1
Brief introduction to phylogenetic estimation

The construction of evolutionary trees is one of the major steps in many biological research studies. For example, evolutionary trees of different species tell us how the species evolved from a common ancestor, and perhaps also shed insights into the morphology of their common ancestors, the speed at which the different lineages evolved, how they and their common ancestors adapted to different environmental conditions, etc.

Because the true evolutionary histories cannot be known and can only be estimated, evolutionary trees are *hypotheses* about what has happened in the past. The tree in Figure 1.1 presents a hypothesis about the evolutionary history of a group of mammals and one bird. According to this tree, cats and grey seals are more closely related to each other than either is to blue whales, and cows are more closely related to blue whales than they are to rats or opossums (or anything else in the figure). Not surprisingly, chickens (since they are birds) are the outgroup, since the others are all mammals. The tree shown is the result of a statistical analysis of molecular sequences taken from the genomes of these species. Hence, the estimation of evolutionary trees, also known as phylogenetic reconstruction, is

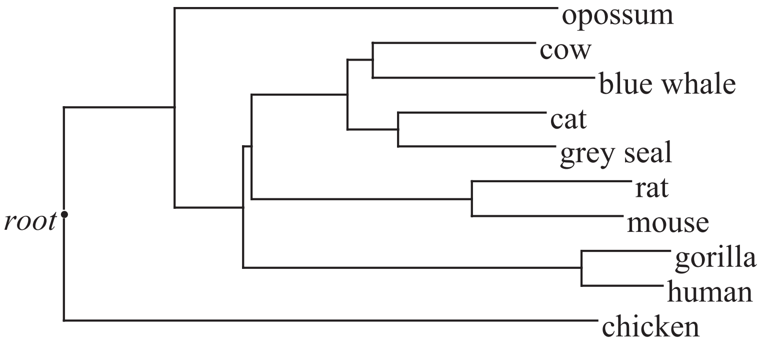


Figure 1.1 (Figure 3.5 from Huson et al. (2010)) **An evolutionary tree of various animals.** The tree is rooted on the left, and has been constructed using molecular sequence data; branch lengths are proportional to evolutionary distances, and not necessarily to time.

a computational problem that involves statistical inference. Furthermore, because it is a statistical inference problem, the accuracy of the tree depends on the model assumptions, the method used to analyze the data, and the data themselves. In other words, the estimation of phylogenies is complicated and difficult. How this phylogeny construction is done, and how to do it *better*, is the point of this text.

Most modern phylogenetic estimation uses molecular sequence data (typically DNA sequences, which can be considered strings over the nucleotide alphabet $\{A, C, T, G\}$), and computes a tree from the set. While there are many ways to compute trees from a set of sequences, understanding when the methods are likely to be accurate requires having some kind of model for how the sequences relate to each other, and more specifically how they *evolved from a common ancestor*.

DNA sequences evolve under fairly complicated processes. At the simplest level, these sequences evolve down trees under processes in which single nucleotides are substituted by other single nucleotides. Many stochastic models have been developed to describe the evolution of sequences down trees under these substitution-only models, and most phylogenetic estimation is based on these models.

However, sequence evolution is more complicated than this. For example, many sequences for the same gene have different lengths, with the changes in length due to processes such as insertions and deletions (jointly called **indels**) of nucleotides, and in some cases duplications or rearrangements of regions within the sequences. Multiple sequence alignments are used to put the sequences into a matrix form so that each column has nucleotides that have a common ancestor, which are then used in phylogeny estimation. Stochastic models to describe sequence evolution have been developed that extend the simpler substitution-only models to include indel events, and are sometimes used to co-estimate alignments and trees.

Genome-scale evolution is even more complicated, since different parts of the genome can evolve under more complicated processes than the models that govern individual portions of the genomes. In particular, due to processes such as incomplete lineage sorting, gene duplication and loss, horizontal gene transfer, hybridization, and recombination, different parts of the genome can evolve down different trees (Maddison, 1997). Again, stochastic models have been developed to describe genome-scale evolution, and are used to estimate genome-scale phylogenies in the presence of one or more of these processes.

Thus, phylogeny estimation is addressed largely through statistical inference under an assumed stochastic model of evolution. While biologically realistic models are fairly complicated, the basic techniques that are used can be described even in the context of very simple models. In this chapter, we introduce the key concepts, issues, and techniques in phylogeny estimation in the context of a very simple binary model of sequence evolution.

1.1 The Cavender-Farris-Neyman model

The **Cavender-Farris-Neyman** (CFN) model describes how a trait (which can either be present or absent) evolves down a tree (Cavender, 1978; Farris, 1973; Neyman, 1971).

Hence, a CFN model has a rooted binary tree T (i.e., a tree in which every node is either a leaf or has two children) with numerical parameters that describe the evolutionary process of a trait. Under the CFN model, the probability of absence (0) or presence (1) is the same at the root, but the state can change on the edges (also called branches) of the tree. Thus, we associate a parameter $p(e)$ to every edge e in the tree, where $p(e)$ denotes the probability that the endpoints of the edge e have different states. In other words, $p(e)$ is the probability of changing state (from 1 to 0, or vice-versa). For reasons that we will explain later, we require that $0 < p(e) < 0.5$.

Under the CFN model, a trait (which is also called a "character") evolves down the tree under this random process, and hence attains a state at every node in the tree, and in particular at the leaves of the tree. You could write a computer program for a CFN model tree that would generate 0s and 1s at the leaves of the tree; thus, CFN is a *generative model*. Each time you ran the program you would get another pattern of 0s and 1s at the leaves of the tree. Thus, if you repeated the process 10 times, each time independently generating a new trait down the tree, you would produce sequences of length 10 at the leaves of the tree.

The task of phylogenetic estimation is generally the inference of the tree from the sequences we see at the leaves of the tree. To do this, we assume that we know something about the sequence evolution model that generated the data. For example, we might assume (whether rightly or wrongly) that the sequences we see were generated by some unknown CFN model tree. Then, we would use what we know about CFN models to estimate the tree. Thus, we can also treat the CFN model as a tool for inference; i.e., CFN can be an *inferential model*. However, suppose we were lucky and the sequences we observe were, in fact, generated by some CFN model tree. Would we be able to reconstruct the tree T from the sequences?

To do this inference, we would assume that each of the sites (i.e., positions) in the sequences we observe had evolved down the same tree, and that each of them had evolved identically and independently (*i.i.d.*). It should be obvious that the ability to infer the tree correctly requires having enough data – i.e., long enough sequences – since otherwise we just can't distinguish between trees. For example, if we have 100 sequences, each of length one, there just isn't enough information to select the true tree with any level of confidence. Therefore, we ask "If sequence length were not an issue, so that we had sequences that were extremely long, would we have enough information in the input sequences to construct the tree exactly with high probability?" We can also formulate this more precisely, as "Suppose M is a method for constructing trees from binary sequences, (T,Θ) is a CFN model tree, and $\varepsilon > 0$. Is there a constant $k > 0$ such that the probability that M returns T given sequences of length at least k is at least $1-\varepsilon$?"

A positive answer to this question would imply that for *any* level of confidence that is desired, there is some sequence length so that the method M would be accurate with that desired probability given sequences of at least that length. A positive answer also indicates that M has this property for all CFN model trees, and not just for some. A method M for which the answer is *Yes* is said to be **statistically consistent** under the CFN model. Thus, what we are actually asking is *Are there any statistically consistent methods for the CFN model?*

1.2 An analogy: determining whether a coin is biased towards heads or tails

Let's consider a related but obviously simpler question. Suppose you have a coin that is biased either towards heads or towards tails, but you don't know which. Can you run an experiment to figure out which type of coin you have?

After a little thought, the answer may seem obvious – toss the coin many times, and see whether heads comes up more often than tails. If it does, say the coin is biased towards heads; otherwise, say it is biased towards tails. The probability that you guess correctly will approach 1 as you increase the number of coin tosses. We express this statement by saying that this method is a *statistically consistent* technique for determining which way the coin is biased (towards heads or towards tails). However, the probability of being correct will clearly depend on the number of coin tosses, so you may need to toss it many times before the probability that you answer correctly will be high.

Now suppose you don't get to toss the coin yourself, but are instead shown a sequence of coin tosses of some length that is chosen by someone else. Now, you can still guess whether the coin is biased towards heads or tails, but the probability of being correct may be small if the coin is not tossed enough times. Note that for this problem - deciding whether the coin is biased towards heads or tails - you will either be 100% correct or 100% wrong. The reason you can be 100% correct is that there are only a finite number of choices. Note also that the probability of being 100% correct can be high, but will never actually be equal to 1; in other words, for any finite number of times you can toss the coin, you will always have some probability of error. Also, the probability of error will depend on how many coin tosses you have and the probability of heads for that coin!

The problem changes in interesting ways if you want to estimate the *actual probability* of a head for that coin, instead of just whether it is biased towards heads or towards tails. However, it's pretty straightforward what you should do - toss the coin as many times as you can, and report the fraction of the coin tosses that come up heads. Note that in this problem your estimations of the probability of a head will generally have error. (For example, if the probability of a head is irrational, then this technique can *never* be completely correct.) Despite this, your estimate *will converge* to the true probability of a head as the number of coin tosses increases. This is expressed by saying that the method is statistically consistent for estimating the probability of a head.

The problem of constructing a CFN tree is very similar to the problem of determining whether a coin is biased towards heads or tails. There are only a finite number of different trees on *n* distinctly labelled leaves, and you are asked to select from among these. Then, if you have a sequence of samples of a random process, you are trying to use the samples to select the tree from that finite set; this is very much like deciding between the two types of biased coins. As we will show, it is possible to *correctly* construct the CFN tree with arbitrarily high probability, given sufficiently long sequences generated on the tree. The problem of constructing the substitution probabilities on the edges of the Cavender-Farris-Neyman tree is similar to the problem of determining the actual probability of a head, in that these are real-valued parameters, and so some error will always be expected. However, if good methods are used, then as the sequence lengths increase the error in the estimated substitution probabilities will decrease, and the estimates will converge to the true values.

While estimating the numeric parameters of a CFN model tree is important for many tasks, we'll focus here on the challenge of estimating the tree T, rather than the numeric parameters. We describe some techniques for estimating this tree from binary sequences, and discuss whether they can estimate the tree correctly with high probability given sufficiently long sequences.

1.3 Estimating the Cavender-Farris-Neyman tree

Recall that the CFN model tree is a pair (T,θ) where T is the rooted binary tree with leaves labelled $s_1, s_2, \ldots, s_n$ and θ provides the values of $p(e)$ for every edge $e \in E(T)$. The CFN model tree describes the evolution of a sequence down the tree, where every site (i.e., position) within the sequence evolves down the model tree identically and independently. Thus the substitution probabilities $p(e)$ on each edge describe the evolutionary process operating on each site in the sequence.

However, this stochastic process can also be described differently, and in a way that is helpful for understanding why some methods can have good statistical properties for estimating CFN model trees. Under the CFN model, the number of substitutions on an edge is modelled by a Poisson random variable $N(e)$ with expected value $\lambda(e)$. Thus, if $N(e) = 0$, then there is no substitution on the edge, while if $N(e) = 1$, then there is a substitution on the edge. Furthermore, if $N(e)$ is even then the endpoints of the edge have the same state, while if $N(e)$ is odd then the endpoints of the edge have different states; hence, $p(e)$ is the probability that $N(e)$ is odd, since we only observe a change on the edge if there is an odd number of substitutions.

Using $\lambda(e)$ (the expected value of $N(e)$) instead of $p(e)$ turns out to be very useful in developing methods for phylogeny estimation. Note that $0 < \lambda(e)$ for all e since $p(e) > 0$. Using the properties of Poisson random variables, it can be shown that

$$\lambda(e) = -\frac{1}{2} ln(1 - 2p(e)).$$

Note that as $p(e) \to 0.5$, $\lambda(e) \to \infty$.

1.3.1 Estimating the CFN tree when evolution is clocklike

An assumption that is sometimes made is that sequence evolution is *clocklike* (also referred to as obeying the **strict molecular clock**), which means that the expected number of changes is proportional to time. If we assume that the leaves represent extant (i.e., living) species, then under the assumption of a strict molecular clock, the total expected number of changes from the root to any leaf is the same. Under the assumption of a strict molecular clock, the matrix of expected distances between the leaves in the tree has properties that make it "ultrametric":

Definition 1.1 An **ultrametric matrix** is an $n \times n$ matrix M corresponding to distances between the leaves in a rooted edge-weighted tree T (with non-negative edge weights) where the sum of the edge weights in the path from the root to any leaf of T does not depend on the selected leaf.

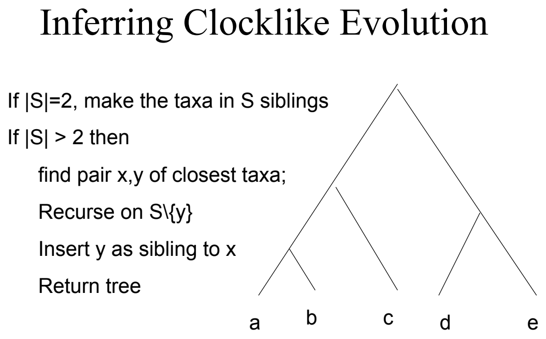
Constructing trees from ultrametric matrices is much easier than the general problem of constructing trees from distance matrices that are not ultrametric. However, the assumption of clocklike evolution may not hold on a given dataset, and is generally not considered realistic (Li and Tanimura, 1987). Furthermore, the ability to reconstruct the tree using a particular technique may depend on whether the evolutionary process is in fact clocklike.

Even though clocklike evolution is generally unlikely, there are some conditions where evolution is close to clocklike. So, let's assume we have a clocklike evolutionary process operating on a CFN tree (T,θ), and so the total number of expected changes from the root to any leaf is the same. We consider a very simple case where the tree T has three leaves, a,b and c. To reconstruct the tree T we need to be able to infer which pair of leaves are siblings, from the sequences we observe at a,b and c. How should we do this?

One very natural approach to estimating the tree would be to select as siblings the two sequences that are the most similar to each other from the three possible pairs. Because the sequence evolution model is clocklike, this technique will correctly construct rooted three-leaf trees with high probability. Furthermore, the method can even be extended to work on trees with more than three leaves, using recursion. For example, consider the model tree given in Figure 1.2, where the branch lengths indicate the expected number of substitutions on the branch. Note that this model tree is *ultrametric*. Thus, under this model, the sequences at leaves a and b will be the most similar to each other of all the possible pairs of sequences at the leaves of the tree. Hence, to estimate this tree, we would first compare all pairs of sequences to find which pair is the most similar, and we'd select a and b as this pair. We'd then correctly infer that species a and b are siblings. We could then remove one of these two sequences (say a), and reconstruct the tree on what remains. Finally, we would add a into the tree we construct on b,c,d,e, by making it a sibling to b.

It is easy to see that a tree computed using this approach, which is a variant of the UPGMA (Sokal and Michener, 1958) method (Unweighted Pair Group Method with Arithmetic Mean), will converge to the true tree as the sequence length increases. That is, it is possible to make mistakes in the construction of the tree – but the probability of making a mistake decreases as the sequence length increases.

However, what if the evolutionary process isn't clocklike? Suppose, for example, that we have a three-leaf CFN model tree with leaves a,b, and c, in which a and b are siblings. Suppose however that the substitution probabilities on the edges leading from the root to b and c are extremely small, while the substitution probability on the single edge incident with a is very large. Then, applying the technique described above would return the tree



Inferring Clocklike Evolution

```
If |S|=2, make the taxa in S siblings
If |S| > 2 then
    find pair x,y of closest taxa;
    Recurse on S\{y}
    Insert y as sibling to x
    Return tree
```

Figure 1.2 **Constructing trees when evolution is clocklike.** We show a cartoon of a model tree, with branch lengths drawn proportional to the expected number of changes. When evolution is clocklike, as it is for this cartoon model, simple techniques such as the one described in the figure will reconstruct the model tree with probability that converges to 1 as the sequence length increases.

with *b* and *c* siblings – i.e., the wrong tree. In other words, this simplified version of UPGMA would converge to a tree other than the true tree as the sequence length increases. This is clearly an undesirable property of a phylogeny estimation method, and is referred to by saying the method is **positively misleading**. An example of a model tree where UPGMA and its variants would not construct the correct tree – even as the sequence length increases - is given in Figure 1.3; the probability of selecting *b* and *c* as the first sibling pair would increase to 1 as the sequence length increases, and so UPGMA and its variants would return the wrong tree.

Clearly, when there is no clock, then sequence evolution can result in datasets for which the inference problem seems to be much harder. Furthermore, for the CFN and other sequence evolution models, if we drop the assumption of the molecular clock, the correct inference of rooted three-leaf trees with high probability is not possible. In fact, the best that can be hoped for is the correct estimation of the *unrooted* version of the model tree with high probability.

1.3.2 Estimating the unrooted CFN tree when evolution is not clocklike

We now discuss how to estimate the underlying unrooted CFN tree from sequences, without assuming clocklike evolution. We will begin with an idealized situation, in which we

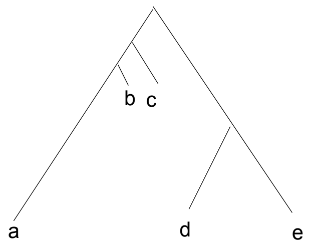


Figure 1.3 **Constructing evolutionary trees when evolution is not clocklike.** We show a cartoon of a model tree on five leaves, with branch lengths drawn proportionally to the expected number of changes of a random site (i.e., position in the sequence alignment). Note that leaves b and c are not siblings, but have the smallest evolutionary distance (measured in terms of expected number of changes). Hence, methods that make the most similar sequences siblings will likely fail to reconstruct the model tree, and the probability of failure will increase to 1 as the number of sites (i.e., sequence length) increases.

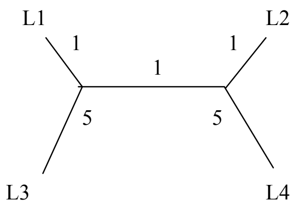
have something we will call "CFN model distances", and show how we can construct the tree from these distances. Afterwards, we will show how to construct the tree from estimated distances rather than from model distances.

CFN model distances: Let (T, θ) be a CFN model tree on leaves $s_1, s_2, \ldots, s_n$, so that T is the rooted binary tree and θ gives all the edge parameters $\lambda(e)$. Let $\lambda_{i,j}$ denote the expected number of changes for a random site on the path $P_{i,j}$ between leaves s_i and s_j in the CFN model tree T; it follows that

$$\lambda_{i,j} = \sum_{e \in P_{i,j}} \lambda(e).$$

The matrix λ is referred to as the **CFN model distance** matrix.

Note that by definition, λ is the matrix of path distances in a tree, where the path distance between two leaves is the sum of the branch lengths and all branch lengths are positive. Matrices that have this property have special mathematical properties, and in particular are examples of **additive** matrices.



		L2	L3	L4
L1		3	6	7
L2			7	6
L3				11
L4				

Figure 1.4 Additive matrix and its edge-weighted tree

Definition 1.2 An $n \times n$ matrix M is **additive** if there is a tree T with leaves labelled $1, 2, ...n$ and non-negative lengths (or weights) on the edges, so that the path distance between i and j in T is equal to $M[i, j]$. An example of an additive matrix is given in Figure 1.4.

In other words, additive matrices correspond to edge-weighted trees in which all edge weights are non-negative; therefore, distance matrices arising from CFN model trees are necessarily additive. Furthermore, CFN model trees have strictly positive branch lengths, and this property additionally constrains the additive matrices corresponding to CFN model trees and makes the inference of CFN model trees particularly easy to do. Techniques to compute trees from additive distance matrices (and even from noisy versions of additive distance matrices) are presented in Chapter 5, and briefly summarized here in the context of CFN distance matrices.

Let's consider the case where the CFN tree T has $n \geq 4$ leaves, and that s_1, s_2, s_3, and s_4 are four of its leaves. Without loss of generality, assume the tree T has one or more internal edges that separate s_1 and s_2 from s_3 and s_4. We describe this by saying that T **induces the quartet tree** $s_1 s_2 | s_3 s_4$. We first show how to compute the quartet tree on these leaves induced by T, and then we will show how to use all these quartet trees to construct T.

Suppose we have the values of $\lambda(e)$ for every edge in T and hence also the additive matrix λ of path distances in the tree. Consider the three following pairwise sums:

- $\lambda_{1,2} + \lambda_{3,4}$
- $\lambda_{1,3} + \lambda_{2,4}$
- $\lambda_{1,4} + \lambda_{2,3}$

Since the weights of the edges are all positive, the smallest of these three pairwise sums has to be $\lambda_{1,2} + \lambda_{3,4}$, since it covers all the edges of T connecting these four leaves *except* for the ones on the path P separating s_1, s_2 from s_3, s_4. Furthermore, the two larger of the three pairwise sums have to be identical, since they cover the same set of edges (every edge in T connecting the four leaves is covered either once or twice, with only the edges in P covered twice). Letting $w(P)$ denote the total weight of the edges in the path P, and

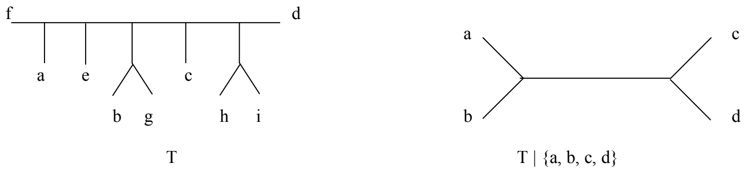


Figure 1.5 Tree T and its homeomorphic subtree on a, b, c, d (i.e., quartet tree)

assuming that T induces the quartet tree $s_1 s_2 | s_3 s_4$,

$$\lambda_{1,2} + \lambda_{3,4} + 2w(P) = \lambda_{1,3} + \lambda_{2,4} = \lambda_{1,4} + \lambda_{2,3}.$$

Since $\lambda(e) > 0$ for every edge e, $w(P) > 0$. Hence, $\lambda_{1,2} + \lambda_{3,4}$ is strictly smaller than the other two pairwise sums.

The **Four Point Condition** is the statement that the two largest values of the three pairwise sums are the same. Hence, the Four Point Condition holds for any additive matrix, which allows branch lengths to be zero (as long as they are never negative). The additional property that the smallest of the three pairwise sums is strictly smaller than the other two is not part of the Four Point Condition, and can fail to hold on additive matrices. It is worth noting that a matrix is additive if and only if it satisfies the Four Point Condition on every four indices.

Now, if we are given a 4×4 additive matrix **D** that corresponds to a tree T with positive branch weights, then we can easily compute T from **D**: we calculate the three pairwise sums, we determine which of the three pairwise sums is the smallest, and use that one to define the split for the four leaves into two sets of two leaves each. We refer to this method as the **Four Point Method**.

Given an $n \times n$ additive matrix **M** with $n \geq 5$ associated to a binary tree T with positive branch lengths, we can construct T using a two-step technique that we now describe. In Step 1, we compute a quartet tree on every four leaves by applying the Four Point Method to each 4×4 submatrix of **M**. In Step 2, we assemble the quartet trees into a tree on the full set of leaves. Step 1 is straightforward, so we only need to describe the technique we use in Step 2, which is called the "All Quartets Method."

The All Quartets Method: Suppose we are given a set Q of fully resolved (i.e., binary) quartet trees, with one tree on every four leaves, and we want to construct a tree T that agrees with all the quartet trees in Q. To see what is meant by "agrees with all the quartets", consider Figure 1.5, which shows an unrooted tree T on leafset $\{a, b, c, d, e, f, g, h, i\}$ and the quartet it defines on $\{a, b, c, d\}$.

The reason this is the quartet tree is that T contains an edge that separates leaves a and b from leaves c and d. We can apply the same reasoning to any unrooted tree and subset of four leaves. In addition, we can even apply this analysis to rooted trees, by first considering them as unrooted trees. For example, Figure 1.3 shows a rooted tree with five leaves, if we ignore the location of the root, we get an unrooted tree with five leaves that we will refer to as T. Now, consider the quartet tree on $\{b,c,d,e\}$ defined by T. With a little care, you should be able to see that T has an edge that separates b,c from d,e; hence, T induces the quartet tree $bc|de$. T also induces the quartet tree $ab|de$, because T has at least one edge (in this case, two) that separates a,b from d,e. We will refer to the set of quartet trees for T constructed in this way by $Q(T)$, and note that

$$Q(T) = \{ab|cd, ab|ce, ab|de, ac|de, bc|de\}$$

Now, suppose we were given $Q(T)$; could we infer T?

The All Quartets Method constructs a binary tree T given its set $Q(T)$ of quartet trees, and has a simple recursive design. We say that a pair of leaves x,y in T that have a common neighbor are **siblings**, thus generalizing the concept of siblinghood to the unrooted tree case. Note that every unrooted binary tree with at least four leaves has at least two disjoint sibling pairs. To construct the unrooted tree T from $Q = Q(T)$, we will use a recursive approach.

First, if $|Q| = 1$, we just return the tree in Q. Otherwise, we search for a pair x,y of leaves that is always together in any quartet tree in Q that contains both x and y. (In other words, for all a,b, the quartet tree in Q on $\{x,y,a,b\}$ is $xy|ab$.) Note that any pair of leaves that are siblings in T will satisfy this property. Furthermore, because Q is the set of quartet trees for T, we are guaranteed to find at least two such sibling pairs, and we only need one. Once we find such a sibling pair x,y, we remove one of the leaves in the pair, say x. Removing x corresponds to removing all quartet trees in Q that contain leaf x, and so reduces the size of Q. This reduced set Q' of quartet trees corresponds to all the quartet trees that T defines on $S \setminus \{x\}$. We then recurse on Q', obtaining the unrooted tree T' realizing Q'. To finish the computation, we add x into T' by making it a sibling of y. Hence, we can prove the following:

Theorem 1.3 *Let* **D** *be an additive matrix corresponding to a binary tree T with positive edge weights. Then the two-step algorithm described above (where we construct a set of quartet trees using the Four Point Method and then apply the All Quartets Method to the set of quartet trees) returns T and runs in polynomial time.*

The proof of correctness uses induction (since the algorithm is recursive) and is straightforward. The running time is also easily verified to be polynomial. Now that we know that the two-step process has good theoretical properties, we ask how we can use this approach when we have binary sequences that have evolved down some unknown CFN tree. To use this approach, we first need to estimate the CFN model distances from sequences, and then possibly modify the algorithm so that we can obtain good results on estimated distances rather than on additive distances.

Estimating CFN distances: Let $p(i,j)$ denote the probability that the leaves s_i and s_j have different states for a random site; the expected number of changes on the path between i and j is therefore

$$\lambda_{i,j} = -\frac{1}{2}\ln(1-2p(i,j)).$$

If we knew all the $p(i,j)$ exactly, we could compute all the $\lambda_{i,j}$ exactly, and hence we would have an additive matrix for the tree T; this means we could reconstruct the model tree and its branch lengths perfectly. (The reconstruction of the model tree topology is straightforward, as we just showed; the calculation of the branch lengths involves some more effort, but is not too difficult.)

Unfortunately, unless we are told the model tree, we cannot know any $p(i,j)$ exactly. Nevertheless, we can *estimate* these values from the data we observe, in a natural way. That is, given sequences s_i and s_j of length k that evolve down the tree T, we can estimate $p(i,j)$. For example, a natural technique would be to use the*fraction* of the number of positions in which s_i and s_j have different states. To put this precisely, letting $H(i,j)$ be the **Hamming distance** between s_i and s_j (i.e., the number of positions in which they are different), then since k is the sequence length, $\frac{H(i,j)}{k}$ is the fraction in which the two sequences are different. Furthermore, as $k \to \infty$, then by the law of large numbers $\frac{H(i,j)}{k} \to p(i,j)$. Hence, we can estimate $\lambda_{i,j}$, the CFN model distance (also known as true evolutionary distance) between sequences s_i and s_j, using the following formula:

$$\hat{\lambda}_{i,j} = -\frac{1}{2}ln(1-2\frac{H(i,j)}{k}).$$

Also, as $k \to \infty$, $\hat{\lambda}_{i,j} \to \lambda_{i,j}$. We call this the **Cavender-Farris-Neyman distance correction**, and the distances that we compute using this distance correction are the **Cavender-Farris-Neyman (CFN) distances**. The distance matrix computed using the Cavender-Farris-Neyman distance correction is an estimate of the model Cavender-Farris-Neyman distance matrix, and converges to the model distance matrix as the sequence length increases.

To say that $\hat{\lambda}_{i,j}$ converges to $\lambda_{i,j}$ for all i,j as the sequence length increases means that for any $\varepsilon > 0$ and $\delta > 0$, there is a sequence length K so that the distance matrix $\hat{\lambda}$ will satisfy $|\hat{\lambda}_{ij} - \lambda_{ij}| < \delta$ for all i,j with probability at least $1-\varepsilon$ given sequence length at least K.

CFN distance matrices may *not* satisfy the triangle inequality, which states that $d_{ik} \leq d_{ij} + d_{jk}$ for all i,j,k, and hence are not properly speaking "distance matrices". However, these estimated distance matrices are symmetric (i.e., $d_{ij} = d_{ji}$) and zero on the diagonal, and so are referred to as **dissimilarity matrices**.

The Naive Quartet Method: We now show how to estimate the unrooted topology of a CFN model tree from a matrix of estimated CFN distances. We can *almost* use exactly the same technique as we described before when the input was an additive matrix, but we need to make two changes to allow for the chance of failure. First, to use the Four Point Method on estimated 4×4 distance matrices, we compute the three pairwise sums, and if the minimum is unique then we return the quartet tree corresponding to that smallest pairwise sum; otherwise, we return *Fail*. This modification is necessary since the input matrix of estimated distances may not uniquely determine the quartet tree. Second, the input Q to the All Quartets Method may not be equal to $Q(T)$ for any tree T; therefore, we have to modify the All Quartets Method so that it can recognize when this happens. The modification is also straightforward: if we fail to find a sibling pair of leaves during some recursive call, we return *Fail*. Also, even if we do construct a tree T, we verify that $Q = Q(T)$, and return *Fail* if $Q \neq Q(T)$.

We call this method the **Naive Quartet Method** because of its simplistic approach to tree estimation. It was originally proposed in Erdös et al. (1999a), where it was called the "Naive Method". We summarize this as follows:

Step 1: Apply the Four Point Method to every four leaves; if any four-leaf subset fails to return a tree, return *Fail*, and exit.

Step 2: Use the All Quartets Method to construct a tree that agrees with all the quartet trees computed in Step 1, if it exists, and otherwise return *Fail*.

This Naive Quartet Method has desirable properties, as we now show:

Theorem 1.4 *Let* **d** *be a* $n \times n$ *dissimilarity matrix and* **D** *be a* $n \times n$ *additive matrix defined by a binary tree* T *with positive edge weights. Suppose that*

$$\max_{ij} |d_{ij} - D_{ij}| < f/2,$$

where f *is the smallest weight of any internal edge in* T*. Then, the Naive Quartet Method applied to* **d** *will return* T*, and runs in polynomial time.*

This theorem and its proof appears in Chapter 5, but the essence of the proof is as follows. When applying the Four Point Method to any 4×4 submatrix of the additive matrix **D**, the gap between the smallest of the three pairwise sums and the other two pairwise sums is at least $2f$. If the entries in the matrix **d** are less than $f/2$ from the entries in matrix **D**, then the smallest of the three pairwise sums given by **d** will have the same 2:2 split as the smallest of the three pairwise sums given by **D**. Hence, the application of the Four Point Method to any 4×4 submatrix of **d** will return the same quartet tree as when it is applied to **D**. Hence, Step 1 will return $Q(T)$. Then, the All Quartets Method will construct T given $Q(T)$.

We summarize this argument, which shows that the Naive Quartet Method is **statistically consistent** under the CFN model (i.e., that it will reconstruct the true tree with probability increasing to 1 as the sequence length k increases):

- We showed that the Naive Quartet Method will reconstruct the unrooted tree topology T of the model CFN tree given an additive matrix defining the model CFN tree.

- We showed that the matrix of estimated CFN distances converges to an additive matrix for the CFN model tree topology as the sequence length goes to infinity.
- We stated (although we did not provide the proof) that whenever the estimated distances are all within $f/2$ of an additive matrix for the model tree T (where f is the length of the shortest internal edge in the model tree), then the Naive Quartet Method will return the unrooted tree topology T.
- Hence, as the sequence length increases, the tree returned by the Naive Quartet Method will be the unrooted topology of the CFN model tree with probability increasing to 1.

The Naive Quartet Method is just one such statistically consistent method, and many other methods have been developed to construct CFN trees from sequence data that have the same basic theoretical guarantee of converging to the true tree as the sequence length increases.

Although the Naive Quartet Method is statistically consistent under the CFN model, when the input dissimilarity matrix **d** is not sufficiently close to additive, this two-step process can fail to return anything! For example, the Four Point Method can fail to determine a unique quartet tree (if the smallest of the three pairwise sums is not unique), and the whole process can fail in that case. Or, even if the Four Point Method returns a unique tree for every set of four leaves, the set of quartet trees may not be compatible, and so the second step can fail to construct a tree on the full dataset. Thus, the two-step process will only succeed in returning a tree under fairly restricted conditions. For this reason, even though this two-step process for constructing trees has nice theoretical guarantees, it is not used in practice. This is why Erdös et al. (1999a) used the word *Naive* in the name of this method – to suggest that the method is really a mathematical construct rather than a practical tool.

1.4 Some comments about the CFN model

In the CFN model, we constrain the substitution probabilities $p(e)$ to be strictly between 0 and 0.5. If we were to allow $p(e) = 0$ on some edge e, then there will be no change on e, and hence it would be impossible to reconstruct the edge with probability converging to 1. At the other extreme, if we were to allow $p(e) = 0.5$ on some edge e, then the two sequences at the endpoints of e will look random with respect to each other, and correctly connecting the two halves of the tree with high probability would be impossible. This is why $p(e)$ is constrained to be strictly between 0 and 0.5.

In the CFN model, the sites evolve down the same model tree. In other words, given any two sites and any edge e in the tree, the expected numbers of changes of the two sites on that edge e are the same, so that all the sites have the *same rate of change*. This assumption is typically relaxed so that each site draws its rate independently from a distribution of rates-across-sites. The meaning of "rates-across-sites" is that each rate gives a multiple for the expected number of changes. Thus, if site i draws rate 2 and site j draws rate 1, then site i has twice as many expected changes as site j on every edge of the tree. Typically, the distribution of rates across sites is modelled using the gamma distribution, but some other distributions (such as gamma plus invariable) are also sometimes used. Note that although the sites can have different rates, they draw their rates independently, and hence all sites evolve under the "same process". This is called the *i.i.d.* (identical and independently distributed) assumption. Finally, given a particular gamma distribution, the entire stochastic model of evolution is fully described by the model tree topology T, the branch lengths, and the gamma distribution.

Biological data typically are not binary sequences, and instead are typically molecular sequences, either of nucleotides (which are over a four-letter alphabet) or amino acids (which are over a 20-letter alphabet). Statistical models of nucleotide and amino acid sequence evolution (discussed in Chapter 8) have also been developed, and methods for estimating trees under these more complex multi-state models have been developed to estimate trees under these models. Despite the increased complexity of the models and methods, for most of these models the theoretical framework and analysis for these more sophisticated methods are basically the same as that which we've described under the CFN model. Thus, even under more biologically realistic models it is possible to reconstruct the unrooted topology of the true tree with high probability, given sufficiently long sequences generated on the tree.

1.5 Phylogeny estimation methods used in practice

There are many phylogeny estimation methods that have been developed, some of which are statistically consistent under the standard statistical models of sequence evolution. One of the methods that has been used to construct trees is the UPGMA method alluded to earlier; UPGMA is an agglomerative clustering method that computes a distance between every pair of sequences, then selects the closest pair of sequences to be siblings, updates the matrix, and repeats the process until a tree is computed for the full dataset. Yet, as we have noted, UPGMA can fail to be statistically consistent under some model conditions.

Maximum parsimony is another approach that has been used to construct many trees. The objective is a tree T in which the input sequences are placed at the leaves of T and additional sequences are placed at the internal nodes of T so that the total treelength, defined to be the total number of changes over the entire tree, is minimized. Another way of defining maximum parsimony is that it is the Hamming Distance Steiner Tree Problem: the input is a set of sequences, and the output is a tree connecting these sequences (which are at the leaves) and with other sequences (i.e., the Steiner points) at the internal nodes, that minimizes the total of the Hamming distances on the edges of the tree. Since the Hamming distance between two sequences of the same length is the number of positions in which they are different, the total of the Hamming distances on the edges of the tree is the same as its treelength.

Finding the best tree under the maximum parsimony criterion is an NP-hard problem (Foulds and Graham, 1982), and hence heuristics, typically based on a combination of hill-climbing and randomization to get out of local optima, are used to find good, though not provably globally optimal, solutions. Maximum parsimony heuristics have improved over the years, but can still be computationally very intensive on large datasets. However, suppose we could solve maximum parsimony exactly (i.e., find global optima); would maximum parsimony then be statistically consistent under the CFN model, or other models?

Unfortunately, maximum parsimony has been proven to be statistically inconsistent under the CFN model and also under standard DNA sequence evolution models, and may even converge to the wrong tree as the sequence length increases (Felsenstein, 1978) (i.e., it can even be *positively misleading*, just like UPGMA). Although UPGMA and maximum parsimony are both statistically inconsistent under standard DNA sequence evolution models, other methods have been developed that are statistically consistent under these models, and are commonly used in practice. Examples of these methods include polynomial time distance-based methods such as neighbor joining (Saitou and Nei, 1987) and FastME (Lefort et al., 2015). The Naive Quartet Method is statistically consistent under the CFN model, and also under standard nucleotide sequence evolution models, and its statistical consistency is extremely easy to prove. The Naive Quartet Method is also polynomial time, and so is a polynomial time statistically consistent method for estimating trees under standard sequence evolution models.

Maximum likelihood is another method that is statistically consistent under standard sequence evolution models (Neyman, 1971; Felsenstein, 1981). To understand maximum likelihood, we describe its use for estimating CFN trees. First, given a CFN model tree (T,θ) and a set S of binary sequences of the same length, we can compute the probability that S was generated by the model tree (we will prove this in Chapter 8). Thus, given S, we can search for the model tree (i.e., the tree topology and the substitution probabilities on the edges) that has the largest probability of generating S. Finding the maximum likelihood tree is NP-hard (Roch, 2006), but if solved exactly it is a statistically consistent estimator of the tree.

Bayesian estimation of phylogenetic trees using Markov Chain Monte Carlo (MCMC) (described in Chapter 8.7) has many theoretical advantages over maximum likelihood estimation and other approaches (Huelsenbeck et al., 2001) and is also able to produce a statistically consistent estimation of the true tree under standard DNA sequence evolution models (Steel, 2013); however, Bayesian MCMC methods needs to run for a long time to have good accuracy. Thus, maximum likelihood and Bayesian MCMC estimation of phylogenetic trees are generally much slower than distance-based estimation methods.

Based on this, one could presume that methods like neighbor joining and the Naive Quartet Method would dominate in practice, since they are polynomial time and statistically consistent, while other methods are statistically inconsistent (e.g., maximum parsimony and UPGMA) or computationally intensive (e.g., maximum likelihood, Bayesian methods, and maximum parsimony), or sometimes both.

1.6 Measuring error rates on simulated datasets

Phylogeny estimation methods are evaluated for accuracy, primarily with respect to the tree topology (as an unrooted tree), using both simulated and biological datasets. Because the true evolutionary history of a biological dataset can rarely be known with confidence, most performance studies are based on simulated datasets. In a simulation study, a model tree is created, and then sequences are evolved down the tree. These sequences are then used to compute a tree, and the computed tree is compared to the model tree. Under the simplest evolutionary models, the sequences evolve just with substitutions, so that individual letters (i.e., nucleotides or amino acids) within the sequences can change during the evolutionary process, but the length of the sequence does not change. More complex models include other processes, such as insertions and deletions (jointly called "indels"), so that the sequences change in length over time. If the sequence evolution process includes insertions and deletions, then a multiple sequence alignment is typically first computed before the tree is estimated. See Figure 1.6 for a graphical description of how a simulation study is performed.

Because the true tree and true alignment are rarely known on any biological dataset, simulation studies are the norm for evaluating phylogeny estimation methods, and are also frequently used to evaluate multiple sequence alignment methods. In a simulation study, a model tree is created, often using a simulation process where a tree is produced under a mathematical model for speciation (e.g., a birth-death process), and then sequences are evolved down the tree under a model that describes the evolutionary process. Often, these models will assume a substitution-only process (such as the Cavender-Farris-Neyman model for binary sequences that we discussed earlier, but also under models such as the Jukes-Cantor (Jukes and Cantor, 1997) and Generalised Time Reversible (Tavaré, 1986) models, which model DNA sequence evolution). When alignment estimation is also of interest, then other models are used in which sequences evolve with insertions, deletions, and sometimes other events. Thus, in one run of the simulation procedure, a set of sequences is generated for which we know the entire evolutionary history relating the sequences, and hence we know the true alignment. Once the sequences are generated, an alignment can be estimated from the unaligned sequences, and a tree can be estimated on the estimated alignment. The estimated alignment and estimated tree can be compared to the true (model) tree and true alignment, and the error can be quantified. By varying the model parameters, the robustness of the estimation methods to different conditions can be explored, and methods can be compared for accuracy.

There are many ways to quantify error in phylogeny estimation, but the most common one measures the distance between two trees in terms of their bipartitions, which we now define. If you take an edge (but not its endpoints) out of a tree, it separates the tree into two subtrees, and so defines a bipartition on the leaves of the tree. Each edge in a tree thus defines a bipartition on the leafset. The bipartitions that are present in the model tree but not in the estimated tree are called **false negatives** (FN), and the bipartitions that are present in the estimated tree but not in the model tree are referred to as **false positives** (FP). Since the bipartitions are defined by edges, we sometimes refer to these as false positive and false negative edges (or as false negative and false positive branches). The edges that are incident with leaves define the trivial bipartitions, and are present in any tree that has the same leafset; the internal edges (which are the ones that are not incident with leaves) define the non-trivial bipartitions. The **Robinson-Foulds** (RF) distance (also called the bipartition distance) between two trees is the number of non-trivial bipartitions that are present in one or the other tree but not in both trees.

Each of these ways of quantifying error in an estimated tree can be normalized to produce a proportion between 0 and 1 (equivalently, a percentage between 0 and 100). For example, the FN error rate would be the percentage of the non-trivial model tree bipartitions that are not present in the estimated tree, and the FP error rate would be the percentage of the non-trivial bipartitions in the estimated tree that are not present in the model tree. Finally, the **Robinson-Foulds error rate** is the RF distance divided by $2n-6$, where n is the number of leaves in the model tree; note that $2n-6$ is the maximum possible RF distance between two trees on the same set of n leaves.

Figure 1.7 provides an example of this comparison; note that the model tree (called the true tree in the figure) is rooted, but the inferred tree is unrooted. To compute the tree error, we unroot the true tree, and treat it only as an unrooted tree. Since both trees are binary (i.e., each non-leaf node has degree three), there are only two internal edges. Each of the two trees have the non-trivial bipartition separating S_1, S_2 from S_3, S_4, S_5, but each tree also has a bipartition that is not in the other tree. Hence, the RF distance between the two trees is 2, out of a maximum possible of 4, and so the RF error rate is 50%. Note also that there is one true positive edge and one false positive edge in the inferred tree, so that the inferred tree has FN and FP rates of 50%.

1.7 Getting branch support

The methods we have described output trees, and most of them output a single tree (i.e., a "point estimate"). The only real exceptions to this are the Bayesian MCMC methods, which output a distribution on trees, and maximum parsimony analyses, which can return the set of all the best trees found during the heuristic search. However, a single tree (or even a collection of trees) is not generally sufficient; typically the biological analysis also needs to have a sense of the statistical support for each edge in the tree.

The estimation of support values of edges in a phylogenetic tree is often performed using non-parametric bootstrapping, where "bootstrap replicate" datasets are created by sampling sites with replacement from the input sequence alignment, and then trees are computed on these bootstrap replicate datasets. The proportion of these trees that have a particular edge (as defined by its bipartition on the leafset) is used as the statistical support for the edge. Bayesian methods output a distribution on trees, and can use the trees in the distribution to compute the support on each edge.

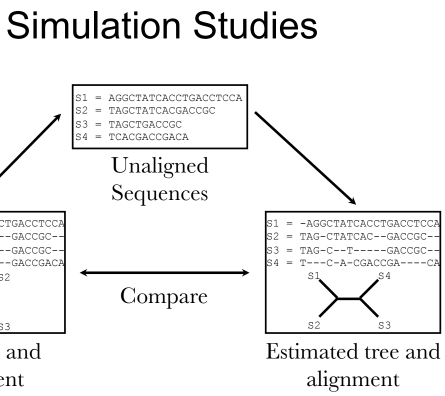

Simulation Studies
S1 = AGGCTATCACCTGACCTCCA
S2 = TAGCTATCACGACCGC
S3 = TAGCTGACCGC
S4 = TCACGACCGACA
Unaligned Sequences
S1 = -AGGCTATCACCTGACCTCCA
S2 = TAG-CTATCAC--GACCGC--
S3 = TAG-CT-------GACCGC--
S4 = -------TCAC--GACCGACA
S1
S2
S4
S3
True tree and alignment
Compare
S1 = -AGGCTATCACCTGACCTCCA
S2 = TAG-CTATCAC--GACCGC--
S3 = TAG-C--T-----GACCGC--
S4 = T---C-A-CGACCGA----CA
S1
S4
S2
S3
Estimated tree and alignment

Figure 1.6 A simulation study protocol. Sequences are evolved down a model tree under a process that includes insertions and deletions; hence, the true alignment and true tree are known. An alignment and tree are estimated on the generated sequences, and then compared to the true alignment and true tree.

1.8 Using simulations to understand methods

Our discussion has introduced the basic theoretical framework for phylogeny estimation, including statistical models of sequence evolution and some simple methods for estimating trees under these models. We have also noted that many methods, including maximum likelihood, Bayesian MCMC, neighbor joining, and the Naive Quartet Method, are statistically consistent methods for estimating the true unrooted tree under standard stochastic models of evolution. In contrast, we showed that UPGMA and maximum parsimony are *not* statistically consistent under the same stochastic models, and are even positively misleading in that they will produce the wrong tree with probability increasing to 1 as the sequence length increases under some conditions. On the face of it, this would seem to suggest that UPGMA and maximum parsimony are both inferior to the Naive Quartet Method and neighbor joining. Thus, perhaps maximum parsimony should never be used instead of the Naive Quartet Method or neighbor joining.

Yet the Naive Quartet Method will only return the true tree if every quartet tree is computed exactly correctly. As many have observed, some quartet trees can be very difficult to compute, even given sequences that have thousands of sites (Huelsenbeck and Hillis, 1993; Hillis et al., 1994). Furthermore, as the number of sequences in the dataset increases, the probability of correctly reconstructing every quartet tree would decrease. Hence, the Naive

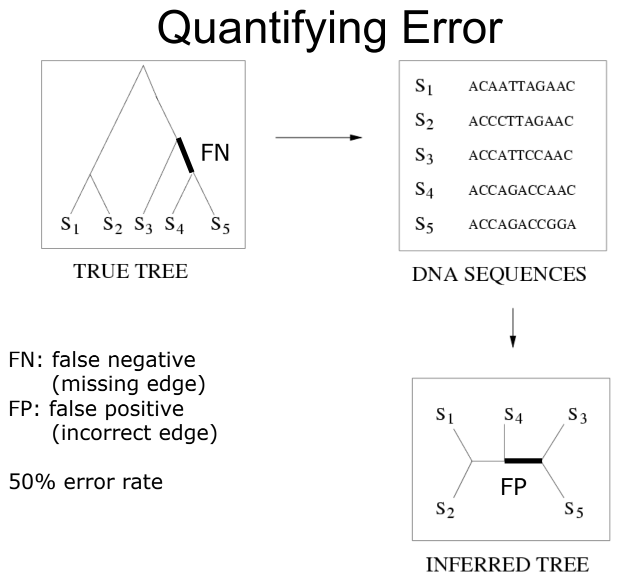


Figure 1.7 **How tree estimation error is calculated in a simulation study.** In a simulation study, the true tree is known, and so can be used to measure error in an estimated tree. Although the true tree is rooted and the inferred tree is unrooted, the error calculation is based on the (non-trivial) bipartitions induced by the internal edges, and so the true tree is interpreted as an unrooted tree. Some of the edges in the two trees are labelled, but others are not. The edges that are not labelled induce bipartitions that are in both trees; all other edges define bipartitions for only one of the two trees. False positive (FP) edges are those that are in the estimated tree but not the model tree, while false negative (FN) edges are those that are in the model tree but not the estimated tree. In this example, one of the two internal edges in the inferred tree is a false positive, and the other is a true positive; hence the false positive rate is 50%. Similarly, although the true tree is rooted, when we treat it as an unrooted tree, one of its internal edges is a true positive and the other is a false negative; hence the false negative rate is 50%. The number of false positive plus false negative edges, divided by $2n-6$ (where n is the number of leaves in each tree) is the Robinson-Foulds (RF) error rate. When both trees are binary, the FN, FP, and RF rates are identical.

Quartet Method would seem to be a rather poor choice of method for phylogeny estimation for any large dataset, even though it is statistically consistent and runs in polynomial time. Indeed, the Naive Quartet Method may not even be useful on most moderate-sized datasets.

In comparison, UPGMA, neighbor joining, and maximum parsimony always return a tree, and so will not have this kind of dramatic failure that the Naive Quartet Method has.

What does the theory suggest about the relative performance between neighbor joining and maximum parsimony? Or, put differently, since neighbor joining is polynomial time and statistically consistent whereas maximum parsimony is neither, does this mean that neighbor joining should be more accurate than maximum parsimony? The answer, perhaps surprisingly, is *no*: there are model conditions and sequence lengths where trees computed using maximum parsimony heuristics are substantially more accurate than trees computed using neighbor joining.

As an example of this phenomenon, see Figure 1.8, which shows some of the results from Nakhleh et al. (2002) comparing a heuristic for maximum parsimony (MP), neighbor joining (NJ), and a variant of neighbor joining called Weighbor (Bruno et al., 2000). The results shown here are for simulated data that evolve down a Kimura 2-parameter (K2P) model tree (see Figure 8.1) with 400 leaves, under varying rates of evolution from low (diameter = 0.2) to high (diameter = 2.0), where the diameter indicates the expected number of changes for a random site on the longest leaf-to-leaf path. The y-axis shows the Robinson-Foulds error rate (i.e., the normalized Robinson-Foulds distance), so the maximum possible is 1.0.

The Kimura 2-parameter model is a DNA sequence evolution model, and neighbor joining and Weighbor are known to be statistically consistent under this model. However, maximum parsimony is not statistically consistent under the Kimura 2-parameter model. Hence, the relative performance of neighbor joining and maximum parsimony on these data is striking, since neighbor joining is less accurate than maximum parsimony under all the tested conditions.

This figure also shows other trends that are very interesting. First, Weighbor is very similar to neighbor joining for low diameters, but for high diameters Weighbor is clearly more accurate. The difference between Weighbor and neighbor joining is most noticeable for the highest diameter condition. In fact, Weighbor is designed explicitly to deal better with high rates of evolution, and it does this by considering the statistical model of evolution in a more nuanced way (in particular, by noting that large distances have high variance).

The third interesting observation is the bell-shaped curve for the three methods: at the lowest diameter, the errors start off somewhat high, then decrease as the diameter increases, and then increase again. Bell-shaped curves are quite common, and the explanation is interesting. At the lowest evolutionary rates, there may not be sufficient amounts of change to reconstruct one or more of the edges in the tree. At the highest evolutionary rates, sequences can seem nearly random, and it can be difficult to distinguish signal from noise. Indeed, there tends to be a “sweet spot” in which the rate of evolution is sufficient to reconstruct the tree with high accuracy, but not so high that the noise overcomes the signal.

This study also shows that not all methods respond the same to increases in evolutionary rates; as seen here, neighbor joining in particular has a stronger negative reaction to high evolutionary rates than Weighbor, which seems to not be negatively impacted at all; this

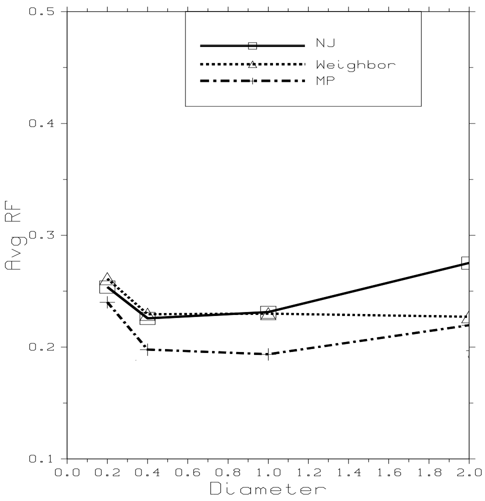


Figure 1.8 (Adapted from Nakhleh et al. (2002)) Tree error of three phylogeny estimation methods on simulated datasets with 400 sequences, as a function of the evolutionary diameter (expected number of changes of a random site across the longest path in the tree). The three methods are neighbor joining (NJ), weighted neighbor joining (Weighbor), and maximum parsimony (MP). The sequence datasets were evolved under Kimura 2-parameter model trees (Kimura, 1980) with gamma distributed rates across sites, and distances between sequences were computed under the K2P model. The study shows that increasing the evolutionary diameter tends to increase the estimation error, that neighbor joining (NJ) is the least accurate method on these data, and that maximum parsimony (MP) is the most accurate. The data also suggests that at higher evolutionary diameters, Weighbor might be more accurate than maximum parsimony.

difference is due to how Weighbor treats large distances in its distance matrix. Maximum parsimony is certainly impacted, but perhaps not as substantially as neighbor joining.

1.9 Genome-scale evolution

As we have described it, DNA sequences evolve down a tree under a process that includes substitutions of single nucleotides and insertions and deletions of DNA strings. Yet evolution is even more complex. For example, different biological processes such as horizontal gene transfer, hybridization, gene duplication and loss, and incomplete lineage sorting (Maddison, 1997) can cause different genomic regions to have different evolutionary histories, and make the inference of how a set of species evolved quite challenging. For example, horizontal gene transfer and hybridization require non-tree graphical models to represent the evolutionary history, and so methods that by design can only return trees are unable to correctly reconstruct these evolutionary scenarios. On the other hand, incomplete lineage sorting and gene duplication can also create tremendous heterogeneity, but a species tree is still an appropriate model. With the increased availability of sequence data, phylogenies based on multiple genes sampled from across the genomes of many species are becoming increasingly commonplace (e.g., Jarvis et al. (2014)), and it is clear that new methods are needed to estimate species trees (or networks) that take biological causes for gene tree heterogeneity into account. All this makes the design of methods for species phylogeny estimation considerably complex.

1.10 Designing methods for improved accuracy and scalability

One of the themes in this textbook is the use of algorithmic strategies to improve the accuracy or scalability of phylogeny estimation methods or multiple sequence alignment methods. Examples of such strategies include divide-and-conquer, whereby the sequence dataset is divided into overlapping subsets, trees are computed on the subsets, and a supertree is obtained by combining subset trees. Divide-and-conquer has also been applied to great success in multiple sequence alignment estimation. Iteration is another powerful algorithmic technique, both for alignment estimation and tree estimation. For example, in the context of tree estimation, each iteration uses the tree computed in the previous iteration to decompose the sequence dataset into subsets, constructs trees on subsets using a preferred phylogeny estimation method, and then combines the trees into a tree on the full dataset.

Examples of divide-and-conquer methods include SATé (Liu et al., 2009a, 2012b) and PASTA (Mirarab et al., 2015a), two methods for co-estimating multiple sequence alignments and trees; DACTAL (Nelesen et al., 2012), a method for estimating a tree without estimating an alignment; and DCM1 (Warnow et al., 2001; Nakhleh et al., 2001a), a method that is designed to improve the statistical properties of distance-based methods such as neighbor joining. Similarly, Bayzid et al. (2014) used a modification of DACTAL to improve the scalability and accuracy of MP-EST (Liu et al., 2010), a statistical method for estimating the species tree from trees computed for different parts of the genome, a topic covered in Chapter 10.

In other words, phylogeny estimation methods can be built using other phylogeny estimation methods, with the goal of improving accuracy and/or speed. However, we are also interested in establishing theoretical guarantees under stochastic models of evolution. Therefore, a proper understanding of the graph theory involved in the divide-and-conquer strategies, and the stochastic models of evolution operating on the sequence data that are used to construct the phylogenetic trees (or phylogenetic networks, as the case may be), is also important. The rest of this text provides these foundations.

1.11 Summary

We began with a discussion of some basic (and fairly simple) methods for phylogeny estimation – UPGMA, maximum parsimony, neighbor joining, and the Naive Quartet Method – and how they perform under some simple statistical models of sequence evolution. We observed that these methods have very different theoretical guarantees, and that neighbor joining and the Naive Quartet Method are both statistically consistent under standard sequence evolution models while UPGMA and maximum parsimony are not. Yet, we also observed that maximum parsimony solved heuristically can be more accurate than neighbor joining, and that the Naive Quartet Method may be unlikely to return any tree at all for large datasets, until the sequence lengths are very large (perhaps unrealistically large). Hence, knowing that a method is statistically consistent and polynomial time does not mean that it is superior on data to another method that may not be statistically consistent.

Later chapters will return to this issue, but under increasingly complex and realistic models of evolution. For example, in Chapter 8, we will discuss the standard sequence evolution models that are used in biological systematics, and the statistical methods that are used to analyze data under these models. Since these models assume sequences evolve only under substitutions, Chapter 9 addresses phylogeny estimation and multiple sequence alignment when sequences evolve also with insertions and deletions. Chapter 10 discusses species tree estimation, under genome-scale evolution models in which the different parts of the genome evolve down different trees due to various evolutionary processes. Finally, Chapter 11 describes algorithmic techniques to scale computationally intensive tree estimation methods to large datasets. In each of these chapters, we will explore the theoretical guarantees of methods as well as their performance (in terms of accuracy) on data. In many cases, the theoretical guarantees established for methods provide insight into the conditions in which they will or will not work well, but in some cases there is a gap between theory and practice.

Note that this gap does not imply that the theory is wrong, but only that it does not predict performance very well. In other words, statistical consistency is a statement about asymptotic performance, and so addresses performance given unbounded amounts of data, and theoretical guarantees about asymptotic performance do not have any direct relevance to performance on finite data.

Predicting performance on finite datasets is a fabulously interesting theoretical question, but very little has been established about this. For example, there are some upper bounds that have been established for the sequence lengths that suffice for some methods to return the true tree with high probability under simple sequence evolution models, and some lower bounds as well. But even here, the theory does not provide reliable insights into the relative performance of methods on datasets – even when those datasets are generated under the same models as the theory assumes!

Simply put, it is very difficult to predict the performance of a phylogeny estimation method based just on theory. In other words, the performance of phylogenetic estimation methods is a good example of a more general phenomenon where *in theory, there is no* *difference between theory and practice, but in practice there is*[1]. The gap between theory and practice is one of the major themes in this text, and which is one of the reasons that phylogenetic method development and evaluation is such an interesting research area.

This chapter has used simulations to complement the theoretical understanding of methods under stochastic models of evolution. Performing simulation studies is a fundamental part of research in phylogenetics, and is helpful for understanding the performance of existing methods, and hence for designing new methods with improved performance. However, real datasets are also essential, and provide important insight into the difference between how biological datasets evolve and the models of evolution that are used to describe the evolutionary process. The challenge in using biological datasets to evaluate accuracy is that the true evolutionary history is typically at best only partially known, and so differences in trees computed on biological datasets are difficult to evaluate (Iantomo et al., 2013; Morrison et al., 2015). Even so, the best understanding of algorithms and how well they can estimate evolutionary histories depends on using both types of datasets. Appendix C provides further discussion about how to evaluate methods well, including issues such as *how to simulate your data, how to vary parameters, how to select benchmarks,* and *how to report results.* The challenge to the algorithm developer is to develop methods that have outstanding performance on data and that also have the desirable theoretical guarantees of being statistically consistent and not requiring excessive amounts of data to return the true tree with high probability. Developing the theoretical framework to design methods with strong guarantees, the empirical framework to evaluate methods on data and determine the conditions in which they perform well or poorly, and algorithm design strategies (including divide-and-conquer) that can enable highly accurate methods to scale to large datasets, are the goals of the remaining chapters of this text.

1.12 Review questions

1. Consider the Cavender-Farris-Neyman (CFN) model. What are the parameters of a CFN model tree? What do these parameters mean?
2. What is meant by the CFN model tree topology?
3. What does it mean to say that a method is statistically consistent for estimating the CFN model tree topology?
4. What is the CFN distance correction? Why is it used?
5. What is the triangle inequality?
6. What are Hamming distances?
7. For a given set S of binary sequences, each of the same length, will the matrix of pairwise Hamming distances satisfy the triangle inequality? Will the matrix of pairwise CFN distances satisfy the triangle inequality?
8. What is the definition of a dissimilarity matrix?
9. What is the definition of an additive matrix?

[1] The source of this quote is unknown; it may be Yogi Berra, Jan van de Snepscheut, Walter Savitch, or perhaps others.

10. Is a square matrix in which all diagonal entries are 0 and all off-diagonal entries are 1 additive? What about ultrametric?
11. What is the Four Point Condition?
12. What is the Four Point Method? If you were given a 4×4 dissimilarity matrix, would you know how to use the Four Point Method to construct a tree on the matrix?
13. Recall the Naive Quartet Method. What is the input, and how does the Naive Quartet Method operate on the input?
14. Given a model tree and an estimated tree, each on the same set of 5 leaves, what is the maximum possible number of false positive edges?

1.13 Homework problems

1. Compute the CFN distance matrix between all pairs of sequences in the set

   - $s_1 = 0011010111$
   - $s_2 = 0011000111$
   - $s_3 = 0011111111$
   - $s_4 = 0011111110$

   Apply the Four Point Method to this dataset. What tree do you get?
2. Suppose e is an edge in a CFN model tree, and $p(e) = 0.1$. What is $\lambda(e)$?
3. Recall the definition of $\lambda(e)$ and $p(e)$ for the CFN model. Write $p(e)$ as a function of $\lambda(e)$.
4. Suppose you have a tree T rooted at leaf R, and R has two children, X and Y, and each of these nodes has two children that are leaves. Hence, T has four leaves: A and B, which are below X, and C and D, which are below Y. Draw T.
5. Suppose you are given a binary tree T on n leaves $s_1, s_2, \ldots, s_n$, with positive branch lengths. Present a polynomial time algorithm to compute the set $Q(T)$ of quartet trees. (See if you can do this in $O(n^4)$ time.)
6. Make up a CFN model tree in which the branch lengths on the edges are all different. Now compute the matrix of the 4×4 distance matrix you get using the branch lengths you wrote down. (Hence your matrix should have values for $\lambda_{A,B}, \lambda_{A,C}, \lambda_{A,D}, \lambda_{B,C}, \lambda_{B,D}$, and $\lambda_{C,D}$.)

   - What is the largest distance in the matrix?
   - What is the smallest distance in the matrix?
7. Consider a rooted tree T where R is the root, the children of R are X and Y, the children of X are A and B, and the children of Y are C and D. Consider the CFN model tree with this rooted topology, where $p(R,X) = p(R,Y) = p(Y,C) = p(Y,D) = 0.1$, and $p(X,A) = p(X,B) = 0.4$

   1. Compute the values for $\lambda(e)$ for every edge e, and draw the CFN tree with these branch lengths.

2. Compute the CFN distance of the root to every leaf. Is this distance the same for every leaf, or does it depend on the leaf?
3. Write down the matrix M of leaf-to-leaf CFN distances for this tree.
4. What is the longest leaf-to-leaf path in this tree?
5. What is the smallest positive value in the M?
6. Are the two leaves with this smallest distance siblings in the tree?
7. Write down the three pairwise sums. Which one is the smallest?
8. Is the matrix additive?

8. Consider the same rooted tree T as for the previous problem, but with $p(R,X) = p(R,Y) = p(Y,C) = p(X,A) = 0.1$, and $p(Y,D) = p(X,B) = 0.4$
   1. Compute the values for $\lambda(e)$ for every edge e, and draw the CFN tree with these branch lengths.
   2. Compute the CFN distance of the root to every leaf. Is this distance the same for every leaf, or does it depend on the leaf?
   3. Compute the matrix M of leaf-to-leaf CFN distances.
   4. What is the longest leaf-to-leaf path in this tree?
   5. What is the smallest positive value in the matrix M? Are the two leaves with this smallest distance siblings in the tree?
   6. Write down the three pairwise sums. Which one is the smallest?
   7. Is the matrix additive?
9. Consider how $\lambda(e)$ is defined by $p(e)$.
   1. Compute $\lim_{p(e)\to 0} \lambda(e)$
   2. Compute $\lim_{p(e)\to 0.5} \lambda(e)$
   3. Graph $\lambda(e)$ as a function of $p(e)$, noting that $0 < p(e) < 0.5$
10. Let A, B and C be three binary sequences, each of length k, and consider the values for $\hat{\lambda}_{A,B}, \hat{\lambda}_{A,C}$, and $\hat{\lambda}_{B,C}$. Prove or disprove: for all A, B, C, $\hat{\lambda}_{A,B} + \hat{\lambda}_{B,C} \geq \hat{\lambda}_{A,C}$ (i.e., that the triangle inequality holds for estimated CFN distances).
11. Give an example of a 4×4 normalized Hamming distance matrix H so that the Four Point Method applied to H yields a tree T that is different from the tree obtained by using the Four Point Method applied to CFN distances computed for H.
12. Consider the 4×4 matrix with 0 on the diagonal and all off-diagonal entries equal to 4. Prove that the matrix is ultrametric by drawing a rooted tree with edge weights that realizes this matrix, and show that all root-to-leaf paths have the same length.
13. Suppose you have two unrooted trees T_1 and T_2, where T_1 is binary (i.e., all non-leaf nodes have degree three) but T_2 may have nodes with degree greater than three. If you treat T_1 as the true tree and T_2 as the estimated tree, is it necessarily the case that the Robinson-Foulds (RF) error rate is the average of the FN (false negative) error rate and the FP (false positive) error rate? If so, prove it, and otherwise give a counterexample.