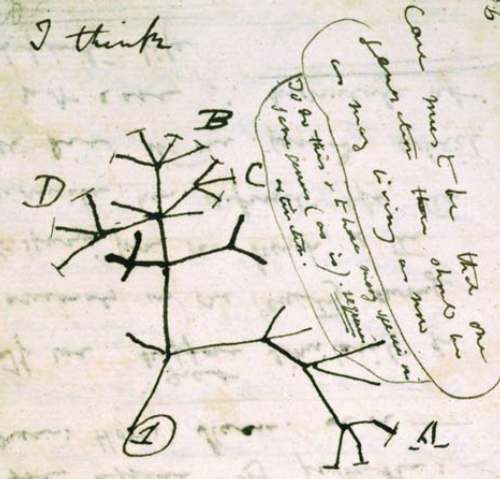
Sean Bird Cold Spring Harbor Laboratory Summary Report

Tucked away on the north shore of Long Island, Cold Spring Harbor Laboratory is a world-renowned research institute focusing on molecular biology and genetics with research programs in cancer, neuroscience, plant biology, and computational biology. In the summer of 2012, I spent three months at CSHL as an intern in the laboratory of Dan Levy, an Assistant Professor in quantitative biology. Dr. Levy’s group specializes in algorithm development and mathematical modeling with a particular focus on the genetics of sporadic disorders such as autism, childhood cancer, and congenital heart disease. With recent technological advances in computation and high throughput sequencing, genomic methods have become a standard tool in understanding basic biological processes and are now impacting virtually all areas of biology. Next generation sequencing technologies produce hundreds of millions of reads per run and making sense of that much information requires advanced algorithms and extensive computing power. Bioinformatics, the study of methods for storing, retrieving and analyzing biological data, is imperative to the advancement of biological sciences as a whole; as many new problems and challenges faced by professors today require the use of intensive computing power, and the knowledge to use them as efficiently as possible.

My project for the summer was to implement neighbor-joining, a popular algorithm for building phylogenetic trees from pairwise distances. Phylogenetic trees tell us how different organisms are related to each other. Since DNA accumulates small changes over time, we can use these mutations to map ancestral relationships.



Charles Darwin - First diagram of an evolutionary tree from his First Notebook on Transmutation of Species (1837).

In 1987, Saitou and Nei published the neighbor-joining algorithm1 which uses the mutational differences between each pair of organisms to reconstruct the original tree. I implemented my version of the algorithm in Python, an object-oriented programming language, widely used in the field of computational biology. My program would read in an input distance matrix, reconstruct the tree, and output the result in the Newick format, a compact text notation for representing trees. My input data was a matrix containing the total number of single nucleotide differences between each pair of individual values between 1,365 mitochondrial genomes. In total, I had 1,863,225 distances to construct a tree from. To check that my method was working properly, I also wrote code that reversed the process, parsing a Newick string into a tree and that tree into a distance matrix. If the algorithm is working well, neighbor joining should turn that distance matrix back into the original tree.

Mathematically, a tree is defined as a connected acyclic graph. That means that the tree contains no directed cycles, formed by a collection of vertices and directed edges such that there is no way to start at some vertex and follow a sequence of edges to eventually loop back to the original vertex. For my program, I represented the tree as a set of nodes, each having its own distance to its parent node, and all leading to a common root of the tree. Each node may itself have children, who may have children themselves, and so on. A node with no children is called a *leaf node*. Each node with descendants represents the most recent common ancestor to those descendants and the distance from an ancestor to its child represents a measure of time: the shorter the distance, the closer the relation. The root of a tree is the first ancestor, from which all the children originate. In my Python program, each node is represented as an object of class Node with information about the nodes name, the nodes distance to parent, a list of the nodes children (if any) and the nodes parent. A strip of code from Python, the Node class I defined looks like such:

class **Node**:

def **\_\_init\_\_**(*self*, name, distance, parent):

*self*.name = name # name of node, if a leaf

*self*.distance = distance # distance from node to parent

*self*.children = [] # list of children

*self*.parent = parent # link to parent node

The neighbor joining algorithm starts with an NxN distance matrix **D** and an equivocal star tree, a tree that contains many ambiguous nodes that have yet to be classified and ordered. We transform the distance matrix **D** to a matrix **Q**, find a minimum element, and join these two nodes. This adds an edge to the tree, two new edge lengths, and reduced D to an (N-1)x(N-1) matrix. We then iterate until the tree is completely resolved determining all splits and branch lengths. Neighbor joining is widely used because it is very fast (order N3) and robust against experimental noise. It can easily resolve trees with hundreds of thousands of taxa.

The heart of neighbor-joining is the **Q**-matrix; a modified distance matrix based on **D** in which the separation between each pair of nodes is adjusted on the basis of their average divergence from all other nodes. The Q-matrix is formed using the following formula:

*(Where i and j represent the rows and columns of matrix Q respectively, and d(i,j) is the distance between taxa i and j. N is the length of matrix D, and the summation of d(i,k) and d(j,k) are the ith and jth row sums.)*

We then find the pair of values in matrix **Q** that have the lowest value. This is to find the pair of nodes that are closest in relationship, the most likely to be siblings in our tree. We name these two nodes *f* and *g*. We join f and g to a new common ancestor node *u*. When the two nodes are linked, and have their common ancestral node added; the previous terminal nodes in the tree are removed from our distance matrix, replaced by the single node *u*. We then calculate the new distances from *f* and gto their parent node *u* using the following formula:

Once the distances between *u* and its children are known, we can then calculate the distance from *u* and the remaining nodes in the tree as follows:

*(Where k is any other node we are finding the distance to from u.)*

This describes one iteration of the neighbor joining algorithm. After one step, we have reduced the size of our distance matrix by one and resolved one relationship in our tree. The tree consisting of all remaining nodes and *u* is now an equivocal tree with N-1 elements, so we can repeat this process again and again until we have only two nodes left and a single distance measure between them. At the end, once every distance and relationship is determined, we are left with a single node, containing a plethora of descendants. This node becomes the “root” of the tree and contains information about its children which contain information about their children, etc. In this way, we recursively represent the entire tree. An example of the star tree neighbor joining starts with, and what its finished tree looks like after it has completed:

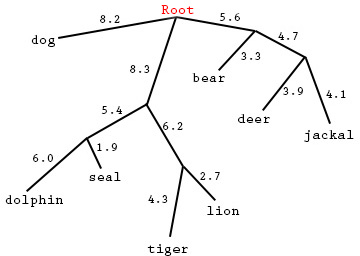
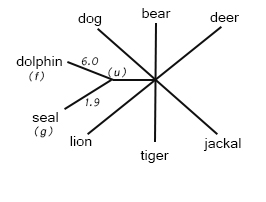
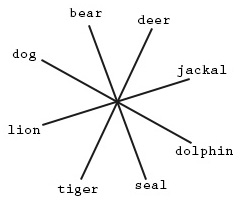
**

Figure 3: Finished, rooted phenetic tree

Figure 2: 1st iteration of neighbor joining

Figure 1: Unfinished, equivocal star tree

1st iteration..

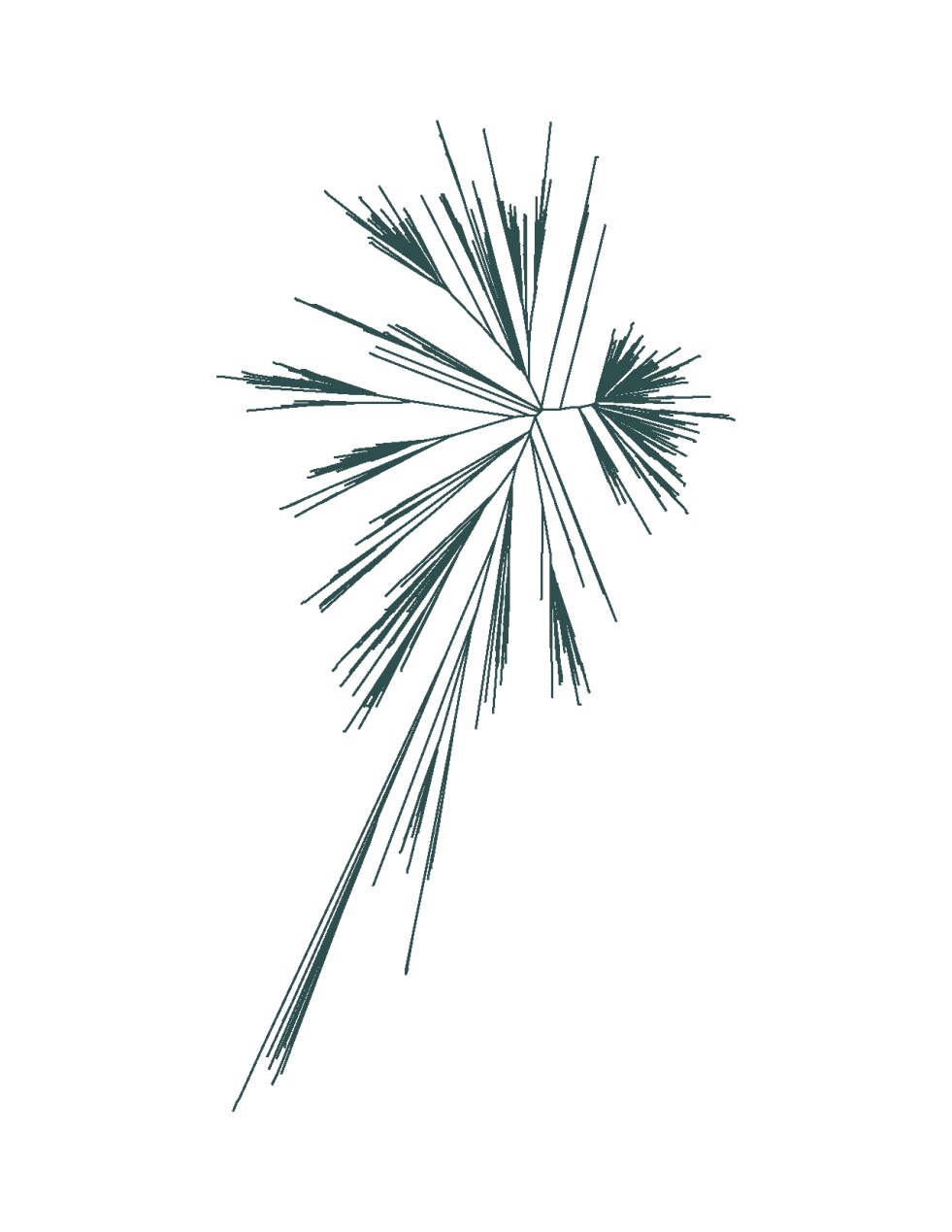
Final Iteration..

This root, which contains an entire family of children, each with distances to their parents and their own children, is how we represent the tree in the computer. The last step in my program tells the computer how to output this tree. A common standard for representing a phylogenetic tree is called the Newick format. In Newick notation, the example tree above looks like:

*(((deer:3.9,jackal:4.1):4.7,bear:3.3):5.6,((dolphin:6.0,seal:1.9):5.4,(tiger:4.3,lion:2.7):6.2):8.3,dog:8.2):0*

The names of the taxa are represented on the left side of the colons, and their corresponding edge lengths are represented on the right side. The commas separate the siblings. The inner nested parenthesis indicates parent/child relationships, with the distance directly outside the parenthesis representing those nodes distance to their parent. The zero at the end represents the root’s distance to itself, which is always zero. In order to convert my one rooted completed tree into Newick notation, I iteratively search through every node in the tree starting from the root, and check if the node is either a leaf node, or a parent node. If the node is a parent, it iterates through its children, and repeats the process of checking for leaf nodes, or parent nodes. All the while, it creates a set of nested parenthesis, carrying each node’s name and edge length. Once the tree is converted into Newick format, it can be read and displayed in any tree visualization software on a computer.

An important step in scientific computing is to validate your code. Even with a small mistake, my algorithm may return a tree; however, it is important that it return the *correct* tree. To confirm that everything is working as it should, I wrote an entirely separate set of methods for building a tree from the Newick format and building a distance matrix from a tree. If my neighbor joining code is correct, the tree derived from that distance matrix should match the original tree. An added benefit of writing both sides of the coin, is that the relationship between trees and distances is approached from complementary perspectives: how a tree gives rise to a distance matrix as well as how a distance matrix can be used to build a tree.

 The final test of my algorithm was to apply it to real data. The Levy lab is part of a large sequencing study of autism families. One incidental consequence of this dataset is some very high coverage data for thousands of mitochondrial genomes. Mitochondrion are the power stations of the eukaryotic cell and have their own DNA. The patterns of inheritance of mitochondrial genes are from the mother, this is called maternal inheritance. Mutations in the mitochondrial genes can therefore be traced through the mother. This is useful in the study of phylogenetics as we can trace mutations through families maternally, and display them through phenetic trees. Referring to Mitochondrial Eve, the matrilineal most recent common ancestor of modern humans, by definition all mitochondrial DNA in every living human is directly descended from hers. By counting the number of differences between each person’s mitochondrial genomes, we were able to construct our #x# distance matrix. I took this matrix as the input for my method and generated a phylogeny, similar to that of the common Tree of Life. The 1,863,225 distances I used to construct my tree, finalized and outputted, looks like this:

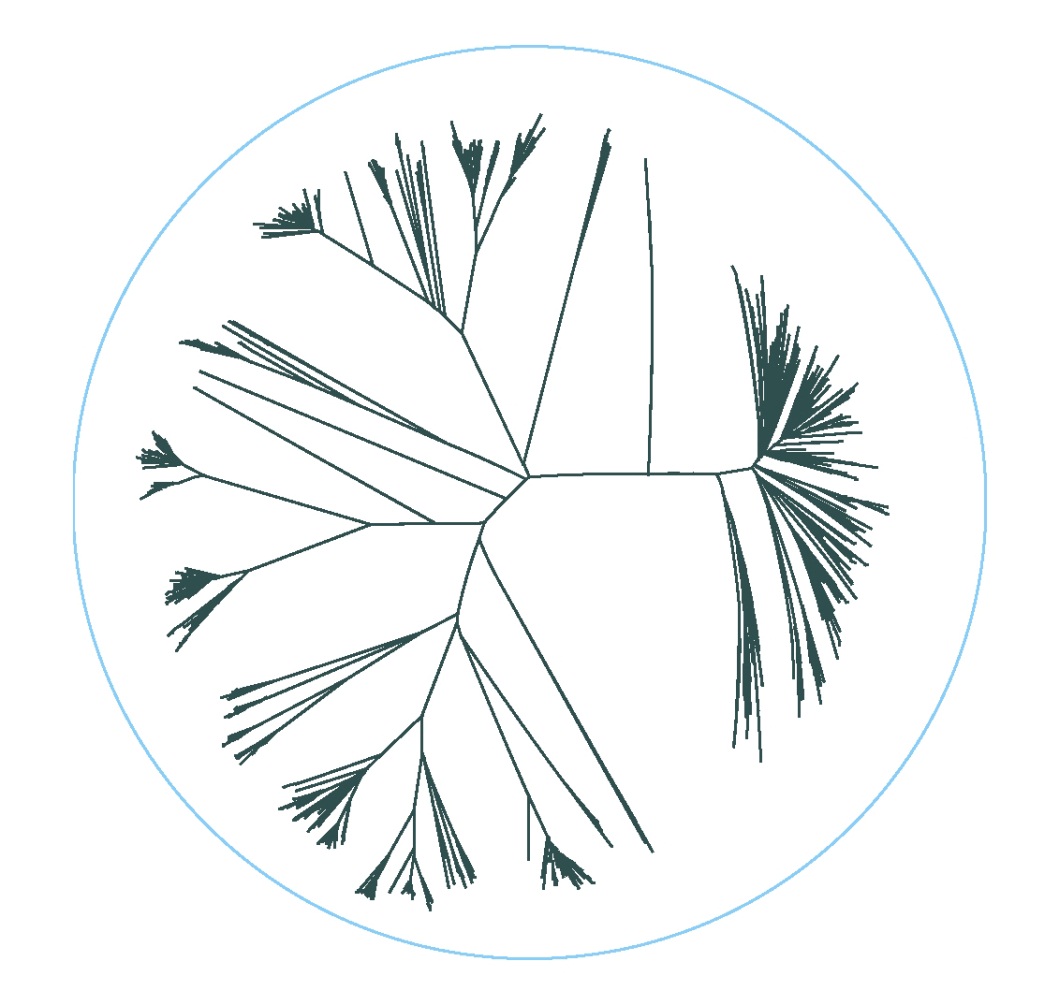


Figure : Hyperbolic Tree

Figure 5: Linear Tree

**References:**

1 Saitou, Naruya; Nei, Masatoshi. “The neighbor-joining method: a new method for reconstructing phylogenetic trees.” Oxford Journals: Molecular Biology and Evolution 4.4 (1987): 406-425.