MCB182 Class Notes: Local Alignment Statistics

Aligning Proteins: Previously we aligned sequences using a +1 match -1 mismatch -1 gap scoring scheme. In practice, this match/mismatch scoring scheme is only used for aligning nucleotides. For aligning proteins, we use scoring matrices like BLOSUM62 to take into account that alignment often preserves chemical properties.

Sequence similarity: To determine protein similarity we simply align two proteins and sum up the amino acid scores. In principle, we could determine similarity scores from local or global alignments. In practice, we use local alignment only. One reason for this is that there is no established procedure for determining global alignment significance.

Alignment scores: What does an alignment score mean? Is a score of 30 *good*? Does 30 mean the proteins are homologous or functionally related? What if the scores in the matrix were scaled by 10 vs 5? Is a score of 100 necessarily better than 50?

Significance: In typical frequentist statistics, one accepts or rejects an hypothesis based on some random model. For local alignments, we use the same idea. Given an alignment score, we would like to know *how often such a score would be expected to occur at random.* If the score is easily attained at random, then it is probably not very significant.

Equation 7

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

E: number of alignments
k: a constant
M: size of sequence 1
N: size of sequence 2
e: 2.7182818...
λ: scaling factor
S: score of alignment

Karlin-Altschul statistics: Local alignment statistics were formalized by Karlin & Altschul using information theoretic methods. Given certain assumptions (see box) the K-A equation (equation 7) tells you how often such a score (or higher) is expected at random. For some intuition in this, imagine comparing two books to see if they have similar sentences. If the books are very short, you don't expect may similar sentences. Conversely, if the books were gigantic, you would expect to find many more similar sentences. The product MN is called the search space, and the number of expected alignments varies linearly with the size of the space. Now imagine that you have a threshold score for what you accept as similar sentences. If you ask for a higher score, you will find fewer sentences. The K-A equation shows that this is an inverse exponential relationship. In other words, a small change increase in score can lead to a large reduction in the number of alignments expected at random. The fact that λ is in the exponent indicates that E is also highly dependent on its value. λ is effectively the inverse of the scaling factor used to create the matrix (but not exactly due to rounding). In other words, λ turns the matrix score into a log-odds score. Now we can begin to answer the questions we previously posed. Is a score of 30 good? It depends on the search space. In a large search space, 30 may be expected at random, but it might be highly significant in a small search space. Is a score of 100 better than 50? If the only difference is the scaling factor, then the significance is the same because λ will normalize them to the same bit score.

Karlin-Altschul Assumptions
1.A positive score must be possible
2.Expected score of matrix must be negative
3.Sequences are infinitely long
4.Letters are independent and identically distributed
5.Alignments do not contain gaps

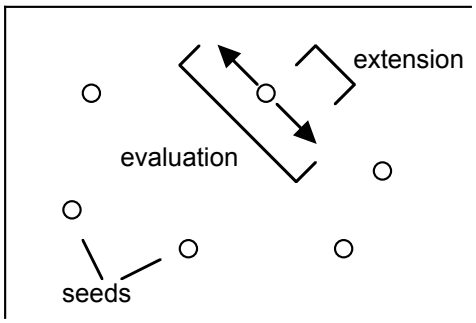
K-A issues: Let's take a closer look at the K-A assumptions. #1 and #2 are true of any scoring matrix derived from multiple alignments. But we can also make up an arbitrary scoring scheme such as our original +1/-1 match/mismatch scheme. Is this legal? What would happen if the scheme was +2/0? What about -1/-2? What about +10/-1? When might +1/-1 be illegal? #3 is only a problem when sequences are very short. To deal with this problem, people consider the search space to be smaller in each dimension by log(kMN)/H, which is the length of the expected random alignment. #4 states that letters are independent and identically distributed. In other words, the probability of finding a sequence such as AAA is simply the product of finding A cubed. Does this make sense? Not really considering that genomes and proteins contain a lot of repeats. #5 disallows gaps. But we know S-W alignments can contain gaps. We will return to the gap problem in a bit.

Lambda revisited: In order to compute E, we need λ for our scoring scheme. We might know this value ahead of time if we created our own scoring matrix, but if someone else created it, or we used a system like +1/-1, we need to be able to derive λ somehow. λ cannot be solved for algebraically, but we can estimate its value to arbitrary precision.

$S_{ij} = \log\left(\frac{Q_{ij}}{P_i P_j}\right)$	The usual equation for the score of any amino acid pair.
$\lambda S_{ij} = \log\left(\frac{Q_{ij}}{P_i P_j}\right)$	λ is the inverse of the scaling factor used when the matrix was scaled and rounded off. When scores are in bits, λ = 1.
$e^{\lambda S_{ij}} = \frac{Q_{ij}}{P_i P_j}$	Exponentiate each side of the equation.
$Q_{ij} = P_i P_j e^{\lambda S_{ij}}$	This is the most important part. It shows that an observed pairing frequency is *implied* given the marginal compositions and a scoring scheme.
$\sum\sum Q_{ij} = 1$	By definition all observed pairing frequencies sum to 1.0
$\sum\sum = P_i P_j e^{\lambda S_{ij}}$	We can solve for λ by making refined guesses at its value. If our guess is too high, the sum will be > 1. If it is too low, the sum will be < 1.

It turns out that our old +1/-1 scoring system implies a pairing frequency of about 75% given that the 4 nucleotides have 25% frequency. If we had started with nucleotide alignments containing about 75% identity, and the marginal nucleotide frequencies were all 25%, we would have ended up with a +1/-1 scoring system. +1/-1 does not imply 75% identity in proteins however.

Gaps revisited: So what do we do about gaps? Gaps make it *easier* to align two sequences. Therefore, gaps effectively reduce H. To account for this in the K-A equation, we can simply decrease λ, and this will decrease the bit score of the alignment and therefore increase the E value. How much we reduce λ depends on the specific match, mismatch, and gap penalties. It is not possible to compute these adjustments algebraically, so they are computed via simulation.



BLAST: One of the most famous and popular bioinformatics applications is BLAST (Basic Local Alignment Search Tool). This combines sequence alignment and statistical evaluation in a single, efficient program. BLAST is similar to S-W in principle: both are local alignment algorithms. But BLAST is much faster because it does not explore the entire search space. There are 3 steps to the BLAST algorithm: (1) seeding (2) extension (3) evaluation. In the seeding phase, regions containing identical (or highly similar) strings are identified. These *points* in the *space* are expected to contain the good local alignments. In the extension phase, each seed undergoes a S-W-like alignment, but the extension stops if the alignment quality degrades too much. In the evaluation phase, the alignment is subjected to the K-A equation to determine how often the alignment is expected by chance. If the E value is less than some user-defined threshold, then the alignment is reported.

Program	Database	Query	Example
BLASTN	DNA	DNA	Align mRNA to genome
BLASTP	AA	AA	Search for proteins related to ____
BLASTX	AA	DNA	Find coding exons in a BAC
TBLASTN	DNA	AA	Search for transcripts similar to ____
TBLASTX	DNA	DNA	Find orthologous coding exons