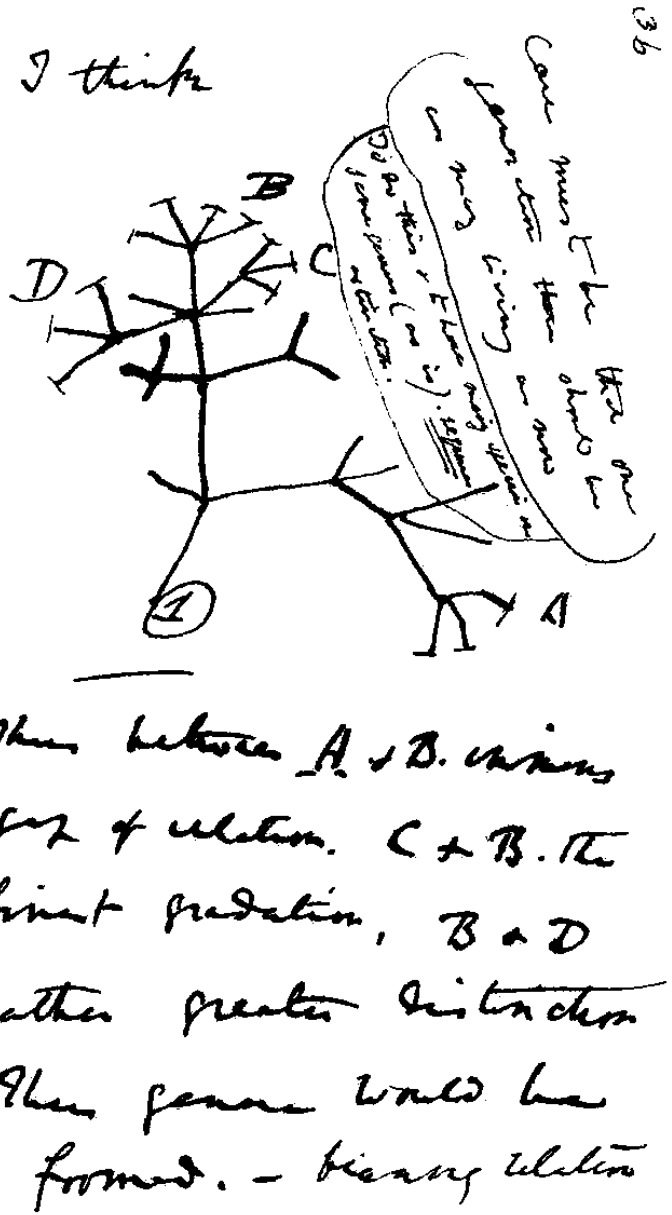
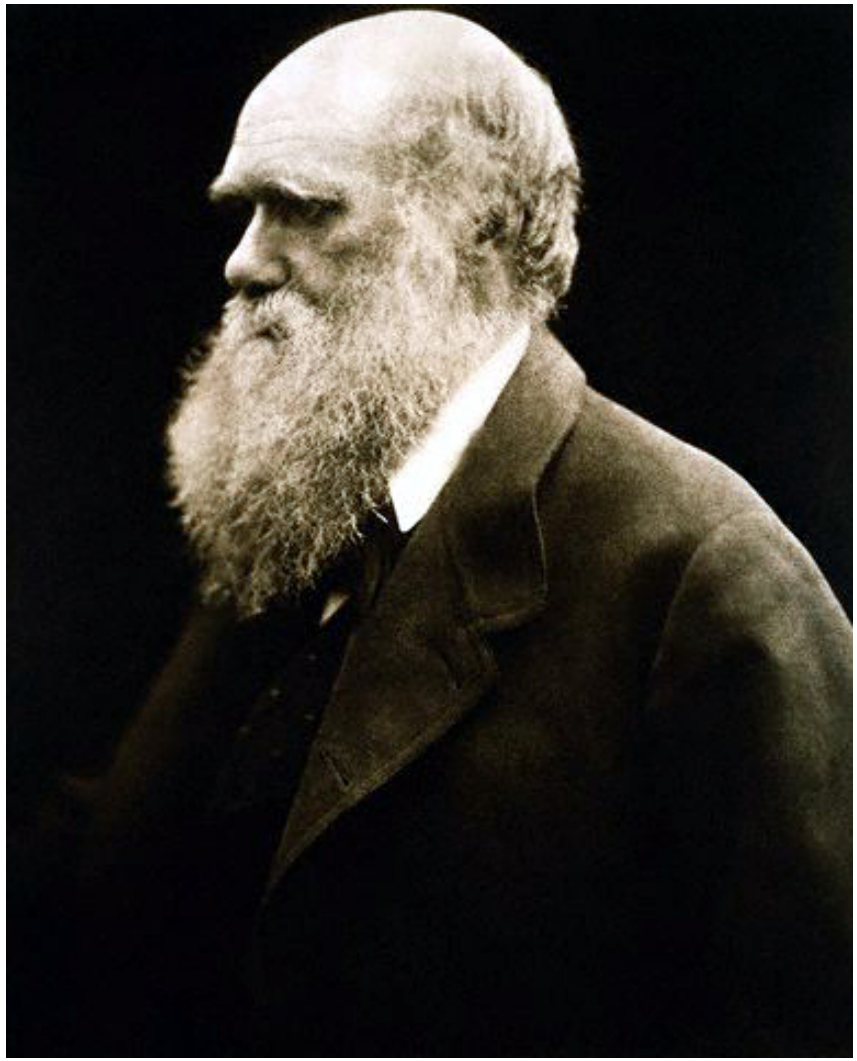
A Short Introduction To Likelihood in Phylogenetics

Center for Integrative Bioinformatics Vienna (CIBIV)
Max F. Perutz Laboratories (MFPL)
Vienna, Austria
http://www.cibiv.at

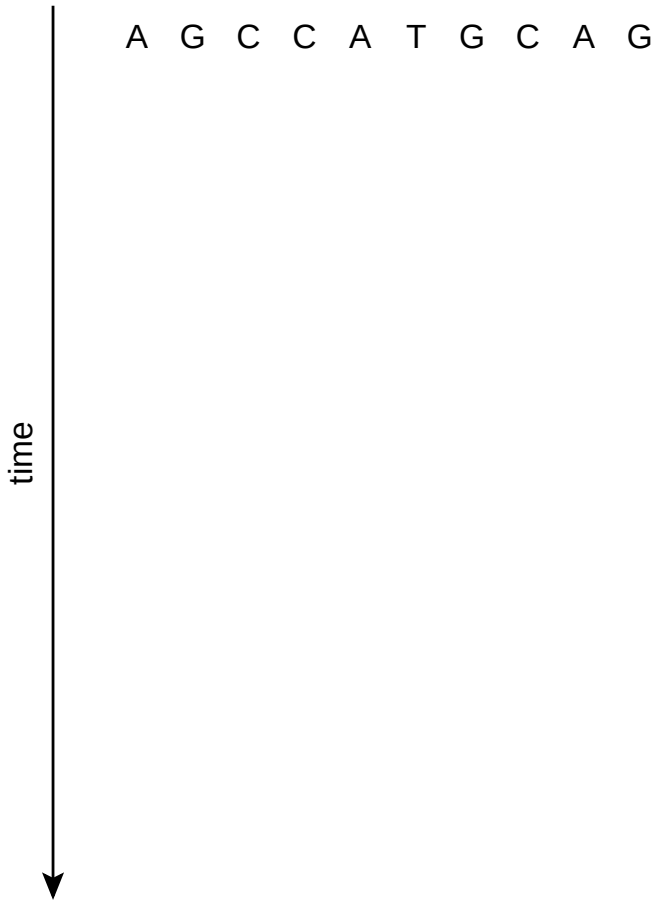
Charles Darwin: Evolutionary Relationships



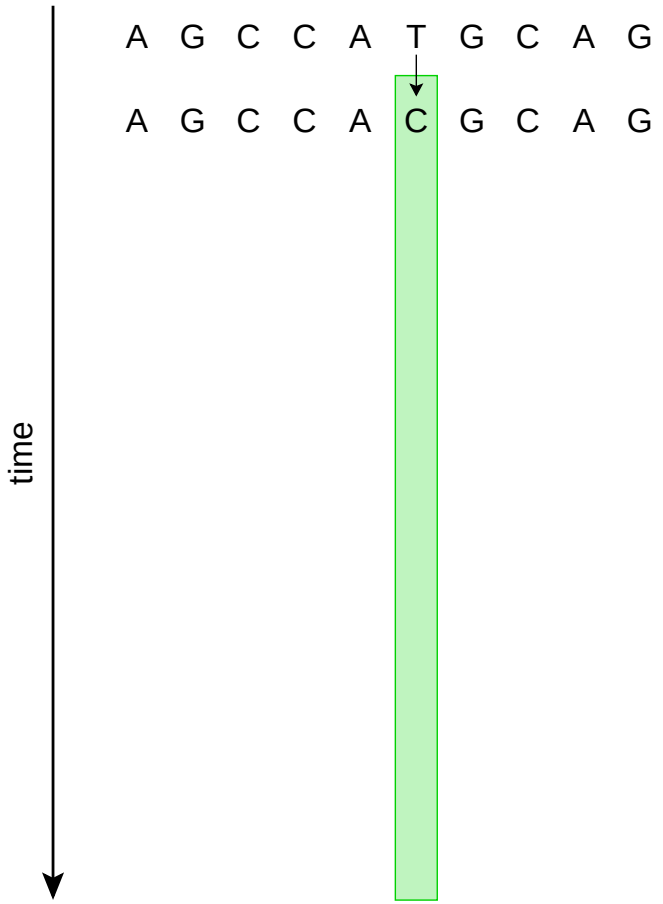


Charles Darwin (1809-1882)

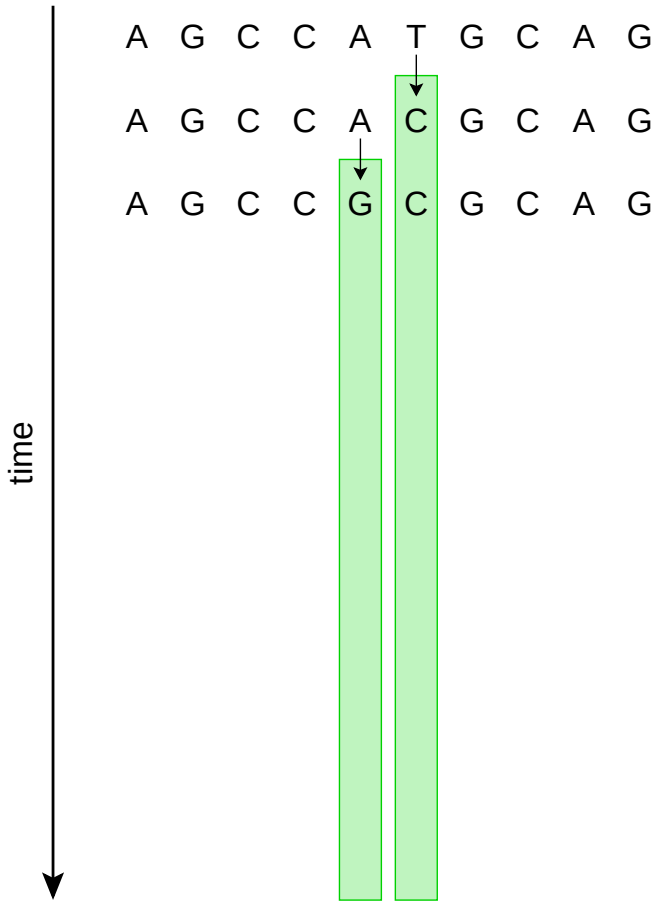
Traces of Sequence Evolution



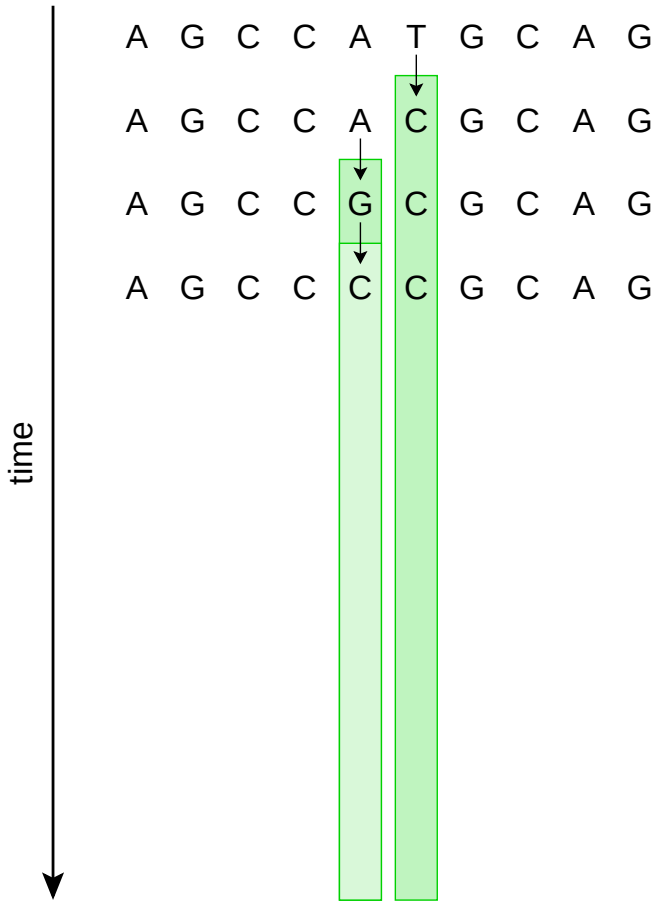
Traces of Sequence Evolution



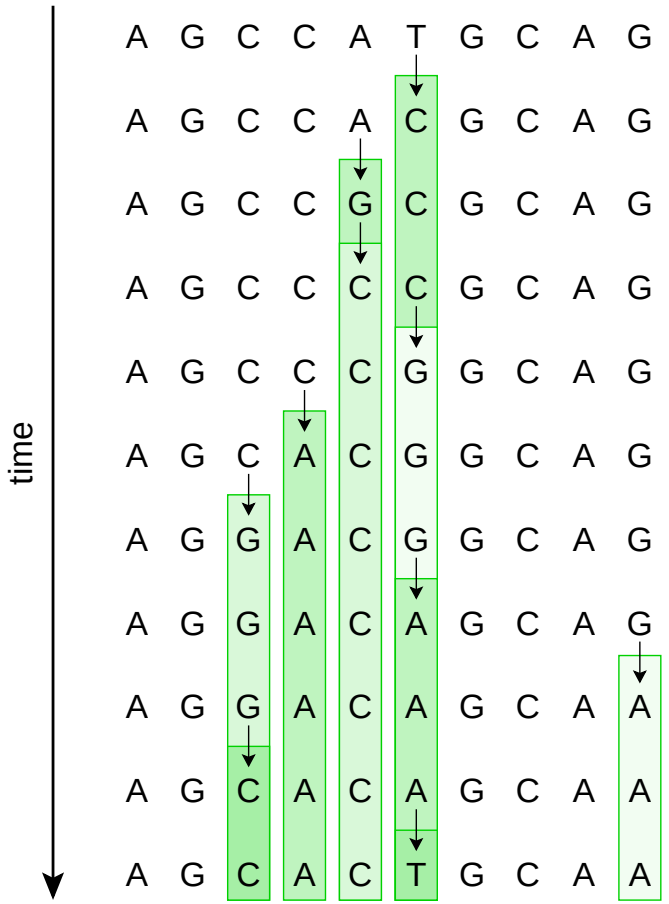
Traces of Sequence Evolution



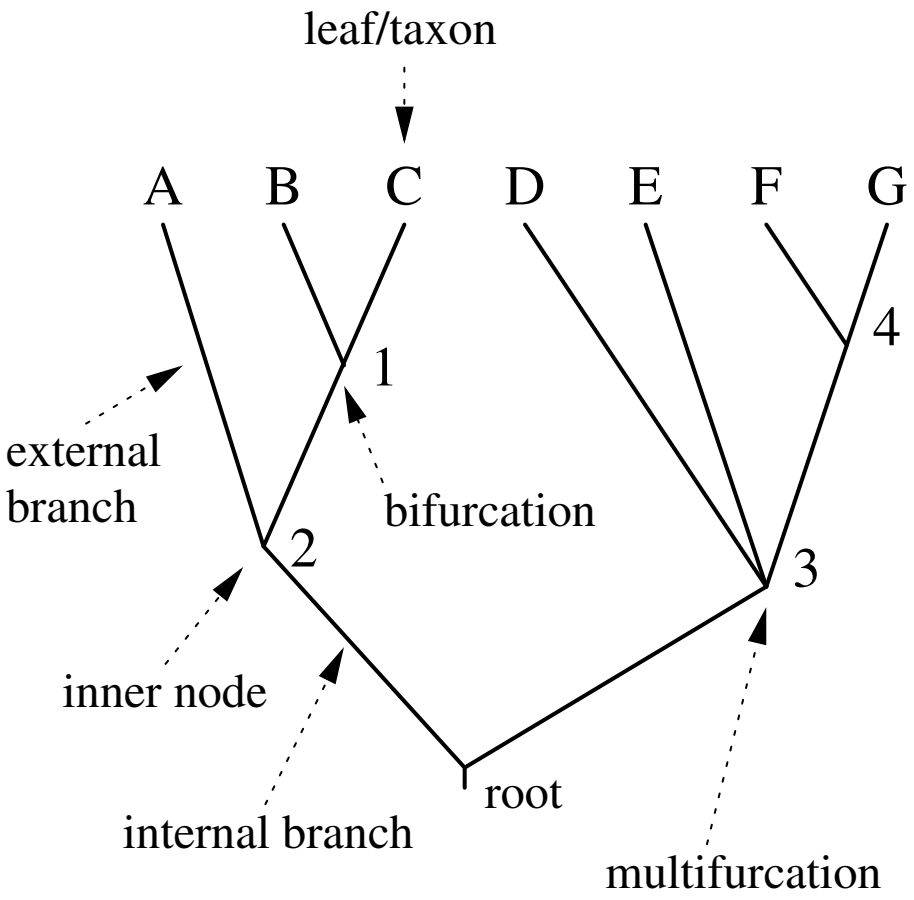
Traces of Sequence Evolution



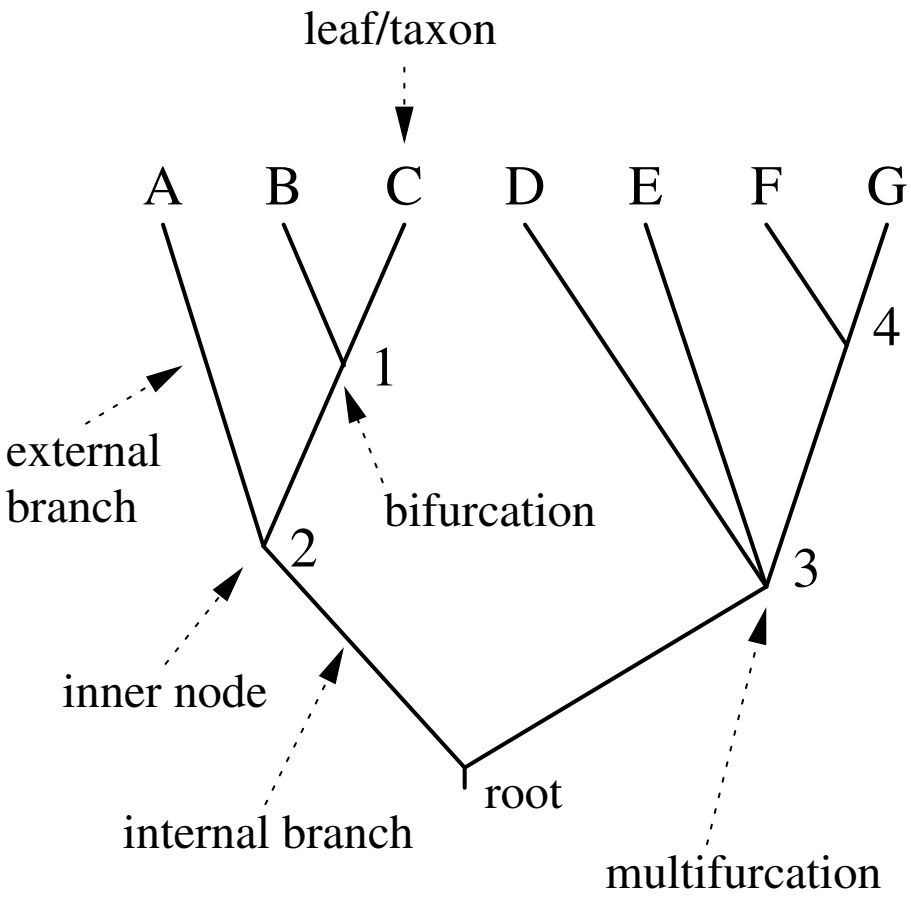
Traces of Sequence Evolution

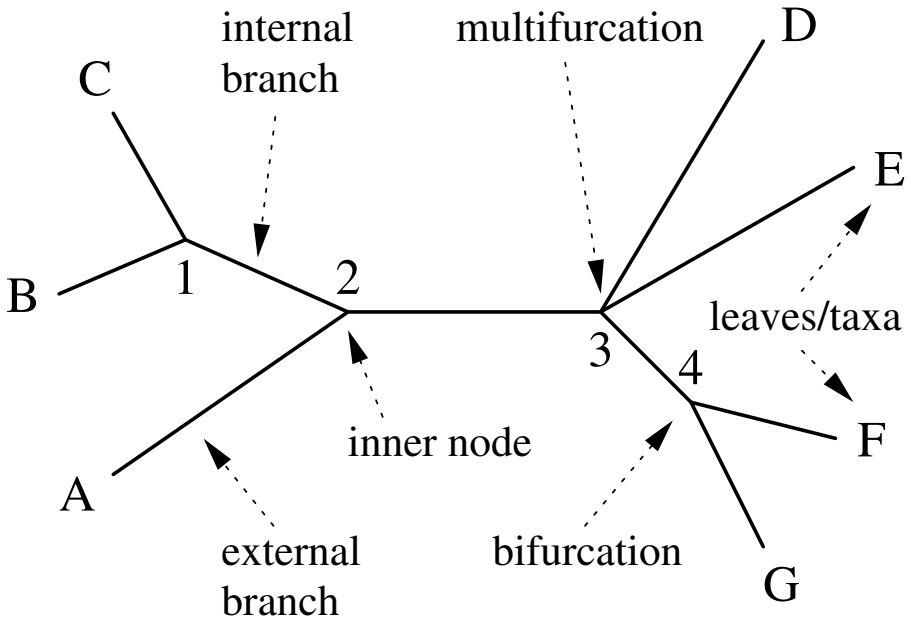


Some Notation

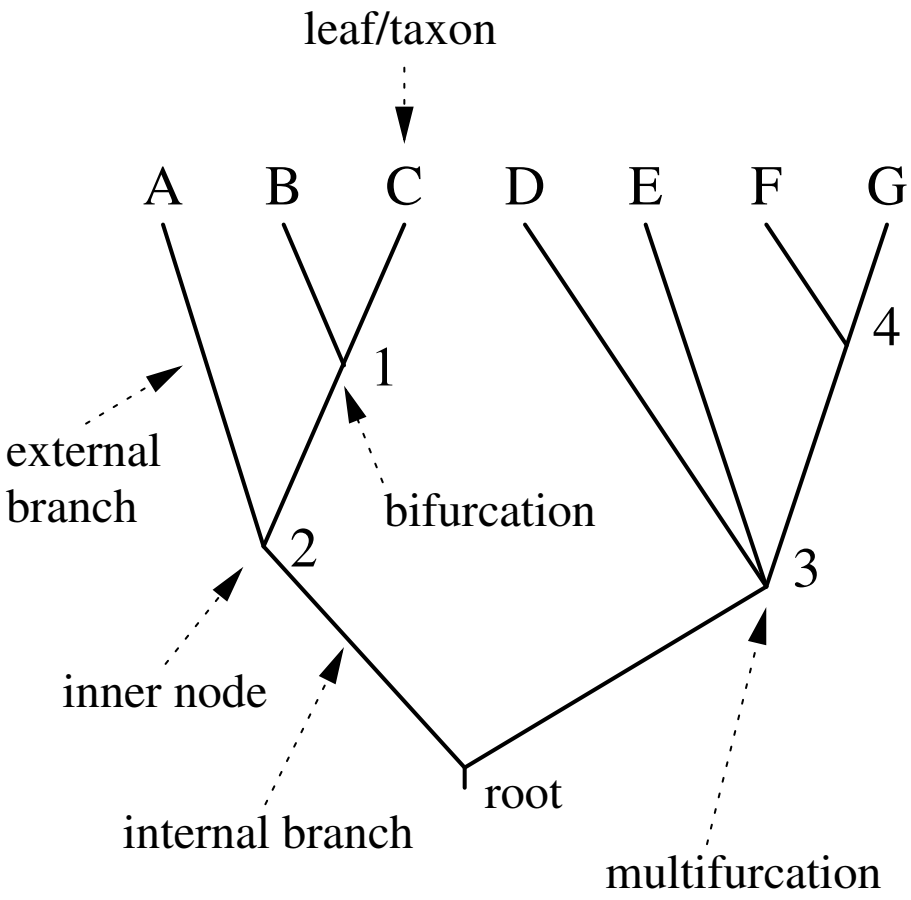


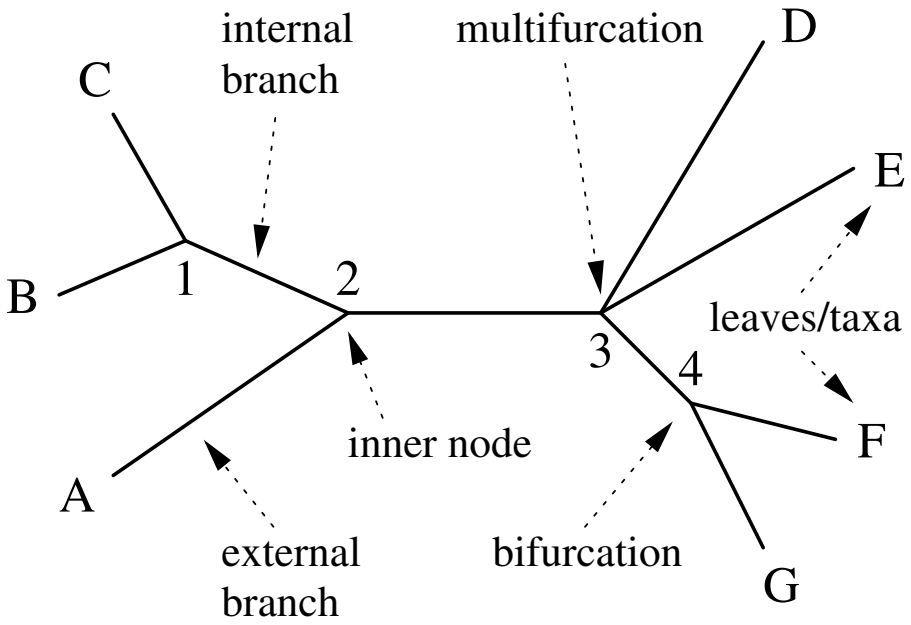
Some Notation





Some Notation





Note: branch = edge = split, external node = leaf = taxon are used interchangebly.

Main Types of Phylogenetic Methods

Data	Method	Evaluation Criterion
Characters (Alignment)	**Maximum Parsimony**	Parsimony
	Statistical Approaches: Likelihood, Bayesian	Evolutionary Models
Distances	**Distance Methods**	Evolutionary Models

The most simple tree: How to get distances?

The most simple tree could be seen as two sequences and the distance between them.

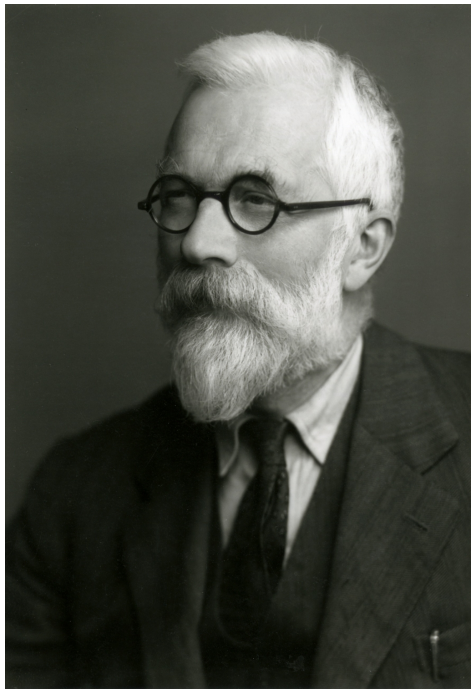
Distances can be computed in various ways...

The most simple tree: How to get distances?

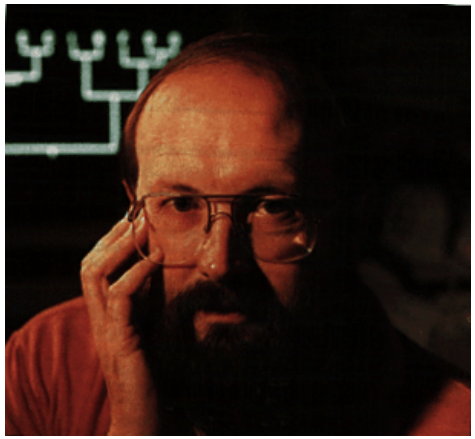
The most simple tree could be seen as two sequences and the distance between them.

Distances can be computed in various ways. . .

Usually via Maximum Likelihood.



Ronald Fisher (1890-1962)



Joe Felsenstein (born 1942)

Introduction: ML on Coin Tossing

Given a box with 3 coins with different levels of fairness $\left(\frac{1}{3}, \frac{1}{2}, \frac{2}{3} \text{ heads}\right)$

Introduction: ML on Coin Tossing

Given a box with 3 coins with different levels of fairness $\left(\frac{1}{3}, \frac{1}{2}, \frac{2}{3} \text{ heads}\right)$

We take out one coin and toss 20 times:

$$H, T, T, H, H, T, T, T, T, H, T, T, H, T, H, T, T, H, T, T$$

Introduction: ML on Coin Tossing

Given a box with 3 coins with different levels of fairness $\left(\frac{1}{3}, \frac{1}{2}, \frac{2}{3} \text{ heads}\right)$

We take out one coin and toss 20 times:

$$H, T, T, H, H, T, T, T, T, H, T, T, H, T, H, T, T, H, T, T$$

Probability

$$p(k \text{ heads in } n \text{ tosses} | \theta)$$

Aim: The ML approach seaches for that parameter set θ for the generating process which maximizes the probability of our given data.

Introduction: ML on Coin Tossing

Given a box with 3 coins with different levels of fairness $\left(\frac{1}{3}, \frac{1}{2}, \frac{2}{3} \text{ heads}\right)$

We take out one coin and toss 20 times:

$$H, T, T, H, H, T, T, T, T, H, T, T, H, T, H, T, T, H, T, T$$

	Probability		**Likelihood**
$p(k$ heads in n tosses$\|\theta)$	$\equiv$	$L(\theta\|k$ heads in n tosses$)$	

Aim: The ML approach seaches for that parameter set θ for the generating process which maximizes the probability of our given data.

Hence, "likelihood flips the probability around."

Introduction: ML on Coin Tossing

Given a box with 3 coins with different levels of fairness $\left(\frac{1}{3}, \frac{1}{2}, \frac{2}{3} \text{ heads}\right)$

We take out one coin and toss 20 times:

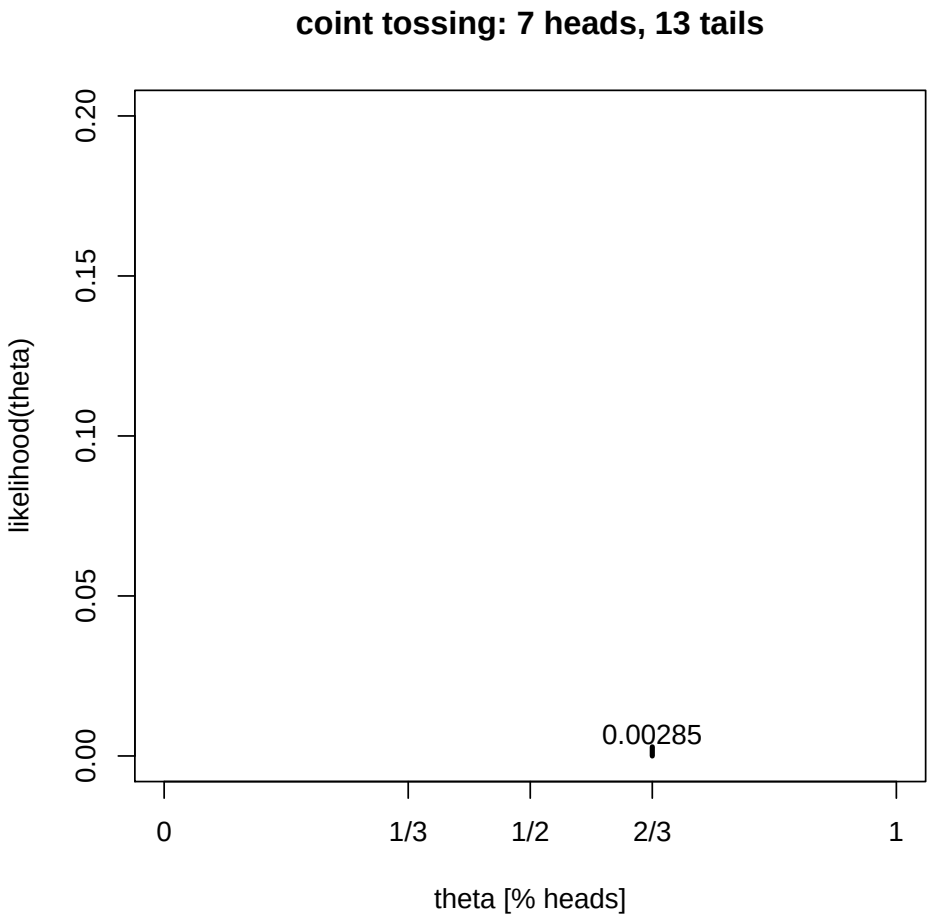
$$H, T, T, H, H, T, T, T, T, H, T, T, H, T, H, T, T, H, T, T$$

	Probability		**Likelihood**
$p(k \text{ heads in } n \text{ tosses} \mid \theta)$	$\equiv$	$L(\theta \mid k \text{ heads in } n \text{ tosses})$	
	$=$	$\binom{n}{k}\theta^k(1-\theta)^{n-k}$	
		(here binomial distribution)	

Aim: The ML approach seaches for that parameter set θ for the generating process which maximizes the probability of our given data.

Hence, "likelihood flips the probability around."

Introduction: ML on Coin Tossing (Estimate)

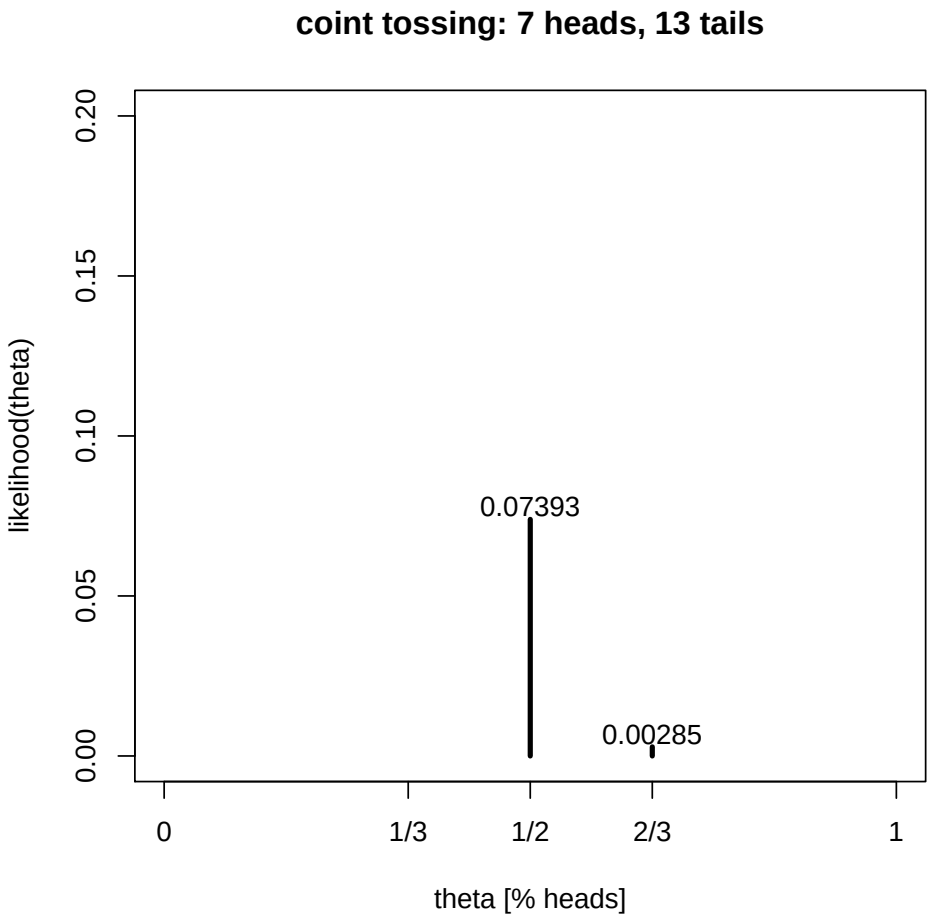


Three coin case

$$L(\theta|7 \text{ heads in } 20) = \binom{20}{7}\theta^7(1-\theta)^{13}$$

for each coin $\theta \in \left\{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\right\}$

Introduction: ML on Coin Tossing (Estimate)

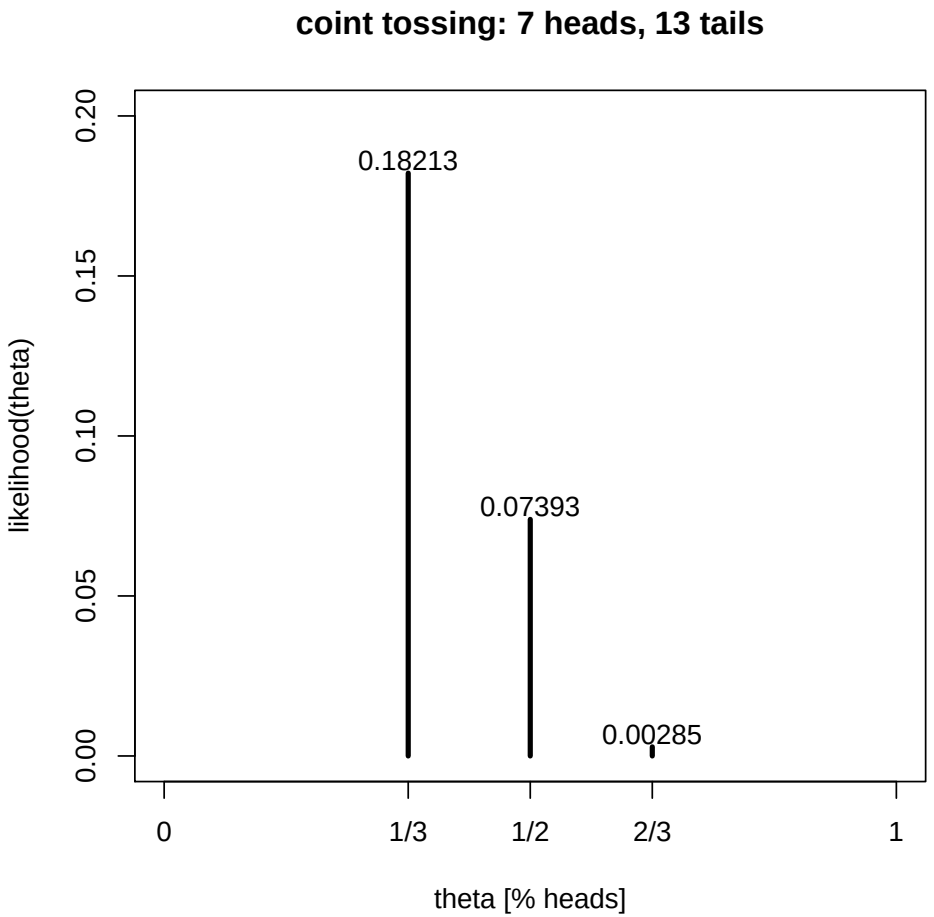


Three coin case

$$L(\theta|7 \text{ heads in } 20) = \binom{20}{7}\theta^7(1-\theta)^{13}$$

for each coin $\theta \in \left\{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\right\}$

Introduction: ML on Coin Tossing (Estimate)

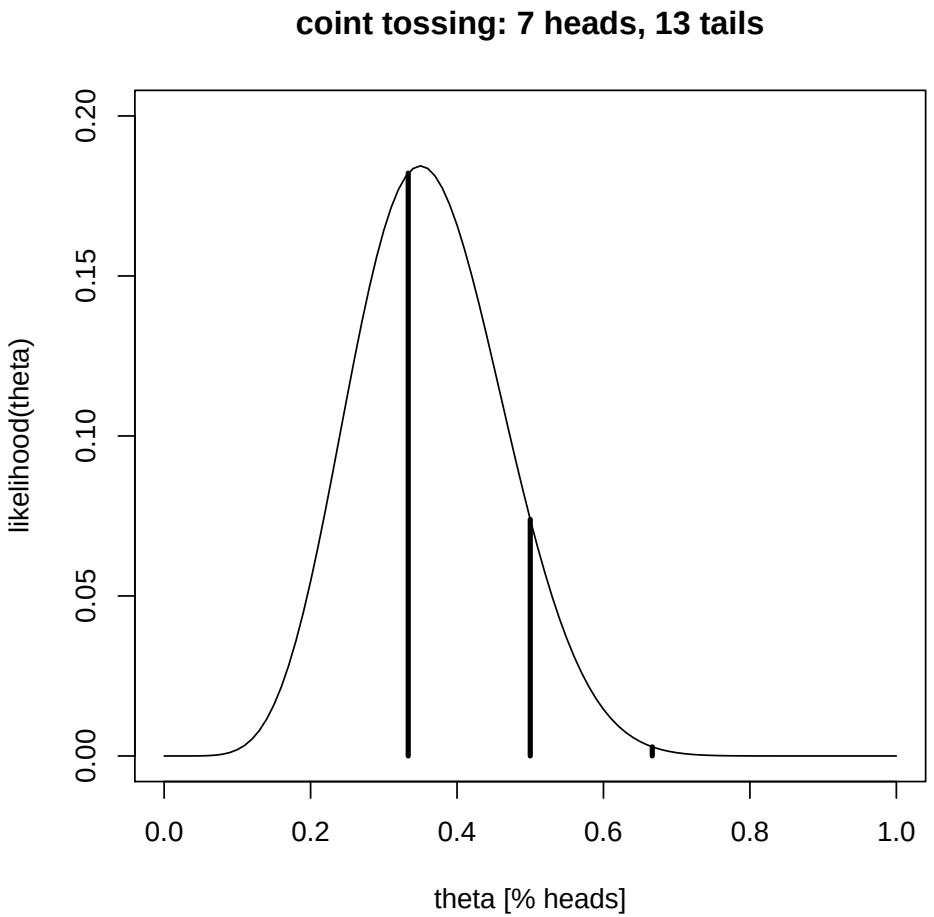


Three coin case

$$L(\theta|7 \text{ heads in } 20) = \binom{20}{7}\theta^7(1-\theta)^{13}$$

for each coin $\theta \in \left\{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\right\}$

Introduction: ML on Coin Tossing (Estimate)



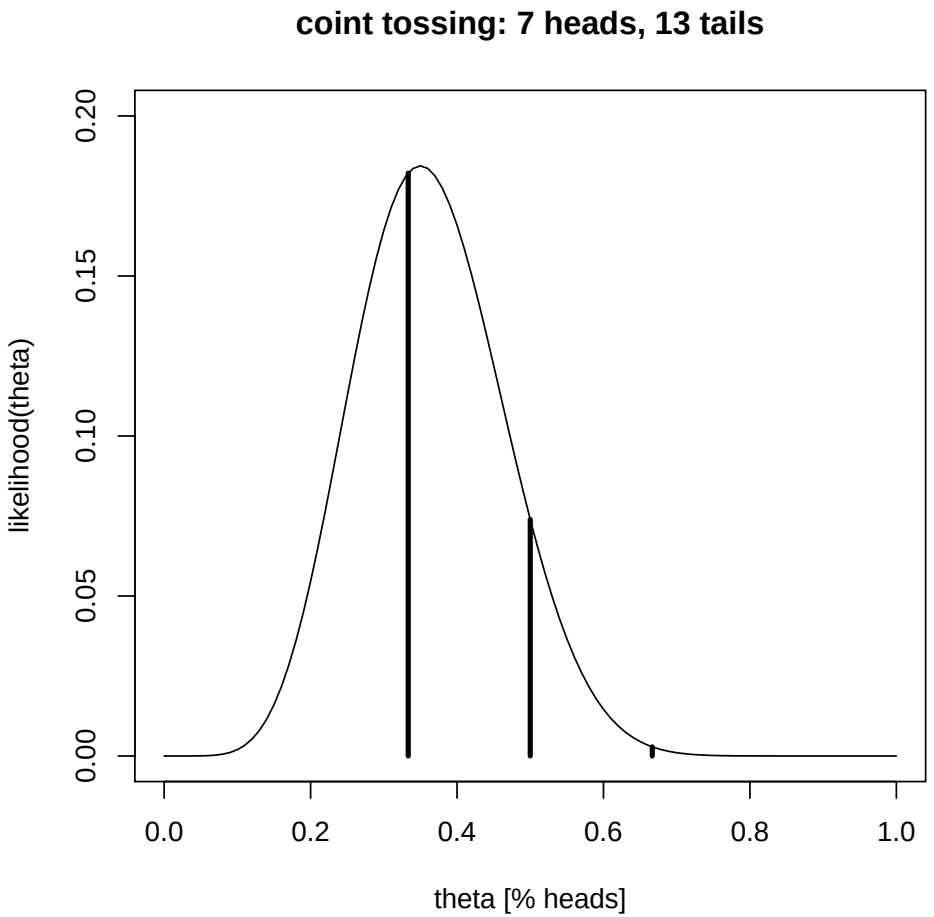
Three coin case

$$L(\theta|7 \text{ heads in } 20) = \binom{20}{7}\theta^7(1-\theta)^{13}$$

for each coin $\theta \in \left\{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\right\}$

For infinitely many coins
$\theta = (0...1)$

Introduction: ML on Coin Tossing (Estimate)



Three coin case

$$L(\theta|7 \text{ heads in } 20) = \binom{20}{7}\theta^7(1-\theta)^{13}$$

for each coin $\theta \in \left\{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\right\}$

For infinitely many coins
$\theta = (0...1)$

ML estimate: $L(\hat{\theta}) = 0.1844$ where coin shows $\hat{\theta} = 0.35$ heads

From Coins to Phylogenies?

While the coin tossing example might look easy, in phylogenetic analysis, the parameter (set) θ comprises:

- evolutionary model
- its parameters
- tree topology
- its branch lengths

That means a high dimensional optimization problem.
Hence, some parameters are often estimated/set separately.

Modeling Evolution

- Evolution is usually modeled as a

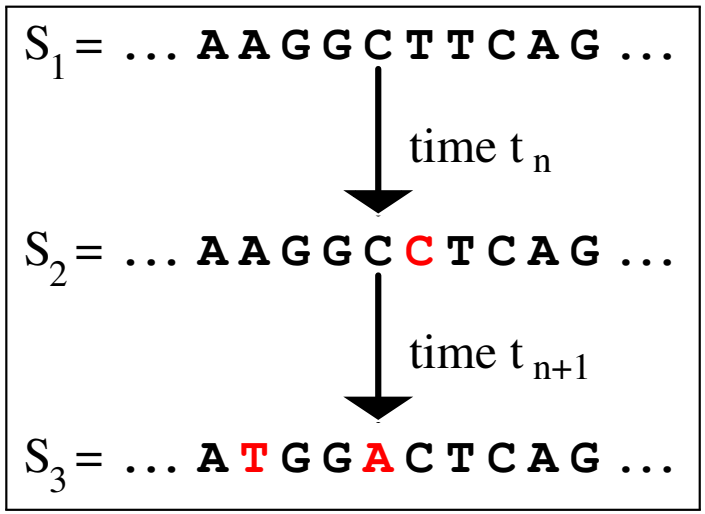
  stationary, time-reversible Markov process.

- What does that mean?

Assumptions on Evolution

Markov Process

The (evolutionary) process evolves without memory, i.e. sequence S_2 mutates to S_3 during time t_{n+1} independent of state of S_1.



Assumptions on Evolution

Stationary:
The overall character frequencies π_j of the nucleotides or amino acids are in an equilibrium and remain constant.

Time-Reversible:
Mutations in either direction are equally likely

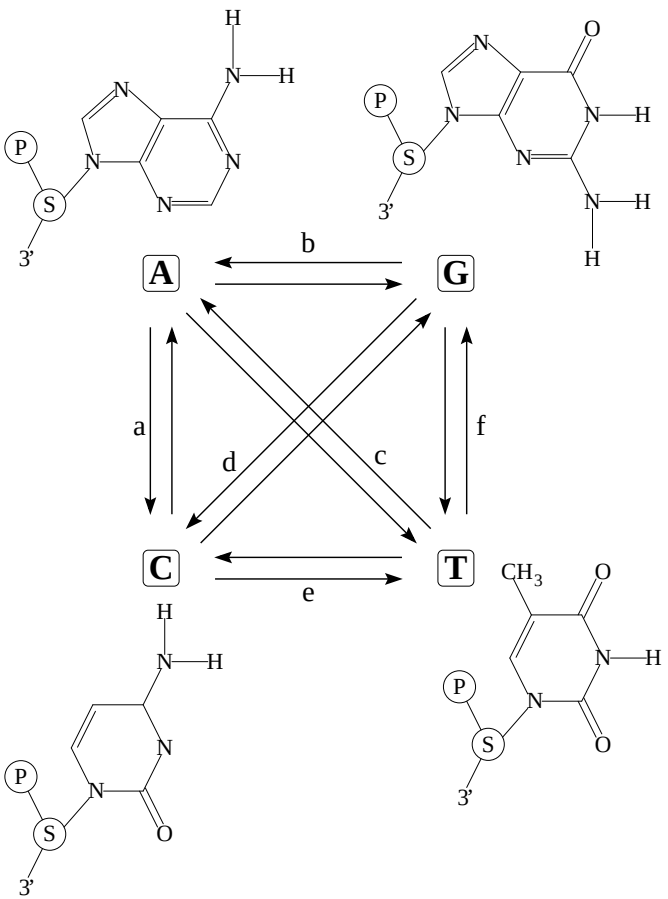
$$\pi_i \cdot P_{ij}(t) = P_{ji}(t) \cdot \pi_j$$

This means a mutation is as likely as its back mutation.

$$P(i \rightarrow j) = P(i \leftarrow j) \qquad \text{(JC69)}$$

Substitution Models

Evolutionary models are often described using a substitution rate matrix R and character frequencies Π. Here, 4×4 matrix for DNA models:

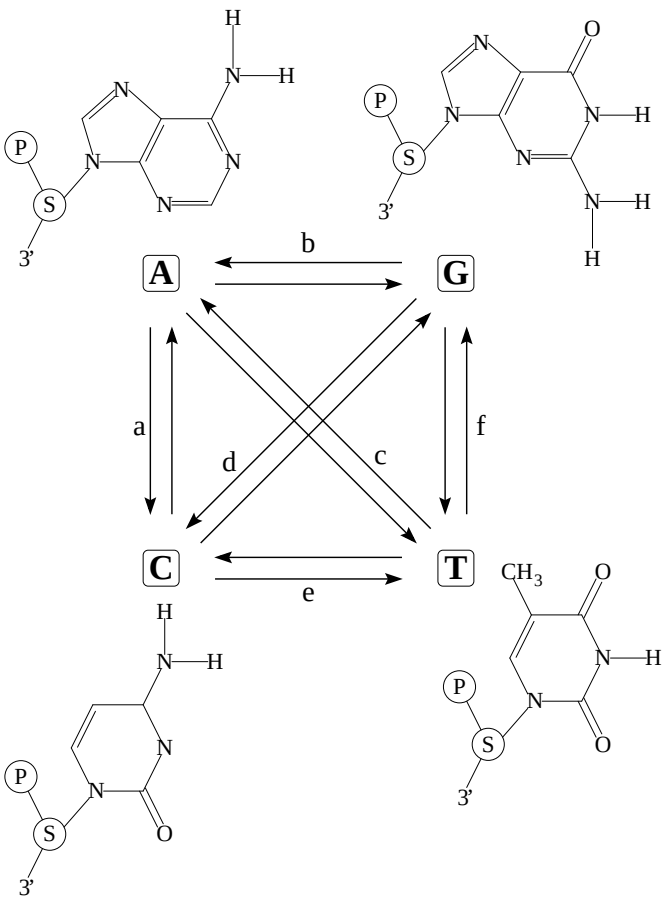


$$R = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} - & a & b & c \\ a & - & d & e \\ b & d & - & f \\ c & e & f & - \end{pmatrix} \end{array}$$

$$\Pi = (\pi_A, \pi_C, \pi_G, \pi_T)$$

Substitution Models

Evolutionary models are often described using a substitution rate matrix R and character frequencies Π. Here, 4×4 matrix for DNA models:



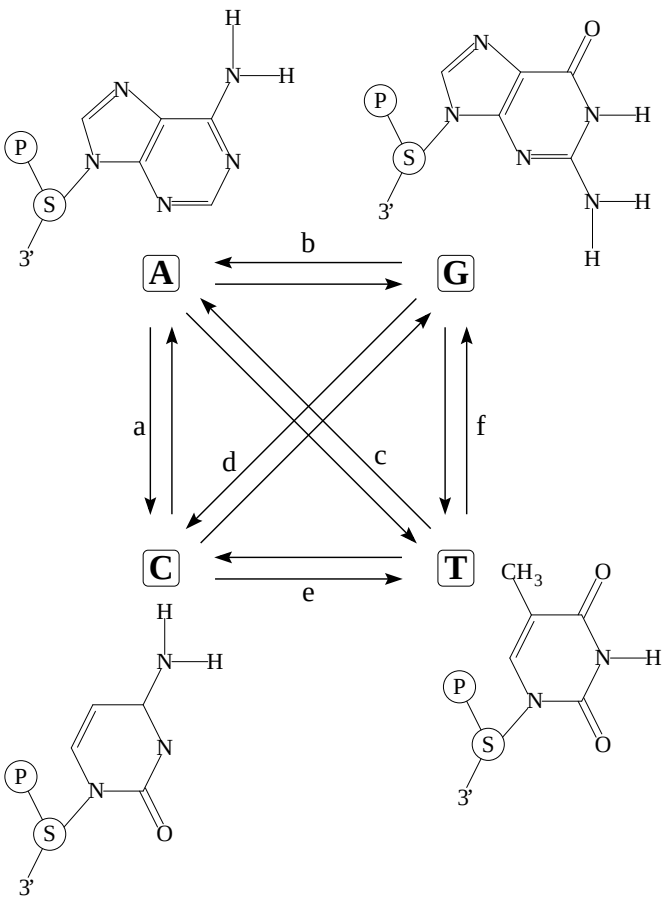
$$R = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} - & a & b & c \\ a & - & d & e \\ b & d & - & f \\ c & e & f & - \end{pmatrix} \end{array}$$

$$\Pi = (\pi_A, \pi_C, \pi_G, \pi_T)$$

From R and Π we reconstruct a substitution probability matrix P

Substitution Models

Evolutionary models are often described using a substitution rate matrix R and character frequencies Π. Here, 4×4 matrix for DNA models:

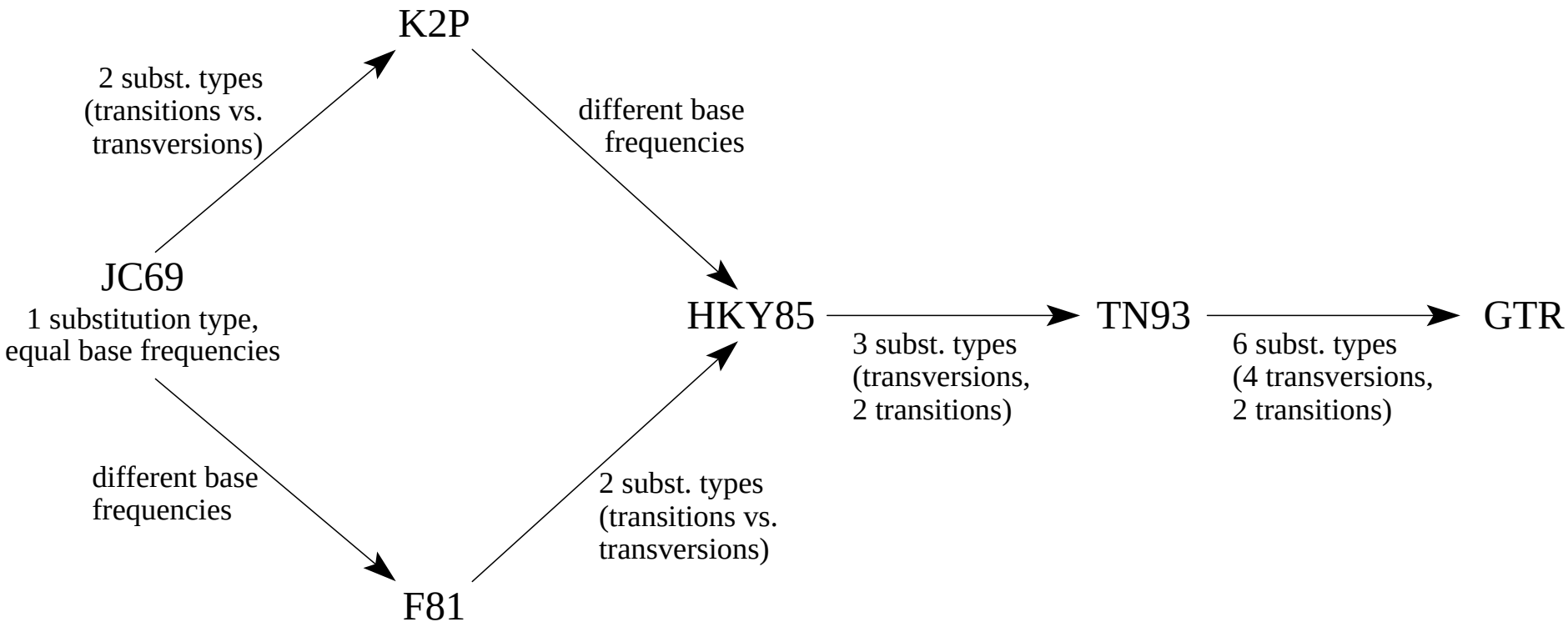


$$R = \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} - & a & b & c \\ a & - & d & e \\ b & d & - & f \\ c & e & f & - \end{pmatrix} \end{array}$$

$$\Pi = (\pi_A, \pi_C, \pi_G, \pi_T)$$

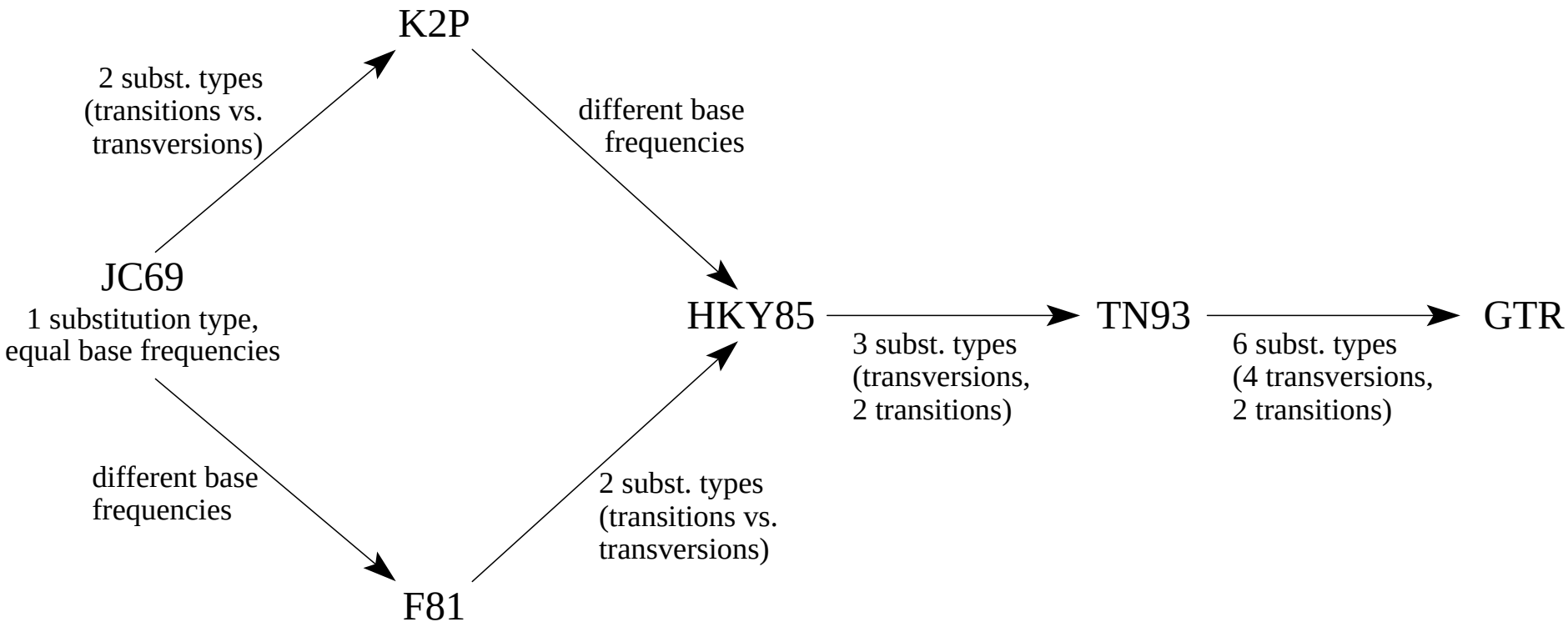
From R and Π we reconstruct a substitution probability matrix P, where $P_{ij}(t)$ is the probability of changing $i \rightarrow j$ in time t.

Relations between DNA models



Further modification:
rate heterogeneity

Relations between DNA models



Further modification:

rate heterogeneity: invariant sites, Γ-distributed rates, mixed.

Protein Models

Generally this is the same for protein sequences, but with 20×20 matrices. Some protein models are:

- Poisson model ("JC69" for proteins, rarely used)
- Dayhoff (Dayhoff et al., 1978, general matrix)
- JTT (Jones et al., 1992, general matrix)
- WAG (Whelan & Goldman, 2000, more distant sequences)
- VT (Müller & Vingron, 2000, distant sequences)
- mtREV (Adachi & Hasegawa, 1996, mitochondrial sequences)
- cpREV (Adachi et al., 2000, cloroplast sequences)
- mtMAM (Yang et al., 1998, Mammalian mitochondria)
- mtART (Abascal et al., 2007, Arthropod mitochondria)
- rtREV (Dimmic et al., 2002, reverse transcriptases)
- ...
- ~~BLOSUM 62 (Henikoff & Henikoff, 1992)~~ $\rightarrow$ for database searching

Computing ML Distances Using $\mathbf{P_{ij}(t)}$

The Likelihood of sequence s evolving to s' in time t:

$$L(t|s \to s') = \prod_{i=1}^{m} \Big(\Pi(s_i) \cdot P_{s_i s'_i}(t) \Big)$$

Likelihood surface for two sequences under JC69:

GATCCTGAGAGAAATAAAC $= s'$
GGTCCTGACAGAAATAAAC $= s$

Computing ML Distances Using $\mathbf{P_{ij}(t)}$

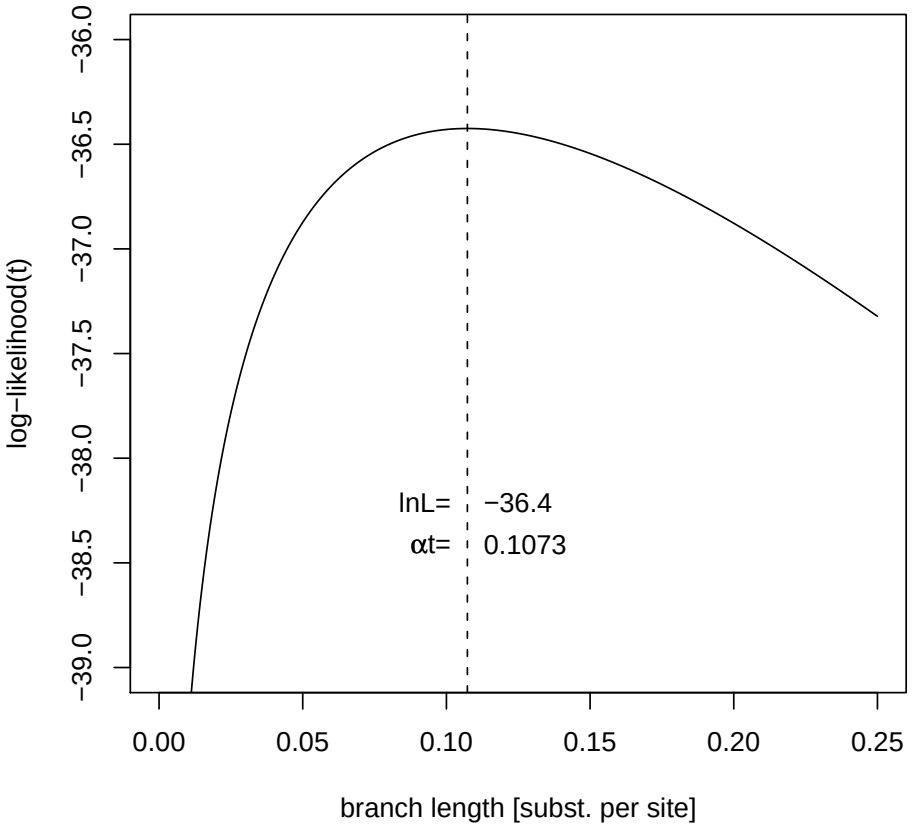
The Likelihood of sequence s evolving to s' in time t:

$$L(t|s \to s') = \prod_{i=1}^{m} \Big(\Pi(s_i) \cdot P_{s_i s'_i}(t) \Big)$$

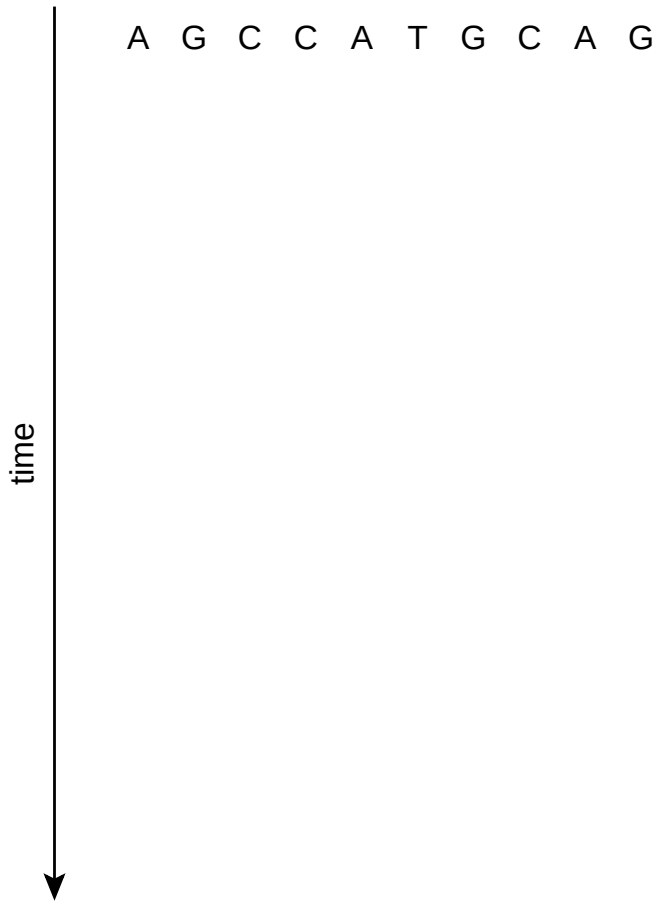
Likelihood surface for two sequences under JC69:

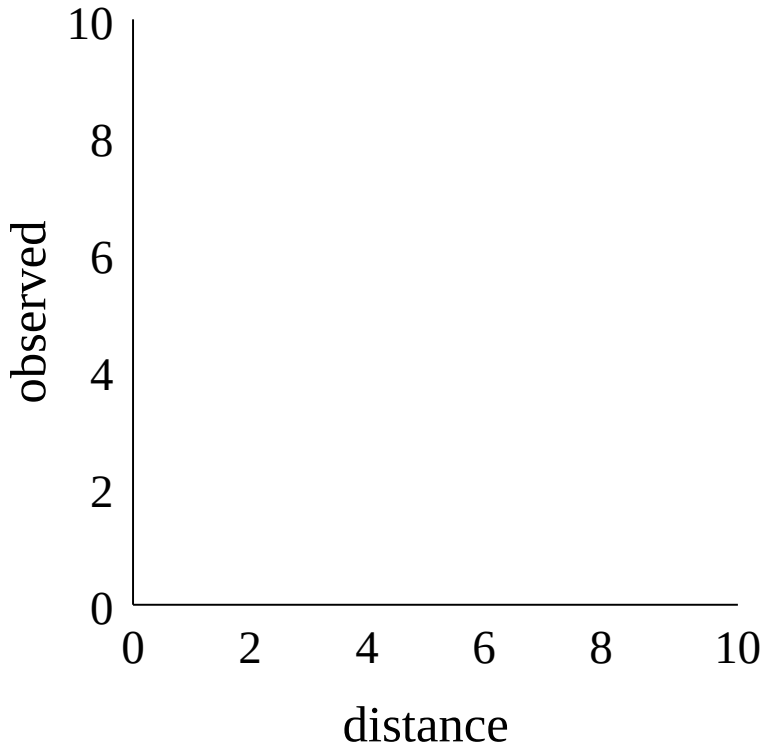
GATCCTGAGAGAAATAAAC $= s'$
GGTCCTGACAGAAATAAAC $= s$

Note: we do not compute the probability of the distance t but that of the data $D = \{s, s'\}$.

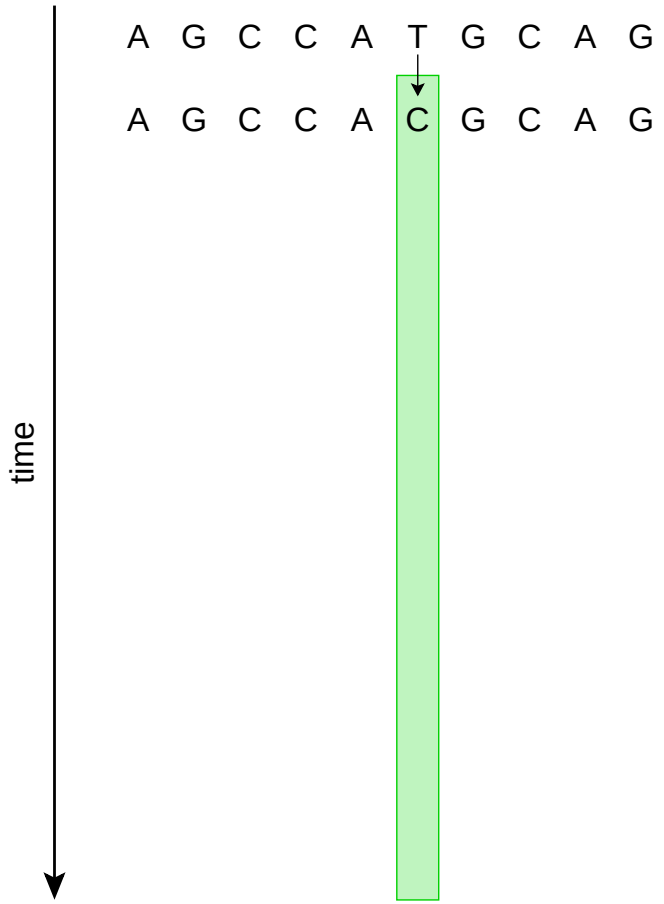


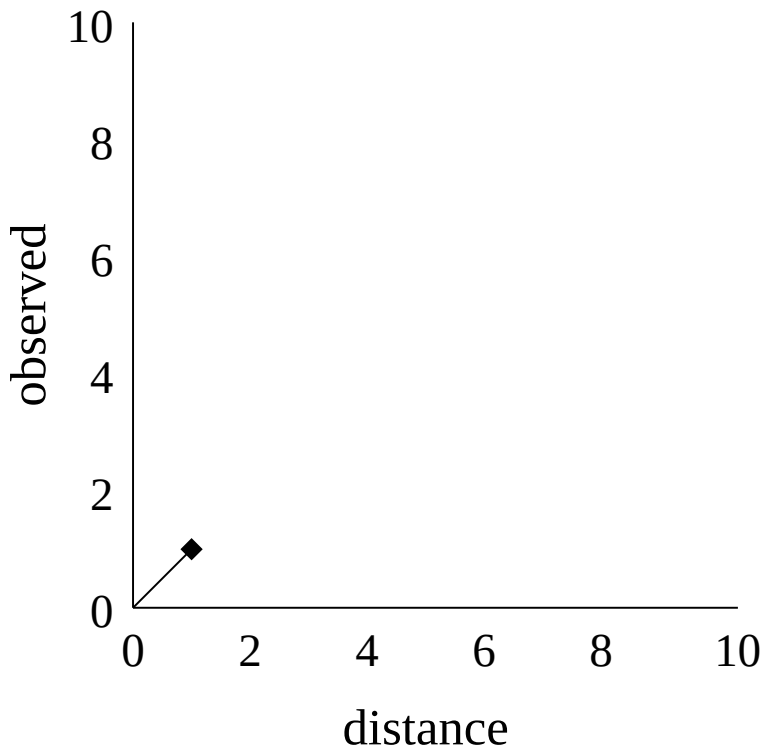
Jukes-Cantor Correction for Multiple Mutations



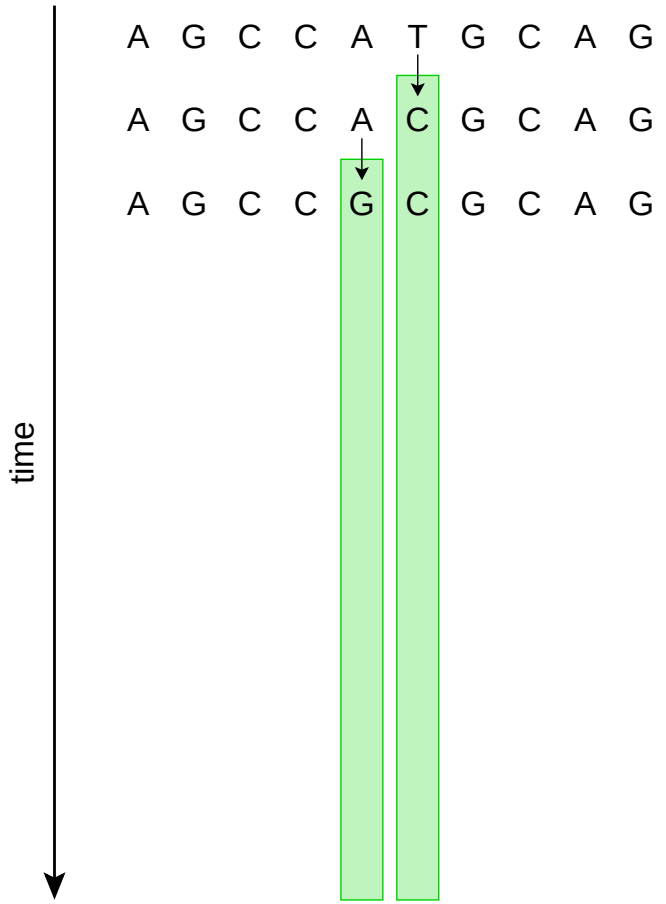


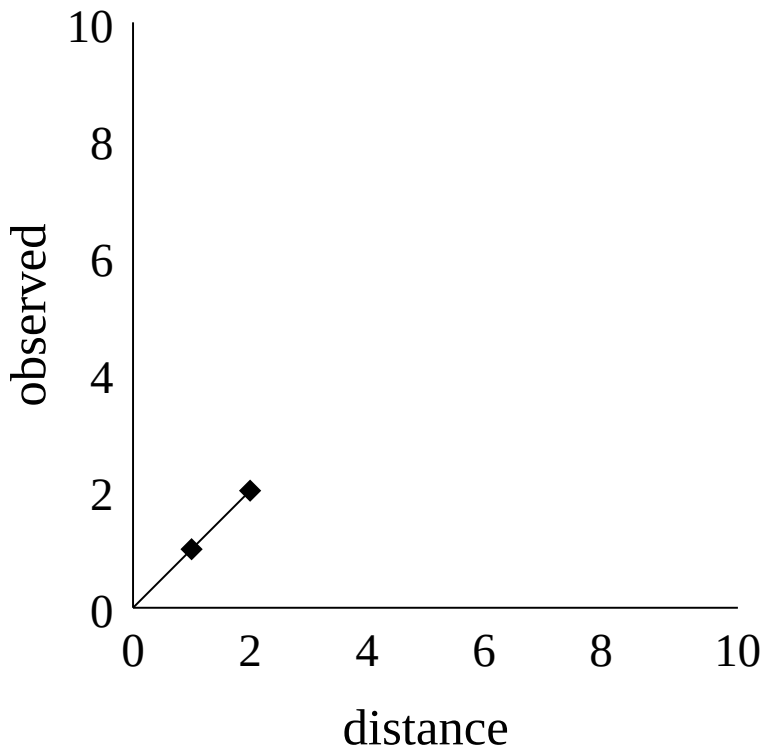
Jukes-Cantor Correction for Multiple Mutations



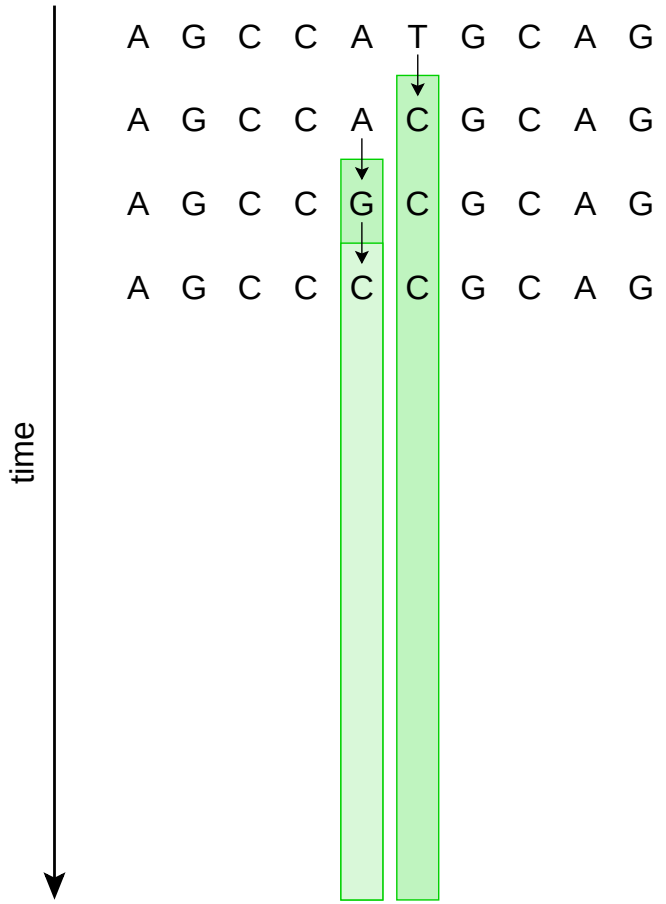


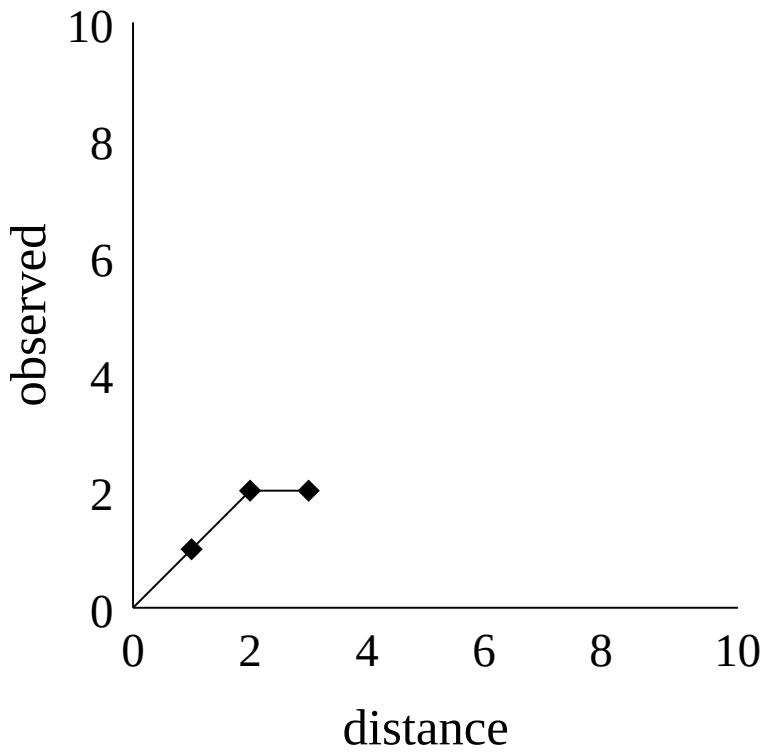
Jukes-Cantor Correction for Multiple Mutations



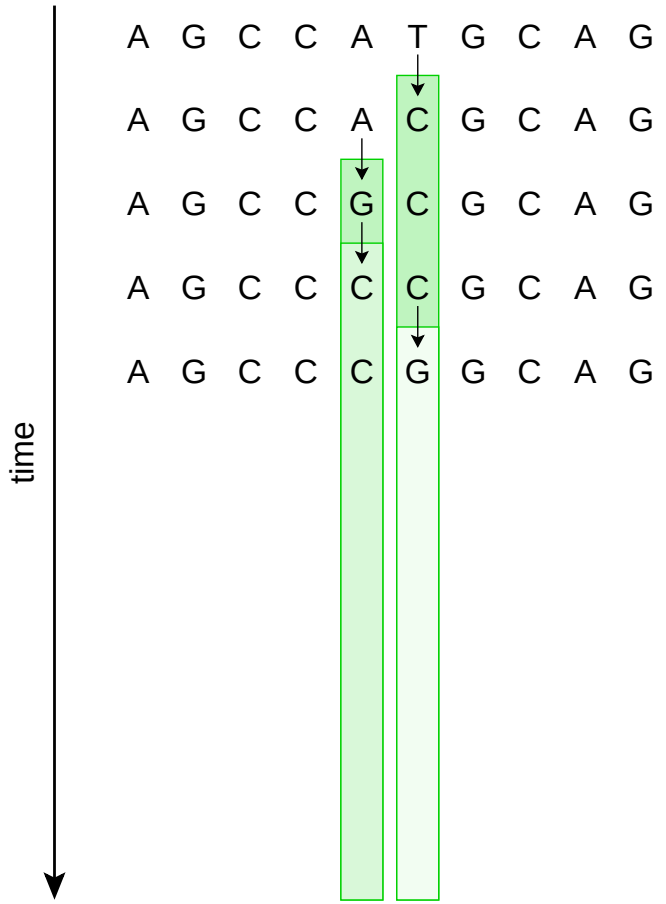


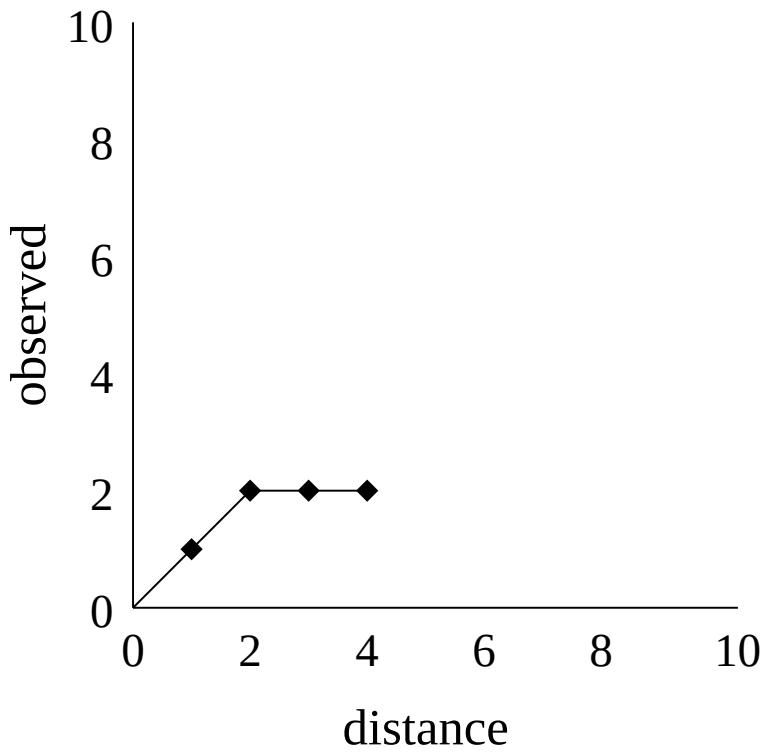
Jukes-Cantor Correction for Multiple Mutations



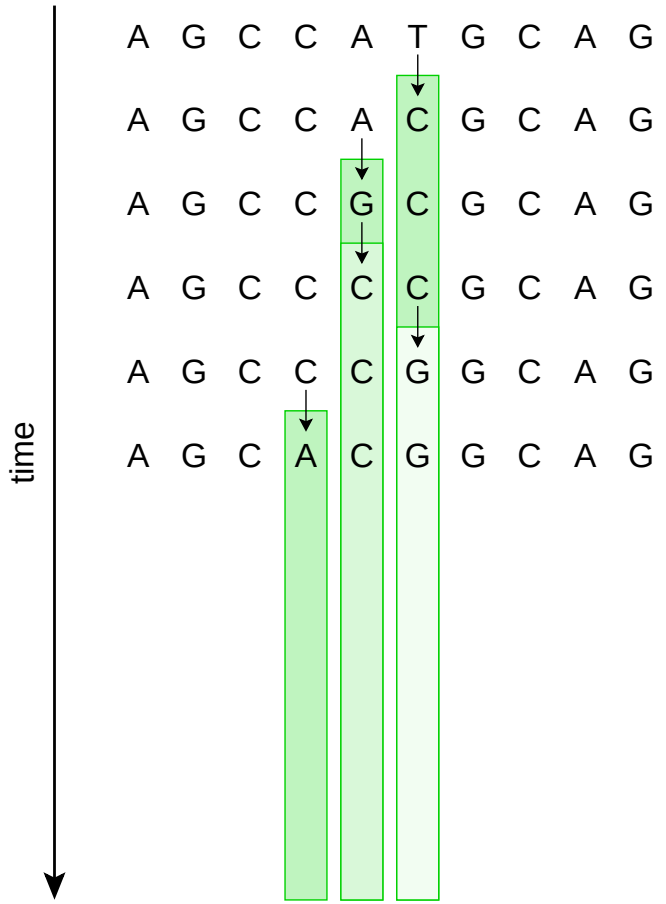


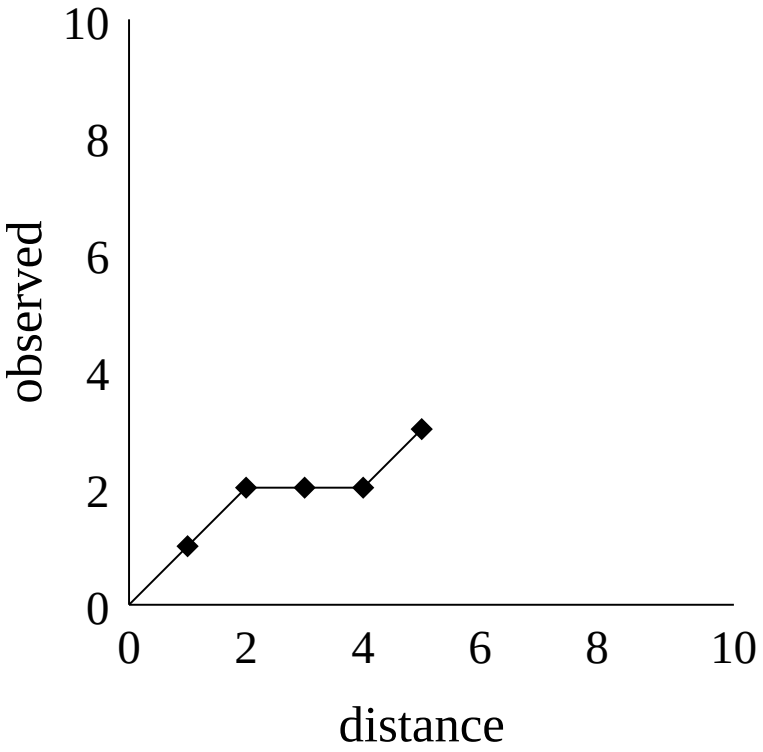
Jukes-Cantor Correction for Multiple Mutations





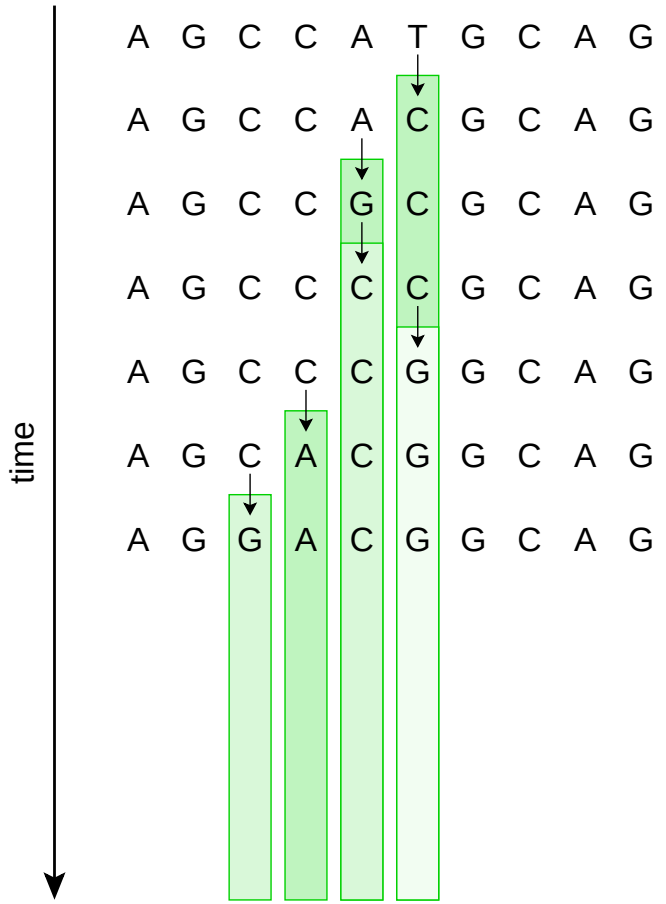
Jukes-Cantor Correction for Multiple Mutations

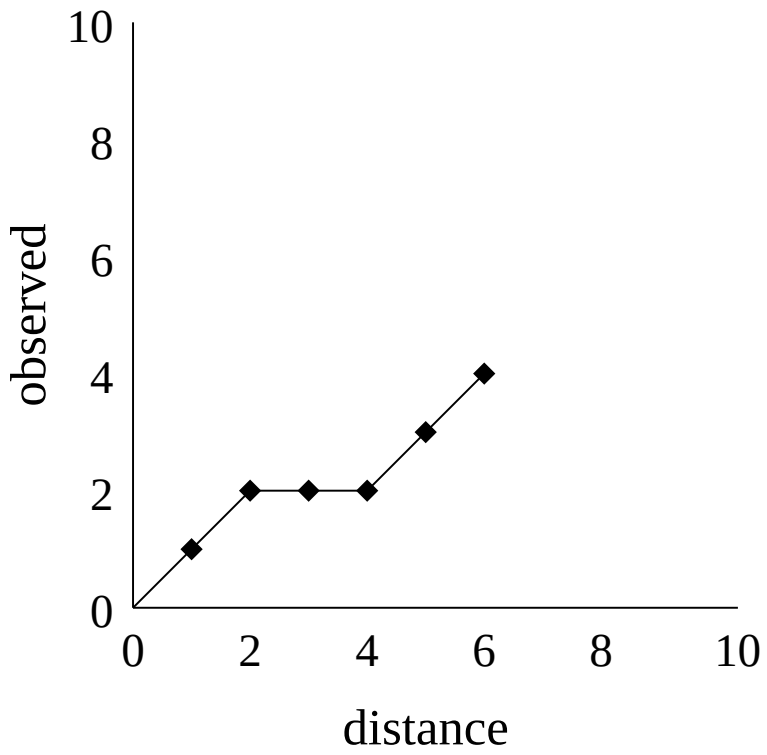




The substitution process is commonly modeled as a Markov process.

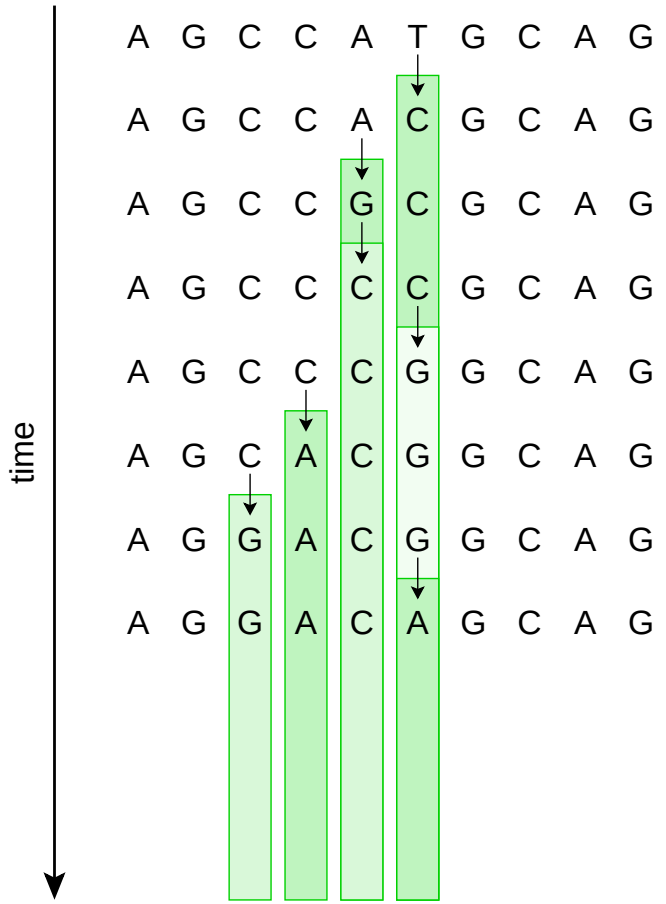
Jukes-Cantor Correction for Multiple Mutations

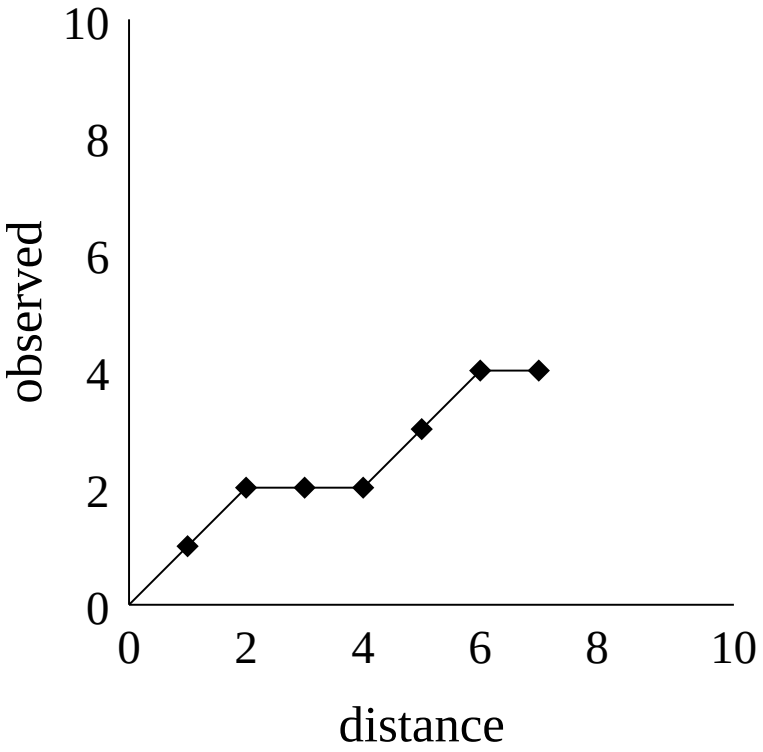




The substitution process is commonly modeled as a Markov process.

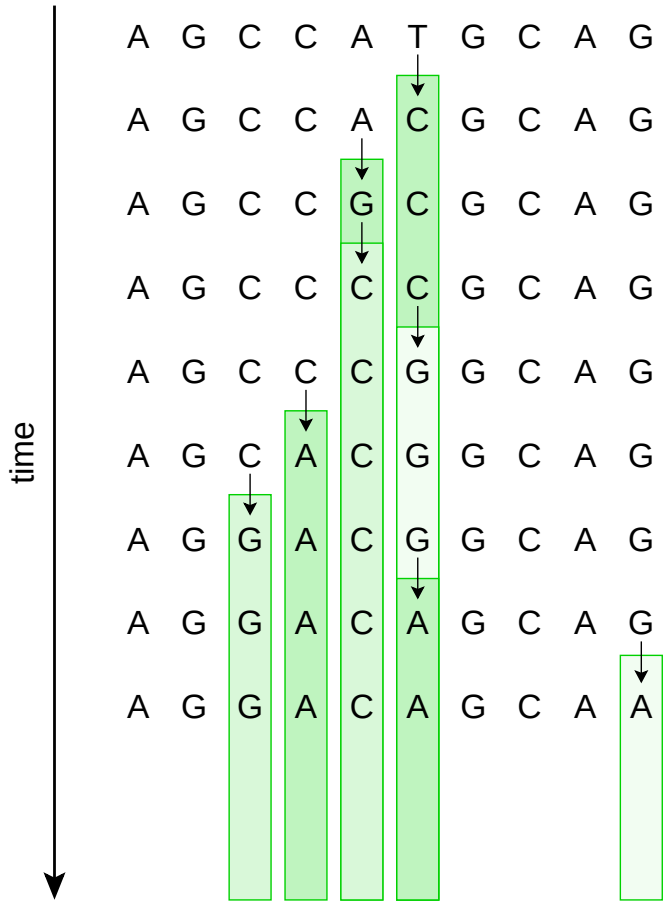
Jukes-Cantor Correction for Multiple Mutations

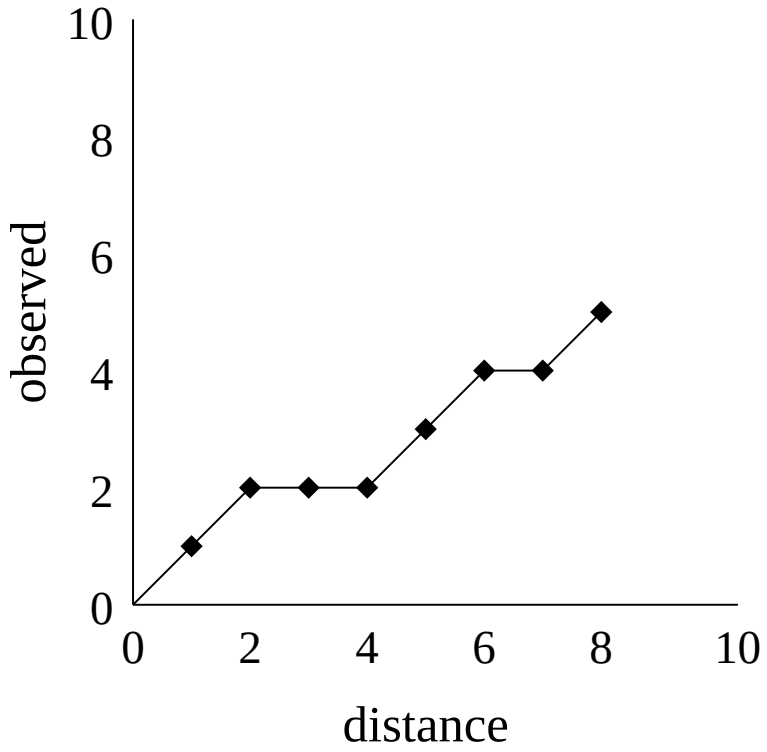




The substitution process is commonly modeled as a Markov process.

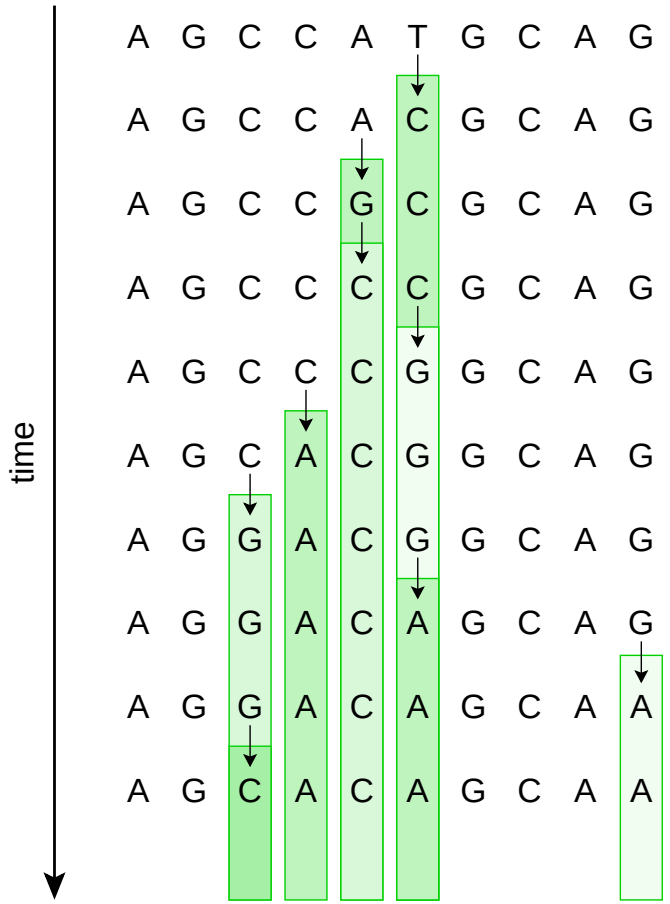
Jukes-Cantor Correction for Multiple Mutations

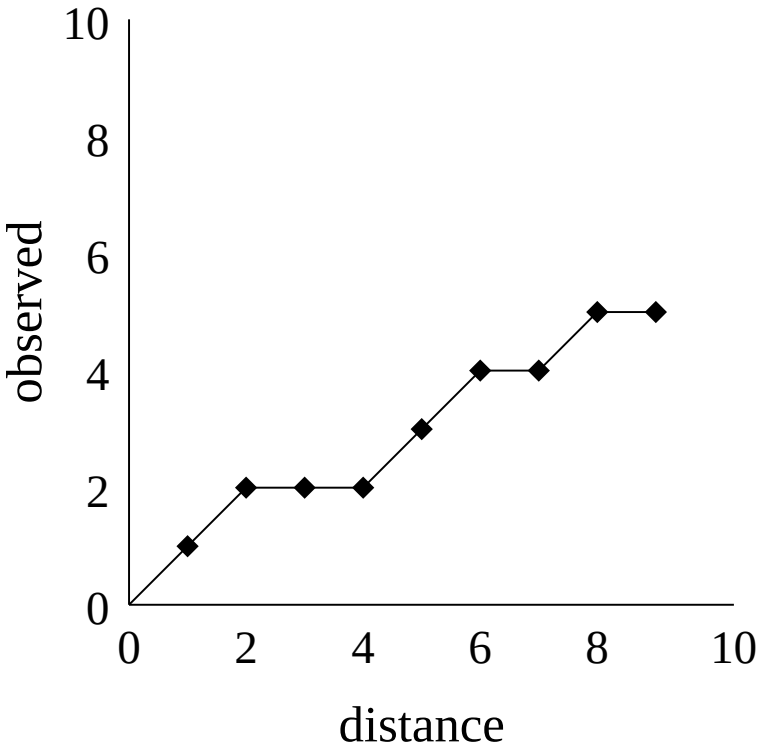




The substitution process is commonly modeled as a Markov process.

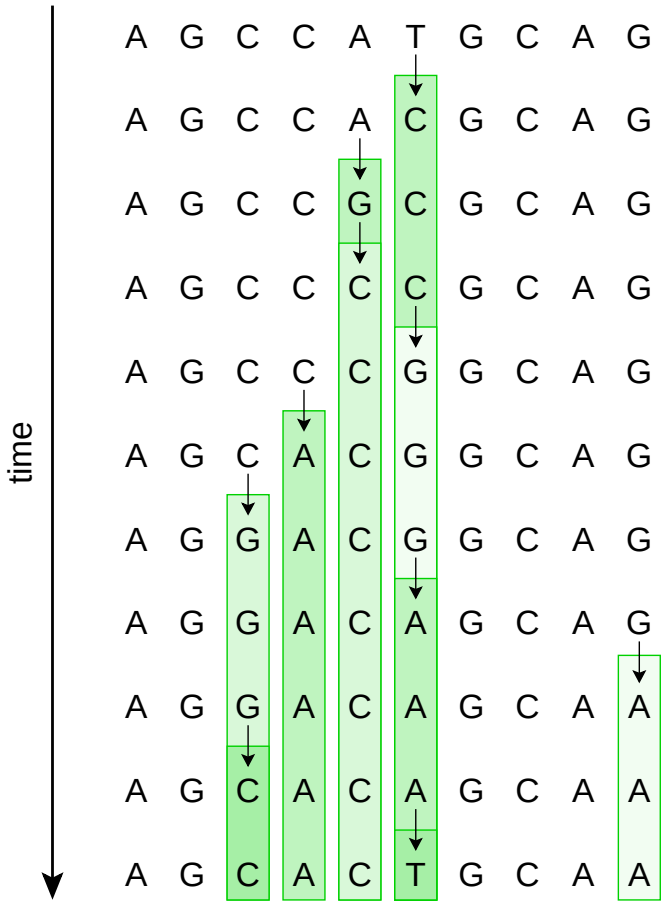
Jukes-Cantor Correction for Multiple Mutations

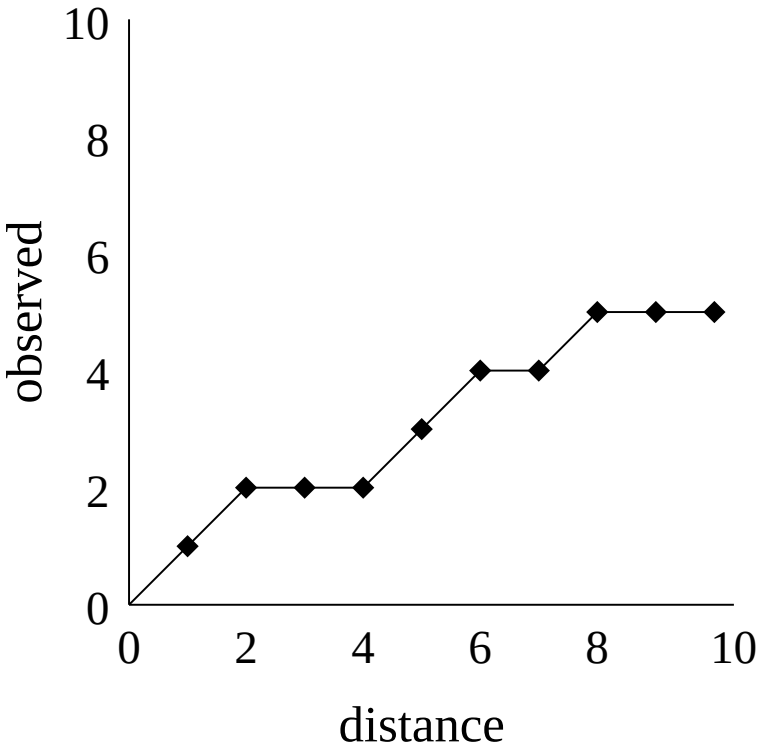




The substitution process is commonly modeled as a Markov process.

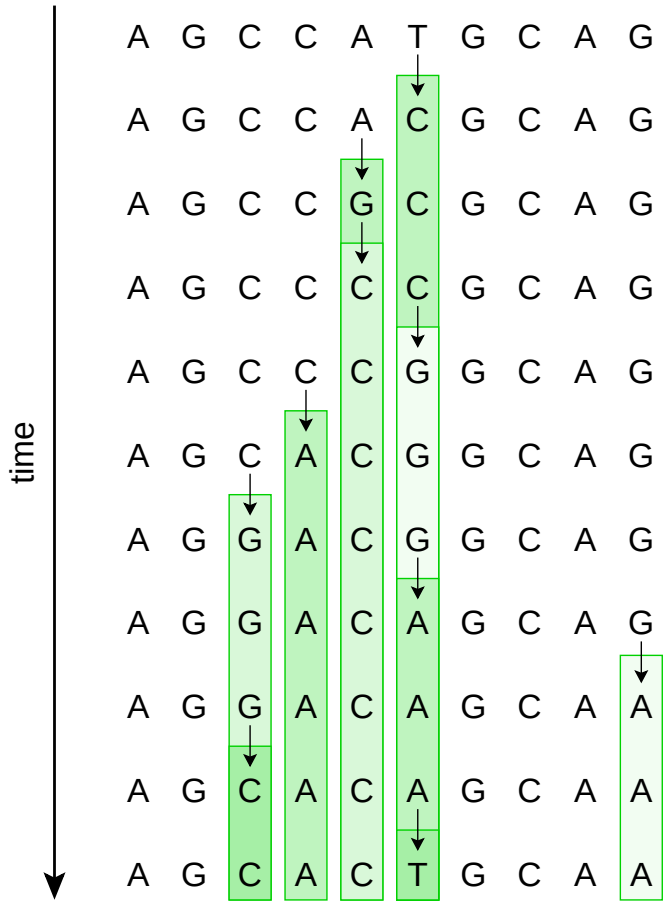
Jukes-Cantor Correction for Multiple Mutations

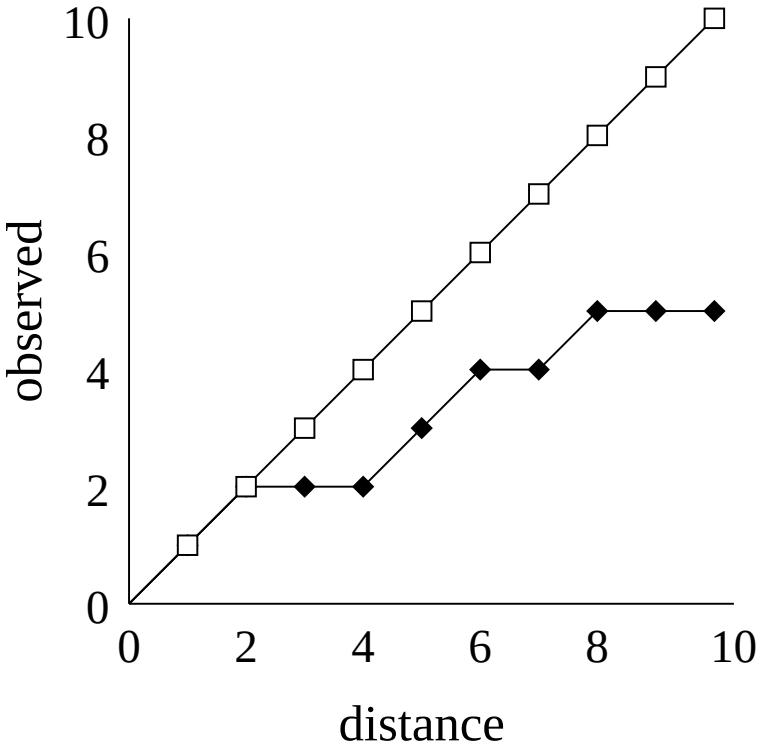




The substitution process is commonly modeled as a Markov process.

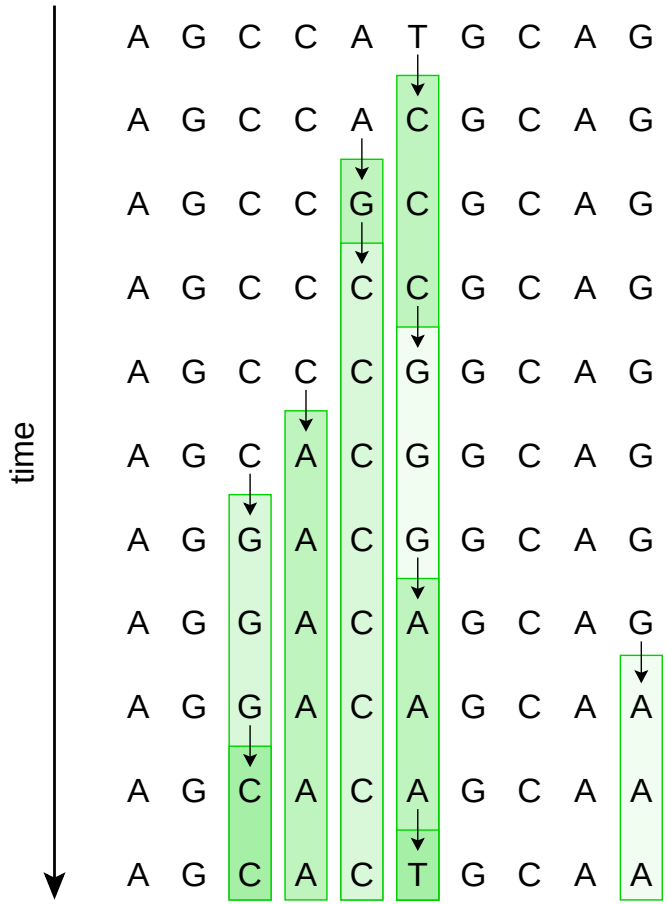
Jukes-Cantor Correction for Multiple Mutations

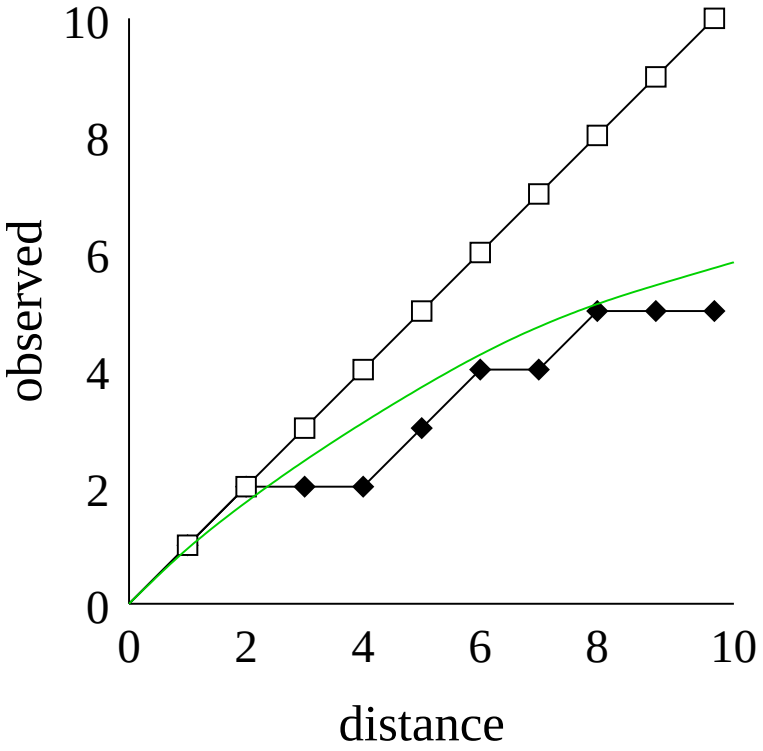




The substitution process is commonly modeled as a Markov process.

Jukes-Cantor Correction for Multiple Mutations

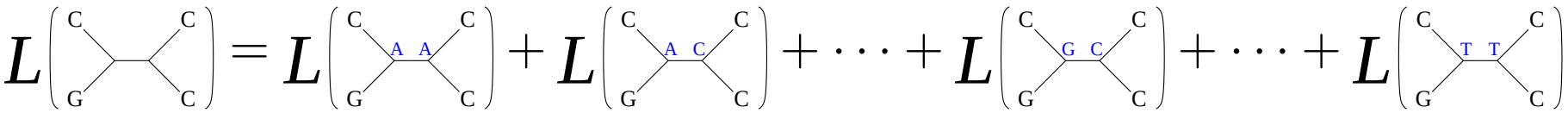




The substitution process is commonly modeled as a Markov process.

Computing Likelihood Values for Trees

Given a tree with branch lengths and sequences for all nodes, the computation of likelihood values for trees is straight forward.
Unfortunately, we usually have no sequences for the inner nodes (ancestral sequences).
Hence we have to evaluate every possible labeling at the inner nodes:



for every column in the alignment. . . but there is a faster algorithm.
(Peeling Algorithm by Felsenstein, 1981)

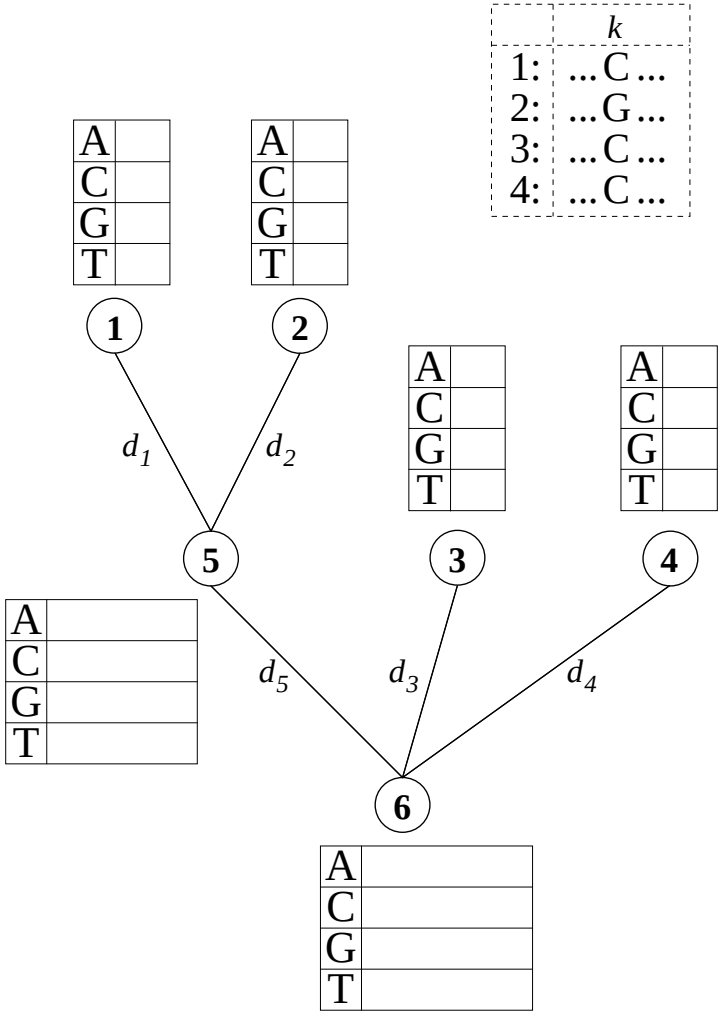
Likelihoods of Trees (Single alignment column, given tree)

For a single alignment column

	k
1:	...C...
2:	...G...
3:	...C...
4:	...C...

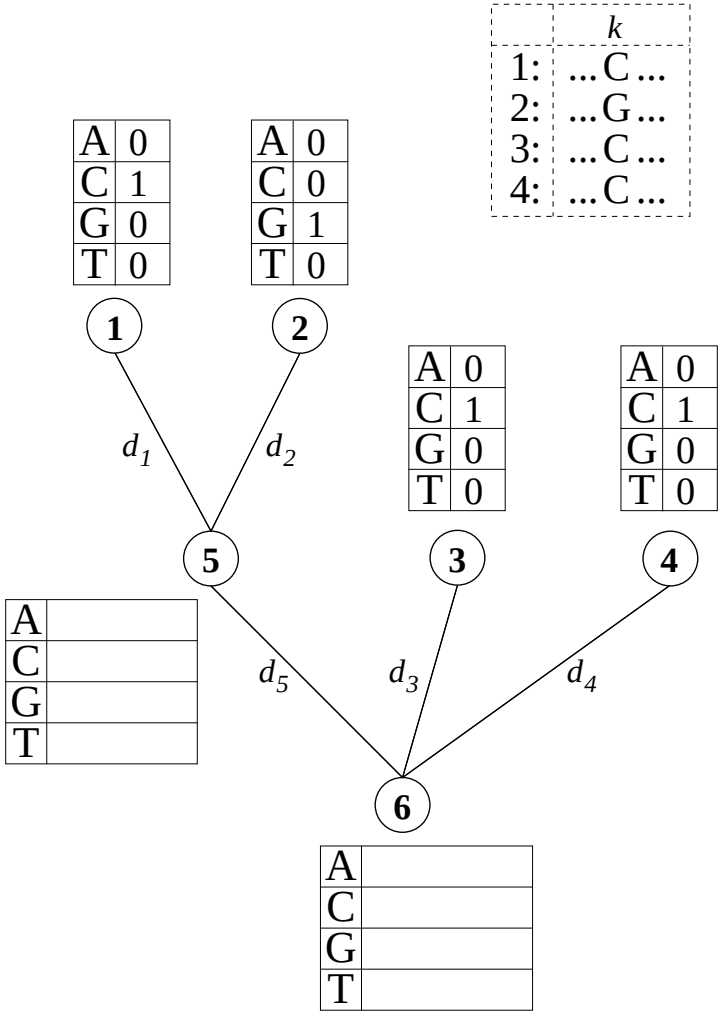
Likelihoods of Trees (Single alignment column, given tree)

For a single alignment column and a given tree:



Likelihoods of Trees (Single alignment column, given tree)

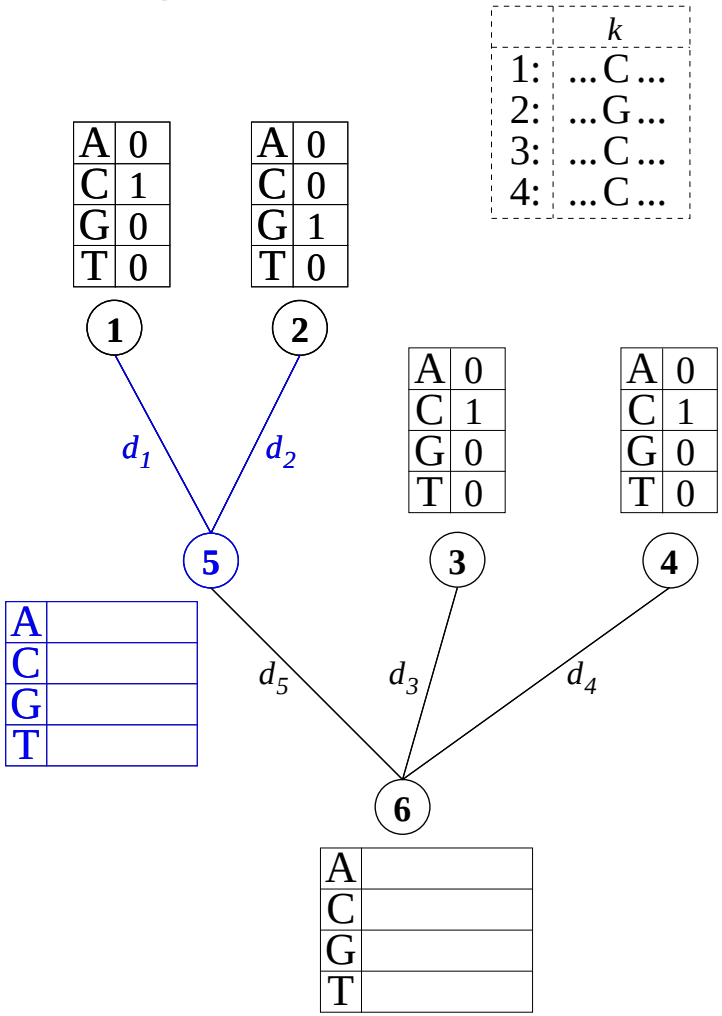
For a single alignment column and a given tree:



with all $d_x = 0.1$ and $P_{ij}(0.1) = \begin{cases} .9068 & i = j \\ .0313 & i \neq j \end{cases}$ (JC)

Likelihoods of Trees (Single alignment column, given tree)

For a single alignment column and a given tree:



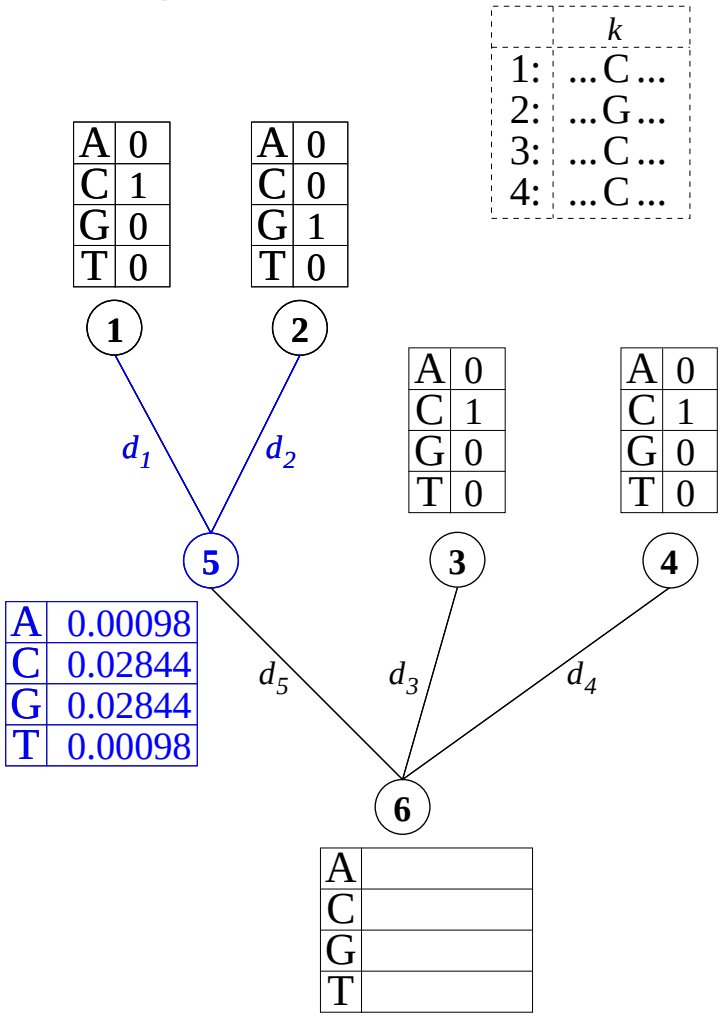
Likelihoods of nucleotides i at inner nodes:

$$L_5(i) = [P_{iC}(d_1) \cdot L_1(C)] \cdot [P_{iG}(d_2) \cdot L_2(G)]$$

with all $d_x = 0.1$ and $P_{ij}(0.1) = \begin{cases} .9068 & i = j \\ .0313 & i \neq j \end{cases}$ (JC)

Likelihoods of Trees (Single alignment column, given tree)

For a single alignment column and a given tree:



Likelihoods of nucleotides i at inner nodes:

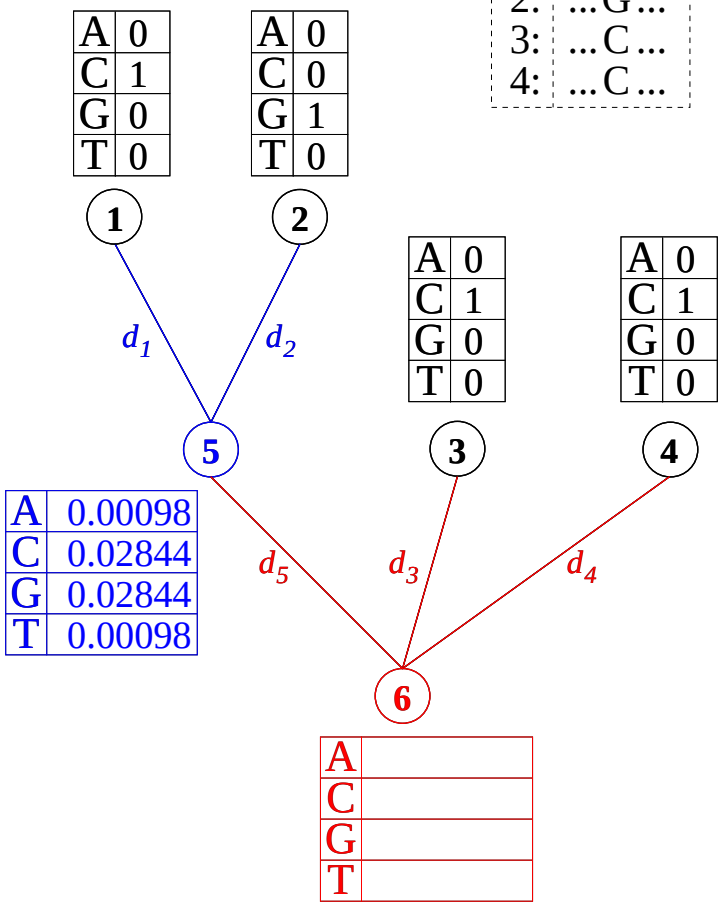
$$L_5(i) = [P_{iC}(d_1) \cdot L_1(C)] \cdot [P_{iG}(d_2) \cdot L_2(G)]$$

with all $d_x = 0.1$ and $P_{ij}(0.1) = \begin{cases} .9068 & i = j \\ .0313 & i \neq j \end{cases}$ (JC)

Likelihoods of Trees (Single alignment column, given tree)

For a single alignment column and a given tree:

	k
1:	...C...
2:	...G...
3:	...C...
4:	...C...



Likelihoods of nucleotides i at inner nodes:

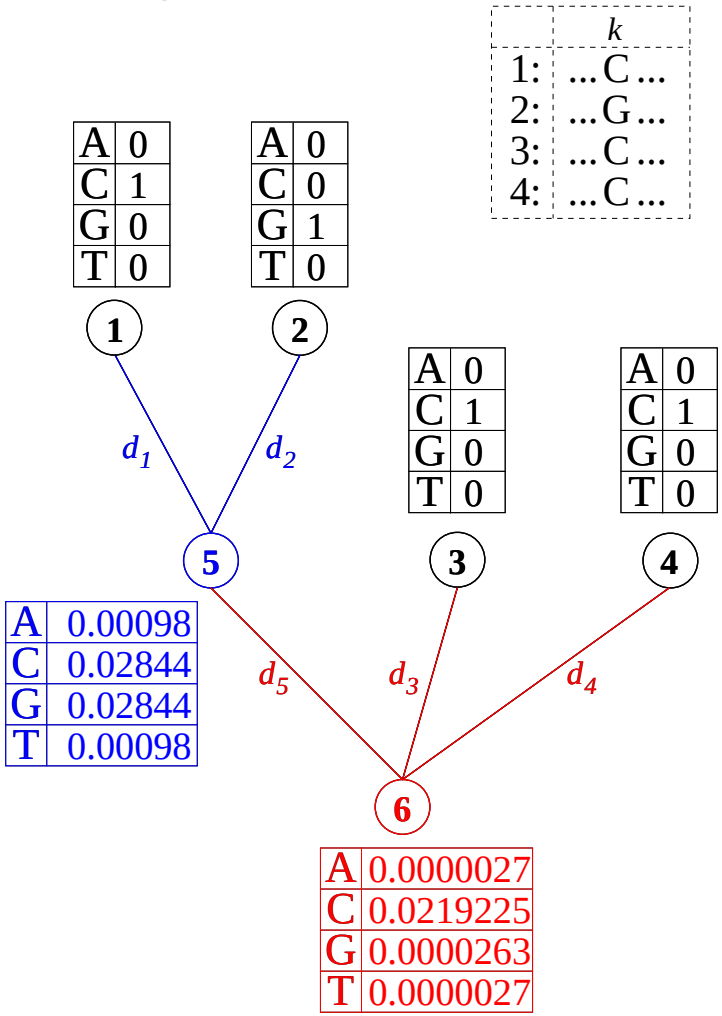
$$L_5(i) = [P_{iC}(d_1) \cdot L_1(C)] \cdot [P_{iG}(d_2) \cdot L_2(G)]$$

$$L_6(i) = \prod_{v=\{3,4,5\}} \left[\sum_{j=\{ACGT\}} P_{ij}(d_v) \cdot L_v(j) \right]$$

with all $d_x = 0.1$ and $P_{ij}(0.1) = \begin{cases} .9068 & i = j \\ .0313 & i \neq j \end{cases}$ (JC)

Likelihoods of Trees (Single alignment column, given tree)

For a single alignment column and a given tree:



Likelihoods of nucleotides i at inner nodes:

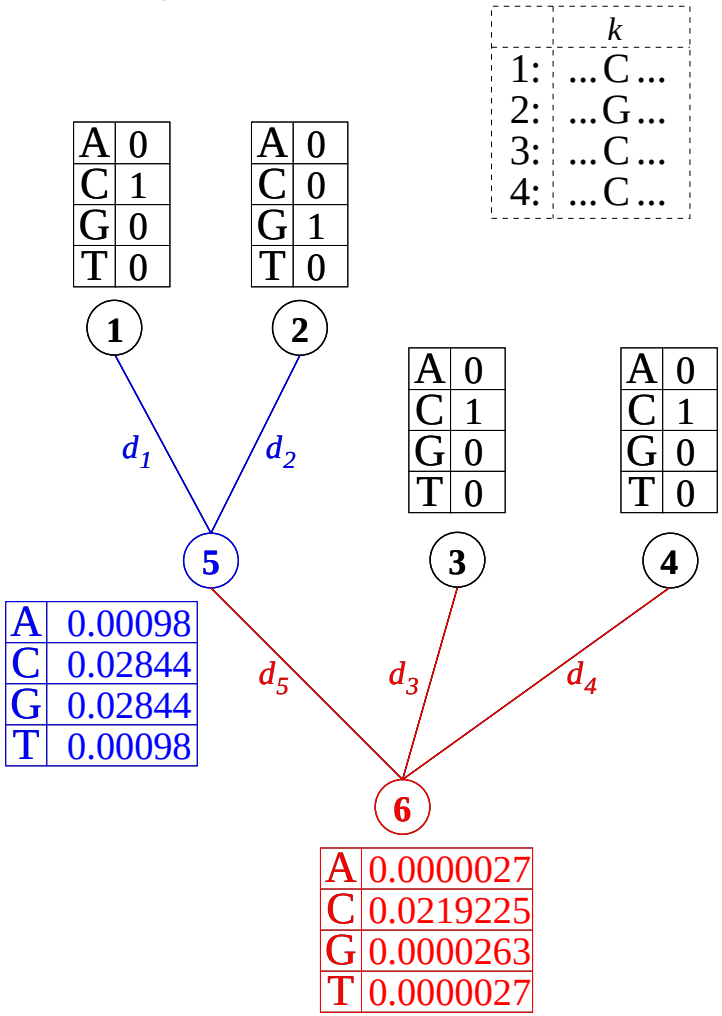
$$L_5(i) = [P_{iC}(d_1) \cdot L_1(C)] \cdot [P_{iG}(d_2) \cdot L_2(G)]$$

$$L_6(i) = \prod_{v=\{3,4,5\}} \left[\sum_{j=\{ACGT\}} P_{ij}(d_v) \cdot L_v(j) \right]$$

with all $d_x = 0.1$ and $P_{ij}(0.1) = \begin{cases} .9068 & i = j \\ .0313 & i \neq j \end{cases}$ (JC)

Likelihoods of Trees (Single alignment column, given tree)

For a single alignment column and a given tree:



Likelihoods of nucleotides i at inner nodes:

$$L_5(i) = [P_{iC}(d_1) \cdot L_1(C)] \cdot [P_{iG}(d_2) \cdot L_2(G)]$$

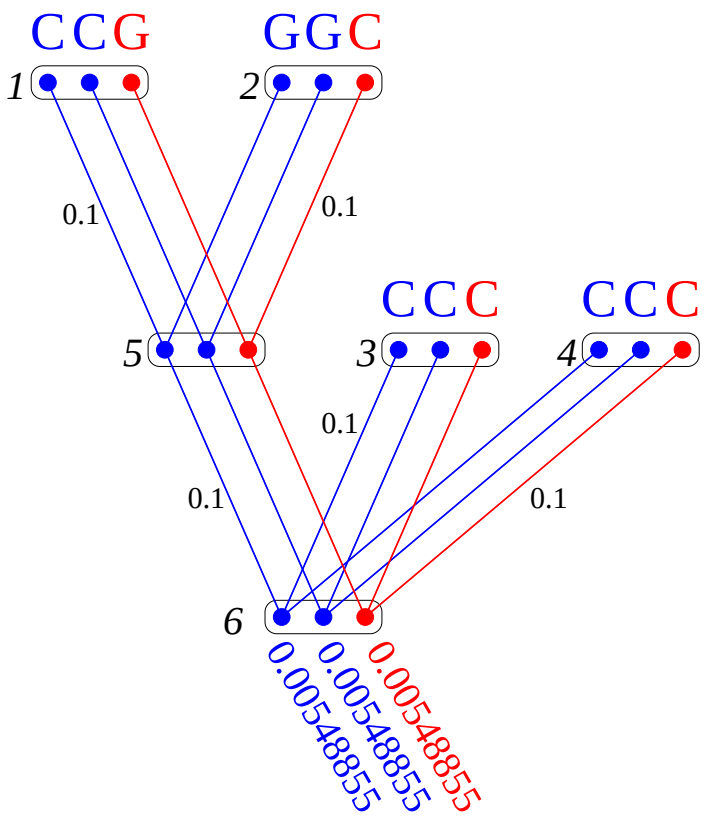
$$L_6(i) = \prod_{v=\{3,4,5\}} \left[\sum_{j=\{ACGT\}} P_{ij}(d_v) \cdot L_v(j) \right]$$

Site-Likelihood of an alignment column k:

$$L^{(k)} = \sum_{i=\{ACGT\}} \pi_i \cdot L_6(i) = 0.005489$$

with all $d_x = 0.1$ and $P_{ij}(0.1) = \begin{cases} .9068 & i = j \\ .0313 & i \neq j \end{cases}$ (JC)

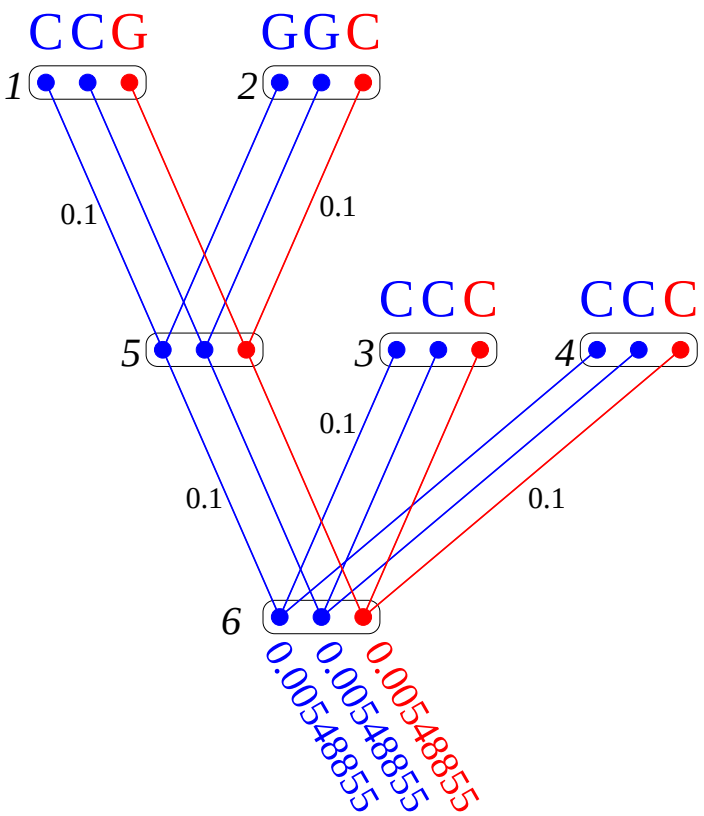
Likelihoods of Trees (multiple columns)



Considering this tree with $n = 4$ sequences of length $m = 3$ the tree likelihood of this tree is

$$\mathcal{L}(T) = \prod_{k=1}^{m} L^{(k)}$$

Likelihoods of Trees (multiple columns)

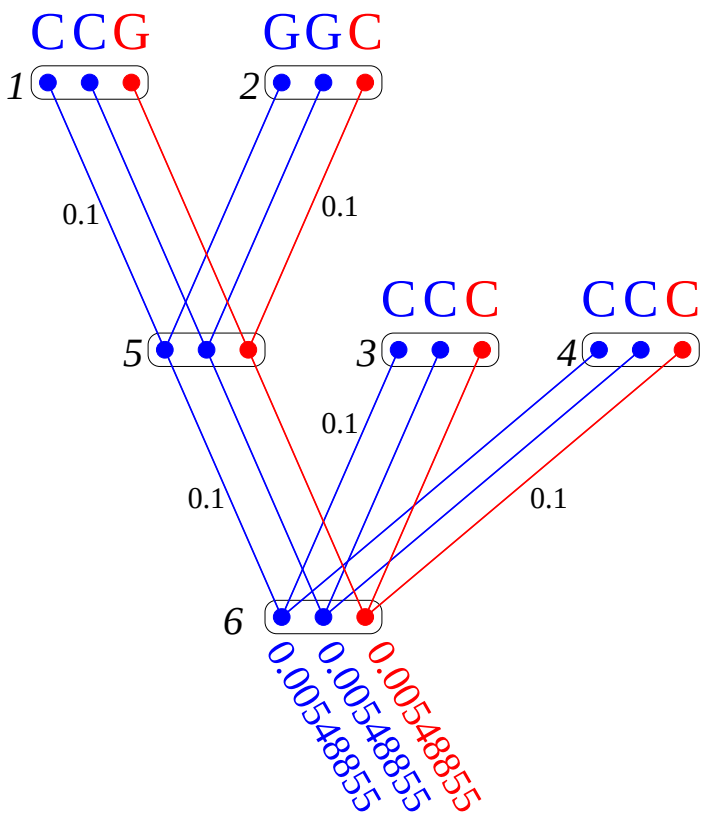


Considering this tree with $n = 4$ sequences of length $m = 3$ the tree likelihood of this tree is

$$\mathcal{L}(T) = \prod_{k=1}^{m} L^{(k)} = 0.005489^2 \cdot 0.005489$$

$$= 0.0000001653381$$

Likelihoods of Trees (multiple columns)



Considering this tree with $n = 4$ sequences of length $m = 3$ the tree likelihood of this tree is

$$\mathcal{L}(T) = \prod_{k=1}^{m} L^{(k)} = 0.005489^2 \cdot 0.005489$$

$$= 0.000001653381$$

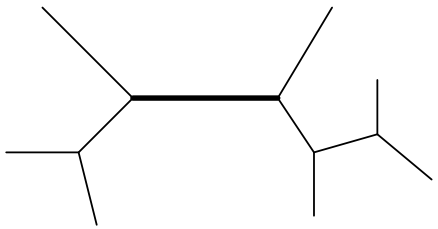
or the log-likelihood

$$\ln \mathcal{L}(T) = \sum_{k=1}^{m} \ln L^{(k)} = -15.61527$$

Adjusting Branch Lengths Step-By-Step

To compute optimal branch lengths do the following. Initialize the branch lengths.
(1.) Choose a branch. (2.) Move the virtual root to an adjacent node.
(3.) Compute all partial likelihoods recursively. (4.) Adjust the branch length to maximize the likelihood value.

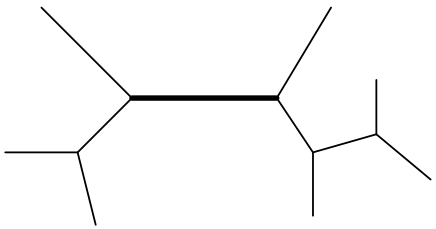
1.



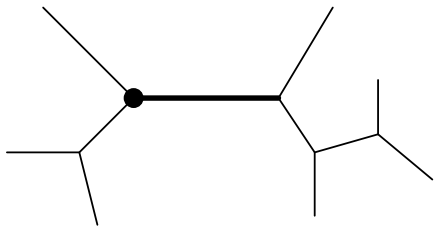
Adjusting Branch Lengths Step-By-Step

To compute optimal branch lengths do the following. Initialize the branch lengths.
(1.) Choose a branch. (2.) Move the virtual root to an adjacent node.
(3.) Compute all partial likelihoods recursively. (4.) Adjust the branch length to maximize the likelihood value.

1.



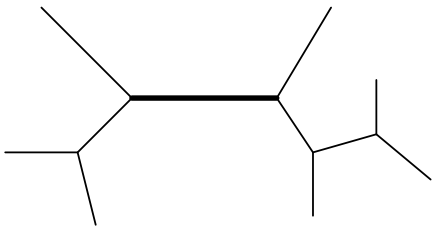
2.



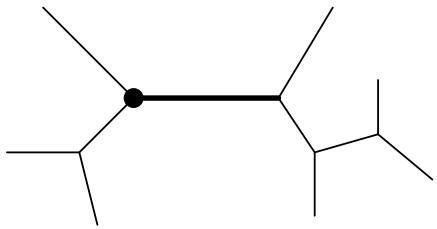
Adjusting Branch Lengths Step-By-Step

To compute optimal branch lengths do the following. Initialize the branch lengths.
(1.) Choose a branch. (2.) Move the virtual root to an adjacent node.
(3.) Compute all partial likelihoods recursively. (4.) Adjust the branch length to maximize the likelihood value.

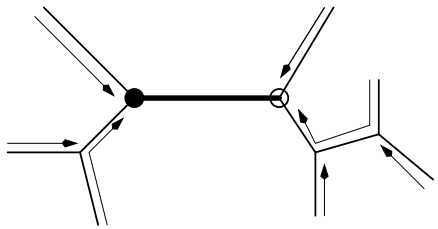
1.



2.



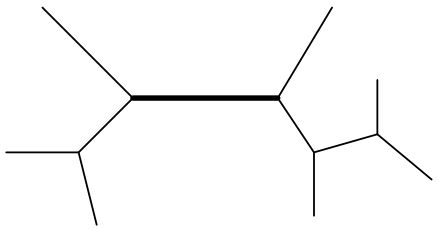
3.



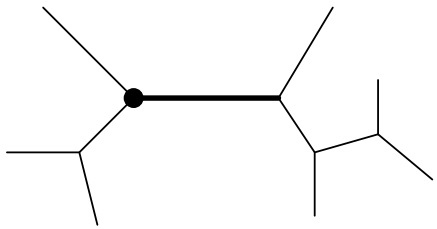
Adjusting Branch Lengths Step-By-Step

To compute optimal branch lengths do the following. Initialize the branch lengths.
(1.) Choose a branch. (2.) Move the virtual root to an adjacent node.
(3.) Compute all partial likelihoods recursively. (4.) Adjust the branch length to maximize the likelihood value.

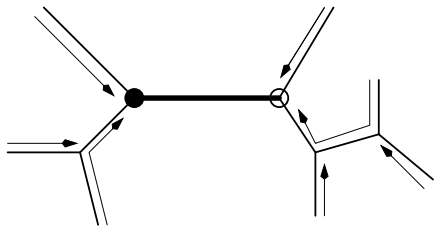
1.



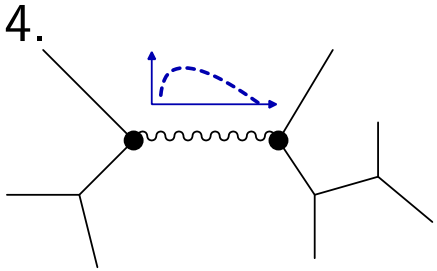
2.



3.

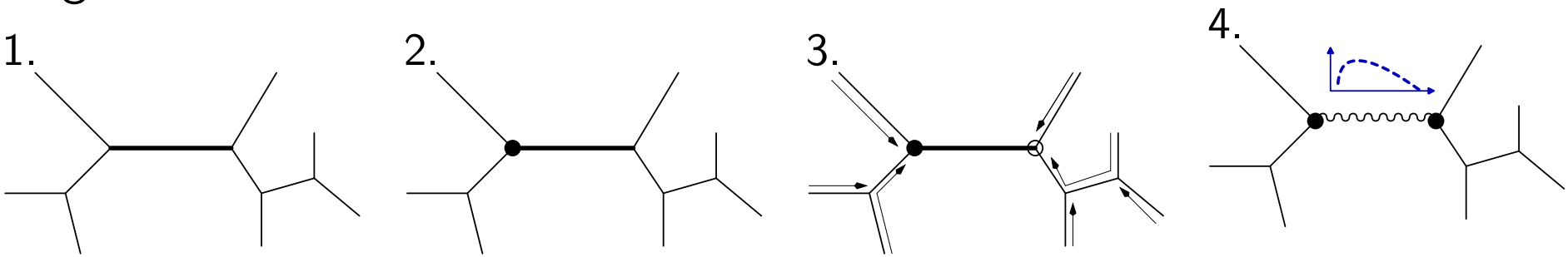


4.



Adjusting Branch Lengths Step-By-Step

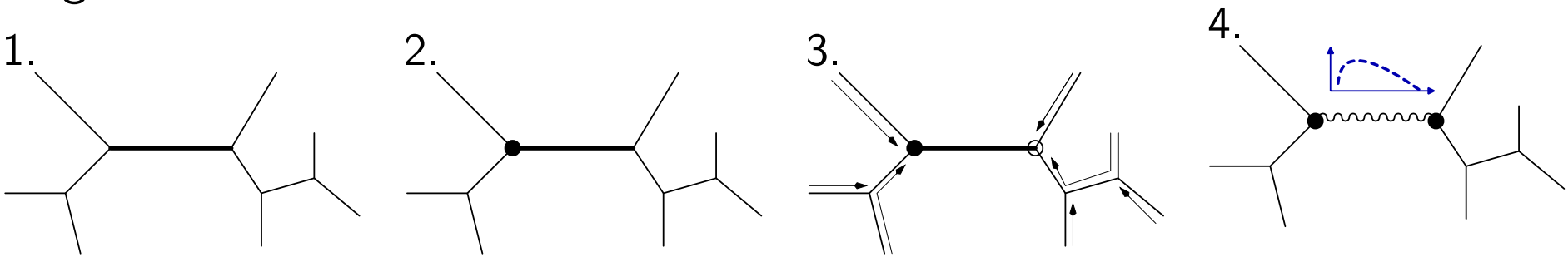
To compute optimal branch lengths do the following. Initialize the branch lengths.
(1.) Choose a branch. (2.) Move the virtual root to an adjacent node.
(3.) Compute all partial likelihoods recursively. (4.) Adjust the branch length to maximize the likelihood value.



Repeat this for every branch until no better likelihood is gained.

Adjusting Branch Lengths Step-By-Step

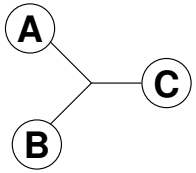
To compute optimal branch lengths do the following. Initialize the branch lengths.
(1.) Choose a branch. (2.) Move the virtual root to an adjacent node.
(3.) Compute all partial likelihoods recursively. (4.) Adjust the branch length to maximize the likelihood value.



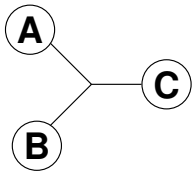
Repeat this for every branch until no better likelihood is gained.

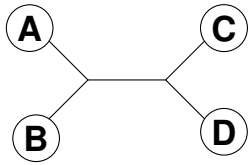
This is based on the Pulley-Principle (Felsenstein, 1981) which states that the root can be moved on the tree but the likelihood doesn't change.

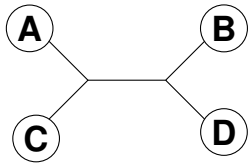
Number of Trees to Examine...

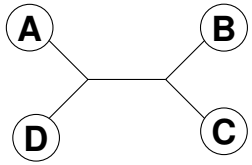


Number of Trees to Examine...

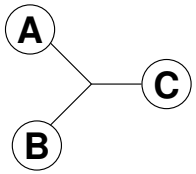


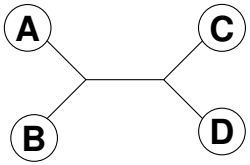


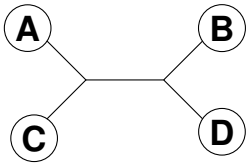


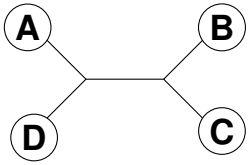


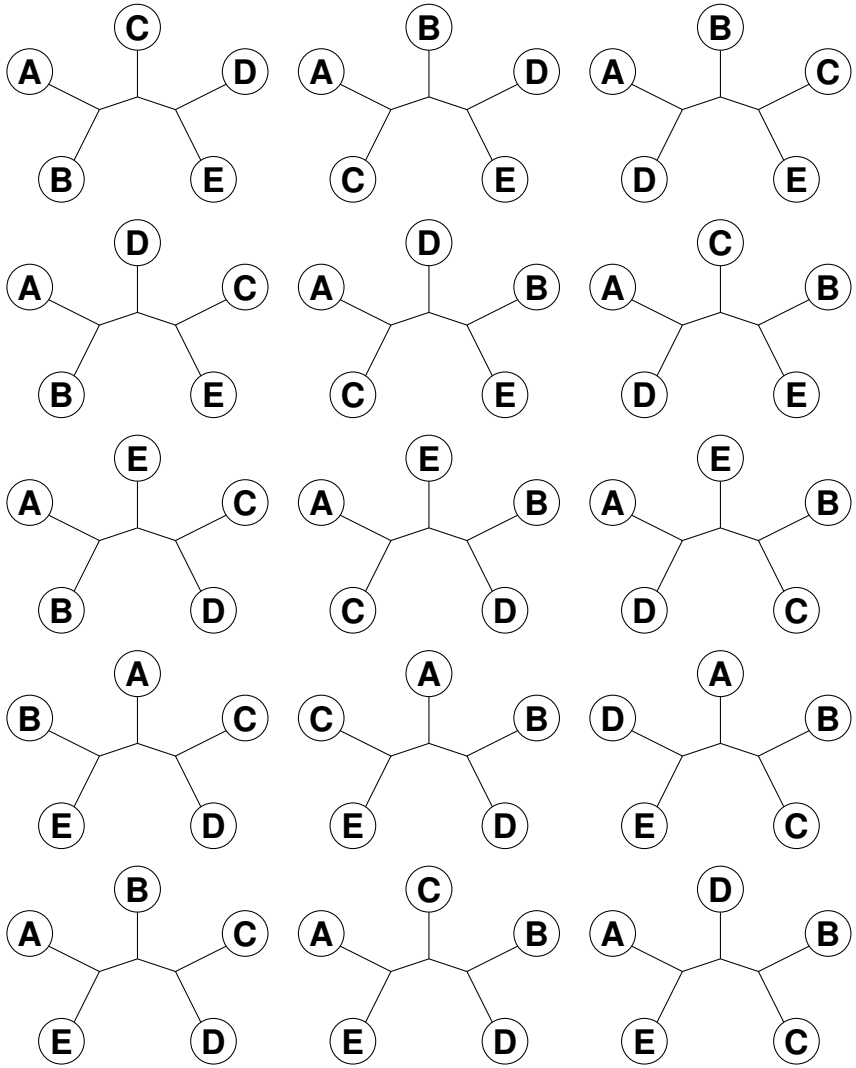
Number of Trees to Examine...



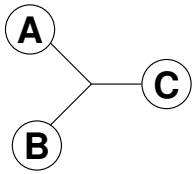


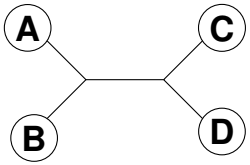


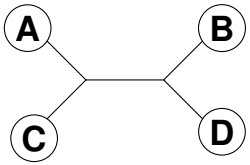


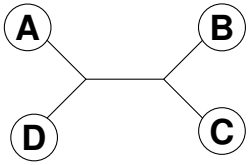


Number of Trees to Examine...

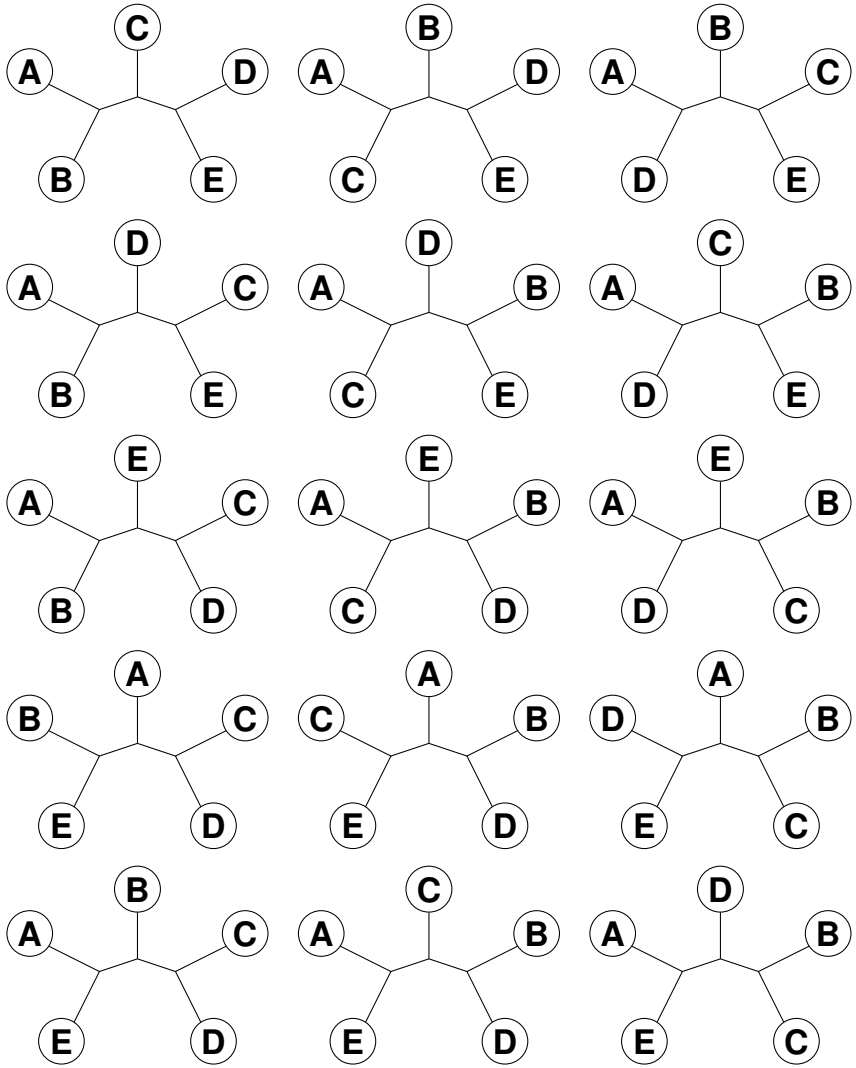




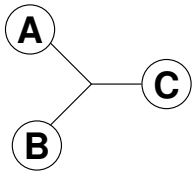


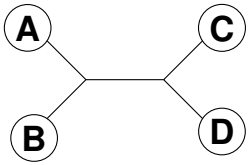


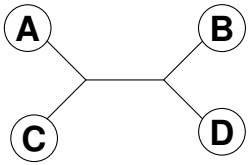
$$B(n) = \frac{(2n-5)!}{2^{n-3}(n-3)!}$$

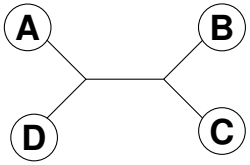


Number of Trees to Examine...







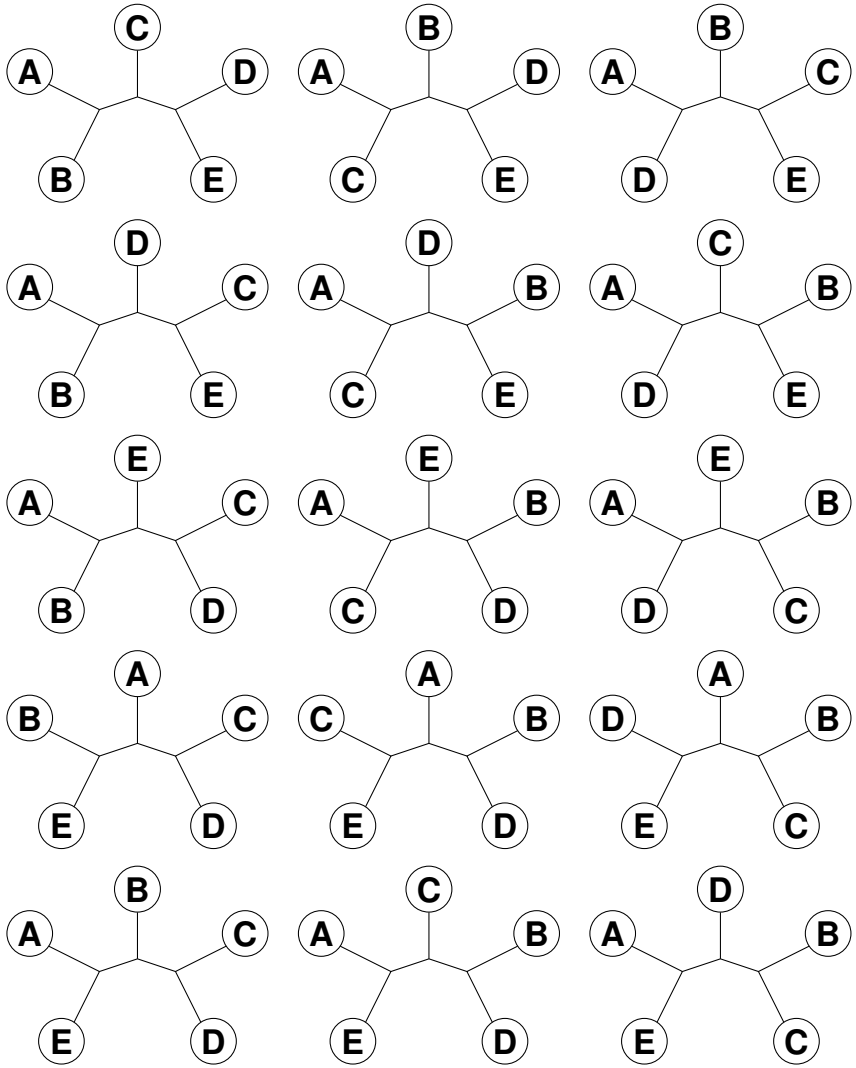


$$B(n) = \frac{(2n-5)!}{2^{n-3}(n-3)!}$$

$$B(10) = 2027025$$

$$B(55) = 2.98 \cdot 10^{84}$$

$$B(100) = 1.70 \cdot 10^{182}$$



Finding the ML Tree

Exhaustive Search: guarantees to find the optimal tree, because all trees are evaluated, but not feasible for more than 10-12 taxa.

Finding the ML Tree

Exhaustive Search: guarantees to find the optimal tree, because all trees are evaluated, but not feasible for more than 10-12 taxa.

Branch and Bound: guarantees to find the optimal tree, without searching certain parts of the tree space – can run on more sequences, but often not for current-day datasets.

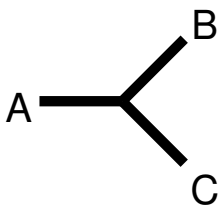
Finding the ML Tree

Exhaustive Search: guarantees to find the optimal tree, because all trees are evaluated, but not feasible for more than 10-12 taxa.

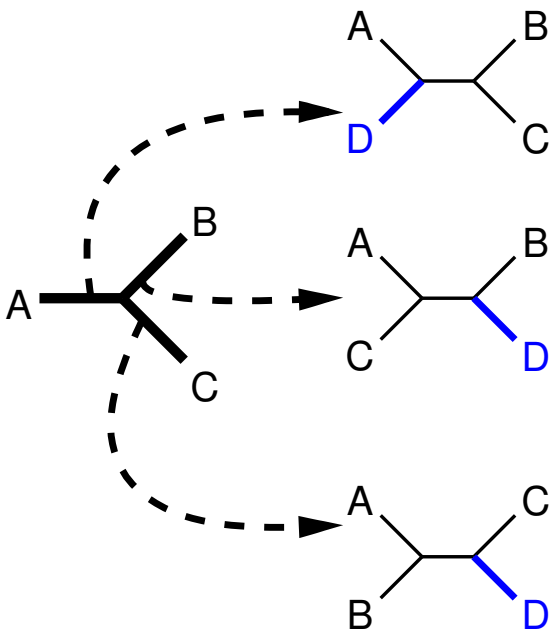
Branch and Bound: guarantees to find the optimal tree, without searching certain parts of the tree space – can run on more sequences, but often not for current-day datasets.

Heuristics: cannot guarantee to find the optimal tree, but are at least able to analyze large datasets.

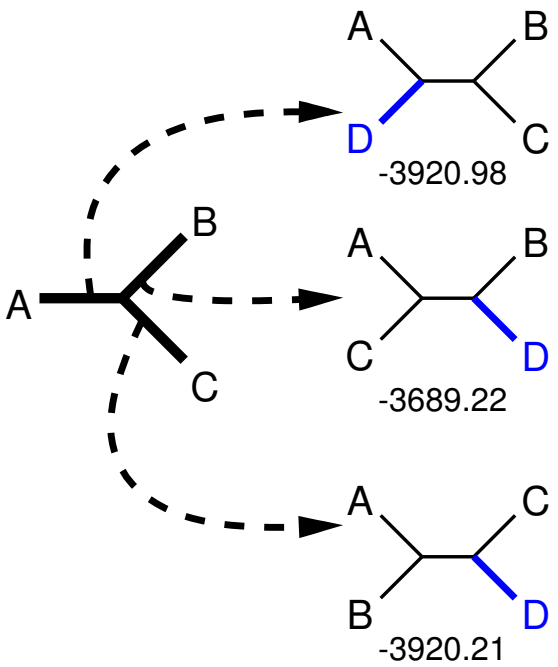
Build up a tree: Stepwise Insertion



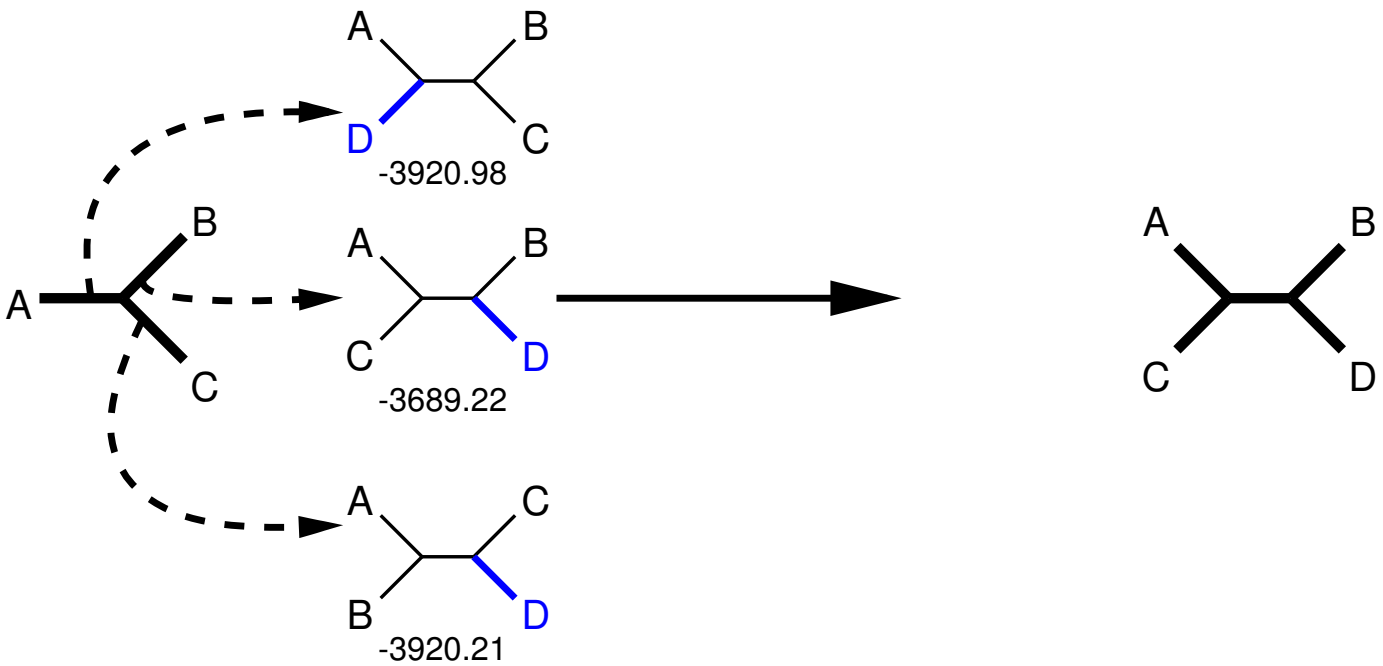
Build up a tree: Stepwise Insertion



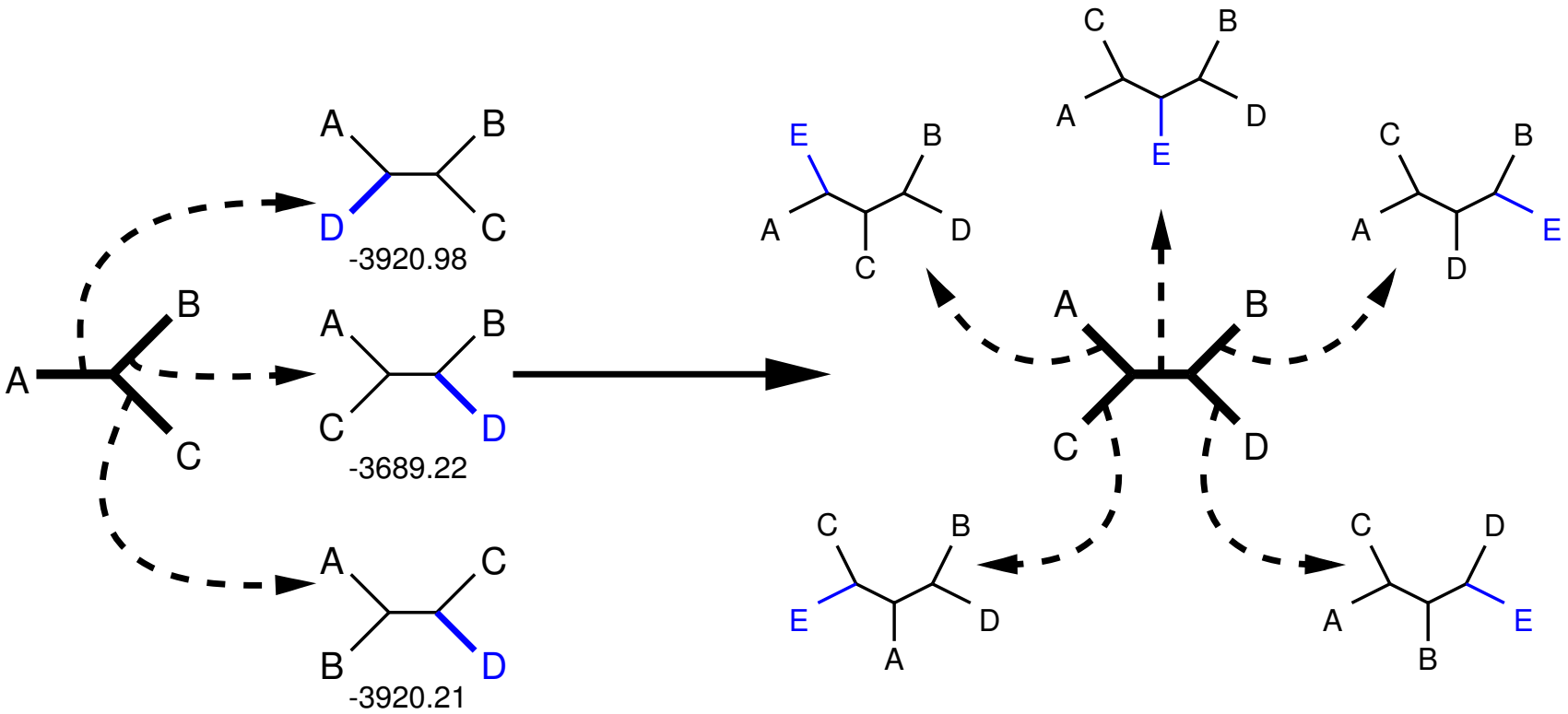
Build up a tree: Stepwise Insertion



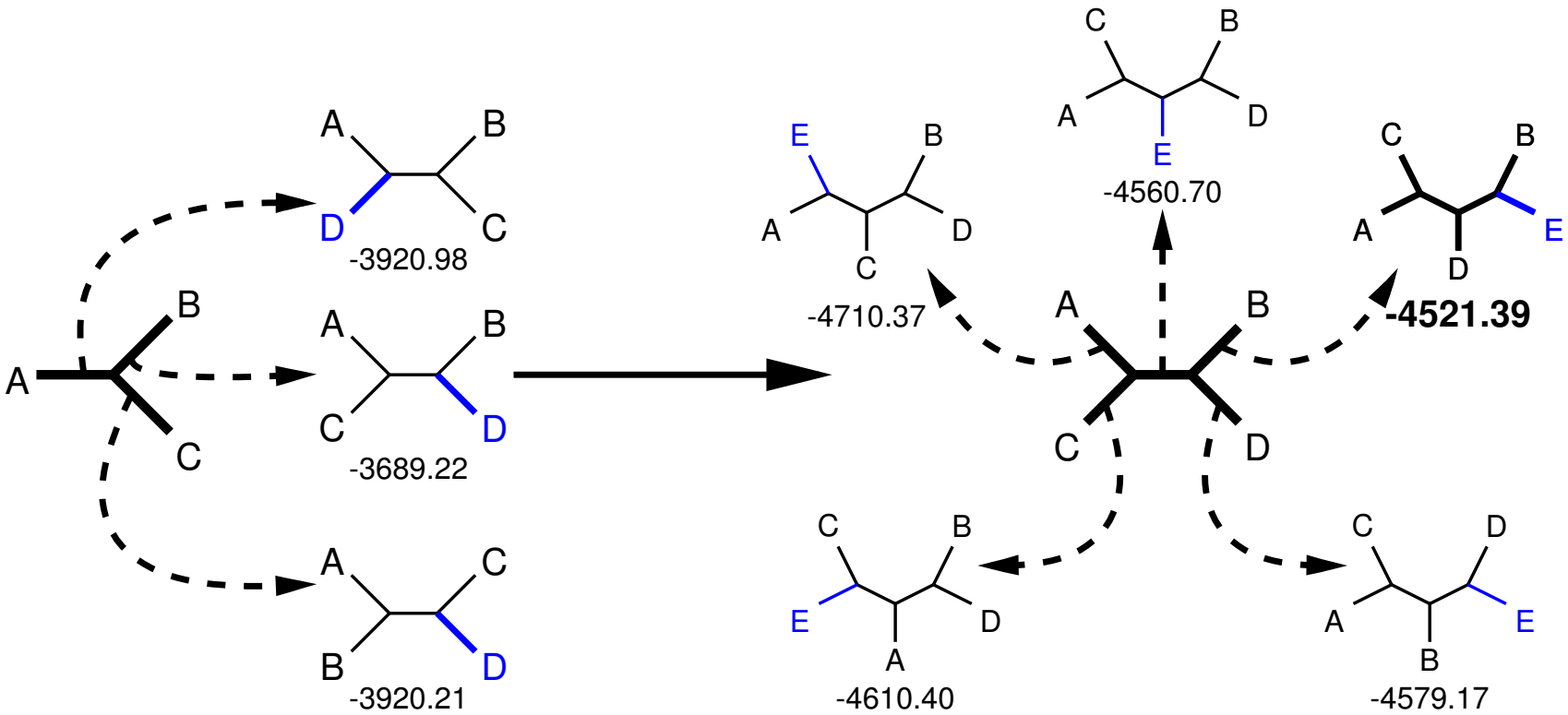
Build up a tree: Stepwise Insertion



Build up a tree: Stepwise Insertion



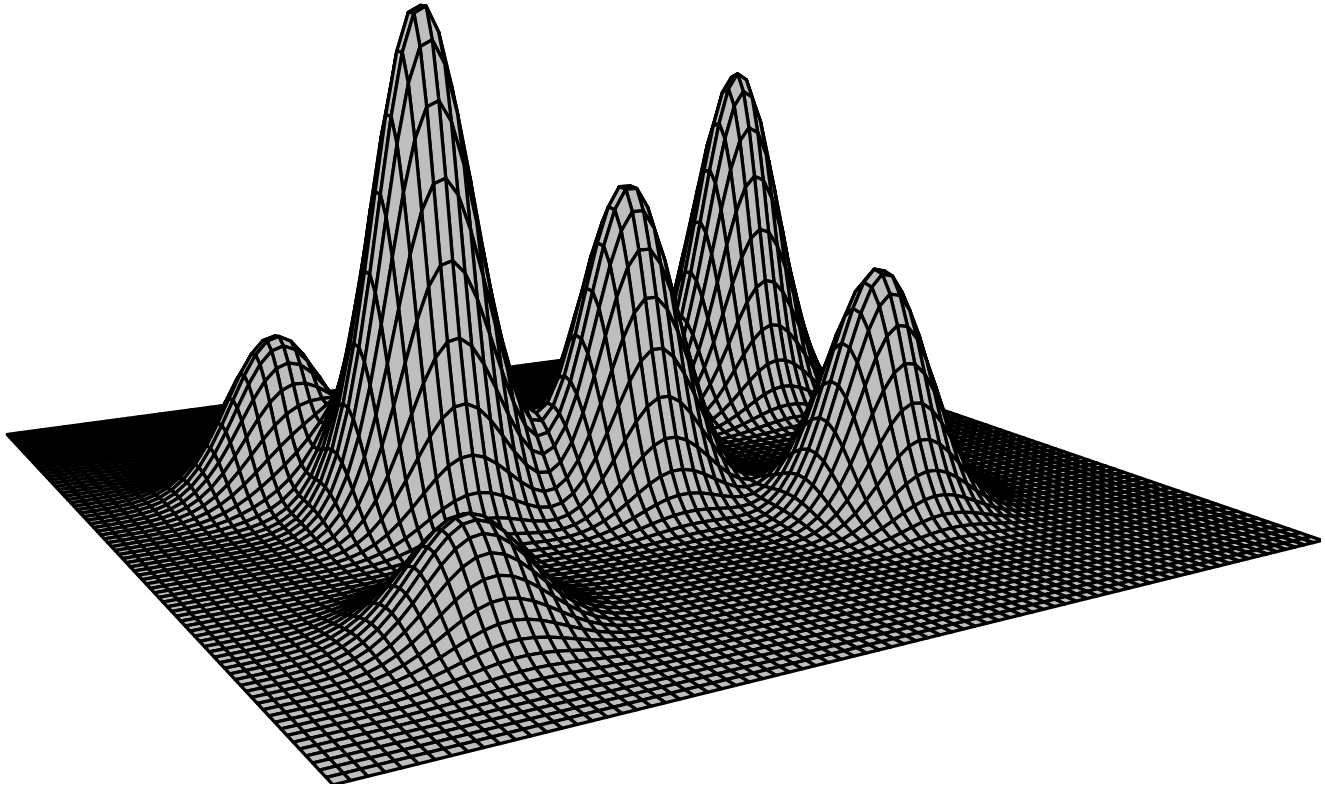
Build up a tree: Stepwise Insertion



Is also used for other (non-ML) methods like parsimony.

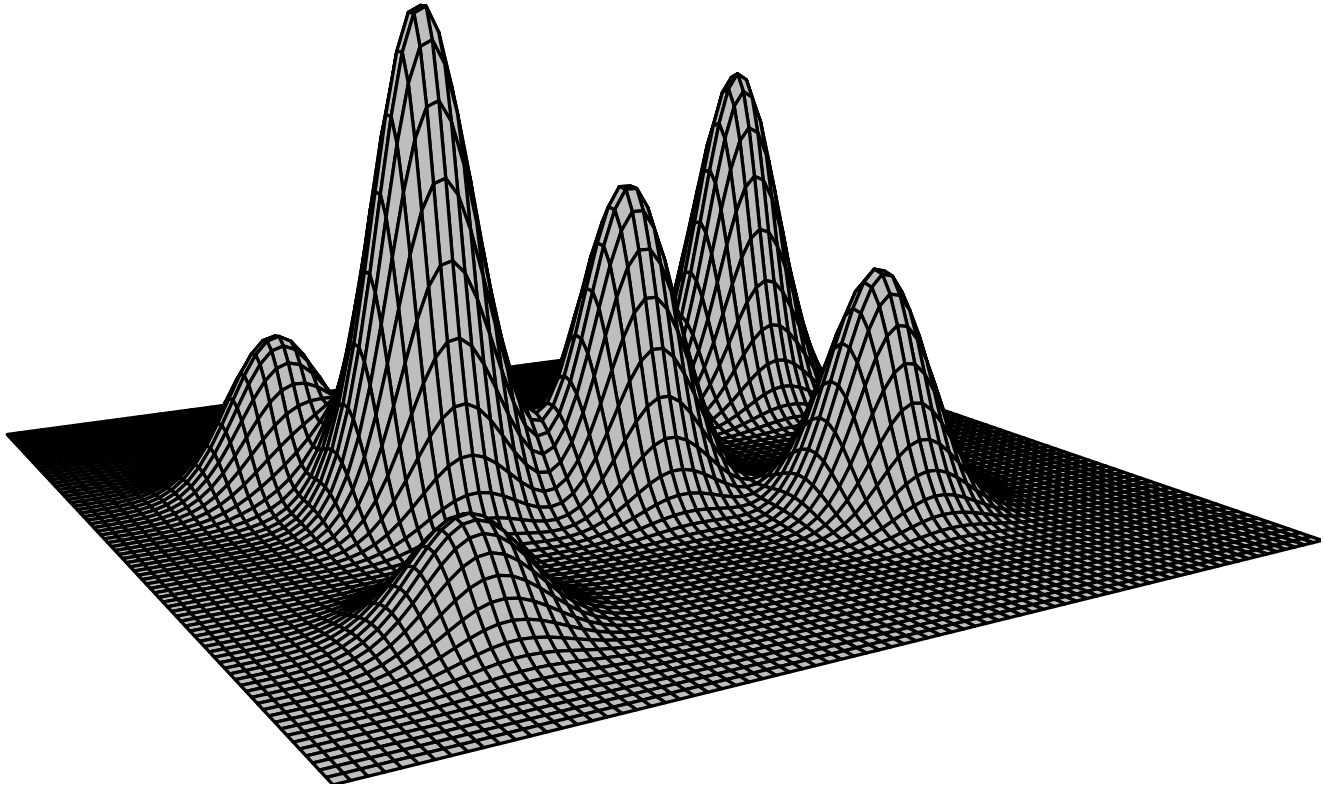
Local Maxima

What if we have multiple maxima in the likelihood surface?



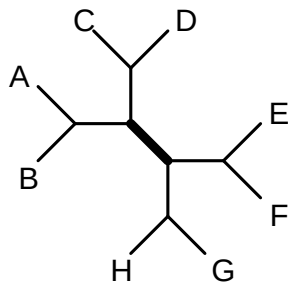
Local Maxima

What if we have multiple maxima in the likelihood surface?



Tree rearrangements to escape local maxima.

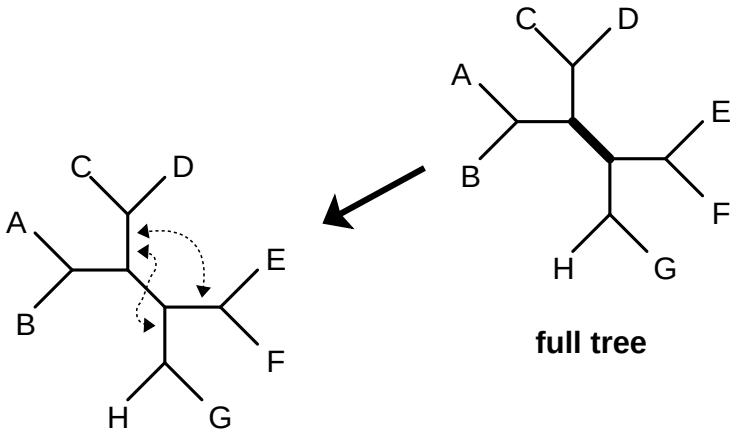
Tree Rearrangements: Scanning a Tree's Neighborhood



full tree

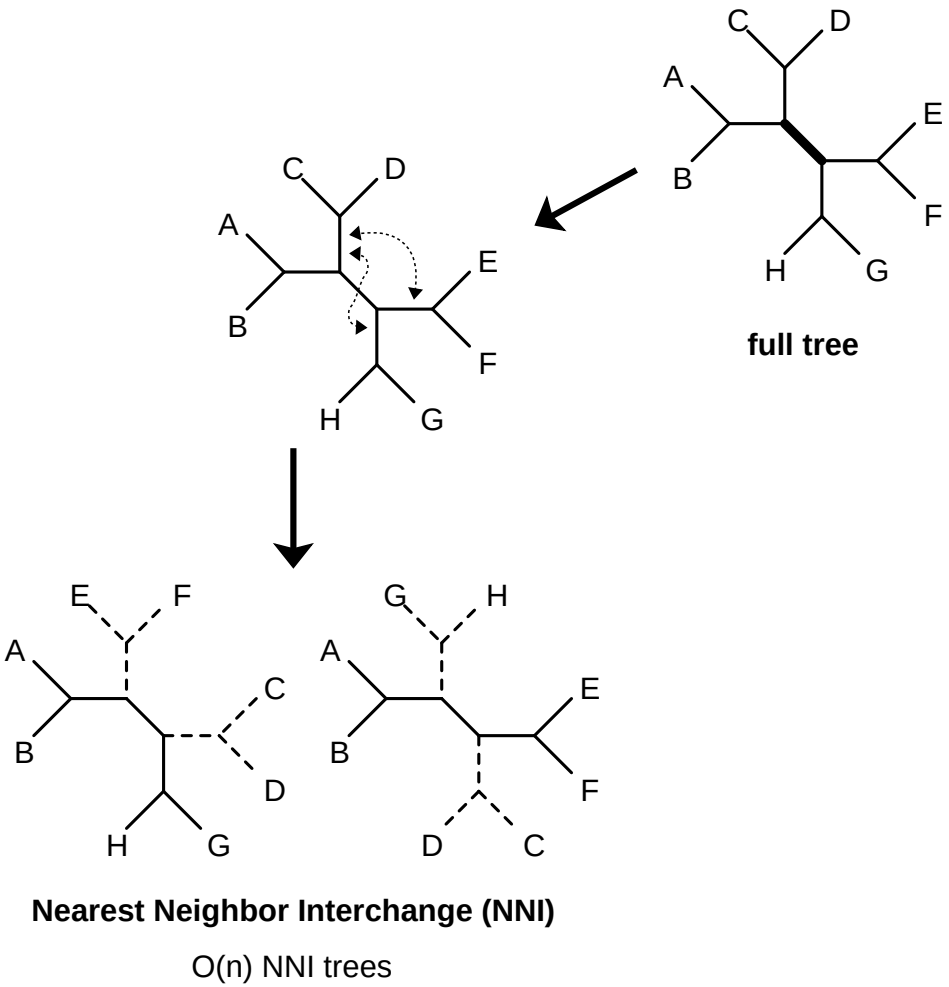
From a current tree construct other trees by rearranging its subtrees and evaluates all resulting trees. Repeat with the best tree found, until no better tree can be found. This also used for other (non-ML) methods, like parsimony.

Tree Rearrangements: Scanning a Tree's Neighborhood



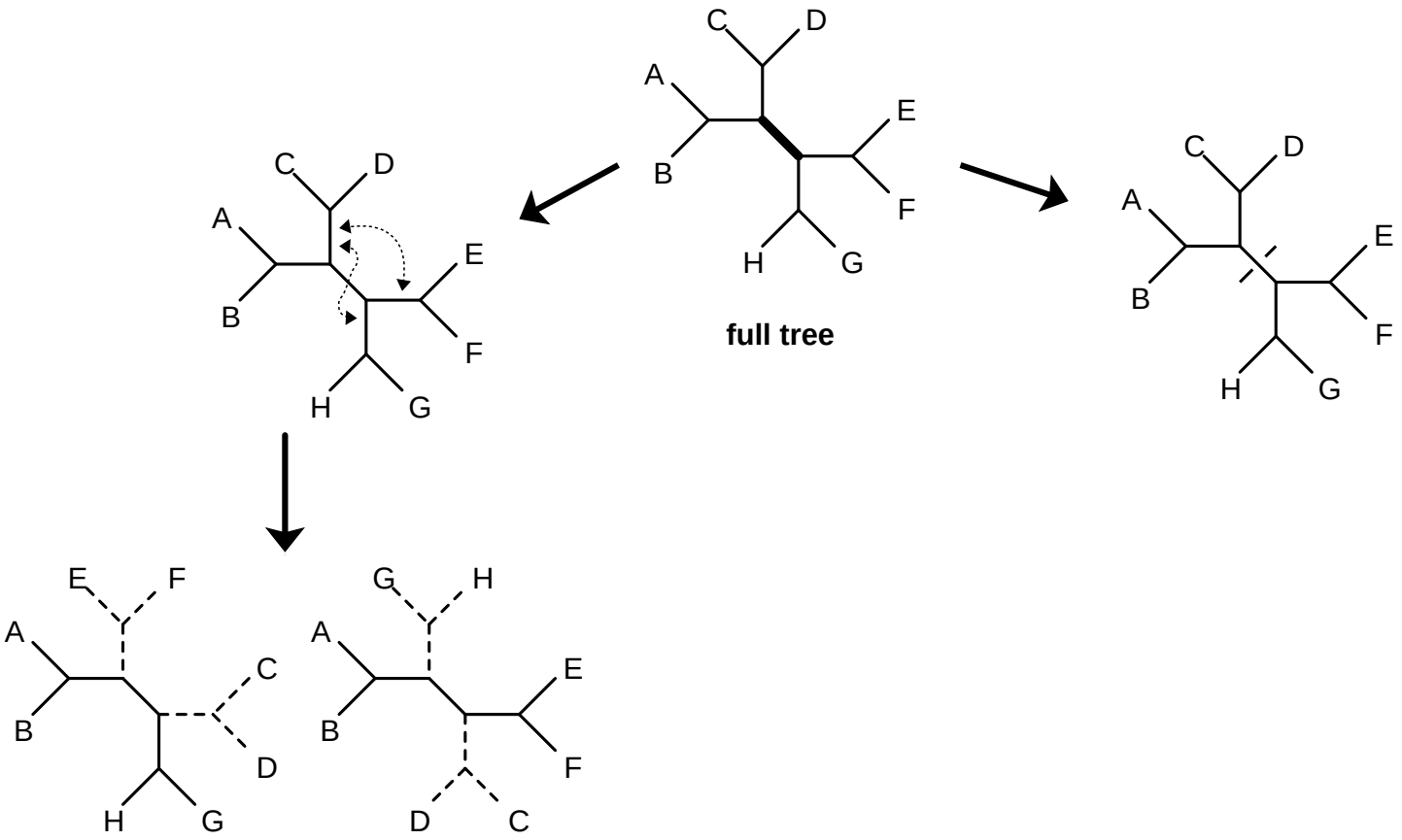
From a current tree construct other trees by rearranging its subtrees and evaluates all resulting trees. Repeat with the best tree found, until no better tree can be found. This also used for other (non-ML) methods, like parsimony.

Tree Rearrangements: Scanning a Tree's Neighborhood



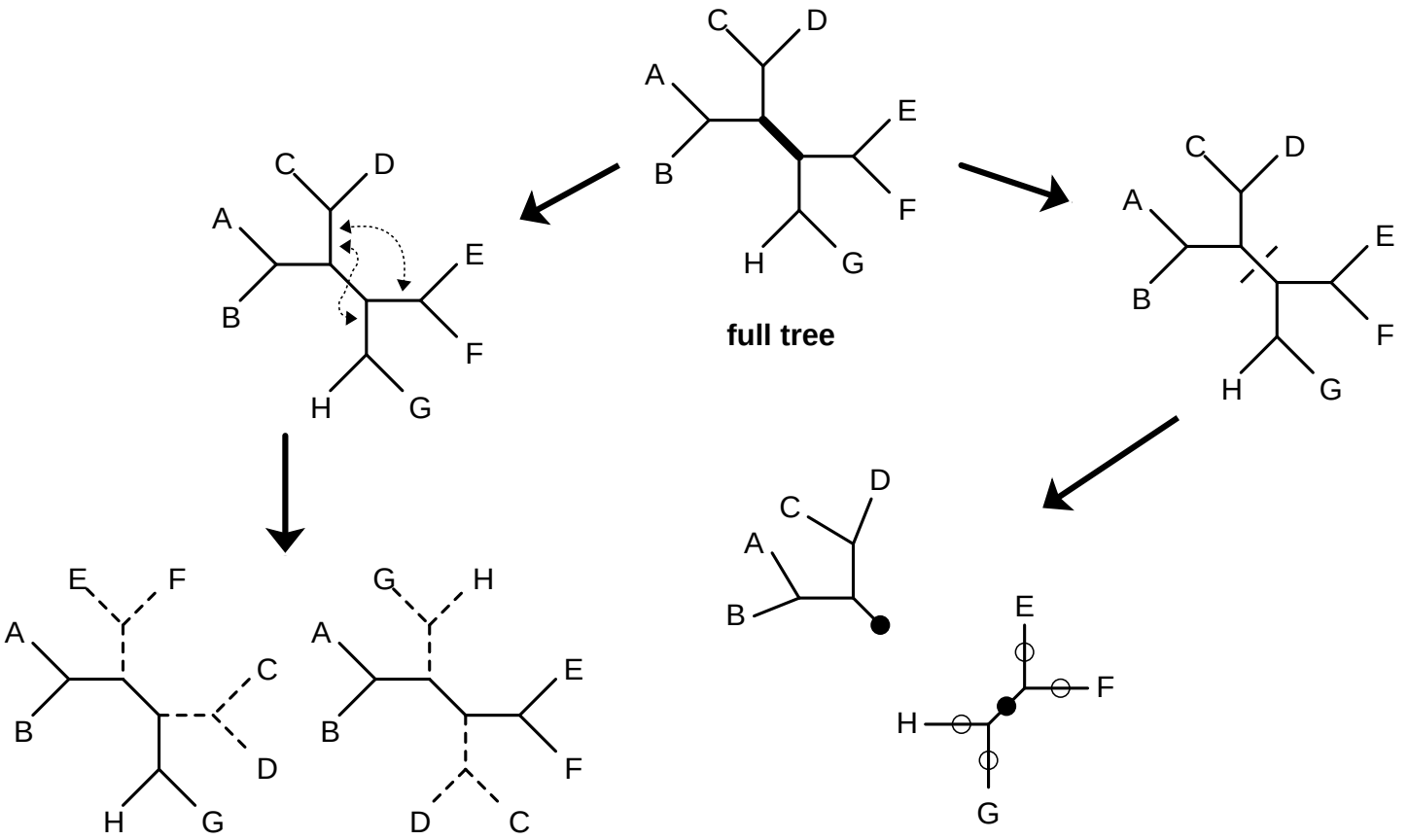
From a current tree construct other trees by rearranging its subtrees and evaluates all resulting trees. Repeat with the best tree found, until no better tree can be found. This also used for other (non-ML) methods, like parsimony.

Tree Rearrangements: Scanning a Tree's Neighborhood



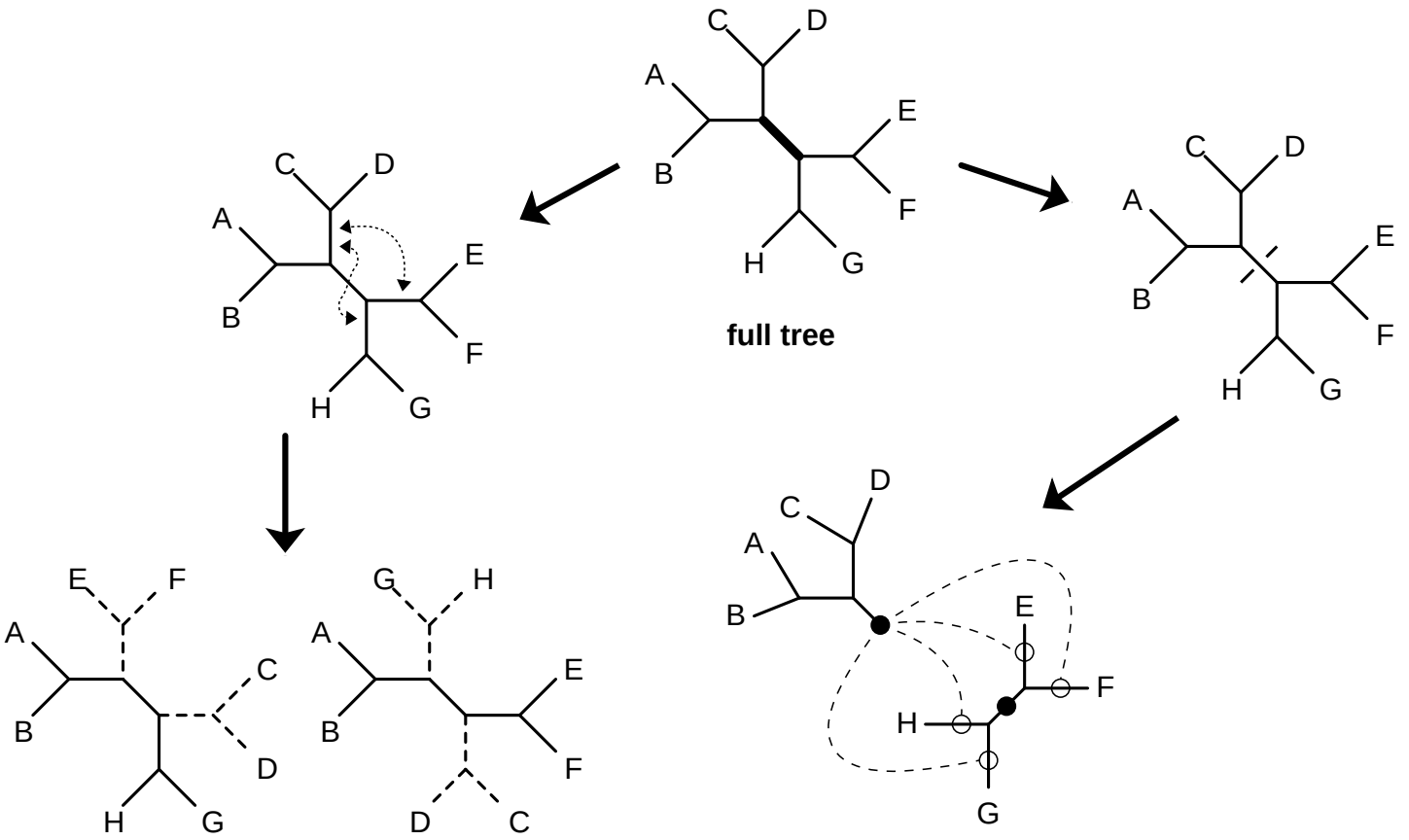
From a current tree construct other trees by rearranging its subtrees and evaluates all resulting trees. Repeat with the best tree found, until no better tree can be found. This also used for other (non-ML) methods, like parsimony.

Tree Rearrangements: Scanning a Tree's Neighborhood



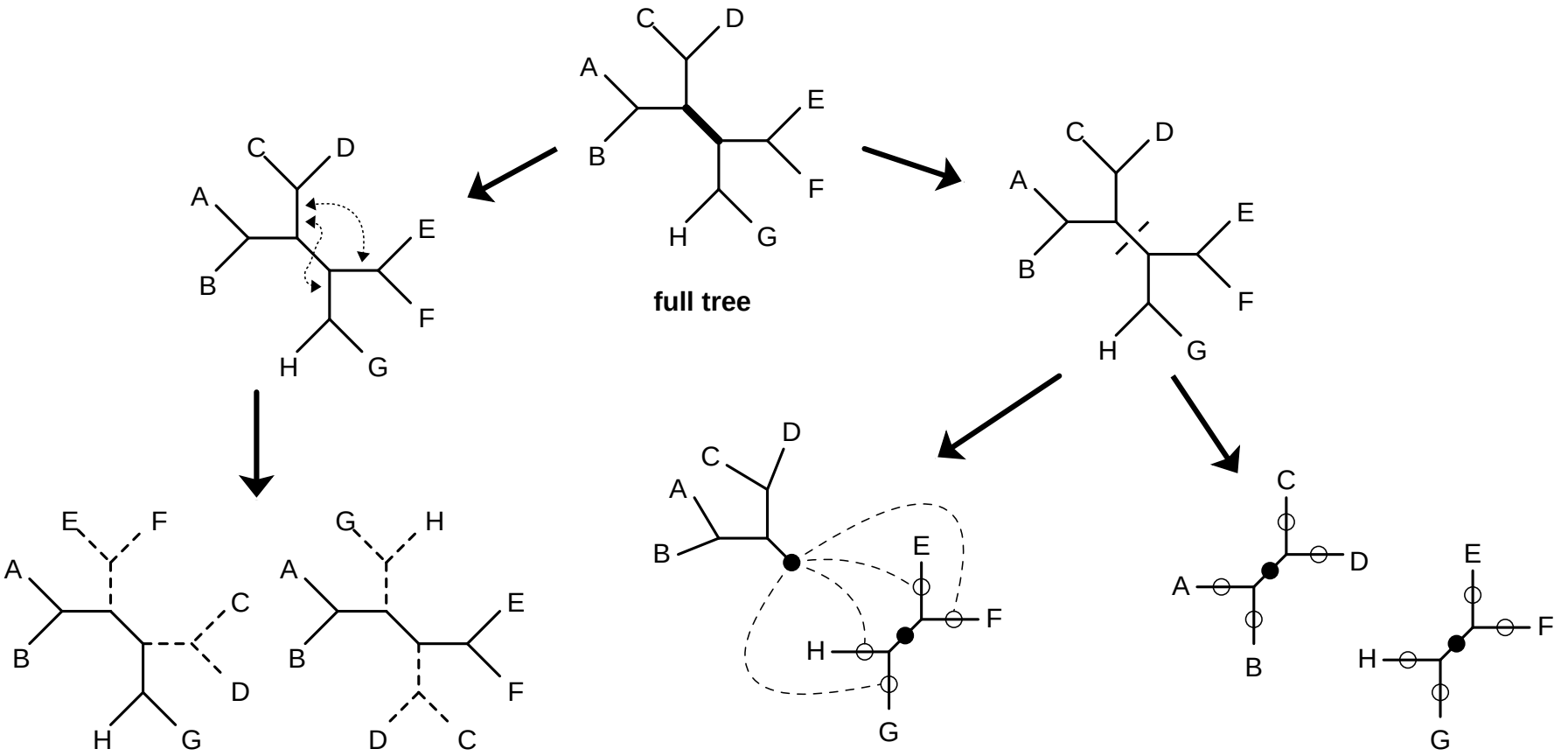
From a current tree construct other trees by rearranging its subtrees and evaluates all resulting trees. Repeat with the best tree found, until no better tree can be found. This also used for other (non-ML) methods, like parsimony.

Tree Rearrangements: Scanning a Tree's Neighborhood



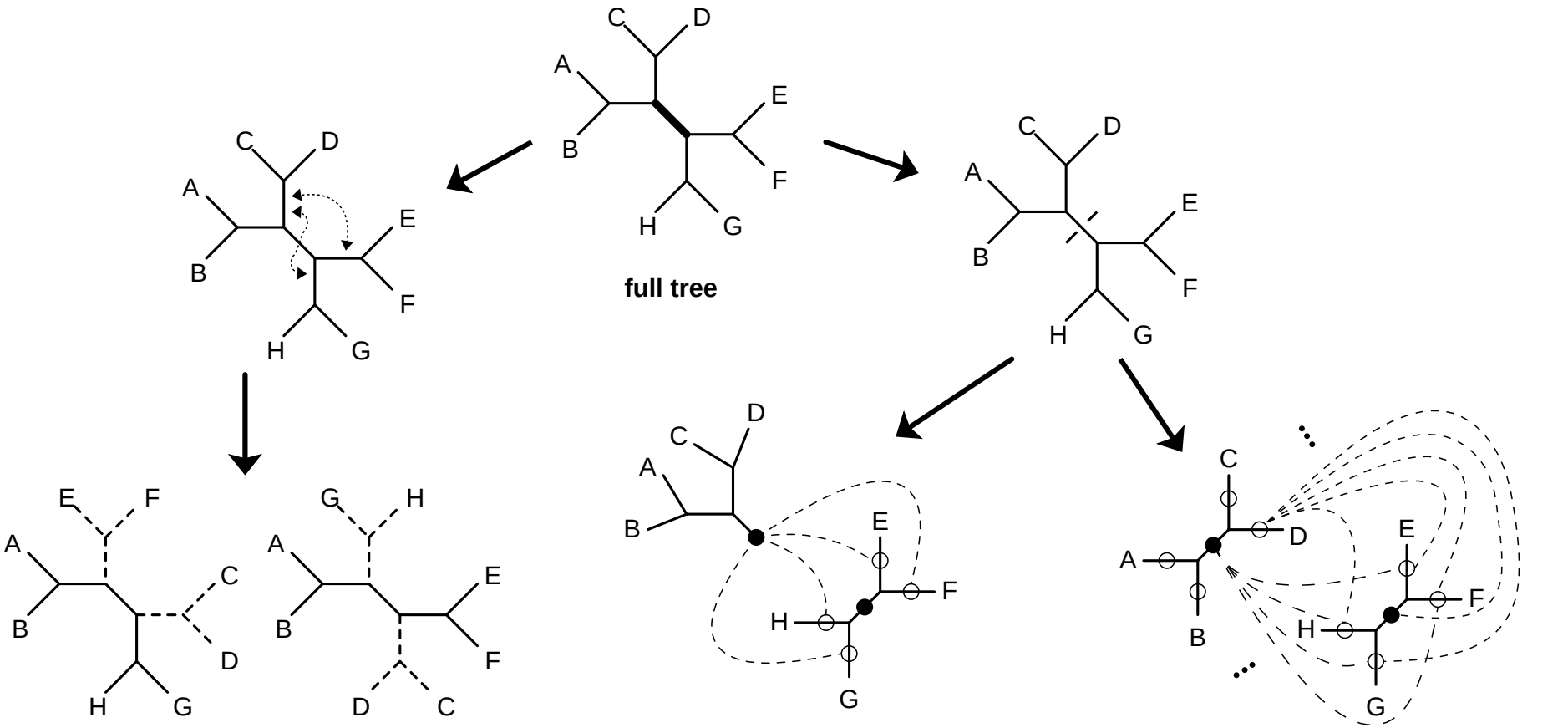
From a current tree construct other trees by rearranging its subtrees and evaluates all resulting trees. Repeat with the best tree found, until no better tree can be found. This also used for other (non-ML) methods, like parsimony.

Tree Rearrangements: Scanning a Tree's Neighborhood



From a current tree construct other trees by rearranging its subtrees and evaluates all resulting trees. Repeat with the best tree found, until no better tree can be found. This also used for other (non-ML) methods, like parsimony.

Tree Rearrangements: Scanning a Tree's Neighborhood



From a current tree construct other trees by rearranging its subtrees and evaluates all resulting trees. Repeat with the best tree found, until no better tree can be found. This also used for other (non-ML) methods, like parsimony.

How reliable is the reconstructed tree:

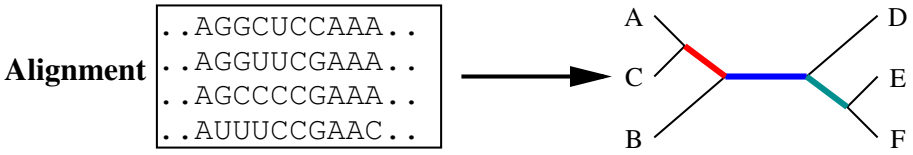
- Usually programs deliver a single (best) tree, but without confidence values for the subtrees.
- How can we assess reliability for the subtree?

Bootstrap and Consensus Tree

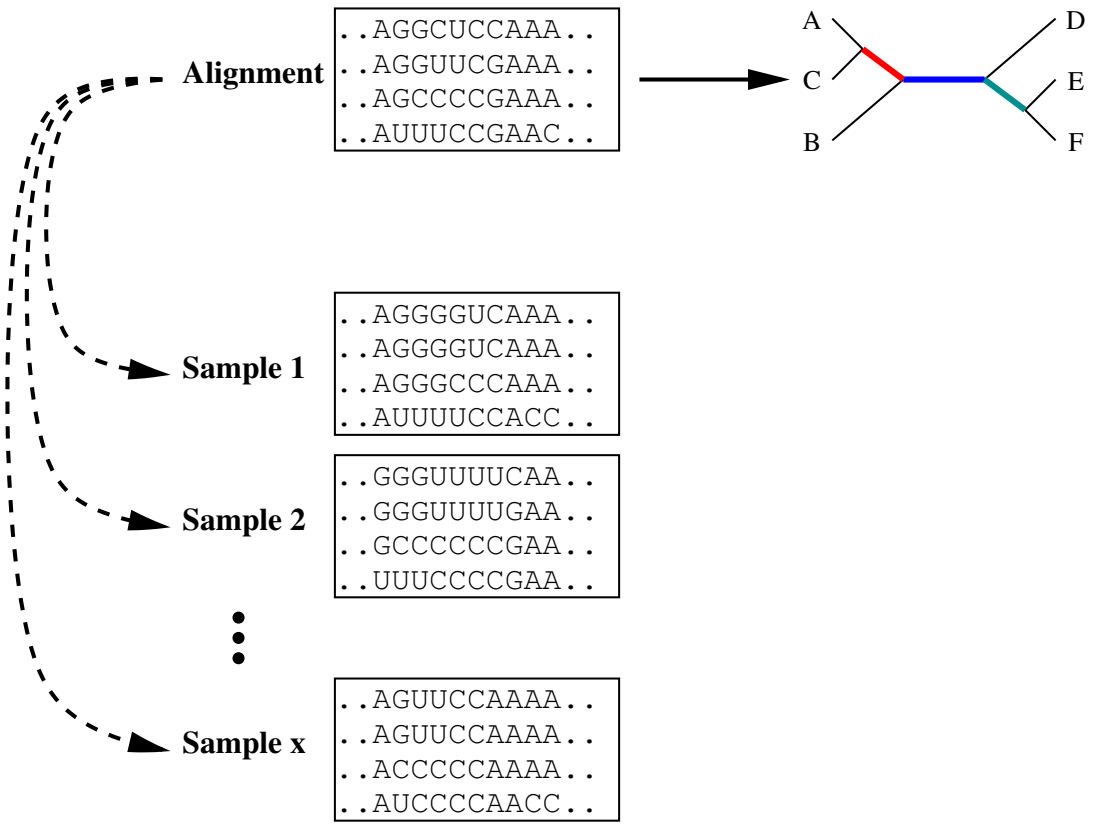
- Bootstrapping creates many pseudo-alignments by sampling alignment columns with replacement from the original alignment.
- From the pseudo-alignment we reconstruct trees.
- From the trees we collect and count all splits.
- From the splits we construct a consensus tree.

- **Definition:** A split $A|B$ in the tree is the bipartition of the leaves/taxa into two subsets A and B induced by removing an edge or branch from the tree.
- **Definition:** Two splits $A|B$ and $C|D$ are compatible, i.e. not contradictory, if at least one intersection of $A \cap C$, $A \cap D$, $B \cap C$, $B \cap D$ is empty.

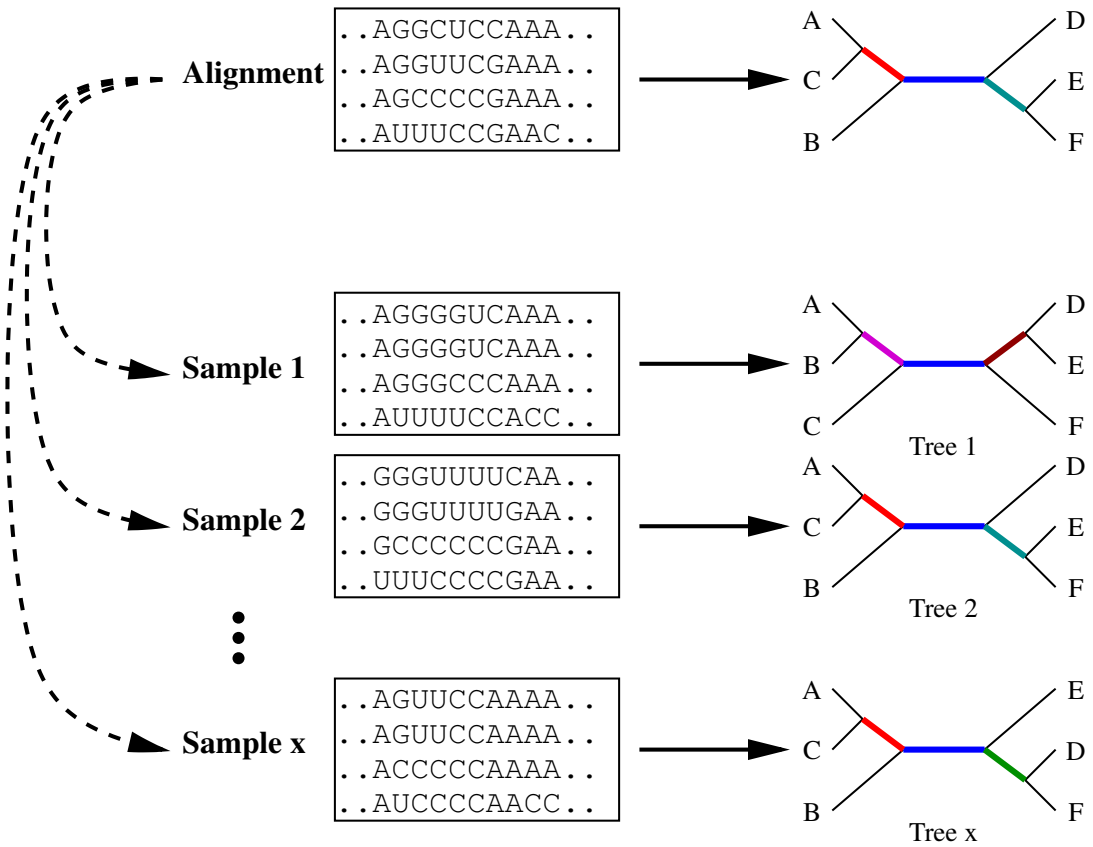
Estimating Confidence: The Bootstrap



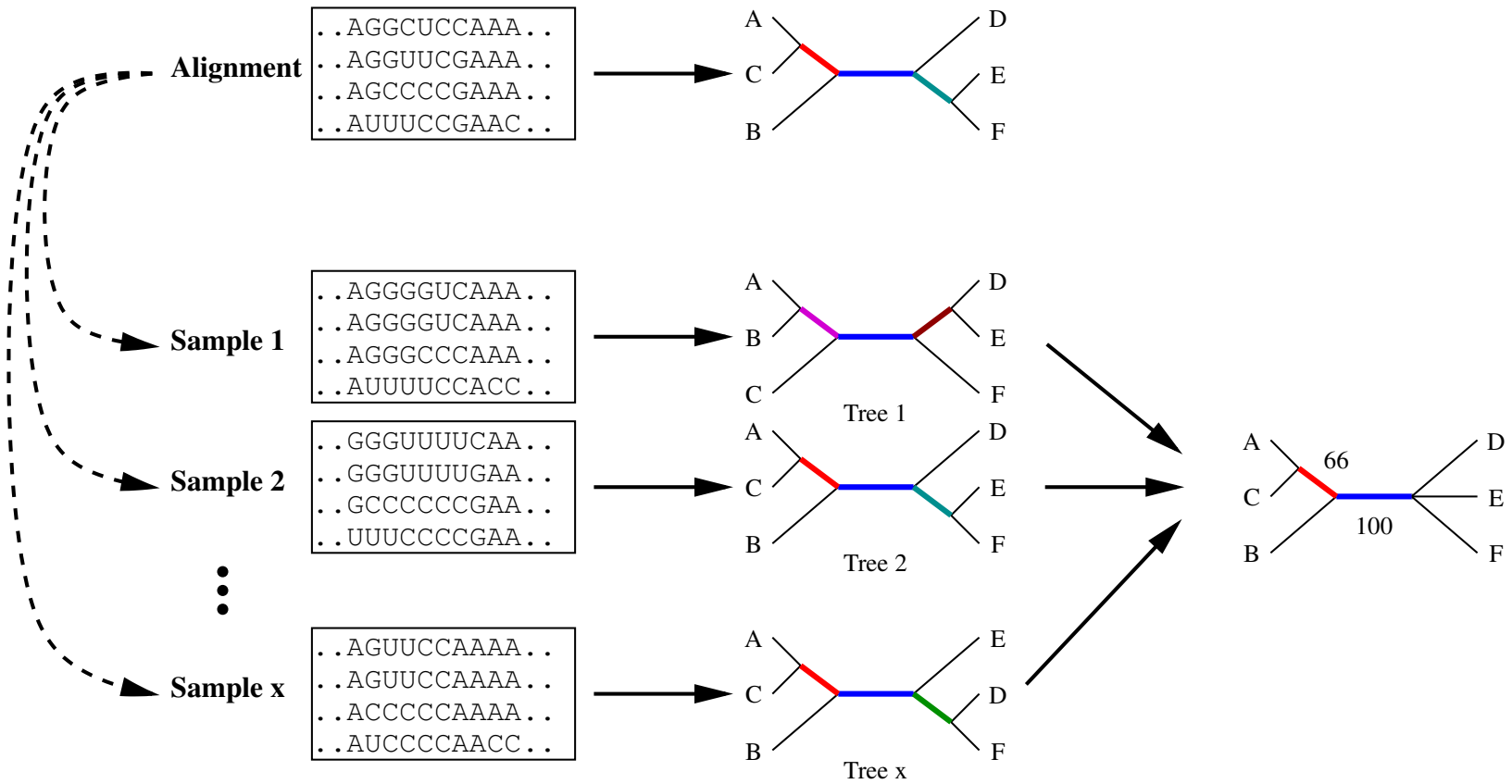
Estimating Confidence: The Bootstrap



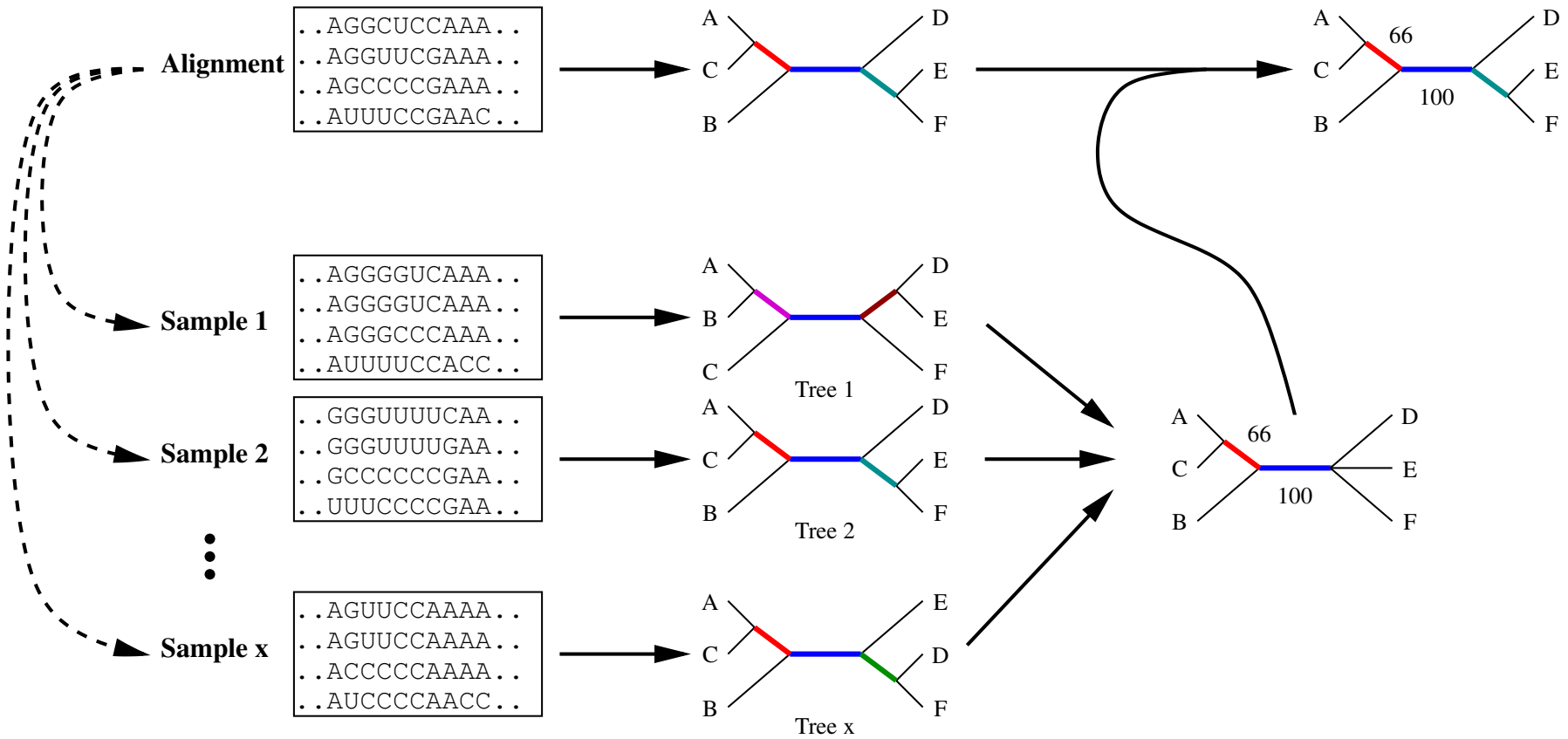
Estimating Confidence: The Bootstrap



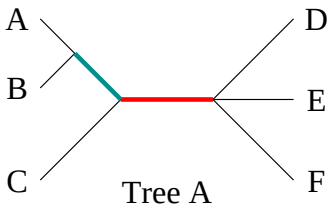
Estimating Confidence: The Bootstrap



Estimating Confidence: The Bootstrap

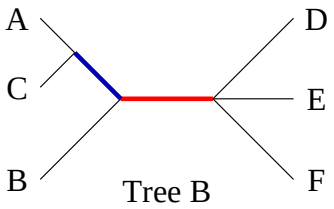


Summarizing Trees: Consensus Methods



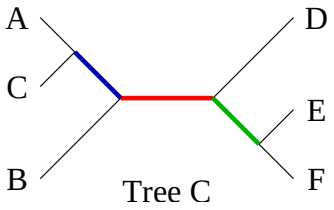
AB|CDEF

ABC|DEF



AC|BDEF

ABC|DEF



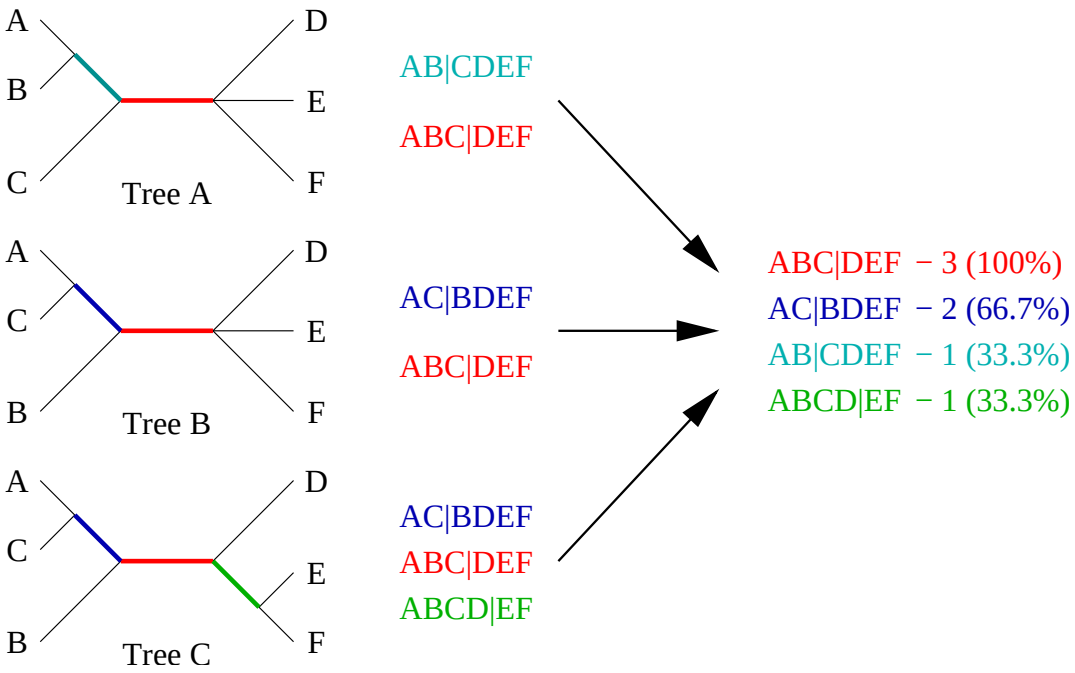
AC|BDEF

ABC|DEF

ABCD|EF

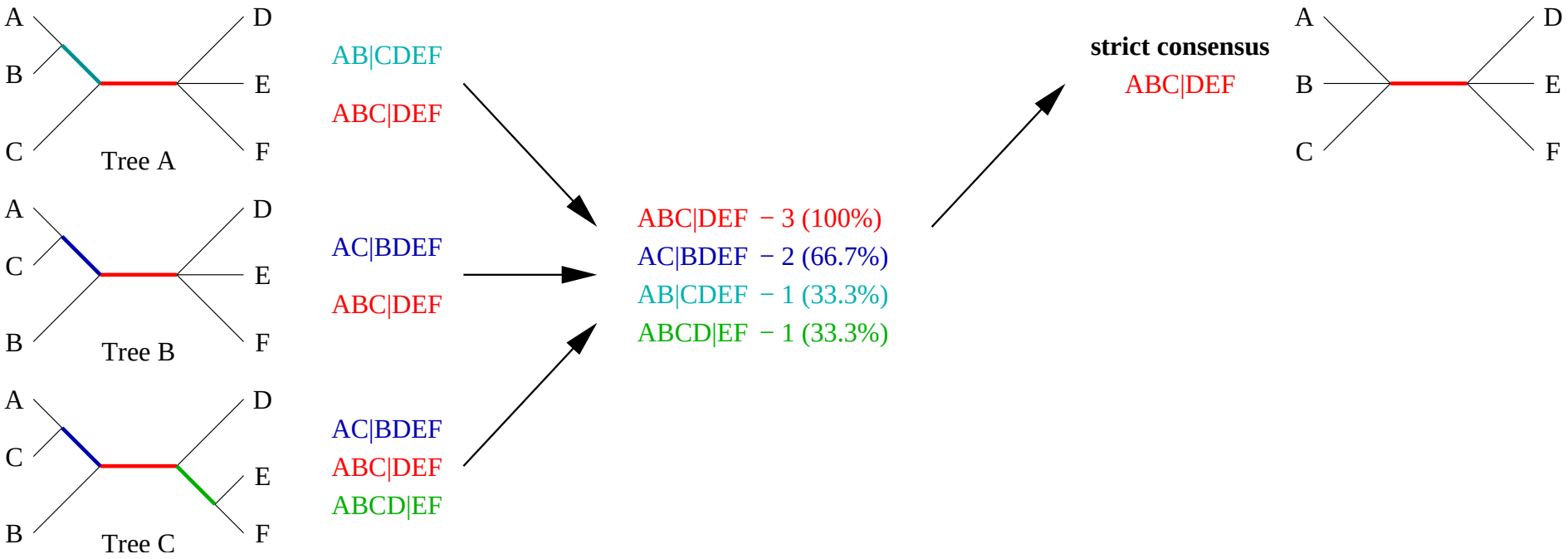
- Strict consensus: contains all splits occuring in all input tree.
- Semi-strict consensus: contains all splits which are not contradicted by any tree.
- Majority consensus M_ℓ: contains all splits which occur in more than ℓ input trees, where $\ell \geq 50\%$ typically exactly 50%.

Summarizing Trees: Consensus Methods



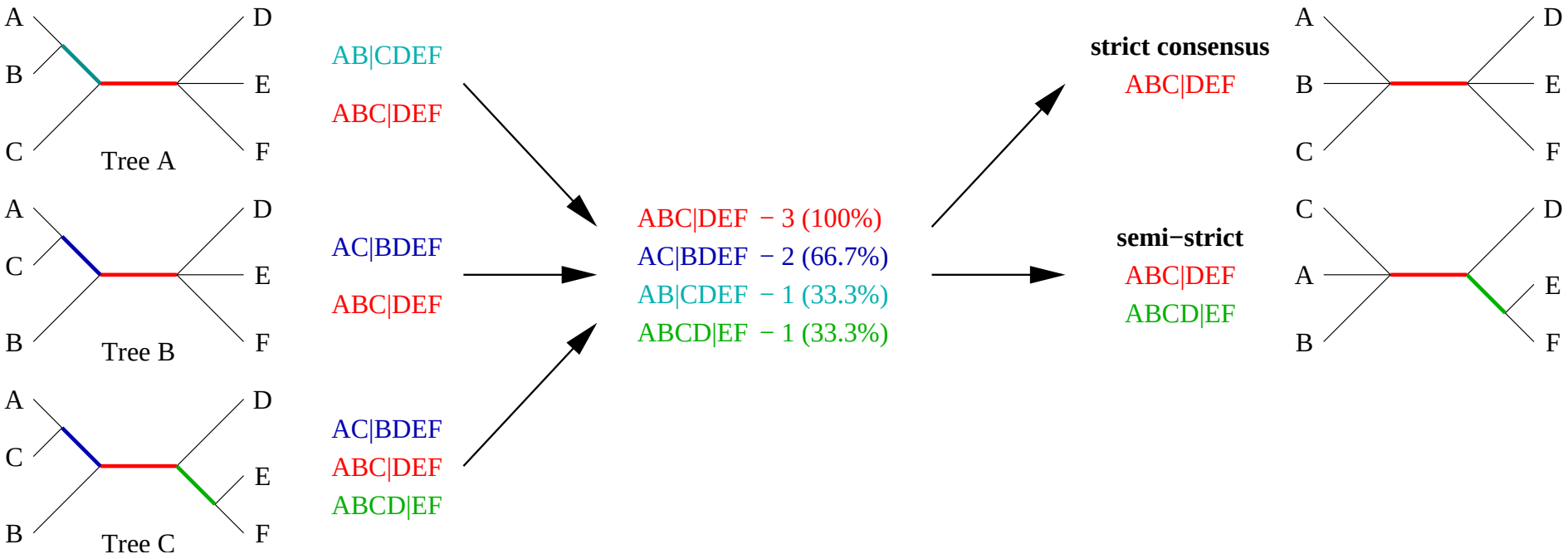
- Strict consensus: contains all splits occuring in all input tree.
- Semi-strict consensus: contains all splits which are not contradicted by any tree.
- Majority consensus M_ℓ: contains all splits which occur in more than ℓ input trees, where $\ell \geq 50\%$ typically exactly 50%.

Summarizing Trees: Consensus Methods



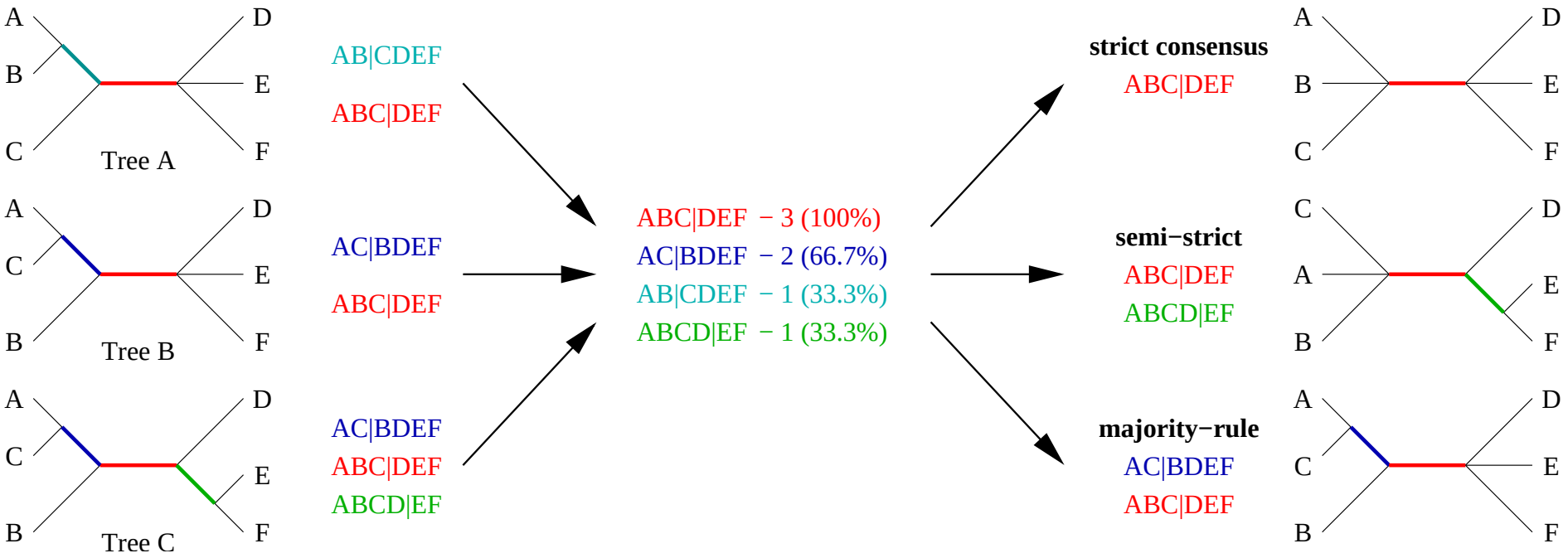
- Strict consensus: contains all splits occuring in all input tree.
- Semi-strict consensus: contains all splits which are not contradicted by any tree.
- Majority consensus M_ℓ: contains all splits which occur in more than ℓ input trees, where $\ell \geq 50\%$ typically exactly 50%.

Summarizing Trees: Consensus Methods



- Strict consensus: contains all splits occuring in all input tree.
- Semi-strict consensus: contains all splits which are not contradicted by any tree.
- Majority consensus M_ℓ: contains all splits which occur in more than ℓ input trees, where $\ell \geq 50\%$ typically exactly 50%.

Summarizing Trees: Consensus Methods



- Strict consensus: contains all splits occuring in all input tree.
- Semi-strict consensus: contains all splits which are not contradicted by any tree.
- Majority consensus M_ℓ: contains all splits which occur in more than ℓ input trees, where $\ell \geq 50\%$ typically exactly 50%.

Bayesian Phylogenetics

The Bayesian approach asks the right question in a hypothesis testing procedure, namely, "What is the probability that this hypothesis is true, given the data?" rather than the classical approach, which asks a question like, "Assuming that this hypothesis is true, what is the probability of the observed data?"

Ewens & Grant: Statistical Methods in Bioinformatics (2010)

Bayesian statistics

Bayesian statistics tries to determine the probability of the hypothesis:

$$\Pr(hypothesis|data) = \frac{\Pr(hypothesis) \times \Pr(data|hypothesis)}{\Pr(data)}$$

where

- $\Pr(data|hypothesis)$ is the 'likelihood'
- $\Pr(hypothesis)$ is the prior distribution
- $\Pr(hypothesis|data)$ is the posterior distribution
- $\Pr(data)$ is the marginal likelihood

Bayesian Phylogenetics

Bayesian statistics tries to determine the probability of the hypothesis (i.e. tree topologies, branch lengths, evolutionary model with its parameters):

$$p(T, M|D) = \frac{p(T) \times p(M) \times \Pr(D|T, M)}{\Pr(D)}$$

where

- $\Pr(D|T, M)$ is the 'likelihood'
- $p(T)$ is the prior distribution on the trees
- $p(M)$ is the prior distribution on the evolutionary model
- $\Pr(T, M|data)$ is the posterior distribution
- $\Pr(D)$ is the marginal likelihood of the data

Bayesian Phylogenetics

Bayesian statistics tries to determine the probability of the hypothesis (i.e. tree topologies, branch lengths, evolutionary model with its parameters):

$$p(T, M|D) = \frac{p(T) \times p(M) \times \Pr(D|T, M)}{\Pr(D)}$$

where

- $\Pr(D|T, M)$ is the 'likelihood'
- $p(T)$ is the prior distribution on the trees
- $p(M)$ is the prior distribution on the evolutionary model
- $\Pr(T, M|data)$ is the posterior distribution
- $\Pr(D)$ is the marginal likelihood of the data
- The output of a Bayesian evolutionary analysis is a probability distribution on trees and parameter values.