COS 597c: Topics in Computational Molecular Biology
Lectures 7, 8 and 9: October 11, 13 and 18, 1999
Lecturer: Mona Singh
Scribes: Ching Law and Casim A. Sarkar [1]

Phylogeny

Introduction

Taxonomy is the science of classification of organisms, and phylogeny is the evolution of a genetically related group of organisms (or species).

The purpose of phylogenetic studies are (1) to reconstruct evolutionary ties between organisms and (2) to estimate the time of divergence between organisms since they last shared a common ancestor.

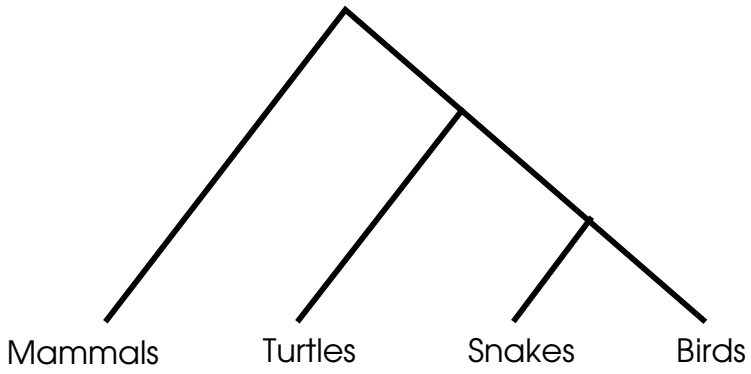


Figure 1: A hypothetical example.

Phylogenetic Trees

We construct phylogenetic trees to illustrate the evolutionary relationships among a group of organisms. The task of phylogenetics is to infer these evolutionary relation-

[1]Notes originally scribed in April 1998, and modified October 1999.

ships based upon existing organisms. The basic idea is to compare specific features of the organisms, under the natural assumption that organisms that share similar features are genetically "close."

There are two ways to build phylogenetic trees: Traditionally, phylogenetic trees were built from morphological features (e.g., beak shapes, presence of feathers, number of legs, etc). Today, we use mostly molecular data like DNA sequences and protein sequences.

Data can be classified into 2 categories:

Discrete characters Each character has a finite number of states. For example, discrete characters include the number of legs of an organism, or a column in an alignment of DNA sequences. In the latter case, the number of states for the column character is 4 (A, C, T, G).

Comparative Numerical Data These data encode the distances between objects and are usually derived from sequence data. For example, we could hypothetically say distance(man,mouse)= 500 and distance(man,chimp)= 100.

Definitions and Terminology

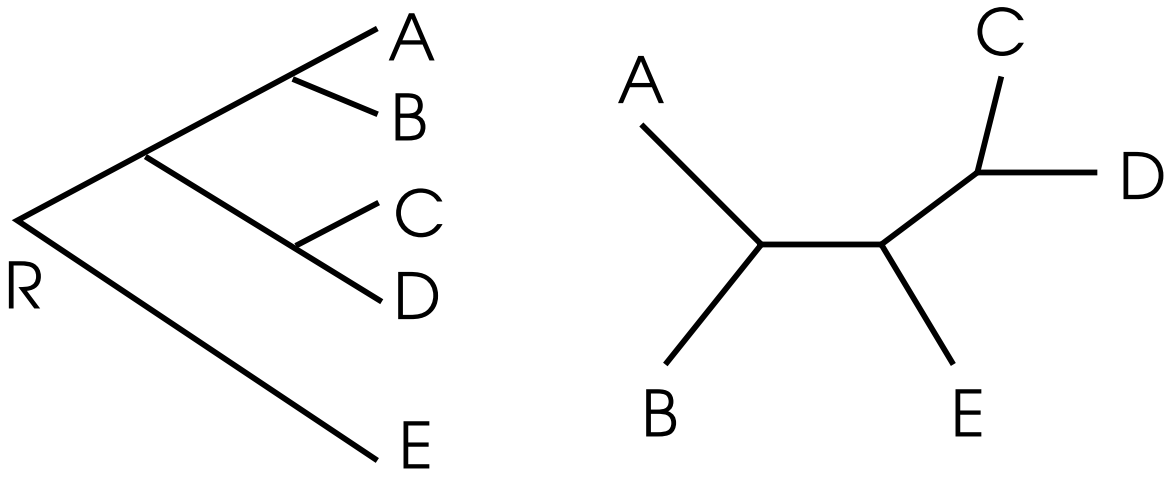


Figure 2: Rooted and Unrooted Trees.

External nodes are things under comparison, also called *operational taxonomic units* (OTUs). *Internal* nodes are *hypothetical* ancestral units. They are used to group

current-day units.

In rooted trees, the root is the common ancestor of all OTUs under study. The path from root to a node defines an evolutionary path. An unrooted tree specifies relationships among OTUs but does not specify evolutionary paths (Figure 2). We can root an unrooted tree by finding an outgroup (i.e., if we have some external reason indicating that a certain OTU branched off first). For example, in Figure 2, the unrooted tree can be transformed to the rooted tree by making E the outgroup.

The *topology* of a tree is the branching pattern of a tree.

All internal nodes of a *bifurcating tree* have 2 descendants if it is rooted, or 3 neighbors if it is unrooted. It is sometimes useful to allow more than 2 descendants (or more than 3 neighbors in the unrooted case), but we will focus on on bifurcating trees.

The branch length can represent the number of changes that have occurred in that branch, or can indicate the genetic distance between nodes connected by that branch, or can indicate the amount of evolutionary time passed along the branch.

Species trees and gene trees. Note that species trees are different from gene trees. For example, a gene's divergence may predate the divergence of species, due to genetic polymorphism in the population. Thus, using this gene to infer the species tree can lead to an overestimate of branch lengths as well as incorrect topology. A solution is to use many genes to infer a species tree.

Gene duplications. Another complication is that during evolution, gene duplications are common, and these duplicated genes continue to evolve separately. Genes which diverged because of speciation (e.g., matching genes in different organisms) are called *orthologues*. Genes which diverged by duplication are called *paralogues*. Our discussion will be limited to constructing species trees, but is valid for studying phylogenies for paralogues as well.

Constructing Phylogenetic Trees

We will cover three major methods for constructing phylogenetic trees:

Distance methods. Evolutionary distances are computed for all OTUs and these are used to construct trees.

Maximum Parsimony. The tree is chosen to minimize the number of changes required to explain the data.

Maximum Likelihood. Under a model of sequence evolution, the tree which gives the highest likelihood of the given data is found.

Pointers to several tree reconstruction packages are available at `http://evolution.genetics.washington.edu/phylip/software.html`. Two popular programs are Paup (Phylogeny Algorithms Using Parsimony) by Swofford, and Phylip (Phylogeny Inference Package) by Felsenstein.

Distance Methods

The problem can be described as follows:

Input: Given an $n \times n$ matrix M where $M_{ij} \geq 0$ and M_{ij} is the distance between objects i and j.

Goal: Build an edge-weighted tree where each leaf corresponds to one object of M, and such that the distances measured on the tree between leaves i and j correspond exactly to the value of M_{ij}. When such a tree can be constructed, we say the distances in M are *additive*.

Example 1 Suppose we are given the distances as in Table 1. Then Figure 3 is a tree which exactly fits this distance data.

	A	B	C	D	E
A	0				
B	12	0			
C	14	12	0		
D	14	12	6	0	
E	15	13	7	3	0

Table 1: Example: A Distance Matrix M.

When is a distance matrix additive? We first recall the definition of a metric space.

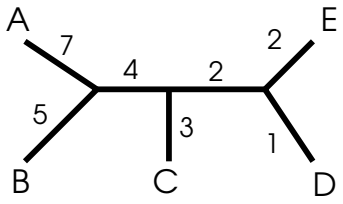


Figure 3: Tree of M.

Definition 1 *A metric space is a set of objects O such that to every pair $i, j \in O$, we associate a non-negative real number d_{ij} (distance) with the following properties:*

$$\begin{array}{lcll} d_{ij} & > & 0 & \text{for } i \neq j \\ d_{ij} & = & 0 & \text{for } i = j \\ d_{ij} & = & d_{ji} \quad \forall i, j & \\ d_{ij} & \leq & d_{ik} + d_{kj} \quad \forall i, j, k & \text{(triangle inequality)} \end{array}$$

It turns out the following exactly characterizes when a metric space is additive.

Claim 1 (4 point condition) *A metric space O is additive iff given any 4 objects of O, we can label them i, j, k, l, such that*

$$d_{ij} + d_{kl} = d_{ik} + d_{jl} \geq d_{il} + d_{jk}$$

We omit the proof of this claim, but provide some intuition. Consider the corresponding tree for an additive matrix M with four species. Here the distances between leaves in the tree are equal to the entries in the matrix M.

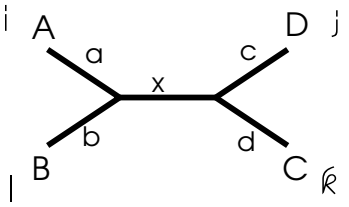


Figure 4: Unrooted tree with 4 leaves.

Then:

$$\begin{aligned} d_{ik} &= a + x + d \\ d_{jl} &= b + x + c \\ d_{ij} &= a + x + c \\ d_{kl} &= b + x + d \\ d_{il} &= a + b \\ d_{jk} &= c + d \end{aligned}$$

And thus we have:

$$\begin{aligned} d_{ij} + d_{kl} &= a + b + c + d + 2x \\ d_{ik} + d_{jl} &= a + b + c + d + 2x \\ &= d_{il} + d_{jk} + 2x \end{aligned}$$

When a matrix is additive, there is an algorithm for finding the phylogenetic tree consistent with the data in $O(n^2)$ time. Unfortunately, distance matrices are rarely additive. In this case, what we can do is to try to find the tree which *best fits* the distance data. Several criteria for "best" are possible; here we give two:

1. Cavalli-Sforza and Edwards criterion. Given observed distances M_{ij}, we choose the weighted tree whose predicted distances d_{ij} minimizes

$$\sum_{i,j} (M_{ij} - d_{ij})^2$$

2. Fitch and Margoliash criterion. We minimize

$$\sum_{i,j} \frac{(M_{ij} - d_{ij})^2}{M_{ij}^2}$$

Note that the predicted distances d_{ij} depend both on topology and branch lengths. Note that an enumeration strategy won't be computationally feasible for a large number of species–just considering all topologies is computationally difficult (see Lemma 1). In fact, both the Cavalli-Sforza and Edwards criterion and the Fitch and Margoliash criterion lead to computationally intractable problems.

Lemma 1 *Given n species, there are $\prod_{i=3}^{n}(2i-5)$ unrooted bifurcating trees (with labeled leaves and unlabeled interior nodes)*

Intuition Consider an unrooted tree with 3 nodes. Only one such tree is possible (Figure 5).

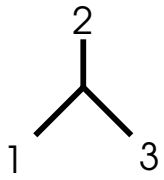


Figure 5: Unrooted tree with 3 nodes.

Now there are three possible branches where we can possibly add a new leaf (Figure 6).

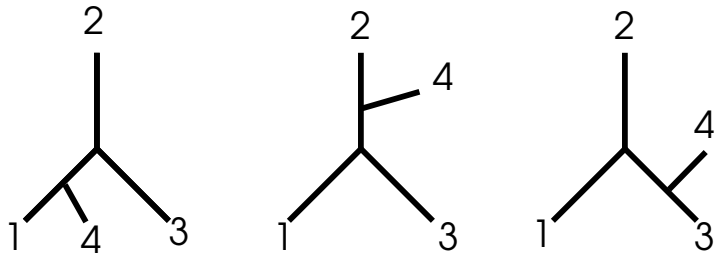


Figure 6: Possible unrooted trees with 4 nodes.

Now for each of these, there are 5 possible branches where we can add the next leaf. Continuing in this manner, we can see how the lemma arises. You can show the lemma more formally by simply using induction. ■

It is also easy to show that the number of rooted trees with n leaves is same as number of unrooted trees with $n+1$ trees: $\prod_{i=3}^{n}(2i-3)$.

Note that by the time we have 10 species or OTUs, there will be $34,459,425$ rooted trees and $2,027,025$ unrooted trees.

Distance Method Heuristics

UPGMA: Unweighted Pair Group Method with Arithmetic Mean

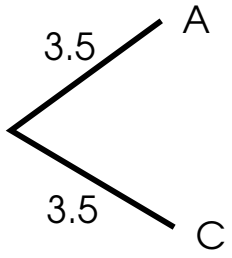
UPGMA is a sequential clustering algorithm. It starts with the pair most similar to build a composite OTU. Now from the new group of OTUs, we pick the pair with highest similarity, and continue in this manner until only 2 OTUs are left. We illustrate this algorithm by example.

Example 2 Say we are given the distance data from Table 2.

	A	B	C	D
A	0			
B	8	0		
C	7	9	0	
D	12	14	11	0

Table 2: Example: A Distance Matrix M.

The closest pair are A and C. Thus we have a new OTU (AC).



Distances to this OTU are computed as arithmetic means:

$$d_{B(AC)} = \frac{d_{BA} + d_{BC}}{2} = 8.5$$
$$d_{D(AC)} = \frac{d_{DA} + d_{DC}}{2} = 11.5$$

Our new matrix is thus:

	(AC)	B	D
(AC)	0		
B	8.5	0	
D	11.5	14	0

Table 3: Distance Matrix M Updated.

Now the closest pair are (AC) and B. The branching node is at $\frac{d_{(AC)B}}{2}$ (Figure 7).

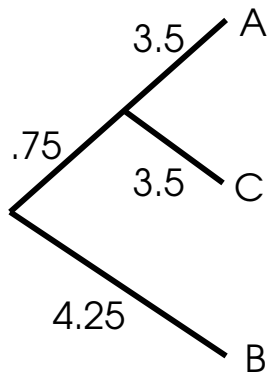


Figure 7: Example: Adding node B.

We recalculate our distance matrix:

$$D_{(ABC)D} = \frac{d_{AD} + d_{BD} + d_{CD}}{3} = \frac{37}{3} = 12\frac{1}{3}$$

	(ABC)	D
(ABC)	0	
D	$12\frac{1}{3}$	0

Table 4: Distance Matrix M Updated twice.

At last, we add D, with branching node at $6\frac{1}{6} \approx 6.17$ (Figure 8).

Here, in each step the distance between composite OTUs is the arithmetic mean of pairwise distances between constituent OTUs of composite OTUs:

$$d_{XY} = \sum_{i \in X, j \in Y} \frac{d_{ij}}{n_x n_y}$$

where n_x and n_y are the number of constituent OTUs in the composite OTUs X and Y, respectively.

Note that in the example just considered, the matrix is actually additive. So using an algorithm for additive matrices, we could have obtained the tree given in Figure 9. Note that the topology of these two trees is different. The UPGMA algorithm works well if the rates of evolution are approximately the same among different lineages (which is not the case for the example we just worked through). If the rates of evolution among different lineages are exactly the same, then we have *ultrametric* data, and this method will work exactly.

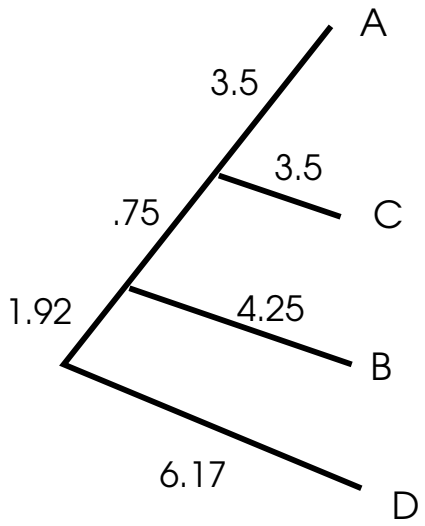


Figure 8: Example: Final tree using UPGMA.

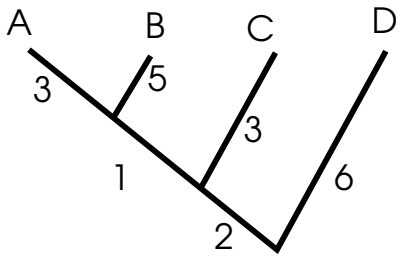


Figure 9: Tree for Table 2.

Neighbor Joining

Neighbor Joining is the most widely used distance based method. The heuristic is to find neighbors sequentially that minimize the total length of the tree.

1. We start with a star tree for the N OTUs(see Figure 10).

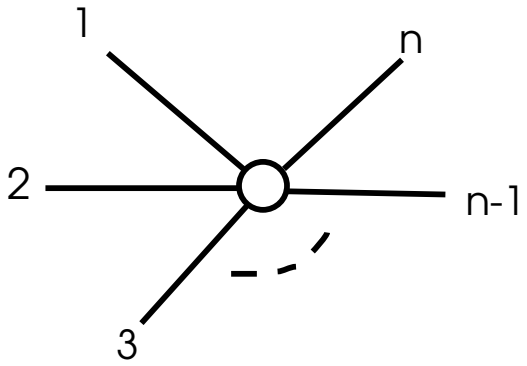


Figure 10: Illustrating the neighbor joining method: Initial tree.

2. The first step is to separate out 2 nodes, say node 1 and node 2, as in Figure 11.

3. For such a tree, the sum of the branch lengths (assuming distances on this tree correspond to our distance matrix), S_{12} is

$$\frac{1}{2(N-2)}\sum_{k=3}^{N}(M_{1k}+M_{2k})+\frac{1}{2}M_{12}+\frac{1}{N-2}\sum_{3\leq i,j\leq N}M_{ij}$$

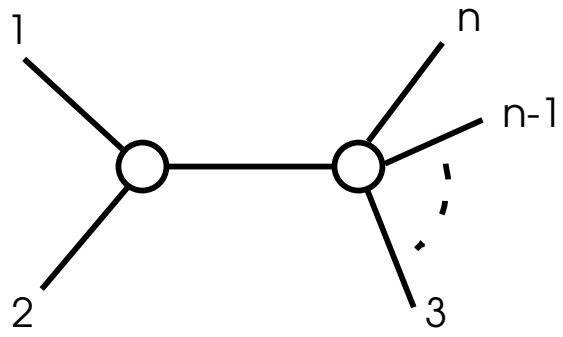


Figure 11: Illustrating the neighbor joining method.

4. Choose the 2 nodes that give the smallest sum of branch lengths. (In Figure 11, it is nodes 1 and 2.) Actually, in practice, what is done is as follows:

   - Let $u_i = \sum_k \frac{M_{ik}}{(n-2)}$
   - Choose i and j for which $Q_{ij} = M_{ij} - u_i - u_j$ is smallest.

   It turns out that $S_{ij} = \frac{Q_{ij}}{2} + \frac{1}{N-2} \sum_{i,j} M_{ij}$, so finding i and j to minimize Q_{ij} is equivalent to minimizing the sum of the branch lengths.

5. Create a merged OTU (i, j), and distances between OTUs are computed to form new distance matrix. The distance from i and j in the tree to node (i, j) is:

$$d_{i,(ij)} = \frac{(M_{ij} + u_i - u_j)}{2}$$

$$d_{j,(ij)} = \frac{(M_{ij} + u_j - u_i)}{2}$$

6. Compute distances between new cluster and all other clusters:

$$M_{(ij)k} = \frac{M_{ik} + M_{jk} - Mij}{2}$$

7. Delete i and j from matrix and replace by (ij)

8. Repeat until all internal branches are found.

For additive matrices, neighbor joining infers the correct topology (although as mentioned earlier, there are other methods in the case of additive distance data, and distance data is rarely additive). On non-additive matrices, neighbor joining often works well in practice, although negative branch lengths are possible!

Computing Distances

We have looked at a couple of distance method heuristics for reconstructing trees, given distance data. One question we could ask at this point is: how do we *obtain* the distance data? One answer is that distance data can be obtained from sequence data. Let us compare the following two sequences:

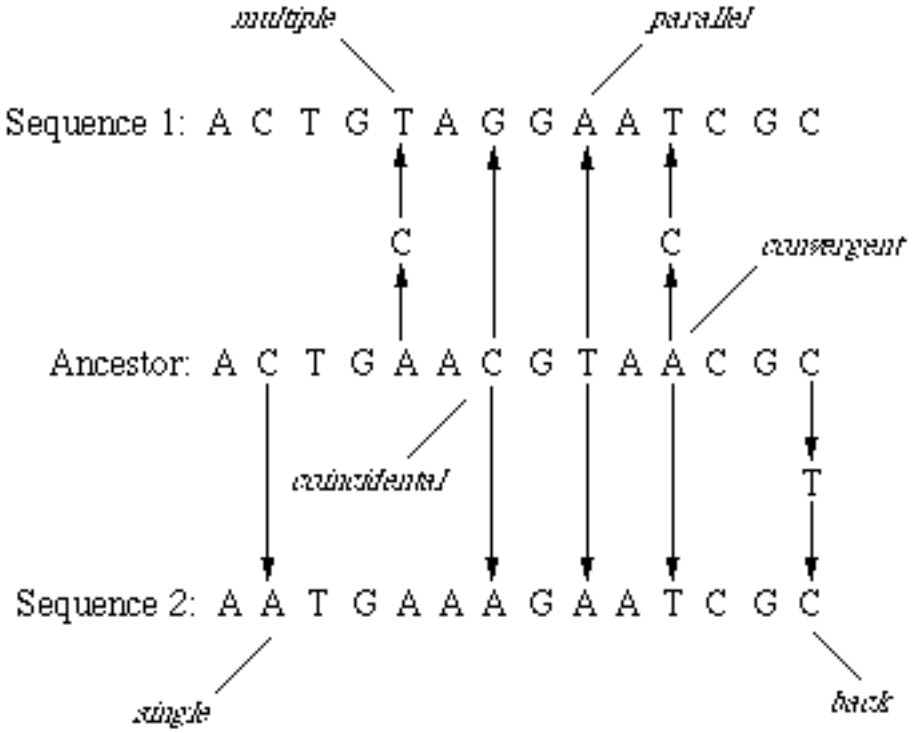


Figure 12: Comparison of Two Sequences with Their Ancestor Shows Several Types of Substitutions

There are only 3 observed difference between the 2 sequences; however, considering the

ancestral sequence, we see that are actually 12 total substitutions. Thus, if multiple substitutions have occurred at any site (*e.g.*, the convergent substitution at site 11), then the naive way of computing distance is an underestimate. How can we correct for multiple substitutions? For DNA sequences, we can use models for nucleotide substitution. For protein sequences, we have already talked about models for amino acid substitution in our discussion of PAM matrices. (We will also use these models when we talk about maximum likelihood methods for phylogenetic reconstruction.)

Models for Nucleotide Substitution

Here we will primarily discuss the Jukes and Cantor model for nucleotide substitution. The key assumptions of this model are that: 1) each position in the DNA sequence is independent, and 2) mutation of a position to any other base is equally likely. These assumptions lead to the following figure of nucleotide substitution (where α is a normalized rate of substitution).

Thus, the rate of substitution is 3α. Given 2 sequences, each of length N, the expected number of total substitutions is $2(3\alpha t)N$.

Let us assume that the nucleotide at a certain site in the DNA sequence is an A at time $t = 0$ [*i.e.*, $p_A(0) = 1$].

Then:

$$\begin{aligned} p_A(1) &= (1 - 3\alpha) \\ p_A(2) &= (1 - 3\alpha) \cdot p_A(1) + \alpha \cdot [1 - p_A(1)] \\ &\vdots \\ p_A(t+1) &= (1 - 3\alpha) \cdot p_A(t) + \alpha \cdot [1 - p_A(t)] \end{aligned}$$

This further implies:

$$\begin{aligned} p_A(t+1) - p_A(t) &= \alpha \cdot [1 - p_A(t)] - 3\alpha \cdot p_A(t) \\ => \Delta p_A(t) &= \alpha - 4\alpha \cdot p_A(t) \end{aligned}$$

Approximating this result by a continuous time model, we obtain:

$$\begin{aligned} \frac{dp_A(t)}{dt} &= \alpha - 4\alpha \cdot p_A(t) \\ => p_A(t) &= \frac{1}{4} + (p_A(0) - \frac{1}{4}) \cdot e^{-4\alpha t} \end{aligned}$$

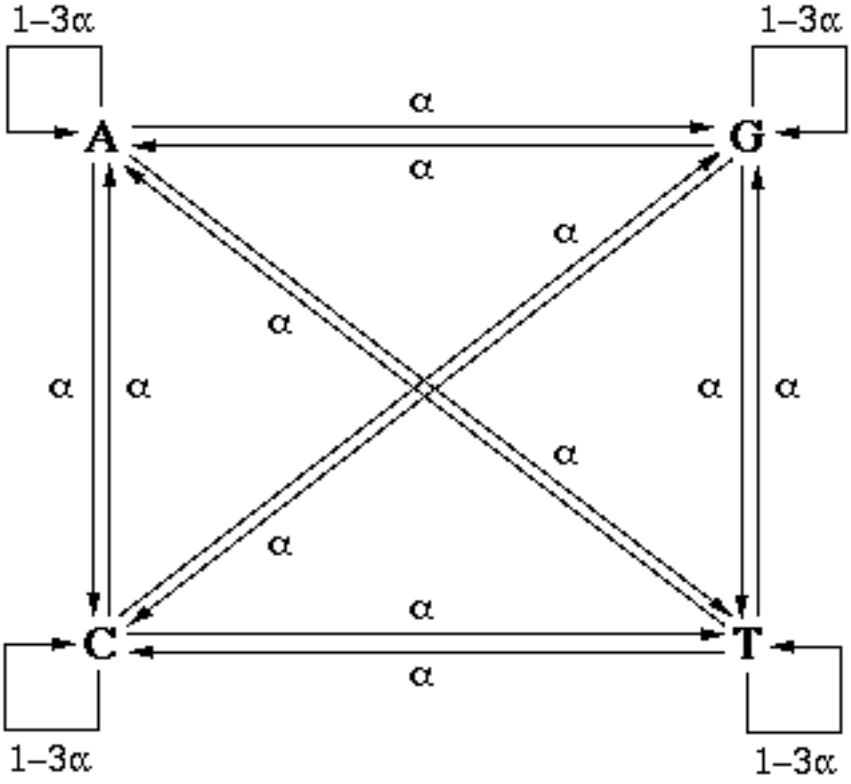


Figure 13: Jukes and Cantor Model for Rates of Nucleotide Substitution

If $p_A(0) = 1$, then $p_A(t) = \frac{1}{4} + \frac{3}{4} \cdot e^{-4\alpha t}$. If $p_A(0) = 0$, then $p_A(t) = \frac{1}{4} - \frac{1}{4} \cdot e^{-4\alpha t}$.

In general,

$p_{ii}(t) = \Pr[i \text{ at time } t | i \text{ at time } 0] = \frac{1}{4} + \frac{3}{4} \cdot e^{-4\alpha t}$
$p_{ij}(t) = \Pr[j \text{ at time } t | i \text{ at time } 0] = \frac{1}{4} - \frac{1}{4} \cdot e^{-4\alpha t}$, where $i \neq j$

Now, consider Figure 14. The probability that the nucleotide A is conserved from the ancestor to both children is $[p_{AA}(t)]^2$. Therefore, the probability that a given site is the same in the two children (not necessarily the ancestor) is:

$$\begin{aligned} I(t) &= [p_{AA}(t)]^2 + [p_{AT}(t)]^2 + [p_{AG}(t)]^2 + [p_{AC}(t)]^2 \\ &= \frac{1}{4} + \frac{3}{4} \cdot e^{-8\alpha t} \end{aligned}$$

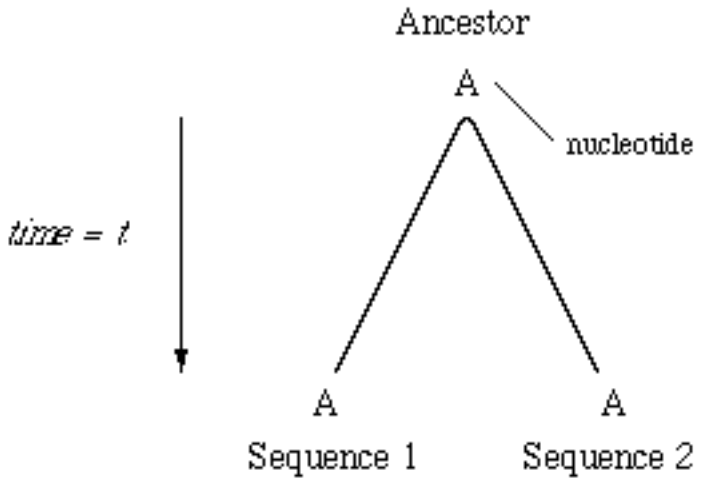


Figure 14: Nucleotide Conservation from Ancestor to Children

The probability that the sites, k_{Seq}, are different is:

$$\begin{aligned} \Pr[k_{Seq1} \neq k_{Seq2}|t] &= 1 - I(t) \\ &= \frac{3}{4} \cdot (1 - e^{-8\alpha t}) \\ => \alpha t &= -\frac{1}{8} \cdot \ln(1 - \frac{4}{3} \cdot \Pr[k_{Seq1} \neq k_{Seq2}|t]) \end{aligned}$$

Earlier, we calculated the expected total number of substitutions for two sequences of length N (*i.e.*, $2(3\alpha t)N$). For a single site comparison, the expected number of substitutions is $6\alpha t$. Substituting for αt using the above equation, the expression for the expected number of substitutions per site becomes:

$$-\tfrac{3}{4} \cdot \ln(1 - \tfrac{4}{3} \cdot \Pr[k_{Seq1} \neq k_{Seq2}|t])$$

We do not know the value of $\Pr[k_{Seq1} \neq k_{Seq2}|t]$, but we know it is equal to the expected number of differences in the two sequences divided by the length of the sequences. We do not know the expected number of differences in the two sequences, but we assume that the observed number of differences is a good approximation. So we estimate $\Pr[k_{Seq1} \neq k_{Seq2}|t]$ using the naive method for distance calculation (*i.e.*, the observed number of differences divided by the sequence length).

For example, given two DNA sequences, each of length 100, with 25 differences, the naive method gives an observed probability, or "distance," of 1/4. Plugging in this value for the probability in the above expression, we estimate the actual number of substitutions per site to be ≈ 0.304.

Another model that is used is Kimura's two-parameter model. This model emphasizes the fact that A-G substitutions and C-T substitutions are the most likely (and are weighted accordingly). All other substitutions are considered less likely (and are also weighted accordingly). Models more general than this also exist.

Character Based Methods

Discrete characters include morphological data (such as the absence or presence of feathers), protein data (20 possible amino acids), and DNA data (four possible nucleotides). All character based methods assume that different characters are independent of each other. Given character data, how does one find a tree to fit the data? What criteria are used to pick the best tree?

Maximum Parsimony

One method is to use maximum parsimony. In this instance, we want to find the tree that minimizes the number of changes needed to explain the data. For example, given the following DNA data, which tree is most parsimonious?

	1	**2**	**3**	**4**	**5**	**6**
A	G	T	C	G	T	A
B	G	T	C	A	C	T
C	G	C	G	G	T	A
D	A	C	G	A	C	A
E	A	C	G	G	A	A

We can create a tree and look at each site individually (see Figure 15).

Sites 1 and 2 each require one change for the given tree. It turns out that the entire data can be explained with a minimum of 9 changes using the tree in Figure 15. However, changing the tree will alter the minimum number of changes required. This example leads us to ask two important questions relating to parsimony:

1. Given a particular tree, how do you find the minimum number of changes needed to explain the data? [Easy]

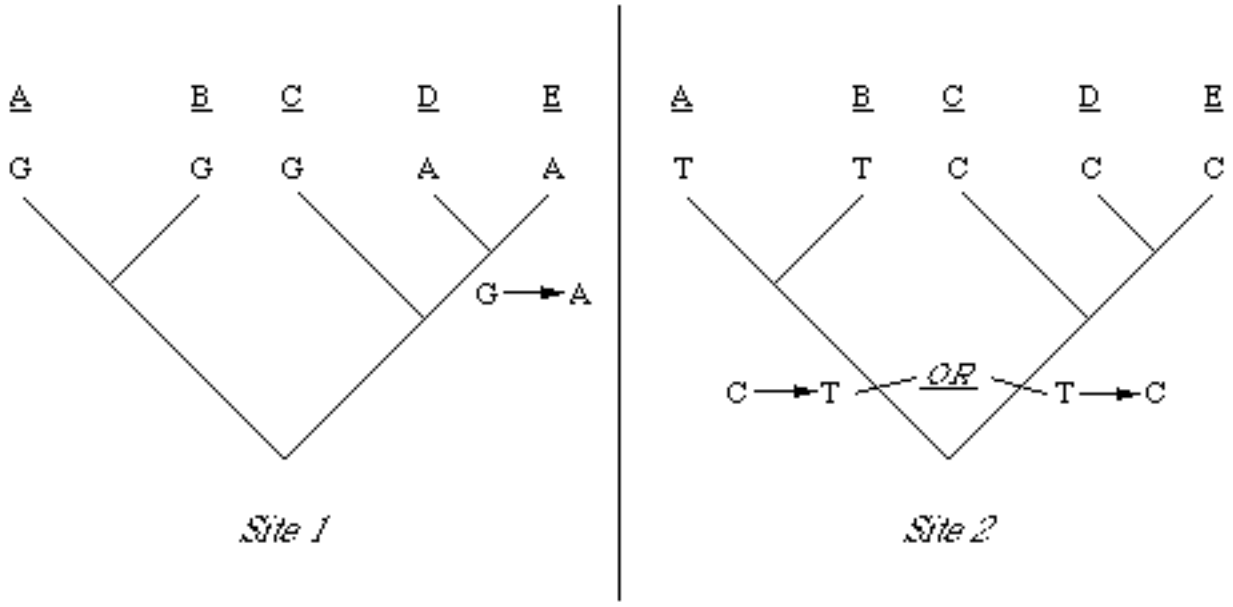


Figure 15: Trees for First Two Sites of Sequences A Through E

2. How do you find the most parsimonious tree? [NP-hard]

To answer the easy first question, we use Fitch's Algorithm. The idea is to construct a set of possible states (*e.g.*, nucleotides) for internal nodes based on the states of the children. For each site, each leaf is labelled by a singleton set containing, for example, the nucleotide at that position. For each internal node i, with children j and k (labels S_j and S_k):

$S_i = S_j \cup S_k$, if $S_j \cap S_k = \emptyset$
$S_i = S_j \cap S_k$ otherwise

The total number of changes equals the total number of union operations. This is illustrated by the Figure 16. We can see from Figure 16 that there are three unions in the tree; this implies that this site requires three changes. It is easy to implement this algorithm by post-order traversal of the tree.

In contrast, the answer to the second question, finding the most parsimonious tree, is not easy. There are many heuristics for doing this. We will quickly talk about two techniques: 1) the branch-and-bound method (prunes search space, and finds

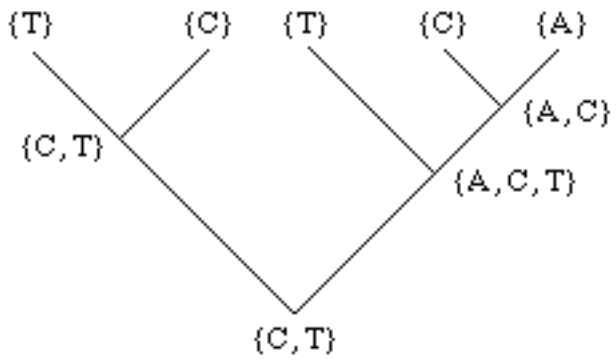


Figure 16: Pictorial Example Employing Fitch's Algorithm for a Given Site

the most parsimonious tree) and 2) the nearest-neighbor interchange method (fast heuristic, which may not find most parsimonious tree).

Branch and bound. The branch-and-bound method (as applied here) counts the number of changes for an initial tree (e.g., an initial tree may be obtained using the neighbor-joining method). Then, starting from scratch, we will search our space by building partial trees (*i.e.*, one branch is added at a time). That is, in the kth level of the search, we will have nodes representing all possible phylogenetic trees with k leaves for the first k species (the order is fixed beforehand arbitrarily). If the cost of any partial tree we are building is greater than that of the initial tree, then search along this line is abandoned. We can improve our search (potentially getting rid of more things) by computing an estimate of the minimum number of changes required to add the additional species.

There is no guarantee with branch and bound on how much of the search space is eliminated.

Nearest-neighbor interchange. The nearest-neighbor interchange method involves rearranging trees at the "neighbor" level and choosing the "neighbor" tree with the best score (*i.e.*, the least number of changes). There are many possibilities for how you can define neighbors. Neighbors in this heuristic procedure are defined as follows. Considering any internal edge, we break up our tree into 4 subtrees. For example, in the tree in Figure 4, the subtrees would consist of the leaves A, B, C and

D, although in general these subtrees consist of more than 1 leaf. This original tree (which has A and B branching off separately from C and D) has two neighbors: one with the roles of B and D switched (i.e., with A and D branching off separately from B and C) and one with the roles of B and C switched (i.e., with A and C branching off separately from B and D). Starting with one tree, we repeatedly choose the neighboring tree with the best score, until there are no neighboring trees with better scores. This is a hill-climbing method, and there is no guarantee that we will find the most parsimonious tree.

There are also many other heuristics for this problem.

Maximum Likelihood Methods

Another method commonly used for reconstructing trees is that of maximum likelihood. Given a probabilistic model for nucleotide substitution (*e.g.*, the Jukes and Cantor model), pick the tree that has the highest probability of generating the observed data. In other words, given character data D and a model M, we want to find the tree T that maximizes the expression $\Pr[D|T, M]$.

We make two independence assumptions to simplify things. We assume that different characters evolve independently. We also assume that after species have diverged, they evolve independently. Thus, if D_i is the data for the i^{th} character, then:

$$\Pr[D|T, M] = \prod_i \Pr[D_i|T, M]$$

Given the tree in Figure 17, and assuming a constant rate of mutation along each branch, we can express the probability as:

$$\Pr[i, j, k, l|T, M] = \sum_x \sum_y \sum_z \quad p_x(0) \cdot p_{xl}(t_1 + t_2 + t_3) \cdot p_{xy}(t_1) \cdot p_{yk}(t_2 + t_3) \cdot p_{yz}(t_2) \cdot p_{zj}(t_3) \cdot p_{zi}(t_3)$$

If we are given the topology of the tree with branch lengths, then we can compute the likelihood of the data using dynamic programming (covered in lecture, but missing from these notes). However, to use maximum likelihood methods for reconstructing evolutionary trees, we must also search over all tree topologies (a hard problem–see Lemma 1), and for each tree topology, we must find the optimal branch lengths. The optimal branch lengths cannot be found analytically and are often computed using the EM algorithm.

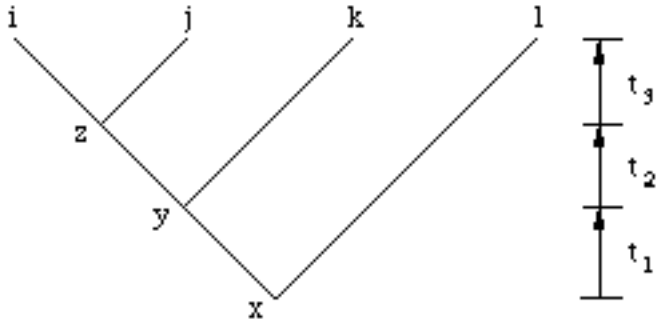


Figure 17: Example Tree for Maximum Likelihood

Bootstrapping: Confidence in Reconstructed Trees

Bootstrapping is a technique commonly used for estimating statistics or parameters when the distribution is difficult to derive analytically. For example, say we were given the following data, and we obtained a tree in which A and B were immediate children of one ancestor and C and D were immediate children of another.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
A	G	C	A	G	T	A	C	T
B	G	T	A	G	T	A	C	T
C	A	C	A	A	T	A	C	C
D	A	C	A	A	C	A	C	T

Now, we would like to get confidence levels that A and B belong together and that C and D belong together. We create pseudo-samples by choosing a site at random and placing it in column 1. We continue to sample with replacement until the number of random columns generated equals the number of sites in the original sequences. Thus, each pseudo-sample we generate omits different columns from our original alignment. Each pseudo-sample created is used to construct a tree. The confidence for each pair being together is the fraction of times they appear together in the trees generated from the many pseudo-samples.

References

[1] Wen-Hsiung Li. *Molecular Evolution.* Sinauer Associates, 1997.

[2] R. Durbin, S. Eddy, A. Krogh and G. Mitchison. *Biological Sequence Analysis.* Cambridge University Press, 1998.