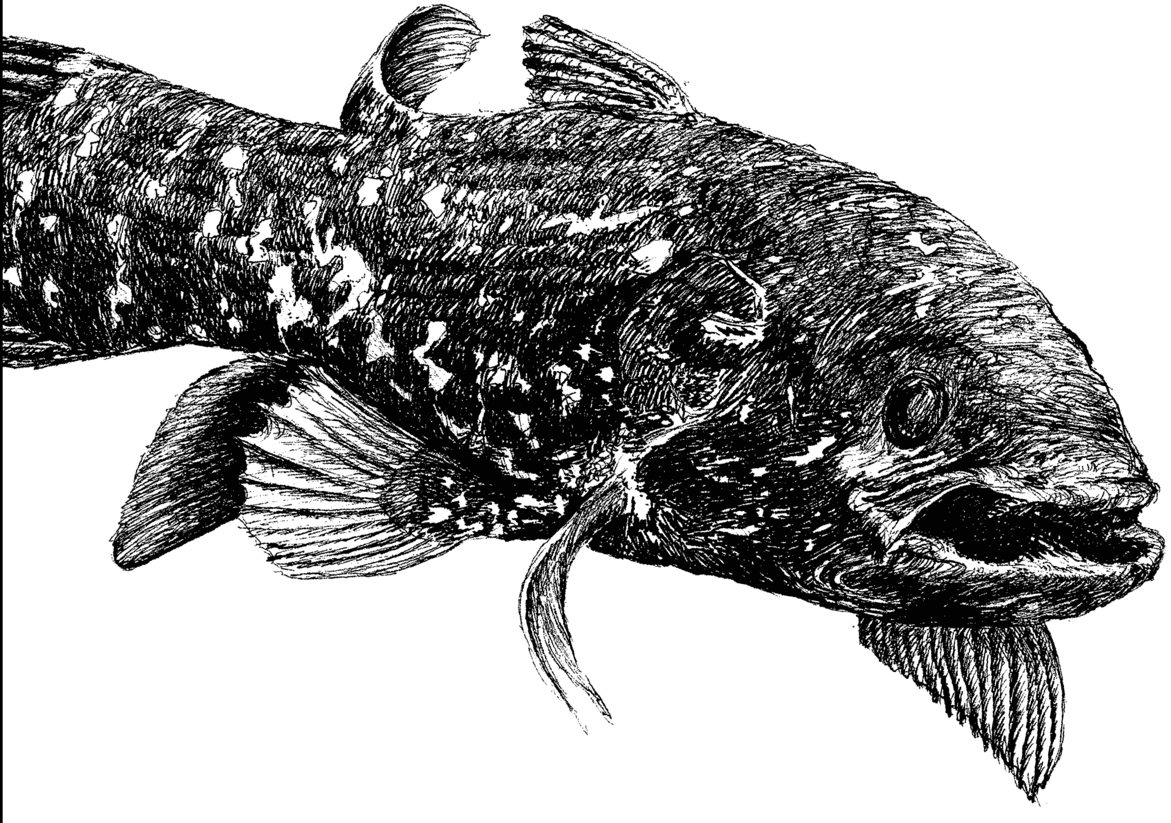
An Essential Guide to the Basic Local Alignment Search Tool

Foreword by Stephen Altschul

BLAST



O'REILLY®

Ian Korf, Mark Yandell & Joseph Bedell

BLAST

BLAST

Ian Korf, Mark Yandell, and Joseph Bedell

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

CHAPTER 4

Sequence Similarity

The fact that the human genome is often referred to as the Book of Life is an apt description because nucleic acids and proteins are often represented (and manipulated) as text files. Chapter 3 described common algorithms for aligning sequences of letters, and *score* is the metric used to determine the best alignment. This chapter shows what scores really are. Some of the introduced terms come from information theory, so the chapter begins with a brief introduction to this branch of mathematics. It then explores the typical ways to measure sequence similarity. You'll see that this approach fits well with the sequence-alignment algorithms described in Chapter 3. The last part of the chapter focuses on the statistical significance of sequence similarity in a database search. The theories discussed in this chapter apply only to local alignment. There is currently no theory for global alignment.

Introduction to Information Theory

In common usage, the word *information* conveys many things. Forget everything you know about this word because you're going to learn the most precise definition. Information is a decrease in uncertainty. You can also think of information as a degree of surprise.

Suppose you're taking care of a child and the response to every question you ask is "no." The child is very predictable, and you are pretty certain of the answer the next time you ask a question. There's no surprise, no information, and no communication. If another child answers "yes" or "no" to some questions, you can communicate a little, but you can communicate more if her vocabulary was greater. If you ask "do you like ice cream," which most children do, you would be informed by either answer, but more surprised if the answer was "no." Qualitatively, you expect more information to be conveyed by a greater vocabulary and from surprising answers. Thus, the information or surprise of an answer is inversely proportional to its probability. Quantitatively, information is represented by either one of the following equivalent formulations shown in Equation 4-1.

$$H(p) = \log_2 \frac{1}{p} \qquad H(p) = -\log_2 p$$

Equation 4-1.

The information, *H*, associated with some probability *p*, is by convention the base 2 logarithm of the inverse of *p*. Values converted to base 2 logarithms are given the unit *bits*, which is a contraction of the words *binary* and *digit* (it is also common to use base *e*, and the corresponding unit is *nats*). For example, if the probability that a child doesn't like ice cream is 0.25, this answer has 2 bits of information (liking ice cream has 0.41 bits).

It is typical to describe information as a *message* of *symbols* emitted from a *source*. For example, tossing a coin is a source of head and tail symbols, and a message of such symbols might be:

```
tththttt
```

Similarly, the numbers 1, 2, 3, 4, 5, and 6 are symbols emitted from a six-sided die source, and the letters A, C, G, and T are emitted from a DNA source. The symbols emitted by a source have a frequency distribution. If there are *n* symbols and the frequency distribution is flat, as it is for a fair coin or die, the probability of any particular symbol is simply 1/*n*. It follows that the information of any symbol is $\log_2(n)$, and this value is also the average. The formal name for the average information per symbol is *entropy*.

But what if all symbols aren't equally probable? To compute the entropy, you need to weigh the information of each symbol by its probability of occurring. This formulation, known as Shannon's Entropy (named after Claude Shannon), is shown in Equation 4-2.

$$H = -\sum_i^n p_i \log_2 p_i$$

Equation 4-2.

Entropy (*H*) is the negative sum over all the symbols (*n*) of the probability of a symbol (p_i) multiplied by the log base 2 of the probability of a symbol ($log_2 p_i$). Let's work through a couple of examples to make this clear. Start with the flip of a coin and assume that *h* and *t* each have a probability 0.5 and therefore a $\log_2$ probability of –1. The entropy of a coin is therefore:

```
- ( (0.5)(-1) + (0.5)(-1) ) = 1 bit
```

Suppose you have a trick coin that comes up heads 3/4 of the time. Since you're a little more certain of the outcome, you expect the entropy to decrease. The entropy of your trick coin is:

```
- ( (0.75)(-0.415) + (0.25)(-2) ) = 0.81 bits
```

A random DNA source has an entropy of:

```
- ( (0.25)(-2) + (0.25)(-2) + (0.25)(-2) + (0.25)(-2) ) = 2 bits
```

However, a DNA source that emits 90 percent A or T and 10 percent G or C has an entropy of:

```
- ( 2(0.45)(-1.15) + 2(0.05)(-4.32) ) = 1.47 bits
```

In these examples, you've been given the frequency distribution as some kind of truth. But it's rare to know such things *a priori*, and the parameters must be estimated from actual data. You may find the following Perl program informative and entertaining. It calculates the entropy of any file.

```
# Shannon Entropy Calculator
my %Count;      # stores the counts of each symbol
my $total = 0;  # total symbols counted

while (<>) {                          # read lines of input
    foreach my $char (split(//, $_)) { # split the line into characters
        $Count{$char}++;              # add one to this character count
        $total++;                     # add one to total counts
    }
}

my $H = 0;                         # H is the entropy
foreach my $char (keys %Count) {    # iterate through characters
    my $p = $Count{$char}/$total;   # probability of character
    $H += $p * log($p);             # p * log(p)
}
$H = -$H/log(2);                   # negate sum, convert base e to base 2
print "H = $H bits\n";             # output
```

Amino Acid Similarity

Molecular biologists usually think of amino acid similarity in terms of chemical similarity (see Table 2-1). Figure 4-1 depicts a rough qualitative categorization. From an evolutionary standpoint, you expect mutations that radically change chemical properties to be rare because they may end up destroying the protein's three-dimensional structure. Conversely, changes between similar amino acids should happen relatively frequently.

In the late '60s and early '70s, Margaret Dayhoff pioneered quantitative techniques for measuring amino acid similarity. Using sequences that were available at the time, she constructed multiple alignments of related proteins and compared the frequencies of amino acid substitutions. As expected, there is quite a bit of variation in amino acid substitution frequency, and the patterns are generally what you'd expect from the chemical properties. For example, phenylalanine (F) is most frequently paired to itself. It is also found relatively frequently with tyrosine (Y) and tryptophan (W), which share similar aromatic ring structures (see Table 2-1), and to a lesser

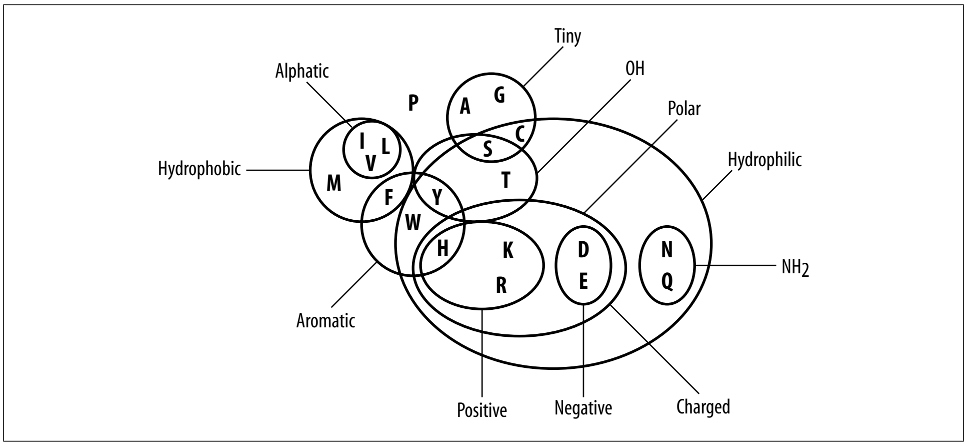


Figure 4-1. Amino acid chemical relationships

extent with the other hydrophobic amino acids (M, V, I, and L). Phenylalanine is infrequently paired with hydrophilic amino acids (R, K, D, E, and others). You can see some of these patterns in the following multiple alignment, which corresponds to a portion of the *cytochrome b* protein from various organisms.

```
PGNPFATPLEILPEWYLYPVFQILRVLPNKLLGIACQGAIPLGLMMVPFIE
PANPFATPLEILPEWYFYPVFQILRTVPNKLLGVLAMAAVPVGLLTVPFIE
PANPMSTPAHIVPEWYFLPVYAILRSIPNKLGGVAAIGLVFVSLLALPFIN
PANPLVTPPHIKPEWYFLFAYAILRSIPNKLGGVLALLFSILMLLLVPFLH
PANPLSTPAHIKPEWYFLFAYAILRSIPNKLGGVLALLLSILVLIFIPMLQ
PANPLSTPPHIKPEWYFLFAYAILRSIPNKLGGVLALLLSILILIFIPMLQ
IANPMNTPTHIKPEWYFLFAYSILRAIPNKLGGVIGLVMSILIL..YIMIF
ESDPMMSPVHIVPEWYFLFAYAILRAIPNKVLGVVSLFASILVL..VVFVL
IVDTLKTSDKILPEWFFLYLFGFLKAIPDKFMGLFLMVILLFSL..FLFIL
```

Dayhoff represented the similarity between amino acids as a $\log_2$ odds ratio, also known as a *lod score*. To derive the lod score of an amino acid, take the $\log_2$ of the ratio of a pairing's observed frequency divided by the pairing's random expected frequency. If the observed and expected frequencies are equal, the lod score is zero. A positive score indicates that a pair of letters is common, while a negative score indicates an unlikely pairing. The general formula for any pair of amino acids is shown in Equation 4-3.

$$S_{ij} = \log(q_{ij}/p_i p_j)$$

Equation 4-3.

The score of two amino acids *i* and *j*, is s_{ij}, their individual probabilities are p_i and p_j, and their frequency of pairing is q_{ij}. For example, suppose the frequencies of methionine (M) and leucine (L) in your data set are 0.01 and 0.1, respectively. By random pairing, you expect 1/1000 amino acid pairs to be M-L. If the observed frequency of pairing is 1/500, the odds ratio is 2/1. Converting this to a base 2 logarithm gives a lod score of +1, or 1 bit. Similarly, if the frequency of arginine (R) is 0.1 and its frequency of pairing with L is 1/500, the lod score of an R-L pair is -2.322 bits. In computers, using base e rather than base 2 is more convenient. The values of +1 and -2.322 bits are 0.6931 and -1.609 nats, respectively.

If you know the direction of change from an evolutionary tree, the pair-wise scores can be asymmetric. That is, the score of M-L and L-M may not be equal. For simplicity, the direction of evolution is usually ignored, though, and the scores are symmetrical.

Scoring Matrices

A two-dimensional matrix containing all possible pair-wise amino acid scores is called a *scoring matrix*. Scoring matrices are also called substitution matrices because the scores represent relative rates of evolutionary substitutions. Scoring matrices are evolution in a nutshell. Take a moment now to peruse the scoring matrix in Figure 4-2 and compare it to the chemical groupings in Figure 4-1.

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

Figure 4-2. BLOSUM62 scoring matrix

Lod scores are real numbers but are usually represented as integers in text files and computer programs. To retain precision, the scores are generally multiplied by some scaling factor before converting them to integers. For example, a lod score of -1.609 nats may be scaled by a factor of two and then rounded off to an integer value of -3. Scores that have been scaled and converted to integers have a unitless quantity and are called *raw scores*.

PAM and BLOSUM Matrices

Two different kinds of amino acid scoring matrices, PAM (Percent Accepted Mutation) and BLOSUM (BLOcks SUbstitution Matrix), are in wide use. The PAM matrices were created by Margaret Dayhoff and coworkers and are thus sometimes referred to as the Dayhoff matrices. These scoring matrices have a strong theoretical component and make a few evolutionary assumptions. The BLOSUM matrices, on the other hand, are more empirical and derive from a larger data set. Most researchers today prefer to use BLOSUM matrices because *in silico* experiments indicate that searches employing BLOSUM matrices have higher sensitivity.

There are several PAM matrices, each one with a numeric suffix. The PAM1 matrix was constructed with a set of proteins that were all 85 percent or more identical to one another. The other matrices in the PAM set were then constructed by multiplying the PAM1 matrix by itself: 100 times for the PAM100; 160 times for the PAM160; and so on, in an attempt to model the course of sequence evolution. Though highly theoretical (and somewhat suspect), it is certainly a reasonable approach. There was little protein sequence data in the 1970s when these matrices were created, so this approach was a good way to extrapolate to larger distances.

Protein databases contained many more sequences by the 1990s so a more empirical approach was possible. The BLOSUM matrices were constructed by extracting ungapped segments, or *blocks,* from a set of multiply aligned protein families, and then further clustering these blocks on the basis of their percent identity. The blocks used to derive the BLOSUM62 matrix, for example, all have at least 62 percent identity to some other member of the block.

Why, then, are the BLOSUM matrices better than the PAM matrices with respect to BLAST? One possible answer is that the extrapolation employed in PAM matrices magnifies small errors in the mutation probabilities for short evolutionary time periods. Another possibility is that the forces governing sequence evolution over short evolutionary times are different from those shaping sequences over longer intervals, and you can't estimate distant substitution frequencies without alignments from distantly related proteins.

Target Frequencies, lambda, and H

The most important property of a scoring matrix is its *target frequencies* and the expected frequencies of the individual amino acid pairs. Target frequencies represent the underlying evolutionary model. While scoring matrices don't actually contain the target frequencies, they are implicit in the scores.

The Karlin-Altschul statistical theory on which BLAST is based (discussed in the next section) states that all scoring schemes for which a positive score is possible (and the expected score is negative) have implicit target frequencies. Thus they are lod-odds scoring schemes; even a simple "+1 match -1 mismatch" scheme is implicitly a log-odds scoring scheme and has target frequencies. You'll learn how to calculate those frequencies in just a bit, but you first need to understand two additional concepts associated with scoring schemes: lambda and relative entropy.

Lambda

Raw score can be a misleading quantity because scaling factors are arbitrary. A *normalized score*, corresponding to the original lod score, is therefore a more useful measure. Converting a raw score to a normalized score requires a matrix-specific constant called *lambda* (or λ). Lambda is approximately the inverse of the original scaling factor, but its value may be slightly different due to integer rounding errors. Let's now derive lambda.

When calculating target frequencies from multiple alignments, the sum of all target frequencies naturally sums to 1 (see Equation 4-4).

$$\sum_{i=1}^{n} \sum_{j=1}^{i} q_{ij} = 1$$

Equation 4-4.

Recall from Equation 4-3 that the score of two amino acids is the log-odds ratio of the observed and expected frequencies. The same equation is presented in Equation 4-5, but the lod score is replaced by the product of lambda and the raw score (in other words, lambda had a value of 1 in Equation 4-3).

$$\lambda S_{ij} = \log_e(q_{ij}/p_i p_j)$$

Equation 4-5.

Equation 4-6 rearranges Equation 4-5 to solve for pair-wise frequency.

$$q_{ij} = p_i p_j e^{\lambda s_{ij}}$$

Equation 4-6.

From Equation 4-6, you can see that a pair-wise frequency (q_{ij}) is implied from individual amino acid frequencies (p_i and p_j) and a normalized score (λS_{ij}). The key to solving for lambda is to provide the individual amino acid frequencies (p_i and p_j) and find a value for lambda where the sum of the implied target frequencies equals one. The formulation is given in Equation 4-7 and later in Example 4-1.

$$\sum_{i=1}^{n}\sum_{j=1}^{i} q_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{i} p_i p_j e^{\lambda s_{ij}} = 1$$

Equation 4-7.

Normally, once lambda is estimated, it is used to calculate the Expect of every HSP in the BLAST report. Unfortunately, the residue frequencies of some proteins deviate widely from the residue frequencies used to construct the original scoring matrix. Recently, some versions of PSI-BLAST and BLASTP have therefore begun to use the query and subject sequence amino acid compositions to calculate a *composition-based lambda***.** These "hit-specific" lambdas have been shown to improve BLAST sensitivity, so this approach may see wider use in the near future.

Relative Entropy

The *expected score* of a scoring matrix is the sum of its raw scores weighted by their frequencies of occurrence (see Equation 4-8). The expected score should always be negative.

$$E = \sum_{i=1}^{20}\sum_{j=1}^{i} p_i p_j S_{ij}$$

Equation 4-8.

The *relative entropy* of a scoring matrix (*H*) conveniently summarizes the general behavior of a scoring matrix. Its formulation is similar to the expected score (see Equation 4-9) but is calculated from normalized scores. *H* is the average number of bits (or nats) per position in an alignment and is always positive.

$$H = -\sum_{i=1}^{20}\sum_{j=1}^{i} q_{ij}\lambda S_{ij}$$

Equation 4-9.

H of PAM1 is greater than the *H* PAM120. Recall that the PAM120 matrix is derived from mutation probabilities for PAM1 extrapolated to 120 PAMs. The PAM120 matrix is therefore less specific, contains less information, and thus has a lower *H*. Similarly, BLOSUM80 has a greater *H* than BLOSUM62. This makes sense since BLOSUM80 was made from sequences that were more similar to one another than BLOSUM62.

Which PAM matrix is most similar to BLOSUM45? To answer this, you only need to determine the PAM matrix with an *H* closest to that of the BLOSUM45 matrix. By relative entropy, PAM250 is closest to BLOSUM45, PAM120 to BLOSUM80, and PAM180 to BLOSUM62.

Match-Mismatch Scoring

Now let's determine the target frequencies of a +1/-1 scoring scheme. We will explore this in the case of DNA alignments where match/mismatch scoring is frequently employed. For generality, assume that all nucleotide frequencies are equal to 0.25. This fixes the previous p_i and p_j terms. Example 4-1 shows a Perl script that contains an implementation for estimating lambda by making increasingly refined guesses at its value. Table 4-1 displays the expected score, lambda, *H*, and the expected percent identity for several nucleotide scoring schemes. Note that the match/mismatch ratio determines *H* and percent identity. As the ratio approaches 0, lambda approaches 2 bits, and the target frequency approaches 100 percent identity. Intuitively, this makes sense; if the mismatch score is -∞, all alignments have 100 percent identity, and observing an A is the same as observing an A-A pair.

Table 4-1. Nucleotide scoring schemes

Match	Mismatch	Expected score	λ (bits)	H (bits)	% ID
+10	-10	-5	0.158	0.793	75
+1	-1	-0.5	1.58	0.791	75
+1	-2	-1.25	1.92	1.62	95
+1	-3	-2	1.98	1.89	99
+5	-4	-1.75	0.277	0.519	65

Example 4-1. A Perl script for estimating lambda

```
#!/usr/bin/perl -w
use strict;
use constant Pn => 0.25; # probability of any nucleotide

die "usage: $0 <match> <mismatch>\n" unless @ARGV == 2;
my ($match, $mismatch) = @ARGV;
my $expected_score = $match * 0.25 + $mismatch * 0.75;
die "illegal scores\n" if $match <= 0 or $expected_score >= 0;

# calculate lambda
my ($lambda, $high, $low) = (1, 2, 0); # initial estimates
while ($high - $low > 0.001) {         # precision

    # calculate the sum of all normalized scores
    my $sum = Pn * Pn * exp($lambda * $match)    * 4
            + Pn * Pn * exp($lambda * $mismatch) * 12;

    # refine guess at lambda
    if ($sum > 1) {
        $high = $lambda;
```

Example 4-1. A Perl script for estimating lambda (continued)

```
        $lambda = ($lambda + $low)/2;
    }
    else {
        $low = $lambda;
        $lambda = ($lambda + $high)/2;
    }
}

# compute target frequency and H
my $targetID = Pn * Pn * exp($lambda * $match) * 4;
my $H = $lambda * $match    *     $targetID
      + $lambda * $mismatch * (1 -$targetID);

# output
print "expscore: $expected_score\n";
print "lambda:   $lambda nats (", $lambda/log(2), " bits)\n";
print "H:        $H nats (", $H/log(2), " bits)\n";
print "%ID:      ", $targetID * 100, "\n";
```

Sequence Similarity

Sequence similarity is a simple extension of amino acid or nucleotide similarity. To determine it, sum up the individual pair-wise scores in an alignment. For example, the raw score of the following BLAST alignment under the BLOSUM62 matrix is 72. Converting 72 to a normalized score is as simple as multiplying by lambda. (Note that for BLAST statistical calculations, the normalized score is λS – lnk.)

```
Query:   885 QCPVCHKKYSNALVLQQHIRLHTGE 909
             +C VC K ++    L++H RLHTGE
Sbjct:   267 ECDVCSKSFTTKYFLKKHKRLHTGE 291
```

Recall from Chapter 3 that the score of each pair of letters is considered independently from the rest of the alignment. This is the same idea. There is a convenient synergy between alignment algorithms and alignment scores. However, when treating the letters independently of one another, you lose contextual information. Can you assume that the probability of A followed by G is the same as the probability of G followed by A? In a natural language such as English, you know that this doesn't make sense. In English, Q is always followed by U. If you treat these letters independently, you lose this restriction. The context rules for biological sequences aren't as strict as for English, but there are tendencies. For example, low entropy sequences appear by chance much more frequently than expected. To avoid becoming sidetracked by the details, accept that you're using an approximation, and note that in practice, it works well.

Karlin-Altschul Statistics

In 1990, Samuel Karlin and Stephen Altschul published a theory of local alignment statistics. Karlin-Altschul statistics make five central assumptions:

- A positive score must be possible.
- The expected score must be negative.
- The letters of the sequences are independent and identically distributed (IID).
- The sequences are infinitely long.
- Alignments don't contain gaps.

The first two assumptions are true for any scoring matrix estimated from real data. The last three assumptions are problematic because biological sequences have context dependencies, aren't infinitely long, and are frequently aligned with gaps. You now know that both alignment and sequence similarity assume independence, and that this is a necessary convenience. You will soon see how sequence length and gaps can be accounted for. For now, though, let's turn to the Karlin-Altschul equation (see Equation 4-10):

$$E = kmne^{-\lambda S}$$

Equation 4-10.

This equation states that the number of alignments expected by chance (*E*) during a sequence database search is a function of the size of the search space (*m***n*), the normalized score (λ*S*), and a minor constant (*k*).

In a database search, the size of the search space is simply the product of the number of letters in the query (*m*) and the number of letters in the database (*n*). A small adjustment (*k*) takes into account the fact that optimal local alignment scores for alignments that start at different places in the two sequences may be highly correlated. For example, a high-scoring alignment starting at residues 1, 1 implies a pretty high alignment score for an alignment starting at residues 2, 2 as well. The value of *k* is usually around 0.1, and its impact on the statistics of alignment scores is relatively minor, so don't bother with its derivation. According to Equation 4-10 the relationship between the expected number of alignments and the search space (*mn*) is linear. If the size of the search space is doubled, the expected number of alignments with a particular score also doubles. The relationship between the expected number of alignments and score is exponential. This means that small changes in score can lead to large differences in *E*.

Gapped Alignments

In practice, gaps reduce the stringency of a scoring scheme. In other words, an alignment score of 30 occurs more often in collection of gapped alignments than it does in a similar collection of ungapped alignments. How much more often depends on the cost of the gaps relative to the scoring matrix values. To determine the statistical significance of gapped alignments with Karlin-Altschul statistics (Equation 4-10), you must find values for lambda, *k*, and *H* for a particular scoring matrix and its associated gap initiation and extension costs. Unfortunately, you can't do this analytically, and the values must be estimated empirically. The procedure involves aligning random sequences with a specific scoring scheme and observing the alignment properties (scores, target frequencies, and lengths). The ungapped scoring matrix whose behavior is most similar to the gapped scoring scheme provides values for lambda, *k*, and *H*.

In the Karlin-Altschul theory, the distribution of alignment scores follows an extreme value distribution, a distribution that looks similar to a normal (Gaussian) distribution but falls off more quickly on one side and more slowly on the other side. Experiments show that gapped alignment scores fit the extreme value distribution as well. This fit is important because it strongly suggests that applying empirically derived values for lambda, *k*, and *H* to gapped alignment is statistically valid. Table 4-2 shows how much the parameters change by allowing gaps given the BLOSUM62 scoring matrix (also see Appendixes A and C).

Table 4-2. Effect of gaps on BLOSUM62

Gap open	Gap extend	λ	k	H (nats)
No gaps allowed	No gaps allowed	0.318	0.134	0.40
11	2	0.297	0.082	0.27
10	2	0.291	0.075	0.23
7	2	0.239	0.027	0.10

Length Correction

The Karlin-Altschul equation (Equation 4-10) gives the search space between two sequences as *m*n*, but not all this space can be effectively explored because some portion of it lies at either end of the sequences. As discussed in Chapter 5, BLAST operates by extending seeds in the alignment space. It can't effectively extend seeds near the ends of the sequences, though, because it runs out of room.

Karlin-Altschul statistics provides a way to calculate just how long a sequence must be before it can produce an alignment with a significant Expect. This minimum length, *l*, is usually referred to as the *expected HSP length* (see Equation 4-11)

$l = \ln(kmn)/H$

Equation 4-11.

Note that the expected HSP (high scoring pair) length is dependent on the search space (m*n) and the relative entropy of the scoring scheme, H, so it varies from search to search.

To take edge effects into account when calculating an Expect, the expected HSP length is subtracted from the actual length of the query, m, and the actual number of residues in the database, n, to give their *effective* lengths, usually denoted by m' and n', respectively (see Equations 4-12 and 4-13).

$m' = m - l$

Equation 4-12.

$n' = n - (l \cdot \text{number\_of\_squences\_in\_db})$

Equation 4-13.

In a large search space, the expected HSP length may be greater than the length of the query, resulting in a negative effective length, m'. In practice, if the effective length is less than $1/k$, it is set to $1/k$, as doing so cancels the contribution of the short sequence to the Expect; setting $m' = 1/k$ for example, gives $E = n'e^{-\lambda S}$, a formulation independent of m'.

Unfortunately, effective lengths of less than $1/k$ aren't uncommon today. Because $l \alpha n$, the large size on many sequence databases can result in large expected HSP lengths. In fact it's not uncommon to see expected HSP lengths approaching 200 when searching some of the larger protein databases. Keep in mind that the average protein is ~300 amino acids long; thus, for many searches, m' is being set to $1/k$ routinely. Recent work by S.F. Altschul and colleagues has suggested that part of the problem may be that Equation 4-11 overestimates l. They have proposed another way to calculate this value that results in shorter effective HSP lengths. Thus, the method used to calculate l may change in the not so distant future.

Sum Statistics and Sum Scores

BLAST uses Equation 4-14 to calculate the normalized score of an individual HSP, but it uses a different function to calculate the normalized score of group of HSPs (see Chapter 7 for more information about sum statistics).

$S'_{nats} = \lambda S - \ln k$

Equation 4-14.

Before tackling the actual method used by BLAST to calculate a sum score, it's helpful to consider the problem from a general perspective. One simple and intuitive approach for calculating a sum score might be to sum the raw scores, S_r for a set of HSPs, and then convert that sum to a normalized score by multiplying by λ, or in mathematical terms (see Equation 4-15).

$$S'_{sum} = \lambda \sum_{i=1}^{r} S_r$$

Equation 4-15.

The problem with such an approach is that summing the scores for a collection of *r* HSPs, always results in a higher score, even if none or those HSPs has a significant score on its own. In practice, BLAST controls for this by penalizing the sum score by a factor proportional to the product of the number of HSPs, *r*, and the search space as shown in Equation 4-16.

$$S'_{sum} = \lambda \sum_{i=1}^{r} S_r - r\ln(kmn)$$

Equation 4-16.

Equation 4-16 is sometimes referred to as an *unordered-sum score* and is suitable for calculating the sum score for a collection of noncollinear HSPs. Ideally, though, you should use a sum score formulation that rewards a collection of HSPs if they are collinear with regards to their query and subject coordinates because the HSPs comprising real BLAST hits often have this property. BLASTX hits for example often consist of collinear HSPs corresponding to the sequential exons of a gene. Equation 4-17 is a sum score formulation that does just that.

$$S'_{sum} = \lambda \sum_{i=1}^{r} S_r - r\ln(kmn) + \ln(r!)$$

Equation 4-17.

Equation 4-18 is sometimes referred to as a *pair-wise ordered sum score*. Note the additional term $\ln r!$, which can be thought of as a bonus added to the sum score when the HSPs under consideration are all consistently ordered.

One shortcoming of Equations 4-16 and 4-17 is that they invoke a sizable penalty for adding an additional HSP raw score to the sum score. To improve the sensitivity of its sum statistics, NCBI-BLASTX employs a modified version of the pair-wise ordered sum score (Equation 4-17) that is influenced less by the search space and contains a term for the size of the gaps between the HSPs (Equation 4-18). The advantage of this formulation is that the gap size, *g*, rather than the search space, *mn*, is multiplied by *r*. For short gaps and big search spaces, this formulation results in larger sum scores.

$$S'_{sum} = \lambda \sum_{i=1}^{r} S_r - \ln(kmn) - (r-1) \cdot (\ln(k) + 2\ln(g)) - \log(r!)$$

Equation 4-18.

Converting a Sum Score to a Sum Probability

The aggregate pair-wise P-value for a sum score can be approximated using Equation 4-19.

$$P_r \approx e^{-S_{sum}} S_{sum}^{r-1} / r!(r-1)!$$

Equation 4-19.

Thus, when sum statistics are being employed, BLAST not only uses a different score, it also uses a different formula to convert that score into a probability—the standard Karlin-Altschul equation $E = kmne^{-\lambda S}$ (Equation 4-10) isn't used to convert a sum score to an Expect.

BLAST groups a set of HSPs only if their aggregate P-value is less than the P-value of any individual member, and that group is an optimal partition such that no other grouping might result in a lower P-value. Obviously, finding these optimal groupings of HSPs requires many significance tests. It is common practice in the statistical world to multiply a P-value associated with a significant discovery by some number proportional to the number of tests performed in the course of its discovery to give a test corrected P-value. The correction function used by BLAST for this purpose is given in Equation 4-20. The resulting value, P'_r is a *pair-wise test-corrected sum-P*.

$$P'_r = P_r / \beta^{r-1}(1-\beta)$$

Equation 4-20.

In this equation, β is the gap decay constant (its value can be found in the footer of a standard BLAST report).

The final step in assigning an E-value to a group of HSPs to adjust the pair-wise test-corrected sum-P for the size of the database The formula used by NCBI-BLAST (Equation 4-21) divides the effective length of the database by the actual length of the particular database sequence in the alignment and then multiplies the pair-wise test-corrected sum-P by the result.

$$\text{Expect}(r) = (\text{effective\_db\_length}/n)P'_r$$

Equation 4-21.

NCBI-BLAST and WU-BLAST compute combined statistical significance a little differently. The previous descriptions apply to NCBI-BLAST only. The two programs probably have many similarities, but the specific formulations for WU-BLAST are unpublished.

Probability Versus Expectation

While NCBI-BLAST reports an Expect, WU-BLAST reports both the E-value and a P-value. An E-value tells you how many alignments with a given score are expected by chance. A P-value tells you how often you can expect to see such an alignment. These measures are interchangeable using Equations 4-22 and 4-23.

$$P = 1 - e^{-E}$$

Equation 4-22.

$$E = -\ln(1 - P)$$

Equation 4-23.

For values of less than 0.001, the E-value and P-value are essentially identical.

Further Reading

Altschul, S.F. (1991). "Amino acid substitution matrices from an information theoretic perspective." *Journal of Molecular Biology*, Vol. 219, pp. 555-565.

Altschul, S.F. (1997). "Evaluating the statistical significance of multiple distinct local alignments." *Theoretical and Computational Methods in Genome Research*, S. Suhai (ed.), pp. 1-14.

Altschul, S.F. (1993). "A protein alignment scoring system sensitive at all evolutionary distances." *Journal of Molecular Evolution*, Vol. 36, pp. 290-300.

Altschul, S.F., M.S. Boguski, W. Gish, and J.C. Wootton (1994). "Issues in searching molecular sequence databases." *Nature Genet.*, Vol. 6, pp. 119-129.

Altschul S.F., R. Bundschuh, R. Olsen, T. Hwa (2001) "The estimation of statistical parameters for local alignment score distributions." *Nucleic Acids Research*, January 15;29(2), pp. 351-361.

Altschul, S.F. and W. Gish (1996). "Local alignment statistics." *Meth. Enzymol.*, Vol. 266, pp. 460-480.

Dayhoff, M.O., R.M. Schwartz, and B.C. Orcutt (1978). "A model of evolutionary change in proteins." *Atlas of Protein Sequence and Structure*, Vol. 5, Suppl. 3. M. O. Dayhoff (ed.), National Biomedical Research Foundation, pp. 345-352.

Henikoff, S. and J.G. Henikoff (1992). "Amino acid substitution matrices from protein blocks." *Proceedings of the National Academy of Sciences*, Vol. 89, pp. 10915-10919.

Karlin, S. and S.F. Altschul (1993). "Applications and statistics for multiple high-scoring segments in molecular sequences." *Proceedings of the National Academy of Sciences*, Vol. 90, pp. 5873-5877.

Karlin, S. and S. F. Altschul (1990). "Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes." *Proceedings of the National Academy of Sciences*, Vol. 87, pp. 2264-2268.

Schaffer A.A., Aravind L., Madden TL., Shavirin S., Spouge JL., Wolf YI., Koonin EV., Altschul SF. (2001). "Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements." *Nucleic Acids Research*, July 15;29(14), pp. 2994-3005.

Schwartz, R.M. and M.O. Dayhoff (1978). "Matrices for detecting distant relationships." *Atlas of Protein Sequence and Structure*, Vol. 5, Suppl. 3. M.O. Dayhoff (ed.), National Biomedical Research Foundation, pp. 353-358.