## Lecture 6A – Introduction to Trees and Optimality Criteria: Parsimony

I. To begin our treatment of estimating phylogeny, I want to first introduce some generalities about trees, so that we'll have a common starting point, and then discuss the various optimality criteria that one may choose for evaluating trees.

A. Some Basic Terms. There are two sets of terms that are commonly used: one by mathematicians and one by systematists.

In a sense, the mathematicians' terms are more appropriate because phylogenetic trees fall within the purview of graph theory, however, having been trained as a systematist, I think in those terms.
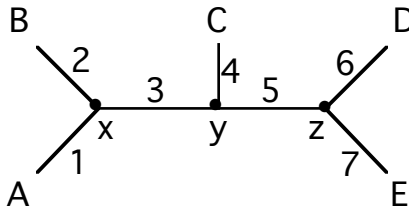
1. Branches: If there are n-taxa in an unrooted tree, there are $2n$-3 branches, labeled 1 – 7.

    - 1, 2, 4, 6, & 7 are external, or terminal branches (leaves in math jargon)
    - 3 & 5 are internal branches (edges); these represent splits, or bipartitions of taxa.

2. Nodes are points where branches intersect or terminate.

    Nodes A – E are terminals, or terminal nodes (there are n), whereas nodes x, y, & z are internal nodes (there are n-2 internal nodes in an unrooted tree).

3. We can denote trees in parenthetical notation (sometimes called New Hampshire or Newick format). This would be represented by ((A,B),C,(D,E)).

4. We can break trees into sub-trees. If we break branch 3, we have two sub-trees (A,B) and (C,(D,E)).
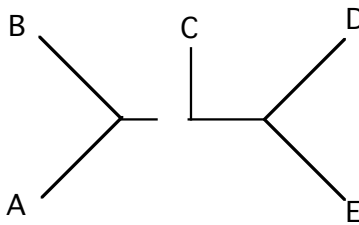
B. Rooting – The tree above is an unrooted tree.

This implies a lack of directionality, such that one may traverse the tree in any direction one finds convenient and start the traversal at any point.

Many phylogenetic analyses estimate unrooted trees, and the root is usually determined by the inclusion of an outgroup. The point at which the outgroup connects with the ingroup is the root.

This is usually included in the estimation, not simply added after an ingroup topology is estimated (as I'll do in this demonstration).

All of the rooted trees below can be derived from the unrooted tree by placing the root (common ancestor) on each branch. So, there are $2n-3$ possible ways to root the tree (10 - 3=7 for our 5-taxon example) and these rooted trees *do* imply directionality; nodes deeper in the tree represent divergence events that happened earlier. Furthermore, this rooting exercise illustrates how rapidly the number of alternative trees grows as one adds taxa. Addition of a single node (the root node) resulted in the seven rooted trees below being derivable from the single unrooted tree above.

A B C D E
1

B A C D E
2

A B C D E
3

C A B D E
4

C A B D E
5

A B C E D
6

A B C D E
7
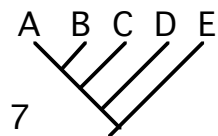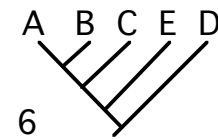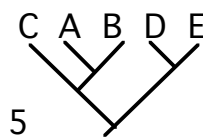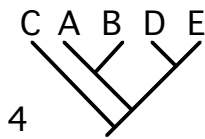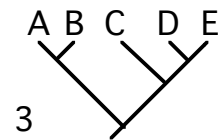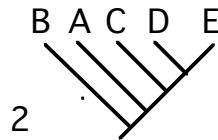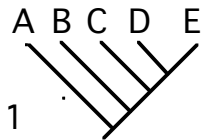
All are consistent with the unrooted tree we started with, even though they imply very different relationships.

Also note that there is free rotation around nodes:
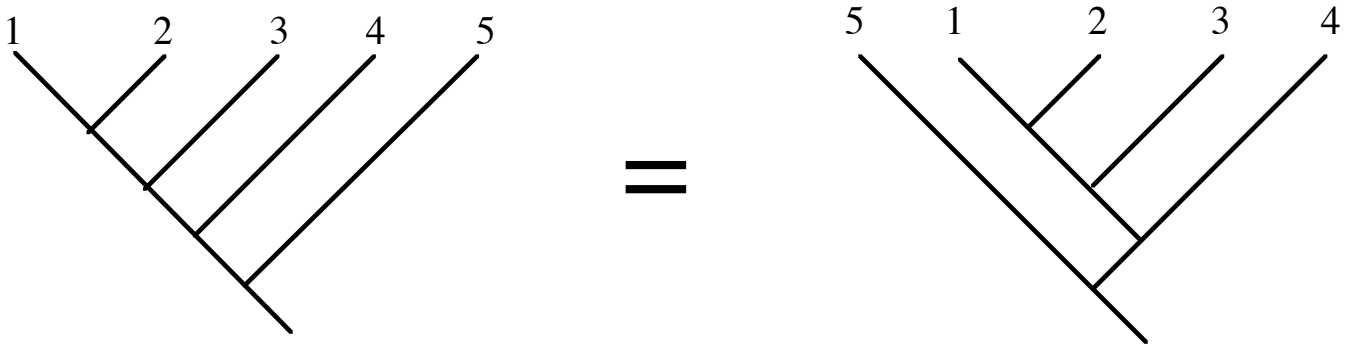
C. The scope of the phylogeny problem – tree space is staggeringly vast.

One other important point can be seen by the rooting exercise.

The number of possible trees increases incredibly fast as the number of taxa increases.

We can think of rooting as adding another taxon to this tree. There are 7 places we can add the 6th taxon and still be congruent with this 5-taxon tree, and there are 15 possible 5-taxon trees. So, there are 7 * 15 = 105 possible 6-taxon trees.

What this means is that the number of possible alternative trees will quickly be far too vast for us to examine every possible tree for a phylogenetic question involving very many taxa.

The number of possible trees for N taxa is given by:

$$B(N) = \prod_{i=3}^{N} (2i - 5)$$

.

Number of Taxa	Number of Unrooted Trees
3	1
4	3
5	15
6	105
7	945
8	10,395
9	135,135
10	2.027×10^6
22	3×10^{23}
50	3×10^{74}
100	2×10^{82}
1000	$2 \times 10^{2,860}$
10,000,000	$5 \times 10^{68,667,340}$

So, the task of phylogeny inference is indeed formidable, which is why the idea that phylogenetic trees can be "*made easy*" (Hall, B. 2017. *Phylogenetic Trees Made Easy*, Oxford University Press, Oxford) is nonsensical.

Some approaches to phylogeny estimation don't try at all to search among these alternatives but rely on a series of rules to build one. These are referred to as algorithmic methods.

Other approaches actually try to search this vast tree space in an effort to find the best tree under some optimality criterion.

II. Optimality Criteria – Before we deal with how to choose from among all these trees, we first need to know how to evaluate any one tree.

Both algorithmic and optimality approaches rely on being able to evaluate a particular tree.

A. Parsimony – This formerly was the most frequently used criterion. It's certainly the easiest to compute, it's also (recall) the only philosophically acceptable criterion of the journal *Cladistics* (Editors. 2016. Cladistics, 32:1).

First, assuming characters are independent, the parsimony score of a tree (i.e., its length, L_τ) for the entire data set is given by:

$$L_\tau = \sum_{i=1}^{s} w_i l_i$$

That is, it's the sum of the weighted (by w_i) character lengths, l_i, across all s characters. For now, we'll consider equal weights (i.e., all $w_i = 1$).

There are two ways one may calculate the l_i's

A. The Fitch Algorithm applies to matrix with all character-state transformations equal and operates on **state sets** and **accumulated lengths**.
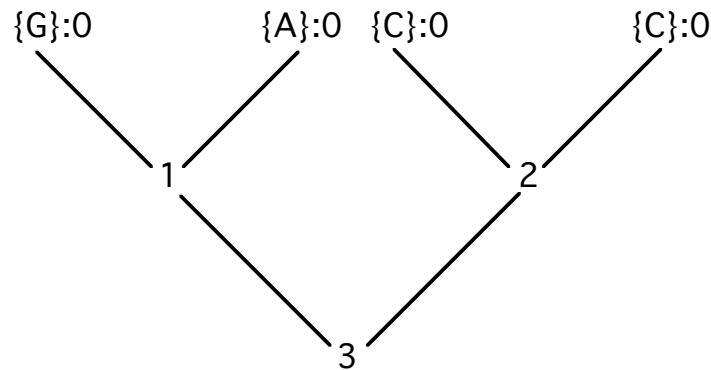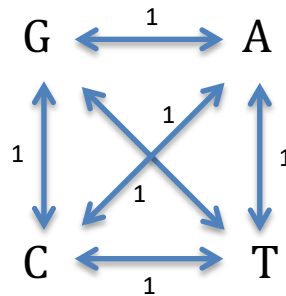

There are four terminals for this (arbitrarily rooted) tree: one with G, one with A and two with C. We erect a state set at each terminal node and assign an accumulated length of zero to terminal nodes. This is the minimum number of changes in the daughter subtree.

We then proceed down the tree to internal nodes and calculate the state sets for each, as follows:

1 – Form the intersection of the state sets of the two daughter nodes. If the intersection is non-empty, assign the set for the internal node equal to the intersection. The

accumulated length of the internal node is the sum of the accumulated lengths of the daughter nodes.

2 - If the intersection is empty, we assign the union of the two daughter nodes to the state set for the internal node. The accumulated length is **the sum of those of the daughter nodes plus one.**

{G}:0　　　　{A}:0　{C}:0　　　　　{C}:0

{G,A}:1　　　　　　　{C}:0

{G,A,C}:2
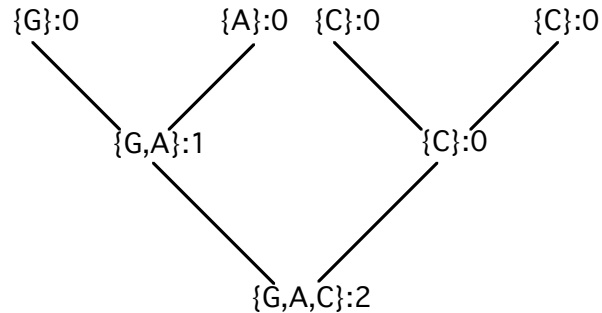
Therefore, this character has a minimum length of 2 steps for this tree. In addition to determining the length of this tree for this character, we've also assigned possible ancestral states at each of the nodes.

Remember that this is only applicable to characters for which transformations among all character states are unordered and have the same cost.

So $l_1 = 2$ (the length of character 1) would be entered into the summation:

$$L_\tau = \sum_{i=1}^{s} w_i l_i$$

And the parsimony score of this particular tree is the weighted sum across all the characters in the character by taxon matrix (i.e., the alignment, for sequence data).

2. Sankoff's Algorithm is more general and is applicable for any type of character.

Let's assume that we think the following transformation matrix (step matrix) is appropriate:

	A	C	G	T
A	--	4	1	4
C	4	--	4	1
G	1	4	--	4
T	4	1	4	--

The convention is to use an alphabetical order, and c_{ij} is the cost (in # of steps) to transform from state i to state j. It was common practice in molecular phylogeny (until ~2000's) to use a non-uniform matrix (called a step matrix), and this one is symmetric. This step matrix implies that transversions are expected to be ca. $4x$ as reliable as transitions.
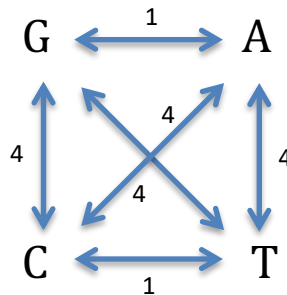

Following the Sankoff algorithm, we erect a **character-state vector** at each node and find the cost of assigning each possible state at each node.

For each node k, we calculate the cost associated with each possible character-state assignment to that node, $s_k(i)$ this is analogous to the accumulated cost above, in that it represents the cost of all changes leading to the daughter nodes, given assignment of state i to node k.

Again, we start at the tips, and the terminal vectors are filled with either 0, if the state is present or a dot (or infinity, or an asterisk…), if the state is absent in that taxon.

Since we know states at the terminal nodes, we only need worry about the cost of transformations to the state present. Therefore, the vector at node 1 is filled in as such:
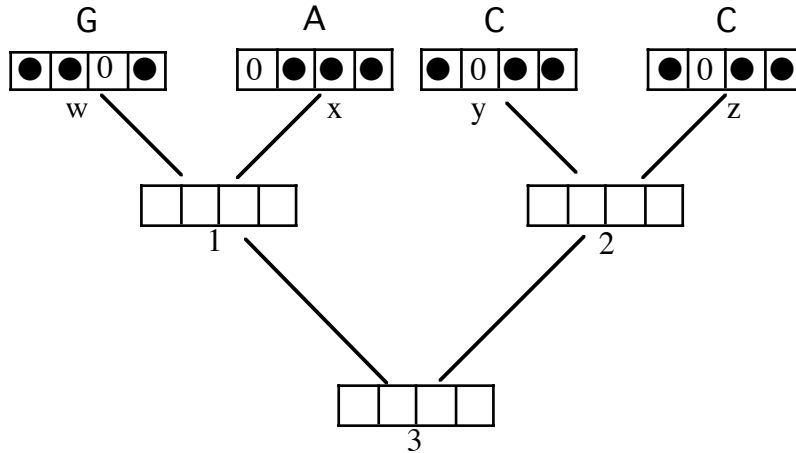
$$s_{1(A)} = c_{AG} + c_{AA} = 1 + 0 = 1,$$
$$s_{1(C)} = c_{CG} + c_{CA} = 4 + 4 = 8,$$
$$s_{1(G)} = c_{GG} + c_{GA} = 0 + 1 = 1$$
$$s_{1(T)} = c_{TG} + c_{TA} = 4 + 4 = 8$$

Similarly, the node 2 character-state vector is as follows:
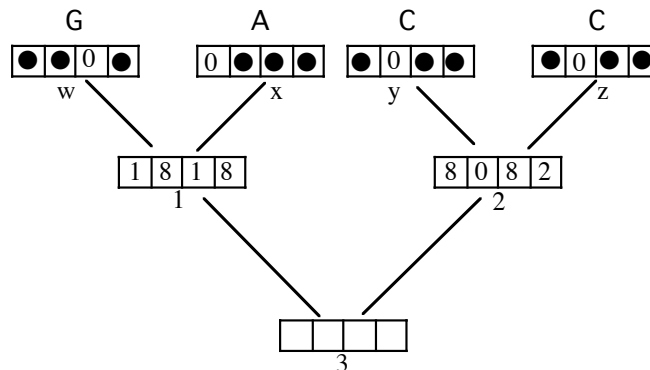$$s_{2(A)} = 4 + 4 = 8$$
$$s_{2(C)} = 0 + 0 = 0$$
$$s_{2(G)} = 4 + 4 = 8$$
$$s_{2(T)} = 1 + 1 = 2$$

Now, since we don't have sequences for nodes 1 & 2, when we calculate the cells of the node 3 vector, we have to consider all possible character states for each.

Thus, the cells of the character-state vector incorporate the minimum values for each possible reconstruction of its immediate daughter nodes.

$s_{3(A)} = \min[s_{1A} + c_{Aj}] + \min[s_{2A} + c_{Aj}]$
$\quad = \min\,[\mathbf{1},12,2,12] + \min[8,\mathbf{4},9,6] = 1 + 4 = 5$

$s_{3(C)} = \min[s_{1C} + c_{Cj}] + \min[s_{2C} + c_{Cj}]$
$\quad = \min\,[\mathbf{5},8,\mathbf{5},9] + \min[12,\mathbf{0},12,3] = 5 + 0 = 5$

$s_{3(G)} = \min[s_{1G} + c_{Gj}] + \min[s_{2G} + c_{Gj}]$
$\quad = \min\,[2,12,\mathbf{1},12] + \min[9,\mathbf{4},8,6] = 1 + 4 = 5$

$s_{3(T)} = \min[s_{1T} + c_{Tj}] + \min[s_{2T} + c_{Tj}]$
$\quad = \min\,[\mathbf{5},9,\mathbf{5},8] + \min[12,\mathbf{1},12,2] = 5 + 1 = 6$
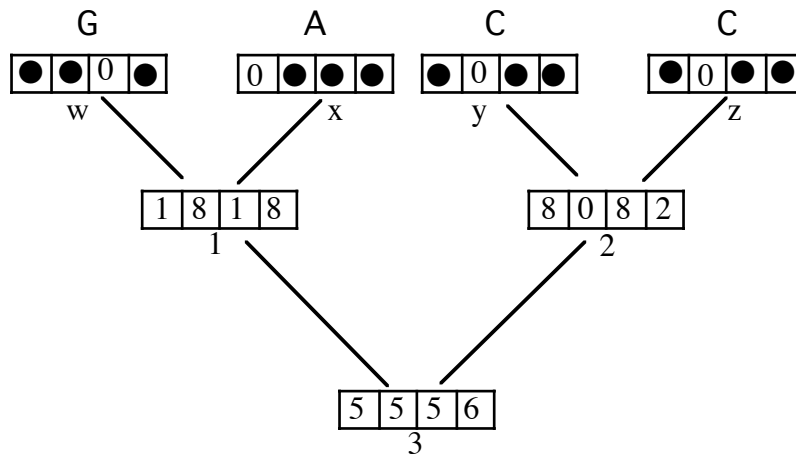
Now, the node-3 character-state vector is defined, and all the nodes have been addressed:

Thus, this character has a minimum cost of 5 steps under a 4:1 weighting of transversions.

Again, we do this for all characters on this tree and a weighted sum taken across characters to compute the parsimony score of this tree.

Points to note:
   1) Two types of weighting are possible: weighting of transformations within characters (which we demonstrated with the step matrix) and weighting among characters, which are reflected in the weighted sum of lengths across characters.

   2) One can't compare tree lengths across weighting schemes. In the first example with all transformations having the same cost, the length of the character on this tree was 2. In

the second, with a 4:1 step matrix to weight transversions more heavily, the length was 5. This prevents us from objectively comparing weighting schemes.