Lecture Notes: Molecular Evolution and Phylogenetic Tree Reconstruction

Dana Wyman

February 23, 2016

1 Introduction

- Main topic of lecture: How to mathematically model the process of molecular evolution, or genome sequences changing over time.
- In a genetic sequence, new mutations may occur in the germline that are passed down to the next generation.
  - Humans have a very low rate of mutations per generation (50-100)
    - Increased maternal/paternal age may lead to more mutations being passed down
  - In this lecture, we will think of generations as very short, and model mutations as a continuous process over many generations
- We will discuss mathematical tools to understand branch length in trees

2 Molecular Evolution

- In the past, people used morphology to infer phylogenetic trees, grouping organisms according to observable characteristics
  - Major weakness: Convergent evolution has resulted in cases where organisms that are not closely related have independently evolved a similar physical trait. An example of this is the body shape of whales and fish, which has evolved independently in several lineages because it is efficient for movement through water.
- Another way: Use sequence comparison
  - Take some sequences, align all sequence pairs, and define a distance metric between the pairs. Then, try to cluster the sequences based on distance.
- Distances based on molecular evolution
  - Start with a base at a given position. Consider its change over time to be a continuous process with mutation rate μ.
  - Note that if the same genomic site undergoes multiple substitutions, some of the changes may mask others, and we won't be able to detect all of them.

- If the base in question is 'A', we can express its mutation rate μ_A as the sum of the rates at which the 'A' mutates to each of the other bases:

$$\mu_A = \mu_{AC} + \mu_{AT} + \mu_{AG}$$

- Consider what happens to this base over the infinitesimally small interval $t + \Delta t$
  - We assume it is not possible to have an initial substitution event and then a later reversal of that substitution, as the interval t + Δt is so small.
  - The probability of the given base being 'A' at $t + \Delta t$ is

$$p_A(t + \Delta t) = p_A(t) - p_A(t)\mu_A\Delta t + p_C(t)\mu_{CA}\Delta t + p_G(t)\mu_{GA}\Delta t + p_T(t)\mu_{TA}\Delta t$$

  - $p_A(t), p_C(t), p_G(t)$, and $p_T(t)$ indicate the probability of starting with base A, C, G, or T at time t, respectively.
  - The term $p_A - p_A(t)\mu_A\Delta t$ represents the probability of starting with base 'A' and then mutating away from it.
  - The other terms represent the probability of starting with a base other than 'A', and then mutating to 'A'
- The expression can also be written in matrix/vector notation:

$$P(t + \Delta t) = P(t) + QP(t)\Delta t$$

  As differential equation

$$P'(t) = QP(t)$$

  - P(t) = vector of probabilities of {A,C,G,T} at time t
  - Q = substitution rate matrix, which contains all the rates at which each base changes to the other bases or stays the same.

$$Q = \begin{pmatrix} -\mu_A & \mu_{GA} & \mu_{CA} & \mu_{TA} \\ \mu_{AG} & -\mu_G & \mu_{CG} & \mu_{TG} \\ \mu_{AC} & \mu_{GC} & -\mu_C & \mu_{TC} \\ \mu_{AT} & \mu_{GT} & \mu_{CT} & -\mu_T \end{pmatrix}$$

  - Q gives us the probability distribution over {A,C,G,T} at each position. Each Q is an evolutionary model, and many of these have been defined previously by population geneticists.

- Evolutionary Models
  - Should have relatively few parameters
  - Should consider the fact that bases do not change to every other base at the same rate
    - Transition: a purine nucleotide is changed to another purine (A $<->$ G) or a pyrimidine nucleotide to another pyrimidine (C $<->$ T)
    - Transversion: a purine is changed to a pyrimidine, or vice versa
    - Transitions are more common because they involve changing between chemically similar bases
  - Jukes-Cantor Model

* The simplest of the models we will discuss
* Every base substitution has the same rate, which as we discussed is not really true in reality

$$Q = \begin{pmatrix} * & \frac{\mu}{4} & \frac{\mu}{4} & \frac{\mu}{4} \\ \frac{\mu}{4} & * & \frac{\mu}{4} & \frac{\mu}{4} \\ \frac{\mu}{4} & \frac{\mu}{4} & * & \frac{\mu}{4} \\ \frac{\mu}{4} & \frac{\mu}{4} & \frac{\mu}{4} & * \end{pmatrix}$$

– Kimura Model

* Most frequently used across the literature
* Separate rate defined for transitions ($\kappa > 1$) and transversions ($= 1$)

$$Q = \begin{pmatrix} * & \kappa & 1 & 1 \\ \kappa & * & 1 & 1 \\ 1 & 1 & * & \kappa \\ 1 & 1 & \kappa & * \end{pmatrix}$$

– Felsenstein Model

* Incorporates the fact that in a given genome, the frequencies of each base may be different
* This can be modeled by having separate rates of change between the different bases

$$Q = \begin{pmatrix} * & \pi_T & \pi_T & \pi_T \\ \pi_C & * & \pi_C & \pi_C \\ \pi_A & \pi_A & * & \pi_A \\ \pi_G & \pi_G & \pi_G & * \end{pmatrix}$$

– Hasegawa, Kishino and Yano (HKY) Model

* Combines Felsenstein and Kimura models (handles transitions and transversions differently, and also accounts for unequal nucleotide frequencies)

$$Q = \begin{pmatrix} * & \kappa\pi_T & \pi_T & \pi_T \\ \kappa\pi_C & * & \pi_C & \pi_C \\ \pi_A & \pi_A & * & \kappa\pi_A \\ \pi_G & \pi_G & \kappa\pi_G & * \end{pmatrix}$$

- The Jukes-Cantor Evolutionary Model in More Detail

– Starting with the differential equation $P'(t) = QP(t)$, we can solve for the probability that a base is different between two sequences, or solve for the expected evolutionary time at which they will be different.

– Let's say we start at a position at time point 0 and want to know which base is there at the future time point t. We can derive a formula for this using the differential equation:

* The Jukes-Cantor model assumes that each base has the same frequency in the genome, so the probability of the base remaining the same from time 0 to t will be equal across all letters. We call this probability r(t)

$$r(t) = P_{AA} = P_{CC} = P_{TT} = P_{GG}$$

$$r(t) = 1 - p$$

where p = percent of sites that differ between two sequences

* Let s(t) = the probability of a base substitution at a particular position during time 0 to t. The Jukes-Cantor model assumes that all base-base changes are equally likely:

$$s(t) = P_{AC} = \ldots = P_{TG}$$

* We get the following matrix of probabilities showing the probability of each base-base transition in time t:

$$P = \begin{pmatrix} \frac{1}{4} + \frac{3}{4}e^{-t\mu} & \frac{1}{4} - \frac{1}{4}e^{-t\mu} & \frac{1}{4} - \frac{1}{4}e^{-t\mu} & \frac{1}{4} - \frac{1}{4}e^{-t\mu} \\ \frac{1}{4} - \frac{1}{4}e^{-t\mu} & \frac{1}{4} + \frac{3}{4}e^{-t\mu} & \frac{1}{4} - \frac{1}{4}e^{-t\mu} & \frac{1}{4} - \frac{1}{4}e^{-t\mu} \\ \frac{1}{4} - \frac{1}{4}e^{-t\mu} & \frac{1}{4} - \frac{1}{4}e^{-t\mu} & \frac{1}{4} + \frac{3}{4}e^{-t\mu} & \frac{1}{4} - \frac{1}{4}e^{-t\mu} \\ \frac{1}{4} - \frac{1}{4}e^{-t\mu} & \frac{1}{4} - \frac{1}{4}e^{-t\mu} & \frac{1}{4} - \frac{1}{4}e^{-t\mu} & \frac{1}{4} + \frac{3}{4}e^{-t\mu} \end{pmatrix}$$

* If we start at a particular base, there are 3 possible ways to mutate away from it, each with rate μ. Generalizing across all four bases,

$$r'(t) = -\frac{3}{4}\mu + \frac{3}{4}s(t)\mu$$

$$s'(t) = -\frac{1}{4}\mu + \frac{1}{4}r(t)\mu$$

* These equations are satisfied by the following closed form formulas:

$$r(t) = \frac{1}{4}(1 + 3e^{-\mu t})$$

$$s(t) = \frac{1}{4}(1 - e^{-\mu t})$$

* At t=0, r(t) is 1, which makes sense because without any time for a mutation to occur, the base must stay the same. In keeping with this principle, s(0) = 0.
* As t $\to \infty$, r(t) and s(t) both approach 1/4 (remember that s(t) is the probability of changing to a specific other letter). The sum of the s(t)s for the 3 possible base changes approaches 3/4 under these conditions.

– Solving for evolutionary time t

* Let p be the probability that a given base is different between two sequences. We can calculate this by aligning the sequences and counting the number of mismatches relative to matches. Try to avoid parts of the sequence that are under positive or negative selection because it is hard to observe the true mutation rates in such regions.

* Because of degeneracy in the genetic code, some codons result in the same amino acid regardless of which base is in the third position. Such sites are called four-fold degenerate, and are a decent place to observe the neutral rate of evolution because there is little or no selection pressure.
* Once we have p, we can solve for t:

$$\mu t = -ln(1 - 4p/3)$$

- Solving for d, the branch length in terms of substitutions per site
  * Let $d = 3/4\mu t$ denote substitutions per site. Substituting in our expression for μt in terms of p, we get

$$d = -\frac{3}{4}ln(1 - 4p/3)$$

  * This tells us the distance between two sequences according to the Jukes-Cantor model. A similar process can be used to find distances according to other models.
  * Keep in mind that this distance is not chronological or even in terms of generations. Many factors, including population size, affect how mutations are introduced into populations.

3 Building Phylogenetic Trees

Let's say we have aligned pairs of sequences from different species, estimated the value of p across each, and applied our evolutionary model of choice to obtain a distance for each that reflects the substitutions per site. This gives us a matrix of pairwise distances between species. We can now start constructing a tree.

- Method 1: Unweighted Pair Group Method Using Arithmetic Averages (UPGMA)
  - Also known as the average linkage method.
  - Molecular clock: the idea that all species on Earht evolve at the same rate. Definitely not true, as we have discussed before. One example is that rats and mice evolve much faster than humans.
  - Assumption: Substitutions per site reflect chronological time. In other words, we assume that there is a perfect correlation between time passed and number of substitutions, and that this holds across different branches in a tree. This assumption simplifies things, but it is not correct.
  - In addition, the algorithm assumes ultrametric distances. A distance function is ultrametric if for any three distances $d_{ij} \leq d_{ik} \leq d_{jk}$, it is true that $d_{ij} \leq d_{ik} = d_{jk}$. This is known as the three point condition.
    * Let's think about why this holds. As before, let i and j be the two closest species of the three. We know that d_{ij} equals two times the distance between i (or j) and their closest common ancestor. To get the distance d_{ik}, we similarly must find the closest common ancestor of i and k. Since i and j are on the same 'level', they will have the same closest common ancestor with k. Consequently, $d_{ik} = d_{jk}$. These distances are less than or equal to d_{ij} because of our initial condition $d_{ij} \leq d_{ik} \leq d_{jk}$.
    * UPGMA is guaranteed to correctly reconstruct a binary tree if the distances provided are ultrametric. The problem is that we often run into cases where distances are not ultrametric, and the incorrect tree may be generated in these cases.

- First step of UPGMA: Define distances between every pair of species. Then, join the species that are closest first. Given two groups of species, we can define the cluster distance as the sum of all the pairwise distances between species across the clusters normalized by the number of such pairs:

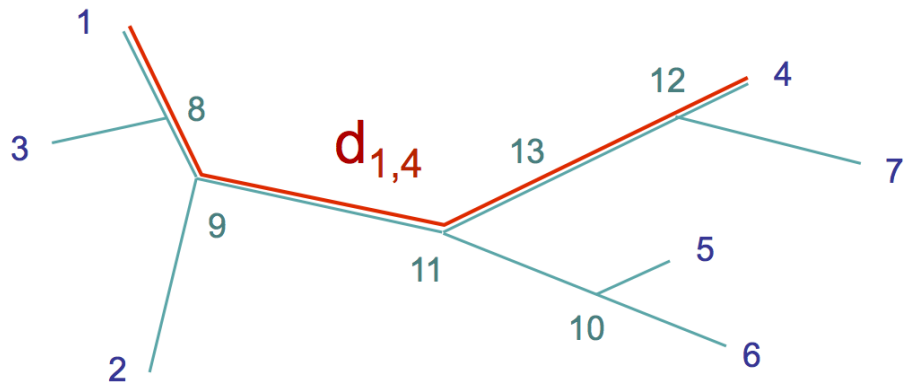
$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} d_{pq}$$

- If we have a third cluster C_l, we can quickly calculate its distance from a joint cluster C_k consisting of C_i and C_j without needing to recalculate all pairwise distances:

$$d_{kl} = \frac{d_{il}|C_i| + d_{jl}|C_j|}{|C_i| + |C_j|}$$

- Average Linkage Algorithm
  - **Initialization:**
    - Each node x_i is in its own cluster
    - Define one leaf per sequence, each with height 0
  - **Iteration:**
    - Select the two clusters C_i and Cj with the smallest pairwise distance.
    - Let $C_k = C_i \cup C_j$.
    - Define a node connecting C_i and Cj, and place it at height $d_{ij}/2$
    - Delete C_i and Cj
  - **Termination:**
    - When only two clusters i and j remain, merge them by placing the root node at height $d_{ij}/2$

- Method 2: Additive Distances



-

- Given a tree, distance measurements are additive given that the distance between any two nodes is equal to the sum of the edges connecting them.
- Example: In the tree above, $d_{14} = d_{1,8} + d_{8,9} + d_{9,11} + d_{11,13} + d_{13,12} + d_{12,4}$.
- Reconstructing edge lengths: Find any two neighbors i and j that have a common parent, k. Place the parent node at a distance $d_{km} = 1/2(d_{im} + d_{jm} - d_{ij})$ from any different node $m \neq i, j$

* The $-d_{ij}$ term is there to prevent us from double-counting the edges connecting i to k and j to k.
* We can repeat this process as needed to get all of the distances in the tree.

– Property: For any four leaves x, y, z, and w in a tree with additive distances, the following holds:

$$d_{xy} + d_{zw} < d_{xz} + d_{yw} = d_{xw} + d_{yz}$$

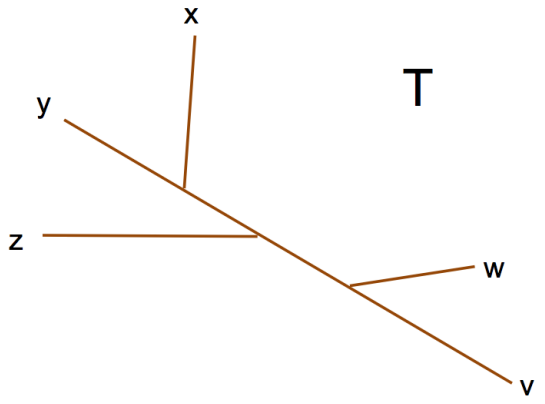
* Basically, one of the possible sums is smaller than the others, and the others are equal. The smaller of the three sums is the one in which the two closest pairs are summed.
* If this property holds in a tree, then the tree is additive.

– **Example: Reconstructing additive distances**

* Given a tree T, we remove the distance labels from the branches. We have all of the pairwise node distances to start.

D

	v	w	x	y	z
v	0	10	17	16	16
w		0	15	14	14
x			0	9	15
y				0	14
z					0



* We find two leaves w and v that have a parent node we will call a. Now, we compute the distance of a from all nodes in the tree (except w and v) using the formula introduced before:

$$d_{ax} = 1/2(d_{vx} + d_{wx} - d_{vw}) = 11$$

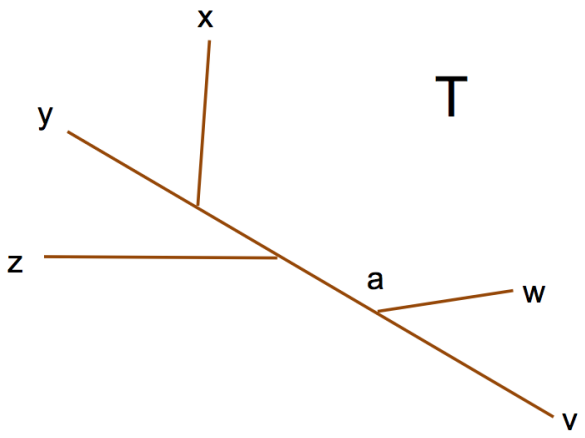
$$d_{ay} = 1/2(d_{vx} + d_{wy} - d_{vw}) = 10$$

$$d_{az} = 1/2(d_{vz} + d_{wz} - d_{vz}) = 10$$

In the distance matrix, replace the entries for v and w with an entry for node a that has the distances we just computed.

D_1

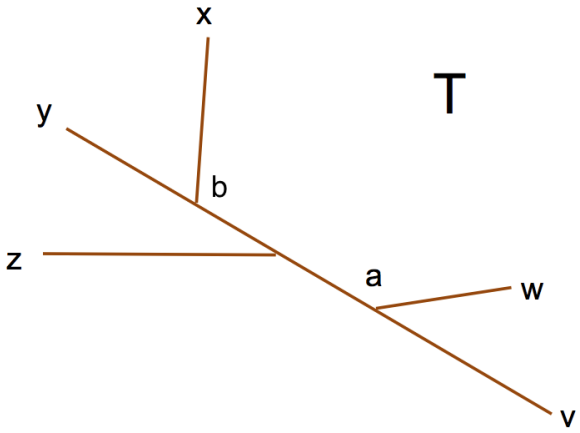
	a	x	y	z
a	0	11	10	10
x		0	9	15
y			0	14
z				0



* Next, we choose leaves x and y with parent b. Following a similar procedure as before, we calculate d_{ba} and d_{bz}.

D_2

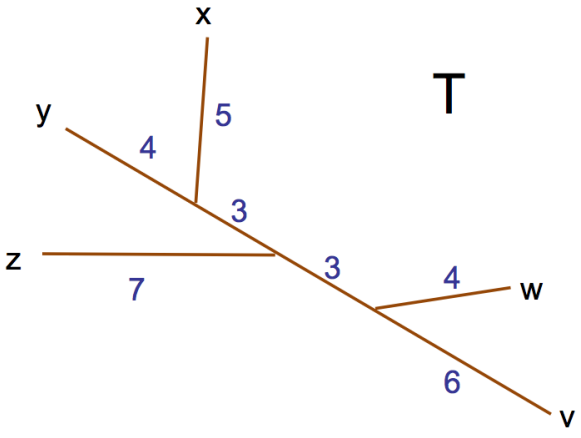
	a	b	z
a	0	6	10
b		0	10
z			0



* Finally, we choose leaves b and z with parent node c, calculating distance d_{ca}. Once we have this, we can fill in the entire tree using the distances in matrices D through $D3$:

D_3

	a	c
a	0	3
c		0



- Method 3: Neighbor Joining
  - We want an algorithm that knows which pair of leaves to join first given a set of additive distances.

    - Trick: This pair is not necessarily the two leaves with the closest distance. Instead, we want to join leaves first that are actually next to each other in the correct tree.
- Guaranteed to produce correct tree if distances are additive, but may produce a good tree without this condition. The algorithm is robust to small errors in the distances between species.
- Neighbor joining transforms the original distance matrix to a different one where the distances between neighboring leaves in the correct tree are the smallest. This is done using the following transformation:

$$D_{ij} = (N-2)d_{ij} - \sum_{k \neq i} d_{ik} - \sum_{k \neq j} d_{jk}$$

- What is going on here: We take the original distances between any pair of leaves i,j and we calculate a new distance D_{ij} for them. N is the number of species (nodes).
- These new distances are minimal for neighboring leaves in the tree.