Computing a tree: II

Genome 559: Introduction to Statistical and Computational Genomics

Prof. James H. Thomas

Parsimony algorithm

1) Construct all possible trees

2) For each informative site in alignment count changes on each tree

3) Add them all up for each tree

4) Pick the lowest scoring

Distance trees

- Measure pairwise distance between each pair of sequences.

- Use a clustering method to build up a tree, starting with the closest pair.

	1	2	3	4	5	6
human	a	g	t	c	t	c
chimp	a	g	a	g	t	c
gorilla	c	g	g	c	a	g
orangutan	c	g	g	g	a	c

human - chimp has 2 changes out of 6 sites □
human - orang has 4 changes of out 6 sites
etc.

Distance matrix from alignment

	human	chimp	gorilla	orang
human	0	2/6	4/6	4/6
chimp		0	5/6	3/6
gorilla			0	2/6
orang				0

(symmetrical, lower left not filled in)

Distance matrix methods

- Methods based on a set of pairwise distances typically from a multiple alignment.

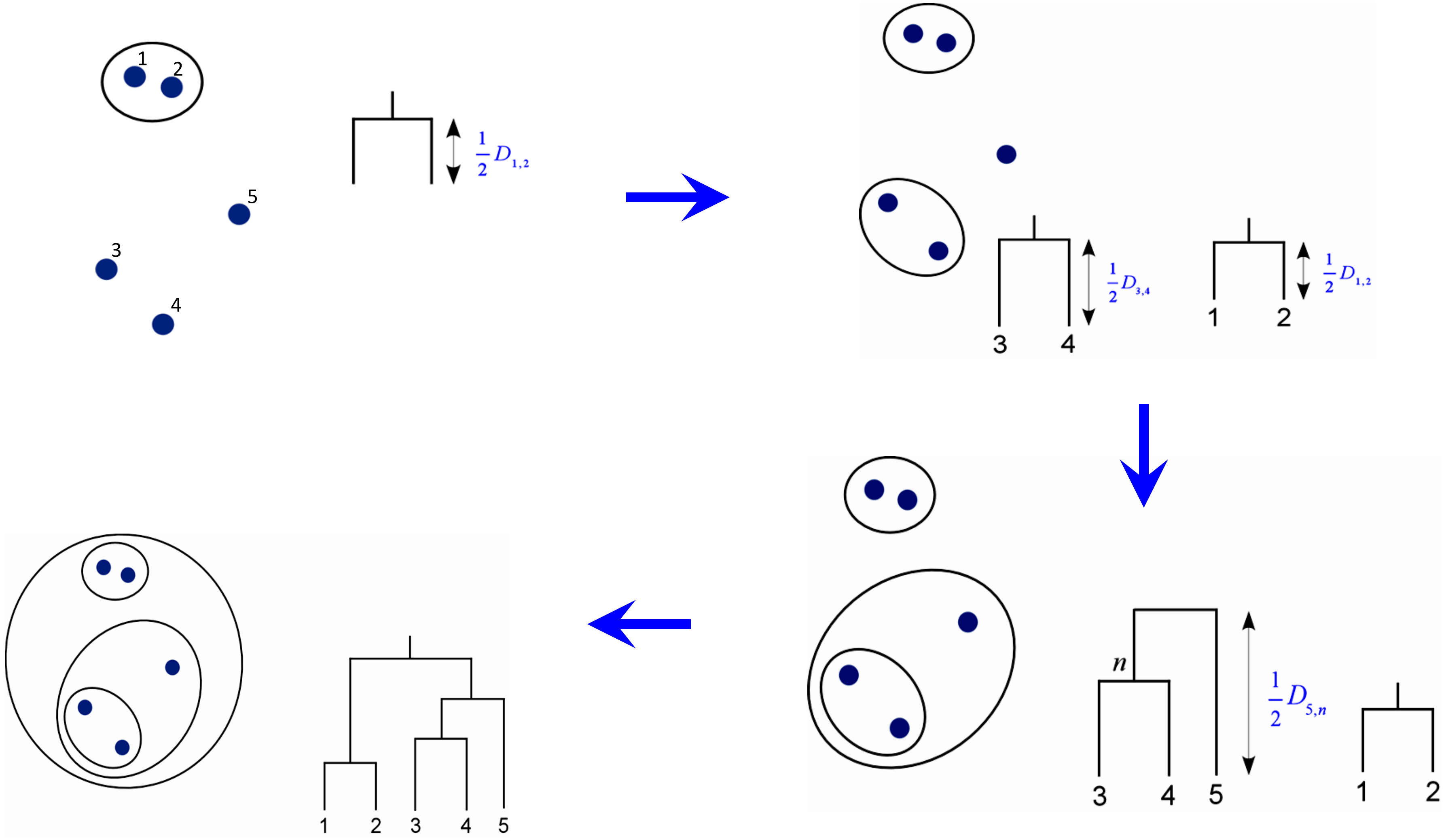
- All methods try to build the tree that best matches the distances.

- Usual standard for "best match" is the least squares of the tree distances compared to the real pairwise distances:

Let D_m be the real distances and D_t be the tree distances.
Find the tree that minimizes:

$$\sum_{i=1}^{N} \left(D_t - D_m \right)^2$$

Sequential clustering approach (UPGMA)



Sequential clustering algorithm

1) generate a table of pairwise sequence distances and assign each sequence to a list of N tree nodes.

2) look through the current list of nodes (initially these will all be leaf "nodes") for the pair with the smallest distance.

3) merge the closest pair, remove the pair of nodes from the list and add the merged node to the list.

4) repeat until there is only one node left - it is the root.

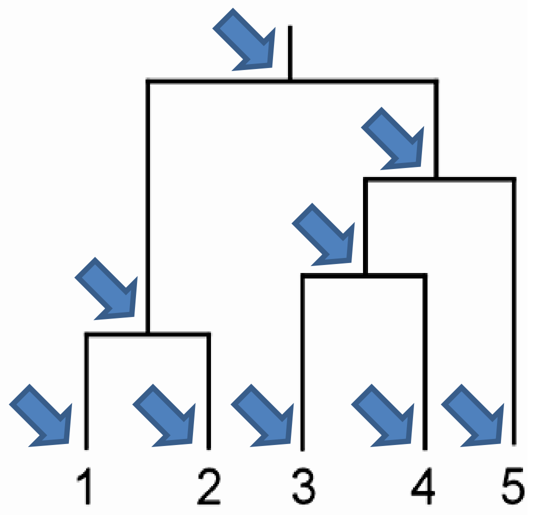
$$D_{n1,n2} = \frac{1}{N} \sum_{i} \sum_{j} d_{ij}$$

where i is each leaf of $n1$ (node1), j is each leaf of $n2$ (node2), and N is the number of distances summed

(in words, this is the arithmetic average of the distances between all the leaves in one node and all the leaves in the other node)

Data structure for a tree

```
class TreeNode:
    <parent node>
    <left-child node>
    <right-child node>
```



The tree itself is made up of `TreeNode` objects, each of which is connected to other `TreeNode` objects based on its three attributes.

A leaf (or tip) has no child nodes. A root has no parent node. All the rest have all three.

You could also add a "name" attribute, which would be assigned only to leaf nodes based on the sequence name for that leaf.

Neighbor-Joining Algorithm (side note)

Essentially as for UPGMA, but correction for distance to other leaves is made.

Specifically, for two leaves *i* and *j*, we denote the set of all other leaves as L, and the size of that set as $|L|$, and we compute the corrected distance D_{ij} as:

$$D_{ij} = d_{ij} - (r_i + r_j)$$

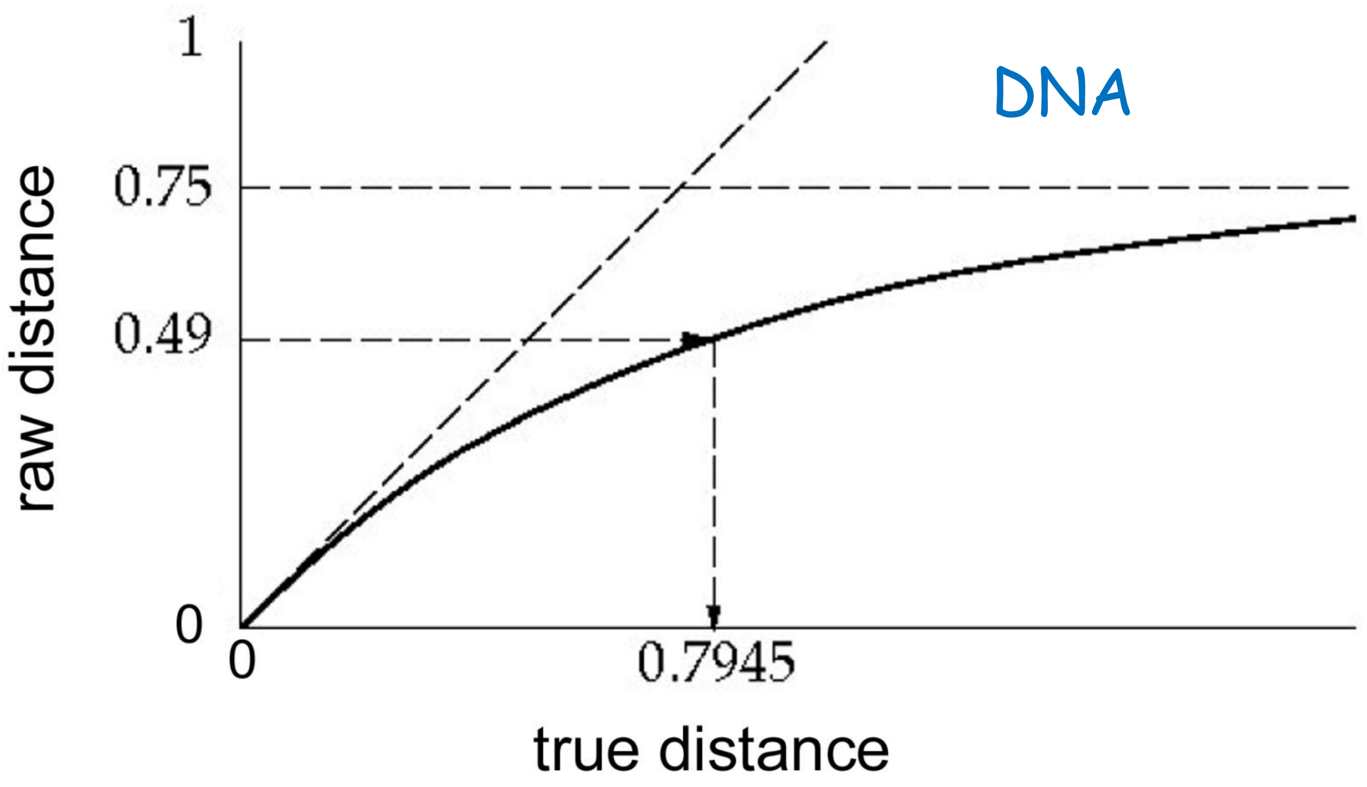
where

$$r_i = \frac{1}{|L|} \sum_{k \in L} d_{ik}$$

(the mean distance from i to all other leaves)

Raw distance correction

- As two DNA sequences diverge, it is easy to see that their maximum raw distance is ~0.75 (assuming equal nt frequencies).
- This graph shows evolutionary distance related to raw distance:



Mutational models for DNA

- Jukes-Cantor (JC) - all mutations equally likely.
- Kimura 2-parameter (K2P) - transitions and transversions have separate rates.
- Generalized Time Reversible (GTR) - all changes have separate rates.

(Models similar to GTR are also available for protein)

Jukes-Cantor

	G	A	T	C
G	$1-3\alpha$	α	α	α
A	α	$1-3\alpha$	α	α
T	α	α	$1-3\alpha$	α
C	α	α	α	$1-3\alpha$

Kimura 2-parameter

	purines		pyrimidines	
	G	A	T	C
G	$1-\alpha-2\beta$	α	β	β
A	α	$1-\alpha-2\beta$	β	β
T	β	β	$1-\alpha-2\beta$	α
C	β	β	α	$1-\alpha-2\beta$

Jukes-Cantor model - distance correction

Jukes-Cantor model:

$$D = -\frac{3}{4}\ln(1-\frac{4}{3}D_{raw})$$

D_{raw} is the raw distance (what we directly measure)

D is the corrected distance (what we want)

Distance trees - summary

- Convert each pairwise raw distance to a corrected distance.

- Build tree as before (UPGMA or neighbor-joining).

- Notice that these methods don't consider all tree topologies - they are very fast, even for large trees.