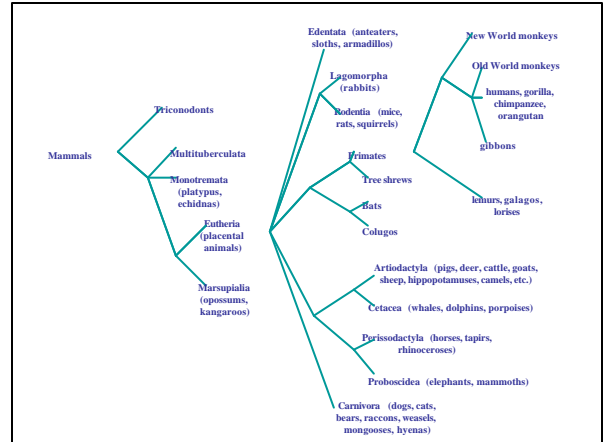
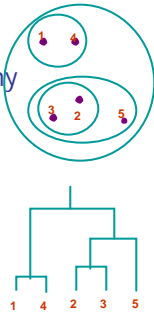


Overview of Phylogeny



Phylogenies

Phylogenies are trees that show the history of life

Genes, repeats, etc., are also connected in phylogenies

Orthologs: Two elements that have diverged because of speciation

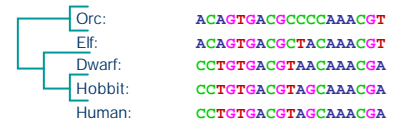
Paralogs: Two elements that have diverged because of duplication

Inferring Phylogenies

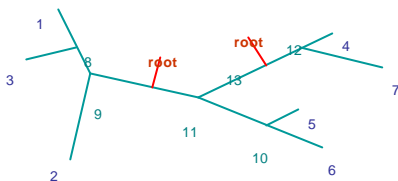
Trees can be inferred:

- Morphology of the organisms
- Sequence comparison

Example:



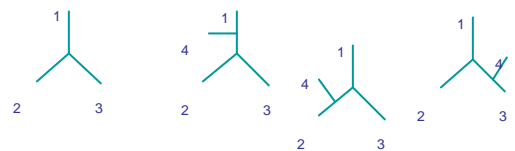
Background on trees



- Each node has three edges (binary)
- Labeled
- Each edge has a length (evolution time)
- Unrooted, or rooted

N leaves \Rightarrow $2N - 2$ nodes unrooted;
 $2N - 1$ nodes rooted

Space of possible trees



1 unrooted tree of 3 leaves
 3 unrooted trees of 4 leaves

...
 $3 \times 5 \times 7 \times \dots \times (2N - 5) = (2N - 5)!!$ unrooted trees with N leaves
 $(2N - 3)!!$ rooted trees with N leaves

Phylogeny and sequence comparison

Basic principles:

- Degree of sequence difference is proportional to length of independent sequence evolution
- Only use positions where alignment is pretty certain – avoid areas with (too many) gaps

Distance between two sequences

Given (portion of) sequences x^i, x^j ,

Define

d_{ij} = distance between the two sequences

One possible definition:

d_{ij} = fraction f of sites u where $x^i[u] \neq x^j[u]$

Better model (Jukes-Cantor):

$d_{ij} = -\frac{3}{4} \log(1 - \frac{4}{3} f)$

A simple clustering method for building tree

UPGMA (unweighted pair group method using arithmetic averages)

Given two disjoint clusters C_i, C_j of sequences,

$$d_{ij} = \frac{1}{|C_i| \times |C_j|} \sum_{(p \in C_i, q \in C_j)} d_{pq}$$

Note that if $C_k = C_i \cup C_j$, then distance to another cluster C_l is:

$$d_{kl} = \frac{d_{il}|C_i| + d_{jl}|C_j|}{|C_i| + |C_j|}$$

Algorithm: UPGMA

Initialization:

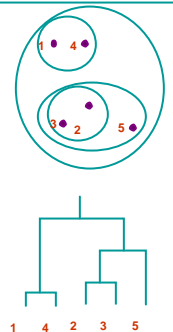
Assign each x_i into its own cluster C_i
Define one leaf per sequence, height 0

Iteration:

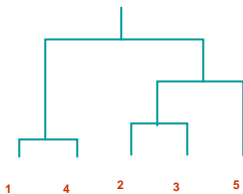
Find two clusters C_i, C_j s.t. d_{ij} is min
Let $C_k = C_i \cup C_j$
Define node connecting C_i, C_j , height $d_{ij}/2$
Delete C_i, C_j

Termination:

When all sequences belong to one cluster



Ultrametric distances & UPGMA



A distance measure is **ultrametric** if for any points i, j, k ,

Either all three distances are equal: $d_{ij} = d_{jk} = d_{ki}$,

Or two of them are equal and one is smaller: $d_{jk} < d_{ij} = d_{ki}$

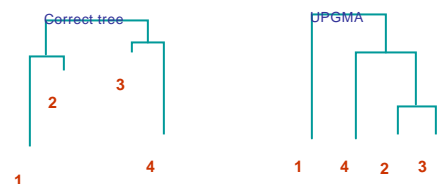
UPGMA is guaranteed to build the correct tree if distance is ultrametric

Weakness of UPGMA

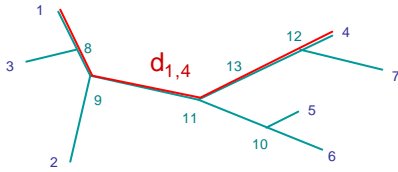
Molecular clock: implied time is constant for all species

However: certain species (e.g., mouse, rat) evolve much faster

Example where UPGMA messes up:



Additivity of distance



Given a tree, a distance measure is additive if the distance between any pair of leaves is the sum of lengths of edges connecting them

A maximum likelihood distance measure is additive given a large amount of data

Neighbor-joining

- Guaranteed to produce the correct tree if distance is additive
- May produce a good tree even when distance is not additive

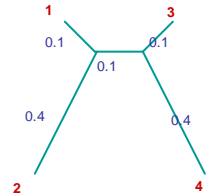
Step 1: Finding neighboring leaves

Define

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

Where

$$r_i = \frac{1}{|L| - 2} \sum_k d_{ik}$$



Algorithm: Neighbor-joining

Initialization:

Define T to be the set of leaf nodes, one per sequence
Let $L = T$

Iteration:

Pick i, j s.t. D_{ij} is minimal
Define a new node k , and set $d_{km} = \frac{1}{2} (d_{im} + d_{jm} - d_{ij})$ for all $m \in L$

Add k to T , with edges of lengths $d_{ki} = \frac{1}{2} (d_{ij} + r_i - r_j)$
Remove i, j from L ;
Add k to L

Termination:

When L consists of two nodes, i, j , and the edge between them of length d_{ij}

Rooting a tree, and definition of *outgroup*

Neighbor-joining produces an unrooted tree

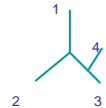
How do we root a tree between N species using n -J?

An **outgroup** is a species that we know to be more distantly related to all remaining species, than they are to one another

Example: Human, mouse, rat, pig, dog, chicken, whale

Which one is an outgroup?

Outgroup can act as a root



Parsimony

- One of the most popular methods

Idea: Find the tree that explains the observed sequences with a minimal number of substitutions

Two computational subproblems:

1. Find the parsimony cost of a given tree (easy)
2. Search through all tree topologies (hard)

Traditional parsimony

Given a tree, given a column u of a multiple alignment:

Initialization:

Set cost $C = 0$; $k = 2N - 1$

Iteration:

If k is a leaf, set $R_k = \{x^k[u]\}$

If k is not a leaf,

Let i, j be the daughter nodes;

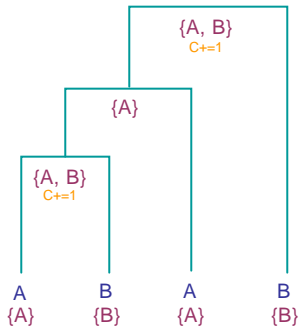
Set $R_k = R_i \cap R_j$ if intersection is nonempty

Set $R_k = R_i \cup R_j$, and $C += 1$, if intersection is empty

Termination:

Minimal cost of tree for column u , $= C$

Example



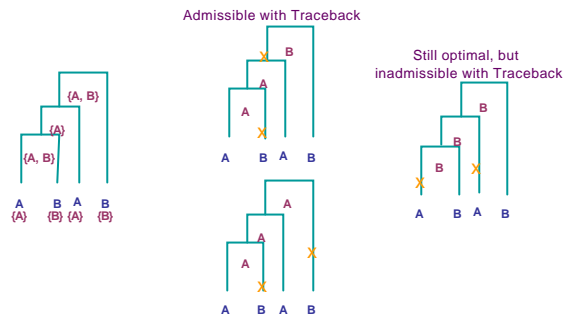
Traceback to find ancestral nucleotides

Traceback:

1. Choose an arbitrary nucleotide from R_{2N-1} for the root
2. Having chosen nucleotide r for parent k ,
If $r \in R_i$ choose r for daughter i
Else, choose arbitrary nucleotide from R_i

Easy to see that this traceback produces some assignment of cost C

Example



Weighted parsimony

Let $S_k(a)$ = minimal cost for the assignment of a to node k

Initialization:

Set $k = 2N - 1$

Iteration:

If k is a leaf:

$$S_k(a) = 0 \text{ for } a = x^k[u], S_k(a) = \infty \text{ otherwise}$$

If k is not a leaf:

$$S_k(a) = \min_i [S_i(b) + s(a, b)] + \min_j [S_j(c) + s(a, c)]$$

Termination:

Minimal cost of tree = $\min_a S_{2N-1}(a)$

Search through tree topologies: Branch and Bound

Observation: adding an edge to an existing tree can only increase the parsimony cost

Enumerate all unrooted trees with at most n leaves:

$$[i_3][i_5][i_7] \dots [i_{2N-5}]$$

where each i_k can take values from 0 (no edge) to k

At each point keep C = smallest cost so far for a complete tree

Start B&B with tree $[1][0][0] \dots [0]$

Whenever cost of current tree T is $> C$, then:

- T is not optimal
- Any tree with more edges containing T , is not optimal:
Increment by 1 the rightmost nonzero counter

Bootstrapping to get the best trees

Main outline of algorithm

1. Select random columns from a multiple alignment – one column can then appear several times
2. Build a phylogenetic tree based on the random sample from (1)
3. Repeat (1), (2) many (say, 1000) times
4. Output the tree that is constructed most frequently

Modeling Evolution

During infinitesimal time Δt , there is not enough time for two substitutions to happen on the same nucleotide

So we can estimate $P(x | y, \Delta t)$, for $x, y \in \{A, C, G, T\}$

Then let

$$S(\Delta t) = \begin{bmatrix} P(A|A, \Delta t) & \dots & P(A|T, \Delta t) \\ \dots & \dots & \dots \\ P(T|A, \Delta t) & \dots & P(T|T, \Delta t) \end{bmatrix}$$

Modeling Evolution

Reasonable assumption: multiplicative
(implying a stationary Markov process)

$$S(t+\Delta t) = S(t)S(\Delta t)$$

$$\text{That is, } P(x | y, \Delta t) = \sum_z P(x | z, \Delta t) P(z | y, \Delta t)$$

Jukes-Cantor: constant rate of evolution

$$\text{For short time } \epsilon, S(\epsilon) = \begin{bmatrix} 1 - 3\alpha\epsilon & \alpha\epsilon & \alpha\epsilon & \alpha\epsilon \\ \alpha\epsilon & 1 - 3\alpha\epsilon & \alpha\epsilon & \alpha\epsilon \\ \alpha\epsilon & \alpha\epsilon & 1 - 3\alpha\epsilon & \alpha\epsilon \\ \alpha\epsilon & \alpha\epsilon & \alpha\epsilon & 1 - 3\alpha\epsilon \end{bmatrix}$$

Modeling Evolution

Jukes-Cantor:

For longer times,

$$S(t) = \begin{bmatrix} r(t) & s(t) & s(t) & s(t) \\ s(t) & r(t) & s(t) & s(t) \\ s(t) & s(t) & r(t) & s(t) \\ s(t) & s(t) & s(t) & r(t) \end{bmatrix}$$

Where we can derive:

$$r(t) = \frac{1}{4} (1 + 3 e^{-4\alpha t})$$

$$s(t) = \frac{1}{4} (1 - e^{-4\alpha t})$$

Modeling Evolution

Kimura:

Transitions: A/G, C/T

Transversions: A/T, A/C, G/T, C/G

Transitions (rate α) are much more likely than transversions (rate β)

$$S(t) = \begin{bmatrix} r(t) & s(t) & u(t) & s(t) \\ s(t) & r(t) & s(t) & u(t) \\ u(t) & s(t) & r(t) & s(t) \\ s(t) & u(t) & s(t) & r(t) \end{bmatrix}$$

Where

$$s(t) = \frac{1}{4} (1 - e^{-4\beta t})$$

$$u(t) = \frac{1}{4} (1 + e^{-4\beta t} - e^{-2(\alpha+\beta)t})$$

$$r(t) = 1 - 2s(t) - u(t)$$