John Burns

Computational Biology and Medicine

December 3, 2017

Tree of Life Analysis and Process

When I first began the assignment, when I would run the simulation the result would be two major categories in my hierarchy: fish and everything else. There would be groupings of highly related genomes (birds, rodents, and fish, etc.), but these grouping seem to be unordered. Some of these groupings were tight, yet others were separated by seeming unrelated species. For example, there were two different species of lamprey eel tucked between the orangutan and the wallaby. Another peculiar finding was that there was a small branch of unrelated animals (domestic cat, chimpanzee, sea lamprey, and bottle nosed dolphin) that had been grouped together (independent of the rest of the hierarchy) in their own branch. Also, there was a subtree of fish on the far left of the graph, and another small grouping of sharks and eels in a subtree on the far right of the main tree, with no direct connection one another. Basically, my dendrogram and tree of life were hot messes. I tried a variety of techniques to obtain a tree of life that is 1) readable and 2) accurate to a degree. Below a few of the things that I tried, achieved, and failed to implement in my tree of life.

Color Schemes:

Being a visual guy, my first action was to experiment with various color schemes so that I could best read my data. I settled on the reverse spectral color scheme. I then moved on to my scoring function.

Scoring Function:

I started with defining a proper scoring function. I read into the documentation for *python.nwalign* and *python.swalign* and they recommended some popular “match”, “mismatch”, and “gap” penalties. I started simply adjusting the penalties that had been provided in the sample code. The alignment documentations said that *match = 2* and *mismatch = -1* were good parameters to use for a scoring function. I had been using *match = 2*, *mismatch = -1*, and *gap = 0* and my dendrogram had no real pattern to it because my parameters were producing data that was too similar. I doubled the mismatch penalty and increased the match penalty by 1. This helped to clear up my data a little bit. Next, I increased the gap penalty to -3. I wanted to make sure that the gap penalty would be greater than the penalty for a mismatch because gaps interrupt the entire polymer chain, whereas mutations do not (via the class slides). This also helped to clear up my data and now I could start to see individual cells in the dendrogram starting to emerge. The next step was to implement the substitution matrix for nucleotide sequences (from the slides). I did this by making a big switch-case block within the scoring method that would check every pair of nucleotides and assign the scored value according to the purine-purine and pyrimidines-pyrimidines scoring matrix. This was another step in the right direction. I could clearly see the symmetry in the matrix with the stark diagonal running down the center. After this I did some more research into the alignment functions and notices that there were scoring functions built into the alignment libraries. At that time, I was using the Needleman-Wunsch global alignment algorithm to align the sequences. I wanted to try using the built-in scoring method from the *nwalign* library. I achieved this using the below code:

score = nw.score\_alignment(alignment[0], alignment[1], gap\_open=-5, gap\_extend=-2, matrix=’scoring\_matrix.txt’)

similarity \_matrix[x,y] = int(score)

I ran the alignment with the 2 sequences and then applied a gap penalty of -5 for opening a new gap and a gap penalty of extending an old gap of -2, and then applied the Nucleotide substitution matrix for mutations. Using the *nw.score\_alignment* method, one is given the opportunity to choose more niche and dynamic scoring penalties. Similar to mismatches, not all gaps should be treated equally. For example, it is a much costlier to open a gap than it is to extend a gap (i.e. add more than one gap at a time). So, I chose my gap penalty for extending a gap by choosing a penalty that was slightly greater than the penalty for a mismatch. I then chose my penalty for opening a new gap by doubling the gap extension penalty and then adding one more to the penalty. The built-in method aligned and scored that sequences with much more accuracy than my “DIY” scoring method did. It also allowed me to apply a substitution matrix that I could import from an external .txt file. The result was clearer dendrograms with less noise (greater degrees of accuracy) and better clustering because my similarity matrix reflected a more accurate degree of similarity between species. I decided to use this as my scoring function and then I began experimenting with different substitution matrices. Before realizing that in the assignment we were dealing with nucleotide sequences, I tried using the BLOSUM62 substitution matrix in my scoring function. This was a mistake because under further investigation, the BLOSUM62 matrix is meant to be used when aligning proteins, not nucleotide sequences. The result was a completely yellow dendrogram with no labels. The program did not crash, but my alignments were also not scored. In the end, I stuck with the built-in *nw.score\_alignment* method with defined gap penalties and a substitution matrix for the scoring metric. This seemed to be the cheapest and best option available to me.

Global Alignment:

Though there is a built-in global alignment method in the *nwalign* library, I wanted to see if I could implement my own. I searched online for some Needleman-Wunsch alignment algorithm source code and stripped it down. I wanted to see if I could create a 2D distance matrix s by calculating the cos()-similarity for each pair of sequences and then converting that into cos()-distance. I thought that passing a distance matrix into the *linkage* method would result in more accurate hierarchical clustering. I succeeded in implementing the algorithm, testing it on small sequences and checking the output matrix. When I tried to run it on one of the nucleotide sequences from the dataset and I received a “max depth of recursion” error. I did not want to give up so I took the recursive algorithm, memoized it, and then translated it into an iterative solution. Once again, I tried to run the treebuilder once more except using the iterative global alignment algorithm and I sat there for 20 minutes while my computer froze and decided to scrap the project. In the end, I stuck with the default *nwalign* global alignment method.

Local Alignment:

After failing to implement my own version of the NW global alignment algorithm, I wanted to try implementing a local alignment method and then use that alignment in my scoring function instead of the Needleman-Wunsch global alignment. I implemented local alignment using the Python module *swalign*. It can be installed via command line using ‘pip’ by name. I used the same scoring matrix for nucleotide sequences from the class slides and used it in my scoring scheme. I then ran through each sequence and performed the local alignment and then used a built-in *sw.score\_alignment* method to set the corresponding cell in the similarity matrix to the value of the scored alignment. The local alignment was drastically slower than the global alignment. I never actually got to see the finished results because the alignments were so slowly. The average speed was approximately one sequence every 1500 seconds, so a single alignment every 30 seconds. I did analyze the console output using dump(), and I was pleasantly surprised by the accuracy of the alignments. The local alignment algorithm produced alignments that were much more complete than the alignments produced by the NW global alignment algorithm. The wait was not worth it, so I defaulted back to the global alignment algorithm.

Linkage Method:

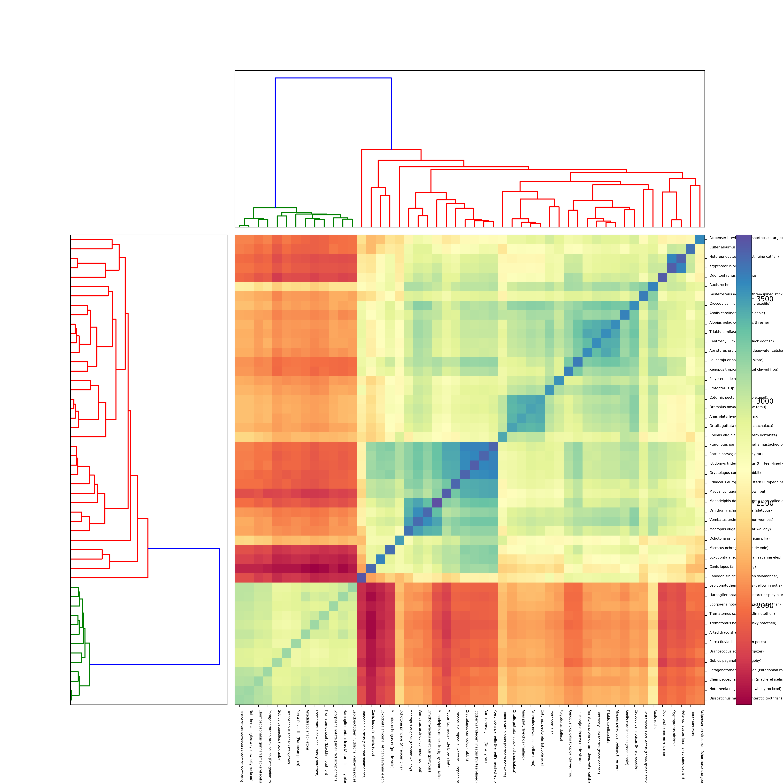
With my finely-tuned scoring scheme and my alignment algorithm picked out, it was time for me to begin thinking about how I was going to get the best hierarchical clustering. The clustering is done via the *linkage* method. This method converts a 2D matrix of observations (in our case “similarity” is our observation) and returns a hierarchical clustering as a linkage matrix. There are a variety of different methods that one can use to evaluate their input matrix and calculate their linkage matrix. I tried all of them to see which ones really gave me readable clustering. The “ward” or “variance minimization” clustering method did not work well. The grouping of species was alright, the mammals were together in the same subtree, the birds were together in the same subtree, the reptiles and amphibians were together in the same subtree. But there was this odd branch consisting of the bottlenose dolphin, a species of lamprey, the chimpanzee, and fruit flies. The “ward” clustering method did not seem to be the best option. Not only because of the unexpected outcome, also because the algorithm miniaturized the hierarchy, making it unreadable. Next, I used a different clustering algorithm. I used the “centroid” method. Similar clustering took place, but the branches were more centrally connected this time, which is a trait of the “centroid” clustering method. This did not improve the noise and overall issues of inconsistency in my dendrogram. I still had this rogue branch in my dendrogram that held a few unrelated species. The method that I found gave me the best clustering was the “weighted” clustering method. The tree was easy to read and the grouping of species was intuitive. The dendrogram was easy to understand and reflected the groupings very nicely. I included labeled dendrograms in a directory in the root folder, some of which are labeled by their clustering methods. I experimented a little bit with the linkage metric, but I found that the default “Euclidean” was the one that produced the clearest results for me. The metric that I experimented with the most was “cos”. This is because I attempted to convert the similarity matrix into a cosine sine similarity matrix. There is a strong correlation between similarity and distance but I did not achieve in implementing a version that worked. I included a graph in the ‘graphs’ directory (Dist\_matrix), and it shows my attempt to create a distance matrix out of the similarity matrix and pass it to the linkage method using a “cos” metric.

Noise:

After I found my clustering algorithm, I began noticing these pesky branches of individual species that did not seem to go together at all. There were a handful of species that shared little to no common connection with any other species in the tree and no matter what I tried, they still would not cooperate with the clustering. They were corrupting the integrity of my dendrogram and I decided to prune these problematic nodes. I wrote a small line of code to allow me to omit problematic species. This did not disrupt or taint the data because the pruned species were replaced with new ones. An example of a noisy node was the domestic cat. No matter the clustering algorithm, it would just get pushed to the end of the tree in its own branch with one or two other outliers, and never exceed over about fifty percent similarity with any other species. The result of the filtering process was a more cohesive tree without single species branches due to potentially bad data.

Results and Other Interesting finds:

Below is an image of my final dendrogram that best represents the genetic clustering of the species analyzed (located in ‘graphs’ directory as final\_weighted\_clustering.png). The thing I was most surprised about was the relation between sharks, reptiles, and birds. The website “Understanding Evolution” [https://evolution.berkeley.edu/evolibrary/article/evograms\_06] helped me to interpret the results of my alignment and scoring. As you can see, there are two main branches, fish, and the rest of the animals. It does seem a bit fishy though how in my clustering, rodents are more closely related to sharks than fish are.

 I wanted to see what the dendrogram would look like for types of bacteria. So, I filtered my search to pick only species of bacteria and the results were interesting. There were very few similarities amongst the bacteria aside from a small grouping of five or size, which were almost identical. From the dendrogram, it seems like there are much less genetic similarities in different species of bacteria than there are in animals. This means that there is a greater variety of bacteria. This can be clearly observed in the dendrogram (labeled bacteria).