Chapter 9

Tree Space

As discussed in Chapter 2, as soon as we have more than a handful of taxa we wish to relate, there are many potential trees to consider. While distance methods give us a quick way to produce one tree, both parsimony and maximum likelihood approaches to tree inference require that we consider all trees in choosing an optimal one to fit data. In practice, this is usually not possible to do in an acceptable amount of time. Instead, we imagine a 'space' of trees, and move from one to another to using some sort of search procedure, trying to find the optimal tree according to the chosen criterion.

In this chapter we will focus on binary topological trees only, with no edge lengths specified. Other notions of tree space, such as for rooted topological trees, or for either rooted or unrooted metric trees can also be developed. We emphasize the unrooted topological case here since it is the one the plays a conceptual role in most current tree inference software.

9.1 What is Tree Space?

Suppose we are interested in relating 4 taxa, by an unrooted tree, using parsimony. Then we know there are only three topological possibilities that are fully resolved, as shown in Figure 9.1.

It is convenient to think of these three possibilities as the only three points in a '4-taxon, binary, unrooted tree space.' As we look for the most parsimonious tree, we might begin at any of the three points and determine its parsimony score. Then we move to another point, and determine its score. Finally, we move to the third point, and determine its score. Having visited each point in tree space, we've then completed an exhaustive search, and can be sure which tree of the three is most parsimonious.

With only 3 points in this tree space, this language of moving between points in a space may seem artificial; we could just say we need to compute scores for every tree. However, for 100 taxa, performing an exhaustive search among the 195!! binary, unrooted, topological trees might be too time consuming to do

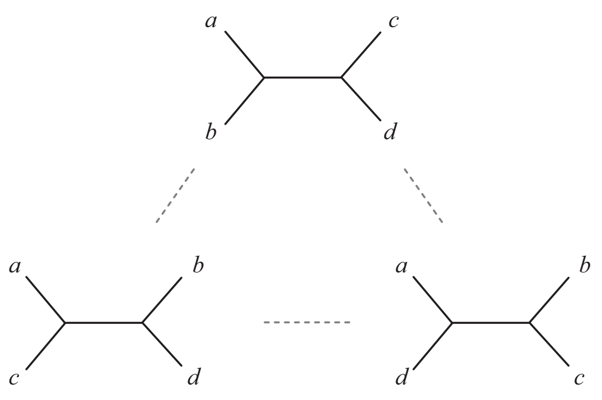

Figure 9.1: The three points of 4-taxon unrooted binary tree space.

completely. It is then more helpful to imagine a space with 195!! points in it, each corresponding to a possible tree, and possible methods of moving from one point to another.

Definition. For any fixed collection X of n taxa, the space of unrooted binary phylogenetic X-trees is a collection of $(2n-5)!!$ points, each of which corresponds to a distinct unrooted topological tree. The space of rooted binary phylogenetic X-trees is a collection of $(2n-3)!!$ points, each of which corresponds to a distinct rooted topological tree.

But this geometric language of 'space' suggests that we should consider some points to be close to one another, and others to be far apart. For instance, of the three trees in Figure 9.2, it seems natural that T_1 and T_2 should be closer to one another than either is to T_3, since even glancing at them shows they have many features in common. If we were searching for the most parsimonious tree, and had found that T_1 has a low parsimony score, we would also want to check the score of T_2, since it has so much in common, and therefore might also have a low score. On the other hand, knowing the score of T_1 doesn't help us get even a rough estimate of the score of T_3, since the trees are so different. To make precise the meaning of the basic intuition that similar trees should have similar scores requires that we pin down the phrase 'similar trees' by giving some notion of how far apart they are in tree space.

There are a number of different ways we can measure the similarity of trees. Each of these is called a *metric* on tree space, and simply gives a way of measuring how far apart we consider two trees to be. Before defining these, however, we first discuss how we might move around in tree space.

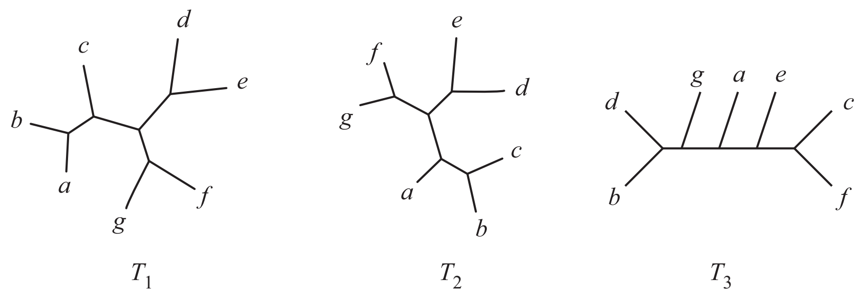

Figure 9.2: Intuitvely, T_1 and T_2 are more similar to each other than to T_3. A metric on tree space will provide a means of quantifying similarity.

9.2 Moves in Tree Space

A move in tree space should take us from one tree to another. In other words, it should begin with a tree, and modify it in a natural way, to produce another tree. While there are many imaginable complex moves, three simple ones are used most often.

Since these moves rearrange a given tree by making changes only to certain parts of it, when we depict these graphically we will group those parts of the tree that are not affected into larger pieces. Thus the triangles appearing in a diagram such as Figure 9.3 represent subtrees whose precise structure does not matter for the move we depict.

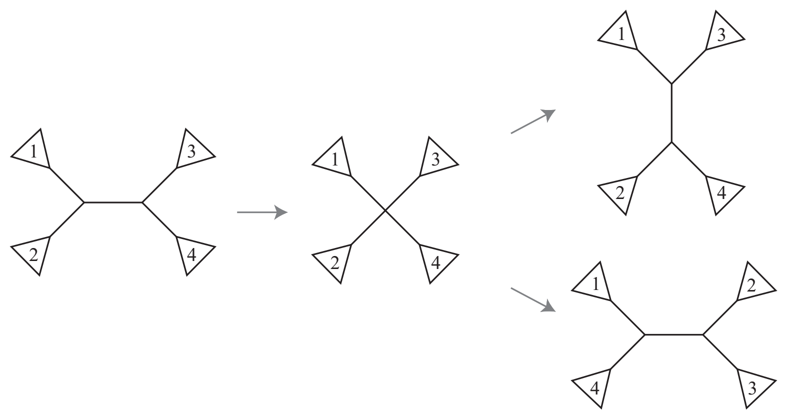

Figure 9.3: NNI moves from one tree to two others.

To create a small change in the tree, the simplest move is called a *nearest neighbor interchange* (NNI). To perform an NNI move, we pick a single internal edge of a tree, and first contract it so that the four edges leading off its endpoints now arise from a single vertex, as shown in Figure 9.3. The resulting tree is not binary, but we can resolve it into a binary tree by pulling a pair of the edges joining this vertex away from the others, and inserting an edge between the two pairs.

If the pair of edges we pull away in this process are ones that already met in the original tree, then we will just obtain the original tree once again. But any other choice of a pair will give us a different topological tree. Since the 4 edges can be grouped into two pairs in 3 ways, and one of these choices leads back to the original tree, there are exactly 2 new trees we can reach by performing an NNI move on a particular edge.

The simplest examples of NNI moves are already present in Figure 9.1. Starting at any tree, an NNI move on the internal edge of the tree can move us to either of the other two points in the space. The contraction of the internal edge leads to all 4 taxa attached by edges to a single central node. Then picking a pair of taxa and pulling these away from the others can produce any of the three trees.

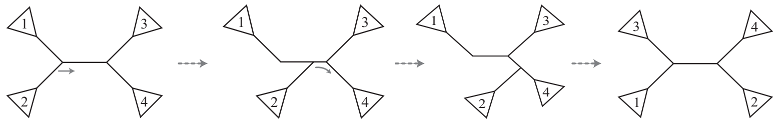

Figure 9.4: An alternate view of an NNI move from one tree to another.

Another way of thinking of an NNI move is shown in Figure 9.4. Focusing on a single internal edge, and the 4 edges that are joined to its ends, we 'grab' one of these 4 edges, and slide its attachment point down the edge toward its other end. There is no change in the tree topology as we do this, until we reach the other end. Then we must make a choice of a new edge onto which we slide the attachment point. (As we do the sliding, we must of course merge the two other edges meeting at the original attachment point into a single edge, and then split the edge on which the new attachment point is located into two edges.) Either of the choices gives a different topology from the original tree, and these are the two trees the NNI move can create. This explanation fits with the terminology 'nearest neighbor interchange', as a subtree has been moved from joining the rest of the tree at a particular location to one of the closest spots that gives a different tree topology.

Given an n-taxon unrooted binary tree, there are $n - 3$ internal edges at which we can perform an NNI move. A move on an edge can be performed to reach 2 different trees. It is not hard to see that every NNI move will produce a different tree, so there are a total of $2(n-3)$ trees one NNI move away from any other. This is of course much smaller than the number of trees. It is therefore reasonable to think of an NNI move as taking a small step in tree space, the smallest step possible, in fact.

The next move we consider is called a *subtree prune and regraft* (SPR). It proceeds by first picking some edge of the tree, and then detaching it from the tree at only one end. The detached edge will still have some taxa joined to it through its other end, forming a subtree that has been 'pruned' from the original tree. We then pick some edge of the other remnant of the tree, and reattach the subtree to it. (Technically, we must divide the chosen edge into two edges, so the subtree can be attached at the new node this introduces.) An illustration is given in Figure 9.5

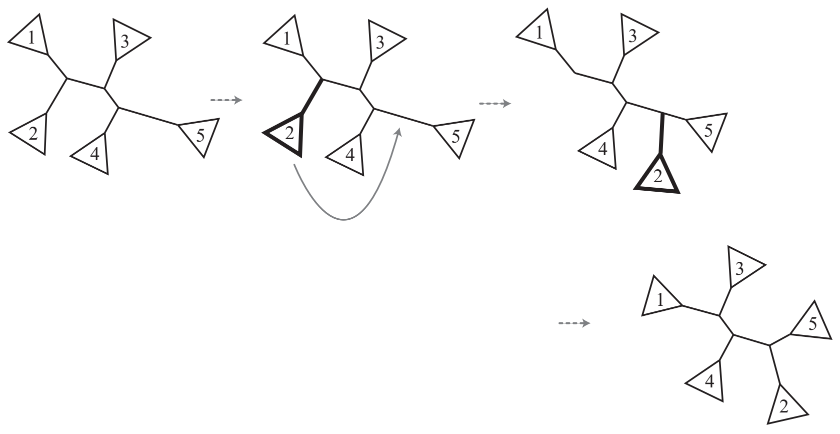

Figure 9.5: An SPR move on a tree.

It should be clear from the second description of an NNI move that any NNI move is also an SPR move; we just choose the regrafting point to be an edge that is very close by. However, for large trees there are many more possible SPR moves than NNIs.

To count how many SPRs could be performed on an n-taxon unrooted binary tree is more involved than counting NNI moves. First, it is helpful to describe the SPR move slightly differently: Instead of detaching an edge at only one end, remove the edge, leaving its endpoints in the two resulting trees components. Then chose one of these endpoints, and one edge in the *other* component of the tree, and introduce a new edge that goes between them. But note that this last choice of an endpoint and an edge in the other component for a regrafting locations is equivalent to just picking the edge for the regrafting location in *either* of the components. After all, once we've chosen the regrafting location, we must use the endpoint in the other component. Thus to specify a particular way of performing an SPR we really need only pick two edges — the one to remove/detach, and then another one where the regrafting occurs. Since the tree has $2n-3$ edges, there are $2n-3$ choices for the first edge, and then $2n-4$ choices for the second, for a total of

$$(2n-3)(2n-4) \tag{9.1}$$

choices of ordered pairs of edges.

However, this overcounts the number of trees we can obtain through an SPR. For instance, if the edge we remove has a vertex in common with the edge where regrafting occurs, the tree topology does not change. Thus we should not count pairs of edges that meet. Since each terminal edge meets 2 other edges, and each internal edge meets 4, there are

$$2n + 4(n-3) = 6(n-2) \tag{9.2}$$

such cases.

A second case of overcounting occurs when the two chosen edges do not meet but have a single edge between them. In this case, the SPR is just an NNI move on the edge between them. But if we either remove the first edge and regraft to the second, or remove the second and regraft to the first, we obtain the same outcome. Thus the order of the two chosen edges does not matter. Since we have already counted the number of different trees we can reach by NNI moves, we know there are $2(n-3)$ of them. However, our earlier count included

$$8(n-3) \tag{9.3}$$

pairs leading to this case: For each of the $n-3$ internal edges of the tree we might have chosen any of 8 ordered pairs of edges that meet it as the pair determining the SPR.

Finally, as long as the two edges determining an SPR move are separated by more than one edge, it is not hard to see that every choices of ordered edges leads to a distinct tree, different than those obtained by NNIs. Counting such choices is most easily done by taking the total count for all ordered pairs of edges in (9.1), and removing the special cases of (9.2) and (9.3). This gives

$$(2n-3)(2n-4) - 6(n-2) - 8(n-3) = 4(n-3)(n-4) \tag{9.4}$$

ordered pairs of this type.

The total number of trees we can reach by a single SPR move is thus this last number, plus the $2(n-3)$ reachable by NNIs. We summarize all these calculations by the following.

Theorem 17. For any n-taxon unrooted binary tree, there are
(i) $2(n-3)$ distinct trees one NNI move away, and
(ii) $2(n-3)(2n-7)$ distinct trees one SPR move away.

The important observation from this theorem is that there are relatively few (approximately $2n$) trees close to a given one by NNIs, while there are many more (approximately $4n^2$) one move away by SPRs. If n is large this difference is substantial. Thus if we move through tree space by NNIs, we will have fewer 'directions' to move in at any point than if we use SPRs. Using SPRs, we can take bigger jumps, changing the tree in many more ways. Neither of these moves is superior to the other; they are just different. If we have already found a pretty good tree, we might want to use NNIs to take small steps around it, exploring tree space thoroughly only in its vicinity. On the other hand, if we want to wander widely over tree space, looking for any trees that seem good, any number of SPRs will allow us to move around more widely than the same number of NNIs.

A third useful type of move is a *tree bisection and reconnection* (TBR). A TBR move involves first picking some edge of the tree to remove. Next an edge is chosen in each of the resulting tree components. Finally, these two edges are connected by a new edge, as shown in Figure 9.6.

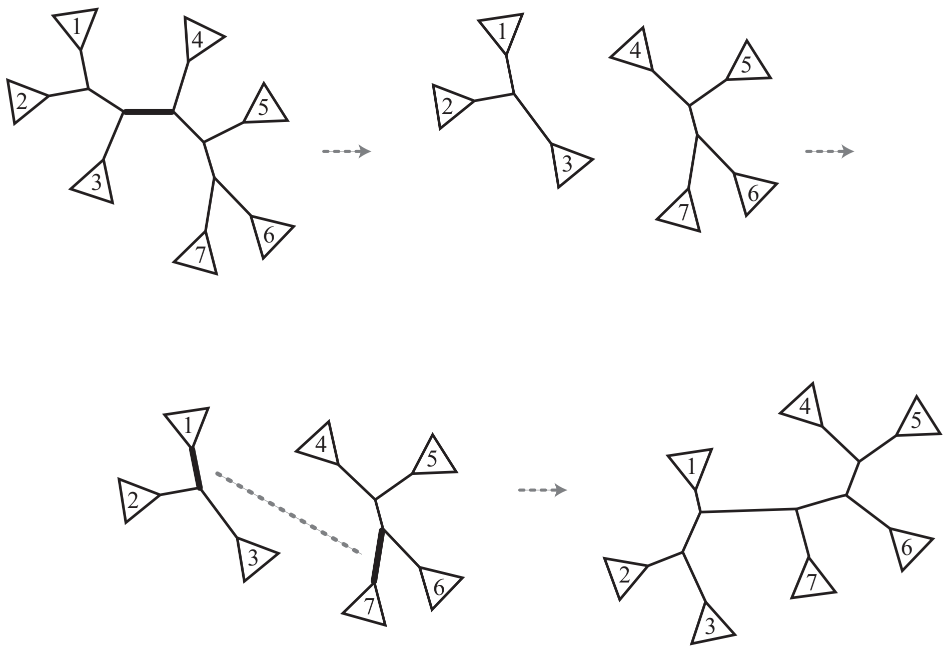

Figure 9.6: A TBR move on a tree.

Notice that while TBR moves are more general than SPRs, SPRs are special cases of TBRs. If a TBR is performed in such a way that we choose one of the edges that met the removed edge for a reconnection point, then the result of the TBR is the same as that of an SPR. Moreover, it is not too hard to see that any TBR can be performed by a succession of two SPRs (see exercises).

Counting the precise number of trees one TBR move away from a given tree is difficult. In fact, the count depends not only on the number of taxa on the tree, but also the tree topology, so there can be no formula of the sort in Theorem 17. However, we can estimate the order of this number fairly easily. To perform a TBR we must first chose an edge to remove, and then choose two other edges to reconnect, for a total choice of 3 edges. Since the number of edges grows linearly in n, choosing 3 edges should lead to something growing at most like n^3. Notice how this grows more quickly than the count for either NNIs or SPRs, as one would expect.

9.3 Searching Tree space

For either parsimony or maximum likelihood methods of tree inference, we must search tree space for the best tree using our chosen criterion. For a given tree, we have already explained how we can efficiently compute either the parsimony score (via the Fitch-Hartigan or Sankoff algorithms), or the maximum of the likelihood function over the numerical parameters of the model (using Felsenstein's pruning algorithm, and techniques of numerical optimization). But then we are faced with the problem that our criterion requires that we compute these scores for *every* tree to choose the best one. However, tree space is big, and in most circumstances an exhaustive search is simply not possible to do in an acceptable amount of time. Instead, we follow a more heuristic procedure.

A rough outline of a search procedure might be as follows:

1. Pick a type of move, say NNI.
2. Choose a tree as a starting point, and evaluate its score.
3. Consider all trees one move away from our current tree, and compute the score for each.
4. If no tree is better than the current one, stop.
5. Otherwise, choose the best tree of these, and move to it. Then go back to step 3.

The strategy here can be visualized in a very natural way. If we imagine each point in tree space as a point in a plane, then we can represent the tree's score as a height above this plane, so that all the scores form something like a surface over tree space., as cartoonishly depicted in Figure 9.7. Our goal is to maximize the score (at least for ML), or equivalently to find the highest point on the surface. We begin at some point on this surface (our initial tree) and look around at all the places we could reach by taking a single step. We then choose to go to the one that takes us the highest. In other words, we try to climb up by always going in the steepest direction. Sooner or later, we will reach the top of a hill, and will not be able to climb higher.

The problem with this strategy, of course, is that there may be several hills, of different heights. We may end up climbing to the top of a short hill, reaching a local maximum. Because our steps are of limited size, we might never be able to reach a higher hill without first descending. Thus there is no guarantee we will ever get to the global maximum. Our strategy is essentially a local one, only considering nearby points in tree space, and doesn't take a long-distance view.

We could try to consider more distant trees as well, perhaps by using an SPR instead of an NNI as our basic move. It will then take us longer to decide in which direction to move, since there will be more options. Moreover, as long as there are some trees we don't consider, we may still miss seeing the highest hill, and still end up climbing something shorter. So while using bigger steps

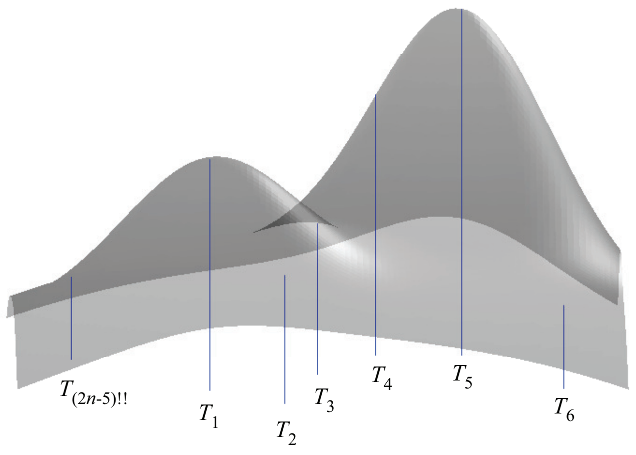

Figure 9.7: Hill climbing on tree space.

makes it less likely that we climb the wrong hill and become trapped at its top, it doesn't reduce the risk to zero.

The simplest solutions to this problem is to repeat the search many times, using different starting points. If the starting points are varied enough, we are more likely to eventually choose one near the highest hill, and climb to its top. There will still be no guarantee that we ever find the true optimum, but we make it more likely that we do so.

There are many variations on this basic search procedure that might result in reaching at least a local optimum faster. For instance, instead of considering all trees one move away from the current one before picking a better one, we could consider them one at a time, and move to a new one whenever we find it. It is hard to guess whether this would produce a faster search, since we may end up moving more often to get small improvements in the tree, when we might have been better off waiting to find the biggest improvement we can. Such a decision must be made in developing software, and may be based on observing how the software performs on real data, rather than on theoretical considerations.

But there are other decisions that may be left to the software user. For instance, should we use only NNI moves, or SPRs, or TBRs in a search? NNIs are small steps in tree space, so using them is likely to produce a very gradual improvement in the tree. The more radical moves of SPRs and TBRs are larger steps in tree space, and have the potential to produce greater improvements in a single step. However, there are many more of them to consider, so it may take longer to find a better tree. We might also end up taking too large a step, and miss the best tree before coming back to it later. One appealing approach is to begin using SPRs or TBRs to crudely search with big steps, taking a step every time we come across a better tree, and then switch to NNIs for a finer local search around the best trees previously found.

Most importantly, though, one should always be sure the search is performed repeatedly, from multiple starting points (or with some random component to the search). This is simply because there may be several locally optimal trees, and a single search may end at one of these even though it is not globally optimal. It is tempting of course to begin a search with a tree you believe may be close to optimal. For instance, since it is quick to perform Neighbor Joining, an NJ tree might be a natural starting point for a ML search. Certainly such a tree could be one of your starting points, since it is quite plausible that you will soon find the optimal tree from it. However, if that is your only starting tree, you may miss the best. The issue here is not simply one of a mathematical possibility that doesn't arise in practice — there are real data sets in which the likelihood has several local maxima on tree space. A single search can become trapped in a local maxima and never find the optimal tree.

There are other techniques that can sometimes be useful to prevent ourselves spending too much time searching areas of tree space that are unlikely to yield good trees. These are generally adaptations of standard ideas from computer science, such as *branch and bound*, which we will not discuss.

9.4 Metrics on Tree Space

It is desirable to have some natural way of quantifying how far apart we consider two trees to be. That is, we would like to have a metric on tree space. Ideally this metric would capture whatever informal idea we have of two trees being similar; a small value would mean two trees 'look mostly alike' and a large value that they are 'very different.'

Each of the moves we've discussed on tree space leads to a way of measuring the differences between two trees. First, it is not hard to see that any tree on a set of taxa can be reached from any other tree by some succession of these moves. Then, if it is possible to move from one tree to another using k moves, but not using fewer moves, we might say these trees are k apart.

Definition. For a fixed choice, $\mu \in \{\text{NNI, SPR, TBR}\}$, of move on tree space the *μ-edit metric* measures the distance between trees T and T', denoted $d_\mu(T, T')$, as the minimal value of $k \geq 0$ such that there exists a sequence of trees

$$T = T_0, T_1, T_2, \ldots, T_k = T'$$

with T_{i+1} one μ-move from T_i for each i.

As should be clear, these give very different notions of distance on tree space. For instance, one of the exercises demonstrates that for any $k \geq 1$ there are trees T and T' such that $d_{\text{SPR}}(T, T') = 1$, yet $d_{\text{NNI}}(T, T') = k$. On the other hand, if

$d_{\mathrm{NNI}}(T,T') = 1$, then $d_{\mathrm{SPR}}(T,T') = d_{\mathrm{TBR}}(T,T') = 1$ as well. More generally,

$$d_{\mathrm{NNI}}(T,T') \geq d_{\mathrm{SPR}}(T,T') \geq d_{\mathrm{TBR}}(T,T'),$$

since any NNI move is an SPR, and any SPR move is a TBR.

It is not clear how to compute any of these distance between trees in an efficient manner, since the definition suggests we consider all sequences of trees between the given two. Since in a sequence of minimal length there can be no repeated trees, there are a finite, though huge, number of possibilities to consider. It is known that computing the NNI- and TBR-edit distances are NP-hard problems. There are also interesting results on the the maximal distance between trees under these metrics, called the *diameter* of tree space. Understanding these diameters is important both for understanding how many steps might be needed to get from one tree to another, and also for interpreting the meaning of the distance between two trees in terms of similarity. For instance, if tree space has a small diameter under a given metric, the metric could only crudely measure similarity of trees.

A metric on tree space that is easier to compute is the *splits metric*, also called the partition metric, symmetric distance, or the Robinson-Foulds metric. Recall from Chapter 4 that associated to any edge of a tree T is a split of the taxa on its leaves; by removing the edge, the taxa are partitioned into two sets, according to the two resulting tree components,

Definition. The *splits metric* on X-trees measures the distance, $d_{\mathrm{splits}}(T,T')$, between two trees as the number of splits that occur on one tree or the other, but not both.

This is straightforward to compute, since we can simply list all the splits. Of course there is no need to include trivial splits corresponding to terminal edges, since these will appear on any tree. For instance, Figure 9.8 shows two trees with 4 different splits occurring among them, so these are distance 4 apart.

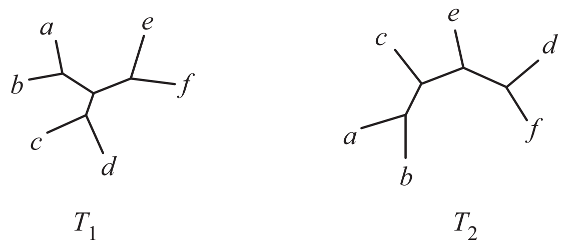

Figure 9.8: The tree T_1 has non-trivial splits $\{a,b\}|\{c,d,e,f\}$, $\{c,d\}|\{a,b,e,f\}$ and $\{e,f\}|\{a,b,c,d\}$. The tree T_2 has non-trivial splits $\{a,b\}|\{c,d,e,f\}$, $\{a,b,c\}|\{d,e,f\}$ and $\{a,b,c,e\}|\{d,f\}$. After deleting the split these trees have in common, four remain, so $d_{\mathrm{splits}}(T_1,T_2) = 4$.

While easier to compute, a drawback of the splits metric is that it can be quite large for two trees that we might consider to be similar in most of their features. Moving only a single taxon via an SPR move on a terminal edge can produce a tree that is very far from the original as measured by the splits metric. The misplacement of a single 'rogue taxon' is thus judged to have a huge effect on a tree. (See Exercise 9.)

Another metric on tree space that can be computed efficiently uses quartets instead of splits

Definition. The *quartet metric* on X-trees measures the distance, $d_{\text{quartet}}(T, T')$, between two trees as the number of 4-element subsets of X that induce different quartet trees on T and T'.

Since a collection of n taxa has $\binom{n}{4} = \frac{n(n-1)(n-2)(n-3)}{24}$ subsets of 4 taxa, one could simply list these and check whether the induced quartet trees are the same, to create a $\mathcal{O}(n^4)$ algorithm for computing this metric. However, there are more efficient ways to compute the quartet metric than this obvious approach, with one method only requiring $\mathcal{O}(n \log n)$ time.

The quartet distance is not affected as much as the split distance by a single taxon being moved from one part of the tree to another. This should not be surprising, since if a single taxon is moved, only those quartets involving that taxon are affected, so many quartets are left unchanged (See exercise 10.)

9.5 Metrics on Metric Tree Space

For searching through tree space in algorithms, moves from one topological tree to another are thought of as separate from modifications to edge lengths in a metric tree. But for measuring how close two metric trees are to one another it's desirable to have metrics on tree space that consider both topology and edge lengths.

One natural way to do this builds on the splits metric. Given n-taxa, there are $2^{n-1} - 1$ possible splits, and thus any topological tree can be represented by a vector of 0s and 1s, of length $d = 2^{n-1} - 1$; we simply choose some ordering of the possible splits and then record a 1 to indicate a split is displayed on the tree, or a 0 to indicate it is not. The splits metric between topological trees can then be found by subtracting their corresponding vectors (producing a vector with 1s and -1s indicating differing splits), taking the absolute value of each entry, and then adding them up.

In more mathematical language, the splits metric just involves mapping each tree to vector of 0s and 1s in $\mathbb{R}^d$, and then using the L^1 norm to compute distance:

$$||\mathbf{v} - \mathbf{w}|| = \sum_{i=1}^{d} |v_i - w_i|.$$

This immediately suggests the following metric splits distance: map each tree into $\mathbb{R}^d$ by recording the length of the edge of a tree associated to a split in the coordinate corresponding to that split, with a 0 in all other coordinates. Then use the L^1 norm on these vectors to compute distances between metric trees. A formula for distance between two X-trees is thus

$$d_{ms}(T_1, T_2) = \sum_{\text{splits } e} |w_1(e) - w_2(e)|.$$

Here we use e to denote a split of X, with $w_i(e)$ being the length of the corresponding edge in T_i if the split is displayed on T_i, and $w_i(e) = 0$ if the split is not displayed on e_i.

One can replace the use of the L^1 norm here with any other norm on $\mathbb{R}^2$. A very natural choice is to use the Euclidean one, giving

$$\tilde{d}_{ms}(T_1, T_2) = \left(\sum_{\text{splits } e} (w_1(e) - w_2(e)) \right)^{1/2}.$$

Notice that in the case of topological trees, whether one uses the Euclidean or L^1 norm, the same tree metric is computed, but for metric trees these give different results.

9.6 Additional Remarks

Although we've motivated this look at tree space by the need to search for optimal trees, the concepts of this chapter are useful for other issues.

SPR moves appear naturally in modeling *recombination* of genes. Whether this recombination occurs through sexual reproduction, or by lateral gene transfer across species, the result is essentially that a subtree has been pruned from one location on a tree, and regrafted elsewhere. Of course there are constraints on where the regrafting can occur, since recombination can occur only between organisms alive at the same time. Thus the most proper framework requires we work with rooted metric trees for this application

Having chosen a metric on tree space also makes it possible to define a *median tree* for a collection of trees. This is another form of a consensus tree, which might be used to as a summary statistic for a collection of trees. Just as the median of a set of numbers is the one 'in the middle,' a median tree T_m is one that minimizes the total distance to all trees in the collection. That is, for a collection $\{T_1, T_2, \dots, T_k\}$,

$$T_m = \arg\min_T \left(\sum_{i=1}^{k} d(T, T_i) \right).$$

9.7 Exercises

1. The text's explanation of the NNI move claimed that there were exactly 3 ways 4 edges could be grouped into 2 pairs. Explain this.

2. The text describes an alternate way of thinking of an NNI move, as sliding one edge down another past a junction of edges. It seems that since we could pick any of 4 edges to 'slide', and for each have a choice of 2 places to slide it to, there should be 8 possible new trees we could get doing this. Explain why there are only 2. (You might just consider each of the 8 possibilities, and group them by their outcomes.)

3. For the 5-taxon tree $((A,B),C,(D,E))$, draw all 4 trees that can be obtained from it by a single NNI move.

4. For 5-taxa, give an example of two trees that are 2 NNI moves apart, but only 1 SPR apart.

5. Generalize the last problem to produce an example of two trees with a large number of taxa that are n NNI moves apart, but only 1 SPR apart.

6. Check all the the arithmetic leading to part (ii) of Theorem 17.

7. How many 5-taxon unrooted binary trees are there? For any of these trees, how many trees are 1 SPR away? Draw all the trees that are 2 or more SPR moves from $((A,B),C,(D,E))$.

8. Explain how any TBR move can be performed by a succession of two SPR moves.

9. Consider the two caterpillar trees with n-taxa,

$$T_1 = (\dots((A_1,A_2),A_3),A_4)\dots,A_n),$$
$$T_2 = (\dots((A_1,A_n),A_2),A_3),\dots,A_{n-1}).$$

   a) Explain how T_2 can be obtained from T_1 by a single SPR move.

   b) What is the splits metric distance between the two trees?

   c) For comparison, what is the maximal distance between two n-taxon trees under the splits metric? What fraction of this is your answer to part (b)?

10. Consider the trees of Exercise 9

    a) What is the quartet metric distance between the two trees?

    b) For comparison, what is the maximal distance between two n-taxon trees under the quartet metric? What fraction of this is your answer to part (a)?

11. Suppose a single SPR move is performed on a large tree.

    a) Describe which splits of the original tree remain in the result. (Hint: Think about which edges are associated to the splits that are lost. The locations of these can be described using geometric language.)

    b) Show that if $d_{\text{SPR}}(T,T') = 1$ then $2 \leq d_{\text{splits}}(T,T') \leq 2(\operatorname{diam} T - 2)$, where $\operatorname{diam} T$, the *diameter* of T, is the length of the longest path in T.

12. Prove the μ-edit metric satisfies the requirements to be a metric on tree space: 1) $d_\mu(T_1, T_2) \geq 0$, and $d_\mu(T_1, T_2) = 0$ only when $T_1 = T_2$, 2) $d_\mu(T_1, T_2) = d_\mu(T_2, T_1)$, and 3) $d_\mu(T_1, T_3) \leq d_\mu(T_1, T_2) + d_\mu(T_2, T_3)$.

13. The presentations of the moves in this chapter can be formalized to use the more technical language of graph theory. Each modifies the set of vertices and edges defining a tree in specific ways. Describe these precisely for:

    a. NNI

    b. SPR

    c. TBR

14. Explain why the L^1 norm and the square of the Euclidean norm give rise to the same metric on *topological* tree space in the construction given in section 9.5.

Chapter 10

Rate-variation and Mixture Models

All the Markov models of DNA mutation introduced thus far assume that every site in the sequences behaves identically; of course 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. Even in coding regions, it may well be that the functional constraints on different parts of a protein molecule vary, so that parts of a gene may evolve at a different rate from other parts. These functional constraints might even change over evolutionary time, so that the behavior of a site may vary from one part of the tree to another.

In general, then, it is desirable to expand the substitution models developed so far, in order to incorporate variation among the sites. It is also important that this be done so that we need not decide in advance which sites behave similarly. Seldom will we have the detailed knowledge to choose an *a priori* classification of sites into different classes. Instead, we use *mixture models* which posit different classes of behavior, but leave the proportion of sites assigned to the different classes as parameters. The class sizes, as well as the parameters determining the behavior of each class, will then be inferred from the data.

10.1 Invariable Sites Models

The simplest form of a rate-variation models is built on two classes of sites. The first class of *variable* sites mutates according to a model, such as the GTR, of the sort discussed earlier, and the second class of *invariable* sites undergoes no mutation at all. When we examine DNA data arising from such a model, if we observe a substitution at a site, then that site is certainly in the first class.

However, if we do not observe any change we generally cannot tell which class the site came from. It may have been free to mutate but did not due to the randomness in the model, or it may have been invariable. Thus the sites that will be observed as unvarying are a larger set than the invariable ones. As a result, we will not be able to easily tell which sites were invariable in order to remove them from data before we analyze it. Instead, we must formalize the model more carefully.

For concreteness, consider a model with invariable sites 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 s, indicating that with probability s a site is variable, and with probability $1-s$ it is invariable. Finally, for the invariable sites we need another distribution vector $\mathbf{p}_{inv}$ indicating the state distribution for the invariable sites. 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.

Note that we could have further assumed the variable and invariable sites had the same state distribution, so $\mathbf{p}_{inv} = \mathbf{p}_\rho$. With that approach, the only new parameter introduced over the GM model would have been s, the class size parameter.

In order to see the relationship between the GM+I model parameters and the joint distribution 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, Chapters 6 and 8 described how to produce the entries of the joint distribution 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 joint distribution array will be n-dimensional, of size $\kappa \times \kappa \times \cdots \times \kappa$. So suppose we've found this array P_1, where $P_1(i_1, i_2, \dots, i_n)$ gives the probability that a variable site shows the pattern $(i_1, i_2, \dots, i_n)$ of states at the leaves.

For the invariable sites we could do a similar calculation, using the root distribution vector $\mathbf{p}_{inv}$ and the identity matrix I for all Markov matrices on edges, since this matrix describes no possibility of substitutions occurring. However, it should be clear that such a computation would produce an n-dimensional $\kappa \times \kappa \times \cdots \times \kappa$ array P_2 such that

$$P_2(i_1, i_2, \dots, i_n) = \begin{cases} q_{i_1} & \text{if } i_1 = i_2 = \cdots = i_n, \\ 0 & \text{otherwise.} \end{cases}$$

That is, the joint distribution for this class gives zero probability for any patterns in which states vary, and probability equal to the state distribution for those that show no variation.

The joint distribution for the full GM+I model is then a weighted sum of the distributions for the two classes,

$$P = sP_1 + (1 - s)P_2. \tag{10.2}$$

With $0 \leq s \leq 1$, the weights s and $1 - s$ here are simply the relative sizes of the classes, or equivalently the probabilities that a randomly chosen site is in a given class. Statistical models such as this, in which several classes are modeled separately, but then combined so that we allow the class to be determined randomly, are called *mixture models*.

There are straightforward variations on this idea using other models in place of GM. For instance, a general time reversible model with invariable sites (GTR+I), simply replaces the Markov matrices of the GM model above by the appropriate exponentials of a rate matrix. The JC+I and K2P+I are defined similarly.

If we believe a mixture model such as GTR+I describes our data, how should this affect our methods of inference?

Note that the results of a parsimony analysis are unaffected by invariable sites as they give uninformative patterns. Thus parsimony is a reasonable method under the same circumstances as it is for variable sites alone. (However, most users of parsimony never mention any model, since the method makes no explicit use of one.)

In contrast, our derivations of distance formulas for the various models all tacitly assumed no invariable sites are present. If only a small fraction are likely to be invariable, we may view the distances as approximately valid. Nonetheless, the use of the model-based distance formulas essentially require that the number of invariable sites be negligible. If this is not the case, there can be a systematic bias in the values of inferred distances between taxa. For the Jukes-Cantor model it is not hard to work out exactly how this effect occurs (see exercises).

In fact, it is relatively easy to prove that even under the model JC+I the proportion of invariable sights cannot be estimated if we only allow the comparison of 2 sequences at a time. That means there can be no way to compute distances for the model JC+I, unless we already know the proportion of invariable sites by some other means. (There are ways to estimate this proportion by comparing larger numbers of sequences at a time, which have been implemented in some distance-based software.)

Fortunately, the statistical framework of maximum likelihood handles a model such as GTR+I with little additional complication. A few additional numerical parameters must be varied as we search for the maximum of the likelihood function on a given tree, but no more substantive changes are needed. It is in this setting that such models are typically used, since a more accurate model of the true substitution process is hoped to lead to more accurate inference.

10.2 Rates-Across-Sites Models

It is now easy to 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.

The most common multiclass mixture is built on a GTR model, in which all sites use the same rate matrix, but scaling factors are introduced to slow down or speed up the process at individual sites.

For a fixed tree T, the parameters will be as follows. We assume a common GTR rate matrix Q for all edges of the tree, 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 *rate scaling parameters* $\lambda_1, \lambda_2, \dots, \lambda_m$, which will serve as factors to speed up or slow down the substitution process for the different classes. We also need a vector $\mathbf{r} = (r_1, r_2, \dots, r_m)$, with entries adding to 1, giving the relative sizes of the rate classes.

Now sites in the ith rate class will evolve using the rate matrix $\lambda_i Q$ throughout the tree. Thus the larger the value of λ_i, the faster substitutions will occur. 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 is now straightforward to compute P_i, the joint distribution array at the leaves for the ith class, using approaches discussed in earlier chapters.

The last step is to combine the distributions from the various classes to get the joint distribution for the mixture model,

$$P = \sum_{i=1}^{m} r_i P_i. \tag{10.3}$$

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

$$P = \int_\lambda r(\lambda) P_\lambda \, d\lambda.$$

Since P_λ is a multidimensional array, the integral here is meant to be performed entry-wise, so it represents a separate integral for each entry, just as the summation on equation (10.3) represents a separate sum for each entry.

In current practice, it is common to use a Γ distribution of rates (with mean 1), whose density function is given by the formula

$$r(\lambda) = \frac{\alpha^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\alpha\lambda}.$$

This is a flexible distribution, whose precise shape varies according to the value of α. It is useful for describing quantities that may vary over the positive numbers, which is the appropriate range for the rate scaling parameters λ. The *shape parameter* α of the GTR+Γ model then adds only one additional parameter to those of the GTR model, yet allows for a better fit to data.

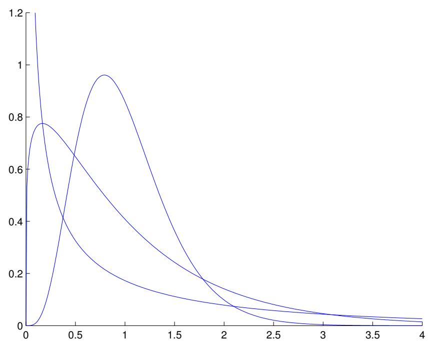

Figure 10.1: The Γ distribution for $\alpha = .3$, 1.2, 4.8, in order from top to bottom at left edge of the graph.

Figure 10.1 shows graphs of the density function $r_\alpha(\lambda)$ for three values of α. When $\alpha < 1$, the distribution has a spike at low values, and a long tail at high ones. This means some sites are nearly invariable, and there is a very wide spread of rates for the others, with some sites evolving quite quickly. For $\alpha > 1$, the distribution has a hump around 1, which becomes taller and narrower as α is increased. Thus for large α, such a distribution will produce a model little different from a standard GTR. Thus by varying α from very large to very small the model may produce behavior ranging from very similar to a straight GTR model, to one in which sites evolve at a very broad spectrum of rates.

Finally, it is easy and often desirable to add an additional class of invariable sites, producing the model designated GTR+Γ+I. When the GTR+Γ and GTR+Γ+I models are fit to the same data set, the first of these models often requires a much smaller value of α than the second. This should seem reasonable, as a low value of α enables the model to capture some invariable, or nearly invariable sites. Explicitly including an invariable class allows the α to be chosen to better fit the more variable sites.

Although the GTR+Γ and GTR+Γ+I are the models most commonly used for routine data analysis, it is important to realize that there is no clear biological justification for the use of a Γ-distribution. There is no mechanistic model of how substitutions occur that leads to the particular form of Γ, or of any other distribution. It is simply an arbitrary choice that provides flexibility with the addition of only one more parameter.

In fact, it is not really the Γ-distribution, but rather a discretization of it with a finite number of rate classes, that is used in most software implementations of maximum likelihood. Typically only a handful of rate classes, say four or five, are used. If, for instance, there are four classes, then the cutoffs for the lowest 25%, 50%, and 75% of the distribution are determined. Then the means of λ in each of the resulting four intervals are computed. These four values are then used as the rate scaling factors λ_i for four classes each of proportion 0.25. Experience with data analysis has shown that using more classes seldom leads to much of an improvement in likelihood scores, and by keeping the number of classes smaller, computations can be done a bit more quickly. Nonetheless a model using this approximation might be more appropriately designated GTR+d$\Gamma(4)$ to emphasize that it in fact uses a 4-class discretized Γ.

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 been implemented in all commonly used software packages. The motivation, which arose in a paper of Fitch and Markowitz in 1970, is biologically quite an attractive one, even though the mathematical formulation does not capture the full process they described.

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 leads to some sites mutating at different rates than others are thus imagined to be unchanging across all lineages throughout the tree.

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.'

In more recent usage for describing probabilistic substitution models the term '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 down the tree. More generally, the terminology is used when characters may undergo a switching of a certain form between any sorts of different classes, such as different rate classes.

Note that the motivation of Fitch and Markowitz for a covarion model is essentially an argument *against* an assumption of independence of the mutation process at different sites. The reason they give why some sites change from being invariable to variable is because of changes in other parts of the gene. However, it is quite unclear how to formulate such a model in a mathematically tractable way. Thus the mathematical formulation of the covarion model will still assume sites behave independently, but will include a switching mechanism so that sites may change their behavior.

The simplest form of the covarion model, introduced by Tuffley and Steel in 1998 uses only 2 classes, one of which is invariable. 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 an 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.

With the ordering of bases as listed above, we interpret the entries of $\tilde{Q}$ as follows. The upper left block $Q - s_1 I$ describes changes from 'on' bases to other 'on' bases. It has the same off-diagonal entries as Q, so these represent the usual rates of substitutions from one base to another, as in a GTR model. The upper right block, $s_1 I$, describes changes from 'on' bases to 'off' bases. For instance, A^{on} switches to A^{off} with rate s_1. Since this block is diagonal, when an 'on' to 'off' instantaneous switch occurs, the nucleotide may not change simultaneously. Note the diagonal entries of the upper left block were adjusted from those of Q by subtracting off $s_1 I$ simply because a rate matrix must have rows adding to 0.

The lower left block of $\tilde{Q}$ describes 'off' bases switching to 'on'. Note that these switches occur with rate s_2. Again, when a switch occurs, the base may not change at that instant. The lower right block should describe base changes from 'off' bases to 'off' bases. Since 'off' means no such change can occur, the off-diagonal entries of this block are all zero. The diagonal entries are then chosen so that the rows of $\tilde{Q}$ add to 0.

We next must choose an 8-element root distribution for the model. Letting

$$\sigma_1 = \frac{s_2}{s_1 + s_2}, \quad \sigma_2 = \frac{s_1}{s_1 + s_2},$$

and

$$\tilde{\mathbf{p}} = (\sigma_1 \mathbf{p}, \sigma_2 \mathbf{p}),$$

one can check (Exercise 7) that

$$\tilde{\mathbf{p}}\tilde{Q} = \tilde{\mathbf{0}}, \tag{10.4}$$

and thus $\tilde{\mathbf{p}}$ 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 8).

Now for any tree T, with a root ρ chosen arbitrarily, we have an 8-state time-reversible 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.)

To incorporate this into the covarion model we 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

$$\begin{pmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 \end{pmatrix} H = \begin{pmatrix} p_1+p_5 & p_2+p_6 & p_3+p_7 & p_4+p_8 \end{pmatrix},$$

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.

Now for internal edges of the tree we make the Markov matrices be the M_e as described above, while for edges leading to leaves use M_eH. (Here we assume we have located the root of the tree at an internal vertex.) While M_eH is not square, it still has non-negative entries, with rows summing to 1, and so has 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 ways discussed in Chapter 8.

The basic example of a covarion model extends naturally to allow for more classes. For instance, one might formulate a covarion model with 3 rate classes, 'fast,' 'medium,' and 'slow.' If Q is a GTR rate matrix for the fast class, then for some rate scaling parameters $\lambda_1 = 1 > \lambda_2 > \lambda_3 \geq 0$, and some switching parameters $s_{ij} \geq 0$, we construct a 12×12 rate matrix

$$\tilde{Q} = \begin{pmatrix} \lambda_1 Q - (s_{12}+s_{13})I & s_{12}I & s_{13}I \\ s_{21}I & \lambda_2 Q - (s_{21}+s_{23})I & s_{23}I \\ s_{31}I & s_{32}I & \lambda_3 Q - (s_{31}+s_{32})I \end{pmatrix}.$$

While some constraints (which we omit here) must be placed on the s_{ij} so that this leads to a time-reversible model, a stable base distribution can be given. Finally, the matrix H which performs the hiding of the rate class will consist of three stacked identity matrices.

Note that because these covarion models have a stable state distribution, the fraction of sites within a given rate class is constant over the tree. Thus while an individual site may change its class, the model assumes that other sites are also changing classes in a way that keeps the class sizes balanced.

Another feature to note is that if all of the switching parameters in a covarion model are $s_{ij} = 0$, then the model reduces to a more standard mixture model in which sites may not change classes. For instance, in the 3-class example above, setting switching parameters to 0 gives $\tilde{Q}$ a simpler block-diagonal form which means only in-class base substitutions are possible.

10.4 General Mixture Models

Once the idea of using a mixture model has arisen to capture rate variation across sites (which includes models with "+I", "+Γ", and "+I+Γ", as well as the covarion models), there is no reason to not consider more complicated mixtures. One could imagine several classes of sites, with each class having essentially unrelated parameters describing its base substitution process.

A simple example would have two classes, each of which evolves according to the GTR model on the same topological tree, but with possibly unrelated rate matrices and branch lengths. Parameters for such a model would be the topological tree, two complete sets of numerical parameters for the GTR model including rate matrices Q_1 and Q_2 and edge lengths, and one additional mixing parameter r that determines the class proportions r, $1-r$. The joint distribution for such a model is simply the weighted sum of the distributions for the two classes, just as in equation (10.2). In fact, the GTR+I model is a submodel of this one, in which we have decided ahead of time that the rate matrix Q_2 is the zero matrix, so that characters in it never change. A two-class rates-across-sites model is also a special case of this one, in which we require that the two rate matrices be multiples of one another, $Q_2 = \lambda Q_1$.

One can easily extend this type of model to have a larger number of classes. Since each additional class increases the number of parameters, though, the more classes that are used, the longer sequences will have to be to obtain good estimates of parameter values. Though models with a large number of classes are unlikely to ever be used for routine data analysis, there has been a fair amount of recent interest in exploring their behavior on real data sets. If there is reason to believe a less standard model is inappropriate, they offer the next step in modeling complexity.

While the mixtures described so far have all had only one topological tree among their parameters, one can also consider models where each class has its own tree parameter. This might be useful, for instance, if either hybridization or some form of lateral gene transfer had occurred so that different parts of the sequences had evolved on different trees. Though these models have been studied theoretically, they have not yet been explored for practical data analysis.

When a complex model is formulated, it is possible that the model has been made so complex that it is mathematically impossible to use it to validly infer parameters. More precisely, the parameter values may not be able to be uniquely *identifiable* from the theoretical distribution of the model, much less from an approximation of it from data. For instance, it might be that two different tree topologies give rise to exactly the same distribution, so that one could not decide which of them led to a given data set. While several papers in the last few years gave alarming indications that mixture models can be problematic in this way, the most recent theoretical work has shown it should not be a real worry to a practitioner. In fact, the number of classes that can be safely used for theoretical good behavior grows with the number of taxa, and for even a moderate number of taxa is much larger than is likely to be used in practice.

Perhaps the most extreme mixture model is one in which every site is allowed to behave differently. In the framework of assuming a single tree topology for all sites, but no more commonality among them, this is usually called the *no common mechanism model.* As a statistical model it is not very useful, since the number of parameters grows with the length of the sequences. As a result longer sequences do not lead to a better ability to recover parameters using Maximum Likelihood, since each additional site introduces many new parameters.

No common mechanism models were introduced primarily to gain theoretical understanding of the interrelationship of different inference methods. Tuffley and Steel (1997) showed that ML inference using a JC no common mechanism model chooses the same tree as Parsimony. Unfortunately this was misinterpreted by some as a justification for Parsimony. Since a no common mechanism model (which doesn't *require* that the sites follow different models, but *allows* it) is perhaps closer to the truth than standard ones, they argued this result showed Parsimony was justified by the standard ML statistical framework. Unfortunately they overlooked the point that ML inference itself was not justified for this model, since the growth in parameters with the number of sites invalidated all the mathematical justifications for ML.

10.5 Exercises

1. Explain the count in formula (10.1).

2. Explain the formula in equation (10.2) by thinking of P_1 and P_2 as giving conditional probabilities of patterns. (Conditioned on what?)

3. How many parameters are needed for a GTR+I model on a binary n-taxon tree, assuming we want the invariable sites to have the same distribution as the stable distribution of variable ones?

4. If data are produced according to by a GTR+I model, but analyzed according to a (misspecified) GTR model, one might expect that the edge lengths of a tree would be estimated to be shorter than they actually were. Why? Explain informally.

5. Suppose a pattern distributions is produced from a JC+I model, with r the proportion of variable sites. With substitution rate $\alpha = 1$ and edge length t_e, what is the matrix giving the joint distribution of patterns on a 1-edge

tree? If the Jukes-Cantor distance (for a model *without* invariable sites) is used to infer the edge length, does it give t_e? If not, is the estimate it gives larger or smaller than the true value?

6. Show that any joint distribution on a 1-edge tree arising from the JC+I model *exactly* matches a joint distribution for the JC model. This means that from 2-taxon comparisons alone it will not be clear whether one needs to include invariable sites in a model used to describe a data set.

7. Show equation (10.4) holds.

8. Show the 8×8 rate matrix $\tilde{Q}$ of the Tuffley-Steel covarion model together with the root distribution vector $\tilde{\mathbf{p}}$ form a time-reversible model.

9. Explain why if M_e is an 8×8 Markov matrix, and H the 8×4 matrix composed of two stacked identity matrices then M_eH will have non-negative entries, with rows summing to 1. In the context of the Tuffley-Steel covarion model, give an interpretation of its entries as conditional probabilities. Also give an interpretation of the entries of H as conditional probabilities.