Report on the Design, Implementation, and Outcomes of Ancestral Reconstruction.py

Jazz Weisman
Jazzlw@gmail.com

16th December, 2015

This program (Ancestral Reconstruction.py) creates an ancestral tree for a hypothetical protein. Starting with a common ancestor sequence, the program creates a binary tree of hypothetical descendants, with a mutation rate specified, and a depth of levels also specified. The user is prompted for these parameters. The program then tries to reconstruct the tree from just the final level. This is analogous to reconstructing a tree from many versions of a modern homologous protein to recreate the common ancestor. It then prints some statistics about both trees, and prints the two trees for comparison. By comparing the accuracy of reconstruction for different error rates, sequences lengths, and numbers of levels in the tree, some interesting insights about basic ancestral reconstruction can be gained.

The basic steps entailed are as follows. First, after getting parameters from the user, the program creates a starting sequence, either as a repeating sequence of the same amino acid, or as a random sequence, depending on user input. This sequence is then used to generate the tree of descendants. First two daughters are created from the starting sequence, with random mutations that occur, on average, once for as many amino acids as the number the user entered for that parameter (between 10 and 100 recommended). After this, two daughters are created from each of these original daughters, and this is repeated until the tree has the user specified number of levels. At this point the tree will have 2^{n-1} leaves, where n is the user specified value.

Once the tree is created, the final leaves are removed, and their sequences are made into a list. This list is shuffled, and becomes the beginnings of the new, reconstructed, tree. First, the best way to pair up these sequences

is determined, to obtain the most likely groupings to make the next level of the tree. This is done to minimize the total hamming distance, summed across all pairs. Once pairs are decided upon, the parent of each pair is made by determining the consensus sequence between them. Ties are broken by determining which of the two amino acids appears most often in the list of leaf sequences, and using that one. This is because more common amino acids are more likely to be part of the common ancestors. These new parent nodes are then linked with the children from which they were created, and the whole process is repeated to make the next level. This repeat continues until all of the levels have been created, ending with the single common ancestor.

After this, for both the original tree and the reconstructed one, a list of average hamming distances per level is printed, to give the user an idea of how far and how fast the sequences diverged. The two trees are then printed next to each other for comparison. When comparing, it is important to note that the reconstructed tree is not in the same order as the original. The levels should ideally consist of close to the same set of sequences within each level (same amount of indent), but they will likely be on different lines, as there are 2^n different equivalent positions in each level for level number n (starting with level 0 of course).

Initial testing has revealed some interesting conclusions about the nature of basic ancestral reconstruction. Overall, it tends to work pretty well! For example, a tree of 25 amino acid sequences with an error rate of one in ten and ten levels in the tree, yielding 512 leaves, the original common ancestor is predicted with only a few errors, despite the average hamming distance of the leaves from the ancestor being 14.5, or well over half of the total amino acids. From a couple experiments using 50 amino acid long sequences, the same error rate as above, and only 6 levels in the tree, which means only 32 leaves, there were only around 5 errors in the common ancestor. This despite an average hamming distance at the leaves of almost 20. More analysis is certainly available, and there is more that this program can teach us. Users are encouraged to try different settings and see what results.