Reconstructing phylogenies

- What is a phylogeny? A diagram that illustrates the evolutionary relationship of a set of species, i.e. parent-child relationships across evolutionary time.
- Should reflect the transmission of genetic information.
- What did this diagram look like for the models we discussed so far?
- Is there a caveat?

Recall: what we need for phylogeny reconstruction

- Substitution models
- Pairwise distances
- Likelihood of a phylogenetic tree: Felsenstein's algorithm
  - Gives us a measure for comparing different trees
- Procedure for reconstructing phylogenetic trees:
  - Maximum likelihood
  - Neighbor-joining

Phylogeny of 3 sequences

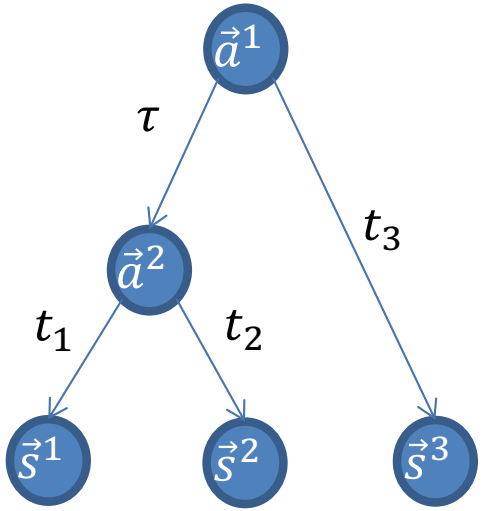
Imagine we have three sequences:

AGTTGCGGTGAGCGACTTTGCCAATGAATAGAATT $\vec{S}^1$

ACTCGCGGTGGCCGACTATGCCAGTGAACAGAATT $\vec{S}^2$

ACTCGGGGTGGCCGGCTATACCAGTGAACAGAACT $\vec{S}^3$

Here is one possible *phylogeny*:



Let's denote

ancestor sequences with a,
observable (leaf) sequences with s,
internal branch lengths with τ
length of branches leading to leaves with t.

Sequences $\vec{s}^1$ and $\vec{s}^2$ have a more recent common ancestor than $\vec{s}^1$ and $\vec{s}^3$ or $\vec{s}^2$ and $\vec{s}^3$

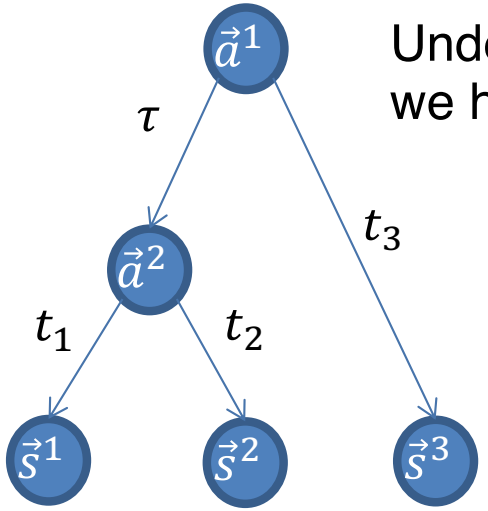
t_1 = Number of generations between $\vec{s}^1$ and ancestor $\vec{a}^2$

t_2 = Number of generations between $\vec{s}^2$ and ancestor $\vec{a}^2$

t_3 = Number of generations between $\vec{s}^3$ and ancestor $\vec{a}^1$

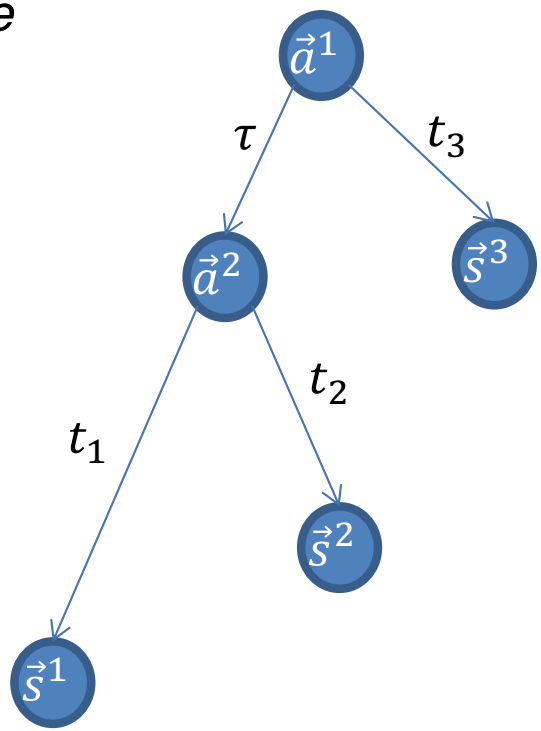
τ = Number of generations between ancestors $\vec{a}^1$ and $\vec{a}^2$

Phylogeny of 3 sequences



Under the molecular clock assumption we have $t_1 = t_2$, $t_1 + \tau = t_2 + \tau = t_3$

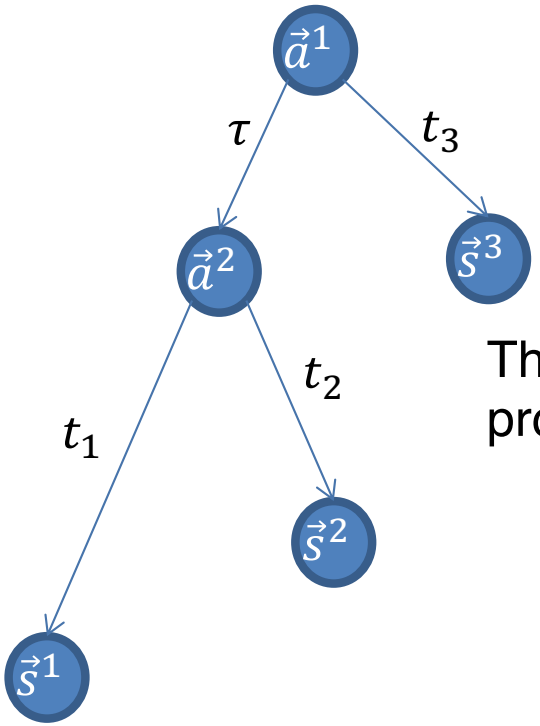
But:
- Different lineages may correspond to different populations with different *generation times*
- The mutation rate may be different in different lineages
- For non-neutral mutations the rate of substitution depends on *population size*



Therefore, unless we have good reasons to assume that substitution rates are the same in all branches, we cannot put constraints on the branch lengths.

Likelihood of a phylogeny

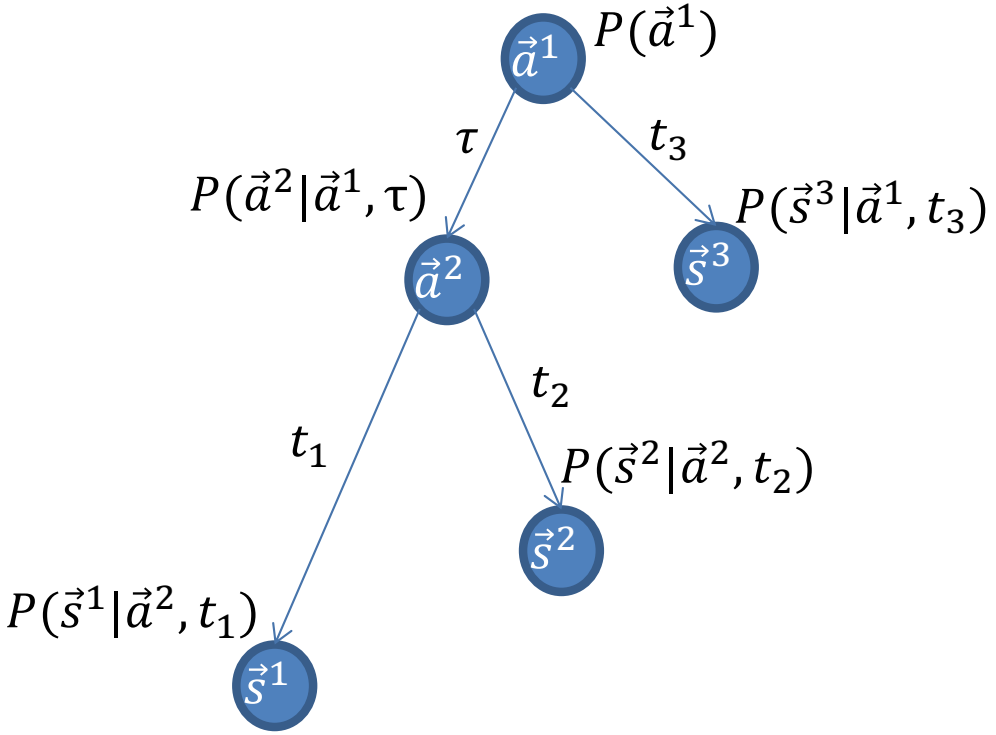
Let us calculate the likelihood of the sequence data under this model:



Topology of the tree

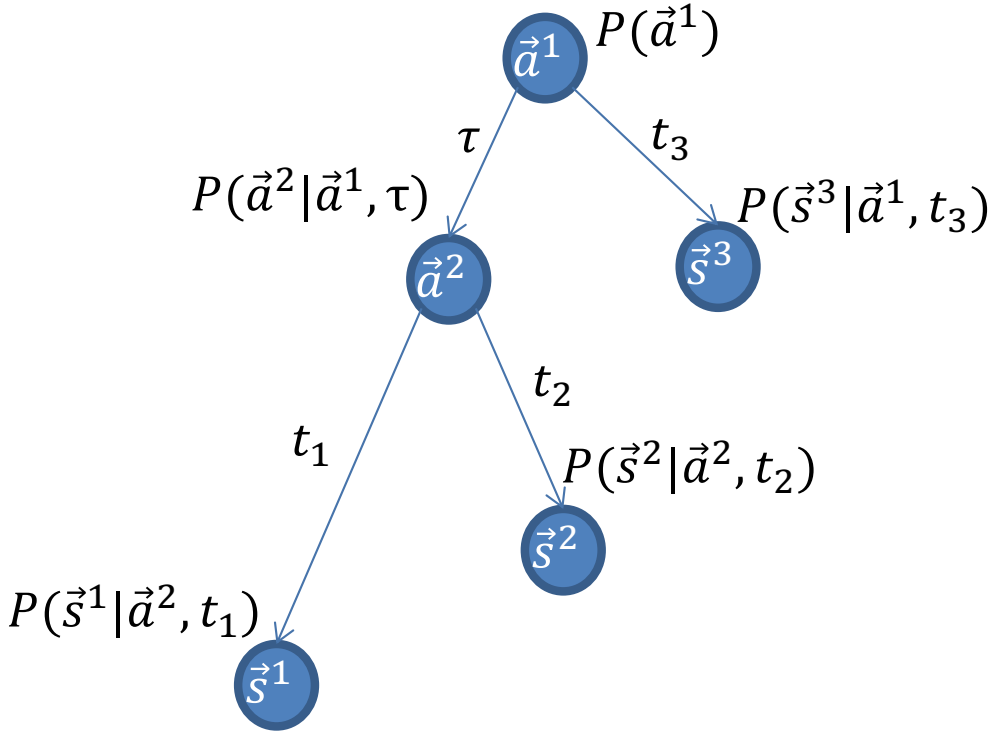
$$P(\vec{s}^1, \vec{s}^2, \vec{s}^3, \vec{a}^1, \vec{a}^2 | t_1, t_2, t_3, \tau, \mathrm{T}) = P(\vec{a}^1)P(\vec{a}^2|\vec{a}^1, \tau)P(\vec{s}^3|\vec{a}^1, t_3)P(\vec{s}^1|\vec{a}^2, t_1)P(\vec{s}^2|\vec{a}^2, t_2)$$

That is simply the probability of the root sequence times the product of transition probabilities at each of the branches.



Likelihood of a phylogeny

$$P(\vec{s}^1, \vec{s}^2, \vec{s}^3, \vec{a}^1, \vec{a}^2 | t_1, t_2, t_3, \tau, \mathrm{T}) =$$
$$P(\vec{a}^1) P(\vec{a}^2 | \vec{a}^1, \tau) P(\vec{s}^3 | \vec{a}^1, t_3) P(\vec{s}^1 | \vec{a}^2, t_1) P(\vec{s}^2 | \vec{a}^2, t_2)$$



Assuming independence of the substitutions at different positions i, we have

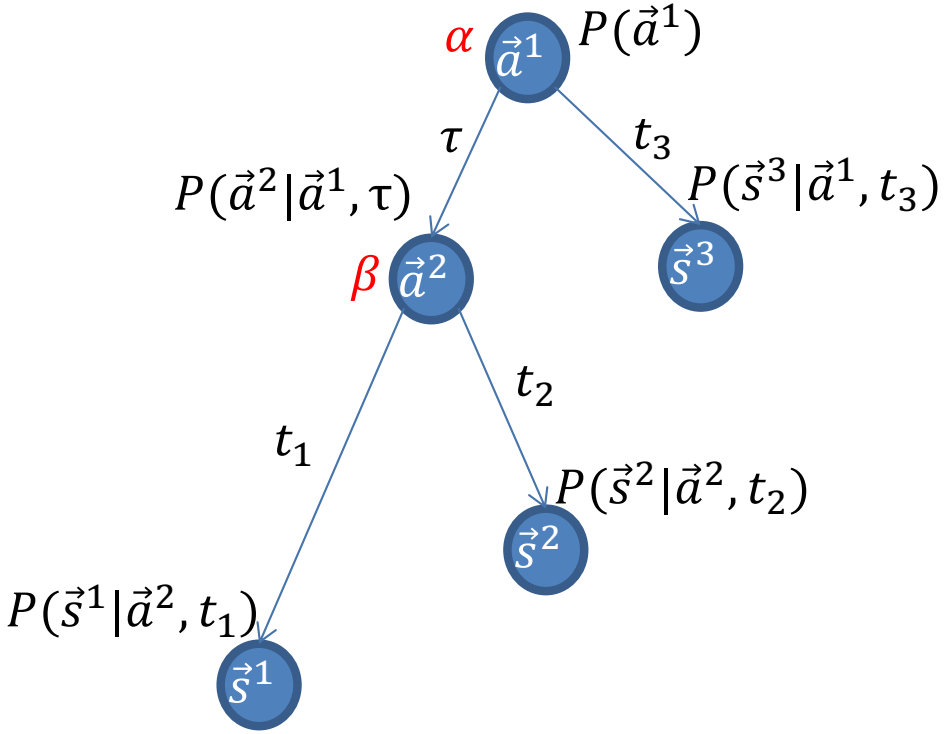
$$P(\vec{s}^1, \vec{s}^2, \vec{s}^3, \vec{a}^1, \vec{a}^2 | t_1, t_2, t_3, \tau, \mathrm{T}) =$$
$$\prod_i P(\overrightarrow{a_i}^1) P(\overrightarrow{a_i}^2 | \overrightarrow{a_i}^1, \tau) P(\overrightarrow{s_i}^3 | \overrightarrow{a_i}^1, t_3) P(\overrightarrow{s_i}^1 | \overrightarrow{a_i}^2, t_1) P(\overrightarrow{s_i}^2 | \overrightarrow{a_i}^2, t_2)$$

So we can focus on the likelihood of a single position…

Likelihood of a phylogeny at a single position

For simplicity, we drop the index of the position in the sequence, and write the likelihood at a given position as

$$P(\vec{s}^1, \vec{s}^2, \vec{s}^3, \vec{a}^1, \vec{a}^2 | t_1, t_2, t_3, \tau, \mathrm{T}) = P(\vec{a}^1) P(\vec{a}^2 | \vec{a}^1, \tau) P(\vec{s}^3 | \vec{a}^1, t_3) P(\vec{s}^1 | \vec{a}^2, t_1) P(\vec{s}^2 | \vec{a}^2, t_2)$$

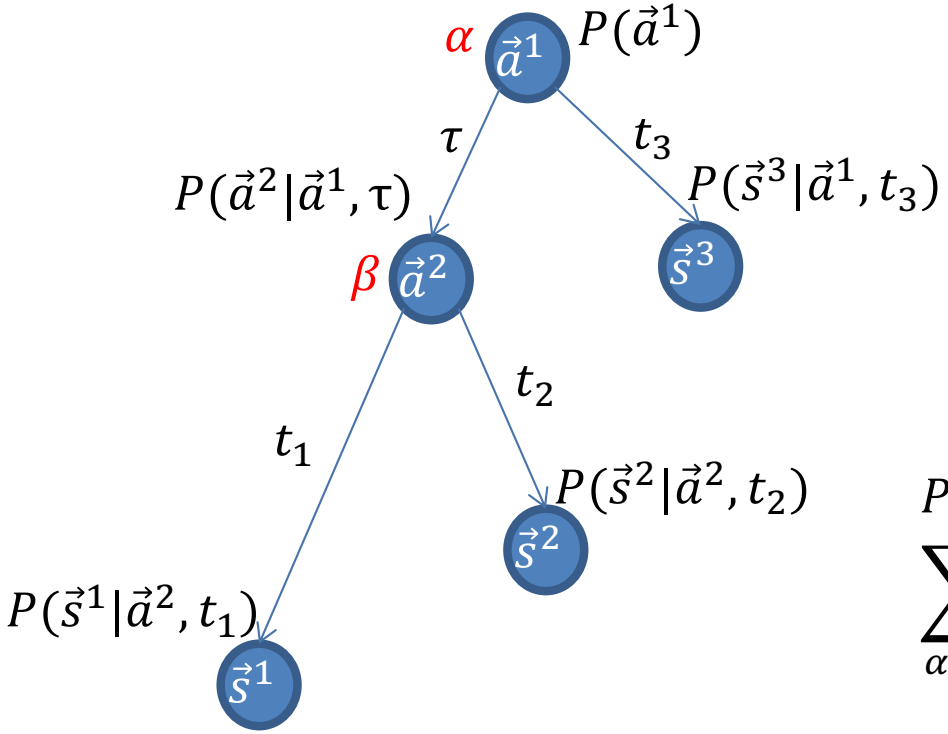


α, β are the nucleotides at the corresponding position in the internal nodes (which we don't know).

Likelihood of a phylogeny at a single position

For simplicity, we drop the index of the position in the sequence, and write the likelihood at a given position as

$$P(\vec{s}^1, \vec{s}^2, \vec{s}^3, \vec{a}^1, \vec{a}^2 | t_1, t_2, t_3, \tau, \mathrm{T}) =$$
$$P(\vec{a}^1) P(\vec{a}^2 | \vec{a}^1, \tau) P(\vec{s}^3 | \vec{a}^1, t_3) P(\vec{s}^1 | \vec{a}^2, t_1) P(\vec{s}^2 | \vec{a}^2, t_2)$$



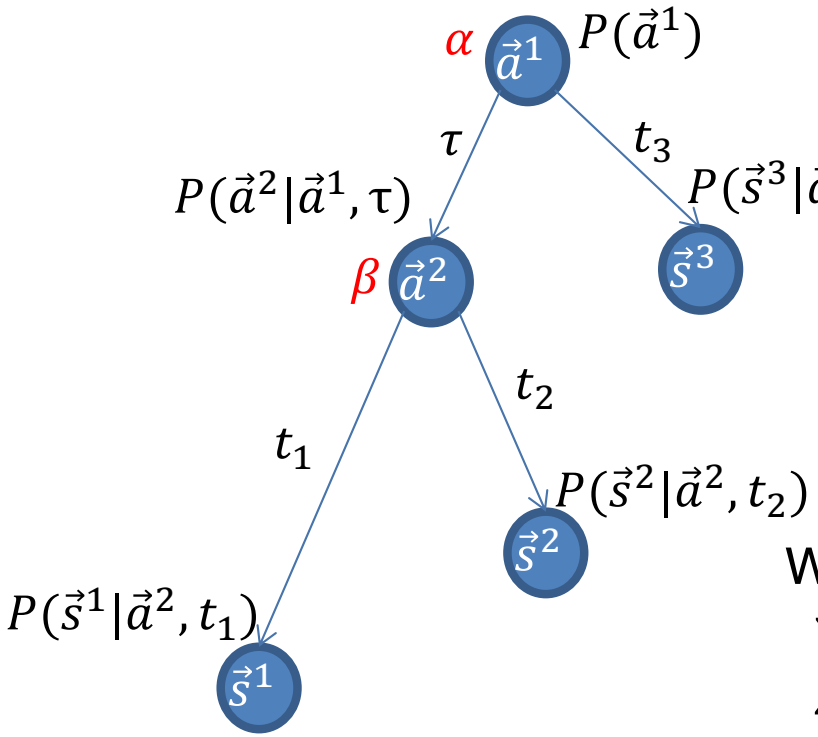
Not knowing what the ancestral states were, we sum over all possibilities

$$P(s^1, s^2, s^3 | t_1, t_2, t_3, \tau, \mathrm{T}) =$$
$$\sum_{\alpha,\beta} q_\alpha \, P(\beta|\alpha,\tau) P(s^3|\alpha,t_3) P(s^1|\beta,t_1) P(s^2|\beta,t_2)$$

Likelihood of a phylogeny at a single position

For simplicity, we drop the index of the position in the sequence, and write the likelihood at a given position as

$$P(\vec{s}^1, \vec{s}^2, \vec{s}^3, \vec{a}^1, \vec{a}^2 | t_1, t_2, t_3, \tau, \mathrm{T}) =$$
$$P(\vec{a}^1) P(\vec{a}^2 | \vec{a}^1, \tau) P(\vec{s}^3 | \vec{a}^1, t_3) P(\vec{s}^1 | \vec{a}^2, t_1) P(\vec{s}^2 | \vec{a}^2, t_2)$$



$$P(s^1, s^2, s^3 | t_1, t_2, t_3, \tau, \mathrm{T}) =$$
$$\sum_{\alpha,\beta} q_\alpha P(\beta|\alpha,\tau) P(s^3|\alpha,t_3) P(s^1|\beta,t_1) P(s^2|\beta,t_2)$$

Due to reversibility $q_\alpha P(\beta|\alpha,\tau) = q_\beta P(\alpha|\beta,\tau)$ and additivity $q_\alpha P(\beta|\alpha,\tau) P(s^3|\alpha,t_3) = q_\beta P(s^3|\beta,t_3+\tau)$

We can rewrite $P(s^1, s^2, s^3 | t_1, t_2, t_3, \tau, \mathrm{T}) =$

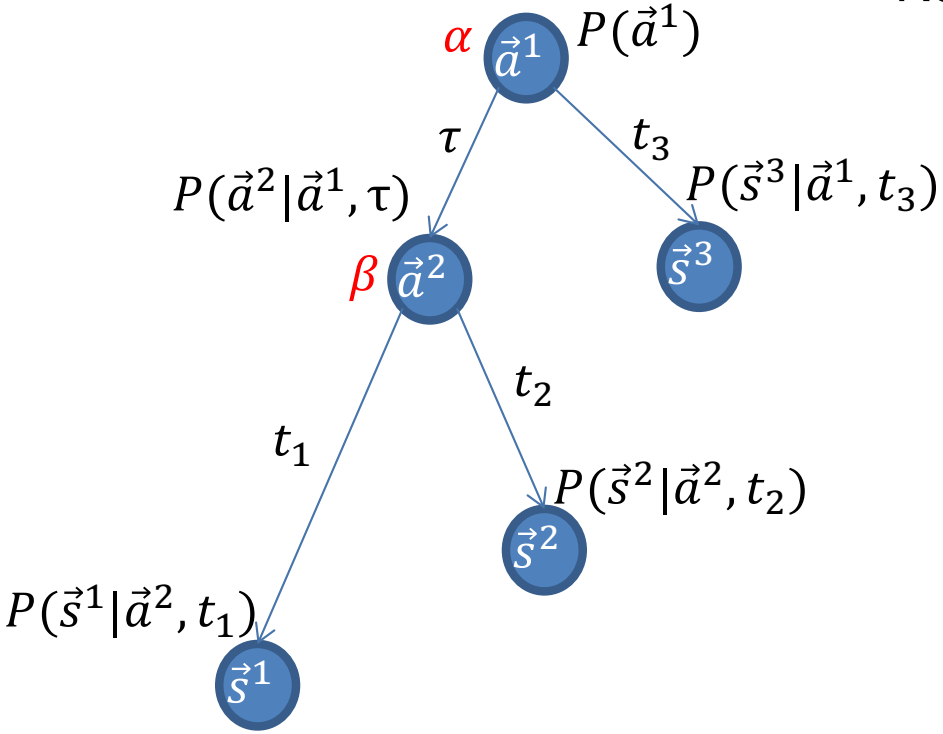
$$\sum_\beta \sum_\alpha q_\alpha P(\beta|\alpha,\tau) P(s^3|\alpha,t_3) P(s^1|\beta,t_1) P(s^2|\beta,t_2) =$$

$$\sum_\beta \sum_\alpha q_\beta P(\alpha|\beta,\tau) P(s^3|\alpha,t_3) P(s^1|\beta,t_1) P(s^2|\beta,t_2) =$$

$$\sum_\beta q_\beta P(s^3|\beta,t_3+\tau) P(s^1|\beta,t_1) P(s^2|\beta,t_2)$$

Likelihood of a phylogeny at a single position

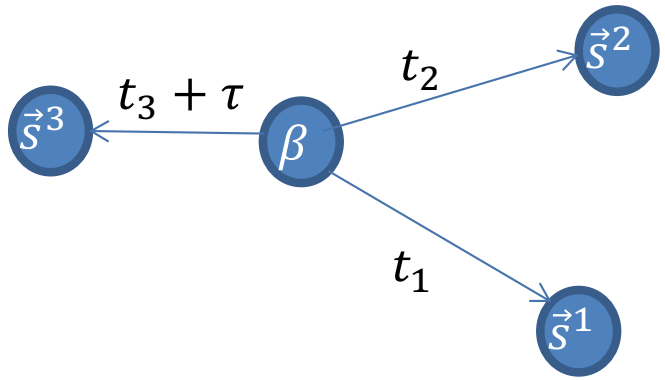
$$P(s^1, s^2, s^3|t_1, t_2, t_3, \tau, \mathrm{T}) = \sum_\beta q_\beta P(s^3|\beta, t_3+\tau) P(s^1|\beta, t_1) P(s^2|\beta, t_2)$$



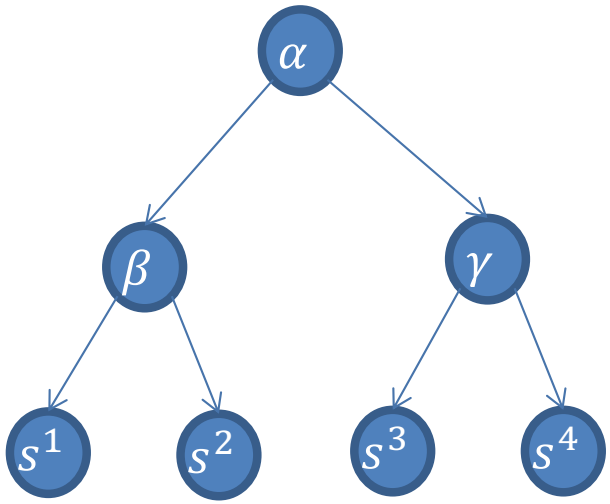
Redefining $t_3 + \tau \rightarrow t_3$ we can write generally

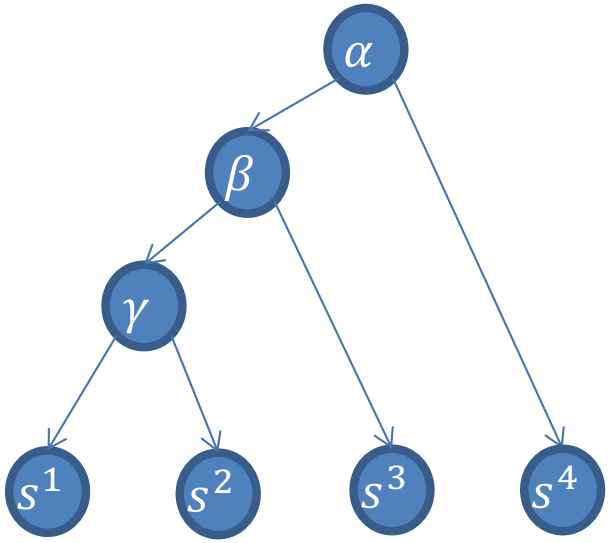
$$P(s^1, s^2, s^3|t_1, t_2, t_3, \tau, \mathrm{T}) = \sum_\beta q_\beta P(s^1|\beta, t_1) P(s^2|\beta, t_2) P(s^3|\beta, t_3)$$

Which means that there is only one possible tree topology for 3 sequences:



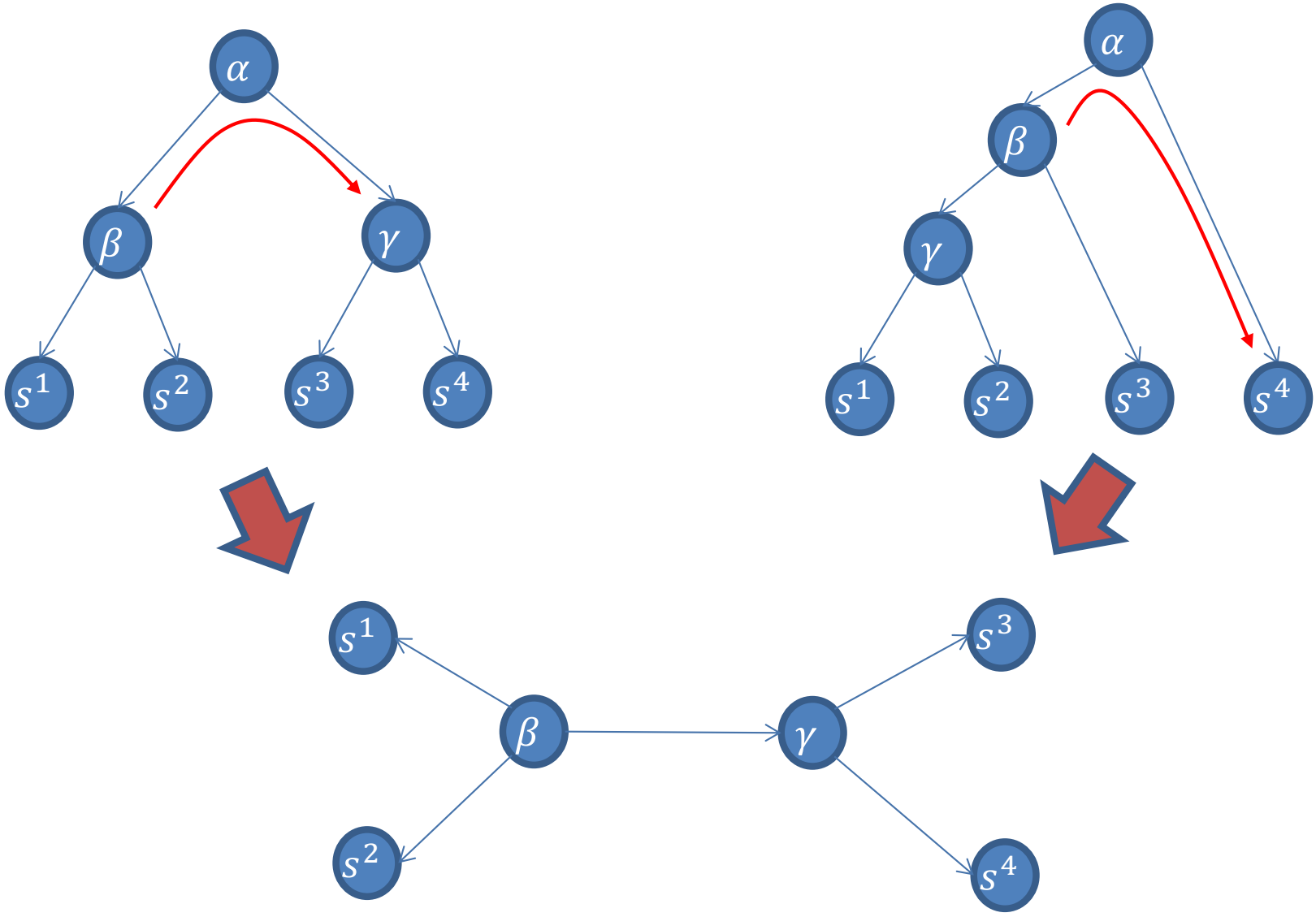
Phylogeny of 4 sequences





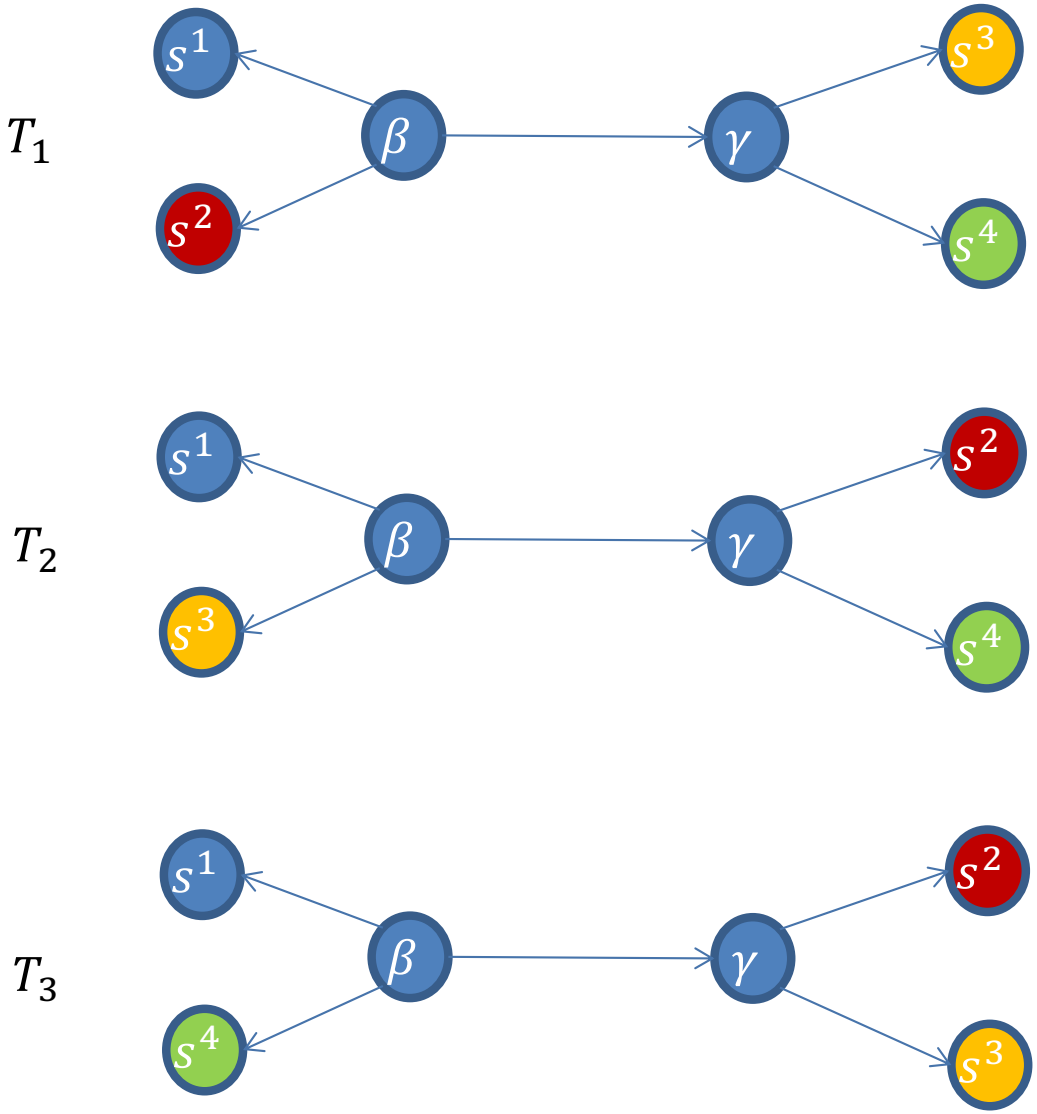
Again we focus on one position at a time

Due to reversibility and summing over one internal letter which we do not know, we can see that both of the trees below are equivalent in likelihood.



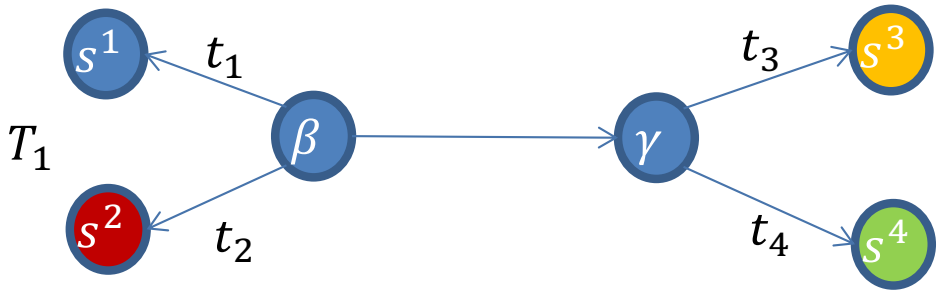
Phylogeny of 4 letters

There are only 3 truly different topologies for 4 letters:



These topologies correspond to all possibilities of choosing the closest neighbor for s^1.

The likelihood of an evolutionary hypothesis



The set of hypotheses for this data set consists of

1. The tree topology T
2. The length of the branches
   t_1, t_2, t_3, t_4, τ
3. The letters at the internal nodes
   α, β

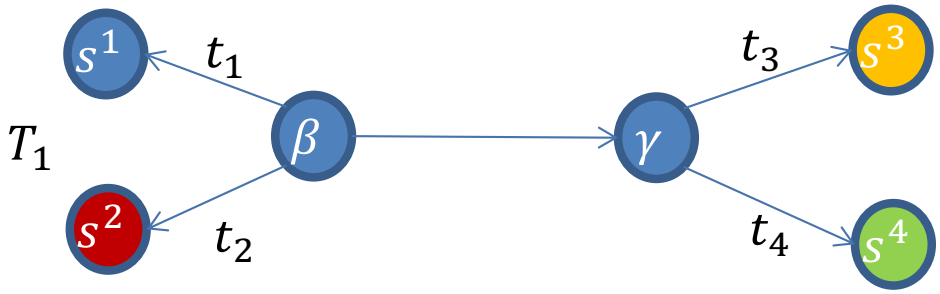
From the rules of probability, we know that what we would have to do for these unknown parameters would be to integrate over 5 branch length, sum over 16 possible letter combinations at internal nodes and sum over possible topologies.

This is not mathematically tractable. What can we do then?

Let's assume first a given topology. Ideally, what we would calculate is

$$P(s^1, s^2, s^3, s^4|\mathrm{T}) = \sum_{\beta,\gamma} \int dt_1\, dt_2 dt_3 dt_4 d\tau\, P(s^1, s^2, s^3, s^4, t_1, t_2, t_3, t_4, \tau|T)$$

The likelihood of an evolutionary hypothesis



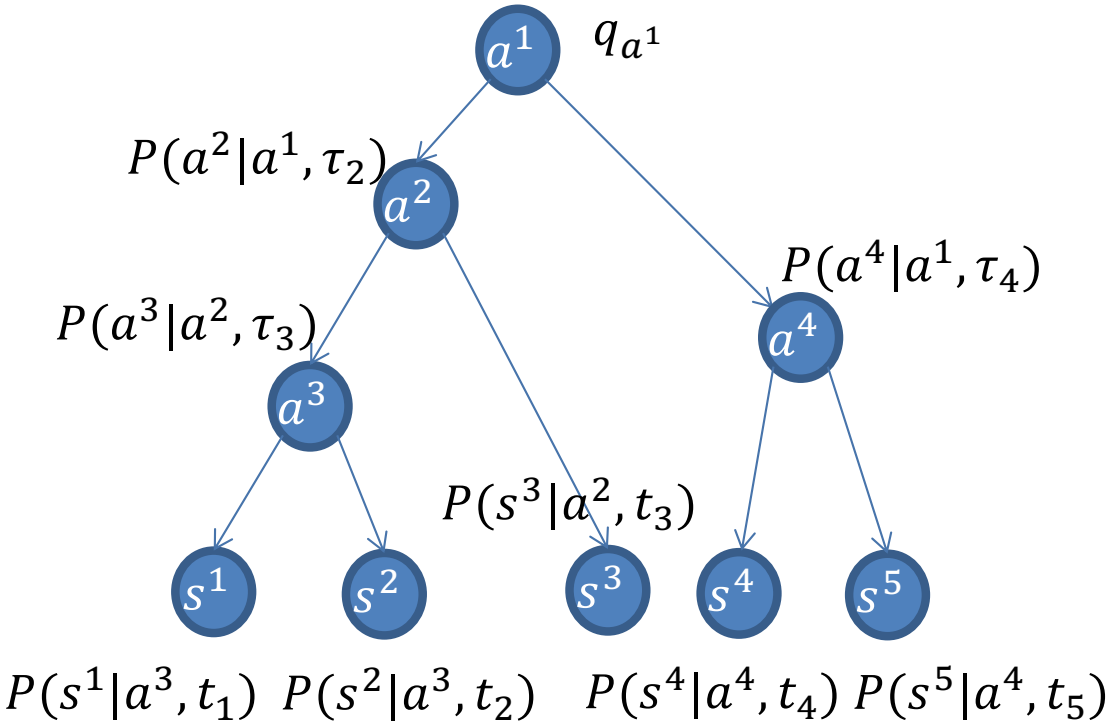
What is tractable is to:

1. Sum over the letters at the internal nodes
2. Find the maximum likelihood branch lengths.

In 1981 Felsenstein presented an algorithm for calculating the likelihood of the letters at the leaves, given the topology and branch lengths, efficiently summing over all the letters at internal nodes. Even for large trees.

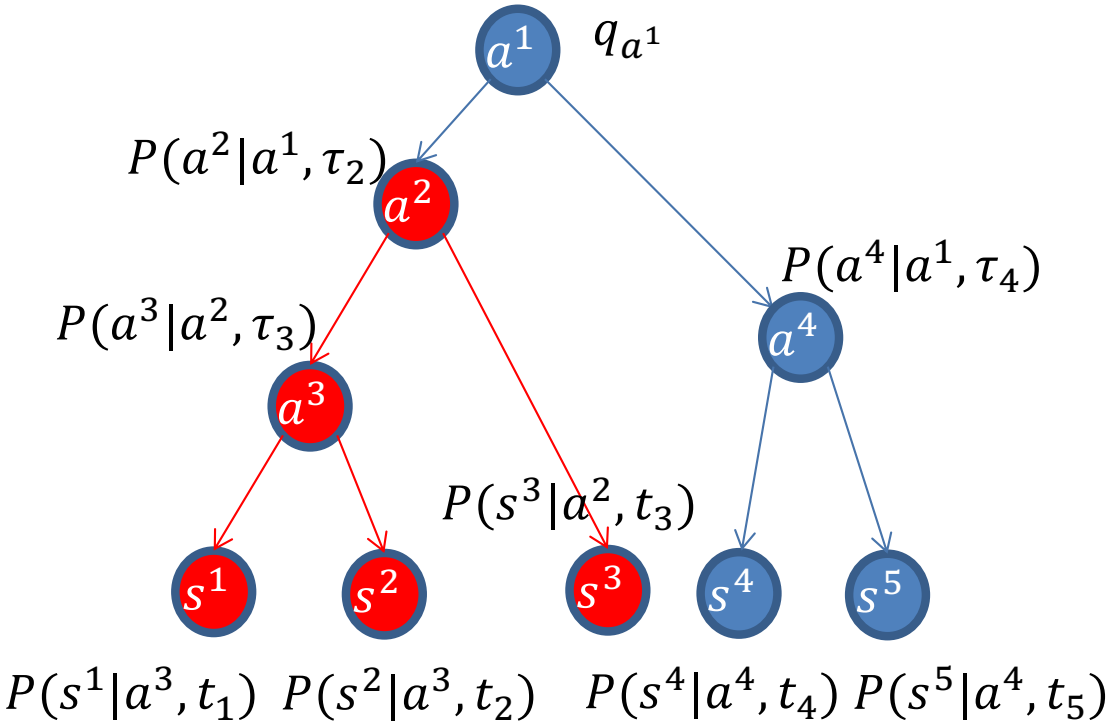
Let's see how it works…

Felsenstein's recursion relations



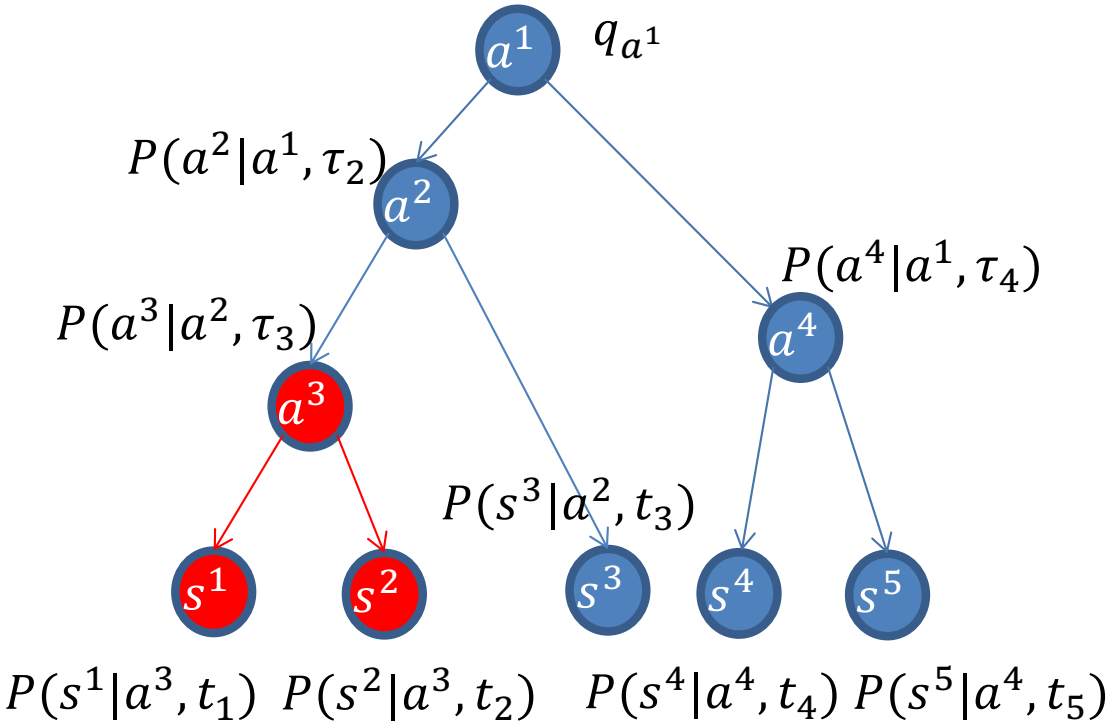
Let $P(D_j|\alpha)$ denote the probability of all the data below node j in the tree, assuming that the letter at node j is α.

Felsenstein's recursion relations



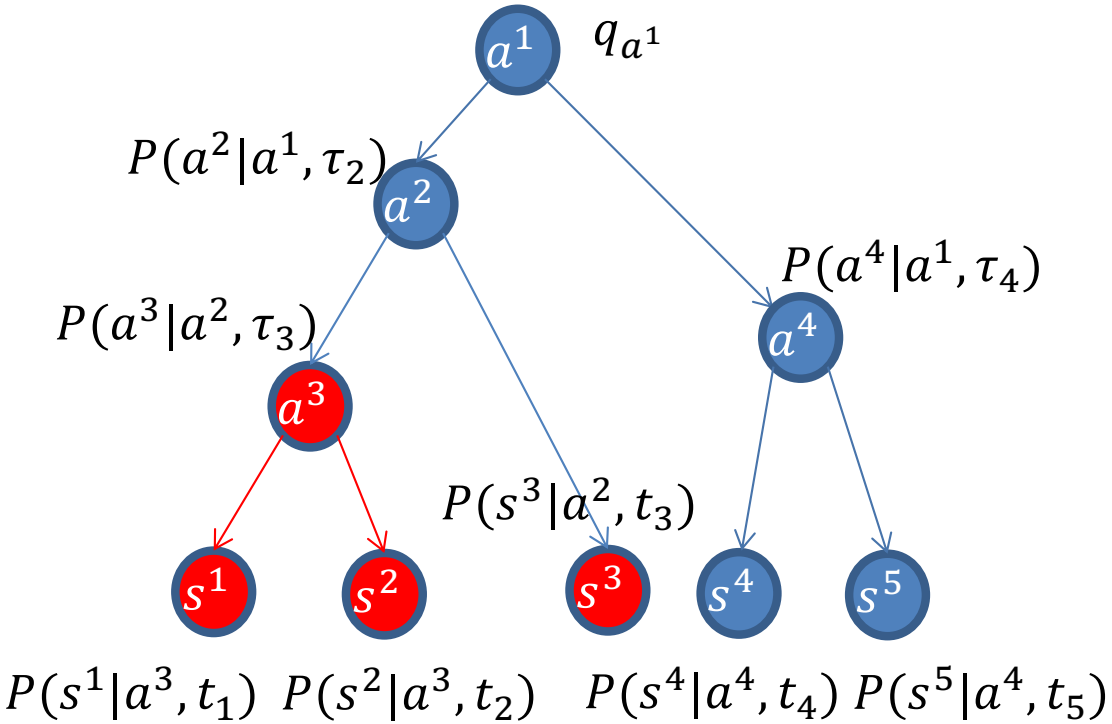
Let $P(D_j|\alpha)$ denote the probability of all the data below node j in the tree, assuming that the letter at node j is α. Example: $P(D_2|a^2)$.

Felsenstein's recursion relations



Let $P(D_j|\alpha)$ denote the probability of all the data below node j in the tree, assuming that the letter at node j is α. Example: $P(D_2|a^2)$. Or $P(D_3|a^3)$.

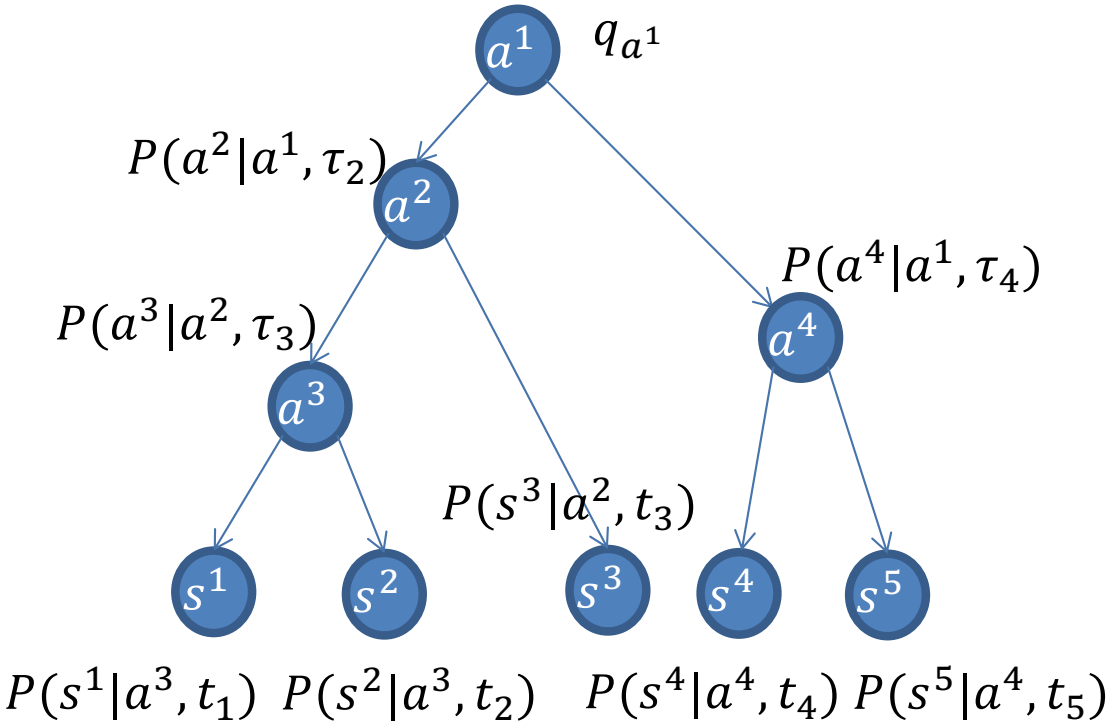
Felsenstein's recursion relations



Let $P(D_j|\alpha)$ denote the probability of all the data below node j in the tree, assuming that the letter at node j is α. Example: $P(D_2|a^2)$. Or $P(D_3|a^3)$.

Note that $P(D_2|a^2) = \left[\sum_{a^3} P(a^3|\, a^2,\tau_3)P(D_3|a^3)\right][P(s^3|a^2,t_3)]$

Felsenstein's recursion relations



For the root node, we assume that nucleotide frequencies correspond to those of the limit distribution, which is realized once the evolutionary process has run for a long time.

In general, if C(j) is the set of children of node j and the letter at node j is α, we have

$$P(D_j|\alpha) = \prod_{i\in C(j)}\left[\sum_{a^i} P(a^i|\alpha,t_i)P(D_i|a^i)\right]$$

If the child i is a leaf in the tree, its contribution in the above expression is simply

$$P(s^i|\alpha,t_i)P(D_i|a^i) = P(s^i|\alpha,t_i)$$

Felsenstein's algorithm

Initialization:

Number the nodes in the tree 'top to bottom'

Set the current node k to $2n-1$ (n - nr of leaves in the tree)

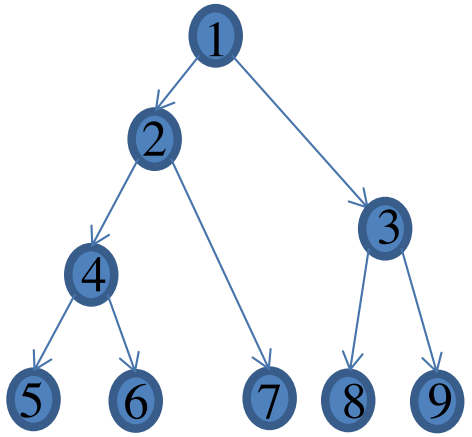
Recursion:

If k is a leaf, set $P(D_k|\alpha) = \delta_{\alpha s^k}, \delta_{\alpha\beta} = \begin{cases} 0, if\ \alpha \neq \beta \\ 1, if\ \alpha = \beta \end{cases}$

If k is a not leaf, set for each α,

$$P(D_k|\alpha) = \prod_{j \in C(k)} \left[\sum_{a^j} P(a^j|\alpha, t_j) P(D_j|a^j) \right]$$

Reduce k by 1 and if $k = 0$, go to termination



Termination:

Set final probability $P(D) = \sum_\alpha q_\alpha P(D_1|\alpha)$

The total number of calculation steps is roughly 10 per node.
Even for many leaves the likelihood can thus be calculated in a relatively short time.

Felsenstein's algorithm

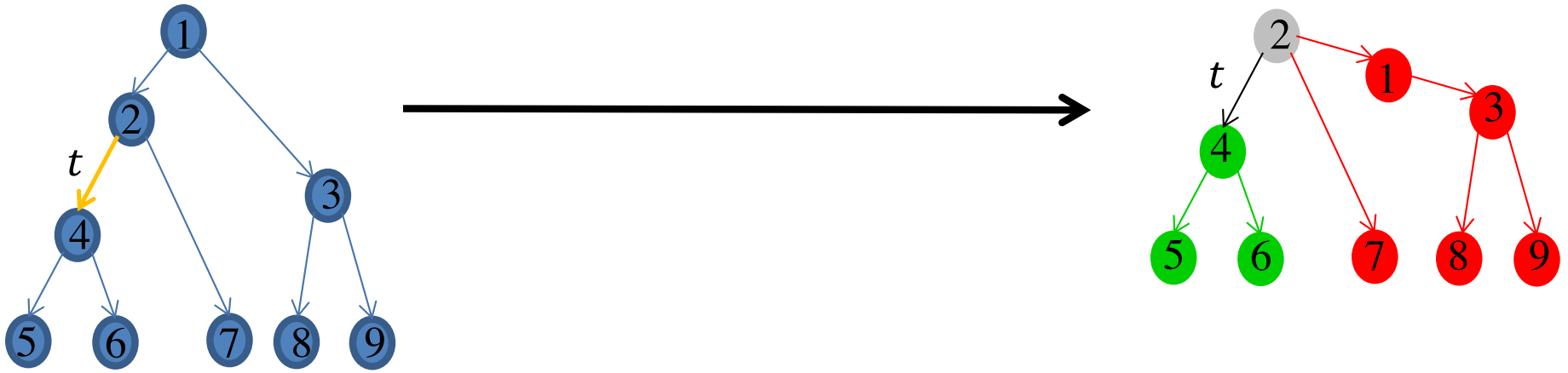
An efficient algorithm to calculate the likelihood of the sequence data, assuming a certain tree topology and the length of the branches.

However, we typically do not know the branch lengths, and we would have to integrate them out, which is not practical.

We may still get quite accurate results if we use instead the maximum likelihood values for the branch lengths.

Optimizing the branch lengths

- Imagine that we want to optimize the length of the branch shown in gold.
- Because of reversibility, the likelihood is independent of where we place the root of the tree. The tree is thus topologically equivalent to:



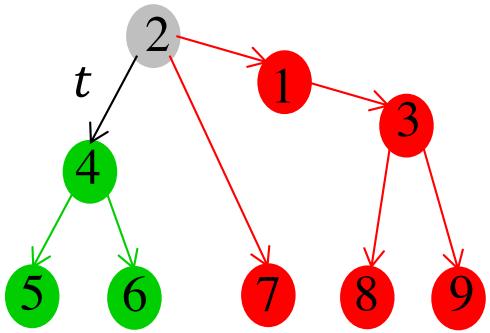
The likelihood of this tree can be written as

$$P(D) = \sum_{a^2,a^4} P(a^4|a^2,t)P(D_4|a^4)P(D_r|a^2)q_{a^2}$$

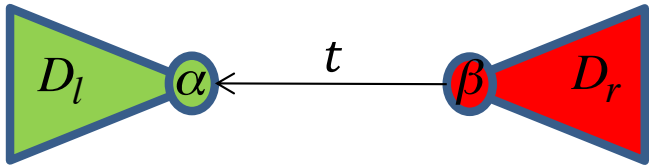
Where $P(D_r|a^2)$ is the likelihood of all the red nodes given that node 2 is a^2

$$P(D_r|a^2) = P(s^7|a^2,t_7)\sum_{a^1} P(a^1|a^2,t_1)P(D_1|a^1)$$

Optimizing the branch lengths



can also be viewed as



and then

$$P(D) = \sum_{a^2,a^4} P(a^4|a^2,t)P(D_4|a^4)P(D_r|a^2)q_{a^2}$$

$$= \sum_{a^2,a^4} P(a^4|a^2,t)P(D_4|a^4)q_{a^2}P(s^7|a^2,t_7)\sum_{a^1} P(a^1|a^2,t_1)P(D_1|a^1)$$

conceptually has the form $P(D) = \sum_{\alpha,\beta} P(D_l|\alpha)P(\alpha|\beta,t)q_\beta P(D_r|\beta)$

The calculation was just for one position in the sequence. But since all positions have the same branch length, the full likelihood will have the form:

$$P(D) = \prod_{i=1}^{L}\left[\sum_{\alpha_i,\beta_i} P\left(D_{l,i}\middle|\alpha_i\right)P(\alpha_i|\beta_i,t)q_{\beta_i}P(D_{r,i}|\beta_i)\right]$$

Optimizing the branch lengths

$$P(D) = \prod_{i=1}^{L} \left[\sum_{\alpha_i, \beta_i} P\left(D_{l,i} \middle| \alpha_i\right) P(\alpha_i | \beta_i, t) q_{\beta_i} P(D_{r,i} | \beta_i) \right]$$

For the Jukes-Cantor model we have $P(\alpha|\beta, t) = \frac{1}{4} + \left(\delta_{\alpha\beta} - \frac{1}{4}\right) e^{-\frac{4\mu t}{3}}$ and we called the probability that the letter remains unchanged $c = \frac{1+3e^{-\frac{4\mu t}{3}}}{4}$

Substituting we find

$$P(D) = \prod_{i=1}^{L} \left[\sum_{\alpha_i} P\left(D_{l,i} \middle| \alpha_i\right) P\left(D_{r,i} \middle| \alpha_i\right) c q_{\alpha_i} + \sum_{\alpha_i \neq \beta_i} P\left(D_{l,i} \middle| \alpha_i\right) P\left(D_{r,i} \middle| \beta_i\right) \left(\frac{1-c}{3}\right) q_{\beta_i} \right]$$

Letting $A_i = \sum_{\alpha_i} P\left(D_{l,i} \middle| \alpha_i\right) P\left(D_{r,i} \middle| \alpha_i\right) q_{\alpha_i}$ and $B_i = \frac{1}{3} \sum_{\alpha_i \neq \beta_i} P\left(D_{l,i} \middle| \alpha_i\right) P\left(D_{r,i} \middle| \beta_i\right) q_{\beta_i}$

We obtain $P(D) = \prod_{i=1}^{L} [A_i c + B_i (1-c)]$

The optimum occurs when $\frac{\partial P(D)}{\partial c} = 0$ or $\frac{\partial \log(P(D))}{\partial c} = 0$, so we have to find the c's satisfying the above condition.

Optimizing the branch lengths

$$P(D) = \prod_{i=1}^{L} [A_i c + B_i(1-c)]$$

The optimum occurs when $\frac{\partial P(D)}{\partial c} = 0$ or $\frac{\partial \log(P(D))}{\partial c} = 0$, which means

$$\sum_{i=1}^{L} \frac{\partial}{\partial c} log[A_i c + B_i(1-c)] = \sum_{i=1}^{L} \frac{A_i - B_i}{A_i c + B_i(1-c)} = 0$$

with $A_i = \sum_{\alpha_i} P(D_{l,i}|\alpha_i) P(D_{r,i}|\alpha_i) q_{\alpha_i}$ $B_i = \frac{1}{3} \sum_{\alpha_i \neq \beta_i} P(D_{l,i}|\alpha_i) P(D_{r,i}|\beta_i) q_{\beta_i}$

We can use, for e.g., a search method to get the c.

Often we can use another method, Expectation-Maximization to solutions that are difficult to get analytically.

Expectation maximization

- Method for finding maximum likelihood estimates of parameters when the model depends on hidden variables.

- Consists of two steps
  - E (expectation) – during which a function for the expectation of the log-likelihood is created based on the data.
  - M (maximization) – during which the parameters that maximize the above distribution are identified. These parameters are used to determine the distribution for the latent variables in the next step.

Optimizing branch lengths by Expectation Maximization

EM algorithm:

1. Start with an initial set of branch lengths (μt with corresponding c)
2. Calculate $P\left(D_n^i \middle| \alpha_i\right)$ for all nodes n and positions i

3. For each branch, calculate A_i and B_i depending on the current branch lengths

4. Update the c's trying to maximize $P(D) = \prod_{i=1}^{L} [A_i c + B_i(1-c)]$

5. Determine the total amount $D = \sum \left|\frac{c'-c}{c'+c}\right|$ by which the branch lengths have changed.

6. If D is below a cutoff, stop. Else, go back to step 2.

It can be shown that the procedure converges to an optimum.

May not be the global optimum if there are multiple (local) optima.

Deriving the update equations

$$P(D) = \prod_{i=1}^{L} [A_i c + B_i(1-c)] \qquad A_i = \sum_{\alpha_i} P(D_{l,i}|\alpha_i) P(D_{r,i}|\alpha_i) q_{\alpha_i}$$

$$B_i = \frac{1}{3} \sum_{\alpha_i \neq \beta_i} P(D_{l,i}|\alpha_i) P(D_{r,i}|\beta_i) q_{\beta_i}$$

The optimum occurs when $\frac{\partial P(D)}{\partial c} = 0$ or $\frac{\partial \log(P(D))}{\partial c} = 0$, which means

$$\sum_{i=1}^{L} \frac{\partial}{\partial c} log[A_i c + B_i(1-c)] = \sum_{i=1}^{L} \frac{A_i - B_i}{A_i c + B_i(1-c)} = 0$$

Deriving the update equations

$$\sum_{i=1}^{L} \frac{A_i - B_i}{A_i c + B_i(1-c)} = 0 \Rightarrow \sum_{i=1}^{L} \frac{A_i c - B_i c}{A_i c + B_i(1-c)} = \sum_{i=1}^{L} \frac{A_i c - B_i c + B_i - B_i}{A_i c + B_i(1-c)} = 0$$

$$\sum_{i=1}^{L} \frac{A_i c - B_i c + B_i - B_i}{A_i c + B_i(1-c)} = \sum_{i=1}^{L} \left[1 - \frac{B_i}{A_i c + B_i(1-c)}\right] = 0 \Rightarrow L = \sum_{i=1}^{L} \frac{B_i}{A_i c + B_i(1-c)}$$

But $$\sum_{i=1}^{L} \frac{A_i - B_i}{A_i c + B_i(1-c)} = 0 \Rightarrow \sum_{i=1}^{L} \frac{A_i}{A_i c + B_i(1-c)} = \sum_{i=1}^{L} \frac{B_i}{A_i c + B_i(1-c)} = L$$

Finally $$\sum_{i=1}^{L} \frac{A_i c}{A_i c + B_i(1-c)} = Lc$$

Which gives us an expression similar to a posterior probability for *c*

That we use as the update rule for *c*

$$c^{new} = \frac{1}{L} \sum_{i=1}^{L} \frac{A_i c^{old}}{A_i c^{old} + B_i(1 - c^{old})}$$

Optimizing branch lengths by Expectation Maximization

EM algorithm:

1. Start with an initial set of branch lengths

2. Calculate $P\left(D_n^i \middle| \alpha_i\right)$ for all nodes n and positions i

3. For each branch, calculate A_i and B_i and set new branch lengths

$$c' = \frac{1}{L}\sum_{i=1}^{L}\frac{A_i c}{A_i c + B_i(1-c)}$$

4. Determine the total amount $D = \sum \left|\frac{c'-c}{c'+c}\right|$ by which the branch lengths have changed.

5. If D is below a cutoff, stop. Else, go back to step 2.

Summary so far

We have algorithms to

- Efficiently calculate the likelihood summing over all internal node sequences
- Find the set of branch lengths that maximizes this likelihood

What we do not have yet is the topology. We still need to:

- Find a way to search among tree topologies
  - Find a good starting topology
  - Generate and evaluate perturbations (e.g. changing the position of branches, swap entire subtrees)

Minimum evolution assumption – Neighbor joining tree

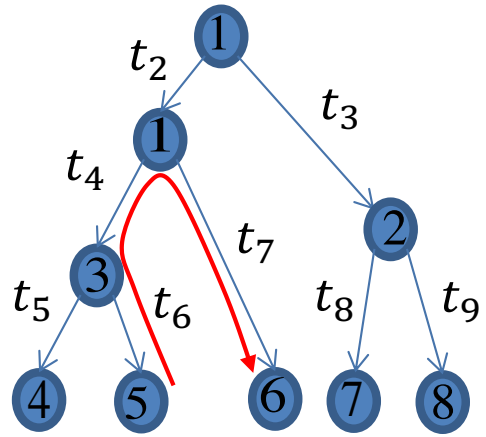
We will look at the procedure for constructing the tree for which the total evolutionary change along all branches is minimal.

In detail: Estimating branch lengths from pairwise distances

We have seen already that we can estimate the evolutionary distance between two sequences from the number of differences in their alignment.

For the Jukes-Cantor model we had

$$\mu t = -\frac{3}{4} log\left[1 - \frac{4d}{3L}\right]$$



Taking the example of leaves 6 and 7,

$$t_{(6,7)} = -\frac{3}{4\mu} log\left[1 - \frac{4d_{(6,7)}}{3L}\right]$$

Generally, for a pair of leaves (i, j)

$$t_{(i,j)} = -\frac{3}{4\mu} log\left[1 - \frac{4d_{(i,j)}}{3L}\right]$$

Assuming additivity of branch lengths, the time computed based on the number of substitutions should match the branch length.

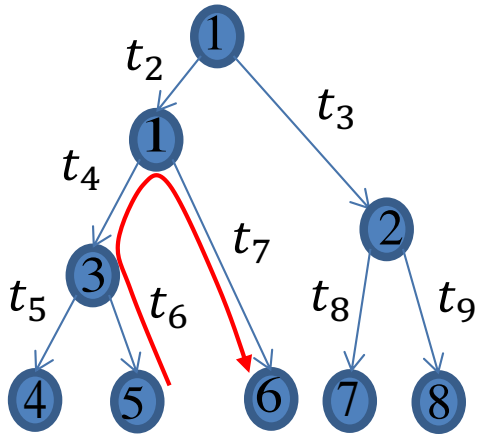
In the example above, $t_{(6,7)} = t_6 + t_4 + t_7$

In detail: Estimating branch lengths from pairwise distances

In general, the evolutionary time computed from pairwise differences should obey

$$t_{(i,j)} = \sum_{b} t_b$$

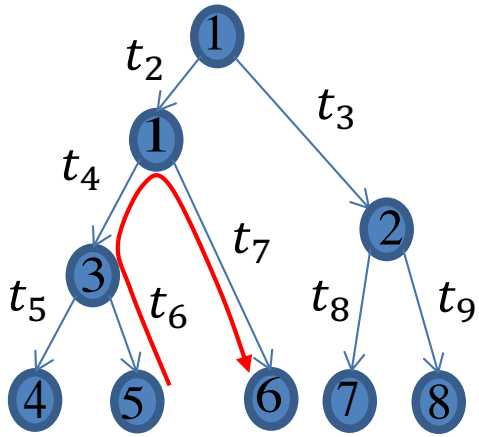
Where t_b is the length of branch b on the path between i and j.



In detail: Estimating branch lengths from pairwise distances

If we define the branch matrix B with

$$B_{(i,j)b} = \begin{cases} 1 \text{ when branch } b \text{ lies on the path connecting } (i,j) \\ 0 \text{ otherwise} \end{cases}$$



We can get a good guess for branch lengths by minimizing

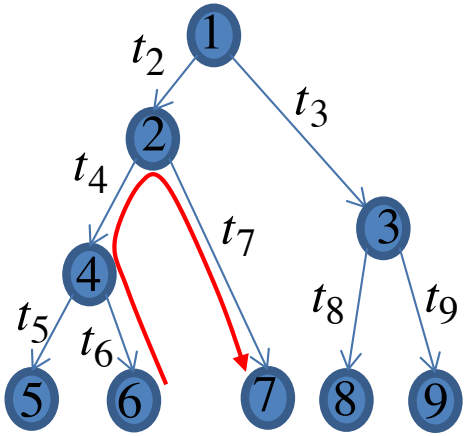
$$\Delta^2 = \sum_{i \neq j} \left(t_{(i,j)} - \sum_{b} B_{(i,j)b} t_b \right)^2$$

Which says that we want to minimize the error in predicting the observed pairwise differences between the sequences.

For n leaves we have $\frac{n(n-1)}{2}$ distances and $2n-1$ branch lengths

$$t_{(5,6)} = t_5 + t_6$$
$$t_{(6,7)} = t_6 + t_4 + t_7$$
$$t_{(5,7)} = t_5 + t_4 + t_7$$
$$\vdots$$

In detail: Estimating branch lengths from pairwise distances



If we define the branch matrix B with

$$B_{(i,j)b} = \begin{cases} 1 \text{ when branch } b \text{ lies on the path connecting } (i,j) \\ 0 \text{ otherwise} \end{cases}$$

To minimize $\Delta^2 = \sum_{i \neq j} \left(t_{(i,j)} - \sum_b B_{(i,j)b} t_b \right)^2$

We have to solve, $\forall k$

$$\frac{\partial \Delta^2}{\partial t_k} = -2 \sum_{i \neq j} \left(t_{(i,j)} - \sum_b B_{(i,j)b} t_b \right) B_{(i,j)k} t_k = 0$$

These equations should give us the maximum likelihood values for the branch lengths $t^*{}_k$ given the topology and assuming that branch lengths are additive.

Filling in these values into $\Delta^2 = \sum_{i \neq j} \left(t_{(i,j)} - \sum_b B_{(i,j)b} t^*{}_b \right)^2$ we also get a measure of consistency between branch lengths and evolutionary distances inferred from differences between sequences.

Which we can use to find an initial tree topology.

Neighbor joining

We set to find a reasonable initial topology, one that minimizes

$$\Delta^2 = \sum_{i \neq j} \left(t_{(i,j)} - \sum_b B_{(i,j)b} t_b \right)^2$$

In 1987 Saitou and Nei proposed ‘neighbor joining’, a method that produces a unique tree, which has been shown to generally correspond to a ‘minimum-evolution’ tree.

We will now define the procedure for constructing this tree.

Neighbor joining algorithm

Initialization

Construct a “star” topology tree with all nodes
Compute pairwise distances between sequences
Use the method that we discussed to infer a set of branch lengths
Calculate for each pair of nodes a new measure T_{ij}

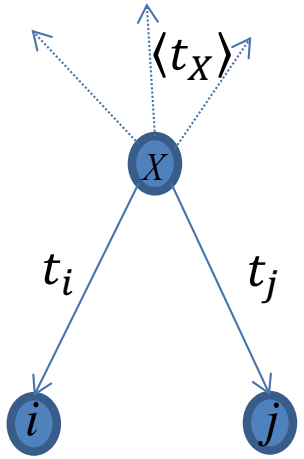
Iteration

Pick the pair (i, j) for which T_{ij} is minimal
Create a new node k as most recent common ancestor of i and j
and use it in place of i and j
Calculate distances from k to other nodes m
Recalculate all T_{ij}'s

We will now define T_{ij}'s and show that the tree constructed with the procedure outline above is the minimum evolution tree.

Neighbor joining

To all other nodes



Define for each leaf node i

$$r_i = \frac{1}{n-2} \sum_{j \neq i} t_{(i,j)}$$

Which has the form of average distance from i to any other node in the tree. n is the number of leaves, and $t_{(i,j)}$ is the distance between leaves i and j, which, under additivity assumptions should be given by the sum of branch lengths on the path linking i and j.

Define $T_{ij} = t_{(i,j)} - r_i - r_j$

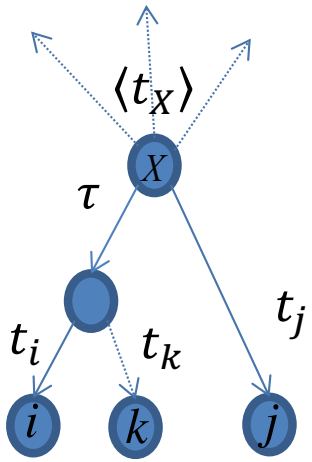
Theorem: the pair (i, j) for which T_{ij} is smallest are neighbors in the tree.

We will prove this by contradiction. That is, we will show that if i and j are not neighbors, there must by another node k such that $T_{ik} < T_{ij}$.

Neighbor joining

Assume that (i,j) are not neighbors, but that there is a node k between them.

To all other nodes



$$T_{ij} - T_{ik} = t_{(i,j)} - r_i - r_j - t_{(i,k)} + r_i + r_k = t_{(i,j)} - t_{(i,k)} + r_k - r_j$$

$$r_j = \frac{1}{n-2}\Big(t_i + \tau + t_j + t_k + \tau + t_j + (n-3)\big(t_j + \langle t_X \rangle\big)\Big)$$

$$r_k = \frac{1}{n-2}\Big(t_i + t_k + t_j + \tau + t_k + (n-3)(t_k + \tau + \langle t_X \rangle)\Big)$$

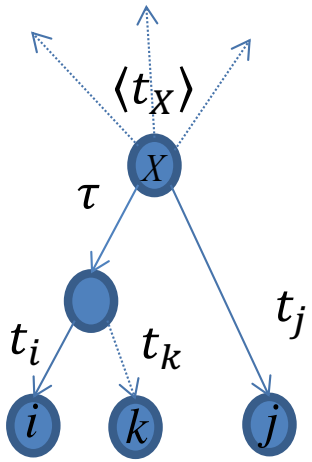
Which gives $$r_k - r_j = \frac{1}{n-2}\Big(t_k - t_j - \tau + (n-3)\big(t_k + \tau - t_j\big)\Big) = t_k - t_j + \frac{n-4}{n-2}\tau$$

Then $$T_{ij} - T_{ik} = t_{(i,j)} - t_{(i,k)} + t_k - t_j + \frac{n-4}{n-2}\tau$$

Neighbor joining

$$T_{ij} - T_{ik} = t_{(i,j)} - t_{(i,k)} + t_k - t_j + \frac{n-4}{n-2}\tau$$

To all other nodes



As $t_{(i,j)} = t_i + \tau + t_j$ and $t_{(i,k)} = t_i + t_k$

We get

$$T_{ij} - T_{ik} = t_i + \tau + t_j - t_i - t_k + t_k - t_j + \frac{n-4}{n-2}\tau = \left(1 + \frac{n-4}{n-2}\right)\tau > 0$$

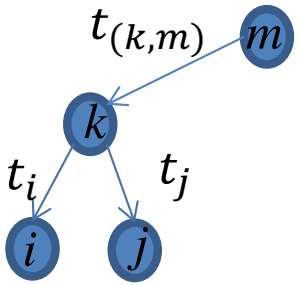
Which implies $T_{ij} - T_{ik} > 0$ and thus T_{ij} is not the smallest.

Which is what we wanted to show.

Neighbor-joining algorithm

Initialization

Construct a “star” topology tree with all nodes
Compute pairwise distances between sequences
Calculate for each pair of nodes $T_{ij} = t_{(i,j)} - r_i - r_j$



Iteration

Pick the pair (i, j) for which T_{ij} is minimal
Define a new node k with distances to other nodes m

$$t_{(k,m)} = \frac{1}{2}\left(t_{(i,m)} + t_{(j,m)} - t_{(i,j)}\right)$$

$$t_{(i,m)} = t_{(i,k)} + t_{(k,m)}$$
$$t_{(j,m)} = t_{(j,k)} + t_{(k,m)}$$

Set the distances to node k, $t_i = \frac{1}{2}\left(t_{(i,j)} + r_i - r_j\right)$ and $t_j = \frac{1}{2}\left(t_{(i,j)} + r_j - r_i\right)$

$$t_{(i,m)} = t_{(i,k)} + t_{(k,m)}$$
$$t_{(i,j)} = t_{(i,k)} + t_{(j,k)}$$

$$t_{(i,k)} = \frac{1}{2}\left(t_{(i,j)} + t_{(i,m)} - t_{(j,m)}\right)$$

averaging over m we get the r’s

Recalculate all $T_{ij} = t_{(i,j)} - r_i - r_j$

Summary

Given a set of sequences, and assuming a specific substitution model

- We can compute pairwise distances
- We can infer an initial topology and branch lengths
- We have a scoring system to infer which topology and branch lengths leads to the highest likelihood of the sequence data

Phylogeny reconstruction

Topology: start with a reasonable guess (e.g. neighbor joining)
Monte Carlo sampling

Branch lengths: Expectation-Maximization

Likelihood of the sequence data: Felsenstein's algorithm

Lots of programs out there

http://evolution.genetics.washington.edu/phylip/software.html