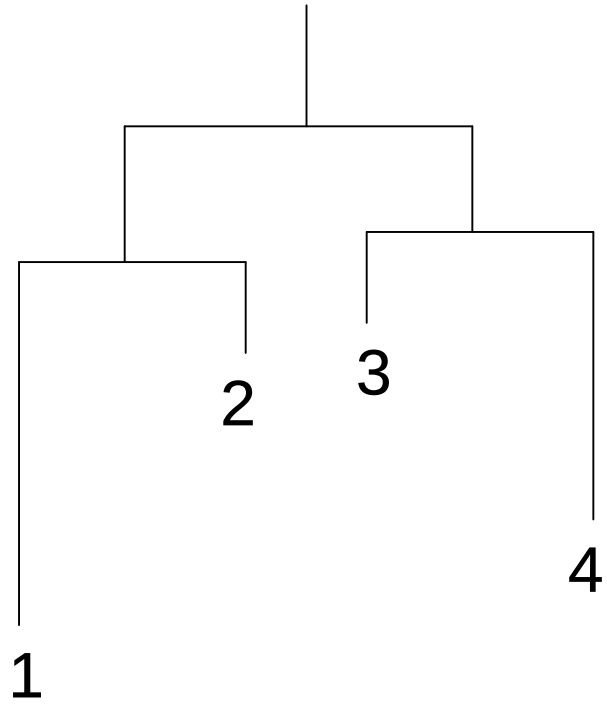
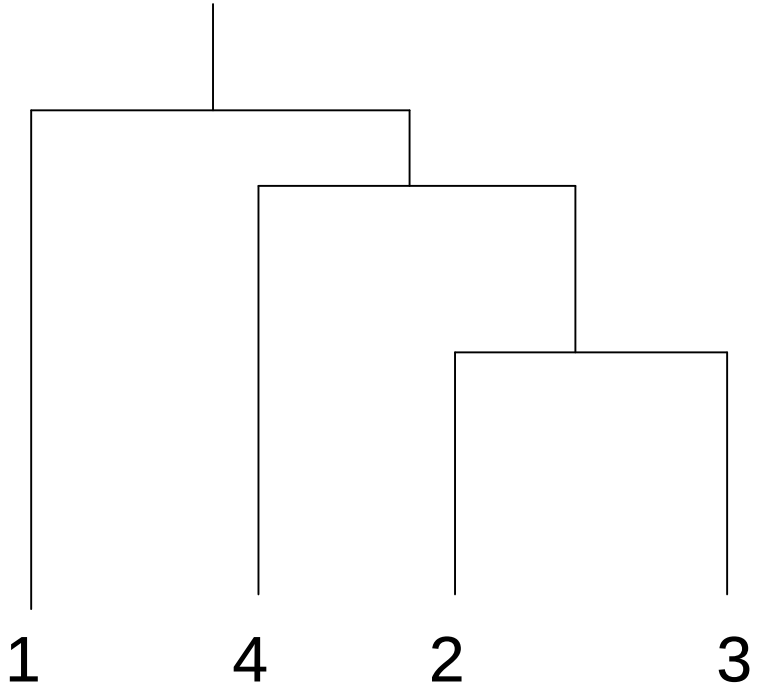
Incorrect reconstruction of non-ultrametric data by UPGMA



Tree which corresponds to non-ultrametric distances

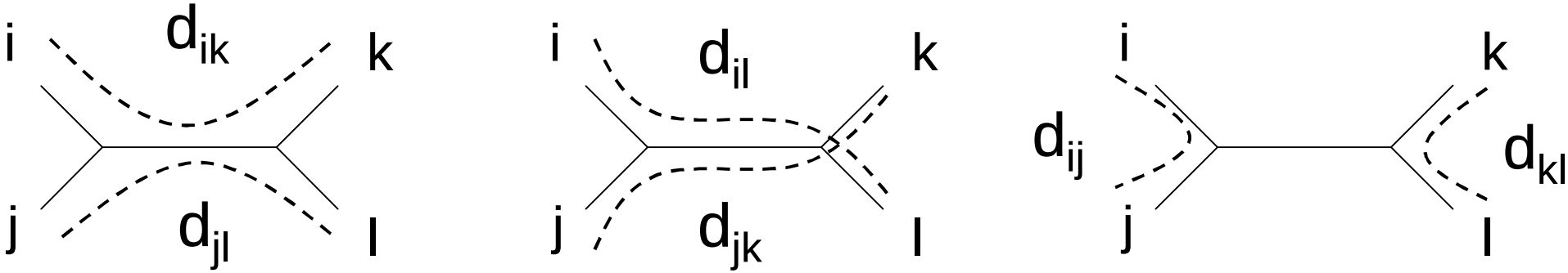


Incorrect ultrametric reconstruction by UPGMA algorithm

Checking for additivity

- How can we check if our data is additive?
- Let i, j, k and l be four *distinct* species
- Compute 3 sums: $d_{ij} + d_{kl}$, $d_{ik} + d_{jl}$, $d_{il} + d_{jk}$

Four-point condition



- The sums are represented by the three figures
  - Left and middle sum cover all edges, right sum does not
- *Four-point condition*: i, j, k and l satisfy the four-point condition if two of the sums $d_{ij} + d_{kl}$, $d_{ik} + d_{jl}$, $d_{il} + d_{jk}$ are the same, and the third one is smaller than these two

Checking for additivity

- An n x n matrix D is additive if and only if the four point condition holds for every 4 distinct elements $1 \leq i, j, k, l \leq n$

Finding an additive phylogenetic tree

- Additive trees can be found with, for example, the neighbor joining method (Saitou & Nei, 1987)
- The neighbor joining method produces unrooted trees, which have to be rooted by other means
  - A common way to root the tree is to use an outgroup
  - Outgroup is a species that is known to be more distantly related to every other species than they are to each other
  - Root node candidate: position where the outgroup would join the phylogenetic tree
- However, in real-world data, even additivity usually does not hold very well

Neighbor joining algorithm

- Neighbor joining works in a similar fashion to UPGMA
  - Find clusters C_1 and C_2 that minimise a function $f(C_1, C_2)$
  - Join the two clusters C_1 and C_2 into a new cluster C
  - Add a node to the tree corresponding to C
  - Assign distances to the new branches
- Differences in
  - The choice of function $f(C_1, C_2)$
  - How to assign the distances

Neighbor joining algorithm

- Recall that the distance d_{ij} for clusters C_i and C_j was

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

- Let $u(C_i)$ be the separation of cluster C_i from other clusters defined by

$$u(C_i) = \frac{1}{n-2} \sum_{C_j} d_{ij}$$

where n is the number of clusters.

Neighbor joining algorithm

- Instead of trying to choose the clusters C_i and C_j closest to each other, neighbor joining at the same time
  - Minimises the distance between clusters C_i and C_j and
  - Maximises the separation of both C_i and C_j from other clusters

Neighbor joining algorithm

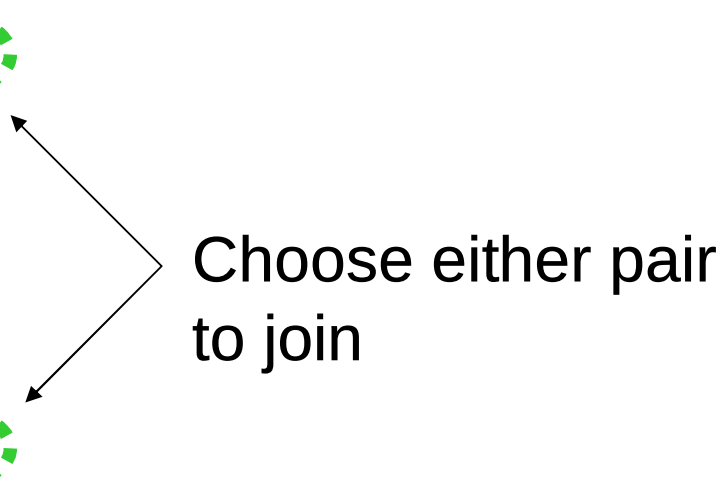
- Initialisation as in UPGMA
- Iteration
  - Find clusters i and j for which $d_{ij} - u(C_i) - u(C_j)$ is minimal
  - Define new cluster k by $C_k = C_i \cup C_j$, and define d_{kl} for all l
  - Define a node k with children i and j. Remove clusters i and j
  - Assign length $\frac{1}{2} d_{ij} + \frac{1}{2} (u(C_i) - u(C_j))$ to the edge i -> k
  - Assign length $\frac{1}{2} d_{ij} + \frac{1}{2} (u(C_j) - u(C_i))$ to the edge j -> k
- Termination:
  - When only one cluster remains

Neighbor joining algorithm: example

	a	b	c	d
a	0	6	7	5
b		0	11	9
c			0	6
d				0

i	u(i)
a	(6+7+5)/2 = 9
b	(6+11+9)/2 = 13
c	(7+11+6)/2 = 12
d	(5+9+6)/2 = 10

i,j	d_{ij} – $u(C_i)$ – $u(C_j)$
a,b	6 - 9 - 13 = -16
a,c	7 - 9 - 12 = -14
a,d	5 - 9 - 10 = -14
b,c	11 - 13 - 12 = -14
b,d	9 - 13 - 10 = -14
c,d	6 - 12 - 10 = -16

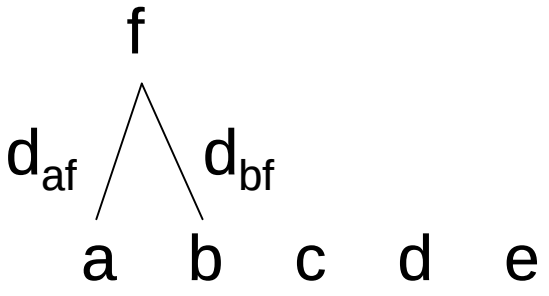


Neighbor joining algorithm: example

	a	b	c	d
a	0	6	7	5
b		0	11	9
c			0	6
d				0

i	u(i)
a	(6+7+5)/2 = 9
b	(6+11+9)/2 = 13
c	(7+11+6)/2 = 12
d	(5+9+6)/2 = 10

i,j	d_{ij} – $u(C_i)$ – $u(C_j)$
a,b	6 - 9 - 13 = -16
a,c	7 - 9 - 12 = -14
a,d	5 - 9 - 10 = -14
b,c	11 - 13 - 12 = -14
b,d	9 - 13 - 10 = -14
c,d	6 - 12 - 10 = -16



$d_{af} = ½\ 6 + ½\ (9 – 13) = 1$

$d_{bf} = ½\ 6 + ½\ (13 – 9) = 5$

Inferring the Past: Phylogenetic Trees (chapter 12)

- The biological problem
- Parsimony and distance methods
- *Models for mutations and estimation of distances*
- Maximum likelihood methods

Estimation of distances

- Many alternative ways to derive the distances d_{ij} exist
  - Simple solution: align each sequence pair and use the alignment score
  - This would not take into account that a change in base might revert back to the original base
  - We would then underestimate the distances
- Next: derivation of a simple stochastic model for the evolution of a DNA sequence
- Obtain the distances from the model

Estimation of distances

Key assumptions:

- mutations at sites are rare events in the course of time => poisson process
- sites evolve individually and by an identical mechanism
- number of mismatched bases is a sum of mutations at individual sites => binomial variable

A stochastic model for base substitutions

- Consider a single homologous site in two sequences
- Assume the sites diverged for time length t: the sites are separated by time 2t
- Suppose that the number of substitutions in any branch of length t has a Poisson distribution with mean λt
- Probability that k substitutions occur is given by the Poisson probability $e^{-\lambda t}(\lambda t)^k/(k!)$, k = 0, 1, 2, …

Substitutions at one site

- General model: P(substitution results in base j | site was base i) = m_{ij}
- Felsenstein model: $m_{ij} = \pi_j$, with $\pi_j \geq 0$ and $\pi_1 + \pi_2 + \pi_3 + \pi_4 = 1$
  - The previous base does not affect the outcome!
- Assume that the set of probabilities π_j is same at every position in the sequence

Substitutions at one site (2)

- Probability $q_{ij}(t)$ that a base i at time 0 is substituted by a base j a time t later
- $q_{ij}(t) = e^{-\lambda t} + (1 - e^{-\lambda t}) \pi_j$, if i = j
- $q_{ij}(t) = (1 - e^{-\lambda t}) \pi_j$, otherwise

Substitutions at one site (3)

- We assume stationarity: distribution of base frequencies is the same for every time t
- In other words, we want that

  P(base a time t later = j) = π_j^0

  where π_j^0 is the frequency of base j at time 0.

- For our simple model, this can be shown to hold

Estimating distances

- Distances should take into account the mutation mechanism
- Average of λt substitutions occur at a particular site on a branch of length t
- However, some of the substitutions do not change the base (A -> A or A -> G -> A, for example)

Mean number of substitutions in time t

- What is the chance H that a substitution actually changes a base?
- $H = \sum \pi_i(1 - \pi_i) = 1 - \sum \pi_i^2$
- Average number of real substitutions is then $\lambda t H$
- Distance K between two sequences is

$K = 2\lambda t H$

Estimating distances from sequence data

- We want to estimate $K = 2\lambda t H$ from sequence data
- The chance $F_{ij}(t)$ that we observe a base i in one sequence and a base j in another is

$$F_{ij}(t) = \sum_l \pi_l q_{li}(t) q_{lj}(t)$$

by averaging over the possible ancestral nucleotides

Estimating distances from sequence data

- Expression $F_{ij}(t) = \sum_l \pi_l q_{li}(t) q_{lj}(t)$ can be simplified by assuming that the mutation process is reversible:

  $\pi_i m_{ij} = \pi_j m_{ji}$ for all $i \neq j$

- From this it can be shown that

  $\pi_l q_{ij}(t) = \pi_j q_{ji}(t)$ for all i, j and $t > 0$

- Now the model simplifies into $F_{ij}(t) = \pi_i q_{ij}(2t)$

Estimating distances from sequence data

- What is the probability F = F(t) that the bases at a particular position in two immediate descendants of the same ancestor are identical?

$$F = \sum_i \pi_i q_{ii}(2t) = e^{-2\lambda t} + (1 - e^{-2\lambda t})(1 - H)$$

Putting the sites together

- Assume that
  - sites evolve independently of one other and
  - mutation process is identical at each site
  - The two sequences have been aligned against each other and gaps have been removed
- Do the bases at site i in the sequences differ?

$X_i = 1$ if the ith pair of sites differs

$X_i = 0$ otherwise

Putting the sites together (2)

- $P(X_i = 1) = 1 – F = (1 - e^{-2\lambda t})H$
- Now $D = X_1 + … + X_s$ is the number of mismatched pairs of bases
- D is a binomial random variable with parameters s and 1 – F
- Notice that D is the Hamming distance for the sequences

Putting the sites together (3)

- F is unknown and has to be estimated from the sequence data
- Recall that the observed proportion of successes is a good estimator of the binomial success probability: estimate 1 – F with D/s
- $D/s = (1 - e^{-2\lambda t})H$
- $2\lambda t = -\log(1 - D/(sH))$
- Finally, we obtain $K = 2\lambda tH = -H \log(1 - D/(sH))$

Jukes-Cantor formula

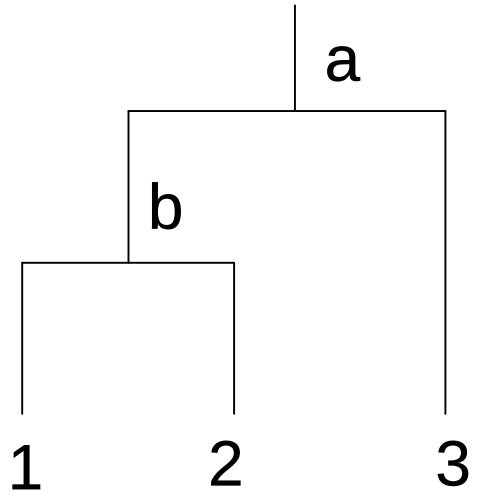
- Estimate $2\lambda tH = -H \log(1 - D/(sH))$ of the distance K is known as the Jukes-Cantor formula
- When H (chance that a substitution actually occurs) approaches 1, the estimate decreases and approaches the Poisson mean $2\lambda t$
- H is usually not known and has to be estimated from the data as well

Inferring the Past: Phylogenetic Trees (chapter 12)

- The biological problem
- Parsimony and distance methods
- Models for mutations and estimation of distances
- *Maximum likelihood methods*

Maximum likelihood methods

- Consider the tree on the right with three sequences
- Probability $p(i_1, i_2, i_3)$ of observing bases i_1, i_2 and i_3 can be computed by summing over all possible ancestral bases,



$$p(i1, i2, i3) = \sum_a \sum_b \pi_a q_{ai3}(t_2) q_{ab}(t_2 - t_1) q_{bi2}(t_1) q_{bi1}(t_1)$$

- Hard to compute for complex trees

Maximum likelihood estimation

- We would like to calculate likelihood $p(i_1, i_2, \ldots, i_n)$ in the general case
- Calculations can be arranged using the peeling algorithm (see exercises)
- Basic idea is to move all summation signs as far to the right as possible

Maximum likelihood estimation

- Likelihood for the data is then obtained by multiplying the likelihoods of individual sites

- General recipe for maximum likelihood estimation:
  - Maximize over all model parameters for a *given* tree
  - Maximize previous expression over *all* possible trees

Problems with tree-building

- Assumptions
  - Sites evolve independently of one other
  - Sites evolve according to the same stochastic model
  - The tree is rooted
  - The sequences are aligned
  - Vertical inheritance

Additional material on phylogenetic trees

- Durbin, Eddy, Krogh, Mitchison: Biological sequence analysis
- Jones, Pevzner: An introduction to bioinformatics algorithms
- Gusfield: Algorithms on strings, trees, and sequences