COS 551: Introduction to Computational Molecular Biology
Lecture 5: September 28, 2000
Lecturer: Mona Singh
Scribe: Neil Pravin Shah '01

Multiple Sequence Alignments II

A common heuristic for multiple sequence alignments is the method of progressive alignments. The following describes the general algorithm for progressive alignments, but the exact implementation details are omitted as they continually evolve.

ClustalW and Pileup are two alignment packages that are based on progressive alignments. Roughly speaking, progressive alignment methods, such as ClustalW, do the following:

1. Determine all pairwise alignments between sequences and determine degrees of similarity between each pair.
2. Construct a "rough" similarity tree
3. Combine the alignments starting from the most closely related groups to most distantly related groups, while maintaining the "once a gap, always a gap" policy.

We illustrate the above algorithm with an example. Given k sequences, $\{s_1, s_2, \ldots, s_k\}$, we want to find an alignment of these k sequences.

Step 1: *Determine all pairwise alignments between sequences and determine degrees of similarity between each pair.*

a. We compute pairwise alignments as we learned in the previous lecture on global sequence alignments.
b. We use these pairwise alignments to compute a "distance" between all pairs of sequences. One method to assign distances is the following. For each pairwise alignment, look at the non-gapped positions and count the number of differences per site. For example, the best alignment for the two sequences in Figure 1 has a distance of 1/4 = .25 for the mismatch between the M and V.

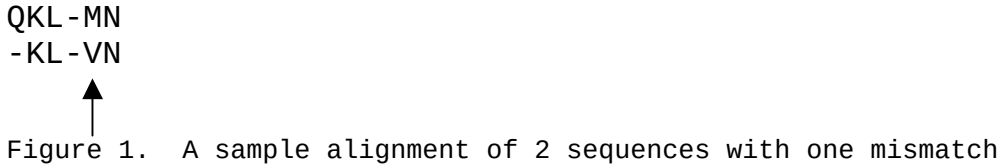

Figure 1. A sample alignment of 2 sequences with one mismatch

Note: The number of observed substitutions underestimates the number of actual substitutions; we will consider this problem in a later lecture.

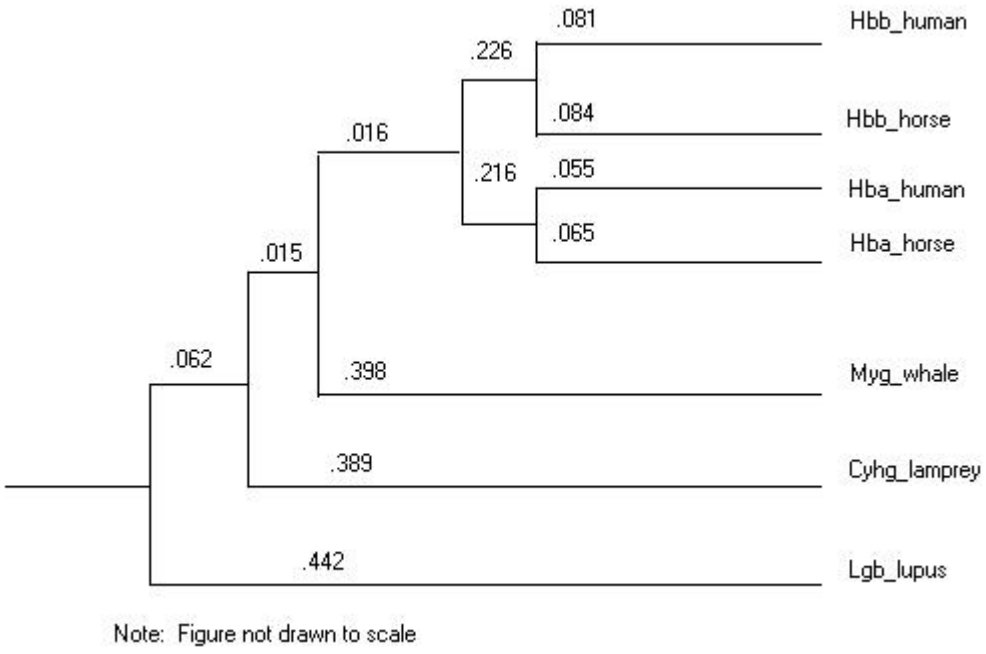

Figure 2. "Rough" tree showing similarities between various globin proteins.

After computing the pairwise distances, we input them into a matrix. Note that we have computed distances to be in the range of 0 to 1, with smaller values indicating more closely related sequences.

Table 1 is an example for globin protein data, with k=7 sequences.

Globin type		1	2	3	4	5	6	7
Hbb_human	1	-						
Hbb_horse	2	.17	-					
Hba_human	3	.59	.60	-				
Hba_horse	4	.59	.59	.13	-			
Myg_whale	5	.77	.77	.75	.75	-		
Cyng_lamprey	6	.81	.82	.73	.74	.80	-	
Lgb_lupus	7	.87	.86	.86	.88	.93	.90	-

Table 1. Globin similarity data.

We can neglect to fill in the rest of the matrix since it is symmetric.

Step 2: *Construct a "rough" similarity tree*

We now construct a tree that is based on the above distance matrix. The exact details of tree construction will be discussed in a later lecture. The ClustalW software uses the neighbor joining (NJ) method to compute this tree.

Figure 2 is an example of a tree for our data. In an ideal case, the distances between elements in the above matrix is exactly equivalent to the sum of the branch lengths between these elements in the tree. However, this case is rarely possible. In Figure 2, we see that the distance between Hbb_human and Hbb_horse in the tree is .081 + .084 = .165 which is close to .17 from the matrix in Table 1.

Step 3: *Combine the alignments starting from the most closely related groups to most distantly related groups, while maintaining the "once a gap, always a gap" policy.*

We build our alignment using the tree as a guide, and proceed from its tips to its root. We start by aligning Hbb_human and Hbb_horse. Then, we align Hba_human and Hba_horse. Now we combine the alignments of the Hbb's and the Hba's. We continue to do this up the tree until we reach the root.

As we combine pairwise, we are forcing gaps in the alignments via the "once a gap, always a gap" policy. We align each *pair of sequences* via the Needleman-Wunsch method (see previous lectures) with an affine gap penalty. That is, we charge a smaller penalty for a gap continuation than for a gap initiation. Note, we must also align *alignments* with other *alignments*. As with pairwise alignments, we do this via dynamic programming, but here the score in each cell of the sim matrix uses the average of all pairwise scores from the 2 sets of sequences used in the 2 alignments. For example, suppose we have the following two alignments, one of 2 sequences and the other of 4 sequences:

```
Alignment 1:      ATA
                  CCA

Alignment 2:      TCAFE
                  TAT-E
                  TATF-
                  AGTFD
```

We would score the first column of the first alignment against the second column in the other alignments using:

```
= 1/8(score(A,C) + score(A,A) + score(A,A) + score(A,G)+
      score(C,C) + score(C,A) + score(C,A) + score(C,G))
```

Here score(A,C) is assigned by aligning A against C; other scores are assigned similarly. For example, in the examples studied in previous lectures, these scores come from match=+1, mismatch =-1, and a gap=-2.

Additional Issues

Sequence Weighting. By giving each sequence equal weighting we are not taking into account any evolutionary relationships. Two sequences that are closely related should receive less

weight than two sequences that are less closely related. The closely related sequences contain duplicate information so we should not give too much weight to this type of data.

A common method to overcome this issue is implemented as follows. As our tree above has distances between each sequence, we can weight the alignments based on the tree distances.

a. We use the lengths from the length from root to sequences to compute weights. Therefore, there is increased weight for more divergent species.
b. If two or more sequences share a branch, the length of the branch is split amongst the sequences. This reduces the weight for related sequences.

In our tree from Figure 2, therefore, Lgb_lupus would get a weight of .442. The Hba_human would get a weight of .194

```
W(Hba_human) = .055+ .216/2+.061/4+ .015/5+ .062/6 = .194
```

Usually, the weights are normalized for each sequence with the maximum weight set to 1. Now, these weights are used to adjust the scores of the alignments.

Caveats.

a. There are no guarantees with respect to the alignment. For example, the alignment found this way says nothing about the optimum MSA with respect to the sum-of-pairs measure we talked about earlier.
b. The initial errors from the "once a gap, always a gap" policy are propagated and compounded.
c. There is more than one optimum pairwise alignment possible, yet we are committing ourselves to only one at the outset.
d. The order in which we add sequences to the alignment (e.g. based on the guide tree) affects the alignment.
e. Parameter settings are always a problem in alignment
   (1). Which matrix should one use? We should use a different matrix for scoring the alignments based on how closely related the sequences are.
   (2). How should we handle gaps?
f. If any pair of sequences are less than 25% identical, then the alignments are prone to be bad.
g. In general, one needs to correct some alignments manually.

Substitution Matrices

In the previous lectures we have used simple scoring schemes based on matches and mismatches. These are often used for DNA alignments (e.g., BLAST's default is match = +1 and mismatch = -3). However, for proteins we can increase sensitivity to weak alignments through better scoring schemes that take into account how particular amino acids substitute for each other. In particular, we know that certain amino acids substitute more easily for one amino acid than another. This is because certain amino acids have similar biochemical properties, such as hydrophobicity, size, or charge.

Using the correct protein substitution matrix when aligning is important for the following reasons:

a. Scoring matrices appear in all analysis involving sequence comparison.
b. The choice of a matrix influences the outcome of the analysis.
c. The scoring matrices implicitly represent a particular theory of evolution.
d. Different sets of values may be desired for comparing very similar as opposed to highly divergent sequences.

Protein substitution matrices can be derived from the following data:

a. Physical/chemical characteristics of the protein sequences, but this is difficult to assess.
b. Comparing how easy it is for simple base changes in DNA sequence codons (triplet of DNA) to create a change in amino acid. For example, GAA codes for glutamic acid while GGA codes for glycine.
c. The most common method, however, is to just conduct direct observations of sequences that are known to be aligned properly. The PAM and BLOSUM matrices are derived from these data.

We will hear more about substitution matrices in the next lecture.

Further Reading

Feng, D. F., Doolittle R. F. "Progressive sequence alignment as a prerequisite to correct phylogenetic trees." *Journal Mol Evol* 1987; 25(4):351-360.

Higgins D. G., Thompson J. D., Gibson T. J. "Using CLUSTAL for multiple sequence alignments. *Methods Enzymol* 1996; 266:383-402.