Part I

Basic Techniques

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 gray 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 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

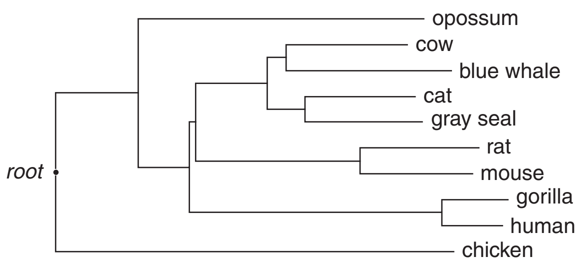


Figure 1.1 (Figure 3.5 from Huson et al. (2010)) A hypothesis of the evolutionary tree of various animals. The tree is rooted on the left, and has been estimated using molecular sequence data; branch lengths are proportional to evolutionary distances, and not necessarily to time.

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 (Neyman, 1971; Farris, 1973; Cavender, 1978). 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 ten times, each time independently generating a new trait down the tree, you would produce sequences of length ten 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 Toward Heads or Tails

Let's consider a related but obviously simpler question. Suppose you have a coin that is biased either toward heads or toward 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 toward heads; otherwise, say it is biased toward 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 (toward heads or toward 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 toward 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 toward heads or tails – you will either be 100 percent correct or 100 percent wrong. The reason you can be 100 percent correct is that there are only a finite number of choices. Note also that the probability of being 100 percent 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 toward heads or toward 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 toward heads or tails. There are only a finite number of different trees on n distinctly labeled 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 CFN 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 modeled 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.

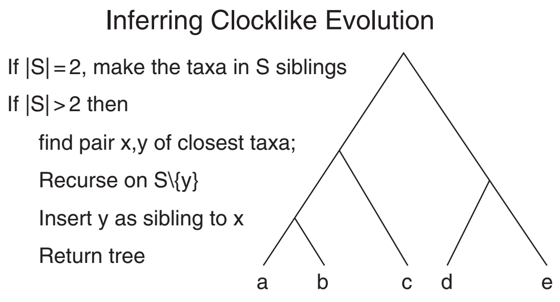
Using $\lambda(e)$ (the expected value of $N(e)$) instead of $p(e)$ turns out to be very useful in developing methods for phylogeny estimation. Note that $0 < \lambda(e)$ for all e since $p(e) > 0$. Using the properties of Poisson random variables, it can be shown that

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

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


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.

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

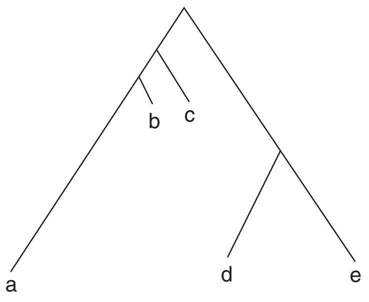


Figure 1.3 Constructing evolutionary trees when evolution is not clocklike. We show a cartoon of a model tree on five leaves, with branch lengths drawn proportionally to the expected number of changes of a random site (i.e., position in the sequence alignment). Note that leaves b and c are not siblings, 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.

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 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 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

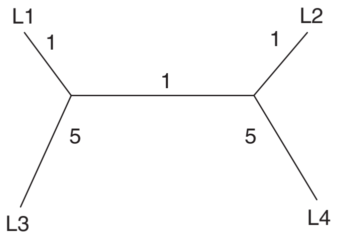
$$\lambda_{i,j} = \sum_{e \in P_{i,j}} \lambda(e).$$

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.

Definition 1.2 An $n \times n$ matrix M is **additive** if there is a tree T with leaves labeled $1,2,\ldots n$ 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



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

Figure 1.4 Additive matrix and its edge-weighted tree.

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** $s_1s_2|s_3s_4$. 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 assuming that T induces the quartet tree $s_1s_2|s_3s_4$,

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

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