will often occur by chance—for example, a search of the nr database, containing 400 million residues, with a 300-residue query sequence, is expected to match 9 identical residues more than 50% of the time. In contrast, relatively low-identity alignments (<20% identical over 300 residues) can be very statistically significant, with \( E() < 10^{-6} \). One should always focus on the statistical significance, or expectation \( [E()] \) value, when evaluating whether two sequences are likely to be homologous.

The BLAST and FASTA packages have programs that perform many of the same functions (Table 3.9.1) for protein:protein, DNA:DNA, and translated protein:DNA comparison. FASTA has several programs for searching with short, ordered or unordered, noncontiguous peptide or DNA sequences (FASTS, FASTF, FASTM). While the type of query sequence and target database usually determine the program, one should search with protein sequences whenever possible. Protein sequence comparison is 5- to 10-fold more sensitive than DNA sequence comparison; it is routine to identify sequences that diverged more than a billion (plants/animals) or even two billion (prokaryotes/eukaryotes) years ago with protein or translated protein sequence comparisons; searches with DNA sequences rarely find significant matches in sequences that diverged more than 250 million years ago. FASTX, FASTY, TFASTX, and TFASTY can align DNA sequences with frameshifts, so even if open reading frames cannot be identified unambiguously because of sequencing errors, the translated DNA sequence can be correctly aligned with an homologous protein. Searching smaller databases also increases sensitivity. Now that a large number of fully sequenced prokaryotic, fungal, plant, and animal genomes are available, it is much more effective to search complete genomes from taxonomic neighbors than to search the comprehensive nr or sp-trembl databases, which contain 800,000 to 1,000,000 entries. Many proteins have homologs in \textit{E. coli} or \textit{Saccharomyces} that can be found by searching the proteome \( [E(5,000) < 10^{-3}] \) but would not have statistically significant scores in the context of nr or sp-trembl. Proteome data sets are available for all fully sequenced genomes.

Most researchers do similarity searches on the Internet, not on their local computers. Internet searches are much more convenient; one does not have to download the BLAST or FASTA packages, download sequence databases, reformat sequence databases, or keep the programs and databases up to date, because the Internet site does all this work. But Internet site searches are often slower and less flexible; by searching on one’s own computer, it is possible to tailor the search parameters, database, and output formats to meet one’s own research needs. The FASTA package includes an implementation of the rigorous Smith-Waterman algorithm, which is not available on many Internet sites.

Even the least expensive modern desktop computers have enough disk space to hold every protein database distribution (about 1 Gbyte) and most can also keep a complete copy of the GenBank DNA database distribution (~30 Gb). An optimal Smith-Waterman search (Ssearch; \textit{UNIT 3.10}) of a 220-residue sequence against the SwissProt protein sequence...
database (120,000 sequences) took 2 min on a 1.7-GHz Athlon and 4 min on a 400 MHz Macintosh; the authors routinely compare bacterial proteomes (4,000 sequences versus 4,000 sequences), with Ssearch using 8 to 16 CPUs in parallel, in a few hours.

**Critical Parameters**

**Selecting the correct program**
The FASTA package provides programs for searching protein, DNA, or translated DNA sequence databases, using proteins, DNA, translated DNA, or short peptides as queries. Table 3.9.1 summarizes the programs that should be used for various analysis problems, and corresponding programs, if any, in the BLAST package (UNIT 3.3 & 3.4). In general, if one has a protein sequence, one should use FASTA or Ssearch (a Smith-Waterman implementation; Smith and Waterman, 1981; UNIT 3.10). If one has a DNA query sequence that codes for protein, one should use FASTX (Pearson et al., 1997), which compares a DNA query to a protein database. For most researchers, FASTA (protein) and FASTX will meet 80% or more of search needs. In both cases a protein sequence database would be searched. Sometimes, it may be desirable to check whether a particular protein sequence is present in an unfinished (or incompletely annotated) genome; in this case, TFASTX, which compares a protein sequence to a DNA sequence database, would be used.

The FASTA package also provides some more specialized programs, particularly FASTS (Mackey et al., 2002), which is designed to search with a set of unordered oligopeptide (or DNA) sequences, and FASTM, which does the same search with an ordered set of oligopeptides. FASTS is designed to identify proteins from de novo tandem mass spectroscopy (MS/MS) sequence data. Three or four oligopeptides of length 4 to 6 are

### Table 3.9.1 Programs in the fasta34 Distribution

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
<th>Parallel version</th>
<th>BLAST equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>fasta</td>
<td>Compare proteins to a protein database; DNA to a DNA database</td>
<td>pv34commpfa, mp34commpfa</td>
<td>BLASTP, BLASTN</td>
</tr>
<tr>
<td>ssearch</td>
<td>Compare proteins to a protein database, using the rigorous Smith-Waterman algorithm (Smith and Waterman, 1981; UNIT 3.10)</td>
<td>pv34compsw, mp34compsw</td>
<td>BLASTP</td>
</tr>
<tr>
<td>fastx, fasty</td>
<td>Compare translated DNA sequences to a protein database, allowing frame-shifts. FASTY uses a more sophisticated, but slower, alignment algorithm.</td>
<td>pv34compfyx, mp34compfyx</td>
<td>BLASTX</td>
</tr>
<tr>
<td>tfastx, tfasty</td>
<td>Compare protein sequences to a translated DNA sequence database, allowing frameshifts</td>
<td>pv34comptfxfx, mp34comptfxfx</td>
<td>TBLASTN</td>
</tr>
<tr>
<td>fasts, tfasts</td>
<td>Compare an unordered set of oligopeptides to a protein database (fasts) or a translated DNA database (tfasts). Designed to identify proteins using MS/MS de novo sequencing. fasts can also be used for oligonucleotide searching.</td>
<td>pv34compfspfx, mp34compfspfx, pv34comptfsfx, mp34comptfsfx</td>
<td>—</td>
</tr>
<tr>
<td>fastm, tfastm</td>
<td>Same as fasts/tfasts, but the oligopeptides (or oligonucleotides) are ordered</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>fastf, tfastf</td>
<td>Compare amino acids from mixed-peptide sequencing to a protein (fastf) or translated DNA (tfastf) database</td>
<td>pv34comppff, mp34comppff</td>
<td>—</td>
</tr>
<tr>
<td>prss/prfx</td>
<td>Estimate statistical significance by comparing two sequences using the Smith-Waterman (prss) or FASTX (prfx) algorithm and then comparing the first sequence to 200 to 1000 shuffles of the second, to estimate statistical parameters</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**Finding Similarities and Inferring Homologies**

3.9.15
typically sufficient to identify, by similarity, sequences that diverged in the past $400 \times 10^6$ years. Thus, MS/MS peptides from a hamster or rabbit can be reliably identified by searching against the human proteome.

**Selecting the database**

The statistical significance, or expectation value, reported by a FASTA or BLAST search is the product of two terms: (1) the probability of the pairwise sequence similarity score, typically calculated using the extreme-value distribution, and (2) the size of the database searched. Both BLAST and FASTA (version 34 and later) report similarity scores in terms of bits; if an alignment of two 300-residue sequences has a bit score of 40, the probability of that pairwise alignment occurring by chance is:

\[
P(40 \text{ bits}) = m \times n \times 2^{(\text{-bit})} = 300 \times 300 \times 2^{-40} = 8.2 \times 10^{-8} \sim 10^{-7}
\]

While probability of $10^{-7}$ may seem significant, it must be corrected for the number of sequences that were examined to find the alignment. If the 40-bit alignment was found after a search of the NCBI NR database, which contains about 1.2 million protein sequences, then the expected number of times a 40-bit score would be seen by chance is:

\[
E(40) = P(40) \times D
\]

where \(D\) is the database size = $8.2 \times 10^{-8} \times 1.2 \times 10^6 = 0.1$

and is thus not statistically significant; a similarity as good or better is expected by chance once in every 10 database searches. In contrast, if one were searching for a homolog in *E. coli*, or many other bacteria, one could either search the bacterial proteome or the proteome of a related bacteria, in which case the expectation \([E() ]\) value for exactly the same alignment score would be:

\[
E(40) = P(40) \times D = 8.2 \times 10^{-8} \times 4000 = 3 \times 10^{-4}
\]

Thus, an alignment score that would be clearly statistically significant in a search of a bacterial proteome would not be significant when searching the nr database.

This relationship between database size, statistical significance, and the ability to infer homology is disconcerting to many researchers. If something is homologous, it is argued, it should be homologous regardless of the database in which it is found. This is true of course, but misses a fundamental asymmetry in similarity searching: sequences that share statistically significant similarity can be inferred to be homologous, but the inverse is not true. Nonsignificant similarity does not imply nonhomology; there are many examples of homologous proteins that do not share significant pairwise sequence similarity. The problem with searching large databases is the “noise” associated with the many additional opportunities to obtain a high score by chance.

In general, one should search the smallest comprehensive database that is likely to contain homologs to the protein of interest. For vertebrate sequences, this would be the human genome; for invertebrate sequences, *Drosophila* and *C. elegans* are available. By searching a group of eukaryotic genomes—e.g., human (40,000 proteins), *Drosophila* (13,000 proteins), *C. elegans* (18,000 proteins), *S. cerevisiae* (6,500 proteins), and *Arabidopsis* (25,000 proteins)—the database will be about the same size as the SwissProt database, but will be both more comprehensive and less redundant for eukaryotic proteins.

Thus, nr should probably be the last, rather than the first, database to search. The most effective strategy would be to search individual complete proteomes from organisms that are close to the query sequence, then a taxonomically deeper set, then SwissProt, then nr. In the example (Fig. 3.9.1), a very small sequence database is searched—the PIR1 annotated protein sequence database—solely for teaching purposes.
### Changing search options

The FASTA programs provide program options to modify the scoring matrix (-s matrix) and gap penalties (-f, -g) in the search, to exclude low-complexity regions (-S), to select sequences in a molecular weight range (-M), and to modify the default statistical methods (-z) and output formats (-m). Table 3.9.2 and Figure 3.9.4 show all the scoring, statistics, and output options. The options in Table 3.9.2 must be specified when the `fasta34` program is started, at the command line. However, they can also be used in “interactive mode” by including them on the command line. Thus:

```bash
fasta34 -q -S -s BL62
```

would run in interactive mode and prompt for the name of the sequence, the library, and the ktup, as in Basic Protocol 1.

All FASTA options begin with a minus sign and must come before the query file, library file, and ktup parameters. Thus:

```bash
fasta34 -q -S -s BL62 query.aa swissprot > results.file
```

is correct, while

```bash
fasta34 query.aa swissprot -q -S -s BL62 > results.file
```

will fail. If the FASTA programs are run from the command line, one will generally want to use the `-q` (quiet) option and save the results to a file, e.g.,

```bash
fasta34 -q query.aa swissprot > results.file
```
<table>
<thead>
<tr>
<th>Command Line Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-a</code></td>
<td>Show entire length of both sequences in alignment (fasta, Ssearch)</td>
</tr>
<tr>
<td><code>-A</code></td>
<td>Do full Smith-Waterman alignment for DNA (fasta)</td>
</tr>
<tr>
<td><code>-b #</code></td>
<td>Number of high scores to display</td>
</tr>
<tr>
<td><code>-c #</code></td>
<td>Length of sequence name labeling aligned sequences</td>
</tr>
<tr>
<td><code>-d #</code></td>
<td>Number of alignments to display</td>
</tr>
<tr>
<td><code>-D</code></td>
<td>Use DEBUG mode</td>
</tr>
<tr>
<td><code>-E #</code></td>
<td>Expectation threshold for displaying scores, alignments</td>
</tr>
<tr>
<td><code>-F #</code></td>
<td>Gap-open penalty (−10 default for proteins with default BLOSUM50 matrix)</td>
</tr>
<tr>
<td><code>-F</code></td>
<td>Expectation lower limit for displaying scores, alignments</td>
</tr>
<tr>
<td><code>-g #</code></td>
<td>Gap-extension penalty (−2 default for proteins with default BLOSUM50 matrix)</td>
</tr>
<tr>
<td><code>-h #</code></td>
<td>Frameshift between codon penalty (fastax, fasty, tfastx, tfasty)</td>
</tr>
<tr>
<td><code>-H</code></td>
<td>Omit histogram (fasta programs); include histogram (pvcomp/mpcomp programs)</td>
</tr>
<tr>
<td><code>-i</code></td>
<td>Search with reverse complement query only (fasta DNA, fastx, fasty)</td>
</tr>
<tr>
<td><code>-j</code></td>
<td>Frameshift within codon penalty (fasty, tfasty)</td>
</tr>
<tr>
<td><code>-J #:#</code></td>
<td>First query, last query (pvcomp/mpcomp)</td>
</tr>
<tr>
<td><code>-K #</code></td>
<td>Node buffer size (pvcomp/mpcomp)</td>
</tr>
<tr>
<td><code>-l file</code></td>
<td>FASTLIBS library</td>
</tr>
<tr>
<td><code>-L</code></td>
<td>Long sequence descriptions on alignments</td>
</tr>
<tr>
<td><code>-m #</code></td>
<td>Alignment display options</td>
</tr>
<tr>
<td><code>-M #:#</code></td>
<td>Library length range; only sequences within the residue range are considered</td>
</tr>
<tr>
<td><code>-n</code></td>
<td>DNA query</td>
</tr>
<tr>
<td><code>-N #</code></td>
<td>Library segment length (examine long DNA sequences in overlapping lengths of nucleotides of a particular number represented by #)</td>
</tr>
<tr>
<td><code>-o</code></td>
<td>Turn off band optimization (fasta)</td>
</tr>
<tr>
<td><code>-O file</code></td>
<td>Output file name (most useful with MacOS9)</td>
</tr>
<tr>
<td><code>-p</code></td>
<td>Query sequence is protein</td>
</tr>
<tr>
<td><code>-q/-Q</code></td>
<td>Quiet: do not prompt for any input</td>
</tr>
<tr>
<td><code>-r #/#</code></td>
<td>DNA match/mismatch scores (+5/−4 default)</td>
</tr>
<tr>
<td><code>-R file</code></td>
<td>Score results file (for debugging)</td>
</tr>
<tr>
<td><code>-s file</code></td>
<td>Scoring matrix: file name or BL50, BL62, BL80, P10, P20,P40,P120, P250</td>
</tr>
<tr>
<td><code>-S</code></td>
<td>Treat lowercase as low-complexity</td>
</tr>
<tr>
<td><code>-t #</code></td>
<td>NCBI genetic code translation table</td>
</tr>
<tr>
<td><code>-T #</code></td>
<td>Number of workers (pvcomp/mpcomp, threaded)</td>
</tr>
<tr>
<td><code>-v</code></td>
<td>Shuffle window size</td>
</tr>
<tr>
<td><code>-w #</code></td>
<td>Alignment display width (default 60)</td>
</tr>
<tr>
<td><code>-W</code></td>
<td>Alignment context (unaligned residues shown around alignment, default=30, fasta, Ssearch)</td>
</tr>
<tr>
<td><code>-x #</code></td>
<td>Score for match to X (protein) or N (DNA), default −1</td>
</tr>
<tr>
<td><code>-X #,#</code></td>
<td>“Virtual” coordinate of query sequence, library sequences</td>
</tr>
<tr>
<td><code>-z #</code></td>
<td>Statistical estimation strategy</td>
</tr>
<tr>
<td><code>-Z #</code></td>
<td>Effective database size</td>
</tr>
<tr>
<td><code>-1</code></td>
<td>Sort results by init1 (best single ungapped region)</td>
</tr>
<tr>
<td><code>-3</code></td>
<td>Search with forward strand only</td>
</tr>
</tbody>
</table>
The most commonly used option should be \(-S\), which causes the program to ignore low-complexity regions when searching suitably formatted databases (see Support Protocol 2). The next most common parameter change should be the gap-open penalty (\(-f\)). By default, the gap-open penalties for the FASTA programs are set as low as practical to produce local alignments. In some marginal cases \([10^{-3} < E() < 10^{-6}]\) statistical significance can be improved by raising the gap penalty (however, for the most distant relationships, this will typically lower the significance).

The \(-S\) option changes the default scoring matrix (BLOSUM50; \textit{UNIT 3.5}). As shown above, \(-S\) BL62 will do a search with the BLOSUM62 scoring matrix used by the BLASTP (\textit{UNIT 3.4}) program (although with different gap penalties, BLASTP uses \(-11/-1\) for gap open/extend; FASTA uses \(-7/-1\)). Other matrices include modern versions of the PAM10 (\(-S\ P10\) PAM20, PAM40, and PAM120 matrices (\(-S\ P20, -S P40, -S P120\)). Any scoring matrix can be used by providing a file of scores in the same format as the BLAST scoring matrix format. By default, each of the scoring matrices has a gap penalty that is most effective with that matrix (Reese and Pearson, 2002), but these penalties can be increased with the \(-f\) and \(-g\) options. In general however, the default scoring matrices and gap penalties are the most effective. If the alignment for a candidate homolog is short, a second search with the BLOSUM62 matrix might improve the statistical significance (perhaps using a higher than default gap penalty).

Alternate statistical parameter estimation routines are specified with the \(-z\) option. Again, the default \(-z 1\) is preferred for most cases, but there are two situations where an alternative might be more effective. The first relates to the fact that the statistical estimation routines assume that most of the sequences in the database are unrelated to the query. If searching a library of related proteins, then one must use the \(-z 11\) or \(-z 16\) option. In this case, the program shuffles each of the library sequences to produce a random sequence, calculates a similarity score, and uses the scores from the random sequences to estimate the statistical parameters. The second situation stems from the fact that, sometimes, sequence alignments have high scores because of shared amino acid composition biases. In general, this problem—"significant" scores for nonhomologous sequences because of composition bias—can be corrected by searching a database with low-complexity regions removed using the \texttt{pseg} program (see Support Protocol 2). However, in rare cases, the composition bias is distributed throughout the protein and cannot be removed by \texttt{pseg}. The \(-z 6\) statistical estimation option uses a maximum-likelihood strategy for estimating statistical parameters that includes a composition component. Alternatively, one could confirm the statistical significance using the \texttt{prss} program.

\textbf{Search options for large-scale comparison}

Large-scale sequence comparison can generate hundreds of megabytes of output data, so it is critical that: (1) every effort be made to reduce false-positive results, and (2) only essential information be captured. False positives (unrelated sequences with apparently statistically significant scores) can be reduced dramatically by including the \(-S\) option and searching sequence databases that have low-complexity regions indicated by lowercase letters with \texttt{pseg}. In general the statistical significance threshold should be lowered as well. For protein sequence comparisons, FASTA reports all alignment scores with \(E() < 10.0\); in a large-scale sequence comparison with thousands of queries, this would produce tens of thousands of nonsignificant scores. For a search with thousands of queries, an expectation threshold of 0.01 (\(-E\ 0.01\)) will reduce the number of false positives substantially. The command for this would be:

\begin{verbatim}
  pv34compfa -S -E 0.01 ecoli_prot.libhuman_ipi.lib > ecoli_human.results
\end{verbatim}
When comparing hundreds or thousands of queries to tens of thousands of sequences, it is rarely possible to look at individual histograms or alignments. By default, the MPI/PVM versions of the FASTA programs do not provide the histogram of the similarity score distribution; the histogram will be shown with the \(-\text{H}\) option (this is the opposite of the behavior with the standard FASTA programs). Recent versions of the FASTA package include the output format options \(-\text{m 9}\) and \(-\text{m 9c}\) which provide all of the information typically needed when evaluating an alignment as part of the initial list of high-scoring sequences. When using the \(-\text{m 9}\) or \(-\text{m 9c}\) options, all the alignment output can be excluded by using the \(-\text{d 0}\) option. The command for this compact representation of search results is:

```
pv34compfa -S -E 0.01 -m 9c -d 0 ecoli_prot.lib human_ipi.lib > eco_hum.res
```

The \(-\text{m 9}\) and \(-\text{m 9c}\) options provide all the critical information about an alignment with the list of high-scoring sequences in a very compact form (Fig. 3.9.5). In addition to the conventional library “hit” information, the identifier and description of the sequence, the raw score, the bit score, and the expectation (\(E\)-value), \(-\text{m 9}\) provides almost all the additional information provided in the actual sequence alignment, including the overall percent identity (\%_id), the percent identity omitting gaps (\%_gid), the optimal Smith-Waterman score (sw), the alignment length (alen), the coordinates of the start and end of the alignments in the query (an0, ax0) and library (an1, ax1) sequences, the start and end of the query (pn0, px0) and library (pn1, px1) sequences, the number of gaps in the query (gapq) and library (gapl) sequences, and the number of frameshifts (fs).

When the \(-\text{m 9c}\) output option is used, the alignment coordinate information contains an additional field, the “alignment code,” which summarizes the positions of the insertions.
Figure 3.9.6  Virtual sequence coordinates. (A) A query sequence that indicates, with the @C:51 token, that the beginning of the sequence should have the virtual coordinate 51. (B) A -m 9 coordinate output indicating that the alignment begins at residue 51 (virtual coordinate) rather than residue 1. (C) Virtual coordinate numbering in the alignment display.

and deletions in the alignment. Figure 3.9.5C shows a conventional FASTA sequence alignment; Figure 3.9.5D shows the -m 9c encoding of the alignment. =number indicates a run of matches or mismatches; +number indicates insertions in the query sequence, -number indicates deletions in the query sequences. For FASTX, TFASTX, FASTY, and TFASTY, \1 and /1 are used to indicate frameshifts. As discussed in Chapter 9, the percent identity, alignment length, and alignment coordinates, as well as the alignment code, can be captured and stored compactly in an alignment results database.

**Working with sequence coordinates**

Sequence similarity searches are more sensitive (and faster) when fewer sequences are compared. Thus, particularly when characterizing partially completed genomes, searching is carried out with the parts of a sequence that have not been characterized using other strategies. For example, one might use a gene-finding program (Chapter 4) or similarity search against a closely related genome in an initial characterization of an assembled “contig,” but then take the intragenic regions not assigned by the initial analysis and search a more comprehensive protein database. In these cases, sequence coordinate bookkeeping is critical; if the query sequence came from nucleotides 1001 to 1500 of a contig, then this information needs to be captured in the alignment. The functions in the FASTA program package that read sequence databases provide a simple mechanism to specify the “true” coordinate of a query or library sequence; the string @C:1001 indicates that the position of the first residue of the sequence is really from position 1001. Alternatively, one might have a library of protein domain sequences that used the @C: coordinate to record the location of the domain in the original protein sequence. Thus, Figure 3.9.6A shows a query sequence, GTM1_DEL50, derived from the GTM1_MOUSE sequence, but which is missing the N-terminal 50 amino acids. When GTM1_DEL50 is matched to the original GTM1_MOUSE and the @C:51 coordinate string is present, the alignment coordinates (Fig. 3.9.6B) and alignment (Fig. 3.9.6C) indicate the correct original coordinates. If the @C:51 string were missing from the sequence description, the alignment would be shown starting at residue 1, rather than residue 51.
The fastlibs file

Each line in the fastlibs file (also see Support Protocol 1 and Fig. 3.9.7) specifies: (a) a descriptive title of the library; (b) whether the library is protein (0) or DNA (1); (c) the abbreviation for the library; and (d) the library’s file name and library type. Thus, in the first line (Fig. 3.9.7):

```
NBRF PIR1 Annotated Protein Database (rel 59)$0A/seqdata/pir1.lseg
```

everything before the $ is the description. The 0 indicates that it is a protein library. After the 0, the library is assigned the abbreviation A. The remainder of the line specifies the file name.

Farther down the list, one of the DNA databases is:

```
GB120.0 Expressed Seq. Tags$1E@/seqdata/gb_est.nam
```

Again, everything prior to the $ is the description. The 1 indicates that it is a DNA library, which has been assigned the abbreviation E. In this case, the file name, @/seqdata/gb_est.nam, does not refer to an actual sequence file; it refers to a file of file names. The @ at the beginning of the file name indicates that the remainder of the file name is the file of file names. The file of file names (/seqdata/gb_est.nam) has the form:

```
</seqlib/genbank/
gbest1 . seq 1
gbest2 . seq 1
gbest3 . seq 1...
gbest101.seq 1
```

A line starting with < indicates the directory that should be used for other file names in the file. Each of the other lines provides the sequence database file name (gbest1.seq) and the database type.
Literature Cited


Contributed by William Pearson
University of Virginia School of Medicine
Charlottesville, Virginia
Ssearch utilizes the Smith and Waterman (1981) algorithm to search a group of sequences (usually a sequence data library such as GenBank or PIR) for local similarities. Sequences containing similar regions can imply a common biological relationship; the Ssearch code is typically used to search a sequence data library for sequences that may be homologous, or share common ancestry, with the query sequence.

The most crucial decision that one makes with any code for searching sequence data libraries is in selecting the rate at which nucleic acid or amino acid substitutions are expected to be found between the sequences being compared. While numerous ways to quantify rates of residue substitution are described in the literature, in practice, when the goal is to find homologous sequences, matrices based on an evolutionary model, such as the explicit PAM model (Dayhoff et al., 1978; Unit 3.5) or the implicit BLOSUM model (Henikoff and Henikoff, 1992; Unit 3.5) usually perform best. Because one does not know in advance the appropriate evolutionary distances between the query sequence and each sequence in the data library, multiple runs should be considered to provide complete coverage at all evolutionary distances (Altschul, 1991; Nicholas et al., 2000).

The second crucial choice that needs to be made is determining the magnitude of the gap penalties used for the analysis. Because alignments between relatively long ungapped segments usually dominate database search results, a relatively high penalty should be assessed for initiating a gap. It is important to select appropriate scoring matrix and gap penalty parameters, because the parameters help to define the measure of similarity that is needed to distinguish between a random and a nonrandom sequence.

Statistics presented by the program can help the researcher answer the question: What is the chance that, in searching a library of N random sequences, a score of X would be obtained? When this number is very low, a firm statistical argument can be presented implying that the two sequences are related. However, it is also not unusual to find, within the results from a search, sequences that are definitely related, but for which a firm statistical argument cannot be made. Sometimes the statistics can be improved for sequences within this “twilight zone” by repeating the search using a scoring matrix that is better tuned to the evolutionary divergence between the two sequences.

This protocol outlines the steps needed to run the Ssearch code contained in release 3.4 of the FASTA package (Unit 3.9).

**Necessary Resources**

**Hardware**

The Ssearch code is a resource-intensive code and thus generally requires a substantial computational platform with adequate CPU, memory, and disk space. However, these requirements will vary greatly depending on the usage of the code. For example, searching against compilations of known protein sequences, such as those that have been placed in the PIR database or Swiss-Prot databases, can be done on an inexpensive PC running Windows, Linux, or Macintosh OS. Performing regular searches of complete nucleic acid libraries with the code is a task best suited to higher-performance, multiprocessor machines.
Software

The Ssearch code is part of the FASTA package (UNIT 3.9) from Dr. William Pearson, which is available via anonymous FTP from ftp://ftp.virginia.edu/pub/fasta/. Dr. Pearson can be contacted at the Department of Biochemistry, School of Medicine, University of Virginia, Charlottesville, Va. 22908.

Files

Sequence files: The Ssearch code requires an input file that contains the query sequence in FASTA format (APPENDIX 1B). It also requires that one or more sequence data libraries (e.g., GenBank, NBRF-PIR, Swiss-Prot, or EMBL) be installed, or, in lieu of installing a sequence data library, simply having a set of sequences against which one wants to compare the query sequence in FASTA format (APPENDIX 1C).

Scoring matrix file: The Ssearch code enables one to use scoring matrices (UNIT 3.5) that are not internal to the code, such as the BLOSUM35 matrix (Fig. 3.10.1). The format of the matrix file should be in the same configuration as is acceptable by the BLAST (Altschul et. al., 1997; UNITS 3.3 & 3.4) family of programs.

A variety of compatible scoring matrices that can be used with the Ssearch program can be found at the NCBI FTP site (ftp://ftp.ncbi.nih.gov/blast/matrices). The Ssearch program does not require the use of an external scoring matrix file if one of the built-in scoring matrices is selected.
Prepare programs and files

1. Install the FASTA package (UNIT 3.9).

2. Select the data to be analyzed.

   Typically, one selects: (1) an input query sequence in FASTA format and (2) a sequence data library or a set of sequences that one wants to compare against it. In the example used in this protocol, the input query sequence is a 65-residue protein sequence called query.fasta while the data library used is a set of ~23,000 sequences in the file called library.fasta.

3. Select an appropriate scoring matrix and gap penalties.

   The default matrix used by the Ssearch program when proteins are being searched is the BLOSUM50 matrix (also see UNIT 3.5) with a penalty of 10 assigned to the first residue aligned with a gap, and a penalty of 2 assigned to each additional residue aligned with a gap. These default parameters are adequate if one is interested in performing only a single search through the data library. However, searching with two or three different matrices will generally give more complete coverage across the evolutionary spectrum and may help to resolve the issue of relatedness of sequences about which the user is not certain (Altschul, 1991).

   If two matrices are to be used, consider using the BLOSUM60 matrix along with the BLOSUM35 matrix, or the PAM120 matrix coupled with the PAM250 matrix. If searches with three different matrices are to be used, consider selecting the BLOSUM60, 40, and 30 matrices, or the PAM40, 120, and 240 matrices. Many of the matrices suggested above are not built into the program, but can be found online at the NCBI FTP site (ftp://ftp.ncbi.nih.gov/blast/matrices). The scoring matrices that are internal to the program are BLOSUM50, BLOSUM62, BLOSUM80, PAM250, PAM120, PAM40, PAM20, MDM10, MDM20, MDM30, and MDM40. MDM are modern PAM matrices (Jones et al., 1992).

   A good rule of thumb for setting the penalties for the first residue aligned with a gap when searching a sequence database is to use a value somewhere between about twice the absolute value of the largest negative value in the scoring matrix and the largest positive value in the scoring matrix. Subsequent residues aligned with a gap should be given a relatively low penalty such as 1 or 2 (Nicholas et al., 2000). Gap values in these ranges will generally give the best separation between the related and unrelated sequences (Pearson, 1995). In the example at the end of this protocol, the BLOSUM35 scoring matrix (a matrix that is not built into the program; called blosum35.matrix) is used, with the first residue aligned with a gap having a penalty of 12 and each subsequent residue aligned with a gap having a penalty of 1. To change these options, the –s (scoring matrix), –f (penalty for the first gap), and –g (penalty for additional gaps) command-line options (see step 7) are utilized.

4. Optional: If nucleic acid sequences are being compared, set a match/mismatch penalty.

   The Ssearch code allows one to specify a match/mismatch penalty. By default this value is set to 4. Again, as with proteins, utilizing a variety of scoring parameters may be beneficial (States et al., 1991).

5. Select the statistical method to be used for the analysis.

   Ssearch can analyze the significance of the results of a sequence database search with several different methods. The methods are all different approaches to fitting the observed scores to the extreme value distribution expected for such database searching scores. The default method developed by Pearson (1998) includes a correction for sequence lengths and is generally the preferred method. In cases where the sequence library being searched does not approximate a collection of random sequences, e.g., when it is a database for a single family of sequences, permutation statistics are preferred and can be produced by the shuffling method. The complexities of estimating the significance of the results of a sequence database search are explored in Spang and Vingron (1998) or Waterman and Vingron (1994). The default statistical method will be used in the examples presented in this protocol.
Figure 3.10.2  Histogram of scores: The leftmost numeric column is the score. The center numeric column is the number of times that score occurred within the data search. The rightmost numeric column indicates the number of times that the score was expected to occur in the data library search, after correcting for library sequence length.
6. Select the desired type of output. 

Search can produce output in a variety of formats. Usually, the default format is adequate for most uses. However, there are a variety of options that allow one to limit the amount of output produced by the program, including the number of sequence alignments (see Advanced Parameters, below). In general, one wants to see a list of sequences and their alignments for all sequences that are statistically significant. Sometimes, one may want to also show sequences that are not statistically significant but that are nevertheless related. In the example used below, the parameter is set so that the top 30 scoring sequences will be displayed.

**Run the program**

The commands and parameters below are given as examples.

7. To run the program using the options described above, enter the following command line:

```
Ssearch -s blossum35.matrix -f 12 -g 1
```

When prompted for the query sequence file name, enter:

```
query.fasta
```

When prompted for the name of the library file, enter:

```
library.fasta
```

At this point, the program displays a histogram of scores (Fig. 3.10.2), followed by a prompt to enter an output filename.

8. At the prompt for an output filename, enter the name of a file in which the results are to be stored:

```
querylibrary.results
```

Now, a prompt is displayed for how many scores will be seen. Generally, one wants to see all relevant high scoring sequences. At this prompt, enter:

```
30
```

At the “more scores” prompt, enter:

```
0
```

9. At the next, prompt, asking if one also wants to display the alignments, enter:

```
Y
```

At the “number of alignments” prompt, enter:

```
30
```

10. Examine the output file (`querylibrary.results`) for the results.

**GUIDELINES FOR UNDERSTANDING RESULTS**

The program’s output consists principally of a histogram of scores (Fig. 3.10.2) followed by a series of alignments between the query sequence and high-scoring sequences from the sequence data library. If search parameters include a sufficiently high open gap penalty, the histogram will follow an extreme value distribution (Altschul et al., 1994). The pattern followed by an extreme value distribution is that the majority of the scores received are centered on a moderately low score, with a few high-scoring outliers. It is for these high-scoring outliers that a statistical argument is presented whereby the sequences are related to the query sequence in some way. The statistics presented by the
The best scores are:  
1LREG2 chaperonin-10, chain G, fragment 2 - Mycob (66) 410 110.7 4.7e-26  
1LEFZ2 chaperonin-10, chain F, fragment 2 - Mycob (66) 410 110.7 4.7e-26  
1LREG2 chaperonin-10, chain E, fragment 2 - Mycob (66) 410 110.7 4.7e-26  
1LEFZ2 chaperonin-10, chain D, fragment 2 - Mycob (66) 410 110.7 4.7e-26  
1GONU groEL GroESL COMPLEX, chain S - Escherichia c (98) 158 45.0 4.1e-06  
1GONU groEL GroESL COMPLEX, chain U - Escherichia c (98) 158 45.0 4.1e-06  
1GONU groEL GroESL COMPLEX, chain O - Escherichia c (98) 158 45.0 4.1e-06  
1GONU groEL GroESL COMPLEX, chain R - Escherichia c (98) 158 45.0 4.1e-06  
1GONU groEL GroESL COMPLEX, chain P - Escherichia c (98) 158 45.0 4.1e-06  
1GONU groEL GroESL COMPLEX, chain T - Escherichia c (98) 158 45.0 4.1e-06  
1GONU groEL GroESL COMPLEX, chain Q - Escherichia c (98) 158 45.0 4.1e-06  
1SHSD small heat shock protein, chain E - Methano (116) 80 24.7 6.6  
1SHSD small heat shock protein, chain D - Methano (116) 80 24.7 6.6  
1SHSD small heat shock protein, chain B - Methano (116) 80 24.7 6.6  
1SHSD small heat shock protein, chain A - Methano (116) 80 24.7 6.6  
1SHSF small heat shock protein, chain F - Methano (116) 80 24.7 6.6  
1SHSF small heat shock protein, chain C - Methano (116) 80 24.7 6.6  
1SHSC small heat shock protein, chain G - Methano (116) 80 24.7 6.6  
1SHSF small heat shock protein, chain H - Methano (116) 80 24.7 6.6  
1LDCB lectin scafet precursor, chain C - bluebell (235) 86 25.5 7.3  
2CSC2 citrate (si)-synthase (EC 4.1.3.7) (with D- (140) 78 24.0 13  
5CSC2 citrate (si)-synthase (EC 4.1.3.7), chain (140) 78 24.0 13  
3CSC2 citrate (si)-synthase (EC 4.1.3.7) (with L- (140) 78 24.0 13  
5CSC2 citrate (si)-synthase (EC 4.1.3.7) (with ox (140) 78 24.0 13  
6CSC2 citrate (si)-synthase (EC 4.1.3.7) (with ci (140) 78 24.0 13  
4CSC2 citrate (si)-synthase (EC 4.1.3.7) (with D- (140) 78 24.0 13  
5CSCA3 citrate (si)-synthase (EC 4.1.3.7), chain (140) 78 24.0 13

Figure 3.10.3 List of high-scoring sequences: Information describing the library sequence is presented first, followed by the length of the sequence in parentheses. The next column contains the score of the comparison with the query sequence. The next column presents the score using an information-content approach. The final column contains the statistical estimate of the likelihood that the match has arisen by chance.

program are based on modeling the expected distribution of unrelated sequences, correcting for the effects caused by library sequence length (Pearson, 1998). This modeling allows accurate statistics to be gleaned for the high-scoring outliers.

Following the histogram is a list of the high-scoring outlier sequences (Fig. 3.10.3). Again, these are the sequences in the data library for which a statistical argument can be presented to infer that the sequences are related to the query sequence. Essentially the statistics presented are giving the researcher the answer to the following question: What is the chance that, in searching through a library of \( N \) random sequences, a score of \( X \) would be obtained? If the resultant number is very low, it is an indication of a very rare event, unlikely to be caused by chance, allowing the researcher to infer that the two sequences are related. If the resultant number is high, it is an indication that the event may be caused by chance. However, it is not unusual to find sequences in the output for which a firm statistical argument cannot be made, but which are in fact homologous with the query sequence. Thus, the statistical information presented should only be used as a guide, not as a substitute for one’s own biological experiments.

While one’s biological judgment can sometimes resolve the issue that some low-scoring sequences are homologous with the query sequence, occasionally there will be sequences presented in the output about which the researcher may not be so certain. If this is the case, it is important to remember that the results presented may not be accurate if the degree of divergence between the two sequences is not close to the degree of divergence
of the scoring matrix used within the search. Thus, one way to improve the accuracy of the results is to repeat the search with a scoring matrix more appropriate to the evolutionary distance of the two sequences being compared. Thus if an initial search made with a BLOSUM62 matrix shows a sequence in the results that one is not quite sure about, sometimes repeating the search with a more distant matrix, such as the BLOSUM45 or the PAM250 matrix, will improve the accuracy enough to make it possible to infer relatedness (UNIT 3.5).

This approach to identifying pairs of sequences as homologous has been pushed to its limits by Kann et al. (2000), who have proposed an iterative refinement for scoring matrices based on their performance in searching a curated database where all of the homologous pairs of sequences are known, thus improving their ability to detect remote homology between pairs of sequences. An alternative to changing the scoring scheme is changing the alignment model from the local, Smith-Waterman model to a global model (Webber and Barton, 2003) and examining the intersection set of sequences reported by both alignment models. This eliminates false and near-false positive results that are of greatly different length from the query sequence.

Other approaches to identifying sequence pairs showing a low level of similarity as homologous are related to the ideas underlying profile searching, in particular the idea that homologous sequences will show conservation of the sequence residues essential to their homologous structure and function. The most direct application of this idea is in the use of sequence profiles or hidden Markov models to model an homologous family of sequences. A wide variety of these techniques are discussed in Bork and Gibson (1996). An extreme case of these methods in which one examines the relationship between the query sequence and two remote potential homologs has become known as an intermediate sequence search (Weizhong et al., 2000) and may be worthwhile in some circumstances.

Masking the low-complexity regions in the query sequence (Altschul et al., 1994) is generally very helpful in identifying very distantly homologous relatives by removing false-positive matches to the low-complexity regions.

Homology searches are based on the following statistical rationale: if the observed similarity between two sequences is very unlikely to be observed by chance, then it may be concluded that the two sequences are homologous. The similarity is measured by an alignment score between the pairs of sequences. Unfortunately for present-day sequence databases, one will find alignment scores with random sequences with one’s query sequence that will be as high or higher than the alignment scores for the same query sequence with homologous sequences (Spang and Vingron, 2001). Additional database searches with scoring matrices designed to detect more remote homologies will often distinguish the sequences that are actually homologous from high-scoring matches to random sequences, because the actual homologs will show a marked improvement in score with the different scoring matrix (Altschul, 1991). Also, distant homologs are expected to still show examples of sequence motifs associated with the conserved functions of the homologous sequence family, while high-scoring matches with random sequences would not be expected to have these motifs.

The next items presented in the results are the alignments between the query sequence and the high-scoring library sequences. It is important to keep in mind that these sequence alignments are based on parameters selected to produce alignments, and corresponding alignment scores, that most effectively distinguish the sequences in the sequence library that are homologous to one’s query sequence from the nonhomologous, accidentally high-scoring sequences in the library. These parameters are not necessarily the parameters that will provide the alignment that is the best estimate of residue-by-residue sequence
homology for the two sequences when used with a global alignment algorithm such as the Needleman-Wunsch (1970) algorithm. Alignment parameters for database searching are selected to produce local alignments that have few gaps, and thus to reduce the number of similarly high-scoring alignments between random sequences which are likely to have many gaps (Vingron and Waterman, 1994).

**COMMENTARY**

**Background Information**

The Ssearch program utilizes the computational technique generally referred to as dynamic programming (Aho et al., 1983). Dynamic programming is a rigorous mathematical technique that works by breaking a problem into a table of partial solutions that are combined to give the optimal solution to the problem.

While optimal solutions are frequently desired, especially when creating an alignment between two similar sequences, finding the optimal solution requires significant computer resources. To search molecular biology databases with an unknown query sequence for sequences that may potentially be related to that query sequence, frequently an optimal solution is not necessary to guide the researcher along the correct path. Faster database-searching programs such as FASTA (Pearson and Lipman, 1988; *UNIT 3.9*) and BLAST (Altschul et al., 1997; *UNITS 3.3 & 3.4*) should generally be tried first. If an adequate solution is not found using those faster algorithms, then the Ssearch program may be run. In general, an Ssearch run will find a related sequence or two that the faster database searching programs were unable to find (Pearson, 1995; Agarwal and States, 1998).

**Critical Parameters and Troubleshooting**

When one selects parameters for any sequence database searching program, it is important to keep in mind that, conceptually, what one is trying to do is to separate the sequence data library into two distinct groups—a small group containing sequences that are related to the query sequence and a much larger group of sequences that are statistically random with respect to the query sequence. The parameters that are most influential on the separation are command-line options for which the user is not prompted when the program is run interactively. These parameters are:

- `-f`: This parameter is used to tell the program the penalty that the researcher wishes to use for the first residue in a gap. When searching a sequence data library for sequences that may be related to the query sequence, this value should be set relatively high. A rule of thumb is to select a value somewhere between about twice the absolute value of the largest negative value used in the selected scoring matrix and the largest positive value in the scoring matrix. A value in this range will generally give the best separation between the related and unrelated sequences. If the best local sequence alignments are desired, a value of about half this size is frequently more appropriate.

- `-g`: This parameter determines the penalty used by extending the gap an additional residue. For database searching, this value is almost always set to 1 or 2. For the best local sequence alignments, this value is usually set between 1 and the value used for the first residue in the gap.

- `-s [file]`: This option is used to specify the scoring matrix file or the built-in scoring matrix to be used. The scoring matrices that are internal to the program are: BLOSUM50 (*-s BL50*), BLOSUM62 (*-s BL62*), BLOSUM80 (*-s BL80*), PAM250 (*-s P250*), PAM120 (*-s P120*), PAM40 (*-s P40*), PAM20 (*-s P20*), MDM10 (*-s M10*), MDM20 (*-s M20*), MDM30 (*-s M30*), and MDM40 (*-s M40*).
For proteins, searching with three implicit or explicit evolutionary matrices will generally give good coverage across all evolutionary distances (e.g., BLOSUM60, 30, and 40, or PAM40, 120, or 250).

- r: This parameter is used to specify match/mismatch scores for DNA comparisons.

Occasionally, unexpected results can occur if the program incorrectly identifies the query sequence as a protein sequence when it is in fact a nucleic acid sequence or vice versa. If one suspects that this has happened, rerun the search with the appropriate command line switch:

- n: Tells the program that the query sequence is a nucleic acid sequence.

- p: Tells the program that the query sequence is an amino acid sequence.

**Advanced Parameters**

There are several other parameters used by the program, which fall into one of three general categories: parameters that control the input to the program, parameters that control the format of the output, and parameters that control the type of statistics produced. The majority of these parameters do not have a significant effect on the results received and are described in detail in the manual page that accompanies the software. However, there are a few advanced command-line options that merit further discussion. They are:

- S: This option treat lowercase characters in the query or library sequences as “low-complexity” residues. Low-complexity regions are sections of sequence that contain biased compositions such as a few types of residues or short periodic repeats. Low-complexity regions are problematic with database searches because they tend to have very different composition from the database as a whole, and, when included, can affect the accuracy of the statistics presented. Furthermore, these regions, when randomized, usually give alignment scores similar to the original sequence (Altschul et al., 1994). There are several good programs available that can be used to detect low-complexity regions in sequences (Brendel et al., 1992; Wootton and Federhen, 1993).

Low-complexity regions are found in both nucleic acids and proteins. In nucleic acids, they are often associated with short repetitive elements such as CpG islands or poly(A) tracts, among others. However, nonrepetitive low-complexity regions are also found, even in coding regions where they often result in low-complexity regions in the encoded protein. (Pizzi and Frontali, 2001). These regions are thought, at least sometimes, to result from selection for AT or GC pairs in the genome.

In proteins, some low-complexity regions mark sites where the protein interacts with other biological macromolecules. For instance, many histones have regions that are very rich in positively charged amino acids, as well as small nonpolar amino acids. As an example, the Swiss-Prot entry H1_Dicdi has the following sequence beginning at residue 103 and running through the end of the sequence:

```
PVAKKPKA KTTATSTETT AAPPATPTKK AAPKKPAAKA
KKNSAKVTKA VSKKPAAKKA PSKKVAAKK
```

The entire fragment is clearly low-complexity and is marked so by the Wootton and Federhen (1993) algorithm. The Swiss-Prot protein Lpc4_Rat contains two low-complexity regions separated by the lowercase “ne” shown below. In database searches, these regions yield matches with a number of glycine-rich proteins that score better than some homologs. The Lpc4_Rat low-complexity regions are:

```
PAGVATGALPGGLLLGTLGGILAnegiLAGQGGLLLGGGLLGDGGLLGGGVLGVLGEGGLIL
```
The structural or functional basis of these low-complexity regions is not known. Numerous and diverse examples can be found by going to an online sequence database such as iProClass or Swiss-Prot and entering a query such as “glycine-rich” or “proline-rich” using virtually any amino acid name.

-M low-high: This parameter allows a researcher to select a range of sequence lengths to be included in the search. The use of this parameter can reduce the amount of sequences searched and minimize the need to compensate database scores based on the lengths of the sequences being compared.

-E [number]: This option limits the amount of output produced by the program by eliminating the listing of any sequence that scores an $E$-value above this value. This option limits the number of sequences presented that are increasingly unlikely to be related to the query sequence.

-F [number]: This option limits the amount of output produced by the program by eliminating the listing of any sequence that scores an $E$-value below this value. This option may be useful if searching for distant relatives of a very large sequence family and one does not wish to look through results that one already knows are highly likely to be related to the query sequence.

Suggestions for Further Analysis

If the Ssearch run does not identify a sufficient group of sequences that are related to the query sequence, consider running the analysis again using a different scoring matrix. The goal is to repeatedly probe the data library with different matrices to ensure that sufficient coverage at all possible evolutionary distances has been obtained (Altschul, 1991; States et al., 1991; Nicholas et al., 2000). Also, it is important to note that, for coding regions of genes, it is generally better to search using the translated amino acid sequence, rather than the nucleic acid sequence (Altschul et al., 1994; Nicholas et al., 2000). If one has been searching only with the nucleic acid sequence against a nucleic acid database, try translating the sequence and search against a protein database.

If the Ssearch run was successful and a group of sequences have been identified that are related to the query sequence, but still more distantly related sequences are desired, consider making a high-quality multiple sequence alignment (Nicholas et al., 2002) from the related sequences (UNIT 3.8). Once the high-quality multiple sequence alignment has been created, the information contained within the alignment can form the basis for an abstract model (e.g., a pattern, profile or hidden Markov model), which can be used to search the data library again for more distantly related sequences (Bork and Gibson, 1996).

Sometimes, additional related regions can be found within the already identified library sequences. An “advanced” derivative of the Smith-Waterman algorithm (Waterman and Eggert, 1987) can be used to probe the identified sequences for additional, nonintersecting alignments between the pairs of sequences. Coded versions of this algorithm (Ropelewski et al., 2000) are particularly useful when the query sequence has more than one domain or when the query sequence contains one or more structural regions that are likely to be repeated, such as the kringle regions in plasminogen activator.


Key References

Altschul et al., 1994. See above.
*This review provides detailed information about local alignment statistics, extreme value distributions, scoring matrices, and low-complexity regions.*

Agarwal and States, 1998. See above.
*This article compares the Smith-Waterman, FASTA, original BLAST code, WU-BLAST2, and Probabilistic Smith-Waterman codes.*

Nicholas et al., 2000. See above.
*This review discusses the advantages and disadvantages of the BLAST, FASTA, and Smith-Waterman search algorithms, how to select appropriate scoring matrices, scoring insertions and deletions, as well as a few different methods by which statistical significance can be computed.*

Pearson, 1995. See above.
*This article compares FASTA, Smith-Waterman and original BLAST algorithms in the context of which method did the best job in finding members of 67 different protein superfamilies.*

Internet Resources

*The FASTA package (in which the Ssearch code is included) can be obtained here.*

*A variety of protein scoring matrices that can be used with the Ssearch code can be obtained here.*

Contributed by Alexander J. Ropelewski, Hugh B. Nicholas Jr., and David W. Deerfield II
Pittsburgh Supercomputing Center
Pittsburgh, Pennsylvania
Mathematically Complete Nucleotide and Protein Sequence Searching Using Ssearch

Ssearch utilizes the Smith and Waterman (1981) algorithm to search a group of sequences (usually a sequence data library such as GenBank or PIR) for local similarities. Sequences containing similar regions can imply a common biological relationship; the Ssearch code is typically used to search a sequence data library for sequences that may be homologous, or share common ancestry, with the query sequence.

The most crucial decision that one makes with any code for searching sequence data libraries is in selecting the rate at which nucleic acid or amino acid substitutions are expected to be found between the sequences being compared. While numerous ways to quantify rates of residue substitution are described in the literature, in practice, when the goal is to find homologous sequences, matrices based on an evolutionary model, such as the explicit PAM model (Dayhoff et al., 1978; UNIT 3.5) or the implicit BLOSUM model (Henikoff and Henikoff, 1992; UNIT 3.5) usually perform best. Because one does not know in advance the appropriate evolutionary distances between the query sequence and each sequence in the data library, multiple runs should be considered to provide complete coverage at all evolutionary distances (Altschul, 1991; Nicholas et al., 2000).

The second crucial choice that needs to be made is determining the magnitude of the gap penalties used for the analysis. Because alignments between relatively long ungapped segments usually dominate database search results, a relatively high penalty should be assessed for initiating a gap. It is important to select appropriate scoring matrix and gap penalty parameters, because the parameters help to define the measure of similarity that is needed to distinguish between a random and a nonrandom sequence.

Statistics presented by the program can help the researcher answer the question: What is the chance that, in searching a library of \( N \) random sequences, a score of \( X \) would be obtained? When this number is very low, a firm statistical argument can be presented implying that the two sequences are related. However, it is also not unusual to find, within the results from a search, sequences that are definitely related, but for which a firm statistical argument cannot be made. Sometimes the statistics can be improved for sequences within this “twilight zone” by repeating the search using a scoring matrix that is better tuned to the evolutionary divergence between the two sequences.

This protocol outlines the steps needed to run the Ssearch code contained in release 3.4 of the FASTA package (UNIT 3.9).

Necessary Resources

Hardware

The Ssearch code is a resource-intensive code and thus generally requires a substantial computational platform with adequate CPU, memory, and disk space. However, these requirements will vary greatly depending on the usage of the code. For example, searching against compilations of known protein sequences, such as those that have been placed in the PIR database or Swiss-Prot databases, can be done on an inexpensive PC running Windows, Linux, or Macintosh OS. Performing regular searches of complete nucleic acid libraries with the code is a task best suited to higher-performance, multiprocessor machines.
Software

The Ssearch code is part of the FASTA package (UNIT 3.9) from Dr. William Pearson, which is available via anonymous FTP from ftp://ftp.virginia.edu/pub/fasta/. Dr. Pearson can be contacted at the Department of Biochemistry, School of Medicine, University of Virginia, Charlottesville, Va. 22908.

Files

Sequence files: The Ssearch code requires an input file that contains the query sequence in FASTA format (APPENDIX 1B). It also requires that one or more sequence data libraries (e.g., GenBank, NBRF-PIR, Swiss-Prot, or EMBL) be installed, or, in lieu of installing a sequence data library, simply having a set of sequences against which one wants to compare the query sequence in FASTA format (APPENDIX 1C).

Scoring matrix file: The Ssearch code enables one to use scoring matrices (UNIT 3.5) that are not internal to the code, such as the BLOSUM35 matrix (Fig. 3.10.1).

The format of the matrix file should be in the same configuration as is acceptable by the BLAST (Altschul et. al., 1997; UNITS 3.3 & 3.4) family of programs.

A variety of compatible scoring matrices that can be used with the Ssearch program can be found at the NCBI FTP site (ftp://ftp.ncbi.nih.gov/blast/matrices). The Ssearch program does not require the use of an external scoring matrix file if one of the built-in scoring matrices is selected.
**Prepare programs and files**

1. Install the FASTA package (UNIT 3.9).

2. Select the data to be analyzed.

   Typically, one selects: (1) an input query sequence in FASTA format and (2) a sequence data library or a set of sequences that one wants to compare against it. In the example used in this protocol, the input query sequence is a 65-residue protein sequence called query.fasta while the data library used is a set of ∼23,000 sequences in the file called library.fasta.

3. Select an appropriate scoring matrix and gap penalties.

   The default matrix used by the Ssearch program when proteins are being searched is the BLOSUM50 matrix (also see UNIT 3.5) with a penalty of 10 assigned to the first residue aligned with a gap, and a penalty of 2 assigned to each additional residue aligned with a gap. These default parameters are adequate if one is interested in performing only a single search through the data library. However, searching with two or three different matrices will generally give more complete coverage across the evolutionary spectrum and may help to resolve the issue of relatedness of sequences about which the user is not certain (Altschul, 1991).

   If two matrices are to be used, consider using the BLOSUM60 matrix along with the BLOSUM35 matrix, or the PAM120 matrix coupled with the PAM250 matrix. If searches with three different matrices are to be used, consider selecting the BLOSUM60, 40, and 30 matrices, or the PAM40, 120, and 240 matrices. Many of the matrices suggested above are not built into the program, but can be found online at the NCBI FTP site (ftp://ftp.ncbi.nih.gov/blast/matrices). The scoring matrices that are internal to the program are BLOSUM50, BLOSUM62, BLOSUM80, PAM250, PAM120, PAM40, PAM20, MDM10, MDM20, MDM30, and MDM40. MDM are modern PAM matrices (Jones et al., 1992).

   A good rule of thumb for setting the penalties for the first residue aligned with a gap when searching a sequence database is to use a value somewhere between about twice the absolute value of the largest negative value in the scoring matrix and the largest positive value in the scoring matrix. Subsequent residues aligned with a gap should be given a relatively low penalty such as 1 or 2 (Nicholas et al., 2000). Gap values in these ranges will generally give the best separation between the related and unrelated sequences (Pearson, 1995). In the example at the end of this protocol, the BLOSUM35 scoring matrix (a matrix that is not built into the program; called blosum35.matrix) is used, with the first residue aligned with a gap having a penalty of 12 and each subsequent residue aligned with a gap having a penalty of 1. To change these options, the –s (scoring matrix), –f (penalty for the first gap), and –g (penalty for additional gaps) command-line options (see step 7) are utilized.

4. Optional: If nucleic acid sequences are being compared, set a match/mismatch penalty.

   The Ssearch code allows one to specify a match/mismatch penalty. By default this value is set to 4. Again, as with proteins, utilizing a variety of scoring parameters may be beneficial (States et al., 1991).

5. Select the statistical method to be used for the analysis.

   Ssearch can analyze the significance of the results of a sequence database search with several different methods. The methods are all different approaches to fitting the observed scores to the extreme value distribution expected for such database searching scores. The default method developed by Pearson (1998) includes a correction for sequence lengths and is generally the preferred method. In cases where the sequence library being searched does not approximate a collection of random sequences, e.g., when it is a database for a single family of sequences, permutation statistics are preferred and can be produced by the shuffling method. The complexities of estimating the significance of the results of a sequence database search are explored in Spang and Vingron (1998) or Waterman and Vingron (1994). The default statistical method will be used in the examples presented in this protocol.
Figure 3.10.2  Histogram of scores: The leftmost numeric column is the score. The center numeric column is the number of times that score occurred within the data search. The rightmost numeric column indicates the number of times that the score was expected to occur in the data library search, after correcting for library sequence length.
6. Select the desired type of output.

Search can produce output in a variety of formats. Usually, the default format is adequate for most uses. However, there are a variety of options that allow one to limit the amount of output produced by the program, including the number of sequence alignments (see Advanced Parameters, below). In general, one wants to see a list of sequences and their alignments for all sequences that are statistically significant. Sometimes, one may want to also show sequences that are not statistically significant but that are nevertheless related. In the example used below, the parameter is set so that the top 30 scoring sequences will be displayed.

Run the program

The commands and parameters below are given as examples.

7. To run the program using the options described above, enter the following command line:

```
Ssearch -s blosum35.matrix -f 12 -g 1
```

When prompted for the query sequence file name, enter:

```
query.fasta
```

When prompted for the name of the library file, enter:

```
library.fasta
```

At this point, the program displays a histogram of scores (Fig. 3.10.2), followed by a prompt to enter an output filename.

8. At the prompt for an output filename, enter the name of a file in which the results are to be stored:

```
querylibrary.results
```

Now, a prompt is displayed for how many scores will be seen. Generally, one wants to see all relevant high scoring sequences. At this prompt, enter:

```
30
```

At the “more scores” prompt, enter:

```
0
```

9. At the next, prompt, asking if one also wants to display the alignments, enter:

```
Y
```

At the “number of alignments” prompt, enter:

```
30
```

10. Examine the output file (querylibrary.results) for the results.

GUIDELINES FOR UNDERSTANDING RESULTS

The program’s output consists principally of a histogram of scores (Fig. 3.10.2) followed by a series of alignments between the query sequence and high-scoring sequences from the sequence data library. If search parameters include a sufficiently high open gap penalty, the histogram will follow an extreme value distribution (Altschul et al., 1994). The pattern followed by an extreme value distribution is that the majority of the scores received are centered on a moderately low score, with a few high-scoring outliers. It is for these high-scoring outliers that a statistical argument is presented whereby the sequences are related to the query sequence in some way. The statistics presented by the
program are based on modeling the expected distribution of unrelated sequences, correcting for the effects caused by library sequence length (Pearson, 1998). This modeling allows accurate statistics to be gleaned for the high-scoring outliers.

Following the histogram is a list of the high-scoring outlier sequences (Fig. 3.10.3). Again, these are the sequences in the data library for which a statistical argument can be presented to infer that the sequences are related to the query sequence. Essentially the statistics presented are giving the researcher the answer to the following question: What is the chance that, in searching through a library of N random sequences, a score of X would be obtained? If the resultant number is very low, it is an indication of a very rare event, unlikely to be caused by chance, allowing the researcher to infer that the two sequences are related. If the resultant number is high, it is an indication that the event may be caused by chance. However, it is not unusual to find sequences in the output for which a firm statistical argument cannot be made, but which are in fact homologous with the query sequence. Thus, the statistical information presented should only be used as a guide, not as a substitute for one’s own biological experiments.

While one’s biological judgment can sometimes resolve the issue that some low-scoring sequences are homologous with the query sequence, occasionally there will be sequences presented in the output about which the researcher may not be so certain. If this is the case, it is important to remember that the results presented may not be accurate if the degree of divergence between the two sequences is not close to the degree of divergence...
of the scoring matrix used within the search. Thus, one way to improve the accuracy of
the results is to repeat the search with a scoring matrix more appropriate to the evolutionary
distance of the two sequences being compared. Thus if an initial search made with a
BLOSUM62 matrix shows a sequence in the results that one is not quite sure about,
sometimes repeating the search with a more distant matrix, such as the BLOSUM45 or
the PAM250 matrix, will improve the accuracy enough to make it possible to infer
relatedness (UNIT 3.5).

This approach to identifying pairs of sequences as homologous has been pushed to its
limits by Kann et al. (2000), who have proposed an iterative refinement for scoring
matrices based on their performance in searching a curated database where all of the
homologous pairs of sequences are known, thus improving their ability to detect remote
homology between pairs of sequences. An alternative to changing the scoring scheme is
changing the alignment model from the local, Smith-Waterman model to a global model
(Webber and Barton, 2003) and examining the intersection set of sequences reported by
both alignment models. This eliminates false and near-false positive results that are of
greatly different length from the query sequence.

Other approaches to identifying sequence pairs showing a low level of similarity as
homologous are related to the ideas underlying profile searching, in particular the idea
that homologous sequences will show conservation of the sequence residues essential to
their homologous structure and function. The most direct application of this idea is in the
use of sequence profiles or hidden Markov models to model an homologous family of
sequences. A wide variety of these techniques are discussed in Bork and Gibson (1996).
An extreme case of these methods in which one examines the relationship between the
query sequence and two remote potential homologs has become known as an intermediate
sequence search (Weizhong et al., 2000) and may be worthwhile in some circumstances.

Masking the low-complexity regions in the query sequence (Altschul et al., 1994) is
generally very helpful in identifying very distantly homologous relatives by removing
false-positive matches to the low-complexity regions.

Homology searches are based on the following statistical rationale: if the observed
similarity between two sequences is very unlikely to be observed by chance, then it may
be concluded that the two sequences are homologous. The similarity is measured by an
alignment score between the pairs of sequences. Unfortunately for present-day sequence
databases, one will find alignment scores with random sequences with one’s query
sequence that will be as high or higher than the alignment scores for the same query
sequence with homologous sequences (Spang and Vingron, 2001). Additional database
searches with scoring matrices designed to detect more remote homologies will often
distinguish the sequences that are actually homologous from high-scoring matches to
random sequences, because the actual homologs will show a marked improvement in
score with the different scoring matrix (Altschul, 1991). Also, distant homologs are
expected to still show examples of sequence motifs associated with the conserved
functions of the homologous sequence family, while high-scoring matches with random
sequences would not be expected to have these motifs.

The next items presented in the results are the alignments between the query sequence
and the high-scoring library sequences. It is important to keep in mind that these sequence
alignments are based on parameters selected to produce alignments, and corresponding
alignment scores, that most effectively distinguish the sequences in the sequence library
that are homologous to one’s query sequence from the nonhomologous, accidentally
high-scoring sequences in the library. These parameters are not necessarily the parameters
that will provide the alignment that is the best estimate of residue-by-residue sequence
homology for the two sequences when used with a global alignment algorithm such as the Needleman-Wunsch (1970) algorithm. Alignment parameters for database searching are selected to produce local alignments that have few gaps, and thus to reduce the number of similarly high-scoring alignments between random sequences which are likely to have many gaps (Vingron and Waterman, 1994).

**COMMENTARY**

**Background Information**

The Ssearch program utilizes the computational technique generally referred to as *dynamic programming* (Aho et al., 1983). Dynamic programming is a rigorous mathematical technique that works by breaking a problem into a table of partial solutions that are combined to give the optimal solution to the problem.

While optimal solutions are frequently desired, especially when creating an alignment between two similar sequences, finding the optimal solution requires significant computer resources. To search molecular biology databases with an unknown query sequence for sequences that may potentially be related to that query sequence, frequently an optimal solution is not necessary to guide the researcher along the correct path. Faster database-searching programs such as FASTA (Pearson and Lipman, 1988; *UNIT 3.9*) and BLAST (Altschul et al., 1997; *UNITS 3.3 & 3.4*) should generally be tried first. If an adequate solution is not found using those faster algorithms, then the Ssearch program may be run. In general, an Ssearch run will find a related sequence or two that the faster database searching programs were unable to find (Pearson, 1995; Agarwal and States, 1998).

**Critical Parameters and Troubleshooting**

When one selects parameters for any sequence database searching program, it is important to keep in mind that, conceptually, what one is trying to do is to separate the sequence data library into two distinct groups—a small group containing sequences that are related to the query sequence and a much larger group of sequences that are statistically random with respect to the query sequence. The parameters that are most influential on the separation are command-line options for which the user is not prompted when the program is run interactively. These parameters are:

- **-f**: This parameter is used to tell the program the penalty that the researcher wishes to use for the first residue in a gap. When searching a sequence data library for sequences that may be related to the query sequence, this value should be set relatively high. A rule of thumb is to select a value somewhere between about twice the absolute value of the largest negative value used in the selected scoring matrix and the largest positive value in the scoring matrix. A value in this range will generally give the best separation between the related and unrelated sequences. If the best local sequence alignments are desired, a value of about half this size is frequently more appropriate.

- **-g**: This parameter determines the penalty used by extending the gap an additional residue. For database searching, this value is almost always set to 1 or 2. For the best local sequence alignments, this value is usually set between 1 and the value used for the first residue in the gap.

- **-s [file]**: This option is used to specify the scoring matrix file or the built-in scoring matrix to be used. The scoring matrices that are internal to the program are: BLOSUM50 (-s BL50), BLOSUM62 (-s BL62), BLOSUM80 (-s BL80), PAM250 (-s P250), PAM120 (-s P120), PAM40 (-s P40), PAM20 (-s P20), MDM10 (-s M10), MDM20 (-s M20), MDM30 (-s M30), and MDM40 (-s M40).
For proteins, searching with three implicit or explicit evolutionary matrices will generally give good coverage across all evolutionary distances (e.g., BLOSUM60, 30, and 40, or PAM40, 120, or 250).

- r: This parameter is used to specify match/mismatch scores for DNA comparisons.

Occasionally, unexpected results can occur if the program incorrectly identifies the query sequence as a protein sequence when it is in fact a nucleic acid sequence or vice versa. If one suspects that this has happened, rerun the search with the appropriate command line switch:

- n: Tells the program that the query sequence is a nucleic acid sequence.

- p: Tells the program that the query sequence is an amino acid sequence.

Advanced Parameters

There are several other parameters used by the program, which fall into one of three general categories: parameters that control the input to the program, parameters that control the format of the output, and parameters that control the type of statistics produced. The majority of these parameters do not have a significant effect on the results received and are described in detail in the manual page that accompanies the software. However, there are a few advanced command-line options that merit further discussion. They are:

- S: This option treat lowercase characters in the query or library sequences as “low-complexity” residues. Low-complexity regions are sections of sequence that contain biased compositions such as a few types of residues or short periodic repeats. Low-complexity regions are problematic with database searches because they tend to have very different composition from the database as a whole, and, when included, can affect the accuracy of the statistics presented. Furthermore, these regions, when randomized, usually give alignment scores similar to the original sequence (Altschul et al., 1994). There are several good programs available that can be used to detect low-complexity regions in sequences (Brendel et al., 1992; Wootton and Federhen, 1993).

Low-complexity regions are found in both nucleic acids and proteins. In nucleic acids, they are often associated with short repetitive elements such as CpG islands or poly(A) tracts, among others. However, nonrepetitive low-complexity regions are also found, even in coding regions where they often result in low-complexity regions in the encoded protein. (Pizzi and Frontali, 2001). These regions are thought, at least sometimes, to result from selection for AT or GC pairs in the genome.

In proteins, some low-complexity regions mark sites where the protein interacts with other biological macromolecules. For instance, many histones have regions that are very rich in positively charged amino acids, as well as small nonpolar amino acids. As an example, the Swiss-Prot entry H1_Dicdi has the following sequence beginning at residue 103 and running through the end of the sequence:

PVAKKPKA KTTATSTETT AAPPATPTKK AAPKKPAAKA KKNSAKVTKA VSKKPAAKKA PSKKVAAKK

The entire fragment is clearly low-complexity and is marked so by the Wootton and Federhen (1993) algorithm. The Swiss-Prot protein Lpc4_Rat contains two low-complexity regions separated by the lowercase “ne” shown below. In database searches, these regions yield matches with a number of glycine-rich proteins that score better than some homologs. The Lpc4_Rat low-complexity regions are:

PAGVATGALGPGLGLGTGGIILAngGILAGQGGLLGGGGLLGDGGLLGGGGVLGVLGGGIL

Finding
Similarities and
Inferring
Homologies

3.10.9
The structural or functional basis of these low-complexity regions is not known. Numerous and diverse examples can be found by going to an online sequence database such as iProClass or Swiss-Prot and entering a query such as “glycine-rich” or “proline-rich” using virtually any amino acid name.

-M low-high: This parameter allows a researcher to select a range of sequence lengths to be included in the search. The use of this parameter can reduce the amount of sequences searched and minimize the need to compensate database scores based on the lengths of the sequences being compared.

-E [number]: This option limits the amount of output produced by the program by eliminating the listing of any sequence that scores an E-value above this value. This option limits the number of sequences presented that are increasingly unlikely to be related to the query sequence.

-F [number]: This option limits the amount of output produced by the program by eliminating the listing of any sequence that scores an E-value below this value. This option may be useful if searching for distant relatives of a very large sequence family and one does not wish to look through results that one already knows are highly likely to be related to the query sequence.

Suggestions for Further Analysis

If the Ssearch run does not identify a sufficient group of sequences that are related to the query sequence, consider running the analysis again using a different scoring matrix. The goal is to repeatedly probe the data library with different matrices to ensure that sufficient coverage at all possible evolutionary distances has been obtained (Altschul, 1991; States et al., 1991; Nicholas et al., 2000). Also, it is important to note that, for coding regions of genes, it is generally better to search using the translated amino acid sequence, rather than the nucleic acid sequence (Altschul et al., 1994; Nicholas et al., 2000). If one has been searching only with the nucleic acid sequence against a nucleic acid database, try translating the sequence and search against a protein database.

If the Ssearch run was successful and a group of sequences have been identified that are related to the query sequence, but still more distantly related sequences are desired, consider making a high-quality multiple sequence alignment (Nicholas et al., 2002) from the related sequences (UNIT 3.8). Once the high-quality multiple sequence alignment has been created, the information contained within the alignment can form the basis for an abstract model (e.g., a pattern, profile or hidden Markov model), which can be used to search the data library again for more distantly related sequences (Bork and Gibson, 1996).

Sometimes, additional related regions can be found within the already identified library sequences. An “advanced” derivative of the Smith-Waterman algorithm (Waterman and Eggert, 1987) can be used to probe the identified sequences for additional, nonintersecting alignments between the pairs of sequences. Coded versions of this algorithm (Ropelewski et al., 2000) are particularly useful when the query sequence has more than one domain or when the query sequence contains one or more structural regions that are likely to be repeated, such as the kringle regions in plasminogen activator.
Literature Cited


Key References

Altschul et al., 1994. See above.
This review provides detailed information about local alignment statistics, extreme value distributions, scoring matrices, and low-complexity regions.

Agarwal and States, 1998. See above.
This article compares the Smith-Waterman, FASTA, original BLAST code, WU-BLAST2, and Probabilistic Smith-Waterman codes.

Nicholas et al., 2000. See above.
This review discusses the advantages and disadvantages of the BLAST, FASTA, and Smith-Waterman search algorithms, how to select appropriate scoring matrices, scoring insertions and deletions, as well as a few different methods by which statistical significance can be computed.

Pearson, 1995. See above.
This article compares FASTA, Smith-Waterman and original BLAST algorithms in the context of which method did the best job in finding members of 67 different protein superfamilies.

Internet Resources

The FASTA package (in which the Ssearch code is included) can be obtained here.

A variety of protein scoring matrices that can be used with the Ssearch code can be obtained here.

Contributed by Alexander J. Ropelewski, Hugh B. Nicholas Jr., and David W. Deerfield II
Pittsburgh Supercomputing Center
Pittsburgh, Pennsylvania
BLAST, the Basic Local Alignment Search Tool (Altschul et al., 1990; Altschul et al., 1997; UNITS 3.3 & 3.4) is one of the most widely used and useful tools in molecular sequence analysis. It is employed for the assignment of function to novel sequences, EST clustering, collection of related sequences for phylogenetic analysis, identification of conserved regions in sequences for structure prediction, and a range of other applications. As a consequence, many bioinformatics protocols start in or go through BLAST, and this will be evident in many other units of *Current Protocols in Bioinformatics*.

This unit describes the various options available to a typical researcher who wishes or needs to run a local version of BLAST. The issues around obtaining and installing the BLAST software on various common computing platforms including Linux and other Unix-like programs (see Basic Protocol) and MS Windows (see Alternate Protocol) are discussed. Information is also included on the aggregation and indexing of sequence data so that they can be used by local versions of BLAST.

**NOTE:** Investigators who are unfamiliar with the Unix environment are encouraged to read APPENDIX 1C & APPENDIX 1D.

### STRATEGIC PLANNING

The most significant strategic planning issues around the decision to use a local version of BLAST are detailed below. The following checklist serves as a guide to determine if a local version of BLAST is the best way to solve the need for use of the BLAST software. Answering “yes” to all or many of the questions below indicates that a local version of BLAST is a requirement for the use of this system.

1. Does organizational policy forbid the use of the Internet for the transmission of data or the use of remote services for data analysis?
2. Are there local data collections that are to be queried using BLAST?
3. Is Internet connectivity slow/unreliable/expensive? Note that if this is the case and it is also necessary to search large databanks that require the Internet for downloading, then it may still be a problem to use BLAST locally, as these databanks themselves may be difficult to obtain without good Internet connectivity.
4. Are there frequent, time-critical, high-throughput BLAST analysis requirements (e.g., 1000 searches a day)?
5. Are ample fast computing resources and large volumes of disk storage capacity available?
6. Is BLAST to be included as a major step in other time-critical bioinformatics pipelines?

The following are other issues that are relevant to the management of local versions of BLAST (and any local bioinformatics analysis for that matter):

1. Does the laboratory have good technical support and/or the necessary skills to manage the software and databanks locally?
2. Is there a good backup strategy in place to restore lost data and software?
Finally, a local version of BLAST may not always be an option. The following conditions may make use of a remote BLAST service a prerequisite or the best option:

1. Are the data to be queried available only through a remote server (i.e., the data cannot be copied by the Internet or shipped on disk in a native or processed format)? If so, a local version of BLAST will be of no help.

2. Is use of BLAST very infrequent and not constrained by access to Internet-based remote services? If so, the effort to install and maintain a copy of BLAST locally may not be worthwhile.

**INSTALLING AND RUNNING BLAST LOCALLY UNDER UNIX-LIKE OPERATING SYSTEMS SUCH AS LINUX**

Installation of BLAST locally is a straightforward procedure. This discussion will focus on Unix-like operating systems (e.g., Linux, AIX, MacOSX, or FreeBSD), as these are the most commonly used in a local environment for bioinformatics. This protocol is adapted from the instructions provided by NCBI in the file `README.bls`, included as part of the BLAST distribution, and from the book *iBiostation Linux: Bioinformatics for Linux* by M. Hobbs, T. G. Littlejohn, and K. Castle (see Internet Resources).

**Necessary Resources**

**Hardware**

The hardware requirements for running BLAST locally are modest indeed: any Intel or equivalent (e.g., AMD)–based architecture will be adequate for running this protocol. However a few considerations on hardware need to be addressed:

1. BLAST can be computationally intensive. For example, if searching large databases, using CPU-intensive BLAST algorithms (e.g., TBLASTX) or searching with many sequences, more CPU power is better.
2. BLAST can be memory-hungry; for ideal performance, one should have enough memory to load the entire indexed database comfortably into RAM.
3. Databases can be large and require large amounts of disk space. Researchers who decide to take on ambitious projects such as downloading the entire GenBank database for local searching should keep this in mind.

The nice thing about BLAST is that the hardware needs can be scaled easily by adding more disk space or RAM or moving to multiprocessor architectures. In addition, if using BLAST to query a large number of sequences against a common database, this process can very easily be parallelized by copying the indexed database, the relevant query sequences, and the `blastall` application to any number of machines on which the analyses are to be performed, and running them at the same time.

**Software**

There are several programs in the stand-alone BLAST package. The main ones that are needed to run BLAST locally are `formatdb` to create BLASTable databases and `blastall` to query these databases using any of the favorite BLAST algorithms (`blastn`, `blastp`, `blastx`, `tblastn`, and `tblastx`). `formatdb` is a program for formatting FASTA formatted databases for searching using BLAST. Details on `formatdb` can be found in the file `README.formatdb` distributed with the BLAST package. The options for `formatdb` are listed in Table 3.11.1. `blastall` is the main BLAST application. It is used for running queries against the indexed databases created with `formatdb`. Details on `blastall` can be found in the file `continued`
Table 3.11.1 Options for formatdb

<table>
<thead>
<tr>
<th>Option</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>-t</td>
<td>Title for database file [String] (optional)</td>
</tr>
<tr>
<td>-i</td>
<td>Input file(s) for formatting [File In] (this parameter must be set)</td>
</tr>
<tr>
<td>-l</td>
<td>Logfile name: [File Out] (optional) default = formatdb.log</td>
</tr>
<tr>
<td>-p</td>
<td>Type of file [T/F] (optional): T = protein, F = nucleotide default = T</td>
</tr>
<tr>
<td>-o</td>
<td>Parse options [T/F] (optional): T (true) = parse SeqID and create indexes F (false) = do not parse SeqID; do not create indexes default = F</td>
</tr>
<tr>
<td>-a</td>
<td>Input file is database in ASN.1 format (otherwise FASTA is expected) [T/F] (optional): T = True, F = False default = F</td>
</tr>
<tr>
<td>-b</td>
<td>ASN.1 database in binary mode [T/F] (optional): T = binary, F = text mode default = F</td>
</tr>
<tr>
<td>-e</td>
<td>Input is a Seq entry [T/F] (optional) default = F</td>
</tr>
<tr>
<td>-n</td>
<td>Base name for BLAST files [String] (optional)</td>
</tr>
<tr>
<td>-v</td>
<td>Number of sequence bases to be created in the volume [Integer] (optional) default = 0</td>
</tr>
<tr>
<td>-s</td>
<td>Create indexes limited only to accessions: sparse [T/F] (optional) default = F</td>
</tr>
<tr>
<td>-V</td>
<td>Verbose: check for nonunique string IDs in the database [T/F] (optional) default = F</td>
</tr>
<tr>
<td>-A</td>
<td>Create ASN.1 structured deflines [T/F] (optional) default = F</td>
</tr>
</tbody>
</table>

For an example of using these options, see Basic Protocol, step 7.

**Files**

Input data files must be in FASTA format (see Appendix 1B)


*NCBI provides distribution files for all commonly used systems. The files contain the precompiled executables needed to run BLAST searches, as well as essential data files and some excellent documentation. The platforms supported include many Unix-like systems including FreeBSD for Intel, IRIX6.2, Solaris2.6 for Sparc, Solaris2.7 for Intel, DEC OSF1 for alpha, and Linux for Intel. Distributions are also available for MacOS 9 and MacOSX for Power PC and MS-Win 32 for Intel. In this example, the Linux distribution of the BLAST package will be described, which at the time of this writing can be found in the file blast-2.2.6-ia32-linux.tar.gz.*
2. Create a directory within the home directory called blast and move the downloaded binary package to this directory, which will be used to temporarily hold the software for the process of the installation. To make a directory, type:

```
mkdir blast
```

at the Unix prompt. Next, move the appropriate distribution to this directory; e.g., if the Linux distribution called blast-2.2.6-ia32-linux.tar.gz was downloaded, then type:

```
mv blast-2.2.6-ia32-linux.tar.gz blast
```

---

**Table 3.11.2 Options for blastall**

<table>
<thead>
<tr>
<th>Option</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>-p</td>
<td>Program name [String]</td>
</tr>
<tr>
<td>-d</td>
<td>Database [String]</td>
</tr>
<tr>
<td>-i</td>
<td>Query file [File In]</td>
</tr>
<tr>
<td>-e</td>
<td>Expectation value (E) [Real]</td>
</tr>
<tr>
<td>-o</td>
<td>BLAST report output file [File Out] (optional)</td>
</tr>
<tr>
<td>-F</td>
<td>Filter query sequence (dust with BLASTN, seg with others) [String]</td>
</tr>
<tr>
<td>-S</td>
<td>Query strands to search against database (for BLAST[NX], and TBLASTX). 3 is both, 1 is top, 2 is bottom [Integer]</td>
</tr>
<tr>
<td>-T</td>
<td>Produce HTML output [T/F]</td>
</tr>
<tr>
<td>-l</td>
<td>Restrict search of database to list of GI’s [String] (optional)</td>
</tr>
<tr>
<td>-U</td>
<td>Use lowercase filtering of FASTA sequence [T/F] (optional)</td>
</tr>
</tbody>
</table>

---

aFor an example of using these options, see Basic Protocol, step 9.

---

2. Create a directory within the home directory called blast and move the downloaded binary package to this directory, which will be used to temporarily hold the software for the process of the installation. To make a directory, type:  

```
mkdir blast
```

at the Unix prompt. Next, move the appropriate distribution to this directory; e.g., if the Linux distribution called blast-2.2.6-ia32-linux.tar.gz was downloaded, then type:

```
mv blast-2.2.6-ia32-linux.tar.gz blast
```
3. Change to the blast directory. Uncompress the file.

The files distributed by NCBI come either as “compressed” (ending with a .Z) or as gzipped (ending with a .gz). To uncompress the file (for the example above), type the following at the Unix prompt:

```
gunzip blast-2.2.6-ia32-linux.tar.gz
```

4. “Un-tar” (tar is a method of archiving files) the uncompressed file by typing the following command at the Unix prompt:

```
tar -xvf blast-2.2.6-ia32-linux.tar
```

This should give a verbose listing of all the files as they are “un-tarred” such as that shown in Figure 3.11.1.

5. Move the files to directories that make sense to the system.

One may need the assistance of a systems administrator for this step. For example, move the data directory and all its contents to /usr/local/share with the following commands (run as root or using the Unix sudo command):
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Contents</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peptide</td>
<td>alu.a.Z</td>
<td>Translations of select Alu repeats from REPBASE</td>
<td>101 kb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>alu.n.Z</td>
<td>Select Alu repeats from REPBASE</td>
<td>24.8 kb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>Human_genomic.Z</td>
<td>Human genome sequences</td>
<td>751 Mb</td>
</tr>
<tr>
<td>Peptide</td>
<td>drosoph.aa.Z</td>
<td><em>Drosophila</em> genome proteins provided by Celera and Berkeley <em>Drosophila</em> Genome Project (BDGP)</td>
<td>4.49 Mb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>drosoph.nt.Z</td>
<td><em>Drosophila</em> genome provided by Celera and Berkeley <em>Drosophila</em> Genome Project (BDGP)</td>
<td>32.7 Mb</td>
</tr>
<tr>
<td>Peptide</td>
<td>ecoli.aa.Z</td>
<td><em>Escherichia coli</em> genomic CDS translations</td>
<td>994 kb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>ecoli.nt.Z</td>
<td><em>Escherichia coli</em> genomic nucleotide sequences</td>
<td>1.28 Mb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>est_human.Z</td>
<td>Database of GenBank+EMBL+DDBJ sequences from EST Divisions</td>
<td>953 Mb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>est_mouse.Z</td>
<td>Database of GenBank+EMBL+DDBJ sequences from EST Divisions</td>
<td>585 Mb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>est_others.Z</td>
<td>Database of GenBank+EMBL+DDBJ sequences from EST Divisions</td>
<td>1.43 Gb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>gss.Z</td>
<td>Genome Survey Sequence, includes single-pass genomic data, exon-trapped sequences, and Alu-PCR sequences</td>
<td>1.09 Gb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>htg.Z</td>
<td>Unfinished high-throughput genomic sequences: phases 0, 1 and 2 (finished, phase 3 HTG sequences are in nr)</td>
<td>2.87 Gb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>igSeqNt.Z</td>
<td>Nucleotide immunoglobulin sequences in GenBank</td>
<td>9.69 Mb</td>
</tr>
<tr>
<td>Peptide</td>
<td>igSeqProt.Z</td>
<td>Peptide immunoglobulin sequences in GenBank</td>
<td>1.89 Mb</td>
</tr>
<tr>
<td>Peptide</td>
<td>mito.aa.Z</td>
<td>Database of mitochondrial sequences</td>
<td>377 kb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>mito.nt.Z</td>
<td>Database of mitochondrial sequences</td>
<td>865 kb</td>
</tr>
<tr>
<td>Peptide</td>
<td>month.aa.Z</td>
<td>All new or revised GenBank CDS translation+PDB+SwissProt+PIR+PRF released in the last 30 days</td>
<td>9.76 Mb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>month.est_human.Z</td>
<td>Monthly human est updates</td>
<td>4.78 Mb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>month.est_mouse.Z</td>
<td>Monthly mouse est updates</td>
<td>5.61 Mb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>month.est_others.Z</td>
<td>Monthly other est updates</td>
<td>61.3 Mb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>month.gss.Z</td>
<td>Monthly gss updates</td>
<td>27.1 Mb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>month.htgs.Z</td>
<td>Monthly htgs updates</td>
<td>171 Mb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>month.nt.Z</td>
<td>All new or revised GenBank+EMBL+DDBJ+PDB sequences released in the last 30 days</td>
<td>73.7 Mb</td>
</tr>
<tr>
<td>Peptide</td>
<td>nr.Z</td>
<td>All nonredundant GenBank CDS translations+PDB+SwissProt+PIR+PRF</td>
<td>411 Mb</td>
</tr>
<tr>
<td>Nucleotide</td>
<td>nt.Z</td>
<td>All GenBank+EMBL+DDBJ+PDB sequences (but no EST, STS, GSS, or phase 0, 1 or 2 HTGS sequences). No longer “nonredundant”</td>
<td>2.39 Gb</td>
</tr>
<tr>
<td>Peptide</td>
<td>pataa.Z</td>
<td>Protein sequences derived from the Patent division of GenBank</td>
<td>16.7 Mb</td>
</tr>
</tbody>
</table>

*continued*
In addition, one may want to move the executable files such as `formatdb` and `blastall` to the `/usr/local/bin` directory.

6. Create a `.ncbirc` file in the home directory or in a directory that will be used to run the local version of BLAST.

   Stand-alone BLAST requires a special file called `.ncbirc`. This file contains information required by the local version of BLAST in order for it to execute properly. The file should contain the following two lines:

   [NCBI]
   Data=/usr/local/share/ncbi/data

   The actual syntax here will depend on the location of the `ncbi` data directory. When the local BLAST package installs, it will create the data subdirectory automatically where the downloaded file was extracted (see the output of the `tar` command in step 4, above). In this case, then, it is assumed that the files were extracted in the directory `/usr/local/share/ncbi`, and that in the step above this directory was created and these files were moved there or created there simply by running the `tar` command as described in step 4 (output shown in Fig. 3.11.1). If this was not done, simply put the path of the location in which the data directory was created or else move it to a more “typical” location such as `/usr/local/share`; a systems manager’s help may be needed if one wants to move these files into a location such as `/usr/local/share`. Note that this file must be in the home directory or in the directory where BLAST will be run (often the directory that contains the data; see below).

7. Format the database of interest so that it can be searched using BLAST.

   NCBI provides a number of FASTA-formatted files that can be downloaded from their FTP site at ftp://ftp.ncbi.nih.gov in the `/blast/db/FASTA` directory. The available files and their approximate sizes as of April 2003 are shown in Table 3.11.3.

   The BLAST package provides software for indexing or formatting databases so that they can be searched using BLAST. This program is called `formatdb` and is run to create BLAST...
searchable databases. These databases need to be provided in FASTA format (APPENDIX 1B).

In this example, a database obtained from NCBI mitochondrial proteins database (also see Table 3.11.3) will be used. This database is in FASTA format and is obtained by FTP from ftp://ftp.ncbi.nih.gov/blast/db/FASTA/mito.aa.Z. Alternatively any database of sequences in FASTA format can be used. It does not really matter where files to be used as databases are placed. Good practice might be to put the files in a project-specific directory or in a common directory that other users of the same computer who want to use BLAST can access. Uncompress this database as described above; at the Unix prompt type:

```
uncompress mito.aa.Z
```

or, if this does not work, try:

```
zcat mito.aa.Z
```

The resulting file, mito.aa, is now ready to be indexed for searching with BLAST. In order to do this, the formatdb program needs to be run. In this instance, the following command would be used:

```
formatdb -p T -i mito.aa
```

The –p T argument indicates that the database is of protein sequences and the -i mito.aa argument indicates that the input file is called mito.aa (see Table 3.11.1).

8. Collect the query sequence(s) of interest.

In this example a sequence in FASTA format—the file fungus.fasta (shown in Fig. 3.11.2) will be compared against the indexed database to see if this sequence shows any similarity to the animal sequences.

9. Run BLAST.

All the commonly used BLAST algorithms can be run from the blastall program. To run BLAST, in this case using the blastp algorithm, use the blastall program using the following command:

```
blastall -p blastp -d mito.aa -i fungus.fasta -o fungus.blastp
```

The output is saved to a file called fungus.blastp. A detailed list of the options for running blastall in Table 3.11.2.

10. Look at the output file fungus.blastp using a file viewer of choice or the Unix more or less commands. The file should look something like Figure 3.1.3.

*The sequence similarity shown by BLAST suggests that this unknown fungal sequence is probably an ATP synthase F0 subunit 6.*
Installing and Running BLAST Locally Under Microsoft Windows

The NCBI team has done a great job in supporting all the common hardware and operating system combinations that are typically used by bioinformatics researchers. With the advent of MacOSX, all of the common operating systems except MS Windows are now Unix-like, so the Basic Protocol, with minor adjustments, will work for all these systems. For further information on the nuances for various Unix platforms, see the file README.bls that came with the BLAST package distribution.

This protocol will focus on using BLAST under MS-Windows. The procedure is adapted from the file README.bls that comes with the BLAST package distribution.

1. Download the BLAST Windows distribution file blastz.exe from ftp://ftp.ncbi.nih.gov/blast/executables/LATEST-BLAST/. Create a new directory (e.g., called blast) and move the file into this directory. The blastz.exe file is a self-extracting archive; simply double-click on it in Windows Explorer or run it from a DOS prompt to extract the files inside.

2. Create an ncbi.ini file (the Windows equivalent of the .ncbirc file used for the Unix distribution; see Basic Protocol, step 6).

   In order for stand-alone BLAST to operate under Windows, it is necessary to have an ncbi.ini file that contains the following lines:

   [NCBI]
   Data="C:\path\data\"

   where “C:\path\data\" is the path to the location of the local BLAST data subdirectory. Next, move this file to the Windows or WINNT directory on the computer’s hard drive.

   Note that if an ncbi.ini file is already on one’s machine from installing other NCBI software (e.g., Entrez or Sequin) this step can be skipped. However, if the following error message is seen when running the BLAST analyses:
Abrupt: code=1
FATAL ERROR: FindPath failed.

the old ncbi.ini file should be renamed as, e.g., ncbi.bak, and the instructions above should be followed to create a new ncbi.ini file.

3. Format the database using the formatdb program that comes with the BLAST distribution.

formatdb for MS-Windows follows the same syntax as the Unix version, e.g., for the example described in the Basic Protocol, simply type:

formatdb -p T -i mito.aa

at the DOS prompt.

4. Run the first search (as in the Basic Protocol) using the command:

blastall -p blastp -d mito.aa -i fungus.fasta -o fungus.blastp

COMMENTARY

Background Information

Why use BLAST?

Other units describe the use and interpretation of BLAST for the analysis of nucleotide and protein sequence data (UNITS 3.3 & 3.4). The purpose of this unit is not to duplicate these detailed protocols, but rather to describe the technical aspects of installing and maintaining the BLAST toolset for use on a stand-alone workstation (even a laptop computer) or an intranet server. Once installed, this instance of the BLAST software is frequently referred to as a “local” copy of BLAST, and that terminology will be adopted in this unit.

There are many reasons why it is desirable or necessary to have a locally installed version of BLAST as opposed to the alternative, remote BLAST services. Reasons for needing a local copy of BLAST include:

Security: Frequently, organizational policy requires that all data stay within the organization, and not be transmitted to remote sites across the Internet. While it can be very convenient to use a remote BLAST server such as that maintained by NCBI, use of these services may contravene such policy.

Unique data: Remote BLAST services are restricted to allow the query of the databanks they support. Frequently, in bioinformatics, research scientists generate data sets from subsets of pre-existing data created by themselves or others. Remote services do not allow the query of these data collections, which are typically stored on scientists’ workstations or intranet servers.

High-throughput analysis: Many remote BLAST services allow analysis of a large number of sequences in succession or even in parallel. Frequently, the computers supplying these services are very powerful, and return results rapidly and efficiently. However, it is not uncommon for these services to restrict the number of BLAST searches that they allow a particular site to perform in a given time. In addition, the power of low-end workstations and clustered compute farms make it quite reasonable to perform large numbers of analyses in-house at an affordable cost. As a consequence, these analyses are best done using a local version of BLAST.

Network reliability: Remote BLAST services offer many advantages: they are maintained by someone else, using their computers, so the hardware and maintenance costs can be negligible for a user of such services. However, slow or unreliable network connections to these sites can make use of such services impractical.

Additional software

The NCBI software distribution includes a large number of applications in addition to the ones explained in the Basic and Alternate Protocols. These applications are described below, and further information on them can be found in the files obtained as part of the BLAST distributions.

BLASTCLUST: This program automatically and systematically clusters protein or DNA sequences based on pairwise matches found using the BLAST algorithm in case of proteins or the MEGABLAST algorithm for DNA.
the latter case, a single MEGABLAST search is performed for all the sequences combined against a database created from the same sequences. BLASTCLUST finds pairs of sequences that have statistically significant matches and clusters them using single-linkage clustering. For more information on BLASTCLUST, see the file README.bcl that comes with the BLAST package distribution.

fastacmd: The program fastacmd retrieves FASTA-formatted sequences from a BLAST database (provided the database was formatted using the -o option (see the file README.formatdb for more information on formatting options using formatdb). For more information on fastacmd, see the file README.fastacmd that comes with the BLAST package distribution.

MEGABLAST: This program is optimized for aligning sequences that differ slightly as a result of sequencing errors or other sequence variations (e.g., polymorphisms). MEGABLAST is up to 10 times faster than more common sequence similarity programs and therefore can be used to swiftly compare two large sets of sequences against each other. For more information on MEGABLAST, see the file README.mbl that comes with the BLAST package distribution.

bl2seq: bl2seq performs a comparison between two sequences using either the BLASTN or BLASTP algorithm. For more information on bl2seq, see the file README.bl2s that comes with the BLAST package distribution.

blastpgp: The blastpgp program can do an iterative search in which sequences found in one round of searching are used to build a score model for the next round of searching. blastpgp can be used in psi-blast (Position-Specific Iterated BLAST) and phi-blast (Pattern-Hit Initiated BLAST) modes. For more information on blastpgp, see the file README.bls that comes with the BLAST package distribution.

RPS-BLAST: This program (Reverse PSI-BLAST) searches a query sequence against a database of profiles. This is the opposite of PSI-BLAST, which searches a profile against a database of sequences, hence the “Reverse.” For more information on RPS-BLAST, see the file README.rps that comes with the BLAST package distribution.

copymat: copymat is a secondary profile preprocessor used as a part of RPS-BLAST analyses. It converts ASCII matrices, produced by the primary preprocessor (makemat), into a database that can be read into memory quickly. For more information on copymat, see the file README.bls that comes with the BLAST package distribution.

makemat: makemat is the primary profile preprocessor used as a part of RPS-BLAST analyses. For more information on makemat, see the file README.bls that comes with the BLAST package distribution.

impala: impala is used for matching a protein sequence against a collection of PSI-BLAST-constructed position-specific score matrices. For more information on impala, see the file README.imp that comes with the BLAST package distribution.

seedtop: seedtop can be used to answer questions such as “Given a sequence and a database of patterns, which patterns occur in the sequence and where?” and “Given a pattern and a sequence database, which sequences contain the pattern and where?”. For more information on seedtop, see the file README.bls that comes with the BLAST package distribution.

Troubleshooting
Beware that the NCBI updates its Web site regularly, and as a consequence the files described in this unit may have moved. The best way to stay on top of such changes is to join the BLAST-Announce E-mail service at http://www.ncbi.nlm.nih.gov/BLAST/blastannounce.html.

Literature Cited

Internet Resources
http://www.ibiostation.com

Contributed by Timothy G. Littlejohn
IBM Life Sciences
St. Leonards, NSW, Australia