Sequence Alignment, cont.

5. Heuristic Algorithms. BLAST.

ALthough Smith-Waterman is a fast method for aligning two sequences optimally, in practice, when the reference sequence is, in fact, the entire GenBank database, for example, this is still too long for large scale use.

SW guarantees the exact soluiton to the optimization problem. A *heuristic* algorithm is one which will search for this optimum, but is not guaranteed to find the best solution. For pairwise sequence alignment, the standard heuristic version of SW is BLAST, which we have already used once. Some information, mostly practical, is available through the BLAST tutorial:

`http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/tut1.html`

I will only give an outline of the BLAST technique. We will talk about the underlying statistics. The basic idea is that in a long, ideal

sequence alignment, there seem to be shorter streetches of high scoring alignment, or high scoring matchess (HSM's). BLAST in effect preprocesses the *query* sequence, looking for "neighborhood words" which might match the query sequence very well over a short stretch. The neighborhood words are 11 positions long for n.a. data, and 3 for a.a. data (i.e., this is the default setting). The list is determined by thresholding, and the thresholding is measured by a score density cutoff. So, if we normalize our probabilistic scores to be measured in bitrs, the cutoff is usally about 2 bits per position. (For n.a., this compares to an abstract maximum of 4.)

Next, BLAST searches the database using the list of these neighborhood words, and pulls out sequences which hav a HSM with one of them. There follows a step which is called "hit extension". Here the HSM is extended in both directions to a maximal scoring hit. This procedure is also heuristic, though seems very good in practice.

As we have seen, BLAST then reports a list of the best scoring matches which it has found, together with the associated alignments, and an E-value. (See below.)

Originally BLAST found matches without gaps, but now does. There is now currently available a morepowerful tool PSI-BLAST, wher PSI = Position-dependent Scoring Iterative. We will be using Hidden Markov Models as a method for achieving position dependent scoring. This amounts to a change of the underlying probability model of the sequences one is comparing.

6. Scoring Statistics.

We will use at first the simplest measures of significance for a score, the p-value.

For a random variable $\mathbf{X}$, we say that a value of $\mathbf{X}$ ("score") has p-value c if

$$Prob(\mathbf{X} > c) \leq p.$$

This measures whether the score is an outlying value, at the high end (there are two-sided versions of this, of course). So, if we want to know how significant it is that our sequence alignment has the score it got, a first measure is this p-value. Lower p-value = more significant.

p-values can be obtained two ways: analytically and by simulation. In either case, one has to have a null model, or reference model.

In the analytic case, one has to solve at least approximately, for the distribution function of $\mathbf{X}$ and its properties, which is rarely done. (Main example for us: Karlin-Altschul statistics for ungapped BLAST.)

In the case of simulation, one relies on a pseudorandom number generator to simulate draws from the score distribution. Thus one would sample the distribution of scores from a random model of sequence generation. Here one

would generate sequences randomly (PRNG) and then calculate the scores. One assumes the sample distribution converges to the distribution of scores. This is how p-values are computed for the SW algorithm scores in the USC software.

The behavior of such scores is a rather subtle matter. Here is an example which began the analytical work on the subject.

Consider flipping a coin, where H appears with bias p. Suppose we flip the coin n times, and we ask ourselves the question: what is the longest run of heads we shall see, if we track the sequence of H's and T's? This is a very simple model of what we are trying to do in sequence comparison: if we look for the longest run of heads, then we are comparing our sequence (query sequence) from the experiment, say

$$H, H, T, H, \ldots, T, H, T.$$

to the reference sequence

$$H, H, H, \ldots, H, H.$$

We score with 1 for a match and $-\infty$ for a mismatch, and no gaps are allowed. (I.e., $s(a, -) = -\infty$.) Then the maximum score will be exactly the length of the longest match. Let $\mathbf{X}$ be this random variable, the "length of the longest match".

It is clear that the answer to this problem should depend on the length n and the bias p. The basic result (from around *1960*!) is due to Erdős and Renyi:

$$E(\mathbf{X}) \approx \log_{1/p} n.$$

Remark: this is somewhat higher than what intuition tells us. This is useful for gambling situations, but has led to some misestimations in the sequence comparison literature.

What makes this a difficult problem is that, although our scores our computed against a null

model of independent positions for sequence score increments, the RV we are looking out involves grouping sveral positions together. Thus if we think of moving from left to right in the sequence, and we know that we have a match of length ℓ, and then we shift over 1 positon, there is a much higher than usual probability that we will have a match of length ℓ beginning at the second position than if we were to take a position at random without knowing what the previous ℓ positions looked like.