Lecture Notes:
The Mathematics of Phylogenetics

Elizabeth S. Allman,
John A. Rhodes

IAS/Park City Mathematics Institute
June-July, 2005

Contents

Introduction

Inferring evolutionary relationships has been an important problem for biology since the days of Darwin. Whether the phylogenies are of interest for their own sake, or are sought for insights into other biological issues, they provide a backbone to much biological understanding. Until relatively recently, assembling evolutionary trees required that researchers devote their careers to the painstaking accumulation of detailed knowledge, often for just a handful of species.

With the advent of modern molecular biology, however, a new source of data has become available. As biological sequences such as DNA and proteins have evolved in organisms, they retained similarities to their ancestral precursors. Because sequences can be described so simply, they are well suited to mathematical methods of analyzing the similarities and differences, with the goal of inferring phylogenies. Over the past few decades a variety of tools have been developed, so that extracting evolutionary information from sequence data has now become a common application of mathematical thinking in biology. Nonetheless, challenges remain to improving methods, so that research is still actively underway.

Our goal in this course is to provide an introduction to the mathematics of molecular phylogenetics. We hope to give students a solid basis in the field, with an emphasis on the mathematics behind methods biologists routinely use. Throughout we'll try to keep mathematical prerequisites to a mid-undergraduate level, and much of what we will do is accessible without any specific background knowledge. There is much more to the field than we can possibly introduce in three weeks, yet those who want to study further should be able to proceed on their own.

The mathematics we use will range over a variety of subfields: graph theory and combinatorics, the algorithmic thinking of computer science, probability using matrix formulations of Markov models, the statistical ideas of Maximum Likelihood methods, with at least one differential equation thrown into the mix. (If we had more time, we could bring in Bayesian statistics, Fourier analysis on finite abelian groups, and even algebraic geometry, which is the basis of our own current research in phylogenetics.)

Our bibliography is a brief one, listing only a few books that provide good introductions or overviews. We call particular attention to several of these:

1) Felsenstein's book [Fel04] is the most thorough survey of phylogenetics that is available. It as written by a major figure in the development of the field, and is extremely readable. Mathematical ideas are presented less formally than in a straight mathematics book, and it pays considerable attention to biology. This makes it particularly good background for those approaching the field through mathematics.

2) The graduate-level text of Semple and Steel [SS03] is the first book focusing on phylogenetics that was written by mathematicians. While it emphasizes combinatorial issues, it covers many topics with greater mathematical precision than [Fel04].

3) The volume [Gas05], edited by Gascuel, appeared just as these notes were written. It provides excellent advanced survey articles on a number of topics in phylogenetics, indicating some of the areas of current research.

All of these books have extensive bibliographies to the research literature.

We hope these lecture notes are reasonably correct and clear, but we suspect that as the course proceeds the flaws will become clear. Please let us know both of any outright mistakes you find, and of any passages you find unclear or confusing, so that future versions can be improved.

Finally, we appreciate Cambridge University Press's permission to borrow bits from own text, [AR04], both for presentation of material and exercises. Chapters 4 and 5 of that text offer an introduction to phylogenetics, though the book is intended for use in a broader undergraduate mathematical biology course with a mixed audience of math and biology students. It therefore takes greater pains to develop some of the background material in probability and linear algebra that these notes will assume. It also omits, or treats more cursorily, some of the topics we present here.

Software

The inference of phylogenies from molecular data, on anything but small problems, and even on many of these, requires software. Although these notes will focus on mathematics, we hope that you will gain some familiarity with current practice of biologists. We strongly recommend you spend some time investigating several of the standard software packages.

While new software is constantly being written, either for specialized problems or for implementing new approaches, three general phylogenetics packages in widespread use are PAUP*, PHYLIP, and SplitsTree4.0. Fortunately, the `nexus` file format for sequence data has been adopted by most packages, and exporting data to other file formats when necessary is usually automated.

Since these packages are written for biologists to use on full datasets, they are not the best for simpler explorations and mathematical experimentation. We have found MATLAB to be a good general environment for computing of that sort, and for producing simulated data.

PAUP* Originally PAUP (Phylogenetic Analysis Using Parsimony), the * denotes a later addition of 'and other methods' to the name. These other methods include distance methods and maximum likelihood, so that the package is quite versatile. The work of David Swofford, now at Florida State University, this is sold through the publisher Sinnauer Associates for a moderate price, and may eventually become free. It is perhaps the most widely used package.

PHYLIP (Phylogeny Inference Package) is developed and freely distributed by Joe Felsenstein, at University of Washington. It covers many methods, but is not quite as complete or as simple to use as PAUP*. It is also very widely used.

SplitsTree4.0 Written by Daniel Huson (University of Tuebingen) and David Bryant (University of Auckland), this package is also freely available. One of the most interesting capabilities of this package is something we will *not* cover in this course, the construction of splits-graphs. (When there is conflicting evidence for different trees, a splits-graph depicts the conflict in an easy-to-comprehend form.) It also implements some distance methods, and is an excellent environment for exploring data.

MATLAB We have borrowed some exercises from [AR04], which make use of several MATLAB programs which are freely downloadable from the Cambridge

University Press website. While we highly recommend learning to use MATLAB well, we have only included minimal instructions for its use for these exercises.

MATLAB programs are called `m`-files, while data files are called `mat`-files, after the filename extensions on them. The `m`-files we provide have been written to minimize necessary background knowledge of MATLAB syntax. To see how to use an `m`-file in your current MATLAB directory, say `nj.m`, just type `help nj`.

A `mat`-file contains data that may only be accessed from within MATLAB. To load such a file, say `seqdata.mat`, type `load seqdata`. The names of any new variables this creates can be seen by then typing `who`, and values stored in those variables can be seen by typing the variable name.

The relevant MATLAB files among those provided for [AR04] are:

- `compseq.m` — compares two DNA sequences, producing a frequency table of the number of sites with each of the possible base combinations
- `distances.m` — computes Jukes-Cantor, Kimura 2-parameter, and log-det (paralinear) distances between all pairs in a collection of DNA sequences
- `distJC.m, distK2.m, distLD.m` — computes Jukes-Cantor, Kimura 2-parameter, or log-det (paralinear) distance for one pair of sequences described by a frequency array of sites with each base combination
- `informative.m` — locates sites in aligned DNA sequences that are informative for the method of maximum parsimony
- `markovJC.m, markovK2.m` — produces a Markov matrix of Jukes-Cantor or Kimura 2-parameter form with specified parameter values
- `mutate.m, mutatef.m` — simulates DNA sequence mutation according to a Markov model of base substitution; the second program is a function version of the first
- `nj.m` — performs the neighbor-joining algorithm to construct a tree from a distance array
- `primatedata.m` — contains mitochondrial DNA sequences from 12 primates, as well as computed distances between them
- `seqdata.mat` — contains simulated DNA sequence data
- `seqgen.m` — generates DNA sequences with specified length and base distribution

Chapter 1

Molecular Evolution

Natural selection is the fundamental mechanism through which evolution occurs, but for selection to be possible there must be some underlying variability in genetic makeup within a species. Since selection usually acts to reduce variability, there must be a source of new genetic variation. This is introduced at the molecular level, in the DNA of individuals, through what are viewed as random changes as the molecules are copied into new generations.

Depending on the nature of these changes in the DNA, offspring may be more, less, or equally viable than the parents. Many of the molecular changes are believed to be selectively neutral, and so may be passed on to further descendants and preserved, essentially 'at random.' The DNA within a particular gene may continue to mutate from generation to generation, gradually accumulating more differences from its ancestral form. Thus several species arising from a common ancestor will have similar, but often not identical, DNA forming a particular gene. The similarities hint at the common ancestor, while the differences point to the evolutionary divergence of the descendants.

Since we can now 'read' the structure of DNA with relative ease, a natural and compelling question arises: Can we reconstruct evolutionary relationships between several modern species by comparing the DNA sequences of their versions of a certain gene?

We of course expect that species that have more similar genetic sequences are probably more closely related. However, this observation really isn't enough to make clear how to infer an evolutionary tree relating a large number of different species, all with varying degrees of similarity in the chosen gene.

Before we can develop more elaborate mathematical ideas for how the problem of phylogenetic inference can be attacked, we need to briefly recount some biological background.

1.1 DNA structure

The DNA molecule forms a double helix, a twisted ladder-like structure. At each of the points where the ladder's upright poles are joined by a rung, one of four possible molecular subunits appears. These subunits, called *nucleotides* or *bases*, are adenine, guanine, cytosine, and thymine, and are denoted by the letters A, G, C, and T. Because of chemical similarity, adenine and guanine are called *purines*, while cytosine and thymine are called *pyrimidines*.

Each base has a complementary base with which it can form the rung of the ladder through a hydrogen bond. We always find either A paired with T, or G paired with C. Thus knowing one side of the ladder structure is enough to deduce the other. For example if along one pole of the ladder we have a sequence of bases

`AGCGCGTATTAG`,

then the other would have the complementary sequence

`TCGCGCATAATC`.

Finally, the DNA molecule has a directional sense (distinguished by what are called its 5' and 3' ends) so that we can make a distinction between a sequence like $ATCGAT$ and the inverted sequence $TAGCTA$. The upshot of all this structure is that we will be able to think of DNA sequences mathematically simply as finite sequences composed from the four-letter alphabet $\{A, G, C, T\}$.

Some sections of DNA form *genes* that encode instructions for the manufacturing of proteins (though the production of the protein is accomplished through the intermediate production of messenger RNA). In these genes, triplets of consecutive bases form *codons*, with each codon specifying a particular amino acid to be placed in the protein chain according to the *genetic code*. For example the codon TGC always means that the amino acid cysteine will occur at that location in the protein. Certain codons also signal the end of the protein sequence. Since there are $4^3 = 64$ different codons, and only 20 amino acids and one 'stop' command, there is some redundancy in the genetic code. For instance, in many codons the third base has no affect on the particular amino acid the codon specifies.

While originally it was thought that genes always encoded for proteins, we now know that some genes encode the production of other types of RNA which are the 'final products' of the gene, with no protein being produced. Finally, not all DNA is organized into the coding sections referred to as genes. About 97% of human DNA, for example, is believed to be non-coding. Some of this is likely to be meaningless raw material (sometimes called *junk DNA*) which may, of course, become meaningful in future generations through evolution. Other parts of the DNA molecules serve regulatory purposes. The picture is quite complicated and still not fully understood.

1.2 Mutations

Before DNA, and the hereditary information it carries, is passed from parent to offspring, a copy must be made. For this to happen, the hydrogen bonds forming the rungs of the ladder in the molecule are broken, leaving two single strands. Then new double strands are formed on these, by assembling the appropriate complementary strands. The biochemical processes are elaborate, with various safeguards to ensure that few mistakes are made in the final product. Nonetheless, changes of an apparently random nature sometimes occur.

The most common mutation that is introduced in the copying of sequences of DNA is a *base substitution*. This is simply the replacement of one base for another at a certain site in the sequence. For instance, if the sequence $AATCGC$ in an ancestor becomes $AATGGC$ in a descendant, then a base substitution $C \rightarrow G$ has occurred at the fourth site. A base substitution that replaces a purine with a purine, or a pyrimidine for a pyrimidine, is called a *transition*, while an interchange of these classes is called a *transversion*. Transitions are often observed to occur more frequently than transversions, perhaps because the chemical structure of the molecule changes less under a transition than a transversion.

Other DNA mutations sometimes observed include the deletion of a base or consecutive bases, the insertion of a base or consecutive bases, and the inversion (reversal) of a section of the sequence. All these mutations tend to be seen more rarely in natural populations. Since these types of mutations usually have a dramatic effect on the protein for which a gene encodes, this is not too surprising. We'll ignore such possibilities, in order to make our modeling task both clearer and mathematically tractable.

Our view of molecular evolution, then, can be depicted by an example such as the following, in which we have an ancestral sequence S0, its descendant S1, and a descendant S2 of S1.

S0 : `ATGTCGCCTGATAATGCC`
S1 : `ATGCCGCTTGACAATGCC`
S2 : `ATGCCGCGTGATAATGCC`

Notice site 4 underwent a net transversion from S0 to S1, and then no further net mutation as S2 evolved. If we were only able to observe S0 and S2, we would still see evidence of one net transversion in this site. However, while site 8 experienced at least two mutations, if we only saw the sequences S0 and S2 we could only be sure that one occurred. Site 12 shows a similar situation, where at least two mutations occurred, yet comparing S0 and S2 alone would show no evidence of either.

This illustrates that there may be *hidden mutations*, such as $C \rightarrow T \rightarrow G$, in which subsequent substitutions obscure earlier ones from our observation when we do not have access to sequences from all generations. A *back mutation* such as $T \rightarrow C \rightarrow T$ is simply a more extreme case of this. The distinction

between observed mutations and actual mutations may be an important one when considering data.

In reality, seldom do we actually have an ancestral DNA sequence, much less several from different times along a line of descent. Instead, we have sequences from several currently living descendants, but no direct information about any of their ancestors. When we compare two sequences, and imagine the mutation process that produced them, the sequence of their *most recent common ancestor*, from which they both evolved, is unknown. This will produce additional complications as our analysis becomes more sophisticated.

1.3 Aligned Orthologous Sequences

Given a DNA sequence from some organism, there are good search algorithms (BLAST) to find similar sequences for other organisms in DNA databases. Thus if a gene has been identified for one organism, we can quickly locate likely candidate sequences for similar genes in related organisms. Perhaps after experimentally verifying that these are in fact genes and that they have a similar function, we can reasonably assume the sequences are *orthologous*, meaning they descended from a common ancestral sequence.

Either by trial and error, or using algorithms we will not discuss in these notes, we can *align* the sequences from the different organisms so that many of the bases match across most of the sequences. For some data sets, alignment is a trivial problem, since so much of the sequences match exactly. For other data sets, finding good alignments may be quite difficult. Faced with a large number of sequences, with much variation between them, even the best of current software may not produce an alignment that on human inspection appears very good. In the end, a mix of algorithms and *ad hoc* human adjustment is sometimes used to get better results. For those interested in learning more about sequence alignment, we suggest [Wat95] as a beginning source.

We take as the starting point for the techniques we discuss is these notes a collection of *aligned orthologous DNA sequences*. Our goal will be to produce a *phylogenetic tree* that describes their likely descent from a common ancestral sequence.

Chapter 2

Combinatorics of Trees I

2.1 Graphs and Trees

Before we discuss any of the methods of inference of phylogenetic trees, we introduce some basic background and terminology from graph theory. This simply formalizes the basic notions of the kinds of trees that biologists use to describe evolution, and gives us useful language.

Definition. A *graph* G is pair $G = (V, E)$ where $V = V(G)$ is a set of *vertices* and $E = E(G)$ is a set of *edges*. Each edge $e \in E$ is a two-element multiset $e = \{v_1, v_2\}$ of vertices $v_1, v_2 \in V$. (The terminology *multiset* allows $v_1 = v_2$, but also means $\{v_1, v_2\} = \{v_2, v_1\}$.)

When $e = \{v_1, v_2\}$, we say v_1 and v_2 are the *ends* of e, that e *joins* v_1 and v_2, that v_1 and v_2 are *incident* to e, and that v_1 and v_2 are *adjacent*.

The *degree* of a vertex is the number of edges to which it is incident.

All of our graphs will be *finite*, meaning V, and hence E, is a finite set. While we could allow distinct edges to have the same ends, that will not be necessary for our applications. We therefore assume all graphs are *simple*, in that the ends of an edge uniquely determine that edge.

Graphs are typically indicated by drawings such as that in Figure 2.1.

In phylogenetics, we often think of each vertex as representing a species, with the edges indicating lines of evolutionary descent. Rather than species, however, we may be working with smaller groups of organisms such as subspecies or even individuals, or larger groups such as genera, etc. Thus we use the more neutral word *taxon* for whatever group is under consideration.

If we are to use graphs to model evolution, with edges indicating lines of descent, then we of course need to rule out any taxon being its own ancestor. To be precise about this, we need a series of definitions.

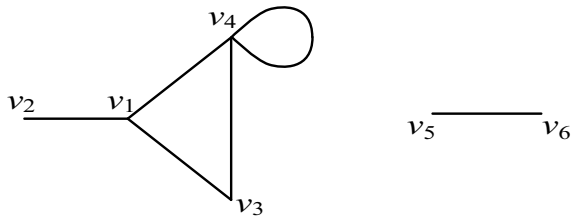

Figure 2.1: A depiction of a graph $G = (V, E)$ with $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_3, v_4\}, \{v_4, v_4\}, \{v_5, v_6\}\}$

Definition. A *path of length* n from vertex v_0 to vertex v_n in a graph is a sequence of vertices $v_0, v_1, v_2, \ldots, v_n$ such that $e_i = \{v_{i-1}, v_i\}$ is an edge for $i = 1, \ldots, n$.

A graph is *connected* if there is a path between any two vertices.

A *cycle* in a graph is a path $v_0, v_1, \ldots, v_n = v_0$ from v_0 to itself with 'no backtracking,' in the sense that $v_i \neq v_{i+2}$ for any i.

A *tree* $T = (V, E)$ is a connected graph with no cycles.

As an example of a tree, we have that depicted in Figure 2.2.

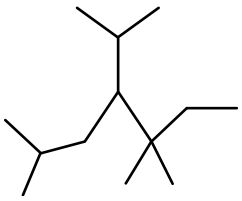

Figure 2.2: A tree.

If a vertex lies in two or more distinct edges of a tree, we say it is an *interior vertex*. If it lies in only one edge, then we call it a *terminal vertex* or a *leaf*.

We will occasionally need to use the following property of trees.

Theorem 1. If v_0 and v_1 are any two vertices in a tree T, then there is a unique path of minimal length from v_0 to v_1.

Proof. We leave a formal proof as an exercise, but summarize the argument: Given two paths without backtracking from v_0 to v_1, we can use them to construct a path going from v_0 through v_1 and then back to v_0. If a graph has no cycles, we deduce that the two paths are the same. □

We call the path described in the theorem the *minimal path* between the vertices, and refer to the number of edges in this path as the *graph-theoretic distance* between the vertices. For example, in Figure 2.3, for the tree on the

left the graph-theoretic distance from a to d is 3, while for the other two graphs it is 4.

To those who are familiar with the notion of a directed graph, it may seem odd that we focus on undirected trees for depictions of evolutionary histories. Indeed, time provides a direction that is of course central to evolutionary processes. However, we will see that many of the mathematical models and inference methods are more naturally associated with undirected trees, and so we make them our basic objects.

With that said, however, sometimes we will pick a particular vertex ρ in a tree and distinguish it as the *root* of the tree. (We may sometimes even introduce a new vertex to be the root, replacing an edge of the original tree with two edges meeting at the new root vertex.) Biologically, we would like to think of a root as the most recent common ancestor of all the taxa in the tree. Sometimes, however, roots are chosen for mathematical convenience rather than biological meaning, so one must be careful with such an interpretation. We use T to denote an unrooted tree, and T^ρ to denote a tree with a choice of a root ρ.

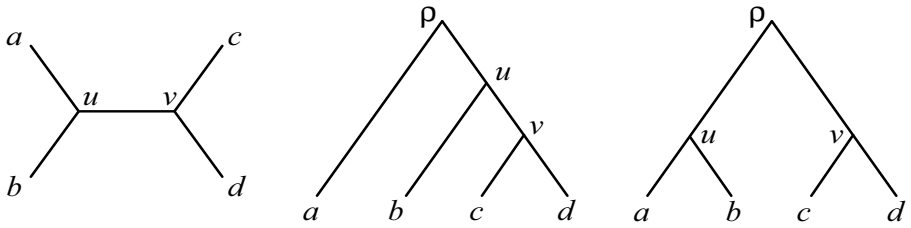

Figure 2.3: An unrooted tree and two rooted counterparts

Viewing ρ as a common ancestor of all other taxa represented in a tree, it is natural to give each edge of the tree an orientation away from the root. More precisely, we treat edges not as multisets, but rather as ordered pairs $e = (v_1, v_2)$, where the order is specified by requiring that the minimal path from ρ to v_1 is shorter than that from ρ to v_2. We say v_1 is the *initial vertex* or *tail* of e and that v_2 is the *final vertex* or *head* of e. We also refer to v_1 as the *parent* of v_2, and v_2 as the *child* of v_1, and more generally use the words *ancestor* and *descendant* in the obvious way.

An unrooted tree is said to be *binary* if each interior vertex has degree three. This terminology is of course motivated by the biological view of a split of one lineage into two, but this leads to the rather odd situation that a synonym for binary is *trivalent*. We call a rooted tree T^ρ binary if all interior vertices other than ρ are of degree three, while ρ is of degree two.

Although it's conceivable biologically that a tree other than a binary one might describe an evolutionary lineage, it is common to ignore that possibility as extremely unlikely. When non-binary trees arise in phylogenetic applications, they usually have vertices with more than 3 edges joining at a vertex to indicate

ambiguity arising from ignorance of a true binary tree, rather than positive knowledge. Biologists generally prefer that trees be 'highly resolved', and a binary tree is the goal.

We should note that by choosing to focus on trees, we are already making biologically important assumptions on what we intend to study. In plants, for instance, hybridization is not uncommon, and thus evolution can be quite non-tree-like. More generally, in recent years we've learned that *lateral gene transfer* can occur in other ways between two taxa that live concurrently, so that two lines of descent from an ancestral taxon may 'rejoin' in a descendant one. Though it remains controversial how common such interchanges are, the possibility is well established. Nonetheless, trees remain a good mathematical idealization of the main story of evolutionary descent, and developing useful non-tree representations and models of evolution remains a fairly open area.

The trees used in phylogenetics have a final distinguishing feature — the leaves represent known taxa, which are typically currently extant and the source of whatever data is used to infer the tree. The internal vertices, in contrast, usually represent taxa that are no longer present, and from which we have no direct data. This is formalized by labeling the leaves of the tree with the names of the known taxa, while leaving unlabeled the interior vertices. Thus for either rooted or unrooted trees we are interested primarily in the following objects.

Definition. Let X denote a finite set of taxa, or labels. Then a *phylogenetic X-tree* is a tree together with a bijection $\phi : X \to L$, where $L \subseteq V$ denotes the set of leaves of the tree. We call ϕ the *labeling map*.

Often we can and will blur the distinction between the leaves of a phylogenetic tree and the labels that are assigned to them. For instance, we will use T or T^ρ to denote a phylogenetic X-tree, often without explicitly mentioning either the set X or the labeling function. However, it is important to note that this labeling is crucial. Two different bijections from X to the leaves of T produce two different phylogenetic trees. In other words, phylogenetic trees can be distinguished from one another either due to different topologies of the underlying trees, or merely by a different labeling of the leaves.

2.2 Counting Binary Trees

Suppose we are interested in relating a collection X of n taxa by a phylogenetic tree. How many such trees might describe the relationship?

To refine the question, both for simplicity and for biological relevance, let $b(n)$ denote the number of distinct (up to isomorphism of partially-labeled graphs) unrooted binary phylogenetic X-trees, where $X = \{1, 2, \ldots, n\}$. One quickly sees $b(2) = 1$, $b(3) = 1$, and $b(4) = 3$, as the diagrams in Figure 2.4 indicate.

The key observation for counting larger trees then appears clearly when we consider $b(5)$: We can begin with any one of the 3 trees relating the first four

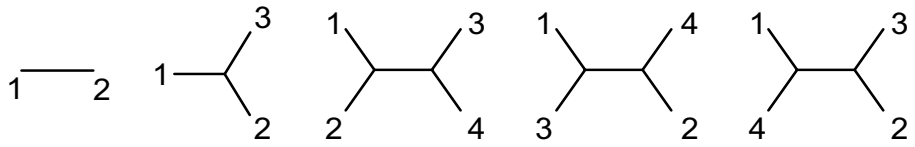

Figure 2.4: All unrooted binary phylogenetic trees for $X = \{1, 2, \ldots, n\}$, $n = 2, 3, 4$.

taxa, and 'graft' another edge leading to a fifth taxon to any of the 5 edges in that four-taxon tree, so that $b(5) = 3 \cdot 5 = 15$. For $b(6)$, we can begin with any of these 15 trees and graft on another edge leading to a sixth taxon to any of the edges in that tree. To obtain a general formula, then, we also need to obtain a formula for the number of edges in these trees.

Theorem 2. An unrooted binary tree with $n \geq 2$ leaves has $2n - 2$ vertices and $2n - 3$ edges.

Proof. The statement is certainly true for $n = 2$, which is the base case for induction.

Suppose T has $n \geq 3$ leaves, and assume the statement is true for a tree with $n-1$ leaves. If v_1 is one of the leaves of T, then v_1 lies on a unique edge $\{v_1, v_2\}$, while v_2 lies additionally on two other edges $\{v_2, v_3\}$ and $\{v_2, v_4\}$. Removing these three edges and the vertices v_1, v_2 from T, and introducing a new edge $\{v_3, v_4\}$ gives a binary tree T' with $n - 1$ leaves. Since both the number of vertices and the number of edges have been decreased by 2, for T the number of vertices must have been $(2(n-1)-2)+2 = 2n-2$, and the number of edges $(2(n-1)-3)+2 = 2n-3$. □

Theorem 3. If $|X| = n \geq 3$, there are

$$b(n) = (2n-5)!! = 1 \cdot 3 \cdot 5 \cdots (2n-5)$$

distinct unrooted binary phylogenetic X-trees.

Proof. Again we use induction, with the base case of $n = 3$ clear.

Suppose then that T has n leaves, and let T' be the $(n-1)$-leaf tree constructed from T as in Theorem 2 by 'removing' the leaf v_1 and adjusting edges appropriately. Then with v_1 fixed, the map $T \mapsto (T', \{v_3, v_4\})$ is a bijection from n-leaf trees to pairs of $(n-1)$-leaf trees and edges. In this pair, we think of the edge as the one onto which we 'graft' a new edge to v_1 to recover T. Counting these pairs shows

$$b(n) = b(n-1) \cdot (2(n-1)-3) = b(n-1) \cdot (2n-5).$$

But since we inductively assume $b(n-1) = (2(n-1)-5)!! = (2n-7)!!$, we obtain the desired formula. □

This last theorem also yields a count for rooted binary trees, by simply noting that adjoining to any rooted binary tree T^ρ a new edge $\{\rho, v_{n+1}\}$, where v_{n+1} is a new leaf, gives a bijection between rooted binary trees with n leaves and unrooted binary trees with $n+1$ leaves.

Corollary 4. The number of rooted binary phylogenetic X-trees relating n taxa is

$$b(n+1) = (2n-3)!! = 1 \cdot 3 \cdot 5 \cdots (2n-3).$$

Of course the formula for $b(n)$ shows it is a very large number even for relatively small values of n. (See exercises.) The implication of this for phylogenetic inference will quickly become clear: Any inference method that relies on considering every possible binary tree will be slow if the number of taxa is at all large.

2.3 Metric Trees

Sometimes it is useful to specify lengths of edges in a trees with non-negative numbers, as in Figure 2.5 below. Note that the lengths may either be specified by explicitly writing them next to the edges, or by simply drawing edges with the appropriate lengths.

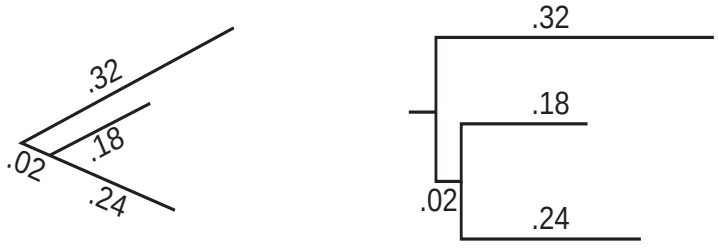

Figure 2.5: Metric trees

Biologically, these lengths are usually interpreted as some sort of measure of how much change occured between the taxa at the ends of the edge, with larger numbers denoting more change. Sometimes they represent elapsed time, though this is less common. To emphasize when lengths are not specified, we will sometimes refer to a tree without edge lengths as a *topological tree*, in contrast to a *metric tree*, which we now define.

Definition. A *metric tree* (T, w) is a rooted or unrooted tree T together with a function $w : E(T) \to \mathbb{R}^{\geq 0}$. We call $w(e)$ the *length* or *weight* of the edge e.

Notice the conflict between the graph-theoretic notion of length of a path (the number of edges in the path), and this new use of the word length. To avoid confusion, graph theorists tend to prefer the term 'weight,' but unfortunately 'length' is more common in phylogenetics.

A metric tree leads to a way of measuring distances between vertices. For any $v_1, v_2 \in V(T)$, define

$$d(v_1, v_2) = \sum_{\substack{e \text{ on the minimal} \\ \text{path from } v_1 \text{ to } v_2}} w(e),$$

where the minimal path still means the one of shortest graph-theoretic length.

Proposition 5. For a metric tree (T, w), the function $d : V \times V \to \mathbb{R}^{\geq 0}$ satisfies

(i) $d(v_1, v_2) \geq 0$ for any v_1, v_2 (non-negativity),

(ii) $d(v_1, v_2) = d(v_2, v_1)$ for any v_1, v_2 (symmetry),

(iii) $d(v_1, v_3) \leq d(v_1, v_2) + d(v_2, v_3)$ for any v_1, v_2, v_3 (triangle inequality).

If all edges of T have positive length, then, in addition,

$$d(v_1, v_2) = 0 \text{ if, and only if, } v_1 = v_2.$$

Proof. See exercises. □

If all edges of T have positive length, then the proposition above shows d is a *metric* on $V(T)$ in the usual sense of the word in topology or analysis. In this case we call d a *tree metric*.

If T has some edges of length zero, we can of course remove those edges, identifying their two ends, to get a metric tree with all positive edge lengths that carries essentially the same information as our original metric tree. (However, if an edge leading to a leaf has length zero, then the resulting tree may no longer be a phylogenetic X-tree, as a label may now be on an internal vertex.) On this collapsed tree we will then have that d is a tree metric. As it is often convenient to work with metric trees that have positive edge lengths, keeping this process in mind we will lose little by making such an assumption at times.

2.4 Molecular Clock Trees

While it may be the case that for a rooted metric phylogenetic X-tree all leaves are equidistant from the root according to a tree metric d, such an assumption is usually not made without explicitly mentioning it. In the case of trees inferred from biological sequence data, this is often referred to as a *molecular clock* hypothesis.

More formally, a molecular clock hypothesis states that mutations occur at a constant rate throughout the evolutionary tree. Since leaves are viewed as currently extant species, they will then all have experienced the same amount of mutation since the time of their most recent common ancestor. In this circumstance all edge lengths can be interpreted as proportional to the amount of elapsed time between the presence of the taxa at the ends of the edges.

Sometimes the molecular clock assumption is biologically reasonable, for instance, if all taxa are reasonably closely related and one suspects little variation in evolutionary processes during the time under study. Other times it is less plausible, if more distantly related taxa are in the tree, and the circumstances under which they evolved may have changed considerably throughout the tree. Regardless, one attraction of a molecular clock hypothesis is that it gives a way of locating a tree root.

Theorem 6. Suppose T^ρ is a rooted metric tree with positive edge lengths and all leaves equidistant from the root ρ according to the tree metric. Let v_1, v_2 be any two leaves with $d(v_1, v_2)$ maximal among distances between leaves. Then ρ is the unique vertex along the graph-theoretic minimal path from v_1 to v_2 with $d(\rho, v_1) = d(\rho, v_2) = d(v_1, v_2)/2$.

Proof. If T^ρ has more than one leaf, then since ρ is equidistant from the leaves it must be an internal vertex. Thus deleting ρ (and its incident edges) from T produces at least two connected components.

Suppose v_1 and v_2 are in different connected components of $T \setminus \{\rho\}$. Then the minimal path from v_1 to v_2 must pass through ρ, and we see by Exercise 10 that $d(v_1, v_2) = d(v_1, \rho) + d(\rho, v_2) = 2d(v_1, \rho)$. Thus ρ lies the specified distance from v_1 and v_2. Since edge lengths are positive, there can be only one such vertex on the path with this property.

If v_1 and v_2 are in the same connected component of $T \setminus \{\rho\}$, then the minimal path between them does not pass through ρ and so by Exercise 10 and the triangle inequality we have $d(v_1, v_2) < d(v_1, \rho) + d(\rho, v_2)$. But since all leaves are equidistant from the root, this shows $d(v_1, v_2) < 2d(v_1, \rho)$, and so the maximal distance between leaves cannot occur between v_1 and v_2. □

2.5 Rooting Trees with Outgroups

In the absence of a molecular clock hypothesis, locating the root of a tree is generally not possible from mathematical considerations alone. Instead, a simple biological idea can often be used. We simply include an extra taxon in our study, beyond those we are primarily interested in relating. If this taxon can be assumed to be more distantly related to each of the taxa of primary interest than any of those are to each other, we call it an *outgroup*. For instance, if we are primarily interested in relating a number of species of duck, we might include some non-duck bird as an outgroup.

Provided we infer a good evolutionary tree for all our taxa including the outgroup, the vertex at which the outgroup is joined to the taxa of primary interest represents the most recent common ancestor of all, and hence serves as a root for the subtree relating only the taxa of primary interest. Thus we use prior biological knowledge of the relationship of the outgroup to the other taxa to solve our rooting problem.

This is illustrated in Figure 2.6, where the taxon o designates the outgroup.

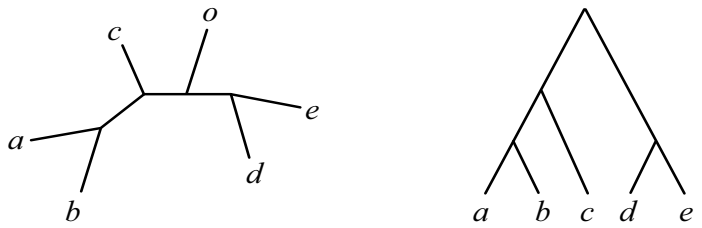

Figure 2.6: Using an outgroup o to infer a root

2.6 Exercises

1. Consider the trees in Figure 2.7.

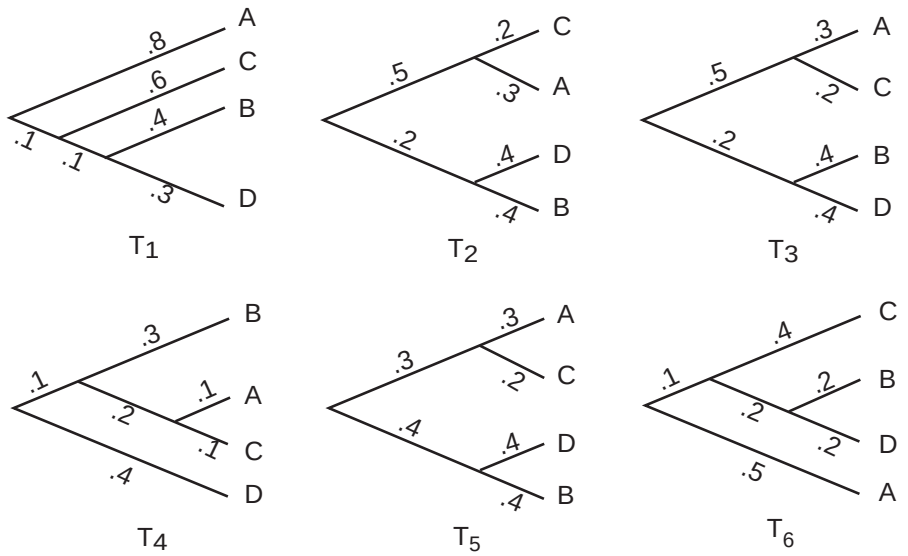

Figure 2.7: Trees for Problem 1

   a. Which of them are the same, as rooted metric trees?

   b. Which of them are the same, as unrooted metric trees, provided the root is deleted and its two incident edges joined into one?

   c. Which of them are the same, as rooted topological trees?

   d. Which of them are the same, as unrooted topological trees, provided the root is deleted and its two incident edges joined into one?

   e. For which trees does a molecular clock appear to be operating?

2. Draw the three topologically-distinct rooted binary trees that could describe

the relationship between 3 taxa a, b, c. How do they naturally relate to the three unrooted 4-taxon trees in Figure 2.4?

3. Draw the 15 unrooted binary trees that could describe the relationship between 5 taxa a, b, c, d, e. Group them naturally according to the relationship they display for the first 4 taxa a, b, c, d.

4. Make a table of values of the function $b(n)$, giving the number of unrooted binary X-trees for $|X| = n \leq 10$, and then graph this function.

5. Show that $b(n) = \dfrac{(2n-5)!}{2^{n-3}(n-3)!}$.

6. Since mitochondrial DNA in humans is inherited solely from the mother, it can be used to construct a tree relating any number of humans from different ethnic groups, assuming we all descended from a single first human female. Depending on the clustering pattern of the ethnic groups this might give insight into the physical location of this woman, who is sometimes called Mitochondrial Eve.

   In 1987 a work by Cann, Stoneking, and Wilson, first purported to locate Mitochondrial Eve in Africa, supporting the 'out of Africa' theory of human origins. A rooted tree was constructed that was claimed to show the relationships between 147 individuals. How many topologically different trees would need to be looked at if every possibility was really examined? (You may need to use the statement in problem 5 and Stirling's formula: $n! \sim \sqrt{2\pi} n^{n+\frac{1}{2}} e^{-n}$. If you are not familiar with the asymptotic symbol '$\sim$' you can loosely interpret it as meaning 'is approximately.')

7. Give a complete proof of Theorem 1, that if T is a tree and $v_1, v_2 \in V(T)$ then there is a unique graph-theoretic minimal path from v_1 to v_2.

8. Show that a path from v_1 to v_2 in a tree T is a minimal path if, and only if, it has no backtracking.

9. Prove Proposition 5.

10. Suppose T is a metric tree with all positive edge lengths. Prove that v_3 lies on the graph-theoretic minimal path from v_1 to v_2 if, and only if, $d(v_1, v_2) = d(v_1, v_3) + d(v_3, v_2)$. Show by example that this may be false if zero-length edges exist.

11. The phylogeny of four terminal taxa A, B, C, and D are related according to a certain metric tree. The tree metric distances between taxa have been found to be as in Table 2.1

    a. Using any approach you wish, determine the correct unrooted tree relating the taxa, as well as all edge lengths. Explain how you rule out other topological trees.

    b. Can you determine the root from this data? Explain why or why not.

	A	B	C	D
A		.6	.6	.2
B			.4	.6
C				.6

Table 2.1: Distances between taxa for Problem 11

c. If you invented a method to recover the metric tree, can you generalize it to work on a similar table for 5 taxa? Create a larger table of distances coming from a metric tree and try it.

d. Suppose you only had a table of numbers that were approximately those coming from a tree metric. Would your method still work? (This is the situation for real phylogenetic problems.)

Chapter 3

Parsimony

The first method we'll discuss for inferring phylogenetic trees is called the method of Maximum Parsimony (MP), or, more informally, parsimony. The essential idea is probably the oldest of those underlying any method, and is applied both to inferring phylogenies from sequence data and from data of other types, such as morphological data.

3.1 The Parsimony Criterion

Suppose we obtained the following aligned DNA sequences for four taxa:

S1 : `AACTTGCGCATTATC`
S2 : `ATCTTGCGCATCATC`
S3 : `ATCTTGGGCATCATC`
S4 : `AACTTGGGCATTATC`

Note the only variations in bases are at sites 2, 7, and 12. We might interpret sites 2 and 12 as evidence that S1 and S4 are closely related, while site 7 might support S1 and S2 being closely related. Since these conflict, we might debate and decide in favor of a topological tree with S1 and S4 joined to a common internal vertex. This is perhaps the more likely candidate for the true evolutionary history, since it seems to require fewer mutations overall than the other possibilities. Both sites 2 and 12 could be explained by a single mutation in a lineage early in the evolutionary history, before the final divergence into the 4 taxa we now see.

To generalize this viewpoint, imagine having n taxa which we wish to relate.

We collect a data sequence of *characters* which for each taxon might be in any of a finite number of *states*. For instance, if our data is aligned orthologous DNA sequences, then the characters correspond to the different sites in the sequences, and their possible states on each of the taxa are $\{A, G, C, T\}$. If

our data is morphological, then we might have characters referring to wingedness, with states {straight, curved}, or some other body part with states such as {segmented, unsegmented, absent}. Other sources of characters might be genetic or biochemical features that vary among the taxa. Thus while our examples below will use DNA data, the method applies more generally.

The simplest form of the principal underlying parsimony is:

> *The best tree to infer from data is the one requiring the fewest changes in states of characters.*

Informally, then, a 'good' tree is one that could describe an evolutionary history of as little change as possible. Some philosophical justifications for this view have been given along the lines of Ockham's razor, that we should prefer scientific explanations that are as simple as possible, minimizing the number of events that we must hypothesize have occurred though we lack direct evidence.

We'll first explore how to apply this principle, deferring comments on how reasonable it might be for various types of biological data.

To mathematically formalize the idea we make a number of definitions:

Definition. A *character* with state set S on a set Z is a function $\chi : Z \to S$. If $s = |S|$, we say χ is an s-state character.

We will assume all sets are finite here. By passing to a numerical encoding of states, we could of course even assume $S = \{1, 2, \ldots, s\}$.

Our data for the parsimony problem for a set of taxa X is then a finite sequence of characters $\mathcal{C} = \{\chi_1, \chi_2, \ldots, \chi_k\}$ where χ_i is an s_i-state character on X for each i. For DNA data, we generally have $s_i = 4$ for all i, though for morphological data the s_i may vary.

Note that we refer to a *sequence* of characters since that terminology is mathematically valid and in accord with biological usage when, say, DNA sequences are used, with each site corresponding to a character. However, we could more generally have referred to a *multiset* of characters since we will not in fact use the specific ordering. Recall that multisets are unordered, but allow an element to be repeated, unlike sets. For morphological data, there is generally no natural ordering to the characters, so the multiset terminology would be more natural.

Consider a fixed phylogenetic X-tree T with leaves labeled by the taxa in X. For the remainder of this chapter we'll make no distinction between the leaves of T and their labels in X.

While our data is composed of characters χ on X, to judge the number of state changes T requires, we also need to consider characters $\tilde{\chi}$ on the full vertex set $V(T)$. In accordance with standard terminology for functions, we say a character $\tilde{\chi}$ on $V(T)$ is an *extension* of χ to T if $\tilde{\chi}(x) = \chi(x)$ for all $x \in X$. We use $Ext_T(\chi)$ to denote the set of all extensions of χ to T, and think of its elements as representing the possible evolutionary histories that are consistent with the observation of χ.

For each edge $e = (v, w) \in E(T)$ and character $\tilde{\chi}$ on $V(T)$, define

$$\delta(e, \tilde{\chi}) = \begin{cases} 1 & \text{if } \tilde{\chi}(v) \neq \tilde{\chi}(w) \\ 0 & \text{otherwise} \end{cases}.$$

Viewed as a function of e, with $\tilde{\chi}$ fixed, this is the indicator function of those edges where a state change occurs in the evolutionary history $\tilde{\chi}$.

Definition. The *state-change count* for $\tilde{\chi}$ on T is

$$c(\tilde{\chi}) = \sum_{e \in E(T)} \delta(e, \tilde{\chi}).$$

The *parsimony score* of T for χ is

$$ps_{\chi}(T) = \min_{\tilde{\chi} \in Ext_T(\chi)} c(\tilde{\chi}),$$

and we say $\tilde{\chi}$ is a *minimal extension* of χ for T if

$$c(\tilde{\chi}) = ps_{\chi}(T).$$

The *parsimony score* of T for a sequence of characters $\{\chi_1, \ldots, \chi_k\}$ is

$$ps_{\{\chi_i\}}(T) = \sum_{i=1}^{k} ps_{\chi_i}(T).$$

The set of *most parsimonious trees* for a sequence of characters $\{\chi_1, \ldots, \chi_k\}$, then, is the collection of trees achieving the minimal parsimony score:

$$\{T \mid ps_{\{\chi_i\}}(T) = \min_{T'} ps_{\{\chi_i\}}(T')\}.$$

Proceeding naively, we now have two difficulties in finding the most parsimonious trees. First, we will need to consider all possible trees that might relate our taxa. We've already dealt with enumerating these, so at least in principal we could methodically consider one tree after another. However, we've seen that for large n, the number of n-taxon bifurcating phylogenetic trees is enormous.

Second, for each tree we consider, to compute the parsimony score we apparently must consider all possible extensions of the data characters. Even if each character has only 2 states, since the number of internal nodes of an n-taxon binary phylogenetic tree is $n - 2$, the number of such extensions grows exponentially with n. Considering each of these in turn would be a disastrous method to actually try to use for large n.

3.2 The Fitch-Hartigan Algorithm

Fortunately, the second of these problems, the *small parsimony problem* of computing $ps_{\{\chi_i\}}(T)$ for a *fixed* tree T, is not as bad as it appears. A simple algorithm was developed independently by Fitch and Hartigan. Although it can

be generalized, we present it here only for bifurcating trees. We illustrate and motivate it by an example:

Suppose we look at a single character (in this example, a site in the DNA) for each of five taxa and have states (bases) as shown:

S1: `A`, S2: `T`, S3:`T`, S4:`G`, S5:`A`.

For the leaf-labeled tree T of Figure 3.1 we wish to compute $ps(T)$.

We will simply trace backwards up the tree, from the leaves to the root, drawing reasonable conclusions as to what the state of the character might be at each interior vertex, assuming the fewest possible state changes occurred. For instance, above S1 and S2 we could have had either an A or a T, but obviously not a C or a G, if we are to minimize state changes. Furthermore at least 1 state change occurred. We thus label that vertex with the set of possibilities $\{A, T\}$ and have a count of 1 so far. However, given what appears at S3, at the vertex joining S3 to S1 and S2, we should have a T (and in fact a T at the descendant vertex joined to S1 and S2); no additional change is necessary, beyond the 1 we already counted. We've now labeled two interior vertices, and still have a count of 1.

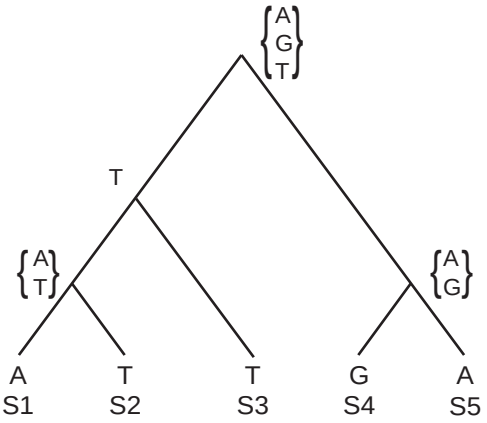

Figure 3.1: Computing the parsimony score for a tree at one site

We continue to trace backward through the tree, placing a base or set of possible bases at each vertex. If below the vertex are two different bases (or disjoint sets of bases), we need to increase our state change count by 1 and combine the two bases (or the sets) to create a larger set of possible bases at the higher vertex. If the two lower bases agree (or the sets are not disjoint), then we label the higher vertex with that base (or the intersection of the sets). In this case no additional changes need to be counted. When all the vertices of the tree are labeled, the final value of the mutation count gives the minimum number of mutations (or state changes) needed if that tree described the evolution of the taxa. Thus the tree in Figure 3.1 would have a parsimony score of 3.

While this illustrates the algorithm, there are several points to be addressed. First, while it may seem reasonable, it is not obvious that this method gives the minimum possible mutations needed for the tree. Perhaps surprisingly, there *can* be assignments of bases to the internal vertices that are not consistent with the assignment this method produces, yet which still achieve the same minimum number of mutations.

Second, we performed the algorithm on a *rooted* tree. While we've been intentionally unclear as to whether we are considering rooted or unrooted trees with the parsimony criterion, in fact it doesn't matter, as we now show.

Theorem 7. If T^ρ is a rooted version of T, then for any character χ,

$$ps_\chi(T^\rho) = ps_\chi(T).$$

Proof. If ρ is a vertex of the tree T, then by the definition of the parsimony score this is clear.

So suppose ρ has been introduced along an edge of T, by replacing that edge with two edges meeting at ρ. Observe that any minimal extension of χ to T^ρ must have the same state at ρ as at least one of the adjacent vertices.

But then we see $ps_\chi(T^\rho) \geq ps_\chi(T)$, since when any minimal extension of χ to T^ρ is restricted to the vertices of T, its state-change count remains the same. But also $ps_\chi(T^\rho) \leq ps_\chi(T)$ since given any minimal extension of χ to T, we can further extend it to T^ρ without altering its state-change count. □

Thus the parsimony score does not depend on the root location, and once we show the Fitch-Hartigan algorithm gives the parsimony score for a rooted tree, it must give the same score regardless of root location. While the details of the algorithmic steps depend on the choice of root, the final score does not.

Before we give more proofs, though, let's return to our example. Now that we've evaluated the parsimony score of the tree in Figure 3.1, let's consider another tree, in Figure 3.2, that might relate the same 1-base sequences. Applying the method above to produce the labeling at the internal vertices, we find this tree has a parsimony score of 2; only two mutations are needed. Thus the tree in Figure 3.2 is more parsimonious than that of Figure 3.1.

To find the most parsimonious tree for these taxa we would need to consider all 15 possible topologies of unrooted trees with 5 taxa and compute the minimum number of mutations for each. Rather than methodically go through the 13 remaining trees, for this simple example we can just think about what trees are likely to have low parsimony scores. If the score is low, S1 and S5 are likely to be near one another, as are S2 and S3, but S4 might be anywhere. With this observation, its easy to come up with several more trees that have parsimony score 2 but that are topologically distinct from that of Figure 3.2.

It's also easy to see that no tree will have a parsimony score of 1, since we need at least 2 mutations to have 3 different bases among the taxa. For this example there are in fact five trees that have a parsimony score of 2, and so are tied for most parsimonious.

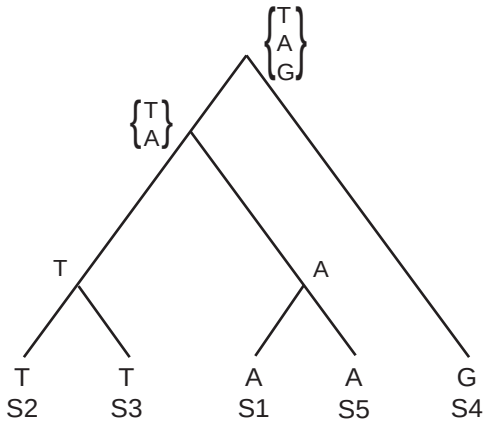

Figure 3.2: A more parsimonious tree

Here's a more succinct presentation of the Fitch-Hartigan algorithm for computing the parsimony score $ps_\chi(T)$ of a binary tree T, for a single character χ on X with state space S.

Algorithm.

1. Arbitrarily introduce a root for T, to get a rooted binary tree T^ρ.
2. Assign to each vertex $v \in V(T)$ a pair (U, m) where $U \subseteq S$ and $m \in \mathbb{Z}^{\geq 0}$ as follows:
   (a) To each leaf $x \in X$, assign the pair $(\{\chi(x)\}, 0)$.
   (b) If the two children of v have been assigned pairs (U_1, m_1) and (U_2, m_2), then assign to v the pair
   $$(U, m) = \begin{cases} (U_1 \cup U_2,\ m_1 + m_2 + 1) & \text{if } U_1 \cap U_2 = \emptyset, \\ (U_1 \cap U_2,\ m_1 + m_2) & \text{otherwise.} \end{cases}$$
   Repeat until all vertices have been assigned pairs.
3. If the pair (U, m) has been assigned to ρ, then $ps_\chi(T) = m$.

When dealing with real data, we of course need to count the number of state changes required for a tree among *all* characters. Since the score for a sequence of characters is the sum of scores for each character, this can be done in the same manner as above, just treating each character in parallel. An example is in Figure 3.3.

Proceeding up the tree beginning with the two taxa sequences ATC and ACC on the far left, we see we don't need mutations in either the first or third site, but do in the second. Thus the mutation count is now 1, and the ancestor

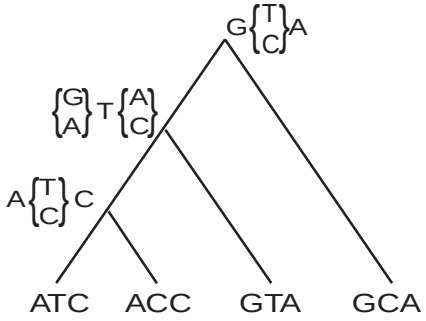

Figure 3.3: Computing a parsimony score for a tree at three sites

vertex is labeled as shown. At the vertex where the edge from the third taxa joins, we find the first site needs a mutation, the second does not, and the third does. This increases the mutation count by 2 to give us 3 so far. Finally, at the root, we discover we need a mutation only in the second site, for a final parsimony score of 4.

While this is not hard to do by hand with only a few sites, as more sites are considered it quickly becomes a job best done by a computer.

We now prove the Fitch-Hartigan algorithm actually produces the correct parsimony score for a rooted tree. For simplicity, we consider only binary trees. The key idea is to prove something slightly stronger, as expressed in the last sentence of the following.

Theorem 8. Let χ be a character on X, and T^ρ a rooted binary X-tree. Then the Fitch-Hartigan algorithm computes $ps_\chi(T)$. Furthermore, the set of states it assigns to the root ρ is exactly the set of states that occur at ρ in the minimal extensions $\tilde{\chi}$ of χ.

Proof. We proceed by induction on $|X|$. The $|X| = 2$ case should be clear.

For general $|X|$, let $v_1, v_2 \in V(T^\rho)$ be the children of ρ. Let T_i be the subtree, rooted at v_i, of descendants of v_i. Let X_i denote the labels on the leaves of T_i, so that $X = X_1 \cup X_2$ is a disjoint union. Let $\chi_i = \chi|_{X_i}$. By the inductive hypothesis, the Fitch-Hartigan algorithm assigns to v_i the pair (U_i, m_i) where U_i is exactly the set of states that occur at v_i in minimal extensions of χ_i on T_i, and m_i is $ps_{\chi_i}(T_i)$.

Suppose $\tilde{\chi}$ is a minimal extension of χ to T, and let $\tilde{\chi}_i = \tilde{\chi}|_{V(T_i)}$. Then (see Exercise 7a) minimality ensures one of the following must hold: (1) $\tilde{\chi}(\rho) = \tilde{\chi}(v_1) = \tilde{\chi}(v_2)$, (2) $\tilde{\chi}(\rho) = \tilde{\chi}(v_1) \neq \tilde{\chi}(v_2)$, or (3) $\tilde{\chi}(\rho) = \tilde{\chi}(v_2) \neq \tilde{\chi}(v_1)$.

Consider first case (2). Then $\tilde{\chi}_2$ must be a minimal extension of χ_2, for otherwise we could contradict the minimality of $\tilde{\chi}$ by defining a character on T that agreed with $\tilde{\chi}$ on $V(T) - V(T_2)$ and agreed with a minimal extension of χ_2 to T_2 on $V(T_2)$.

We similarly see that $\tilde{\chi}_1$ must be a minimal extension of χ_1 by applying the same argument to the minimal extension $\tilde{\tilde{\chi}}$ of χ defined by

$$\tilde{\tilde{\chi}}(v) = \begin{cases} \tilde{\chi}(v) & \text{if } v \neq \rho, \\ \tilde{\chi}(v_2) & \text{if } v = \rho. \end{cases}$$

Thus in this case the $\tilde{\chi}_i$ are both minimal extensions of the χ_i, while $\tilde{\chi}(v_1) \neq \tilde{\chi}(v_2)$. This shows $ps_\chi(T) = m_1 + m_2 + 1$. It also shows there do not exist any minimal extensions of χ_i on the T_i which agree on v_1 and v_2, since the existence of such would allow us to construct an extension of χ on T with a state-change count of $m_1 + m_2$, contradicting the minimality of $\tilde{\chi}$. Thus the U_i are disjoint, and $\tilde{\chi}(\rho) \in U_1 \cup U_2$. Finally, we note that for each choice of an element of $U_1 \cup U_2$ we can use minimal extensions of the χ_i on T_i to define a minimal extension of χ taking on the specified choice of state at ρ. This completes case (2). Case (3) is essentially the same.

For case (1), we cannot conclude that both $\tilde{\chi}_i$ are minimal extensions of the χ_i, but only that at least one must be (see Exercise 7b). Assume then that $\tilde{\chi}_1$ is minimal. We consider two subcases, according to whether $\tilde{\chi}_2$ is (a) minimal, or (b) not minimal.

In subcase (1a), we have that both $\tilde{\chi}_i$ are minimal, and $\tilde{\chi}(\rho) = \tilde{\chi}(v_1) = \tilde{\chi}(v_2)$. Thus $ps_\chi(T) = m_1 + m_2$, the U_i are not disjoint, and $\tilde{\chi}(\rho) \in U_1 \cap U_2$. Finally to produce any minimal extension of χ to T, we can and must 'piece together' two minimal extensions of the χ_i that have a common state on v_1 and v_2, defining the state at ρ to be the same.

In subcase (1b), pick any minimal extension $\tilde{\tilde{\chi}}_2$ of χ_2. Define a new character $\tilde{\tilde{\chi}}$ on $V(T)$ to agree with $\tilde{\chi}$ on $V(T) - V(T_2)$, and agree with $\tilde{\tilde{\chi}}_2$ on $V(T_2)$. Then since the state-change count for $\tilde{\tilde{\chi}}$ cannot be lower than that of $\tilde{\chi}$, $\tilde{\tilde{\chi}}$ must also be a minimal extension of χ, and $\tilde{\tilde{\chi}}(v_2) \neq \tilde{\chi}(v_2)$. This shows that $ps_\chi(T) = m_1 + m_2 + 1$, and that $\tilde{\chi}(\rho) \in U_1 \cup U_2$. From $ps_\chi(T) = m_1 + m_2 + 1$, it is straightforward to see that the U_i must be disjoint. Finally, that a minimal extension of χ exists with any element in $U_1 \cup U_2$ as its value at ρ is as in case (2). □

3.3 Informative Characters

We can save some computational effort in finding parsimonious trees if we observe that some characters do not affect the parsimony scores of trees in ways that affect our judgment of which is optimal.

The obvious case is a character χ that assigns the same state to all taxa, i.e., a constant character. This character extends to a constant character on $V(T)$ for any T, and hence $ps_\chi(T) = 0$ for all T. Thus we can simply discard the character from a sequence of characters, and not change parsimony scores.

A less obvious case is a character χ that assigns the same state to all taxa but a few, assigning different states to each of these few taxa. More precisely, suppose $|\chi^{-1}(i)| > 1$ for at most one state i. In this case, let j denote a state

with $|\chi^{-1}(j)|$ maximal, and $l = |\chi(X)|$. Then for any T we can extend χ to T by assigning to each internal vertex the state j. This shows $ps_\chi(T) \leq l - 1$. Since $ps_\chi(T) \geq l - 1$ for any χ, T (see Exercise 8), this means $ps_\chi(T) = l - 1$ for all trees T. While the appearance of such a χ among a sequence of characters *does* affect the parsimony score of all trees, it simply inflates them all by the same amount, and so *does not* affect the selection of the most parsimonious trees.

This leads to the idea of an *informative character*.

Definition. A character on X is *informative* if it assigns to the taxa at least two different states at least twice each. That is, χ is informative if, and only if,

$$\left|\left\{i \in S \ : \ \left|\chi^{-1}(i)\right| > 1\right\}\right| > 1.$$

Before applying the Fitch-Hartigan parsimony algorithm, we can eliminate all non-informative characters from our data since they will not affect the choice of most parsimonious trees. For DNA sequence data, many characters are likely to be non-informative, since identical, or nearly-identical sites are needed to identify and align orthologous sequences. In the examples above, you'll notice only informative sites have been used.

3.4 Complexity

Suppose we have M informative s-state characters for the taxa in X. Then solving the small parsimony problem for a fixed bifurcating X-tree T by the Fitch-Hartigan algorithm requires we perform at most a fixed amount of work with sets of size at most s at each of the $|X| - 2$ internal vertices for each of the M characters. Therefore the algorithm can be performed in time $\mathcal{O}(sM|X|)$.

Unfortunately, this is only for one tree T, and there are $(2|X| - 5)!!$ trees to consider, and searching through them all is horribly slow. A natural question asks if this slowness is unavoidable.

The answer to this last question appears to be 'yes.' More technically, finding a most parsimonious tree has been shown to be NP-hard, and so we do not expect fast algorithms that guarantee we have found all (or even one) optimal trees when $|X|$ is large. In practice, there are heuristic search methods that seem to work well. Still, only when the number of taxa is small can we be sure we've found the most parsimonious trees. While the methods of exploring 'tree-space' in a heuristic search are interesting, we will not discuss them here.

3.5 Weighted Parsimony

A natural generalization of the basic idea of parsimony penalizes different state changes in different ways as we choose the most parsimonious trees. For instance, in DNA sequence data it is often clear that transitions and transversions occur at different rates. If transversions are more rare, we might want to give them a higher weight in calculating parsimony scores, reasoning that rarer

events might carry a greater phylogenetic signal. We now develop an algorithm for computing weighted parsimony scores.

The Fitch-Hartigan algorithm for computing unweighted parsimony scores is an example of a *dynamic programming* algorithm, a rather common algorithmic approach to solving problems. Roughly this means the algorithm proceeds by finding optimal solutions to smaller instances of the problem (on subtrees) in order to use those solutions to get an optimal solution to the problem we care about.

The dynamic programming approach can also be used to compute weighted scores, though we must keep track of more things as we work our way through the tree.

Suppose χ is an s-state character on X, and fix an $s \times s$ matrix $W = (w_{ij})$ of weights. Here w_{ij} is the cost we wish to impose for a state change from state i to state j in the descent from the root. While it is of course natural to assume $w_{ii} = 0$ and $w_{ij} \geq 0$ for $i \neq j$, we do not have to.

For a rooted X-tree T^ρ, the weighted parsimony score is defined as follows:

Definition. For any extension $\tilde{\chi}$ of χ on T^ρ, the *cost* of $\tilde{\chi}$ is

$$c_W(\tilde{\chi}) = \sum_{e \in E(T)} w_{\tilde{\chi}(t_e)\tilde{\chi}(h_e)}.$$

Here $t_e, h_e \in V(T)$ denote the tail and head vertices of the edge e directed away from the root.

Definition. The *weighted parsimony score* of T^ρ is

$$ps_{\chi,W}(T^\rho) = \min_{\tilde{\chi} \in Ext_{T^\rho}(\chi)} c_W(\tilde{\chi}).$$

Rather then begin with an example, we'll formally state an algorithm for computing weighted parsimony scores first. Again, we only treat binary trees, since generalizing to arbitrary trees is straightforward. Suppose X, T^ρ, and χ are given.

Algorithm.

1. Assign to each leaf $x \in X$ of T^ρ an s-element vector $\mathbf{c}$, where

$$c_i = \begin{cases} 0 & \text{if } \chi(x) = i, \\ \infty & \text{otherwise .} \end{cases}$$

2. If the two children of $v \in V(T)$ have been assigned vectors $\mathbf{a}$ and $\mathbf{b}$, then assign to v the vector $\mathbf{c}$ with

$$c_i = \min_{j \in S}(w_{ij} + a_j) + \min_{k \in S}(w_{ik} + b_k).$$

   Repeat until all vertices have been assigned vectors.

3. If the vector $\mathbf{c}$ has been assigned to ρ, output $m = \min_i(c_i)$.

Figure 3.4 shows an example of the algorithm, where transitions have been given weight 1, transversions weight 2, and no substitution weight 0. The final score for the tree is thus found to be 3. (Note that this character would be non-informative for unweighted parsimony. Is it informative with this weighting scheme?)

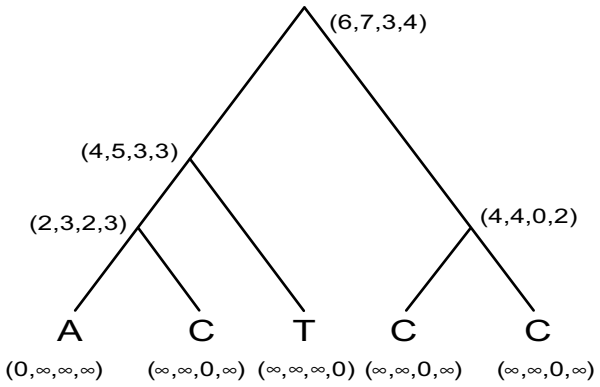

Figure 3.4: Computing weighted parsimony scores, with transitions weighted 1 and transversions weighted 2. Base order is A, G, C, T for scoring vectors.

We'll omit a formal proof that this algorithm gives the correct weighted parsimony score, since it is much easier to see this than for the Fitch-Hartigan algorithm. In effect, we're considering all possible extensions of χ (without introducing new states) that achieve optimal scores on subtrees, for each possible state at the root of the subtree. Keep in mind, though, that while the correctness of this algorithm is easier to understand, it is also more work to perform it than to perform the Fitch-Hartigan one.

Note also that allowing weights to be assigned to state changes may have made the parsimony score of a tree dependent on the root location. While this may be desirable for some purposes, it means we will have to search through all rooted trees rather than just unrooted ones. If we restrict the allowable choices of weights for state changes, however, we can preserve the independence of the parsimony score from the root location. We leave as Exercise 17b determining how this is done.

3.6 Recovering Minimal Extensions

In addition to computing the parsimony score of a tree, we may actually want to know what the minimal extensions of χ are that achieve this minimal cost. One reason for this is that thinking of minimal extensions as likely evolutionary

histories, we might want to know for each such history which state changes occurred on which edges. Across many characters, this would allow us to identify on which edges we had many state changes and which had few, thus indicating something about either the length of time those edges might represent, or at least whether circumstances were highly favorable to mutations during those periods. Of course the various minimal extensions may indicate state changes along different edges, so at best we will be able to assign a range of the number of changes (or the total cost for weighted parsimony) along each edge.

We'll indicate how to find all minimal extensions from the weighted parsimony algorithm, though there is also a variation for the unweighted case that builds on the Fitch-Hartigan algorithm.

Suppose then that we have computed the weighted parsimony score of a character χ on a tree T^ρ by the algorithm above. If $\mathbf{c}$ is the vector assigned to ρ, then for each i with c_i minimal, there will be a minimal extension of χ taking on state i at ρ.

Considering only one such minimal c_i, let v_1 and v_2 be the children of ρ, with assigned vectors $\mathbf{a}, \mathbf{b}$. Then for each choice of state j at v_1 and state k at v_2 with

$$c_i = w_{ij} + a_j + w_{ik} + b_k,$$

there will be a minimal extension of χ taking on the states i, j, k at ρ, v_1, v_2, respectively. We simply continue down the tree in this way to eventually produce all minimal extensions.

3.7 Other Issues

There are a number of issues with parsimony that we won't discuss in detail, but that we leave for you to think about:

1. If several trees tie for most parsimonious (as they often do), what should we do? There are various approaches to defining *consensus trees* that attempt to summarize the common features of several trees. We will touch on this in Chapter 4

2. What about trees that are 'close' to most parsimonious? Are we really committed enough to the parsimony principal to feel that being only a few state changes off from most parsimonious means a tree should not be considered as biologically reasonable?

3. With more than a handful of taxa, there are many possible binary trees to consider that might relate them. If there are too many to search through to be sure of finding the most parsimonious, what heuristic search methods are likely to do well in practice?

4. If weighted parsimony is to be used, how should weights be assigned? Since assigning weights can affect which trees we view as 'best,' how can we objectively choose?

5. A final issue, which has been the subject of much controversy among biologists, concerns the philosophical underpinning of parsimony. There are strong proponents of parsimony who believe it is the *only* valid way of inferring phylogenies. There are others with a wider view, who may prefer other methods but admit parsimony is a reasonable approach under some circumstances, but not others. In Chapter 9 we will see that some reasonable probabilistic models of mutation processes can, under some circumstances, lead to data that will cause parsimony to infer an incorrect tree.

   We will not explore this issue fully now, but will give a hint as to part of the problem. If, say, we are discussing DNA mutation, then along an edge of a tree that represents a long period of time a single site may mutate several times, for instance $A \to C \to G$, or even $A \to C \to A$. Thus while two state changes occurred, either one or none is observed if we only consider the ends of the edge. The parsimony principle, however, seems to reject the possibility of multiple changes as a preferred explanation.

   As long as state changes are rare on all edges of a tree, parsimony is a quite reasonable approach to inferring a phylogeny. It is when state changes are less rare that potential problems arise. For characters describing morphology, or other larger-scale observations, we expect few, if any, hidden mutations. For sequence data, the situation is less clear and may well depend on the data set being examined.

3.8 Exercises

1. a. Compute the minimum number of base changes needed for the trees in Figure 3.5.

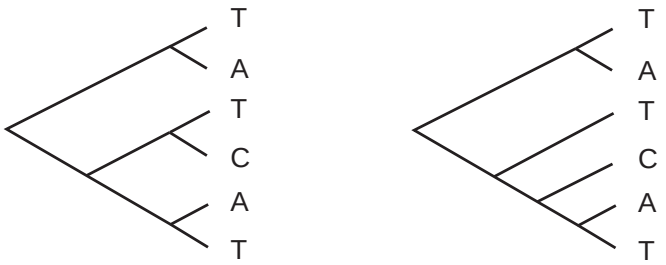

Figure 3.5: Trees for Problem 1

   b. Give at least three trees that tie for most parsimonious for the one-base sequences used in part (a). (Remember, you can list the taxa in a different order.)

   c. For trees tracing evolution at only one site as in (a) and (b), why can

we always find a tree requiring no more than 3 substitutions no matter how many taxa are present?

2. a. Find the parsimony score of the trees in Figure 3.6. (Only informative sites in the DNA sequences are shown.)

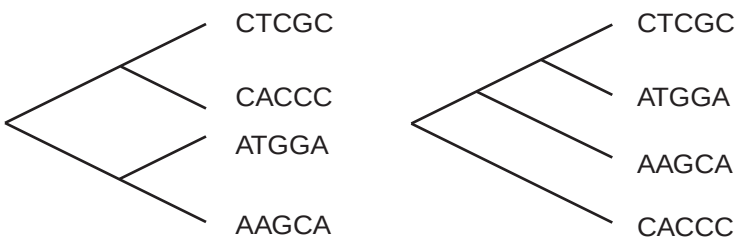

Figure 3.6: Trees for Problem 2

b. Draw the third possible (unrooted) topological tree relating these sequences and find its parsimony score. Which of the three trees is most parsimonious?

3. Repeat the last problem, but use weighted parsimony with 1:2 weights for transitions:transversions.

4. Consider the following sequences from four taxa.

S1: `AATCGCTGCTCGACC`
S2: `AAATGCTACTGGACC`
S3: `AAACGTTACTGGAGC`
S4: `AATCGTGGCTCGATC`

a. Which sites are informative?

b. Use the informative sites to determine the most parsimonious unrooted tree relating the sequences.

c. If S4 is known to be an outgroup, use your answer to (b) to give a rooted tree relating S1, S2 and S3.

5. Though non-informative sites in DNA do not affect which tree is judged to be the most parsimonious, they do affect the parsimony score. Explain why, if P_{all} and P_{info} are the parsimony scores for a tree using all sites and just informative sites, then

$$P_{\text{all}} = P_{\text{info}} + n_1 + 2n_2 + 3n_3,$$

where, for $i = 1, 2, 3$, by n_i we denote the number of sites with all taxa in agreement except for i taxa which are all different . (Notice that while P_{all} and P_{info} may be different for different topologies, $n_1 + 2n_2 + 3n_3$ does not depend on the topology.)

6. Show that if χ is an informative character on X, then there exist two phylogenetic X-trees T_1, T_2 with $ps_\chi(T_1) \neq ps_\chi(T_2)$.

7. Complete the proof of Theorem 8 by doing the following:

   (a) Suppose $\tilde{\chi}$ is a minimal extension of χ to T, and let $\tilde{\chi}_i = \tilde{\chi}|_{V(T_i)}$, with the T_i as defined in the proof. Explain why minimality ensures one of the following must hold: (1) $\tilde{\chi}(\rho) = \tilde{\chi}(v_1) = \tilde{\chi}(v_2)$, (2) $\tilde{\chi}(\rho) = \tilde{\chi}(v_1) \neq \tilde{\chi}(v_2)$, or (3) $\tilde{\chi}(\rho) = \tilde{\chi}(v_2) \neq \tilde{\chi}(v_1)$.

   (b) Explain why in case (1) at least one of the $\tilde{\chi}_i$ is a minimal extension of χ_i. Give an example to show both need not be minimal.

8. Suppose χ is any character on X, and let $l = |\chi(X)|$. Explain why the lower bound $ps_T(\chi) \geq l - 1$ holds for any phylogenetic X-tree T.

9. For the character and first tree in Figure 3.7, calculate the parsimony score, labeling the interior vertices according to the Fitch-Hartigan algorithm. Then show that the second tree requires exactly the same number of base changes, even though it is not consistent with the way you labeled the interior vertices on the first tree. (The moral of this problem is that naively interpreting the Fitch-Hartigan algorithm will not produce all minimal extensions of a character.)

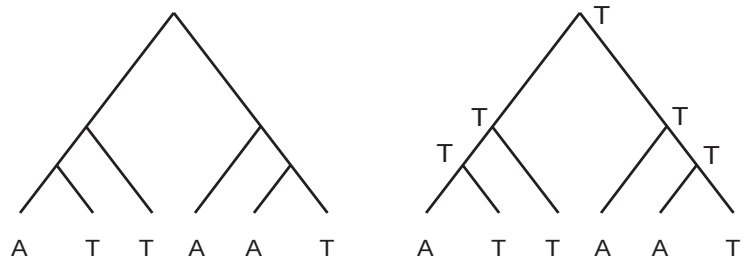

Figure 3.7: Trees for Problem 9

10. The Fitch-Hartigan algorithm proceeds from leaves to root of T^ρ to find all possible states at the root of minimal extensions of χ. Invent an algorithm that can be used afterwards, proceeding from the root to the leaves, to produce a complete list of minimal extensions of χ.

11. If characters are given for 3 terminal taxa, there can be no informative ones. Explain why this is the case, and why it doesn't matter.

12. The bases at a particular site in aligned DNA sequences from different taxa form a *pattern*. For instance, in comparing $n = 5$ sequences at a site, the pattern $(ATTGA)$ means A appears at that site in the first taxon's sequence, T in the second's, T in the third's, G in the fourth's, and A in the fifth's.

a. Explain why in comparing sequences for n taxa, there are 4^n possible patterns that might appear.

b. Some patterns are not informative, such as the 4 patterns showing the same base in all sequences. Explain why there are $(4)(3)n$ non-informative patterns that have all sequences but one in agreement.

c. How many patterns are non-informative because 2 bases each appear once, and all the others agree?

d. How many patterns are non-informative because 3 bases each appear once, and all others agree?

e. Combine your answers to calculate the number of informative patterns for n taxa. For large n, are most patterns informative?

13. A computer program that computes parsimony scores might operate as follows: First compare sequences and count the number of sites f_{pattern} for each informative pattern that appears. Then, for a given tree T, calculate the parsimony score $ps_{\text{pattern}}(T)$ for each of these patterns. Finally, use this information to compute the parsimony score for the tree using the entire sequences. What formula is needed to do this final step? In other words, give the parsimony score of the tree in terms of the f_{pattern} and $ps_{\text{pattern}}(T)$.

14. Parsimony scores can be calculated even more efficiently by using the fact that several different patterns always give the same score. For instance, in relating four taxa, the patterns $(ATTA)$ and $(CAAC)$ will have the same score.

    a. Using this observation, for 4 taxa how many different informative patterns must be considered to know the parsimony score for all?

    b. Repeat part (a) for 5 taxa.

15. Use the maximum parsimony method to construct an unrooted tree for the simulated sequences `a1`, `a2`, `a3`, and `a4` in the MATLAB data file `seqdata.mat`. First put the sequences into the rows of an array with the command `a=[a1;a2;a3;a4]`. Then find the informative sites with `infosites=informative(a)`. Finally, extract the informative sites with `ainfo=a(:,infosites)`.

    a. What percentage of the sites are informative?

    b. How many different trees must be considered to find the most parsimonious one relating the four taxa?

    c. You may find it too difficult to use all informative sites for a hand calculation. If so, use at least the first ten informative sites to pick the most parsimonious tree.

    d. Use PAUP* on the full data sequences to find the most parsimonious trees.

16. In this problem you will attempt to use the maximum parsimony method to construct an unrooted tree for the simulated sequences `d1`, `d2`, `d3`, `d4`, `d5`, and `d6` in the data file `seqdata.mat`. Begin by finding the informative sites as in the last problem.

    a. What percentage of the sites are informative?

    b. Compute the number of unrooted trees that must be examined if we really consider all possibilities.

    c. Using only the first 10 informative sites, compute parsimony scores of at least 5 candidate trees that you think are likely to be most parsimonious.

    e. How confident are you that the most parsimonious tree you found is actually the most parsimonious? What percentage of the possible trees did you compute parsimony scores for? What percentage of the informative sites did you use?

    d. Use PAUP* on the full data sequences to find the most parsimonious trees.

17. a. What weight matrix makes weighted parsimony the same as unweighted parsimony?

    b. What condition on W ensures $ps_{\chi,W}(T^{\rho_1}) = ps_{\chi,W}(T^{\rho_2})$ for any two rooted version T^{ρ_i} of the same unrooted tree T and all characters χ?

18. What changes are necessary to the weighted parsimony algorithm if we drop the assumption that trees be binary?

19. Create and develop an example of the algorithm for the small weighted parsimony problem, other than that given in Figure 3.4. Check your example with PAUP*.

20. Create an example of 4 sequences where unweighted parsimony leads to a different optimal tree than weighted parsimony with transversion weight twice that of transitions. Check your example with PAUP*.

21. Is there a useful notion of 'informative characters' for weighted parsimony? Explain.

22. Explain why the weighted parsimony algorithm for one s-state character on a set of taxa X will involve $\mathcal{O}(s^2|X|)$ steps.

23. When applying weighted parsimony to sequence data such as for DNA, it is reasonable to require the weight matrix satisfy the condition

$$w_{ij} + w_{jk} \geq w_{ik}, \text{ for all } i, j, k.$$

    Explain why, and how this relates to the possibility of multiple mutations along an edge.

Chapter 4

Combinatorics of Trees II

Having seen one method of phylogenetic inference, we return to combinatorial considerations of phylogenetic trees. The theme of this section is different ways we might express partial information about a phylogenetic tree. Our treatment of these topics is abbreviated, giving only the briefest exposure. Much more can be found in [SS03].

4.1 Splits

Suppose T is an unrooted phylogenetic X-tree. Then if we delete any one edge e of T, we partition the taxa X into two subsets, according to the two connected components of the resulting graph. For instance, deleting the marked edge in the tree T of Figure 4.1 produces the partition $\{a, b, d\}, \{c, e, f\}$, which we call the *split* induced by e.

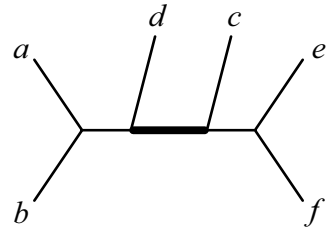

Figure 4.1: An edge inducing the split $\{a, b, d\}|\{c, e, f\}$.

Definition. A *split* of X is any partition of X into two non-empty disjoint subsets. We write $X_0|X_1$ to denote the split with subsets X_0, X_1.

Note that if $X_0|X_1$ is a split, then $X_1 = X \setminus X_0 = X_0^c$ is the complement of X_0.

Suppose now that T is a phylogenetic X-tree and we consider the splits induced by two different edges e_1, e_2 of T. Removing both e_1 and e_2 from T produces three connected components, and thus a partition of X into three disjoint subsets X_1, X_2, X_3. With appropriate numbering of these sets, the split induced by e_1 is $X_1|X_2 \cup X_3$ and that induced by e_2 is $X_1 \cup X_2|X_3$. This shows splits induced from edges of a phylogenetic X-tree will have the following pairwise property.

Definition. Two splits $Y_0|Y_1$ and $Y_0'|Y_1'$ of a set Y are said to be *compatible* if for some i, j we have $Y_i \cap Y_j' = \emptyset$.

A natural question is to what extent a collection of pairwise compatible X-splits corresponds to a phylogenetic tree. To answer this question, we need a slight generalization of a phylogenetic tree. While binary phylogenetic trees remain the biologists' ideal, sometimes the best tree we can produce has higher-valence vertices, some labels on internal vertices (which you might think of as leaves which have not yet been resolved from the interior of the tree, and multiple labels on some vertices (which you might also think of as a result of not yet resolving the branching of all taxa from one another). All that we will need is captured in the following definition.

Definition. An *X-tree* is a tree T together with a labeling map $\phi : X \to V(T)$ such that for each $v \in V(T)$ with degree ≤ 2, $\phi^{-1}(v) \neq \emptyset$.

Obviously we can refer to splits of X induced by edges of an X-tree, just as for phylogenetic trees.

We now establish the *Splits Equivalence Theorem*, by a variant of the proof of [SS03].

Theorem 9. Let S be a collection of splits of X. Then there is an X-tree whose induced splits are precisely those of S if, and only if, the splits in S are pairwise compatible. Furthermore, this X-tree is unique, up to isomorphism of partially-labeled graphs.

Proof. That an X-tree induces pairwise compatible splits has been discussed, so suppose we have a set of compatible splits S and wish to construct such an X-tree. We proceed by induction on the size of S, with the case $|S| = 1$ clear.

Let $S = S' \cup \{X_0|X_1\}$ where $X_0|X_1 \notin S'$. Then by induction, there is an X-tree T' whose induced splits are precisely those of S'. Color red all those vertices of T' that are labeled by elements of X_0, and color blue all those labeled by elements of X_1, so every vertex is either red, blue, red-blue, or uncolored.

Suppose two or more vertices are colored red-blue. Then pick any edge on the minimal path between them, and consider the split of X it induces. Such a split in S' would not be compatible with $X_0|X_1$, since each partition set contains labels in both X_0 and X_1. Thus at most one vertex is red-blue.

Our next goal is to show there is a unique vertex v in T' whose removal results in connected components all of whose colored vertices have the same color. To do this, we consider two cases.

First, suppose no vertex of T' is red-blue. Let T_1 be the minimal spanning tree of all blue vertices in T', and T_2 the minimal spanning tree of all red vertices in T'. Observe T_1 and T_2 cannot have a common edge, since if they did that edge of T' would induce a split not compatible with $X_0|X_1$. However, if T_1 and T_2 are disjoint then they are joined by some path in T', but picking any edge on that path induces the split $X_0|X_1$, and since $X_0|X_1 \notin S'$ we have a contradiction. Thus T_1 and T_2 must have exactly one vertex in common. Calling this vertex v, note that when v is removed from T in each of the resulting connected components all colored vertices have the same color. Furthermore v is the unique vertex with this property.

Second, suppose T' has a unique red-blue vertex and denote it by v. Then again we see that when v is removed from T', in each resulting connected component all colored vertices must have the same color (See Exercise 3). No other vertex has this property, since v would be in one of the resulting components.

Now in either case, we modify T' to get T by replacing v by two vertices v_0 and v_1 connected by an edge e, reconnecting the edges of T' incident to v so that red components join to v_0 and blue components to v_1. We modify the labeling map for T' to get a labeling map for T by reassigning all of v's labels from X_0 to v_0, and all of v's labels from X_1 to v_1. We now see that the new edge e of T induces the split $X_0|X_1$, while all other edges of T induce precisely the splits of S'.

That T is uniquely determined by S we leave as Exercise 4. □

Notice that this proof in fact gives an algorithm (called *tree-popping*) for constructing an X-tree from a collection of splits. To illustrate this, we show only one step which might be done in the middle of a longer process. Suppose the graph on the left of Figure 4.2 has already been constructed from some splits, each of which was compatible with the split $\{de\}|\{abcf\}$. Then we color

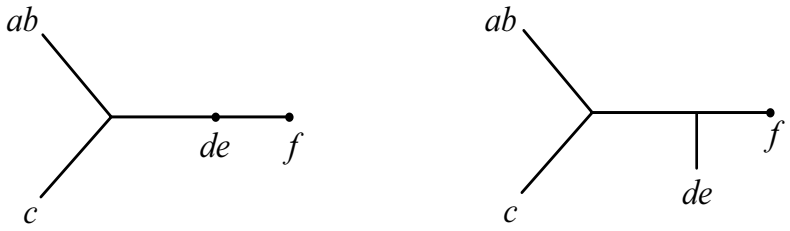

Figure 4.2: Incorporating the split $\{de\}|\{abcf\}$ into a tree following the proof of Theorem 9.

the vertex labeled de red, and the other labeled vertices blue. Since there is no red-blue vertex, to distinguish the vertex v that will be replaced, we consider the minimal spanning trees of the vertices of each color. For red, this just a single vertex, while for the blue, it is the entire tree. These intersect at the vertex labeled de. We thus remove this vertex, replacing it by an edge with all

red vertices joined at one end, and all blue at the other. This gives the tree on the right of Figure 4.2.

It is instructive to work through a more complete example, such as in Exercise 6.

4.2 Refinements and Consensus Trees

To illustrate the utility of the splits viewpoint on trees, we give two applications.

First, we can use Theorem 9 to determine when two X-trees can be 'combined' into a more refined X-tree. This is useful in a biological setting if we have several non-binary X-trees that we've inferred, perhaps from different data sets, that have managed to resolve different parts of some unknown binary tree. We want to combine the information they contain to give a tree that shows more resolution. For example, in Figure 4.3, we see two trees which have a common minimal refinement, in a sense that can be made precise through splits.

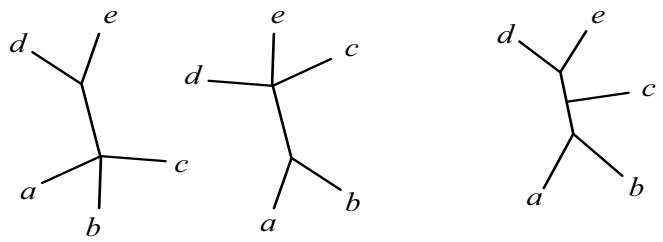

Figure 4.3: Two trees and their common refinement

More formally, we say a tree T_2 is a *refinement* of T_1 if every split induced by T_1 is also induced by T_2.

By Theorem 9 we easily see the existence of a common refinement of two trees is equivalent to the compatibility of the splits of the two trees.

Corollary 10. Suppose $T_1, T_2, \ldots, T_n$ are X-trees. Then all induced edge splits from these trees are pairwise compatible if, and only if, there exists an X-tree T whose induced splits are precisely those of the T_i. Furthermore, if this X-tree T exists, it is unique.

To apply the Splits Equivalence Theorem in another direction, we may have produced several X-trees which are highly resolved, but we want to combine them into a single *consensus tree* that only shows features common to all or most. This is frequently done when parsimony has been used to infer trees, since many trees may tie for most parsimonious. Other situations where this may be desirable include when different genes have been used to infer highly resolved trees for the same set of taxa, but the trees are not in complete agreement.

Given X-trees $T_1, T_2, \ldots, T_n$, a *strict consensus tree* is a tree inducing precisely the splits that are induced by all T_i. Note that the splits induced by all

T_i must be pairwise compatible since they are all induced by a single tree, say T_1. Thus by the Splits Equivalence Theorem, a strict consensus tree exists and is easily constructed.

Sometimes a strict consensus tree is so poorly resolved, however, that a less radical approach is taken. A *majority-rule consensus tree* for $T_1, T_2, \ldots, T_n$ is a tree inducing precisely the splits that are induced by more than half of the T_i. This set of splits must be pairwise compatible, since if two splits are each induced by more than half the trees, there must be at least one tree inducing both. Again the Splits Equivalence Theorem can be applied. In summary, we have shown the following.

Corollary 11. For any collection of X-trees, there is a unique strict consensus tree, and a unique majority-rule consensus tree.

Finally, note that one could consider consensus trees between the strict and majority rule levels, by choosing a parameter q between $1/2$ and 1. Taking all the splits induced by more than qn of the n trees, we again obtain a pairwise compatible set, and thus determine a unique tree inducing precisely those splits.

4.3 Quartets

A possible approach to determining a phylogenetic tree for a large number of taxa is to try to determine trees for smaller subsets of taxa, and then somehow piece these together. For this, we'll use unrooted trees, since the trees relating these smaller sets might not all contain a common vertex.

If we focus on small subsets, then *quartets* of 4 taxa are the smallest ones we should consider, as an unrooted tree relating 3 taxa gives us no information on a tree topology.

Definition. A *quartet tree* is a unrooted binary tree with 4 labeled leaves. We denote the tree by $ab|cd$ if it induces the split $\{a, b\}|\{c, d\}$.

Now any phylogenetic X-tree T induces a collection $\mathcal{Q}(T)$ of quartet trees:

$$\mathcal{Q}(T) = \{ab|cd \ : \text{ for some } X_0|X_1 \text{ induced by } T,\ a, b \in X_0 \text{ and } c, d \in X_1\}.$$

If T is binary, then for every 4-element subset $\{a, b, c, d\}$ of X, one of the quartets $ab|cd$, $ac|bd$, or $ad|bc$ is in $\mathcal{Q}(T)$, so $\mathcal{Q}(T)$ has $\binom{|X|}{4}$ elements. For non-binary $|X|$, of course, it has fewer.

If using quartets to deduce phylogenies of larger collections of taxa is to have a chance of working, we must have the following.

Theorem 12. The collection $\mathcal{Q}(T)$ determines T for a binary tree.

Proof. If $|X| \leq 4$ the result is clear, and so we proceed by induction.

If $X = \{a_1, \ldots, a_n\}$, let $\mathcal{Q}'$ be those quartets in $\mathcal{Q}(T)$ that only involve taxa in $X' = \{a_1, a_2, \ldots, a_{n-1}\}$. Then $\mathcal{Q}' = \mathcal{Q}(T')$ for a tree T' obtained from T in the obvious way, by deleting a_n and its incident edge, and conjoining the two

other edges that edge meets, to get a binary tree. Thus $\mathcal{Q}'$ determines T' by the induction hypothesis.

To determine T, we need only determine the conjoined edge of T' to which the edge leading to a_n should attach. So considering each edge e of T' in turn, suppose e induces the split $X'_0|X'_1$ of X'. Assuming each of the sets in the split has at least two elements, then if for all $a, b \in X'_0$ and $c, d \in X'_1$ both $ab|ca_n$ and $aa_n|cd$ are in $\mathcal{Q}(T)$, e is the edge we seek. If one of the sets in the split has only one element, e can be determined by a similar test, the formulation of which we leave as Exercise 9. □

This theorem can be extended to non-binary trees as well.

As you'll see in the exercises, a subset of $\mathcal{Q}(T)$ may be enough to identify T. In fact only $|X| - 3$ quartets are needed, though not every collection of quartets of that size is sufficient.

Of course the real difficulty with determining a phylogenetic tree from quartets is usually not the issue of having insufficiently many quartets. Rather, whatever inference method is used to infer quartets is likely to give some wrong results, and so produce a set of quartets that are incompatible. What is needed is a construction of a tree that somehow reconciles as much information as possible from inconsistent sources. Such *quartet methods* for tree construction have been explored in a number of research papers in recent years.

4.4 Supertrees

Assembling a tree from quartets is an example of a more general problem of constructing a *supertree*. We might have obtained a collection of different trees, with overlapping but different sets of taxa, which we wish to combine into a single tree. Doing this might enable us to see relationships between species which do not appear together on any of the individual trees.

There are a number of different approaches, a few of which are mentioned in [SS03] along with more pointers to the literature. Here we describe only one approach which is relatively popular among biologists, called Matrix Representation with Parsimony (MRP).

First, we need to extend the idea of parsimony to encompass characters with missing state information. More formally, given any character $\chi_Y : Y \to S_Y$ defined on a subset $Y \subset X$, we can extend it to a character on X by picking any $m \notin S_Y$ where m will signify 'missing', letting $S = S_Y \cup \{m\}$, and defining

$$\chi(x) = \begin{cases} \chi_Y(x) & \text{if } x \in Y \\ m & \text{if } x \notin Y \end{cases}.$$

We then can modify the notion of parsimony score, and the Fitch-Hartigan algorithm for computing it (Exercise 13) so that state changes from m to any other state are not counted.

Now given a collection $\{T_i\}$ where T_i is a phylogenetic X_i-tree, let $X = \cup X_i$. For each split induced by each T_i, define a two-state character on X_i by assigning states in accord with the partition sets. Then consider the sequence of all these characters, viewed as 3-state characters on X encoding missing state information. The MRP supertree is then the maximum parsimony tree for this character sequence.

Of course as with any use of parsimony, such a tree may not be unique. If there are multiple trees that are most parsimonious, it is common to take a consensus tree of those.

The name 'Matrix Representation with Parsimony' may seem strange to a mathematician. The terminology comes from the use of the word 'matrix' to describe character information that has been placed in a table, with rows corresponding to taxa, columns to characters, and entries specifying states. In MRP, the trees we wish to combine are 'represented' by a matrix encoding characters corresponding to their splits.

4.5 Exercises

1. Show that two splits $Y_0|Y_1$ and $Y_0'|Y_1'$ of a set Y are compatible if and only if $Y_i \subseteq Y_j'$ for some i, j. Further show that $Y_i \subseteq Y_j'$ if and only if $Y_{1-i} \supseteq Y_{1-j}'$.

2. Show that if splits $Y_0|Y_1$ and $Y_0'|Y_1'$ of a set Y are compatible then there is exactly one pair i, j with $Y_i \cap Y_j' = \emptyset$.

3. Complete the proof of Theorem 9, the Splits Equivalence Theorem, by showing that if v is the unique red-blue vertex, then each connected component that results from its removal from T' has all colored vertices of the same color. (Hint: Suppose not, and produce an edge of T' that induces a split incompatible with $X_0|X_1$.)

4. Show the tree T whose existence was established in our proof of the splits equivalence theorem, Theorem 9, is in fact unique.

5. Elaborate on the given proof of Theorem 9 to explain why the T we construct has the X-tree property that all vertices of degree ≤ 2 are labeled.

6. For $X = \{a, b, c, d, e, f, g\}$ consider the pairwise compatible splits

$$a|bcdefg, eg|abcdf, b|acdefg, af|bcdeg, f|abcdeg.$$

   By tree-popping, find an X-tree inducing precisely these splits. Then use tree-popping again, but with the splits in some other order, to again find the same tree.

7. For some set X, draw an interesting X-tree, lists its induced splits, and then use tree-popping to recreate X from the splits.

8. If T_1, T_2 are X-trees with T_2 a refinement of T_1, write $T_1 \leq T_2$. Show that '$\leq$' is a partial order on the set of X-trees. Then characterize X-trees that are maximal with respect to '$\leq$'.

9. Complete the proof of Theorem 12, by addressing the issue in its final line.

10. For a 5-leaf binary phylogenetic X-tree T, $\mathcal{Q}(T)$ has 5 elements. Give a set of only two of these quartets that still determine T. Give another set of two of these quartets that does not determine T.

11. Generalize the result of the last problem by showing that for any binary phylogenetic X-tree, there are $|X| - 3$ quartets that determine T.

12. Give another proof of Theorem 12, by using quartets to determine splits and then applying Theorem 9.

13. Define the notion of (unweighted) parsimony score of a character on a binary tree in a setting where state m denotes 'missing' state information. Then give a modified Fitch-Hartigan algorithm to compute it.

Chapter 5

Distance Algorithms

The next class of methods for constructing phylogenetic trees that we will discuss are *distance methods*. These attempt to build a metric tree from quantitative measures of sequence similarity. We'll focus in this section on algorithmic approaches, but include several problems on other distance methods.

Definition. A *dissimilarity* map for the set X of taxa is a function $\delta : X \times X \to \mathbb{R}$ such that $\delta(x, x) = 0$ and $\delta(x, y) = \delta(y, x)$.

The simplest such natural map is essentially the *Hamming metric* on finite sequences: If we are given a sequence of characters $\chi_1, \chi_2, \ldots, \chi_n$ on X, set

$$\delta(x, y) = \frac{1}{n} \sum_{i=1}^{n} \delta_{\chi_i(x), \chi_i(y)},$$

where

$$\delta_{a,b} = \begin{cases} 0 & \text{if } a = b \\ 1 & \text{if } a \neq b \end{cases}.$$

For instance, given sequences such as

$$x : \texttt{AACTAGATCCTGTATCGA}$$
$$y : \texttt{ACCTAGGTCCTGTTTCGC}$$

we count 4 differences among 18 sites, so $\delta(x, y) = 4/18$.

Of course one can easily imagine variations on this, that might assign different weightings to transitions or transversions. We won't pursue such possibilities, though, since we'll eventually construct much more sophisticated dissimilarity maps using a mathematical model of molecular evolution. We introduce this particular dissimilarity map here only to provide a concrete example to keep in mind as we explore the concept.

We'd like to think of dissimilarity values as representing distances between taxa along some metric tree, as discussed in Section 2.3. That is, we hope there

is some metric tree (T, w), with positive edge lengths and tree metric d, so that the dissimilarity map δ is just the restriction of the tree metric d to $X \times X$. Of course there's nothing in our definition of a dissimilarity map to ensure this, or even to suggest that the dissimilarity values are even close to such distances measured along a metric tree. However, we will simply hope for the best, and forge ahead.

Suppose, then that we have 4 taxa, and using some dissimilarity map we've computed a table of dissimilarities such as:

	S1	S2	S3	S4
S1		.45	.27	.53
S2			.40	.50
S3				.62

Table 5.1: Dissimilarities between taxa

How might we find a metric tree (T, w) for which this data is at least approximately the same as distances computed by the tree metric? More informally, if we know how far apart we want the leaves of a tree to be, how can we come up with a tree topology and edge lengths to roughly ensure that?

5.1 UPGMA

Proceeding naively, it's not too hard to make up some kind of an algorithm to produce a tree from dissimilarities. When pressed to do so, most students come up with some variant of an approach that is called the *average distance method*, or, more formally, the *unweighted pair-group method with arithmetic means (UPGMA)*. Rather than present the algorithm formally, we'll develop it through the example data in Table 5.1 above.

In going through this example we're thinking of dissimilarities as approximations of distances that in fact come from a tree metric. In order to make our language simpler, most of the time we'll simply refer to 'distances' for both values of the dissimilarity map and values of the metric on the tree we construct.

The first natural step is to assume that the two taxa which are shown as closest by the dissimilarity map are probably closest in the tree. With the data table above, we pick the two closest taxa, S1 and S3, and join them to a common ancestral vertex by edges. Drawing Figure 5.1, since $S1$ and $S3$ are .27 apart, we decide to split this, making each edge $.27/2 = .135$ long.

Since S1 and S3 have been joined, we now collapse them into a single combined group S1-S3. To say how far this group is from another taxon, we simply average the distances from S1 and S3 to that taxon. For example, the distance between S1-S3 and S2 is $(.45 + .40)/2 = .425$, and the distance between S1-S3 and S4 is $(.53 + .62)/2 = .575$. Our dissimilarity table thus collapses to Table 5.2.

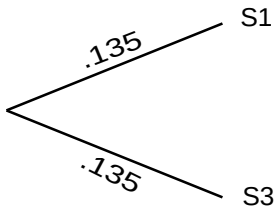

Figure 5.1: UPGMA, step 1

	S1-S3	S2	S4
S1-S3		.425	.575
S2			.50

Table 5.2: Distances between groups; UPGMA, step 1

Now we simply repeat the process, using the distances in the collapsed table. Since the closest taxa and/or groups in the new table are S1-S3 and S2, which are .425 apart, we draw Figure 5.2.

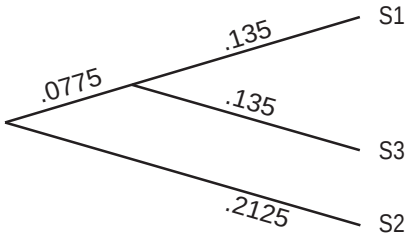

Figure 5.2: UPGMA; step 2

Note the edge to S2 must have length $.425/2 = .2125$. However the other new edge must have length $(.425/2) - .135 = .0775$, since we already have the edges of length .135 to account for some of the distance between S2 and the other taxa.

Again combining taxa, we form a group S1-S2-S3, and compute its distance from S4 by averaging *the original distances from S4 to each of S1, S2, and S3*. This gives us $(.53+.5+.62)/3 = .55$. (Note that this is *not* the same as averaging the distance from S4 to S1-S3 and to S2.) Since a new collapsed distance table would have this as its only entry, there's no need to give it. We draw Figure 5.3, estimating that S4 is $.55/2 = .275$ from the root. The final edge has length .0625, since that places the other taxa .275 from the root as well.

As we suspected, the tree we've constructed for the data does not exactly fit the data. The distance on the tree from S3 to S4, for instance, is .55, while according to the original data it should be .62. Nonetheless, the tree distances

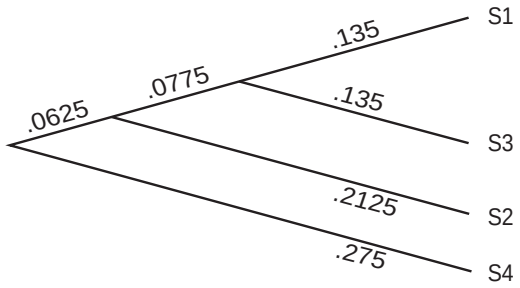

Figure 5.3: UPGMA; step 3

are at least reasonably close to the distances given by the data.

If we had more taxa to relate, we'd have to do more steps to complete the UPGMA process, but there would be no new ideas involved. At each step, we join the two closest taxa or groups together, always placing them at equal distances from a common ancestor. We then collapse the joined taxa into a group, using averaging to compute a distance from that group to the taxa and groups still to be joined. The one point to be particularly careful about is that when the distances between two groups are computed, we must average *all* the original distances from members of one group to another — if one group has n members and another has m members, we have to average nm distances. Each step of the algorithm reduces the size of the distance table by one group/taxon, so that after enough steps, all of the taxa are joined into a single tree.

A few comments on the algorithm are in order here.

First, notice how little work UPGMA requires to give us a tree — at least in comparison to using a parsimony approach where we search through all possible trees, performing an algorithm on each. UPGMA requires no searching whatsoever. We simply perform the algorithmic steps and out pops a single tree. This can be viewed as a strength, as the algorithm is very fast (of polynomial time), and we get a single answer. But it also can be viewed as a weakness, since it is unclear why we consider the output tree 'good.' We have not explicitly formulated a criterion for judging trees and then found the tree that is optimal.

Second, notice that the molecular clock assumption is implicit in UPGMA. In this example, when we placed S1 and S3 at the ends of equal length branches, we assumed that the amount of mutation each underwent from their common ancestor was equal. UPGMA always places all the taxa at the same distance from the root, so that the amount of mutation from the root to any taxon is identical.

5.2 Unequal Branch Lengths

It is not always desirable to impose a molecular clock hypothesis, as use of UPGMA requires. One way of dealing with this arose in a suggested algorithm

of Fitch and Margoliash. It builds on the basic approach of UPGMA, but attempts to drop the molecular clock assumption by an additional step.

Before giving the algorithm, we make a few mathematical observations. First, if we attempt to put 3 taxa on an unrooted tree, then there is only one topology that needs to be considered. Furthermore, for 3 taxa we can assign lengths to the edges to *exactly* fit any given dissimilarities. To see this consider

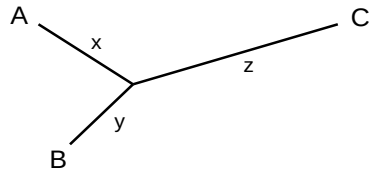

Figure 5.4: The unrooted 3-taxon tree

the tree in Figure 5.4. If we have some dissimilarity data δ_{AB}, δ_{AC}, and δ_{BC}, then

$$\begin{array}{rcrcrcl} x & + & y & & & = & \delta_{AB}, \\ x & + & & & z & = & \delta_{AC}, \\ & & y & + & z & = & \delta_{BC}. \end{array} \tag{5.1}$$

Solving these equations leads to

$$\begin{aligned} x &= (\delta_{AB} + \delta_{AC} - \delta_{BC})/2, \\ y &= (\delta_{AB} + \delta_{BC} - \delta_{AC})/2, \\ z &= (\delta_{AC} + \delta_{BC} - \delta_{AB})/2. \end{aligned} \tag{5.2}$$

We'll refer to the formulas in equations (5.2) as the *3-point formulas* for fitting taxa to a tree. Unfortunately, with more than 3 taxa, exactly fitting dissimilarities to a tree is usually not possible (see Exercise 8). The Fitch-Margoliash algorithm uses the 3 taxa case, however, to handle more taxa.

Now we explain the operation of the algorithm with an example. We'll use the dissimilarity data in Table 5.3.

	S1	S2	S3	S4	S5
S1		.31	1.01	.75	1.03
S2			1.00	.69	.90
S3				.61	.42
S4					.37

Table 5.3: Dissimilarities between taxa

We begin by choosing the closest pair of taxa to join, just as we did with UPGMA. Looking at our distance table, S1 and S2 are the first pair to join.

In order to join them *without* placing them at an equal distance from a common ancestor, we temporarily reduce to the 3 taxa case by combining *all other* taxa into a group. For our data, we thus introduce the group S3-S4-S5. We find the distance from each of S1 and S2 to the group by averaging their distances to each group member. The distance from S1 to S3-S4-S5 is thus $d(\text{S1}, \text{S3-S4-S5}) = (1.01 + .75 + 1.03)/3 = .93$, while the distance from S2 to S3-S4-S5 is $d(\text{S2}, \text{S3-S4-S5}) = (1.00 + .69 + .90)/3 = .863$. This gives us Table 5.4.

	S1	S2	S3-S4-S5
S1		.31	.93
S2			.863

Table 5.4: Distances between groups; FM algorithm, step 1a

With only three taxa in this table, we can exactly fit the data to the tree using the 3-point formulas to get Figure 5.5. The key point here is that the

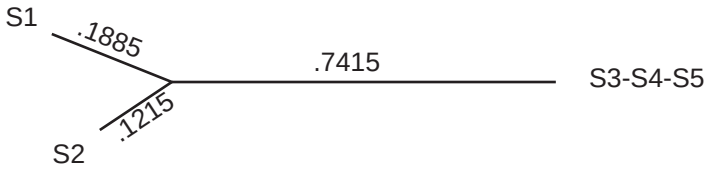

Figure 5.5: FM Algorithm; step 1

3-point formulas, unlike UPGMA, can produce unequal distances of taxa from a common ancestor.

We now keep only the edges ending at S1 and S2 in Figure 5.5, and return to our original data. Remember, the group S3-S4-S5 was only needed temporarily so we could use the 3-point formulas; we didn't intend to join those taxa together yet. Since we have joined S1 and S2, however, we combine them into a group for the rest of the algorithm, just as we would have done with UPGMA. This gives us Table 5.5.

	S1-S2	S3	S4	S5
S1-S2		1.005	.72	.965
S3			.61	.42
S4				.37

Table 5.5: Distances between groups; FM algorithm, step 1b

We again look for the closest pair (now S4 and S5), and join them in a similar manner. We combine everything but S4 and S5 into a single temporary

group S1-S2-S3 and compute $d(\text{S4}, \text{S1-S2-S3}) = (.75 + .69 + .61)/3 = .683$ and $d(\text{S5}, \text{S1-S2-S3}) = (1.03 + .90 + .42)/3 = .783$. This gives us Table 5.6. Applying

	S1-S2-S3	S4	S5
S1-S2-S3		.683	.783
S4			.37

Table 5.6: Distances between groups; FM algorithm, step 2a

the 3-point formulas to Table 5.6 produces Figure 5.6.

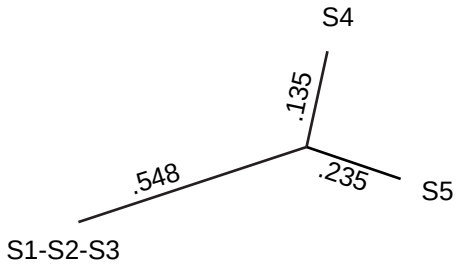

Figure 5.6: FM Algorithm; step 2

We keep the edges joining S4 and S5 in Figure 5.6, discarding the edge leading to the temporary group S1-S2-S3. Thus we now have two joined groups, S1-S2 and S4-S5. To compute a new table containing these two groups we've found, we average $d(\text{S1-S2}, \text{S4-S5}) = (.75 + 1.03 + .69 + .90)/4 = .8425$ and $d(\text{S3}, \text{S4-S5}) = (.61 + .42)/2 = .515$. We've already computed $d(\text{S1-S2}, \text{S3})$ so we produce Table 5.7. At this point we can fit a tree exactly to the table by a

	S1-S2	S3	S4-S5
S1-S2		1.005	.8425
S3			.515

Table 5.7: Distances between groups; FM algorithm, step 2b

final application of the 3-point formulas, yielding Figure 5.7.

Now we replace the groups in this last diagram with the branching patterns we've already found for them. This gives Figure 5.8.

Our final step is to fill in the remaining lengths a and b, using the lengths in Figure 5.7. Since S1 and S2 are on average $(.1885 + .1215)/2 = .155$ from the vertex joining them and S4 and S5 are on average $(.135 + .235)/2 = .185$ from the vertex joining them, we compute $a = .66625 - .155 = .51125$ and $b = .17625 - .185 = -.00875$ to assign lengths to the remaining sides.

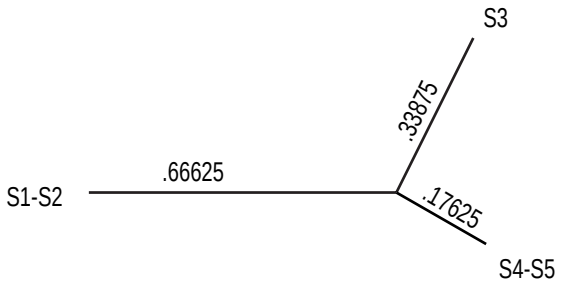

Figure 5.7: FM Algorithm; step 3

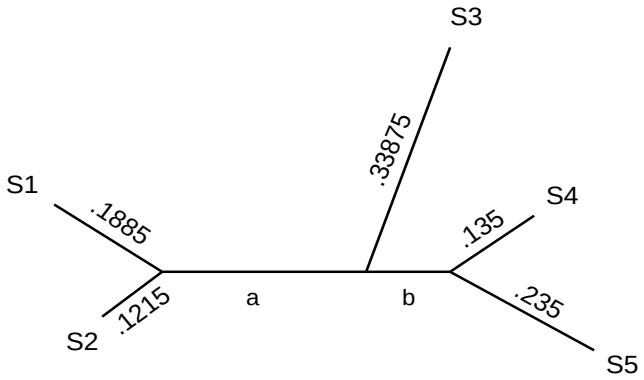

Figure 5.8: FM Algorithm; completion

Notice that one edge has turned out to have negative length. Since that can't really be meaningful, many practitioners would choose to simply reassign the length as 0. If this happens, however, we should at least check that the negative length was close to 0 or we would worry about the quality of the data.

While it may seem surprising at first, both the Fitch-Margoliash algorithm and UPGMA will produce exactly the same topological tree when applied to a data set. The reason for this is that when deciding which taxa or groups to join at each step, both methods consider exactly the same collapsed data table and both choose the pair corresponding to the smallest entry in the table. It is only the metric features of the resulting trees that will differ. This undermines a bit the hope that the Fitch-Margoliash algorithm is much better than UPGMA. While it may produce a better metric tree, topologically it never differs.

To be fair, Fitch and Margoliash actually proposed their algorithm not as an end in itself, but rather as a heuristic method for producing a tree likely to have a certain optimality property (see Exercise 9). We are viewing it here, like UPGMA, as a step toward the Neighbor Joining algorithm which will be introduced shortly. Familiarity with UPGMA and the Fitch-Margoliash algorithm

will aid us in understanding that more elaborate method.

5.3 The Four-point Condition

If we are interested in non-molecular clock trees, as biologists often are, there is a fundamental flaw in using UPGMA to construct trees, and this flaw is not corrected by the use of the 3-point formulas alone.

To see this, consider the metric tree with 4 taxa in Figure 5.9. Here x and y represent specific lengths, with x much smaller than y.

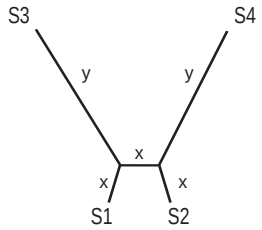

Figure 5.9: The true relationships of taxa S1, S2, S3, S4

Then perfect dissimilarity data would give us the distances in Table 5.8.

	S1	S2	S3	S4
S1		$3x$	$x+y$	$2x+y$
S2			$2x+y$	$x+y$
S3				$x+2y$

Table 5.8: Distances between taxa in Figure 5.9

But if y is much bigger than x, (in fact, $y > 2x$ is good enough) then the closest taxa by distance are S1 and S2. Now the first step of UPGMA will be to look for the smallest dissimilarity in the table, and join those two taxa. This means we'll choose $S1$ and $S2$ as the most closely related, and relate them by a tree that already has a topological mistake. No matter what we do to compute edge lengths, or how the subsequent steps proceed, we will not recover the original tree topology. The essential problem here is a conflict between closeness of taxa by the metric/dissimilarity and closeness in the graph theoretic sense.

UPGMA can fail to reconstruct the correct tree topology, even if given dissimilarities that arise from a tree metric, because it uses the minimal dissimilarity as a way of guessing which leaves are at a minimal graph-theoretic distance. But the tree-metric distance and the graph-theoretic distance are not necessarily minimized between the same pair of leaves. This is the issue we need to explore theoretically, so that in the next section we can give a practical algorithm to address the problem.

Definition. Two leaves of a tree that are graph-theoretic distance 2 apart are said to form a *cherry*, or are said to be *neighbors*.

We'll focus on quartet trees — trees relating 4 taxa. Note that a binary quartet tree is determined by specifying which pairs of taxa are neighbors.

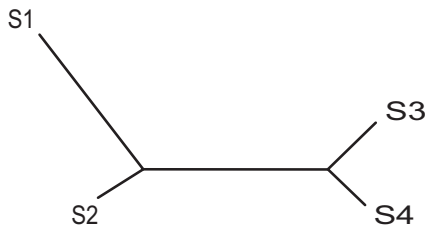

Figure 5.10: A quartet tree with neighbors S1, S2 and S3, S4

Consider the quartet tree in Figure 5.10, viewed as a metric tree with positive edge lengths. Letting $d_{ij} = d(\mathrm{S}i, \mathrm{S}j)$ denote the tree metric between leaves, we see that

$$d_{12} + d_{34} < d_{13} + d_{24},$$

since the quantity on the left includes only the lengths of the four edges leading from the leaves of the tree, while the quantity on the right includes all of those and, in addition, twice the central edge length. Notice also that

$$d_{13} + d_{24} = d_{14} + d_{23}$$

by similar reasoning.

These observations give us a way of using metric information to identify which leaves form a cherry. We summarize this as a theorem.

Theorem 13. A metric quartet tree relating taxa a, b, c, d which has positive edge lengths has neighbor pairs a, b and c, d if, and only if, any (and hence all) of the three inequalities/equalities in the following hold:

$$d(a, b) + d(c, d) < d(a, c) + d(b, d) = d(a, d) + d(b, c).$$

It should already be clear this observation will be useful, since identifying quartet trees enables us to identify larger tree topologies: For a positive-edge-length binary phylogenetic X-tree T, we get induced positive-edge-length trees for every 4-element subset of X, and hence we can identify the appropriate quartet tree for each such subset. Then, by results in Chapter 4, these quartet trees determine the topology of the tree T.

However, we can do even better in this vein, giving a condition on a dissimilarity map that enables us to relate it to a tree metric.

Definition. A dissimilarity map δ on X satisfies the *four-point condition* if for every choice of four elements $x, y, z, w \in X$ (including non-distinct ones),

$$\delta(x, y) + \delta(z, w) \leq \max\{\delta(x, z) + \delta(y, w), \delta(x, w) + \delta(y, z)\}. \tag{5.3}$$

Note the non-strict inequality in this definition allows us to formulate the following for trees that are not necessarily binary.

Theorem 14. For any metric tree with positive edge lengths, the tree metric will satisfy the four-point condition.

Proof. We leave the proof as Exercise 19. □

The converse to this theorem is more remarkable, even if the proof we give is a bit tedious.

Theorem 15. Suppose $|X| \geq 3$, and for a dissimilarity map δ on X, $\delta(x, y) \neq 0$ whenever $x \neq y$. Then if δ satisfies the four-point condition, it is the restriction of a tree metric from a unique metric X-tree with positive edge lengths.

Proof. The case of $|X| = 3$ follows from the 3-point formulas with a little extra effort to show edge lengths are positive (see Exercise 20). The case $|X| = 4$ is Exercise 21.

For larger $|X| = N$, we proceed by induction. Suppose $X = \{\mathrm{S1}, \mathrm{S2}, \ldots, \mathrm{S}N\}$. Let T' be the unique metric X'-tree with positive edge lengths relating $X' = \{\mathrm{S1}, \mathrm{S2}, \ldots, \mathrm{S}(N-1)\}$ whose tree metric restricts to δ on X'. For any taxon labeling an internal vertex of T', temporarily attach a new edge at that vertex and move the label to its other end, so all labels are now on unique leaves of a phylogenetic X'-tree. Now pick a pair of neighbors on this modified tree, and, assume these are S1, S2 by renaming them if necessary. We will say such a pair of taxa are 'generalized neighbors' on the original T'.

Consider all the unique metric trees for the subsets $\{\mathrm{S1}, \mathrm{S2}, \mathrm{S}j, \mathrm{S}N\}$ whose tree metrics agree with δ. We claim that at least one of the pairs $\mathrm{S1}, \mathrm{S}N$, or $\mathrm{S2}, \mathrm{S}N$, or $\mathrm{S1}, \mathrm{S2}$ is a pair of generalized neighbors in all of these 4-leaf trees. To see why this is so, suppose $\mathrm{S1}, \mathrm{S}N$ are generalized neighbors in the tree for $\{\mathrm{S1}, \mathrm{S2}, \mathrm{S}k, \mathrm{S}N\}$ for some k. Then by the 4-leaf case, the vertex where paths to S1, S2 and SN meet is at the same or smaller graph-theoretic (and metric) distance from S1 than the vertex where paths from S1, S2, Sk meet. But since S1 and S2 are generalized neighbors in T', this last vertex is where paths from S1, S2, and Sj meet for all $j < N$. Thus S1 and SN are generalized neighbors in the trees for all $\{\mathrm{S1}, \mathrm{S2}, \mathrm{S}j, \mathrm{S}N\}$. Similarly if $\mathrm{S2}, \mathrm{S}N$ are generalized neighbors for one of the 4-leaf trees, they are generalized neighbors for all. If there are no 4-leaf trees where either $\mathrm{S1}, \mathrm{S}N$ or $\mathrm{S2}, \mathrm{S}N$ are generalized neighbors, then $\mathrm{S1}, \mathrm{S2}$ must be generalized neighbors in all.

Using the claim of the last paragraph, assume $\mathrm{S1}, \mathrm{S}N$ are always generalized neighbors for all quartets of the form $\mathrm{S1}, \mathrm{S2}, \mathrm{S}j, \mathrm{S}N$ by interchanging the names of $\mathrm{S1}, \mathrm{S2}, \mathrm{S}N$ if necessary. Again let T' be the tree for $\{\mathrm{S1}, \mathrm{S2}, \ldots, \mathrm{S}(N-1)\}$. (So now S1 and S2 may not be generalized neighbors in T'.) We leave to the reader the final step of giving the unique way to 'attach' SN to T' consistent with the dissimilarity values (Exercise 22).

□

5.4 The Neighbor Joining Algorithm

In practice, UPGMA with its molecular clock assumption is seldom used on biological data. Much preferred is a more elaborate algorithm, called Neighbor Joining, which is built on the four-point condition.

However, it is important that Neighbor Joining *not require* that the four-point condition actually be met for the dissimilarity data to which it is applied, since dissimilarities computed from data should be expected at best to be only roughly consistent with a metric tree. We should never expect an exact fit to a metric tree, and so we will want to perform various averaging processes as we go along in order to smooth out some of the errors in fit.

To motivate the algorithm, we imagine a binary positive-edge-length tree in which taxa S1 and S2 are neighbors joined at vertex V, with V somehow joined to the remaining taxa S3, S4,..., SN, as in Figure 5.11.

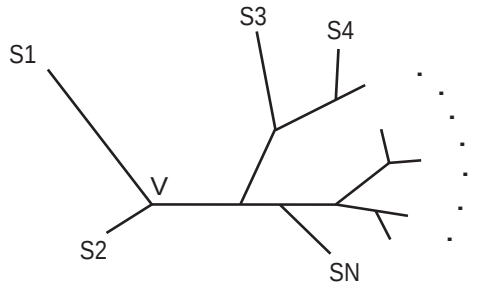

Figure 5.11: Tree with S1 and S2 neighbors

If our dissimilarity data agreed exactly with a metric for this tree then for every $i, j = 3, 4, \ldots, N$, we'd find from the four-point condition that

$$d_{12} + d_{ij} < d_{1i} + d_{2j}. \tag{5.4}$$

For fixed i, there are $N - 3$ possible choices of j with $3 \leq j \leq N$ and $j \neq i$. If we sum the inequalities (5.4) for these j we get

$$(N-3)d_{12} + \sum_{\substack{j=3 \\ j\neq i}}^{N} d_{ij} < (N-3)d_{1i} + \sum_{\substack{j=3 \\ j\neq i}}^{N} d_{2j}. \tag{5.5}$$

To simplify this, define the total dissimilarity between taxon Si and all other taxa as

$$R_i = \sum_{j=1}^{N} d_{ij}.$$

Then adding $d_{i1} + d_{i2} + d_{12}$ to each side of inequality (5.5) allows us to write it in the simpler form

$$(N-2)d_{12} + R_i < (N-2)d_{1i} + R_2.$$

Subtracting $R_1+R_2+R_i$ from each side of this then gives it the more symmetric form

$$(N-2)d_{12} - R_1 - R_2 < (N-2)d_{1i} - R_1 - R_i.$$

If we apply the same argument to Sn and Sm, rather than S1 and S2, we are led to define

$$M_{nm} = (N-2)d_{nm} - R_n - R_m. \tag{5.6}$$

Then if Sn and Sm are neighbors, we'll have that

$$M_{nm} < M_{nk}$$

for all $k \neq m$.

This gives us the criterion used for Neighbor Joining: From the dissimilarity data, compute a new table of values for M_{ij} using equation (5.6). Then choose to join the pair Si, Sj of taxa with the smallest value of M_{ij}.

The argument above shows that if Si and Sj are neighbors in a metric tree producing the dissimilarity data, then the value M_{ij} will be the smallest of the values in the ith row and jth column of the table for M. However, this is not enough! An additional argument is still needed to show the smallest entry in the *entire table* for M truly identifies a pair of neighbors. Though this claim is plausible, we outline a proof of it in Exercise 28.

Since the full Neighbor Joining algorithm is fairly complicated, we state it formally.

Algorithm.

1. Given dissimilarity data for N taxa, compute a new table of values of M using equation (5.6). Choose the smallest value in the table for M to determine which taxa to join. (This value may be, and usually is, negative, so 'smallest' means the negative number with the greatest absolute value.)

2. If Si and Sj are to be joined at a new vertex V, temporarily collapse all other taxa into a single group G, and determine the lengths of the edges from Si and Sj to V by using using the 3-point formulas for Si, Sj, and G as in the algorithm of Fitch and Margoliash.

3. Determine distances/dissimilarities from each of the taxa Sk in G to V by applying the 3-point formulas to the distance data for the three taxa Si, Sj and Sk. Now include V in the table of dissimilarity data, and drop Si and Sj.

4. The distance table now includes $N-1$ taxa. If there are only 3 taxa, use the 3-point formulas to finish. Otherwise go back to step 1.

Exercise 28 completes the proof of the following theorem, by showing that for dissimilarity data consistent with a tree metric, the Neighbor Joining algorithm really does pick neighbors to join.

Theorem 16. Suppose a dissimilarity map on X is the restriction of a tree metric for a binary metric phylogenetic X-tree T with all positive edge lengths. Then the Neighbor Joining algorithm will reconstruct T and its edge lengths.

As you can see already, Neighbor Joining is not pleasant to do by hand. Even though each step is relatively straightforward, it's easy to get lost in the process with so much arithmetic to do. In the exercises you'll find an example partially worked that you should complete to be sure you understand the steps. After that, we suggest you use a computer program to avoid mistakes (or, even better, write your own program).

The accuracy of various tree construction methods – the ones outlined so far in these notes and many others – has been tested primarily through simulating DNA mutation according to certain specified models of mutation along phylogenetic trees and then applying the methods to see how often they recover the tree that was the basis for the simulation. (A study has also been done with real taxa related by a known phylogenetic tree created in a laboratory. However, it is not easy to produce such data.) These tests have lead researchers to be considerably more confident of the results given by Neighbor Joining than by UPGMA. While UPGMA may be reliable, or even preferred, in some circumstances, Neighbor Joining works well on a broader range of data. For instance, if no molecular clock is operating, Neighbor Joining is superior, since it makes no assumptions that equal amounts of mutation occur between an ancestral sequence and all its currently extant descendants. Since there is much data indicating the molecular clock hypothesis is often violated, Neighbor Joining has become the distance method of choice for tree construction.

5.5 Exercises

1. For the tree in Figure 5.3 constructed by UPGMA, compute a table of distances between taxa along the tree. How does this compare to the original data table of dissimilarities?

2. Suppose four sequences S1, S2, S3, and S4 of DNA are separated by dissimilarities as in Table 5.9. Construct a rooted tree showing the relationships between S1, S2, S3, and S4 by UPGMA.

	S1	S2	S3	S4
S1		1.2	.9	1.7
S2			1.1	1.9
S3				1.6

Table 5.9: Dissimilarity data for Problems 2 and 5

3. Perform UPGMA on the data in Table 5.3 that was used in the text in the example of the FM algorithm. Does UPGMA produce the same tree as the FM algorithm topologically? metrically?

4. The FM algorithm utilizes the fact that dissimilarity data relating three terminal taxa can be exactly fit by the single unrooted tree relating them.

a. Derive the 3-point formulas of equation 5.1.

b. If the dissimilarities are $d_{AB} = .634$, $d_{AC} = 1.327$, and $d_{BC} = .851$, what are the lengths x, y, and z?

5. Use the FM algorithm to construct an unrooted tree for the data in Table 5.9 that was also used in Problem 2. How different is the result?

6. A desirable feature of a dissimilarity map on sequences is that it be *additive*, in the sense that if S0 is an ancestor of S1, which is in turn an ancestor of S2, then

$$d(\text{S0}, \text{S2}) = d(\text{S0}, \text{S1}) + d(\text{S1}, \text{S2}).$$

a. Explain why an additive dissimilarity map is desirable if we are trying to use dissimilarities to construct metric trees.

b. Give an example to illustrate the Hamming dissimilarity might not be additive.

c. If mutations are rare, why might the Hamming dissimilarity be approximately additive?

7. Suppose three terminal taxa are related by an unrooted metric tree.

a. If the three edge lengths are .1, .2, and .3, explain why a molecular clock hypothesis must be invalid, no matter where the root is located.

b. If the three edge lengths are .1, .1 and .2, explain why the molecular clock hypothesis *might* be valid. If it is, where would the root be located?

c. If the three edge lengths are .1, .2 and .2, explain why the molecular clock hypothesis must be invalid, no matter where the root is located.

8. While distance data for 3 terminal taxa can be exactly fit to an unrooted tree, if there are 4 (or more) taxa, this is usually not possible.

a. Draw an unrooted tree with terminal taxa A, B, C, D. Denote the lengths of the five edges by r, s, t, u, and v.

b. Denoting distances between terminal taxa with notation like d_{AB}, write down equations for each of the 6 such distances in terms of r, s, t, u, and v. Explain why, if you are given numerical distances between terminal taxa, these equations are not likely to have an exact solution.

c. Give a concrete example of values of the 6 distances between terminal taxa so that the equations in part (b) cannot be solved exactly. Give another example of values where the equations can be solved.

d. How many equations in how many unknowns would need to be solved to exactly fit dissimilarity data between n taxa to a specific binary tree?

9. A number of different measures of goodness of fit between distance data and metric trees have been proposed. Let d_{ij} denote the distance between taxa i and j obtained from experimental data, e_{ij} denote the distance from i to j along the tree. A few of the measures that have been proposed are:

$$s_{FM} = \left(\sum_{i,j} \left(\frac{d_{ij} - e_{ij}}{d_{ij}} \right)^2 \right)^{\frac{1}{2}},$$
$$s_F = \sum_{i,j} |d_{ij} - e_{ij}|,$$
$$s_{TNT} = \left(\sum_{i,j} (d_{ij} - e_{ij})^2 \right)^{\frac{1}{2}}.$$

In all these measures, the sums include terms for each distinct pair of taxa, i and j.

a. Compute these measures for the tree constructed in the text using the FM algorithm, as well as the tree constructed from the same data using UPGMA in Problem 3. According to each of these measures, which of the two trees is a better fit to the data?

b. Explain why these formulas are reasonable ones to use to measure goodness of fit. Explain how the differences between the formulas make them more or less sensitive to different types of errors.

Note: Fitch and Margoliash proposed choosing the optimal metric tree to fit data as the one that minimized s_{FM}. The FM algorithm was introduced in an attempt to get an approximately optimal tree.

10. Read the data file `seqdata.mat` into MATLAB by typing `load seqdata`. Then use UPGMA with the Hamming dissimilarity to construct a tree for the sequences `a1`, `a2`, `a3`, and `a4`.

    To compute the dissimilarity between sequences `a1,a2` try the following commands: `a1~=a2,` `sum(a1~=a2),` `sum(a1~=a2)/length(a1)` .

    a. Draw the UPGMA tree for the four taxa, labeling each edge with its length.

    b. From your edge lengths, compute the distances between taxa along the tree. Are these close to the original distances?

    Note: This data was simulated according to a Jukes-Cantor model (see Chapter 6) with a molecular clock.

11. Repeat the last problem, but use the FM algorithm instead of UPGMA. How different is the tree you produce from the UPGMA tree? Explain.

12. Investigate the performance of UPGMA with the Hamming dissimilarity to construct a tree for the sequences `b1`, `b2`, `b3`, `b4`, and `b5` in the data file `seqdata.mat`. See Problem 10 for useful MATLAB commands.

    a. Draw the UPGMA tree for the five taxa, labeling each edge with its length.

b. From your edge lengths, compute the distances between taxa along the tree. Are these close to the original data?

Note: This data was simulated according to a Jukes-Cantor model (Chapter 6), but *without* a molecular clock.

13. Repeat the last problem, but use the FM algorithm instead of UPGMA. How different is the tree you produce from the UPGMA tree? Explain.

14. Suppose the unrooted metric tree in Figure 5.12 correctly describes the evolution of taxa A, B, C, and D.

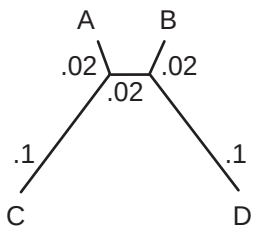

Figure 5.12: Tree for problem 14

a. Explain why, regardless of the location of the root, a molecular clock could not have operated.

b. Give the array of distances between each pair of the four taxa. Perform UPGMA on that data.

c. UPGMA did not reconstruct the correct tree. Where did it go wrong? What was it about this metric tree that led it astray?

d. Explain why the FM algorithm will also not reconstruct the correct tree.

15. Show that every binary phylogenetic tree with at least three taxa has at least two cherries. (A binary tree with n leaves that has only two cherries is sometimes called a *caterpillar*. Despite the unfortunate mixing of metaphors, why is this terminology reasonable?)

16. Show that if a dissimilarity map on X satisfies the four-point condition, then it satisfies the triangle inequality on X. (Rather than deducing this from Theorem 15, show it directly by taking several of x, y, z, w to be equal.)

17. If a dissimilarity map is the restriction of a tree metric to X, then it is a metric on X. Show that the converse is false by giving an example of a metric on a set X that is not the restriction of a tree metric.

18. Show that the four-point condition is equivalent to the following statement: For every choice of $x, y, z, w \in X$, of the three quantities

$$\delta(x,y)+\delta(z,w),\ \ \delta(x,z)+\delta(y,w),\ \ \delta(x,w)+\delta(y,z)$$

the two (or three) largest are equal.

19. Prove Theorem 14, by first observing that it's enough to prove it for 4-leaf trees. For 4-leaf trees, be sure you consider both binary and non-binary trees, and cases where x, y, z, w are all distinct and when they are not.

20. Prove Theorem 15 for the case $|X| = 3$. (Be sure you show all edge lengths are positive!)

21. Prove Theorem 15 for the case $|X| = 4$.

22. Give the final step of the proof of Theorem 15.

23. Before working through an example of Neighbor Joining, it's helpful to derive formulas for Steps 2 and 3 of the algorithm. Suppose we've chosen to join Si and Sj in Step 1.

    a. Show that for Step 2, the distances of Si and Sj to the internal vertex V can be computed by

$$d(\mathrm{S}i, V) = \frac{\delta(\mathrm{S}i, \mathrm{S}j)}{2} + \frac{R_i - R_j}{2(N-2)},$$
$$d(\mathrm{S}j, V) = \frac{\delta(\mathrm{S}i, \mathrm{S}j)}{2} + \frac{R_j - R_i}{2(N-2)}.$$

    Then show the second of these formulas can be replaced by

$$d(\mathrm{S}j, V) = \delta(\mathrm{S}i, \mathrm{S}j) - d(\mathrm{S}i, V).$$

    b. Show that for Step 3, the distances of Sk to V, for $k \neq i, j$ can be computed by

$$d(\mathrm{S}k, V) = \frac{\delta(\mathrm{S}i, \mathrm{S}k) + \delta(\mathrm{S}j, \mathrm{S}k) - \delta(\mathrm{S}i, \mathrm{S}j)}{2}.$$

24. Consider the distance data of Table 5.10. Use the Neighbor Joining algo-

	S1	S2	S3	S4
S1		.83	.28	.41
S2			.72	.97
S3				.48

Table 5.10: Taxon distances for Problem 24

    rithm to construct a tree as follows:

    a. Compute R_1, R_2, R_3 and R_4 and then a table of values for M for the taxa S1, S2, S3, and S4. To get you started

$$R_1 = .83 + .28 + .41 = 1.52 \text{ and } R_2 = .83 + .72 + .97 = 2.52$$

so

$$M(\mathrm{S1},\mathrm{S2}) = (4-2).83 - 1.52 - 2.52 = -2.38.$$

b. If you did part (a) correctly, you should have a tie for the smallest value of M. One of these smallest values is $M(\mathrm{S1},\mathrm{S4}) = -2.56$, so let's join S1 and S4 first.

For the new vertex V where S1 and S4 join, compute $d(\mathrm{S1}, V)$ and $d(\mathrm{S4}, V)$ by the formulas in part (a) of the previous problem.

c. Compute $d(\mathrm{S2}, V)$ and $d(\mathrm{S3}, V)$ by the formulas in part (b) of the previous problem.

Put your answers into the new distance Table 5.11.

	V	S2	S3
V		—	—
S2			.72

Table 5.11: Group distances for Problem 24

d. Since there are only 3 taxa left, use the 3-point formulas to fit V, S2, and S3 to a tree.

e. Draw your final tree by attaching S1 and S4 to V with the distances given in part (b).

25. Consider the distance data in Table 5.12, which is exactly fit by the tree of

	S1	S2	S3	S4
S1		.3	.4	.5
S2			.5	.4
S3				.7

Table 5.12: Taxon distances for Problem 25

Figure 5.9, with $x = .1$ and $y = .3$.

a. Use UPGMA to reconstruct a tree from this data. Is it correct?

b. Use Neighbor Joining to reconstruct a tree from this data. Is it correct?

26. Perform the Neighbor Joining algorithm on the distance data used in the examples in the text of Sections 5.1 and 5.2. To use MATLAB to do this for the first example, enter the distance array as

```
D=[0 .45 .27 .53; 0 0 .40 .50; 0 0 0 .62; 0 0 0 0]
```

and taxa names as

```
Taxa={'S1','S2','S3','S4'}
```

Then type `nj(D,Taxa{:})`.

a. Does NJ on the 4-taxon example produce the same tree as UPGMA?

b. Does NJ on the 5-taxon example produce the same tree as the Fitch-Margoliash algorithm?

27. Use the Hamming dissimilarity and the Neighbor Joining program `nj` to construct trees for the following simulated sequence data in `seqdata.mat`. Compare your results to that produced by other methods in Exercises 10–13. Has whether a molecular clock operated in the simulation affected the results?

    a. `a1`, `a2`, `a3`, and `a4` (molecular clock)

    b. `b1`, `b2`, `b3`, `b4`, and `b5` (no molecular clock)

28. Complete the main remaining part of the proof of Theorem 16, that the criterion used to pick neighbors on dissimilarity data arising from a positive edge length binary metric tree is valid, by following the outline below of the proof of Studier-Keppler.

    a. Show $M_{mn} - M_{ij} = \sum_{k \neq i,j,m,n} ((d_{ik} + d_{jk} - d_{ij}) - (d_{mk} + d_{nk} - d_{mn}))$.

    Now suppose M_{ij} is minimal but that i and j are not neighbors.

    b. Explain why neither i nor j have neighbors.

    Pick some pair of neighbors Sm, Sn, and consider the minimal subtree Q of T joining Si, Sj, Sm, Sn inside T. Denote the internal vertices of it with valence 3 by u (joined to i, j) and v (joined to m, n). For all additional taxa Sk, show

    c. If the minimal path for Sk to Q joins at a vertex x along the path from i to j, then $(d_{ik} + d_{jk} - d_{ij}) - (d_{mk} + d_{nk} - d_{mn}) = -2d_{ux} - 2d_{uv}$.

    d. If the minimal path for Sk to Q joins at a vertex x along the minimal path from u to v, then $(d_{ik} + d_{jk} - d_{ij}) - (d_{mk} + d_{nk} - d_{mn}) = -4d_{vx} + 2d_{uv}$.

    e. Conclude from the fact that left hand side of the equality in (a) is non-negative that at least as many of the Sk are described by (d) as by (c); and thus there are strictly more leaves that are joined to the path from i to j at u than at any other vertex.

    f. Explain why since i and j have no neighbors there must be a pair of neighbors r, s which are joined to the path from i to j at some vertex other than u.

    g. Applying the argument of parts (c,d,e) to r, s instead of m, n, arrive at a contradiction.

Chapter 6

Probabilistic Models of DNA Mutation

So far we've avoided a detailed attempt to mathematically describe the mutation processes DNA undergoes as evolution proceeds. As long as we think mutations are rare, then this isn't too problematic. Indeed the principal of parsimony, and the Hamming dissimilarity measure for sequences both seem reasonable under such an assumption.

However, if mutations are not so rare, then we might expect that along any edge of a tree a particular site might experience more than one mutation. Though only one mutation is observable by comparing initial and final bases, several might have occurred. Even worse, due to a back mutation, there could have been several even though we observe none. Under these circumstances both the principal of parsimony and the Hamming dissimilarity map would give inappropriate views as to how much mutation has occurred — they both underestimate the true amount of change.

To do better, we need explicit mathematical models of how mutations occur, and since mutations appear to be random events, we formulate these probabilistically. Ultimately these models will be used both to develop improved dissimilarity maps for distance methods, and as a basis for the third major method of phylogenetic inference we'll discuss, Maximum Likelihood.

6.1 A Simple Example

To begin, we'll present a very simple example in order to illustrate the basic modeling approach. We'll keep this discussion as informal as possible, and then be more careful with our terminology later on. For those who have seen Markov models before, either in a linear algebra or probability course, the framework will be familiar, but we will not assume previous exposure.

We will model *one site* in a DNA sequence, and how it changes from an ancestral sequence to a descendant sequence. In other words, the tree we imagine

is particularly simple: it has only one edge.

To simplify things we focus only on whether at that site a purine R or a pyrimidine Y appears. We only care about these classes, not on the precise bases. To be concrete about the situation we wish to describe, suppose we somehow had access to an ancestral sequence S0 and a descendant sequence S1.

S0 : `RRYRYRYYRYYYRYRYYRRYY`

S1 : `RYYRYYYYRYYYRYRYYRRYR`

To describe an arbitrary site in an ancestral sequence, we simply specify the probabilities that site might be occupied by either an R or Y. For instance $(\mathcal{P}(S0 = R), \mathcal{P}(S0 = Y)) = (p_R, p_Y) = (.5, .5)$ would indicates an equal chance of each, while $(p_R, p_Y) = (.6, .4)$ would indicate a greater likelihood of a purine. Since our site must have either a purine or a pyrimidine, note the two probabilities must add to 1.

Although this model is describing only one site, we view the data sequences as being many different trials of the same probabilistic process. Thus (p_R, p_Y), the probabilities that a site in an idealized ancestral sequence of infinite length is occupied by an R or Y, can be estimated by the frequencies at which these occur in the observed sequence. For example, the 21-base sequence S0 has 9 Rs and 12 Ys, which leads us to estimate $p_R = 9/21$ and $p_Y = 12/21$. In the language of statistics, to obtain this estimate we are making an i.i.d. assumption, that each site behaves *independently* with an *identical distribution*.

With the ancestral sequence described, we now focus on probabilities of mutations as S0 evolves into S1. There are 4 possibilities here, $R \to R$, $R \to Y$, $Y \to R$ and $Y \to Y$, which in our data occurred in a total of 7, 2, 1, and 11 sites, respectively. It is most convenient to summarize this through conditional probabilities. For instance, the conditional probability that an ancestral R remains an R is denoted by

$$\mathcal{P}(\text{S1} = R \mid \text{S0} = R),$$

which we read as 'the probability that we have an R in S1 *given that* we had an R in S0.' For our data, we estimate

$$\mathcal{P}(\text{S1} = R \mid \text{S0} = R) = 7/9,$$

since of the 9 ancestral Rs, 7 remained Rs. Similarly we estimate

$$\begin{aligned}
\mathcal{P}(\text{S1} = Y \mid \text{S0} = R) &= 2/9, \\
\mathcal{P}(\text{S1} = R \mid \text{S0} = Y) &= 1/12, \\
\mathcal{P}(\text{S1} = Y \mid \text{S0} = Y) &= 11/12.
\end{aligned}$$

Notice

$$\begin{aligned}
\mathcal{P}(\text{S1} = R \mid \text{S0} = R) + \mathcal{P}(\text{S1} = Y \mid \text{S0} = R) &= 1, \\
\mathcal{P}(\text{S1} = R \mid \text{S0} = Y) + \mathcal{P}(\text{S1} = Y \mid \text{S0} = Y) &= 1,
\end{aligned}$$

as the total conditional probability of either an X or Y appearing in S1, assuming that the base in S0 is given, must be 1.

Now our model is summarized by six probabilities, which we organize into a row vector and a matrix:

$$\mathbf{p}_0 = \begin{pmatrix} p_R & p_Y \end{pmatrix} = \begin{pmatrix} 9/21 & 12/21 \end{pmatrix},$$

$$\begin{aligned} M &= \begin{pmatrix} \mathcal{P}(\mathrm{S1} = R \mid \mathrm{S0} = R) & \mathcal{P}(\mathrm{S1} = Y \mid \mathrm{S0} = R) \\ \mathcal{P}(\mathrm{S1} = R \mid \mathrm{S0} = Y) & \mathcal{P}(\mathrm{S1} = Y \mid \mathrm{S0} = Y) \end{pmatrix} \\ &= \begin{pmatrix} p_{RR} & p_{RY} \\ p_{YR} & p_{YY} \end{pmatrix} = \begin{pmatrix} 7/9 & 2/9 \\ 1/12 & 11/12 \end{pmatrix}. \end{aligned}$$

(Note the switch in order here, where $\mathcal{P}(\mathrm{S1} = R \mid \mathrm{S0} = Y) = p_{YR}$, for instance.)

One point of this matrix notation is that the product $\mathbf{p}_0 M$ has meaningful entries:

$$\mathbf{p}_0 M = \begin{pmatrix} p_R & p_Y \end{pmatrix} \begin{pmatrix} p_{RR} & p_{RY} \\ p_{YR} & p_{YY} \end{pmatrix} = \begin{pmatrix} p_R p_{RR} + p_Y p_{YR} & p_R p_{RY} + p_Y p_{YY} \end{pmatrix}$$

where, for instance, the left entry is

$$\begin{aligned} p_R p_{RR} + p_Y p_{YR} &= \mathcal{P}(\mathrm{S1} = R \mid \mathrm{S0} = R)\mathcal{P}(\mathrm{S0} = R) \\ &\qquad + \mathcal{P}(\mathrm{S1} = R \mid \mathrm{S0} = Y)\mathcal{P}(\mathrm{S0} = Y) \\ &= \mathcal{P}(\mathrm{S1} = R), \end{aligned}$$

using various multiplication and sum rules of probabilities to interpret this. Similarly, the right entry is $\mathcal{P}(\mathrm{S1} = Y)$, so we have

$$\mathbf{p}_1 = \mathbf{p}_0 M,$$

where $\mathbf{p}_1$ gives us probability distribution of Rs and Ys for S1. Thus the entries of M not only describe the mutation process between the two sequences described by $\mathbf{p}_0$ and $\mathbf{p}_1$, but the process of matrix multiplication actually 'does' the mutation process.

What, then, might happen if the sequence S1 continues to evolve? Assuming circumstances are similar to those during the evolution of S0 to S1, and the elapsed time is similar, it's reasonable to hypothesize that we would obtain a sequence S2 whose R/Y composition would be described by

$$\mathbf{p}_2 = \mathbf{p}_1 M = \mathbf{p}_0 M^2.$$

Thinking of M as describing one time-step, we now have a *model* of sequence evolution using discrete time, with a descendant sequence after n time steps being described by

$$\mathbf{p}_n = \mathbf{p}_0 M^n.$$

The *parameters* of our model are a vector $\mathbf{p}_0$ of non-negative numbers that sum to 1, which we call the *root distribution vector*, and a matrix M of non-negative numbers whose rows sum to one, which we call a *Markov matrix*.

In this presentation, we initially thought of M as describing evolution along a full edge. Then we imagined it as describing evolution for just one time-step, and that an edge might be many time steps long. While both of these views are useful in some settings, they essentially use a discrete notion of time. A third form of a model imagines mutations occurring according to a continuous flow of time.

A Continuous-time version

Considered on a fine enough time scale, it is most natural to view mutations as occurring at discrete times corresponding to generations of the evolving organism. However, since the span of a generation is usually quite small relative to evolutionary time scales, it is also common to use continuous time. We sketch this for those who have an appropriate background in differential equations, or are willing to accept a few mathematical facts.

In this formulation, we imagine that there are certain *rates* at which the various types of mutations occur, and we organize them into a matrix such as

$$Q = \begin{pmatrix} q_{RR} & q_{RY} \\ q_{YR} & q_{YY} \end{pmatrix}.$$

Here, for instance q_{RY} denotes the instantaneous rate at which bases R are replaced by Y, and would be measured in units like (substitutions per site)/year. Note that the entries in each row of Q must add to give 0 (unlike the 1 of the discrete time formulation). Off-diagonal entries must be non-negative, so diagonal ones must be non-positive.

Letting $\mathbf{p}_t = (p_R(t)\ p_Y(t))$ denote the distribution vector of purines and pyrimadines at time t, with $t = 0$ for the ancestral sequence, we have a system of differential equations:

$$\begin{aligned} \frac{d}{dt}p_R(t) &= p_R(t)q_{RR} + p_Y(t)q_{YR} \\ \frac{d}{dt}p_Y(t) &= p_R(t)q_{RY} + p_Y(t)q_{YY}. \end{aligned}$$

Expressing this system in matrix form, by letting $\mathbf{p}_t = (p_R(t), p_Y(t))$, yields

$$\frac{d}{dt}\mathbf{p}_t = \mathbf{p}_t Q.$$

(As for the discrete time models, we use row vectors, multiplied by matrices on the right, rather than the notation more common in differential equations courses, which is the transpose.)

Using the initial values given by $\mathbf{p}_0$, we obtain the solution

$$\mathbf{p}_t = \mathbf{p}_0 e^{Qt}.$$

This formula involves the exponential of a matrix, which can be defined by the usual Taylor series formula, where all terms are reinterpreted as matrices, so that for any square matrix A

$$e^A = I + A + \frac{1}{2}A^2 + \frac{1}{3!}A^3 + \frac{1}{4!}A^4 + \dots. \tag{6.1}$$

Provided A can be diagonalized, then writing $A = S\Lambda S^{-1}$ with Λ the diagonal matrix of eigenvalues of A and S the matrix whose columns are the corresponding (right) eigenvectors, we also have (Exercise 18)

$$e^A = Se^{\Lambda}S^{-1}, \tag{6.2}$$

where e^{Λ} is a diagonal matrix with diagonal entries the exponentials of the entries of Λ.

Returning to our model, since $\mathbf{p}_t = \mathbf{p}_0 e^{Qt}$, it is natural to define

$$M(t) = e^{Qt}$$

as the Markov matrix which encodes the substitutions of bases that occur when an amount of time t passes. The entries of $M(t)$ are thus the conditional probabilities of the various mutations being observed as we compare sequences from time 0 to those from time t, so that $M(t)$ is a single matrix describing the mutation along an edge representing a time of length t.

In our terminology, when we refer to a 'Markov matrix', the rows sum to 1, and we are describing a mutation process either along an entire edge, or over a discrete time step. When we refer to a 'rate matrix', the rows sum to 0, and we are describing the instantaneous mutation process. Applying the exponential function to a rate matrix, or a scalar multiple of a rate matrix, gives a Markov matrix.

6.2 Markov Models of DNA Base Substitution on Trees

A general DNA substitution model

We now more carefully define a rather general model of character evolution along a tree. We again have in mind as our character a single site in a DNA sequence, which at each vertex of the tree might be in one of the four states A,G,C,T. We always use this order for the four bases, so that the purines precede pyrimidines, with each in alphabetical order.

Consider a fixed rooted tree T^ρ. Then parameters for the *general Markov model* model on T^ρ consist of the following:

1) A *root distribution vector* $\mathbf{p}_\rho = (p_A, p_G, p_C, p_T)$, where all entries are non-negative and $p_A + p_G + p_C + p_T = 1$. We interpret these entries as giving

the probabilities that an arbitrary site in a DNA sequence at ρ is occupied by the corresponding base, or, equivalently, as the frequencies with which we will observe these bases in a sequence at ρ. (Note the i.i.d. assumption here.)

2) For each edge $e = (u, v)$ of T directed away from ρ, a 4×4 *Markov matrix*, M_e, whose entries are non-negative and whose rows sum to 1. The i, j-entry of M_e we interpret as the conditional probability that if base i occurs at the site in the parent vertex on the edge, then base j occurs at the descendant vertex. Using abbreviated notation such as $p_{ij} = \mathcal{P}(S_v = j \mid S_u = i)$, where S_u, S_v denote sequences at u and v respectively, we let

$$M_e = \begin{pmatrix} p_{AA} & p_{AG} & p_{AC} & p_{AT} \\ p_{GA} & p_{GG} & p_{GC} & p_{GT} \\ p_{CA} & p_{CG} & p_{CC} & p_{CT} \\ p_{TA} & p_{TG} & p_{TC} & p_{TT} \end{pmatrix}.$$

Note that the only mutations this model allows are base substitutions; there are no insertions, deletions, inversions, or anything else.

We will shortly discuss both more restrictive models, where we place additional requirements on these parameters, as well as generalizations.

Finally, since we have fixed the ordering A, G, C, T, and will often need to refer to particular entries of vectors and matrices, it will also be convenient to sometimes use numbers to refer to the bases, with

$$1 = A,\ 2 = G,\ 3 = C,\ 4 = T.$$

A common rate-matrix variant of the model

Sometimes the parameters in 2) above are replaced by the following, in order to specify that the mutation processes along the various edges of the tree have some commonality:

2a) A continuous-time 4×4 rate matrix Q, whose rows add to 0, and whose off-diagonal entries are nonnegative. With

$$Q = \begin{pmatrix} q_{AA} & q_{AG} & q_{AC} & q_{AT} \\ q_{GA} & q_{GG} & q_{GC} & q_{GT} \\ q_{CA} & q_{CG} & q_{CC} & q_{CT} \\ q_{TA} & q_{TG} & q_{TC} & q_{TT} \end{pmatrix},$$

we interpret an entry q_{ij} as the instantaneous rate (per site) at which base i is replaced by base j.

2b) For each edge e of the tree a non-negative scalar length t_e. Then the Markov matrix

$$M_e = M(t_e) = e^{Qt_e}$$

can be interpreted as in 2) above for the edge e, directed away from the root.

Of course for this continuous time version of the model we may also impose additional assumptions on the form of Q and $\mathbf{p}$.

The joint distribution at the leaves

With parameters for a model specified, we can compute probabilities of observing any particular combination of bases in aligned sequences at the leaves of a tree. We focus only on predicting probabilities of observations at the leaves, since that is the only sort of data that we typically can obtain from real organisms. (In practice then, we will *not* know appropriate parameters to use for our model, but will have to somehow infer them from leaf data, inverting the process we are explaining here.)

To see how to compute leaf distributions, we begin with several examples for very simple trees.

A one-edge tree

This case is similar to that discussed in the purine/pyrimidine example above.

If $\mathbf{p}_\rho$ and M_e are the parameters on a one edge tree from root ρ to descendant S1, then

$$\mathbf{p}_1 = \mathbf{p}_\rho M_e$$

gives a vector of probabilities of observing the four bases at any site in the descendant sequence. For instance, from the definition of matrix multiplication, the first entry of $\mathbf{p}_1$, which should refer to the probability of observing an A in the descendant sequence, is

$$p_1 M_e(1,1) + p_2 M_e(2,1) + p_3 M_e(3,1) + p_4 M_e(4,1) = \\ p_A p_{AA} + p_G p_{GA} + p_C p_{CA} + p_T p_{TA},$$

which has the claimed probabilistic interpretation.

A two-edge tree

Suppose now S1, S2 are two children of a root ρ, with M_i the Markov matrix on the edge leading to Si, as in Figure 6.1.

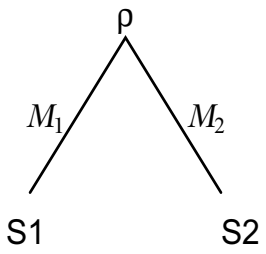

Figure 6.1: A two-edge tree

Then, if we ignore S2, we can compute the probabilities of the various bases appearing in S1 by the product $\mathbf{p}_\rho M_1$, and similarly for S2. But what is the

joint probability that at a particular site we observe base i in S1, and base j at S2?

For this observation to be produced, we might have any base k at ρ, but then this k must become an i in S1 and a j in S2. For a particular k, this means the probability is

$$p_k M_1(k,i) M_2(k,j).$$

But since k could be anything, we need to sum over the possibilities to get

$$\sum_{k=1}^{4} p_k M_1(k,i) M_2(k,j) = p_1 M_1(1,i) M_2(1,j) + p_2 M_1(2,i) M_2(2,j) + p_3 M_1(3,i) M_2(3,j) + p_4 M_1(4,i) M_2(4,j).$$

This can be more succinctly expressed as a matrix equation as follows. Let P be the 4×4 matrix whose entries give the joint distribution of bases at the leaves, so that $P(i,j)$ is the probability in any site of observing a base i at S1 and base j at S2. Then

$$P = M_1^T \operatorname{diag}(\mathbf{p}_\rho) M_2. \tag{6.3}$$

Here $\operatorname{diag}(\mathbf{v})$ denotes a diagonal matrix with the entries of v on the diagonal. We leave checking this claim as Exercise 6.

We should also point out that once all the entries of any joint distribution P are computed, it is easy to recover the distribution of bases at a single leaf, by summing over the other indices, a process usually called *marginalization* in statistics. For instance, the probabilities of observing the base j at S2, without regard to what appears at S1 is found by summing over the index corresponding to S1, giving

$$\sum_{i=1}^{4} P(i,j).$$

That this agrees with the jth entry of $\mathbf{p}_\rho M_2$, which was our earlier claim, is Exercise 8.

A many-edge tree

Suppose now T^ρ has many edges, such as in Figure 6.2

In general, we will not be able to give a simple formula for the joint distribution of bases at the leaves of the tree using standard matrix notation. Indeed, we should not expect to, because the joint distribution itself will be a multi-dimensional array (which we'll call a *tensor*) of more than 2 dimensions. For instance, in the tree above, since there are 3 leaves, the joint distribution tensor P will be a $4 \times 4 \times 4$ array, with the entry $P(i,j,k)$ giving the probabilities that a site has bases i, j, k at the leaves $S1, S2, S3$, respectively. Using slightly different terminology, $P(i,j,k)$ gives the probability of observing the *pattern* ijk at a site in aligned sequences for $S1, S2, S3$.

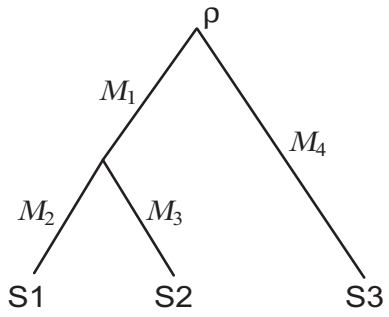

Figure 6.2: A small, many-edged tree

A formula for any particular entry $P(i, j, k)$ of the joint distribution tensor is easily given, however, by following the same sort of reasoning as for the two-edge tree. We simply imagine each possible assignments of bases to all internal nodes of the tree, and then write a product expression for the probability of this particular evolutionary history. Afterwards, we sum over all such assignments, since these are all the distinct ways of obtaining the pattern we are interested in. For example, for the tree above we have

$$P(3, 1, 1) = \sum_{i=1}^{4} \sum_{j=1}^{4} p_i M_1(i, j) M_2(j, 3) M_3(j, 1) M_4(i, 1) \tag{6.4}$$

as one of the 64 entries of P. The other 63 entries are given by quite similar polynomial formulas.

For a tree relating more taxa, of course the formulas get more complicated, but the idea is clear. We should emphasize, though, that the formulas for the entries of P depend not just on the number of taxa, but also on the particular tree topology. Indeed, until we fix a tree topology to consider, we can't even relate Markov matrices to the particular edges.

Assumptions of the models

While we've been calling the matrices of conditional probabilities Markov matrices, we should say a little more about this terminology.

A probabilistic model of a system is called a *Markov model* if it incorporates an assumption of the following sort:

> What happens to the system over any given time step depends only on the state the system is in at the beginning of that step. The earlier history of the system can affect what happens only through having affected the state the system is currently in.

More formally, in a Markov model the probability that a particular state change occurs given the system is in state i is the same as the probability

of the same change, given any entire earlier history of states ending in state i. In particular, there can be no 'memory' of what state changes might have occurred during earlier times that has any effect on future changes. We say the probabilities of state changes are *independent* of the past history.

While it is certainly reasonable to believe that molecular evolution processes satisfy a Markov assumption, we should at least be explicit that it is an assumption.

In our DNA model we are also assuming that each site in the sequence behaves identically, and independently of every other site (i.i.d). We used these assumptions in our introductory example in order to find the various probabilities we needed from our sequence data, by thinking of each site as an independent trial of the same probabilistic process. We will continue to do this with our more elaborate models.

The i.i.d assumption is in fact *not* very reasonable for DNA in some genes. For instance, since the genetic code allows for many changes in the third site of each codon that have no affect on the product of the gene, one could argue that substitutions in the third sites might be more likely than in the first two sites, violating the assumption that each site behaves identically. Also, since genes may lead to the production of proteins which are part of life's processes, the chance of change at one site may well be tied to changes at another, through the need for the gene product to have a particular form. This too violates the assumption of independence. Particular examples of this probably arise from the *secondary structure* of RNA formed by the transcription of genes. In the single-stranded RNA sequence, some bases form bonds to bases further down the sequence, causing particular three-dimensional configurations of the molecule. This means that mutations at one site may well be tied to mutations at sites that are not even nearby in the sequence.

It's easy to come up with a host of other problems with the i.i.d. assumption: If sequences encode both coding and noncoding regions, should they really evolve the same way? Might some genes tend to mutate faster than others? Might not the bases in some sites be so crucial to a sequence's functioning that they are effectively prevented from mutating? While modifications to the basic modeling framework can at least partially address some of these issues, some relic of an i.i.d assumption must always remain if we are to interpret data sequences as giving many trials of a process. This seems to be crucial for a statistical approach to phylogenetics.

Properties of Markov matrices

A Markov matrix (or a stochastic matrix) is a matrix with non-negative entries whose rows add to 1. There are quite a number of theorems concerning Markov matrices that are useful to know about, though we won't go into the proofs. Two that are relevant are:

Theorem 17. A Markov matrix always has $\lambda_1 = 1$ as its largest eigenvalue, and has all eigenvalues satisfying $\mid \lambda \mid \leq 1$. The eigenvector corresponding to λ_1

has all non-negative entries.

Unfortunately, this doesn't rule out having several different eigenvectors with eigenvalue 1. However, there is also:

Theorem 18. A Markov matrix all of whose entries are positive (*i.e.*, non-zero) always has 1 as a strictly largest eigenvalue. There will be only one eigenvector (up to scalar multiplication) associated to $\lambda_1 = 1$.

6.3 Jukes-Cantor and Kimura Models

There are a few special Markov models of base substitutions used for DNA sequences which we can analyze very thoroughly. We present a few of these, first in discrete time formulations, and then in continuous time ones.

The Jukes-Cantor model

The simplest Markov model of base substitution, the Jukes-Cantor model, adds several additional assumptions to the general Markov model.

First, it assumes the root distribution vector describes all bases occurring with equal probability in the ancestral sequence. Thus

$$\mathbf{p}_\rho = \begin{pmatrix} 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix}.$$

Second, for every fixed edge we consider, the conditional probabilities describing an observable base substitution from any base to any other base are all the same. Thus all possible substitutions are equally likely; $A \leftrightarrow T$, $A \leftrightarrow C$, $A \leftrightarrow G$, $C \leftrightarrow T$, $C \leftrightarrow G$, and $T \leftrightarrow G$ have exactly the same chance of occurring. If we let $\frac{a}{3}$ denote the conditional probability of a base substitution of any type occurring, then for a parent S0 and child S1,

$$\mathcal{P}(\mathrm{S}_1 = i \mid \mathrm{S}_0 = j) = \frac{a}{3}, \text{ for all } i \neq j,$$

and so the 12 off-diagonal entries of the matrix M will all be $\frac{a}{3}$. The fact that rows must sum to 1 then determines the diagonal entries of M.

Therefore, for the Jukes-Cantor model, we use the transition matrices of the form

$$M = \begin{pmatrix} 1-a & a/3 & a/3 & a/3 \\ a/3 & 1-a & a/3 & a/3 \\ a/3 & a/3 & 1-a & a/3 \\ a/3 & a/3 & a/3 & 1-a \end{pmatrix}.$$

We are likely, of course, to use a different value of a for each edge of the tree, with larger a values denoting more mutation occurring on that edge.

The Jukes-Cantor model implies a *stable base distribution* at all vertices of the tree. To see this, we simply compute

$$\begin{aligned}\mathbf{p}_\rho M &= \begin{pmatrix}1/4 & 1/4 & 1/4 & 1/4\end{pmatrix}\begin{pmatrix}1-a & a/3 & a/3 & a/3\\ a/3 & 1-a & a/3 & a/3\\ a/3 & a/3 & 1-a & a/3\\ a/3 & a/3 & a/3 & 1-a\end{pmatrix}\\ &= \begin{pmatrix}1/4 & 1/4 & 1/4 & 1/4\end{pmatrix}.\end{aligned}$$

Thus we find the base composition of the sequence doesn't change as evolution proceeds along any edge under the Jukes-Cantor model. In the language of linear algebra, we'd say that the vector $(1/4, 1/4, 1/4, 1/4)$ is a (left) eigenvector of M with eigenvalue 1. (In fact, it's the one promised by the two theorems above on Markov matrices.)

We've given a discrete time formulation of the Jukes-Cantor model, so now we'll deduce a continuous time version *from* it. This is opposite from the direction of deduction given earlier in this chapter, so it is instructive to work through.

Our first step is to diagonalize the matrix M above, and so we first find all its eigenvectors. This can be done by means of the standard approach in linear algebra courses, finding roots of the characteristic equation (which fortunately is easily solved in $\mathbb{Q}(a)$). However, the structure of M is simple enough that we can also just make an intelligent guess.

We've already seen one left eigenvector (the stable base distribution) and its eigenvalue, but a full set is the row vectors and corresponding eignevalues

$$\begin{aligned}\mathbf{v}_1 &= \begin{pmatrix}1 & 1 & 1 & 1\end{pmatrix}, & \lambda_1 &= 1,\\ \mathbf{v}_2 &= \begin{pmatrix}1 & -1 & 1 & -1\end{pmatrix}, & \lambda_2 &= 1-\frac{4}{3}a,\\ \mathbf{v}_3 &= \begin{pmatrix}1 & 1 & -1 & -1\end{pmatrix}, & \lambda_3 &= 1-\frac{4}{3}a,\\ \mathbf{v}_4 &= \begin{pmatrix}1 & -1 & -1 & 1\end{pmatrix}, & \lambda_4 &= 1-\frac{4}{3}a.\end{aligned}$$

Note that while we are considering left eigenvectors here, the fact that M is symmetric means they are right eigenvectors as well.

Also notice that the eigenvectors for the Jukes-Cantor model do not depend on the value of the parameter a, though the eigenvalues do.

We thus have $M = S^{-1}\Lambda S$, where

$$S = \begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{pmatrix}, \quad \Lambda = \operatorname{diag}\left(1,\, 1-\frac{4}{3}a,\, 1-\frac{4}{3}a,\, 1-\frac{4}{3}a\right).$$

The matrix S is quite special, and is sometimes called a *Hadamard matrix*. Obviously S is symmetric, but it is also convenient to observe that S is, up to rescaling, an orthogonal matrix: $SS^T = SS = 4I$, so $S^{-1} = \frac{1}{4}S$.

Motivated by this expression for M, to find an appropriate rate matrix, for $\alpha > 0$ define

$$Q = S^{-1} \operatorname{diag}\left(0, -\frac{4}{3}\alpha, -\frac{4}{3}\alpha, -\frac{4}{3}\alpha\right) S, \tag{6.5}$$

then

$$e^{Qt} = S^{-1} \operatorname{diag}(1, e^{-\frac{4}{3}\alpha t}, e^{-\frac{4}{3}\alpha t}, e^{-\frac{4}{3}\alpha t}) S. \tag{6.6}$$

Thus to make $e^{Qt} = M$, we choose any α and t so that

$$1 - \frac{4}{3}a = e^{-\frac{4}{3}\alpha t}.$$

Perhaps the simplest is $t = 1$, $\alpha = -\frac{3}{4}\ln(1 - \frac{4}{3}a)$, but any choice with

$$\alpha t = -\frac{3}{4}\ln(1 - \frac{4}{3}a) \tag{6.7}$$

is fine. Other choices simply replace our α with a non-zero scalar multiple, and divide t by the same scalar. This can be interpreted as changing the time scale used along an edge and adjusting the rate matrix accordingly. Indeed, the constant $-\frac{4}{3}$ we've inserted in our definition in equation (6.5) has this effect, and was chosen so that Q has the simple expression

$$Q = \begin{pmatrix} -\alpha & \alpha/3 & \alpha/3 & \alpha/3 \\ \alpha/3 & -\alpha & -\alpha/3 & \alpha/3 \\ \alpha/3 & \alpha/3 & -\alpha & \alpha/3 \\ \alpha/3 & \alpha/3 & \alpha/3 & -\alpha \end{pmatrix}. \tag{6.8}$$

We leave checking this computation to the reader, as Exercise 20.

We will return to equation (6.7) in the next chapter, as αt has an important interpretation. Since α is a mutation rate, measured in units like (number of substitutions per site)/year, and t represents an amount of time, measured in units such as years, the product tells us the number of substitutions that should occur per site over the elapsed time t. While both α and t depend on our choice of units of time, the product has a meaning independent of those units.

Mutation rates such as α for DNA in real organisms are not easily found, since estimating them appears to require both an ancestral and descendant sequence, and knowledge of the evolutionary time separating them. In fact we can avoid finding an ancestral sequence, but do need at least an independent estimate of the divergence time of two descendants from their common ancestor. Various estimates of α place it around 1.1×10^{-9} mutations per site per year for certain sections of chloroplast DNA of maize and barley and around 10^{-8} mutations per site per year for mitochondrial DNA in mammals. The mutation rate for the influenza A virus has been estimated to be as high as .01 mutations per site per year. The rate of mutation is generally found to be a bit lower in coding regions of nuclear DNA than in non-coding DNA.

In reality, the mutation rate may not be constant; it may change with time or with location within the DNA. Certainly over the entire evolution of humans from primordial slime it is unreasonable to think that mutation rates have always been the same. However, for shorter periods of time and for DNA serving a fixed purpose, the molecular clock assumption of a constant mutation rate may be reasonable.

The Kimura models

The Jukes-Cantor model is a *one-parameter* (per edge) model of mutation since it depends on the single parameter a to specify the mutation. (Equivalently, for the continuous time formulation, with an arbitrary choice of α in the rate matrix Q made and fixed, the edge lengths t_e are the single parameters.)

The model can be made more flexible by allowing several parameters per edge. A good example of this is the Kimura 2-parameter model, which allows for different probabilities of transitions and transversions. Imagine that over the course of an edge we have mutation probabilities b for transitions and c for each of the possible transversions. If we assume these probabilities are independent of the initial base, then we are saying the off-diagonal entries of the transition matrix are given by:

$$M = \begin{pmatrix} * & b & c & c \\ b & * & c & c \\ c & c & * & b \\ c & c & b & * \end{pmatrix}.$$

Note the specification of the order A, G, C, T for the bases has been used here.

Since the rows must sum to 1 in the discrete formulation, this means all the diagonal entries must be $1-b-2c$, while in the continuous time formulation the rate matrix has off-diagonal entries in the same pattern, but with rows summing to 0. Thus

$$Q = \begin{pmatrix} * & \beta & \gamma & \gamma \\ \beta & * & \gamma & \gamma \\ \gamma & \gamma & * & \beta \\ \gamma & \gamma & \beta & * \end{pmatrix},$$

with the diagonal entries $-\beta-2\gamma$. Notice that if the probabilities of a transition and each transversion are equal so $\beta = \gamma$, then this model includes the Jukes-Cantor one as a special case with $\alpha = 3\beta = 3\gamma$.

An even more general model is the Kimura 3-parameter model, which assumes a transition matrix of the form

$$M = \begin{pmatrix} * & b & c & d \\ b & * & d & c \\ c & d & * & b \\ d & c & b & * \end{pmatrix},$$

or a rate matrix of the form

$$Q = \begin{pmatrix} * & \beta & \gamma & \delta \\ \beta & * & \delta & \gamma \\ \gamma & \delta & * & \beta \\ \delta & \gamma & \beta & * \end{pmatrix}.$$

By appropriate choice of the parameters, this includes both the Jukes-Cantor and Kimura 2-parameter models as special cases. (While the structure of the Kimura 2-parameter model is suggested by biology, the generalization to the 3-parameter one seems to be entirely for mathematical reasons.)

The Kimura models both assume that the root base distribution vector is

$$\mathbf{p}_\rho = \begin{pmatrix} 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix}.$$

Since this vector is an eigenvector with eigenvalue 1 for both the Kimura 2- and 3-parameter matrices, sequences evolving according to these models have this uniform base distribution at all times.

As you'll see in the Exercises 25 and 26, the work done above relating the discrete and continuous time versions of the Jukes-Cantor model can be performed for the Kimura 3-parameter model as well. In fact, the Kimura models have the same eigenvectors as given above for the Jukes-Cantor model, but with several different eigenvalues. Also note that the fact that the rate matrices and discrete time matrices have such a similar form is special to these models — there is usually not such an obvious relationship between the pattern of entries of a rate matrix Q and the pattern of entries of Markov matrices $M = e^{Qt}$.

A note of caution: When a Kimura model is used on a tree, there are two ways it might be used. It may be that different and unrelated Kimura matrices are used on each edge (giving 2 or 3 parameters per edge), or in a common rate matrix formulation, it may be that one Kimura rate matrix is used for the entire tree with scalar lengths assigned to each edge (giving 1 or 2 parameters for the rate matrix, and 1 additional parameter for each edge.)

6.4 Time-reversible Models

An important feature of the Jukes-Cantor and Kimura models is that they are *time-reversible*. What this means is that given an ancestral and a descendant sequence separated by one edge of the tree, if we reverse the flow of time, interchanging which sequence we view as ancestor and descendant, we would describe evolution by exactly the same model parameters.

To see what this means mathematically, suppose $\mathbf{p}$ is the ancestral base distribution and M the Markov matrix describing the descent. Let P denote the joint distribution of bases in the ancestral and descendant sequences, so $P(i, j)$ gives the probability of an ancestral i and a descendant j appearing at a site. Then since $P(i, j) = p_i M(i, j)$, we have the matrix equation

$$P = \text{diag}(\mathbf{p})M.$$

Now interchanging our view of what sequence is ancestral means interchanging the order of indices, or equivalently transposing P. Thus time reversibility means $P = P^T$, or

$$\operatorname{diag}(\mathbf{p})M = (\operatorname{diag}(\mathbf{p})M)^T = M^T \operatorname{diag}(\mathbf{p}). \tag{6.9}$$

In fact, this equation implies the intuitive fact that time-reversibility requires that $\mathbf{p}$ be a stable base distribution for M. (See Exercise 28.)

That equation (6.9) holds for the Jukes-Cantor and Kimura models is easily checked in the discrete time formulation. However there are many more possible parameters for a time-reversible model.

In the continuous time formulation of a model, for time-reversibility to hold we need

$$\operatorname{diag}(\mathbf{p})Q = Q^T \operatorname{diag}(\mathbf{p}). \tag{6.10}$$

If we let

$$\mathbf{p} = \begin{pmatrix} p_A & p_G & p_C & p_T \end{pmatrix}$$

be the root distribution, then a little work (Exercise 29) shows Q should be of the form

$$Q = \begin{pmatrix} * & p_G\alpha & p_C\beta & p_T\gamma \\ p_A\alpha & * & p_C\delta & p_T\epsilon \\ p_A\beta & p_G\delta & * & p_T\eta \\ p_A\gamma & p_G\epsilon & p_C\eta & * \end{pmatrix}, \tag{6.11}$$

with $\alpha, \beta, \gamma, \delta, \epsilon, \eta \geq 0$ and where the diagonal entries are chosen so rows sum to zero.

A commonly used model by biologists (particularly for maximum likelihood approaches to inference) is the *general time-reversible model* (GTR), which assumes as parameters

1) a common time-reversible rate matrix Q on all edges,

2) scalar edge lengths and

3) a root base distribution that is stable, so that $\mathbf{p}$ is an eigenvector of Q of eigenvalue 0.

Practically, this model has a number of nice features. First time reversibility means we will be able to ignore issues of root location, thereby reducing search time for optimal trees. Having a common rate matrix also reduces the number of parameters considerably, making searches much easier. A common rate matrix also imposes some commonality to the mutation process throughout the tree, which in some circumstances seems biologically reasonable. Moreover, a time-reversible $Q, \mathbf{p}$ give a few more free parameters than a Kimura model, allowing for a better fit to data.

Though it's hard to imagine a justification of this model solely on biological grounds (what biological reason could there be for time-reversibility?), models with more parameters can be intractable for computation with a large number of taxa. There are also statistical issues involved in this choice of model, since it's desirable to use models with fewer parameters, provided of course they capture

key aspects of reality. Keeping the number of parameters from being excessively large avoids overfitting data, and keeps variances of inferred trees lower.

Finally we note that a model need not be time reversible in the above sense to be applicable independent of root location. For instance, the general Markov model has this feature, as Exercise 33 shows.

6.5 Exercises

1. Suppose ancestral and descendant sequences of purines and pyrimidines are

   S0 = RRYRYYRRRRYYYRYYRRYYRYYR

   S1 = RYYRYRRRRRRYYRRYRRYYRYYY

   Use this data to estimate a distribution vector for S0 and a Markov matrix describing the mutation process from S0 to S1.

2. The joint frequencies of purines and pyrimidines in ancestral and descendant sequences S0 and S1 is summarized in Table 6.1. Use this data to estimate a distribution vector for S0 and a Markov matrix describing the mutation process from S0 to S1.

$S1 \diagdown S0$	R	Y
R	183	15
Y	32	211

Table 6.1: Frequencies from site comparisons for a pair of sequences

3. An ancestral DNA sequence of 40 bases was

   CTAGGCTTACGATTACGAGGATCCAAATGGCACCAATGCT,

   but in a descendant it had mutated to

   CTACGCTTACGACAACGAGGATCCGAATGGCACCATTGCT.

   a. Give an initial base distribution vector and a Markov matrix to describe the mutation process.

   b. These sequences were actually produced by a Jukes-Cantor simulation. Is that surprising? Explain. What value would you choose for the Jukes-Cantor parameter a to approximate your matrix by a Jukes-Cantor one?

4. Data from two comparisons of 400-base ancestral and descendant sequences are shown in Table 6.2.

   a. For one of these pairs of sequences a Jukes-Cantor model is appropriate. Which one, and why?

$S_1 \diagdown S_0$	A	G	C	T
A	92	15	2	2
G	13	84	4	4
C	0	1	77	16
T	4	2	14	70

$S_1' \diagdown S_0'$	A	G	C	T
A	90	3	3	2
G	3	79	8	2
C	2	4	96	5
T	5	1	3	94

Table 6.2: Frequencies from 400 site comparisons for two pairs of sequences

b. What model would be appropriate for the other pair of sequences? Explain.

5. The Markov matrices that describe real DNA mutation tend to have their largest entries along the main diagonal in the (1,1), (2,2), (3,3), and (4,4) positions. Why should this be the case?

6. Check that the matrix equation (6.3) is correct.

7. Draw a 'balanced' rooted 4-taxon binary tree with a cherry attached to each edge leading from the root. Give a formula like that in equation (6.4) for the joint distribution of bases at the leaves in terms of parameters of the general Markov model. Do the same for an 'unbalanced' rooted 4-taxon tree.

8. Let u denote a column vector with all entries 1. Explain why for a Markov matrix M that $Mu = u$. Then use this fact to show that summing over the first index of

$$M_1^T \operatorname{diag}(\mathbf{p}_\rho) M_2$$

gives

$$\mathbf{p}_\rho M_2.$$

Interpret this as explaining why the marginalization of a joint distribution of bases at two leaves gives the distribution of bases at one leaf.

9. Although the Jukes-Cantor model assumes $\mathbf{p}_0 = (.25, .25, .25, .25)$, a Jukes-Cantor matrix could describe mutations even with a different $\mathbf{p}_0$. Investigate the behavior of a model using a Jukes-Cantor matrix as you vary $\mathbf{p}_0$ by using a calculator, or MATLAB. For instance, with $a = .03$, and $\mathbf{p}_0 = (.2, .3, .4, .1)$ you might use the MATLAB commands such as

```
a=.03, b=a/3
M=[1-a,b,b,b;b,1-a,b,b;b,b,1-a,b;b,b,b,1-a]
p=[.2 .3 .4 .1]
P=p
for i=1:10
p=p*M
```

```
P=[P; p]
end
plot(P)
```

a. With the value of M and $\mathbf{p}_0$ suggested, do you see $\mathbf{p}_t$ approach the stable distribution? Approximately how many time steps are necessary for all the entries of $\mathbf{p}_t$ to be within .05 of the stable values? within .01?

b. Make several other choices of $\mathbf{p}_0$ and repeat step (a).

c. Using $\mathbf{p}_0 = (.25, .25, .25, .25)$, what do you observe? Why?

d. Using $\mathbf{p}_0 = (0, 1, 0, 0)$ what do you observe? What is the biological meaning of this $\mathbf{p}_0$?

10. Investigate the effect of varying a on the behavior produced by the Jukes-Cantor matrix. Let $\mathbf{p}_0 = (.2, .3, .4, .1)$ and use MATLAB commands such as those in the previous exercise to:

    a. Compare the behavior of the model for $a = .03$ and $a = .06$. For which value of a does the model approach the stable distribution fastest?

    b. Does your observation in part (a) hold for other initial choices of $\mathbf{p}_0$?

    c. Explain in intuitive terms why larger values of a should result in a quicker approach to the stable distribution.

11. Make up a 4×4 Markov matrix M with all positive entries, and an initial $\mathbf{p}_0$. To be biologically realistic, make sure the diagonal entries of M are the largest.

    a. Use a computer to observe that after many time steps $\mathbf{p}_t = \mathbf{p}_0 M^t$ appears to approach some equilibrium. Estimate the equilibrium vector as accurately as you can.

    b. Is your estimate in part (a) a left eigenvector of M with eigenvalue 1? If not, does it appear to be close to having this property?

    c. Use a computer to compute the eigenvectors and eigenvalues of M. (In MATLAB the command `[S D]=eig(M)` computes right eigenvectors, so you will have to apply it to `M'`). Is 1 an eigenvalue? Is your estimate of the equilibrium close to its eigenvector?

    d. Are your computations in part (c) consistent with the two theorems about Markov matrices appearing in the text?

12. Express the Kimura 2-parameter model using a 4×4 matrix, but with the bases in the order A,C,G,T. Is this the same as the matrix in the text? Explain.

13. In MATLAB, type `load seqdata` to read in some simulated sequence data. The three pairs of sequences `s0` and `s1`, `t0` and `t1`, `u0` and `u1`, are simulated ancestor and descendant sequences produced according to three different models. Which one was made according to the Jukes-Cantor model? the

Kimura 2-parameter model? a general Markov model? Explain how you can tell. To easily compare sequences by producing a frequency array, use a command like `compseq(s0,s1)`.

14. Suppose we wish to model molecular evolution not at the level of DNA sequences, but rather at the level of the proteins that genes encode.

    a. Create a simple one-parameter mathematical model (similar to the Jukes-Cantor model) describing the process. You will need to use that there are 20 different amino acids from which proteins are constructed in linear chains.

    b. In this situation, how many parameters would the general Markov model have?

15. The MATLAB program `mutate` can be used to simulate the mutation of a DNA sequence according to a Markov model. It will allow you to specify a 4×4 Markov matrix M and initial base distribution vector $\mathbf{p}_0$, as well as the number of bases you would like in your sequences. (Unfortunately, `mutate` assumes vectors are written on the right of matrices, so you must transpose M and $\mathbf{p}_0$ from the presentation in these notes.)

    a. Use the MATLAB program `mutate` to perform a 10 base simulation for the Jukes-Cantor model with $\alpha = .1$ and $\mathbf{p}_0 = (.25, .25, .25, .25)$. Now imagine that the results of your simulation were two data sequences. Use them to estimate probabilities for an initial base distribution vector and a Markov matrix. (The program `compseq` will be useful for this.) Are your estimates close to what you began with?

    b. Repeat part (a) but using sequences of length 100, and then of length 1000.

    c. The difference between a probabilistic model's description and what actually happens under that model when only a finite number of trials are performed is sometimes called *stochastic error*. What conclusions can you draw from parts (a) and (b) about the stochastic error for short sequences as opposed to long ones?

16. Repeat the last problem, but using your own choice of a 4×4 Markov model and initial base distribution. Are the results similar?

17. Suppose you have compared two sequences S_α and S_β of length 1000 sites and obtained the data in Table 6.3 for the number of sites with each pair of bases.

    a. Assuming S_α is the ancestral sequence, find an initial base distribution $\mathbf{p}_0$ and a Markov matrix M to describe the data. Is your matrix M Jukes-Cantor? Is $\mathbf{p}_0$ a stable distribution for M?

    b. Assuming S_β is the ancestral sequence, find an initial base distribution $\mathbf{p}'_0$ and a Markov matrix M' to describe the data. Is your matrix M' Jukes-Cantor? Is $\mathbf{p}'_0$ a stable distribution for M'?

$S_\beta \diagdown S_\alpha$	A	G	C	T
A	105	25	35	25
G	15	175	35	25
C	15	25	245	25
T	15	25	35	175

Table 6.3: Frequencies of $S_\beta = i$ and $S_\alpha = j$ in 1000 site sequence comparison

You should have found that one of your matrices was Jukes-Cantor and the other was not. This can't happen if both S_α and S_β have base distribution $(.25, .25, .25, .25)$.

18. Assuming $A = S\Lambda S^{-1}$, show that equation (6.1) implies equation (6.2).

19. Show that if the rows of Q sum to 0, then the rows of e^{Qt} sum to 1. (Hint: That the rows of Q sum to 0 is expressible as $Qu = 0$ where u is a column vector with all entries 1.)

20. Check the computation of the Jukes-Cantor rate matrix Q in equation (6.8) from its diagonalization.

21. The formula for e^{Qt} for the Jukes-Cantor model in equation (6.6) can be used to see what happens as $t \to \infty$.

    a. If $\alpha > 0$, what is $\lim_{t\to\infty} e^{-\frac{4}{3}\alpha t}$.

    b. Use this to explain why

$$e^{Qt} \to \begin{pmatrix} .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \end{pmatrix}.$$

    Note that each of the rows of this matrix is the stable distribution.

    c. Why did we exclude $\alpha = 0$ from our analysis?

22. Based on the last problem, one might conjecture that powers of a Markov matrix all of whose entries are positive approach a matrix whose rows are the stable distribution. On a computer, investigate this experimentally by creating a Markov matrix, computing very high powers of it to see if the rows become approximately the same, and then checking whether this row is a left eigenvector with eigenvalue 1 of the original matrix.

23. Show the product of two Jukes-Cantor matrices (discrete time) is again a Jukes-Cantor matrix as follows: Let $M(a_1)$ be the Jukes-Cantor matrix with parameter a_1, and $M(a_2)$ the Jukes-Cantor matrix with parameter a_2. Compute $M(a_1)M(a_2)$ to show it has the form $M(a_3)$. Give a formula for a_3 in terms of a_1 and a_2.

24. Show the product of two Kimura 3-parameter matrices is again a Kimura 3-parameter matrix.

25. Show the Kimura 3-parameter matrices (both Markov and rate) have the same eigenvectors as those given in the text for the Jukes-Cantor matrices. What are the eigenvalues of the Kimura 3-parameter matrices?

26. Use the results of the last problem to find the entries of e^{Qt} where $Q = Q(\beta, \gamma, \delta)$ is the Kimura 3-parameter matrix. Your result should be a discrete time Kimura 3-parameter matrix. Give formulas for the discrete time parameters b, c, d in terms of β, γ, δ, t.

27. The Jukes-Cantor model can be presented in a different form as a 2×2 Markov model. Let q_t represent the fraction of sites that agree between the ancestral sequence and the descendant sequence at time t, and p_t the fraction that differ, so $q_0 = 1$ and $p_0 = 0$. Assume that the instantaneous rate at which base substitutions occurs is α, and that each of the 3 possible base substitutions is equally likely. Then

$$\begin{pmatrix} q'(t) \\ p'(t) \end{pmatrix} = \begin{pmatrix} 1-\alpha & \frac{\alpha}{3} \\ \alpha & 1-\frac{\alpha}{3} \end{pmatrix} \begin{pmatrix} q(t) \\ p(t) \end{pmatrix}, \quad \begin{pmatrix} q(0) \\ p(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

    a. Explain why each entry in the matrix has the value it does. (Observe that $1 - \frac{\alpha}{3} = (1-\alpha) + \frac{2\alpha}{3}$.)

    b. Compute the stable distribution of the model by finding the eigenvector with eigenvalue 1.

    c. Find the other eigenvalue and eigenvector for the matrix.

    d. Use (b) and (c), together with the initial conditions to give a formula for $q(t)$ and $p(t)$ as functions of time.

28. Show equation (6.9) implies that $\mathbf{p}$ is a stable base distribution for M.

29. Show that a time reversible rate matrix Q can be expressed by the formula (6.11).

30. Suppose Q is a rate matrix.

    a) Show that if $\mathbf{p}Q = \lambda\mathbf{p}$ and $M(t) = e^{Qt}$ for some t, then $\mathbf{p}M(t) = e^{\lambda t}\mathbf{p}$.

    b) From a) deduce that if $\mathbf{p}Q = 0$, then $\mathbf{p}$ is a stable distribution for all $M(t)$.

31. Suppose Q is a time-reversible rate matrix with stable base distribution $\mathbf{p}$.

    a. Explain why replacing Q with any positive scalar multiple cQ can lead to exactly the same joint distributions on any tree, if edge lengths are adjusted appropriately. Why is this change equivalent to using a new time scale?

    b. In light of part (a), it is sometimes convenient to choose a specific normalization of Q. Explain why $-\operatorname{Tr}(\operatorname{diag}(\mathbf{p})Q)$ gives the instantaneous rate of substitutions for the model, and why we can always rescale Q so this is 1. Here $\operatorname{Tr}(M) = \sum_{i=1}^{n} M_{i,i}$ denotes the trace of a matrix M.

32. Show that a time-reversible Markov matrix M with a stable distribution vector $\mathbf{p}$ which has all positive entries must have a full set of real eigenvalues and eigenvectors. (Hint: Consider $\text{diag}(\mathbf{p})^{1/2} M \,\text{diag}(\mathbf{p})^{-1/2}$.)

33. The general Markov model is not time reversible, but has a related weaker property.

    a. Show it is not time reversible by giving a specific $\mathbf{p}$, M that do not satisfy equation (6.9).

    b. Show that with mild conditions on $\mathbf{p}$, M, there exist $\tilde{\mathbf{p}}$, $\tilde{M}$ so that

    $$\text{diag}(\mathbf{p}) M = \tilde{M}^T \,\text{diag}(\tilde{\mathbf{p}}).$$

    Thus, by changing parameter values for the general Markov model, we can change our viewpoint as to what is ancestral and what is descendant.

    c. Explain the connection between part (b) and Bayes Theorem.

Chapter 7

Phylogenetic Distances

With explicit model of DNA mutation in hand, we can now develop better dissimilarity measures than the Hamming one. The key point here is that the models enable us to estimate how much hidden mutation might have occurred. We will then be able to use a measure of total mutation, rather than just observed mutations, as a measure of dissimilarity. These improved measures might then be used in an algorithm such as Neighbor Joining to produce a tree.

In biological literature, the Hamming dissimilarity is sometimes called the *uncorrected* distance or p-distance, while the ones we will develop from explicit models are called *corrected* distances.

7.1 Jukes-Cantor Distance

To frame the issue we want to address more clearly, let's consider the Jukes-Cantor model, which is in many ways the simplest. We imagine sequence mutation is governed by a Jukes-Cantor rate matrix

$$Q = Q(\alpha) = \begin{pmatrix} -\alpha & \alpha/3 & \alpha/3 & \alpha/3 \\ \alpha/3 & -\alpha & \alpha/3 & \alpha/3 \\ \alpha/3 & \alpha/3 & -\alpha & \alpha/3 \\ \alpha/3 & \alpha/3 & \alpha/3 & -\alpha \end{pmatrix}.$$

Recall that $\alpha/3$ can be interpreted as the instantaneous rate at which a particular base mutates into a different particular base. However, since this rate is the same for all choices of different basses, we can also say α is the rate at which any given base mutates into a different base.

As we saw in the last chapter, if an amount of time t passes, then the total mutation process over that elapsed time can be described by

$$M = e^{Q(\alpha)t} = e^{Q(\alpha t)} = \begin{pmatrix} 1-a & a/3 & a/3 & a/3 \\ a/3 & 1-a & a/3 & a/3 \\ a/3 & a/3 & 1-a & a/3 \\ a/3 & a/3 & a/3 & 1-a \end{pmatrix},$$

where a and αt are related by equation (6.7), which we recall was

$$\alpha t = -\frac{3}{4}\ln\left(1 - \frac{4}{3}a\right).$$

But the expression αt has a simple interpretation. It is the product of a rate measured, say, in units of substitutions per site per year, and an elapsed time measured, say, in years. Therefore αt is the total number of substitutions per site that occur during the full time period, *including all those that are hidden due to multiple mutations at that site.* While αt is not something we can directly observe by comparing initial and final sequences, it is the correct measure of the total amount of mutation that occurred.

Now the diagonal entries of M give conditional probabilities that the base at time t is the same as the base at time 0. Since all the diagonal entries are the same, we can thus interpret $1-a$ as the probability of observing no change when the site at time 0 is compared to the one at time t, and a itself is the probability of observing a change after time t has elapsed. Assuming we have an ancestral sequence S0 and a descendant sequence S1 to compare, we can directly estimate a by comparing the sequences. In fact, a reasonable estimate is just

$$\hat{a} = \frac{\text{no. of sites that changed}}{\text{total no. of sites}},$$

the Hamming dissimilarity measure we introduced in Chapter 5.

We therefore define the *Jukes-Cantor distance* between DNA sequences S_0 and S_1 as

$$d_{JC}(\mathrm{S}_0, \mathrm{S}_1) = -\frac{3}{4}\ln\left(1 - \frac{4}{3}\hat{a}\right),$$

where $\hat{a}$ is the fraction of sites that disagree in comparing S_0 to S_1. Provided the Jukes-Cantor model accurately describes the evolution of one sequence into another, this distance is an estimate of the total number of substitutions per site that occurred during the evolution.

Example. Suppose an ancestral sequence $ATTGAC$ has evolved into a descendant sequence $ATGGCC$. (For real data, we'd use much longer sequences with hundreds of sites, or have little confidence in our work.) We estimate $\hat{a} = 2/6 \approx .333$, so that on average we observe 1/3 of a substitution per site when we compare the sequences.

Then

$$d_{JC}(S0, S1) = -\frac{3}{4}\ln\left(1 - \left(\frac{4}{3}\right)\left(\frac{2}{6}\right)\right) = .4408.$$

Note that the Jukes-Cantor distance is larger than $\hat{a}$, as it should be since it accounts for hidden mutations as well. We have estimated that, on average, there were actually about .4408 substitutions of bases in each site during the evolution of S0 to S1.

In practice, biologists seldom have ancestral and descendant sequences to compare. Typically only the sequences at the leaves of a tree, from currently

living taxa, are available. However, the Jukes-Cantor model was time reversible, so this does not matter. Mathematically, we can view any leaf as the root of the tree for a Jukes-Cantor model, and compute the distances between this leaf/root and any other leaf. We then interpret the distance as a measure of the total mutation that occurred along all edges in the path joining them.

For 'perfect' data from a Jukes-Cantor model, the Jukes-Cantor distance will be a dissimilarity measure on the leaves that is the restriction of a tree metric: Each edge e of the tree has length αt_e, where $M_e = e^{Q(\alpha)t_e}$. While this is already clear from the interpretation of αt as total amount of mutation, it is instructive to show more formally (as in the Exercise 8) that 'perfect' values of d_{JC} come from a tree metric.

If we assume a molecular clock, so the same mutation rate α applies to all edges of the tree, then the Jukes-Cantor distance is directly proportional to elapsed time. This gives us a way to estimate mutation rates, provided we have two descendant sequences S1, S2 from a common ancestor S0 as in Figure 7.1. Provided we can somehow obtain an independent estimate of the time t involved

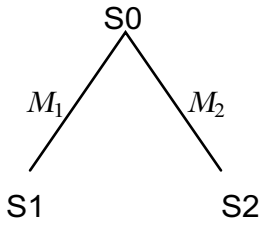

Figure 7.1: Two descendants of an ancestor, assuming a molecular clock

in their descent:

$$\alpha = \frac{\alpha t}{t} \approx \frac{d_{JC}(S0, S1) + d_{JC}(S0, S2)}{2t} = \frac{d_{JC}(S1, S2)}{2t}.$$

To get an independent estimate of the time t of descent is of course not so easy. If the taxa are evolving very rapidly, as for some viruses, this may be directly observable. In other cases, estimates of time from the fossil record can be used.

To further explore the Jukes-Cantor distance, fix a rate α for the Jukes-Cantor model, and denote by $a(t)$ the probability that we observe a mutation in comparing ancestral and descendant sequences at time t. Then from the work above it's easy to show $a(t) = \frac{3}{4} - \frac{3}{4}e^{-\frac{4}{3}\alpha t}$, which is graphed in Figure 7.2.

We of course see that $a(0) = 0$, since at time $t = 0$ no substitutions have yet occurred. For small values of t, we find $a(t) \approx \alpha t$, so whether we use the Jukes-Cantor distance or simply $\hat{a}$ as a dissimilarity measure has little effect. However, once t is large, $a(t)$ can approach the 'saturation point' of 3/4, where we find 3 out of 4 sites are observed to have changed. Of course that is what we would expect for two randomly chosen unrelated sequences, so if the evolutionary

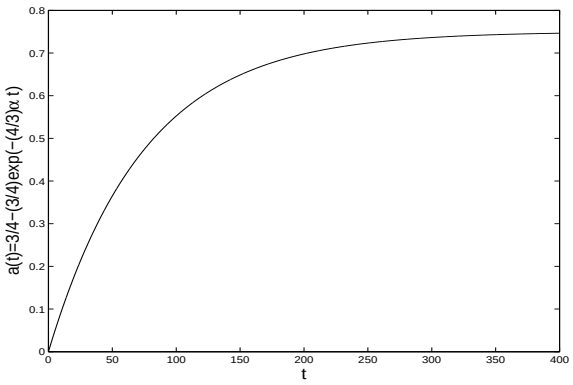

Figure 7.2: The Jukes-Cantor model, $\alpha = .01$: Fraction of differing sites after elapsed time t

history is long enough, we might not be able to see enough connection between the sequences to know they are related. It is really for midsized values of $a(t)$ that the Jukes-Cantor formula offers an improved dissimilarity measure.

Before we leave the Jukes-Cantor model, we note that the justification we've given for the Jukes-Cantor distance, while clearly based on a mathematical model, has shortchanged some statistical issues. In particular, while it is certainly reasonable, we haven't really explained why the estimate $\hat{a}$ is a good one to use for the true but unknown value a in the distance formula. We'll return to this issue when we discuss Maximum Likelihood methods in Chapter 8.

7.2 Kimura Distances

Given any Markov model of base substitutions we could hope to imitate the steps above to derive an appropriate formula reconstructing the amount of mutation that has occurred. For the Kimura models, you'll find an exercise that steps you through the procedure. The final formula for the Kimura 3-parameter model is

$$d_{K3} = -\frac{1}{4}\left(\ln(1-2\hat{b}-2\hat{c}) + \ln(1-2\hat{b}-2\hat{d}) + \ln(1-2\hat{c}-2\hat{d})\right),$$

where $\hat{b}$, $\hat{c}$, and $\hat{d}$ are estimates of parameters for a Kimura 3-parameter matrix describing the mutation of the initial sequence to the final.

Of course, if $\hat{c} = \hat{d}$, this also gives a distance for the Kimura 2-parameter model. In that case, $\hat{b}$ is the probability of a transition, while $\hat{c} + \hat{d} = 2\hat{c}$ is the probability of a transversion. Thus if from sequence data we estimate the probability of a transition $\hat{p}_1$ by counting all transitions and dividing by the length of the sequence, and the probability of a transversion $\hat{p}_2$ similarly, we

have

$$d_{K2} = -\frac{1}{2}\ln(1 - 2\hat{p}_1 - \hat{p}_2) - \frac{1}{4}\ln(1 - 2\hat{p}_2).$$

If sequence data seems to indicate that transitions and each transversion type did not proceed at equal rates, then the Jukes-Cantor model is a poor one, and so the Kimura distance formulas are better choices for estimating the total amount of mutation.

7.3 Log-det Distance

It's possible to develop distance formulas in more general settings, continuing to focus on a 'distance' as a reconstruction of the average number of mutations per site that occurred, including hidden ones. For instance we can derive such a distance for the GTR model, though its formula involves a matrix logarithm. (See Exercise 19.)

However, to preserve this interpretation of distance as a reconstruction of mutations seems to require that the model assume a stable base distribution. Since it is known that base distributions are not always stable for real data, we must take a different approach to developing the idea of a distance in such circumstances. In fact, there is a distance that is consistent with the general Markov model, though it is not one with a simple biological interpretation. Instead, we motivate it mathematically.

We won't focus on reconstructing the total number of base substitutions that occurred, but rather on the properties we need if our distance is to be a restriction of a tree metric.

These are:

1) $d(\mathrm{S0}, \mathrm{S1}) \geq 0$, and $d(\mathrm{S0}, \mathrm{S1}) = 0$ if, and only if, S0 = S1,

2) $d(\mathrm{S0}, \mathrm{S1}) = d(\mathrm{S1}, \mathrm{S0})$,

3) $d(\mathrm{S0}, \mathrm{S2}) = d(\mathrm{S0}, \mathrm{S1}) + d(\mathrm{S1}, \mathrm{S2})$ provided S1 lies at an internal vertex along the path between S0 and S1.

The last of these properties, called *additivity*, is the important one here.

To define such a distance for the general Markov model, suppose F is the 4×4 frequency array obtained by comparing sites in sequences S_0 and S_1. Thus $F(i, j)$ is the fraction of sites with base i in the S0 sequence and j in the S1 sequence. Also let $\mathbf{f}_0$ and $\mathbf{f}_1$ be the frequency vectors for the bases in S0 and S1, respectively. Thus $\mathbf{f}_0$ and $\mathbf{f}_1$ are found by taking row and column sums of F.

Then the *log-det distance* (also called the *paralinear distance*) between S0 and S1 is defined by

$$d_{LD}(\mathrm{S0}, \mathrm{S1}) = -\frac{1}{4}\left(\ln|\det(F)| - \frac{1}{2}\ln(g_0 g_1)\right),$$

where g_i is the product of the 4 entries in $\mathbf{f}_i$.

That this distance satisfies the properties will be shown in Exercise 18.

Unlike the other distance formulas discussed here, the log-det distance usually is not the total number of mutations per site that must have occurred over the evolutionary history. Still, this distance has the formal properties 1), 2), and 3) that allow it to be thought of as some measure of the amount of mutation that has occurred. In special circumstances, such as when the Jukes-Cantor or Kimura models apply exactly, it gives the same result as they do. (See Exercise 16.)

Actually, there is sometimes confusion over the use of the word 'distance' in phylogenetics, since mathematicians tend to think of the word as interchangable with 'metric,' while biologists tend to think of it as meaning an estimate of elapsed time or total mutation. With many models and corresponding distance formulas, there is no conflict in this, but with the log-det distance for the general Markov model there is. However, for data that exhibits changing base distributions through the tree, the log-det distance is the only formula this is applicable.

Finally, if a Jukes-Cantor, Kimura, or GTR model is adequate for describing sequence data, it is better to use distance formulas designed for the most restrictive yet adequate model. The use of a more general model allows 'overfitting' of the data, making statistical infererence of the amount of mutation that occurred less reliable. Since distances computed from data will not exactly fit a metric tree, getting values closer to the 'true' distances by using the most restrictive model that is appropriate will improve our chance of finding the 'true' tree.

7.4 Exercises

1. Calculate $d_{JC}(S_0, S_1)$ for the two 40 base sequences

   S_0 : `CTAGGCTTACGATTACGAGGATCCAAATGGCACCAATGCT`
   S_1 : `CTACGCTTACGACAACGAGGATCCGAATGGCACCATTGCT`.

2. Ancestral and descendant sequences of 400 bases were simulated according to the Jukes-Cantor model. A comparison of aligned sites gave the frequency data in Table 7.1.

$S_1 \diagdown S_0$	A	G	C	T
A	90	3	3	2
G	3	79	8	2
C	2	4	96	5
T	5	1	3	94

Table 7.1: Frequencies of $S_1 = i$ and $S_0 = j$ in 400 site sequence comparison

   a. Compute the Jukes-Cantor distance to ten decimal digits, showing all steps.

b. Compute the Kimura 2-parameter distance to ten decimal digits, showing all steps.

c. Are the answers to (a) and (b) identical? Explain.

3. Ancestral and descendant sequences of 400 bases were simulated according to the Kimura 2-parameter model with $\gamma = \beta/5$. A comparison of aligned sites gave the frequency data in Table 7.2.

$S_1 \diagdown S_0$	A	G	C	T
A	92	15	2	2
G	13	84	4	4
C	0	1	77	16
T	4	2	14	70

Table 7.2: Frequencies of $S_1 = i$ and $S_0 = j$ in 400 site sequence comparison

a. Compute the Jukes-Cantor distance to ten decimal digits, showing all steps.

b. Compute the Kimura 2-parameter distance to ten decimal digits, showing all steps.

c. Which of these is likely to be a better estimate of the number of substitutions per site that actually occurred? Explain.

4. Compute the Kimura 3-parameter and log-det (paralinear) distance for the sequences of the last two problems.

5. Graph d_{JC} as a function of $\hat{a}$.

a. Why does $d_{JC} = 0$ if two sequences are identical?

b. Why does d_{JC} not make sense if two sequences differ in 3/4 or more of the sites? Should this cause problems when trying to use the formula on real data?

c. Explain in biological terms why if two sequences differ in just under 3/4 of the sites, the value of d_{JC} should be very large.

6. The Jukes-Cantor distance formula is sometimes stated as

$$d_{JC} = -\frac{3}{4} \ln \left(\frac{4q - 1}{3} \right),$$

where q is the proportion of bases that are the same in the 'before' and 'after' sequences. Derive this formula from the one in the text.

7. When transitions are more frequent than transversions, the Kimura 2-parameter distance often gives a larger value than the Jukes-Cantor distance. Explain this informally by explaining why hidden mutations are more likely under this circumstance.

8. Suppose that the Jukes-Cantor model perfectly describes sequence evolution along a metric tree T with positive edge lengths. The rate parameter $\alpha = 1$ is fixed in the Jukes-Cantor rate matrix Q. Each edge e_i has length t_i, with associated Markov matrix $M_i = e^{Qt_i}$.

   a) Suppose a minimal path from taxon S0 to taxon S1 in a tree is composed of edges of length $t_1, t_2, \ldots, t_n$. Show that if sequence data is exactly described by the distribution the Jukes-Cantor model predicts, then $d_{JC}(S_0, S_1) = \sum_{i=1}^{n} t_i$, and thus that the Jukes-Cantor distance agrees with the tree metric.

   b) What if the rate parameter α is some number other than 1? How will the tree metric and the Jukes-Cantor distance between taxa be related?

9. Show that the formula for the Jukes-Cantor distance can be recovered from the formula for the Kimura 3-parameter distance by letting b, c, and d all be $a/3$.

10. Use the MATLAB program `mutate` to simulate a sequence evolving according to the Jukes-Cantor model for $t = 400$ time steps, using a matrix with parameter $a = .001$ for each time step. Compute a frequency array of base combinations with `F=compseq(Sinit,Sfinal)` and then compute the Jukes-Cantor distance with `distJC(F)`. Is the computed distance approximately $\alpha t = .4$? Explain why or why not.

11. In MATLAB, type `load seqdata` to read in some simulated sequence data. Type `who` to see the names of the things you just loaded.

    a. Compute all six Jukes-Cantor distances between the sequences $a1$, $a2$, $a3$, and $a4$. You can compute a frequency array for base combinations with `F=compseq(a1,a2)` and then compute the distance with `distJC(F)`.

    b. Suppose these sequences came from currently living species whose evolutionary relationships we would like to deduce. Use Neighbor Joining with these distances to produce a phylogenetic tree for them.

12. Use the Jukes-Cantor distance and the Neighbor Joining program `nj` to construct trees for the following simulated sequence data in `seqdata.mat`. Compare your results to that produced by UPGMA. How has whether a molecular clock operated in the simulation affected the results?

    a. `a1`, `a2`, `a3`, and `a4` (molecular clock)

    b. `b1`, `b2`, `b3`, `b4`, and `b5` (no molecular clock)

13. The sequences `c1`, `c2`, `c3`, `c4`, and `c5` in `seqdata.mat` were simulated using a Kimura 2-parameter model.

    a. Even without knowing what model was used, how might comparing some of these sequences suggested that the Kimura 2-parameter distance was a good choice for these sequences?

    b. Construct the NJ tree using the Kimura 2-parameter distance.

c. Does your tree roughly support a molecular clock hypothesis? Explain.

14. The sequences `d1`, `d2`, `d3`, `d4`, `d5`, and `d6` are in `seqdata.mat`.

    a. Choose a distance formula to use for these sequences and explain why your choice is reasonable.

    b. Construct a NJ tree from the data.

    c. One of the 6 taxa is an outgroup that was included in order to provide a rooted tree for the others. Which one is the outgroup? Draw the rooted metric tree relating only the other taxa.

15. Derive the formula for the Kimura 3-parameter distance. Refer to Exercise 26 of the Chapter 6, in which you found formulas for the parameters b, c, and d in $M = e^{Qt}$ in terms of β, γ, δ, t.

16. The goal of this problem is to show that the Jukes-Cantor distance is a special case of the log-det distance. You will need to know the following two facts about determinants of $k \times k$ matrices that are proved in a Linear Algebra course:

    i) $\det(cA) = c^k \det(A)$.

    ii) $\det(A) =$ the product of A's k eigenvalues.

    a. Suppose two sequences S_0 and S_1 of length N were compared and the frequency table F was found to be *exactly* described by a Jukes-Cantor matrix $M(\alpha)$ with base distribution $(1/4, 1/4, 1/4, 1/4)$ for S_0. Show that $F = \frac{1}{4}M(\alpha)$.

    b. Explain why $\mathbf{f}_1 = \mathbf{f}_2 = (1/4, 1/4, 1/4, 1/4)$.

    c. Use the facts above to show that in this case $d_{LD}(S_0, S_1) = d_{JC}(S_0, S_1)$.

17. Proceeding as in the last problem, show that the Kimura 3-parameter distance is a special case of the log-det distance.

18. Show the log-det distance formula is additive and symmetric through the following steps. You will need to know the following three facts about determinants of $k \times k$ matrices that are proved in a Linear Algebra course:

    i) $\det(AB) = \det(A)\det(B)$.

    ii) If the (i, j)-entries of D are all zero for $i \neq j$, then

    $$\det(D) = D(1,1) \cdot D(2,2) \cdots D(k,k).$$

    iii) $\det(A^T) = \det(A)$, where A^T, the *transpose* of A, is a matrix whose (i, j)-entry is the (j, i)-entry of A.

    a. Suppose S0 is the parent of S1 which is the parent of S2, the initial base distribution for S0 is $\mathbf{p}_0$, and the Markov matrices describing mutations are $M_{0\to1}$ and $M_{1\to2}$, respectively. Let $M_{0\to2} = M_{0\to1}M_{1\to2}$. Explain why

$\mathbf{p}_1 = \mathbf{p}_0 M_{0\to 1}$ and $\mathbf{p}_2 = \mathbf{p}_1 M_{1\to 2}$ are the base distributions in S1 and S2 respectively, and explain the meaning of $M_{0\to 2}$.

b. For the vector $\mathbf{p}_i = (a, b, c, d)$ let

$$D_i = \begin{pmatrix} \sqrt{a} & 0 & 0 & 0 \\ 0 & \sqrt{b} & 0 & 0 \\ 0 & 0 & \sqrt{c} & 0 \\ 0 & 0 & 0 & \sqrt{d} \end{pmatrix}.$$

Then for each pair i, j with $0 \le i < j \le 2$, define the matrix

$$N_{i\to j} = D_i M_{i\to j} D_j^{-1}.$$

Show $N_{0\to 1} N_{1\to 2} = N_{0\to 2}$, and use fact (i) to conclude

$$\ln|\det(N_{0\to 1})| + \ln|\det(N_{1\to 2})| = \ln|\det(N_{0\to 2})|.$$

c. Show the frequency array for comparing Si to Sj is $F_{i\to j} = D_i N_{i\to j} D_j$, and then use fact (i) to show

$$\ln|\det(F_{i\to j})| = \ln|\det(N_{i\to j})| + \ln|\det(D_i)| + \ln|\det(D_j)|.$$

d. Combine (b), (c), and fact (ii) to show the log-det distance is additive.

e. Explain why $F_{j\to i} = F_{i\to j}^T$, and then use fact (iii) to show the log-det distance is symmetric.

19. Suppose Q is a time-reversible rate matrix with stable base distribution $\mathbf{p}$.

a. Explain why $-\operatorname{Tr}(\operatorname{diag}(\mathbf{p})Q)$ gives the instantaneous rate of substitutions. Here $\operatorname{Tr}(A) = \sum_{i=1}^n A_{i,i}$ denotes the trace of a matrix A.

b. Since for a one-edge tree of length t this model predicts a joint distribution matrix $P = \operatorname{diag}(\mathbf{p})e^{Qt}$, explain why $-\operatorname{Tr}(\operatorname{diag}(\mathbf{p})\ln(\operatorname{diag}(\mathbf{p})^{-1}P)$ gives the total number of substitutions per site on the edge. This is the appropriate phylogenetic distance for such a model.

Chapter 8

Maximum Likelihood

Perhaps the most common currently-used approach to phylogenetic inference is Maximum Likelihood (ML). It is a model-based method that is well-grounded in standard statistical viewpoints. However, it is also computationally intensive, so that other methods, such as Neighbor Joining with an appropriate choice of a model-based distance, might be used for an initial construction of a 'pretty good' tree, which then serves as a starting point for a heuristic search for a maximum likelihood solution.

Since we are not assuming a strong statistics background — and in fact standard undergraduate statistics courses often say little about maximum likelihood as a general framework for statistical inference — we will provide a bit of informal background on Maximum Likelihood in general.

8.1 Probabilities and Likelihoods

Let's step back from phylogenetics and frame a very general problem in statistics.

Suppose we have a probabilistic model that we believe predicts outcomes in some experiment. Our model depends on one or more *parameters*, which are typically numerical quantities. However, we do not know what values of these parameters are appropriate to fit our data. How can we infer a 'best' choice of parameter values from experimental data?

We'll give two examples of such a problem, the first as simple as possible, the second a bit more complex, in order to motivate the ML approach to addressing such an issue.

Example (1). Our experiment is the toss of a (possibly unfair) coin. Our model is simply that the toss will produce heads with probability p and tails with probability $1 - p$. The parameter here is p, which might have any numerical value from 0 to 1.

Suppose we conducted many trials of the experiment in Example (1). While

we either know or assume our model applies to the experiment, we do not know what the value of the parameter p is. How should we estimate p?

For Example (1), we do not need to be very sophisticated because the model is so simple. Using a frequentist interpretation of probability, p simply expresses what fraction of a large number of trials we believe will produce heads. So if, for instance, out of 100 trials we record 37 heads, we estimate $\hat{p} = 37/100 = .37$. In general, if we find m heads out of n trials, we estimate $\hat{p} = m/n$.

But there is another way we can arrive at the same result. Naively, we might decide the best estimate for p would be the numerical value that is most probable, given that we obtained the specific data we did. That is, we'd like to choose our estimate $\hat{p}$ to make as large as possible the conditional probability that $p = \hat{p}$ given the data that was observed. That is, we want $\hat{p}$ to be the value of p that maximizes

$$\mathcal{P}(p \mid \text{data}).$$

Unfortunately, it isn't at all clear how to do this, since we have no idea how to compute such a conditional probability. However, we can observe that by formal properties of probabilities

$$\mathcal{P}(p \mid \text{data})\mathcal{P}(\text{data}) = \mathcal{P}(p,\ \text{data}) = \mathcal{P}(\text{data} \mid p)\mathcal{P}(p).$$

This implies the result, which is an instance of *Bayes theorem*,

$$\mathcal{P}(p \mid \text{data}) = \mathcal{P}(\text{data} \mid p)\frac{\mathcal{P}(p)}{\mathcal{P}(\text{data})}.$$

Now the terms $\mathcal{P}(p)$ and $\mathcal{P}(\text{data})$ in the fraction on the right are not things we can address (unless we are willing to claim a prior belief in the distribution for p, and by this become Baysian statisticians). Thus we choose to focus on

$$L(p) = \mathcal{P}(\text{data} \mid p)$$

which is something we can calculate from our model. We call this function L the *likelihood function* for our model and data. Note that while it is a conditional probability, it is *not* the conditional probability we originally were interested in. This is why we call it the likelihood function, to remind us it is most definitely not telling us the probability of any p given the data.

Definition. Given some data presumed to be in accord with a model, the *maximum likelihood estimate* for the set of parameters p for a model is the (unique, we hope) set of values $\hat{p}$ that maximize the likelihood function $L(p)$.

To find the maximum likelihood estimate for p in Example (1), suppose our data is again m heads out of n tosses, in some particular order. Then the likelihood function, which depends on the unknown p, is $L(p) = \mathcal{P}(\text{data} \mid p)$. But, because we assume our tosses are independent, this probability is simply the product of the probabilities of the outcomes of each individual toss. That is,

$$L(p) = \mathcal{P}(\text{data} \mid p) = p^m(1-p)^{n-m}.$$

To find the maximum in the interval $[0, 1]$, we compute

$$\begin{aligned}\frac{d}{dp}p^m(1-p)^{n-m} &= mp^{m-1}(1-p)^{n-m} - (n-m)p^m(1-p)^{n-m-1} \\ &= p^{m-1}(1-p)^{n-m-1}\left(m(1-p) - (n-m)p\right).\end{aligned}$$

But then this is zero when

$$0 = m(1-p) - (n-m)p = m - np.$$

Thus we find $p = m/n$, exactly as before.

In computing maximum likelihood estimates of model parameters, it is common to consider the logarithm of the likelihood function, rather than the likelihood function itself. Of course these two functions are maximized at the same parameter values, so this does not affect our judgment of the best estimates to infer. However, the form of the log-likelihood function often makes the algebra in finding critical points simpler, as in fact it does even for Example (1):

$$\ln L(p) = m \ln p + (n-m)\ln(1-p),$$

so

$$\frac{d}{dp}\ln L(p) = \frac{m}{p} - \frac{n-m}{1-p}.$$

Setting this equal to zero again yields $p = m/n$.

We now consider a more elaborate example.

Example (2). Suppose we have two (possibly unfair) coins, with probabilities of heads p_1, p_2 respectively, giving us two parameters. We toss the first coin, and depending on whether it gives heads or tails, either retain it, or switch to the second coin for a second toss. We then make a second toss and report the (ordered) result of these two tosses as the outcome of the experiment.

Then we find the probability of the various outcomes are

$$p_{hh} = p_1^2, \quad p_{ht} = p_1(1-p_1), \quad p_{th} = (1-p_1)p_2, \quad p_{tt} = (1-p_1)(1-p_2).$$

Now we wish to estimate p_1 and p_2 from some data: Suppose in n trials, we find $n_{hh}, n_{ht}, n_{th}, n_{tt}$ occurrences of the 4 outcomes, where

$$n = n_{hh} + n_{ht} + n_{th} + n_{tt}.$$

Before taking an ML approach, we might hope to just set $p_{ij} = n_{ij}/n$ for each $i, j \in \{h, t\}$ and solve for p_1, p_2. However, this is not likely to work, since it gives 3 independent equations in 2 unknowns. (While at first it appears we have 4 equations, the fact that $\sum_{i,j\in\{h,t\}} p_{ij} = 1$ means one of the equations is implied by the others.) But for a system of 3 equations in only 2 unknowns, we generally do not expect a solution to exist. Indeed, for most data there will be no solutions to this polynomial system.

Taking a maximum likelihood approach to the estimation of p_1, p_2, however, is straightforward. Using an assumption that each of our trials are independent so that we can multiple probabilities, the likelihood function is

$$L(x,y) = (x^2)^{n_1}\,(x(1-x))^{n_2}\,((1-x)y)^{n_3}((1-x)(1-y))^{n_4},$$

so the log-likelihood is

$$\ln L(x,y) = (2n_1+n_2)\log x + (n_2+n_3+n_4)\log(1-x) + n_3 \log y + n_4 \log(1-y).$$

(For readability we've changed our variable names here, with $x = p_1, y = p_2$.) To find critical points we compute partial derivatives and equate them to zero and see

$$0 = \frac{2n_1+n_2}{x} - \frac{n_2+n_3+n_4}{1-x}, \quad 0 = \frac{n_3}{y} - \frac{n_4}{1-y}.$$

Solving for y gives

$$\hat{p}_2 = \frac{n_3}{n_3+n_4}, \tag{8.1}$$

which is a formula we could have guessed, in light of Example (1). However, solving for x gives the less obvious

$$\hat{p}_1 = \frac{2n_1+n_2}{2n_1+2n_2+n_3+n_4}. \tag{8.2}$$

We leave as an exercise checking that for 'perfect' data, where $n_{ij} = np_{ij}$, these formulas recover the correct values $\hat{p}_1 = p_1$, $\hat{p}_2 = p_2$.

We've emphasized here that maximum likelihood estimators give us a (hopefully unique) reasonable way of estimating parameters from data that can be used in many settings. In fact, ML estimators have important statistical properties as well. For instance, it's possible to prove in great generality that maximum likelihood estimators are *consistent*, in the sense that as the number of trials described by the model is increased to infinity, the estimators converge to the true parameters. They also are *efficient*, in that they have minimal variance as the number of trials is increased. Note both of the statements refer to having large amounts of data, though, and say nothing about behavior for small data sets. While ML is quite common throughout statistics and is widely accepted, it is not the only statistical viewpoint and some leading statisticians have expressed discomfort with it.

On a more practical level, there are often difficult computational issues involved in ML. In these examples, we have found maximum likelihood estimators by first computing derivatives of the log-likelihood function, and then solving several simultaneous equations. This last step in particular may be quite difficult for more complicated models. For instance, even if the model is expressed through polynomial equations, we may obtain rational expressions of high degree from the partial derivatives. If the model is given by transcendental formulas, we may be forced to solve transcendental equations. In practice, then, various

numerical approaches to finding maxima may be used. Unfortunately this is often at the price of being sure the global maximum has been located. Numerical searches may become trapped in a local maximum, so some effort must be taken to try to prevent this from happening.

8.2 ML Estimators for One-edge Trees

As a first application of maximum likelihood ideas to phylogenetics, consider a Jukes-Cantor model from an ancestral sequence S0 to a descendant sequence S1 along a one-edge tree. We view this as a discrete time process, so that the Markov matrix along the edge is

$$\begin{pmatrix} 1-a & a/3 & a/3 & a/3 \\ a/3 & 1-a & a/3 & a/3 \\ a/3 & a/3 & 1-a & a/3 \\ a/3 & a/3 & a/3 & 1-a \end{pmatrix},$$

while we have a uniform base distribution $\begin{pmatrix} 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix}$ in the ancestral sequence S0.

Suppose we are interested in estimating the parameter a from aligned sequence data for S0 and S1. We summarize the sequence data by counting how often each pattern appears, and let $N(i,j) = n_{ij}$ be the number of sites with base i in S0 and base j in S1. This N is a 4×4 matrix of data counts.

We have a corresponding joint distribution matrix of probabilities of bases predicted by the model, with

$$\begin{aligned} P = (p_{ij}) &= \operatorname{diag}(1/4, 1/4, 1/4, 1/4) \begin{pmatrix} 1-a & a/3 & a/3 & a/3 \\ a/3 & 1-a & a/3 & a/3 \\ a/3 & a/3 & 1-a & a/3 \\ a/3 & a/3 & a/3 & 1-a \end{pmatrix} \\ &= \begin{pmatrix} (1-a)/4 & a/12 & a/12 & a/12 \\ a/12 & (1-a)/4 & a/12 & a/12 \\ a/12 & a/12 & (1-a)/4 & a/12 \\ a/12 & a/12 & a/12 & (1-a)/4 \end{pmatrix}. \end{aligned}$$

Thus the likelihood function is

$$L(a) = \prod_{i,j=1}^{4} p_{ij}^{n_{ij}},$$

and the log-likelihood is

$$\ln L(a) = \sum_{i,j=1}^{4} n_{ij} \ln p_{ij} = \ln(a/12) \sum_{i \neq j} n_{ij} + \ln((1-a)/4) \sum_{i} n_{ii}.$$

At a maximum $\hat{a}$ we find

$$0 = \frac{\sum_{i \neq j} n_{ij}}{\hat{a}} - \frac{\sum_i n_{ii}}{1 - \hat{a}},$$

so

$$\hat{a} = \frac{\sum_{i \neq j} n_{ij}}{\sum_{i,j} n_{ij}}.$$

This is not be too surprising, since the parameter a described the proportion of sites in which we expect to observe a substitution, and the formula for $\hat{a}$ computes the proportion of sites in which we actually observe one. Indeed, we suggested in Chapter 7 that this be the quantity used in the formula for the Jukes-Cantor distance, but now we have shown we are in fact using a maximum likelihood estimate of the parameter, further justifying that suggestion.

Making some fixed choice of α (which is equivalent to choosing an appropriate time scale) a similar calculation of maximum likelihood estimates could be done for the parameter t in the continuous time formulation of the Jukes-Cantor model. This of course leads to the same estimate for the Jukes-Cantor distance, so we omit the work.

The Kimura models, and general Markov models, can also be handled similarly, so we leave that for exercises. The resulting formulas for parameter estimates are again not surprising, as is often the case when maximum likelihood is used in simple situations.

8.3 Inferring Trees by ML

So how does maximum likelihood give us a framework for phylogenetic inference of trees? Suppose we have aligned sequence data for the n taxa in a set X, and we assume a particular model of molecular evolution.

To be concrete, we might use a general time-reversible model with a common rate matrix Q for all edges, and a root base distribution which is an eigenvector of Q with eigenvalue 0. Then if the root distribution is given by

$$\mathbf{p} = \begin{pmatrix} p_A & p_G & p_C & p_T \end{pmatrix},$$

we can take Q of the form

$$Q = \begin{pmatrix} * & p_G\alpha & p_C\beta & p_T\gamma \\ p_A\alpha & * & p_C\delta & p_T\epsilon \\ p_A\beta & p_G\delta & * & p_T\eta \\ p_A\gamma & p_G\epsilon & p_C\eta & * \end{pmatrix},$$

where the diagonal entries are chosen so rows sum to zero. We can also choose, say, $\alpha = 1$ to specify a time-scale. The parameters for our model are 1) a binary phylogenetic X-tree T, 2) any 3 of the entries of $\mathbf{p}$, 3) $\beta, \gamma, \delta, \epsilon, \eta$, and 4) the edge lengths $\{t_e\}_{e \in E(T)}$. There are additional restrictions on parameter values

so that certain matrix entries have appropriate signs for them to be biologically meaningful, but we will not be explicit about them.

Notice here that the tree itself is a parameter — a non-numerical one, but a parameter none the less. When we attempt to maximize the likelihood function, we will have to do so over all allowable numerical parameters as well as the discrete variable of the tree topology.

We thus consider a log-likelihood function for each fixed tree T,

$$\ln L_T = \sum_{(i_1,\ldots,i_n)\in\{A,G,C,T\}^n} n(i_1,\ldots,i_n)\ln(p(i_1,\ldots,i_n)),$$

where $p(i_1,\ldots,i_n)$ is a function of the numerical parameters giving the probability of observing the pattern $(i_1,\ldots,i_n)$ at a site (as computed in Chapter 6), and $n(i_1,\ldots,i_n)$ is the count of this pattern in the aligned sequence data. We emphasize that $\ln L_T$ is a function of all the numerical model parameters associated to T — essentially a function of the variable entries of $\mathbf{p}_\rho$, Q, and $\{t_e\}_{e\in E(T)}$.

We now need to find the values of the numerical parameters that maximize L_T. In practice, for more than a few taxa this will have to be done by numerical methods rather than by exact solution.

Once we have maximized L_T for a fixed T, however, we will have to do the same for all other T, comparing the maximum values we found for each. We then pick the tree T and the numerical parameter values that maximize its likelihood function as the ML estimators for our data.

As should be obvious, these computation are simply too involved to be done by hand, even in a toy problem for instructional purposes. Armed with a computer, we still have a tremendous amount of work to do, optimizing numerical parameters for each fixed tree, as we search among all trees. Heuristic approaches are of course necessary to make the calculations tractable. We will seldom be absolutely sure we have found the true maximum, and will be limited in how many taxa we can deal with by the power of our computer and the design of the software.

When maximum likelihood is used for tree inference, it is typical to assume a model such as GTR (or an extension of it with rate variation, as will be discussed in Chapter 10). There are practical reasons for this, since using a common rate matrix on all edges and not needing to consider root location as a variable keeps the number of parameters low, making the optimization much more tractable. Sometimes the common rate matrix and stable base distribution assumptions of GTR are reasonable on biological grounds as well. If they are, then using a model with fewer parameters is desirable as well, since a too general model risks 'overfitting' the data.

However, there are data sets for which it is clear the base distribution is not stable, and standard maximum likelihood implementations seem to have little ability to deal with this. While the general Markov model allows changing base distributions, because of the large number of parameters it requires, it is not dealt with by standard software.

8.4 Exercises

1. When finding maxima of the likelihood functions in Examples (1) and (2) in the text, we ignored issues of whether these occurred at endpoints of the interval [0,1], or equivalently we ignored showing our unique critical points were in fact maxima. Correct this shortcoming in our presentation.

2. Show that if in Example (1) we have 'perfect' data from N trials, in the sense that $n = Np$, and $m = N(1-p)$, then the maximum likelihood estimate $\hat{p}$ recovers the true value of p.

3. Show that if in Example (2) we have 'perfect' data from N trials, in the sense that $n_{hh} = Np_1^2$, $h_{ht} = Np_1(1-p_1)$, $n_{th} = N(1-p_1)p_2$ and $n_{tt} = N(1-p_1)(1-p_2)$, then the maximum likelihood estimates $\hat{p}_1, \hat{p}_2$ recover the true values of p_1, p_2.

4. Give an intuitive explanation of why formula (8.2) is reasonable, by identifying all single coin tosses in the full many-trial experiment that can be used to estimate p_1.

5. For the Kimura 2-parameter model describing the evolution of a sequence $S0$ to $S1$ along a single edge, guess reasonable formulas to estimate the parameters β, γ. Then find formulas for the maximum likelihood estimators. Do they agree?

6. Repeat the last exercise for the Kimura 3-parameter model.

7. Repeat the last exercise for the general Markov model.

8. For the GTR common rate matrix model discussed above, explicitly give all restrictions on parameter values that are necessary for them to be biologically meaningful.

9. For the GTR common rate matrix model, how many variables appear in the likelihood function for a particular binary tree T relating n taxa?

Chapter 9

Consistency and Long Branch Attraction

It's important, especially for mathematicians, to keep in mind that the basic motivating problem of phylogenetics is one of *inference*: Given *data*, how do we pick the 'best' tree? Real data is not exactly described by a mathematical model, and so a method that can infer the 'correct' tree from a mathematical idealization of data may not work well in practice.

It's also too much to hope that any method could give us the true evolutionary tree underlying any data set we might try it on. We would at least like to know under what circumstances it is likely to do well, and what circumstances might be problematic. For a method to provide a useful tool, we thus not only need to understand its performance in theory with 'perfect' data sets arising from attractive models, but also understand how it might behave in practice, on data sets of shorter sequences, where model assumptions are violated.

This is, of course, a big task. With few evolutionary histories known beyond all doubt, we typically cannot test inference methods on real data to see how well they do. Instead, we can simulate data (according to some model, perhaps different from the one underlying the inference method) on some specific tree, and then see how well the method reconstructs the tree we originally chose. This still cannot get at the issue of how close our models are to biological reality, but it does shed light on some of the difficulties of tree inference.

In this chapter, we discuss one circumstance that is well-understood to cause difficulties in phylogenetic inference: the phenomenon of *long branch attraction*.

Consider the metric quartet tree of Figure 9.1. Here two of the pendent edges, in different cherries, are much longer than the other edges. That such a tree can be difficult to correctly infer from data is not too surprising. The two taxa that are metrically the most closely related are not most closely related topologically. In fact, this was the problem that motivated our introduction of neighbor joining as an improvement over UPGMA when we discussed distance

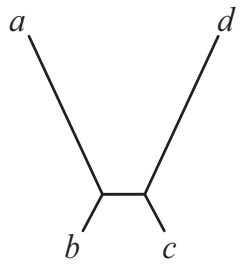

Figure 9.1: A metric quartet tree

methods.

But the neighbor joining algorithm, or even a full maximum likelihood approach to inference, can still perform poorly for real data from such a tree. Because of the small size of the state space of the characters we typically use (4 for DNA), for finite length sequences the small number of mutations occurring on the internal edge might be subsequently obscured by the many mutations occurring on the long edges. Our goal in this chapter is to try be more precise about this issue.

9.1 Inconsistency of Parsimony

Suppose we choose to use parsimony on a quartet dataset to infer a tree. For the 4 taxa a, b, c, d, we have a sequence of characters. In this setting, parsimony reduces to a simple scheme.

First, there are only a few types of characters that are informative. An informative character must have for the 4 taxa a, b, c, d, in order, states

$$xxyy, \quad xyxy, \text{ or } xyyx,$$

where x, y denote two distinct states. So letting n_1, n_2, n_3 denote the count of such characters in a data set, we compute the parsimony score for the 3 quartet trees

$$T_1 = ab|cd, \quad T_2 = ac|bd, \quad T_3 = ad|bc$$

as

$$\begin{aligned} ps(T_1) &= n_1 + 2n_2 + 2n_3 = (2n_1 + 2n_2 + 2n_3) - n_1, \\ ps(T_2) &= 2n_1 + n_2 + 2n_3 = (2n_1 + 2n_2 + 2n_3) - n_2, \\ ps(T_3) &= 2n_1 + 2n_2 + n_3 = (2n_1 + 2n_2 + 2n_3) - n_3. \end{aligned} \tag{9.1}$$

To choose the most parsimonious tree(s) T_i, we therefore simply pick the value(s) of i maximizing n_i.

Now suppose our data is generated by a probabilistic model of the sort in Chapter 6. Then we can compute expected values of the numbers n_i, and

see whether parsimony will infer the correct tree. Of course, using expected values of the n_i is essentially the same as imagining we have infinitely long data sequences that are produced exactly in accord with our model. We will not be investigating issues arising from stochastic variation in shorter sequences, or any issue of poor model fit. Rather, the question is whether parsimony is a statistically *consistent* method of inference for data from a Markov model. More precisely, we are asking the question: If parsimony is applied to longer and longer sequences produced exactly according to the model, do the inferred trees eventually stabilize on the correct one?

The first examples of parameter choices for a Markov model leading to inconsistency of parsimony are due to Felsenstein, for a two-state model. Although the result has been generalized to models with more states, and to more complicated trees, for simplicity we follow the original argument.

Theorem 19. For a two-state Markov model on quartet trees, there are parameters for which parsimony is an inconsistent inference method.

Proof. For the tree in Figure 9.1, place the root ρ at the internal vertex joined to a, b, and denote the other internal vertex by v. Consider the two-state Markov model with parameters

$$p_\rho = \begin{pmatrix} 1/2 & 1/2 \end{pmatrix},$$
$$M_{(\rho,a)} = M_{(v,d)} = \begin{pmatrix} 1-q & q \\ q & 1-q \end{pmatrix},$$
$$M_{(\rho,b)} = M_{(\rho,v)} = M_{(v,c)} = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}.$$

Then the probabilities of the 3 patterns that are informative for parsimony are

$$\begin{aligned} p_1 &= p_{xxyy} = (1-q)^2 p(1-p)^2 + 2q(1-q)p(1-p)^2 + q^2 p^3, \\ p_2 &= p_{xyyx} = (1-q)^2 p^2(1-p) + 2q(1-q)p^2(1-p) + q^2(1-p)^3, \\ p_3 &= p_{xyxy} = (1-q)^2 p^3 + 2q(1-q)p(1-p)^2 + q^2 p(1-p)^2. \end{aligned} \tag{9.2}$$

But as the sequence length, N, goes to infinity, we have that $n_i/N \to p_i$. Since parsimony will be consistent only when $n_1 > n_2, n_3$, we have consistency only when $p_1 > p_2, p_3$.

Now, by straightforward algebra, equations (9.2) imply

$$\begin{aligned} p_1 - p_2 &= (1-2p)(p(1-p) - q^2), \\ p_1 - p_3 &= p(1-2p)(1-2q). \end{aligned} \tag{9.3}$$

In these formulas we should only consider values of $p, q \in (0, 1/2)$ as biologically plausible. (See also Exercise 4.) Thus for parameters in this range $p_1 > p_3$ always holds, while $p_1 > p_2$ holds exactly when

$$p(1-p) > q^2.$$

However, there are values of p, q in the allowed range where this last inequality does not hold, and so parsimony is inconsistent for such choices. □

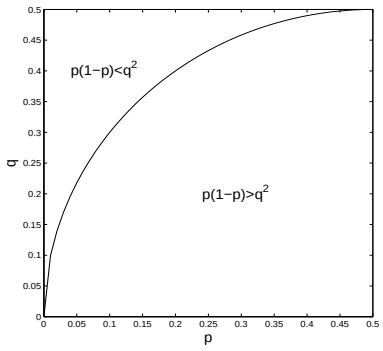

Figure 9.2: Parsimony is consistent only for parameters below the curve.

A graph of the (p, q)-parameter space for the model in the proof is given in Figure 9.2, indicating the regions for which parsimony is a consistent method of inference. We see that if p is sufficiently small in comparison to q, so the internal branch of the tree is very short, we have inconsistency of parsimony.

Note that when the parameters fall in the region of inconsistency (sometimes called the *Felsenstein zone*), the proof has shown $p_2 > p_1$, so the tree parsimony will infer from perfect data is T_2. In other words, the two long branches are erroneously joined, and we have an instance of *long branch attraction*.

An extreme interpretation of this theorem and figure is that since parsimony is sometimes inconsistent, it is not a good method to use. A more moderate view is that parsimony is in fact consistent over quite a wide range of parameters, and we now have some idea of what might lead it to behave poorly. Of course for real data we might not have much idea in advance of how the tree should look, so that whether the tendency for long branches to attract one another is a problem can be difficult to determine. However, if we infer a tree with several long branches joined to one another, we should consider the possibility.

From this four-taxon investigation it's easy to imagine that long branch attraction might not occur for molecular clock trees. However, there are examples of molecular clock trees with more taxa for which long branch attraction occurs under parsimony, even on perfect data, and thus for which parsimony is not consistent.

9.2 Consistency of Neighbor Joining and Maximum Likelihood

The other methods of inferring trees that we've discussed — distance algorithms and maximum likelihood — are statistically consistent, at least if performed assuming the correct model, and with a few mild and biologically reasonable

restrictions on parameters. We won't attempt formal proofs of these facts, but will instead sketch the ideas.

Consider first the computation of distances from data sequences produced according to a specific model. Then as sequence length grows, it is easy to show that provided we use a distance formula that is associated with the model, we will infer distances that approach the true ones for the parameter choice. (Thus if our data is generated by a Kimura 2-parameter model, we may use Kimura 2-, or 3-parameter distances, or the log-det distance, but not the Jukes-Cantor distance.) In fact, this basic fact follows from the continuity of the distance formulas.

Once we have distances, we need to understand the behavior of the Neighbor Joining algorithm. We've already claimed in Theorem 16 (and shown in Exercise 28 of Chapter 5) that Neighbor Joining recovers the correct tree provided we have *exact* distances from a binary tree with all positive edge lengths. It's necessary, then, to show that from dissimilarities sufficiently close to the exact distances, we recover the same tree. So what is needed is in essence a statement that the output of the algorithm is a continuous function of the input dissimilarities, at least in the neighborhood of exact distances for binary trees with all positive edge lengths. We will not show this, though it is certainly plausible.

Note that the requirements that the tree be binary and edge lengths be positive are the mild restrictions that we indicated were needed beforehand.

Turning now to Maximum Likelihood, there are rather general proofs that maximum likelihood is a consistent method of statistical inference in many settings, and these can be modified for phylogenetic models. However, they require that one first prove the model is *identifiable*. In the phylogenetic setting, this means that any joint distribution of bases at the leaves of a tree arises from a unique tree and choice of Markov parameters.

In fact, the general Markov model is not strictly identifiable. For instance, any two n-taxon tree topologies with the the same root distribution and the identity matrix chosen for all Markov matrix parameters will give the same joint distribution of bases at the leaves. Thus for these parameter choices we cannot reconstruct the tree topology from the joint distribution tensor. Ruling out this and related parameter choices, however, we can impose restrictions so that the tree topology is identifiable by means of the log-det distance, for instance. For this, it is sufficient to require that the tree be binary and all Markov matrices have determinant $\neq \pm 1, 0$, so that all edge lengths are positive.

There remain other non-identifiability issues, though. For instance, one can permute the names of the bases at an internal node of the tree, adjusting the Markov parameters on incident edges accordingly by permuting rows or columns, and again not change the joint distribution (Exercise 7). This gives only finitely many parameter choices for each joint distribution, though. We can make a unique choice from these by imposing biologically reasonable assumptions that the diagonal entries of all matrices be 'largest.'

Of course it remains to be shown that with these restrictions on parameters the map to joint distributions is one-to-one. Since the function is a polynomial

one in many variables, this is by no means trivial. Nonetheless, it can be shown and a proof of statistical consistency can be completed.

Finally, we should emphasize that while statistical consistency is certainly desirable for an inference method, it does not lay to rest all qualms we might have. First, a claim of consistency is a statement about behavior when we know the correct model. If, say, we attempt to use maximum likelihood with a Jukes-Cantor model for inference, and the data actually is not fit well by that model, the consistency results above give us no guarantees. Since biological data is unlikely to be described perfectly by any simple model, we certainly have a violation of assumptions in any real-world application. How that violation of assumptions effects inference results is a question of *robustness*.

9.3 Performance on Short Sequences

Another issue with statistical consistency is that it is a statement about behavior of a method on infinitely long sequences, or more properly, about limiting behavior as the sequence length goes to infinity. Real biological sequences are of course of finite length. So how do methods like Neighbor Joining and maximum likelihood behave on finite length sequences?

Probably the easiest way to investigate this is through simulation. We can choose a model, tree, and numerical parameters, and simulate data according to these. Then we can apply the inference method to the simulated data, and see if it recovers the original tree. (See Exercise 8.)

When this is done, we find the long branch attraction phenomenon seems quite universal, regardless of the method of inference. For instance, for any fixed sequence length, there is a region of parameter space much like that in Figure 9.2 in which we often infer the wrong tree. The precise shape and size depends on the method of inference and the sequence length, but the region is there. When we are able to prove consistency, we know this region must shrink as we make the sequence length longer. However, in practice we may not be able to obtain longer data sequences whose mutation we feel we can model well, so the assurance of statistical consistency may not be helpful.

Several long edges leading from the ends of a short edge, then, are quite generally problematic. It is wise to keep this is mind when inference produces a tree with several long edges leading from a common vertex. When such a tree is inferred, it might be possible to include additional taxa in the data set in order to create additional bifurcations breaking up the long edges. With an expanded data set, we may be able to better infer a tree.

9.4 Exercises

1. Explain the computation of the parsimony scores in equations (9.1) for the quartet trees.

2. Check the formulas for p_1, p_2, p_3 in equations (9.2) in the proof of Theorem 19.

3. Check the formulas for $p_1 - p_2$ and $p_1 - p_3$ in equations (9.3) in the proof of Theorem 19 (using software, or by hand).

4. Show that if the two-state Markov matrix

$$M(p) = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}$$

is of the form $M = e^{Qt}$ for some non-zero rate matrix Q, $t > 0$, then $p \in (0, 1/2)$. Why is this biologically plausible?

5. For the Markov model in the proof of Theorem 19, explain how Figure 9.2 fits with the intuition that as long as sufficiently few mutations occur, parsimony should infer the correct tree. Is this intuition strictly correct? Show the curve $p(1-p) = q^2$ has a vertical tangent at (0,0), and explain why this is relevant.

6. Assuming we use a valid distance formula for our model, explain informally why UPGMA will be statistically consistent if the underlying tree has all leaves equidistant from the root, but will not be without this assumption on the tree.

7. Consider a 3-taxon tree, with root ρ at the central vertex. Suppose for the general Markov model on this tree we have parameters $\mathbf{p}_\rho$, M_1, M_2, M_3, producing a certain joint distribution of states at the leaves. If P is a permutation matrix and $\mathbf{p}'_\rho = \mathbf{p}_\rho P$, what Markov matrices M'_1, M'_2, M'_3 will, along with $\mathbf{p}'_\rho$, produce the same joint distribution as the original parameters?

8. Using MATLAB, simulate some sequences of length 300 bases according to a Jukes-Cantor model on the tree in Figure 9.1 using a variety of parameter choices (p, q) as in the proof of Theorem 19. Then use Neighbor Joining with the Jukes-Cantor distance to infer a tree. Find parameter choices for which the method seems to usually give the correct tree, and other parameter choices for which you usually see long branch attraction.

   If you are ambitious, perform many simulations for enough values of (p, q) to produce an empirical diagram like Figure 9.2, showing when this approach to this inference problem usually produces the correct tree.

Chapter 10

Rate-variation Models

All the Markov models of DNA mutation introduced thus far assume that every site in the sequences behaves identically. However, this assumption is far from justifiable biologically. For instance, in coding DNA, due to the redundancy of the genetic code, the third base of many codons can undergo a substitution without any effect on the protein product. Also, if both coding and non-coding sites are included in a sequence, it's reasonable that the non-coding sites might be able to change more freely, at least if they have no regulatory purpose. In general, then, it is desirable to model the mutation process so that some sites might mutate quickly, others slowly, and yet others not at all.

10.1 Invariable Sites Models

The first approach to introducing rate variation models is to create two classes of sites. The first class of *variable* sites mutates according to a model of the sort discussed earlier, and the second class of *invariable* sites undergoes no mutation at all. When we examine data from such a model, if we observe a mutation at a site it is certainly in the first class. However, if we do not observe a mutation, we generally cannot tell which class the site came from. It may have been free to mutate, but just didn't, or it may have been invariable. Thus the sites that are observed as unvarying are a larger set than the invariable ones.

For concreteness, let's consider a model with invariable sites of this sort in which variable sites mutate according to the general Markov model, usually called the GM+I model. We'll formulate this for characters with κ states, for trees relating n taxa.

For any fixed rooted tree T relating n taxa, we have the usual parameters for the variable sites of a root distribution vector $\mathbf{p}_\rho$ and Markov matrices for each edge $\{M_e\}_{e\in E(T)}$. In addition we need a class size parameter r, indicating that with probability r a site is variable, and with probability $1-r$ it is invariable. Finally, for the invariable sites we need another distribution vector $\mathbf{q}$ indicating the probabilities that an invariable site is occupied by each of the possible bases.

Thus a binary n-taxon tree will require

$$1 + 2(\kappa - 1) + (2n - 3)\kappa(\kappa - 1) \tag{10.1}$$

parameters, which is only κ more than the GM model.

In order to see the relationship between the GM+I model parameters and the joint distribution tensor describing observations of bases at the leaves of a tree, we begin by analyzing separately the two classes of sites in the model.

For the GM model, we've seen in Chapter 6 how to produce the entries of the joint distribution tensor as polynomials in the parameters that are entries of $\mathbf{p}_\rho$, $\{M_e\}_{e \in E(T)}$. For a tree relating n-taxa, this tensor will be n-dimensional, of size $\kappa \times \kappa \times \cdots \times \kappa$. So suppose we've found this array P_1, where $P_1(i_1, i_2, \ldots, i_n)$ gives the probability that a variable site shows the pattern $(i_1, i_2, \ldots, i_n)$ of states at the leaves.

Now for the invariable sites we can do a similar calculation, using the root distribution vector $\mathbf{q}$ and the identity matrix I for all Markov matrices on edges. But it's also easy to see this will just give us an n-dimensional $\kappa \times \kappa \times \cdots \times \kappa$ array P_2 such that

$$P_2(i_1, i_2, \ldots, i_n) = \begin{cases} q_{i_1} & \text{if } i_1 = i_2 = \cdots = i_n, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the joint distribution for the full GM+I model will be

$$P = rP_1 + (1 - r)P_2, \tag{10.2}$$

a weighted sum of the distributions for each class.

In statistics a model such as this, in which several classes are modeled individually, but then combined so that we no longer know which class an individual is in, are called *mixture models*.

Of course there are many variations on this idea. One might consider for instance a general time reversible model, with invariable sites (GTR+I). One might also decide it is more appropriate that the invariable site distribution be the same as the root distribution for the variable sites so that $\mathbf{q} = \mathbf{p}_\rho$.

If we believe a model such as GM+I describes our data, how should this affect our methods of inference?

The results of parsimony are unaffected by invariable sites, so its use is reasonable under the same circumstances as it would be for the variable sites alone. Of course if we use parsimony for inference we need not be explicit about the model anyway, as the method makes no use of it.

Our derivations of distance formulas for the various models all tacitly assumed no invariable sites are present. However, if a small number are present, we may view the distances as approximately valid. Nonetheless, the use of the distance formulas we've developed essentially requires that the number of invariable sites be negligible. To continue to use a distance method requires we

either find a new distance formula appropriate to this new model, or accept a potential error from pretending no invariable sites exist.

However, the maximum likelihood framework handles a model with invariable sites with little additional complication. There are a few additional numerical parameters that must be varied as we search for maxima of the likelihood functions, but that is all. It is typically in this setting that such models are used.

10.2 Rates Across Sites Models

It's easy to now imagine how one might have a finite number of classes of sites, each modeled by a different choice of GM parameters on a tree, in a more complex mixture than a GM+I model. For each class we produce a joint distribution array, and then take a weighted sum of these, much as in equation (10.2). The GM+I model is just a particularly simple form, where the parameters for one of the two classes allow no mutation. In current practice, though, models with many fewer parameters than a mixture of GMs are typically used.

For a fixed tree T we assume a common rate matrix Q for all edges of the tree, say one for the GTR model, and a root distribution vector that is a stable base distribution for Q. We have scalar edge lengths $\{t_e\}_{e\in E(T)}$ for all edges of the tree. If we choose to use m classes of sites, we create m scalar *rate parameters* $\lambda_1, \lambda_2, \ldots, \lambda_m$, as well as a vector $\mathbf{r} = (r_1, r_2, \ldots, r_m)$ giving the distribution of sites in the rate classes.

Now for the ith rate class we will use the rate matrix $\lambda_i Q$ throughout the tree. For that class, then, on an edge e of the tree we have the Markov matrix $M_{e,i} = e^{t_e \lambda_i Q}$. It's now straightforward to compute P_i, the joint distribution array at the leaves for the ith class.

Finally, we combine the classes to get the joint distribution for the mixture model

$$P = \sum_{i=1}^{m} r_i P_i.$$

To be a bit more sophisticated, one can also imagine a continuous distribution of rates, given by a density function $r(\lambda)$, in which case we have

$$P = \int_{\lambda} r(\lambda) P_\lambda \, d\lambda,$$

where the integration is entrywise, just as the sum was in the case of a finite number of classes.

In practice, its common to use a Γ-distribution of rates (with mean 1),

$$r_\alpha(\lambda) = \frac{\alpha^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\alpha\lambda}.$$

The shape parameter α then adds only one more parameter to our model, yet allows flexibility in the distribution for a better maximum likelihood fit.

Unfortunately, there is no clear biological justification for the use of a Γ-distribution, instead of some other distribution. Even though Γ is common, it is a rather arbitrary choice.

In fact, it is not even the Γ-distribution, but rather a discretization of it with a finite number of rate classes, that is used in software implementations of maximum likelihood. Typically biologists choose to use only a handful (four or five) of rate classes since additional ones slow computations and seem to produce little improvement in likelihood scores on real data.

Finally, it is straightforward and often desirable to add an additional class of invariable sites, producing models with designations such as $GTR + \Gamma + I$.

10.3 The Covarion Model

There is another way of introducing a form of rate variation into models, though its mathematical development is more recent than the rates-across-sites approach, and it has not yet been implemented in the most commonly used software packages. The basic idea, which arose in a paper of Fitch and Markowitz in 1970, is biologically quite an attractive one.

Note that in a rates-across-sites model, each site is presumed to have a fixed rate parameter λ which remains constant throughout the tree. Whatever constraints are responsible for some sites mutating at different rates than others are thus modeled as unchanging across all lineages.

Particularly if we consider evolution over long time periods, however, we might expect this to be unreasonable. Perhaps early on some sites are unable to change because they code for a part of a protein that is essential for the organism to live. After other parts of the sequence have evolved, however, the part of the protein those sites code for may no longer be so essential, and so they become free to vary in a different part of the tree. Fitch and Markowitz called the codons that were free to vary at a particular time 'covarions' as shorthand for 'concomitantly variable codons,' and now the word 'covarion' has come to refer to models in which characters may undergo some switching between being free and not free to vary as evolution proceeds across the tree.

Note that the motivation given here for a covarion model is essentially an argument against an assumption of independence of the mutation process at different sites. However, it is quite unclear how to formulate such a model in a mathematically tractable way. Thus the model we will give, introduced by Steel and Tuffley, will preserve independence, and simply include a switching mechanism.

For a DNA model, instead of the usual 4 states A, G, C, T for our characters, we will have 8 states, which we designate by

$$A^{\text{on}}, G^{\text{on}}, C^{\text{on}}, T^{\text{on}}, A^{\text{off}}, G^{\text{off}}, C^{\text{on}}, T^{\text{off}}.$$

The superscript 'on' means the site is currently free to vary, while 'off' designates it is currently held invariable.

For the 'on' states we assume a instantaneous rate matrix Q of a GTR model, and let $\mathbf{p}$ be its stable base distribution, so $\mathbf{p}Q = \mathbf{0}$. We need two additional parameters, an instantaneous rate s_1 at which 'on' states switch to 'off' states, and an instantaneous rate s_2 at which 'off' states switch to 'on.' We construct an 8×8 rate matrix

$$\tilde{Q} = \begin{pmatrix} Q - s_1 I & s_1 I \\ s_2 I & -s_2 I \end{pmatrix},$$

where I denotes a 4×4 identity matrix. Now letting

$$\sigma_1 = \frac{s_2}{s_1 + s_2}, \quad \sigma_2 = \frac{s_1}{s_1 + s_2},$$
$$\tilde{\mathbf{p}} = (\sigma_1 \mathbf{p}, \sigma_2 \mathbf{p}),$$

we have (Exercise 3) that

$$\tilde{\mathbf{p}}\tilde{Q} = \tilde{\mathbf{0}}, \tag{10.3}$$

so that $\tilde{\mathbf{p}}$ is a stable distribution for $\tilde{Q}$. In fact, the rate matrix $\tilde{Q}$ and root distribution vector $\tilde{\mathbf{p}}$ form a time reversible model (Exercise 4).

Now for any tree T, with a root ρ chosen arbitrarily, we have an 8-state model with root distribution vector $\tilde{\mathbf{p}}$, rate matrix $\tilde{Q}$, and edge lengths $\{t_e\}_{e \in E(T)}$, where the Markov matrix $M_e = \exp(t_e \tilde{Q})$ is assigned to the edge e.

There is one remaining feature of the covarion model, however, to be formulated. When we observe sequences, we are not able to distinguish whether a site is currently 'on' or 'off'. For instance, both states A^{on} and A^{off} are observed simply as A. (For those familiar with hidden Markov models, the covarion model is of that sort, with some of the state information unable to be observed.)

We will show how to incorporate this into the model in two slightly different, but equivalent formulations. Both make use of the 8×4 matrix

$$H = \begin{pmatrix} I \\ I \end{pmatrix},$$

constructed from two stacked identity matrices, which has the effect of 'hiding' the 'on/off' feature of a base. More specifically, since

$$\begin{pmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 \end{pmatrix} H = \begin{pmatrix} p_1 + p_5 & p_2 + p_6 & p_3 + p_7 & p_4 + p_8 \end{pmatrix},$$

then H acts on any vector giving a distribution of the 8 states in our chosen order to give the 4 combined states corresponding to the bases in DNA.

One approach is simply to make the Markov matrices on each edge of the tree be the M_e as described above for internal edges, while for edges leading to leaves use $M_e H$. (Here we assume we have located the root of the tree at an internal vertex.) While this matrix is not square, it still has non-negative entries, with rows summing to 1, and so a perfectly reasonable stochastic interpretation. With parameters on all edges of the tree, and a root distribution, we can now calculate the joint distribution at the leaves in the usual way.

The other approach is to first calculate the joint distribution at the leaves using the 8×8 Markov matrices on all edges, giving a $8 \times 8 \times \cdots 8$ array P of dimension n for an n-taxon tree. Then, to account for our inability to observe the 'on/off' aspect of states, we multiply the array P by H along each of its dimensions, in the same way we multiple matrices. For example, to multiply in the first dimension we compute for $j_1 = 1, 2, 3, 4$,

$$P'(j_1, i_2, \ldots, i_n) = \sum_{i_1=1}^{8} P(i_1, i_2, \ldots, i_n) H(i_1, j_1),$$

so P' is $4 \times 8 \times 8 \times \cdots \times 8$. After a similar multiplication in the other indices, we arrive at the $4 \times 4 \cdots \times 4$ array giving the joint distribution at the leaves for the covarion model.

10.4 Exercises

1. Explain the count in formula (10.1).

2. How many parameters are needed for a GTR+I model on a binary n-taxon tree, if we want the invariable sites to have the same distribution as the stable distribution of variable ones?

3. Show equation (10.3) holds.

4. Show the 8×8 rate matrix $\tilde{Q}$ of the covarion model together with the root distribution vector $\tilde{\mathbf{p}}$ form a time reversible model.

5. Explain why for the covarion model if M_e is an 8×8 Markov matrix, then $M_e H$ will have non-negative entries, with rows summing to 1. Give an interpretation of its entries as conditional probabilities. Also give an interpretation of the entries of H as conditional probabilities.

Bibliography

[AR04] Elizabeth S. Allman and John A. Rhodes. *Mathematical Models in Biology: An Introduction*. Cambridge University Press, Cambridge, 2004.

[Fel04] Joseph Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Sunderland, MA, 2004.

[Gas05] Olivier Gascuel, editor. *Mathematics of Evolution and Phylogeny*. Oxford University Press, Oxford, 2005.

[SS03] Charles Semple and Mike Steel. *Phylogenetics*, volume 24 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2003.

[Wat95] Michael S. Waterman. *Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman and Hall, London, 1995.