Algorithmen der Bioinformatik I
WS 2017/2018

Burkhard Morgenstern
Peter Meinicke

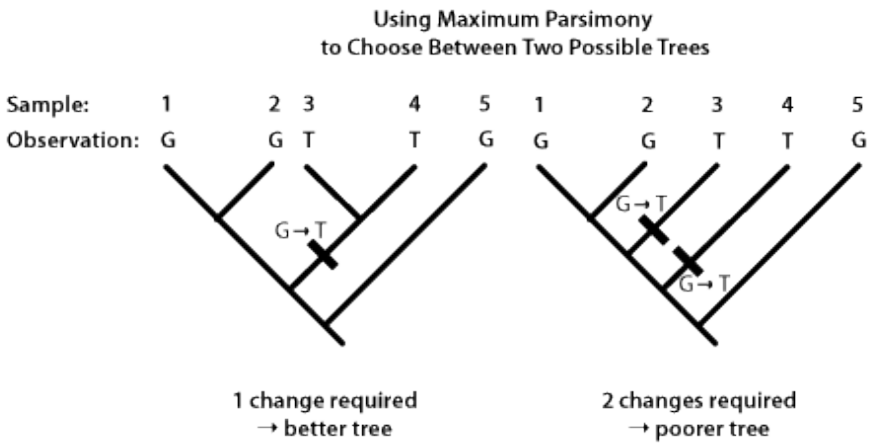
Dept. Bioinformatics
Institute of Microbiology and Genetics (IMG)
University of Göttingen

January 22, 2018

Maximum Parsimony, recap.

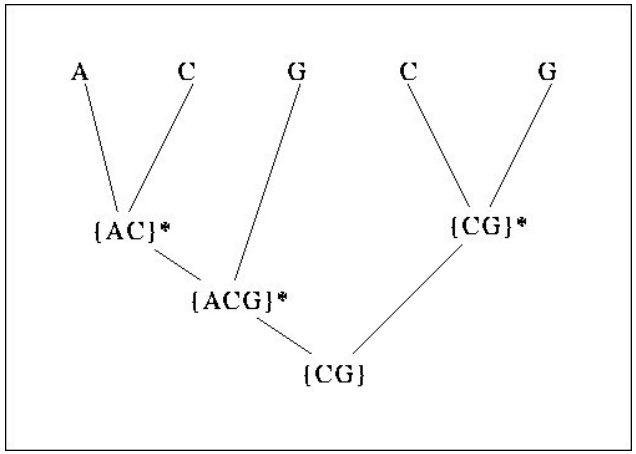
For sequence data: multiple alignment as ‘data matrix’. How many substitutions in alignment column?

1	. . .	*G*	. . .
2	. . .	*G*	. . .
3	. . .	*T*	. . .
4	. . .	*T*	. . .
5	. . .	*G*	. . .



http://www.allanwilsoncentre.ac.nz/

The ‘small parsimony problem’, recap.



Sets R_k of possible nucleotides at inner nodes of a tree for a given column in the alignment.

Recursion:

$$R_k = \begin{cases} R_i \cap R_j & \text{if } R_i \cap R_j \neq \emptyset \quad (1) \\ R_i \cup R_j & \text{if } R_i \cap R_j = \emptyset \quad (2) \end{cases}$$

The ‘small parsimony problem’, recap.

Generalisation: *weighted parsimony*

- ‘cost’ $S(a, b)$ for mutation $a \to b$ or $b \to a$.
- Wanted: tree that minimizes sum of costs of necessary mutations.

Calculate for each node k and nucleotide a minimal costs

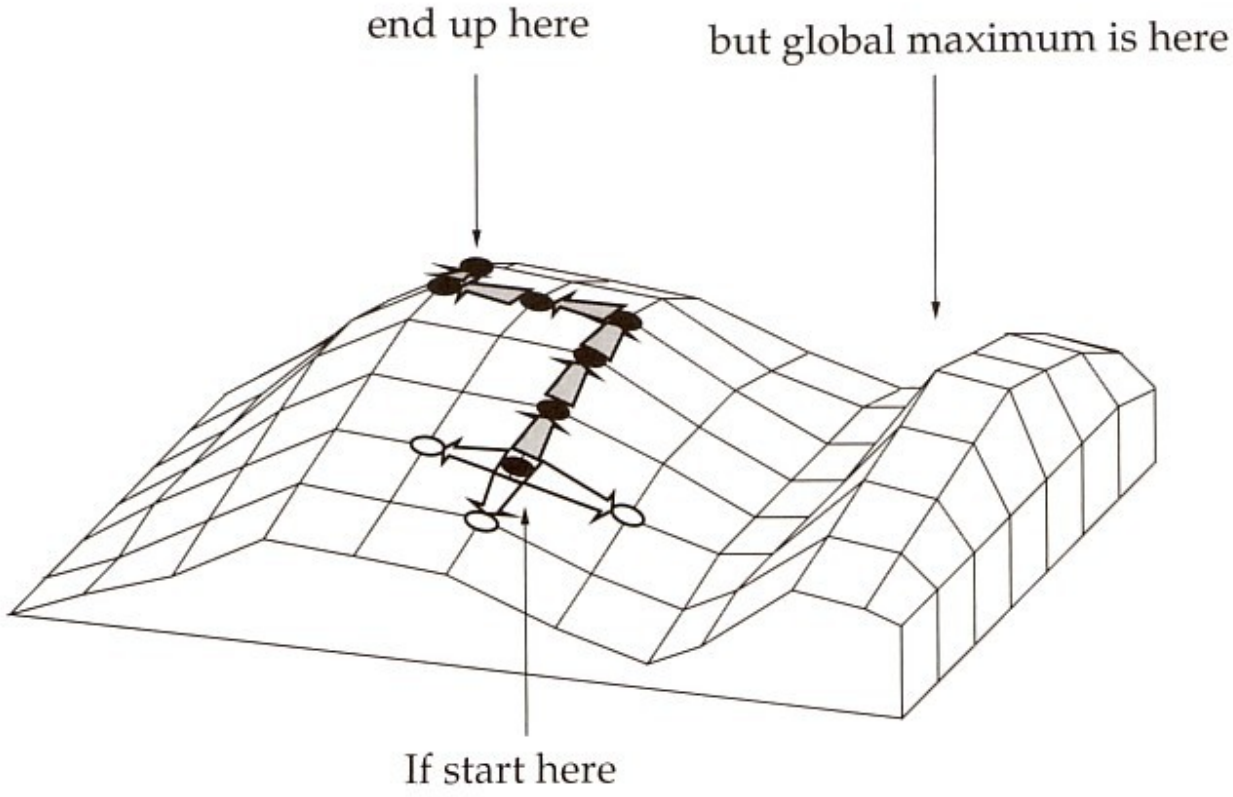
$$S_k(a)$$

for subtree T_k below k, if a is at k

Recursion for inner node k with children i and j:

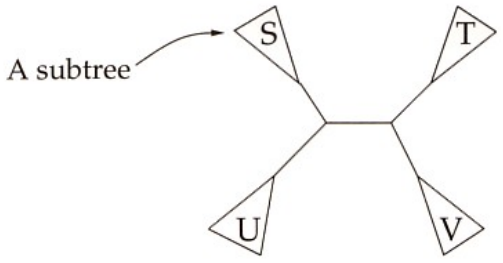
$$S_k(a) = \min_b[S_i(b) + S(a, b)] + \min_b[S_j(b) + S(a, b)]$$

The ‘big parsimony problem’, recap.

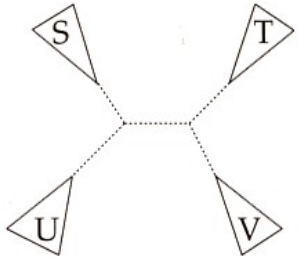


Hill climbing. Height in landscape represents quality of solution.

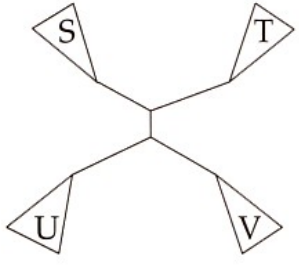
The ‘big parsimony problem’, recap.

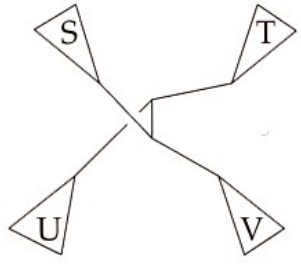


is rearranged by dissolving the connections to an interior branch



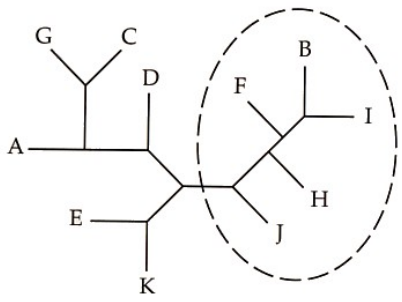
and reforming them in one of the two possible alternative ways:



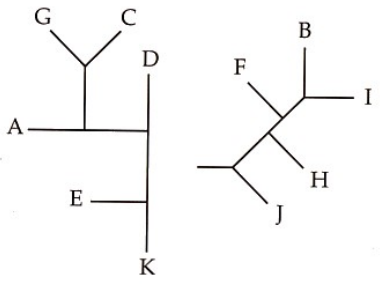


Nearest-neighbour interchanges

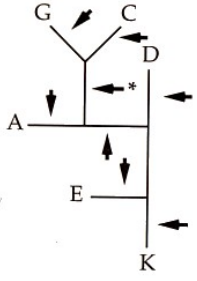
The ‘big parsimony problem’, recap.



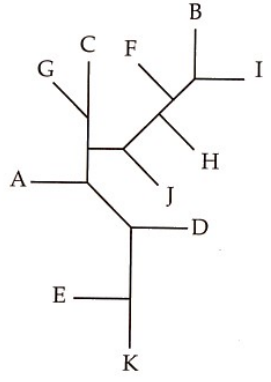
Break a branch, remove a subtree



Add it in, attaching it to one (*) of the other branches

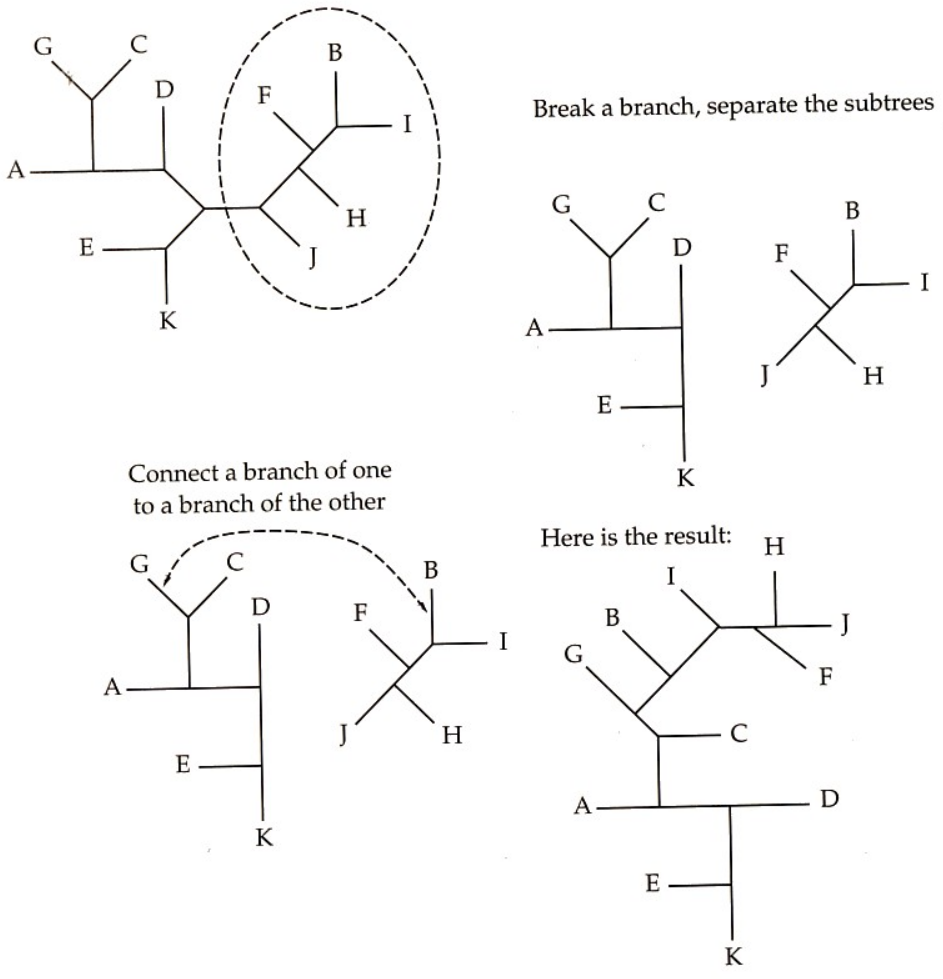


Here is the result:



Neighbouring trees with ‘subtree pruning and regrafting’

The ‘big parsimony problem’, recap.



Neighbouring trees with ‘subtree bisection and reconnection’

The ‘big parsimony problem’

- All three possibilities allow to transform every possible tree topology into every other topology.
- Important: Start *hill climbing* with good initial tree T_0
- Possible construction using *greedy* algorithm, *e.g. sequential addition*.
  Start with 3 leaves, add subsequent leaves in a (locally) optimal way.

The ‘big parsimony problem’

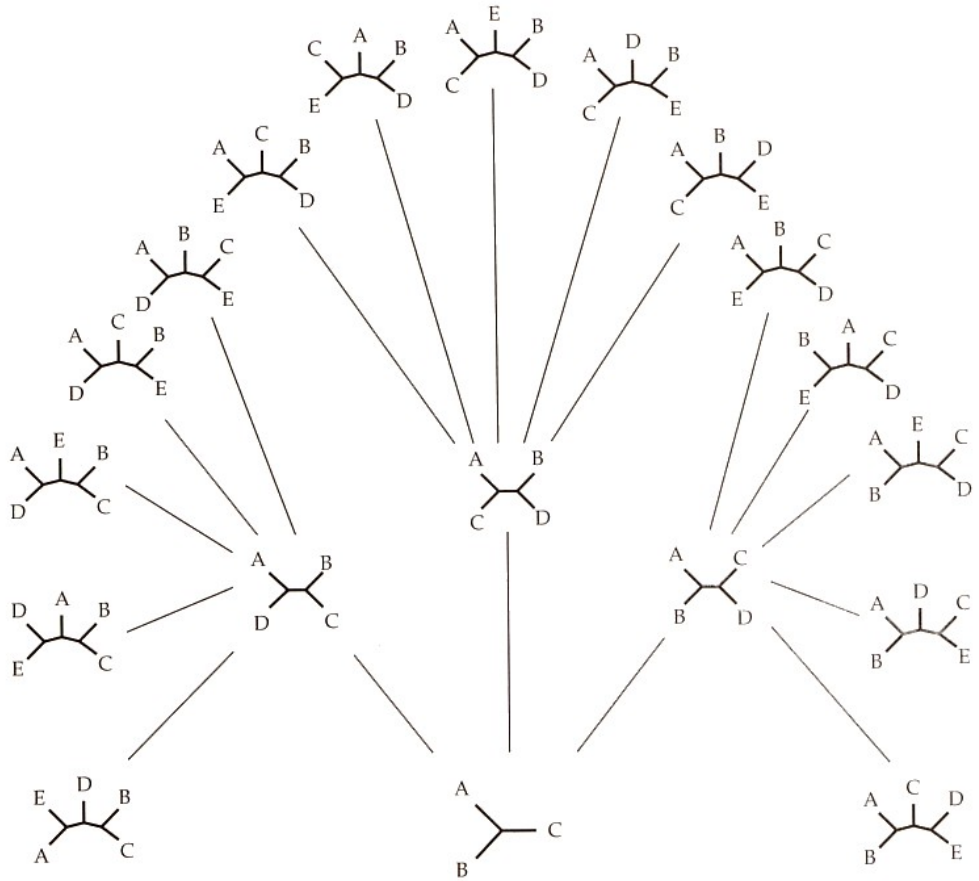
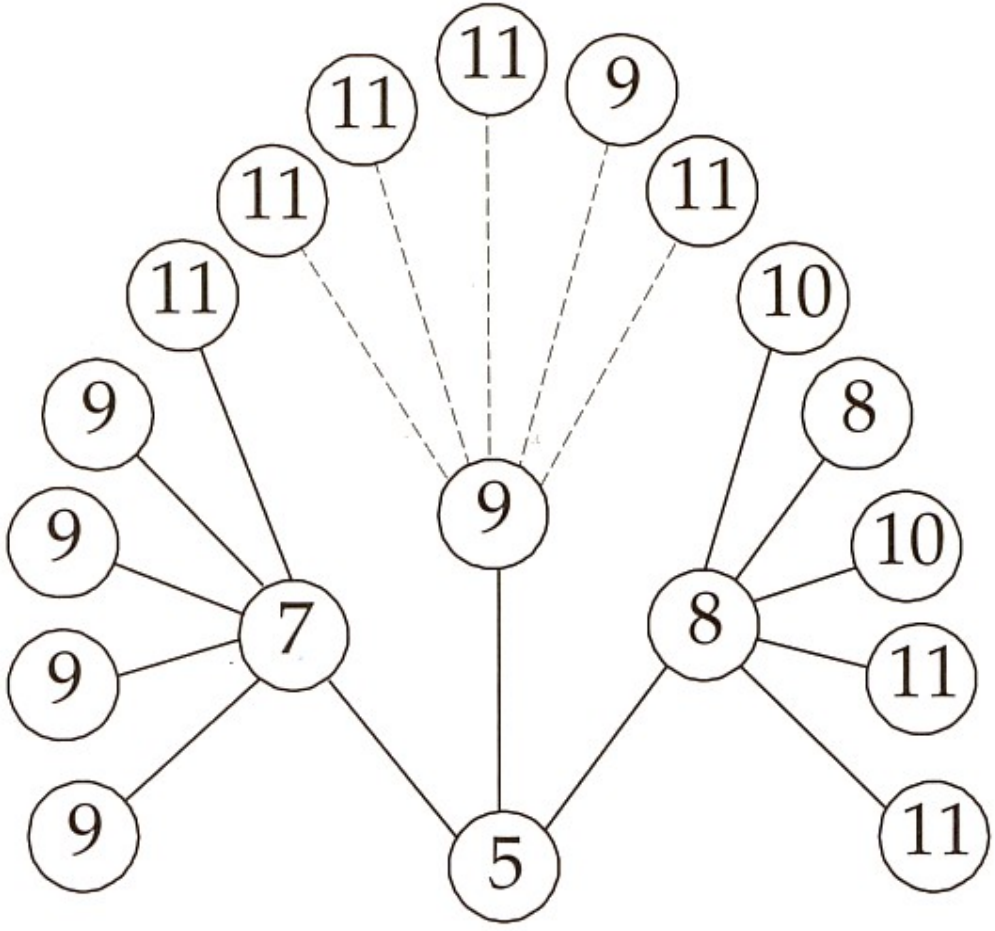


Figure: ‘Greedy’ to find (sub-)optimal topology: sort ‘species’ *A*, *B*, *C*, *D*, *E*. start with topology for *A*, *B*, *C*, place next species optimally into existing topology *etc*.

The ‘big parsimony problem’



Maximum Likelihood

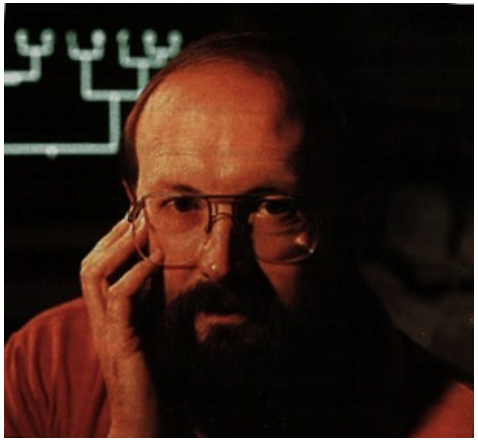


Figure: Joe Felsenstein

Maximum Likelihood

J Mol Evol (1981) 17:368–376

Journal of Molecular Evolution

Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach

Joseph Felsenstein

Department of Genetics, University of Washington, Seattle, Washington 98195, USA

Summary. The application of maximum likelihood techniques to the estimation of evolutionary trees from nucleic acid sequence data is discussed. A computationally feasible method for finding such maximum likelihood estimates is developed, and a computer program is available. This method has advantages over the traditional parsimony algorithms, which can give misleading results if rates of evolution differ in different lineages. It also allows the testing of hypotheses about the constancy of evolutionary rates by likelihood ratio tests, and gives rough indication of the error of the estimate of the tree.

produced by parsimony methods (Edwards 1963; Edwards and Cavalli-Sforza 1964; Camin and Sokal 1965). These methods implicitly assume that change is improbable a priori (Felsenstein 1973, 1979). If the amount of change is small over the evolutionary times being considered, parsimony methods will be well-justified statistical methods.

Most data involve moderate to large amounts of change, and it is in such cases that parsimony methods can fail. When amounts of evolutionary change in different lineages are sufficiently unequal, it can be shown (Felsenstein 1978b) that parsimony methods make an

Figure: Felsenstein, 1981

Maximum Likelihood

Example (Likelihood)

S_1	a	g	c	t	t	c
S_2	a	g	c	t	t	g
S_3	a	g	t	a	t	c
S_4	c	g	t	a	t	c

- Question: What is the *probability* that the observed nucleotides in an alignment column evolved from an (unknown) ancester along a tree T?

Use probabilistic model for substitutions $a \to b$

Maximum Likelihood

What is likelihood?

Situation in random experiment:

Observable data *D* and non-observable *hypothesis H* 'behind data' both depend on chance.

- Known: data *D* (result of random experiments)
- Wanted: 'best' hypothesis *H* 'behind' data *D*

Maximum Likelihood

Example (Dice fair or unfair?)

Different dice (*e.g.* fair and unfair) available, one die randomly drawn and rolled.

- Observed data D: numbers seen on die
- Hypothesis H: a certain die was drawn (*e.g.* fair die)

Question: which hypothesis H should be accepted based on observed data D?

Maximum Likelihood

Consider

$$P(D|H)$$

i.e. conditional probability of D under hypothesis H.

Maximum Likelihood

Distinguish:

- For fixed Hypothesis H,

  $$P(D|H)$$

  defines a *probability measure* on the set of all possible outcomes of random experiment (all possible D).

  E.g. if results $D_1, \ldots, D_k$ are possible, one has

  $$\sum_i P(D_i|H) = 1$$

  for each hypothesis H

Maximum Likelihood

- But: for fixed observed data D,

$$P(D|H)$$

does *not* define a probability measure on the set of all hypotheses. (*e.g.* sum of conditional probabilities in general *not* equal 1)

For observed D, *likelihood* of hypothesis H defined as $P(D|H)$.

Maximum likelihood (ML) principle: accept hypothesis that maximizes likelihood $P(D|H)$, given observed data D.

Maximum Likelihood

Example (fair and unfair dice)

For unfair dice:

$$P(1) = \cdots = P(5) = 1/10 \quad (1)$$
$$P(6) = 1/2 \quad (2)$$

For fair dice:

$$P(1) = \cdots = P(6) = 1/6$$

Experiment: draw one die randomly, roll it 3 times.

Maximum Likelihood

Example (fair and unfair dice)

Result (observed data D): $4, 6, 6$

Two hypotheses considered: H_1: die fair, H_2: die unfair

Then *likelihood* of H_1 is

$$P(D|H_1) = (1/6)^3 \approx 0.0046\ldots$$

likelihood of H_2 is

$$P(D|H_2) = 1/10 \cdot (1/2)^2 = 0.025$$

Thus, H_2 has higher likelihood.

Maximum Likelihood

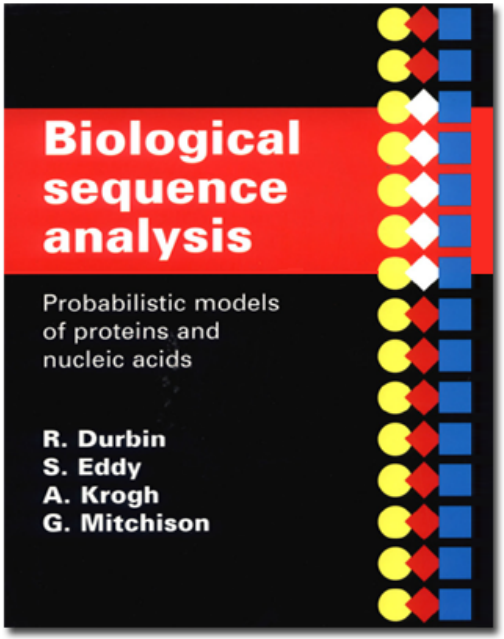


Figure: In the following: Figures and notation from R. Durbin *et al.*

Maximum Likelihood

Application of ML *to phylogeny reconstruction:*

Consider tree T (including branch lengths) as *hypothesis* H, set of aligned sequences $x^1, \ldots, x^n$ as observed data D.

Goal: find tree T with maximal likelihood

$$P(x^1, \ldots, x^n | T)$$

Question: how to calculate this (conditional) probability?

Maximum Likelihood

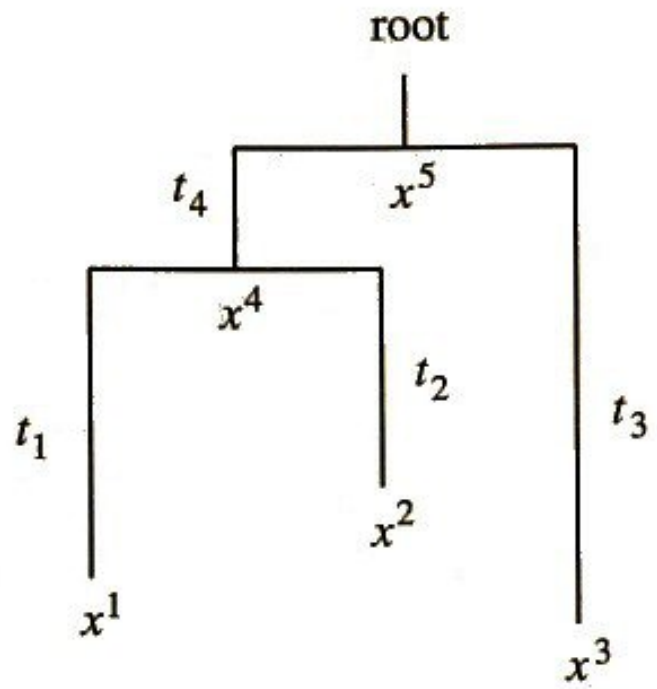


Figure: Tree T with branch lengths $t_1, \ldots, t_4$ for observed sequences x^1, x^2, x^3 at leaves and unknown sequences x^4, x^5 at internal (ancestral) nodes

Maximum Likelihood

Calculate $P(x^1, \ldots, x^n | T)$ for given:

- Multiple sequence alignment (ignore insertions/deletions)
- Probabilistic model that defines for residues a and b and branch length t:
  - *Conditional* probability $P(b|a,t)$ of observing b if there was a at the same position for time difference t
  - *Background* probability q_a

t interpreted as *time* or, more realistically, as *time* $\times$ *mutation rate*

Maximum Likelihood

Calculate for tree T and given probabilistic model:

- Probability of nucleotides in *single* columns of multiple alignment
- Probability of (aligned) sequences, given T as *product* of probabilities of alignment columns.

Maximum Likelihood

Simplest case: two sequences x^1, x^2, only one position u considered.

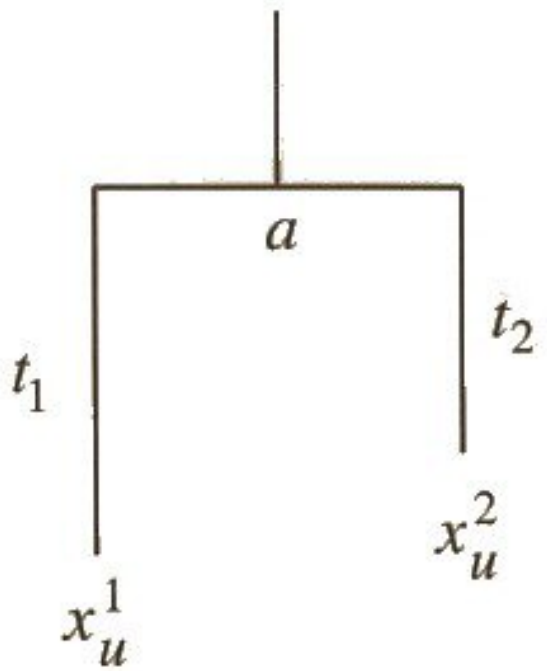


Figure: Tree T with branch lengths t_1, t_2 for observed nucleotides x_u^1, x_u^2 at leaves and nucleotide a at root

Maximum Likelihood

In general, for events A and B:

$$P(A, B) = P(A|B) \cdot P(B)$$

Therefore, probability to observe x_u^1, x_u^2 and a at leaves and root, respectively:

$$P(x_u^1, x_u^2, a|T) = q_a P(x_u^1|a, t_1) P(x_u^2|a, t_2)$$

Maximum Likelihood

If nucleotide at root position unknown, one has to sum over *all possible* nucleotides at the root. Therefore:

$$P(x_u^1, x_u^2|T) = \sum_a q_a P(x_u^1|a, t_1) P(x_u^2|a, t_2)$$

for one single column u.

Probability of *entire* alignment as product of probabilities of all columns:

$$P(x^1, x^2|T) = \prod_u \sum_a q_a P(x_u^1|a, t_1) P(x_u^2|a, t_2)$$

Maximum Likelihood

General case for n sequences: calculate probability for observed nucleotides in column u:

$$P(x_u^1, \ldots, x_u^n | T)$$

To this end: calculate sum over *all* combinations of internal nodes

$$x_u^{n+1}, \ldots, x_u^{2n-1}$$

Maximum Likelihood

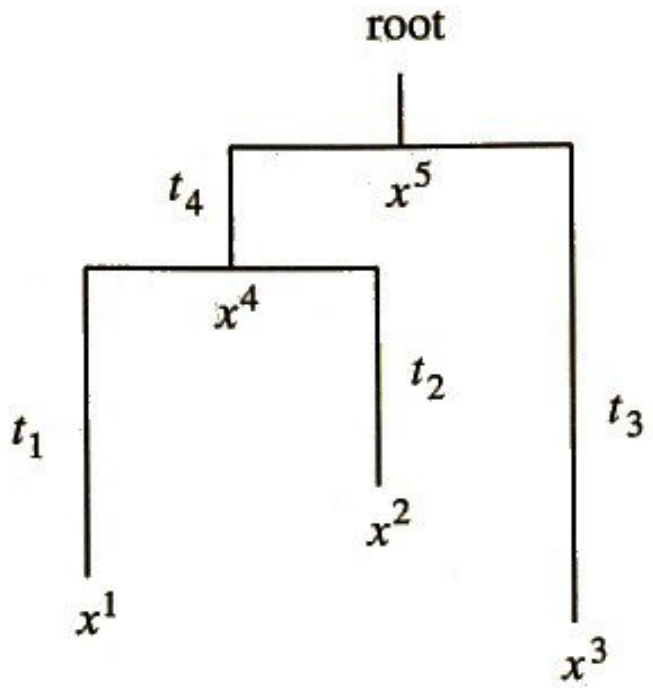


Figure: For *given* nucleotides $x_u^{n+1}, \ldots, x_u^{2n-1}$ at internal nodes: $P(x_u^1, \ldots, x_u^n)$ calculated as produkt of probabilities $P(x_u^i | x_u^j, t_i)$ For unknown internal nodes: take sum over all combinations of internal nodes.

Maximum Likelihood

Probability for alignment column u, given tree $T, t_1, \ldots, t_{2n-2}$:

$$P(x_u^1, \ldots, x_u^n | T, t_1, \ldots, t_{2n-2}) =$$

$$\sum_{x_u^{n+1}, \ldots, x_u^{n2-1}} q_{x_u^{2n-1}} \prod_{i=n+1}^{2n-2} P(x_u^i | x_u^{\alpha(i)}, t_i) \prod_{i}^{n} P(x_u^i | x_u^{\alpha(i)}, t_i)$$

For full sequences: calculate *product* of probabilities of alignment columns u

Maximum Likelihood

Problem: for n leaves, sum over 4^{n-1} possibilities for nucleotides at internal nodes.

Solution: dynamic programming (similar as with parsimony)

Calculate recursively:
$P(L_k|a)$ = probability to observe nucleotides at leaves in sub-tree under node k, if nucleotide a at node k.

Maximum Likelihood

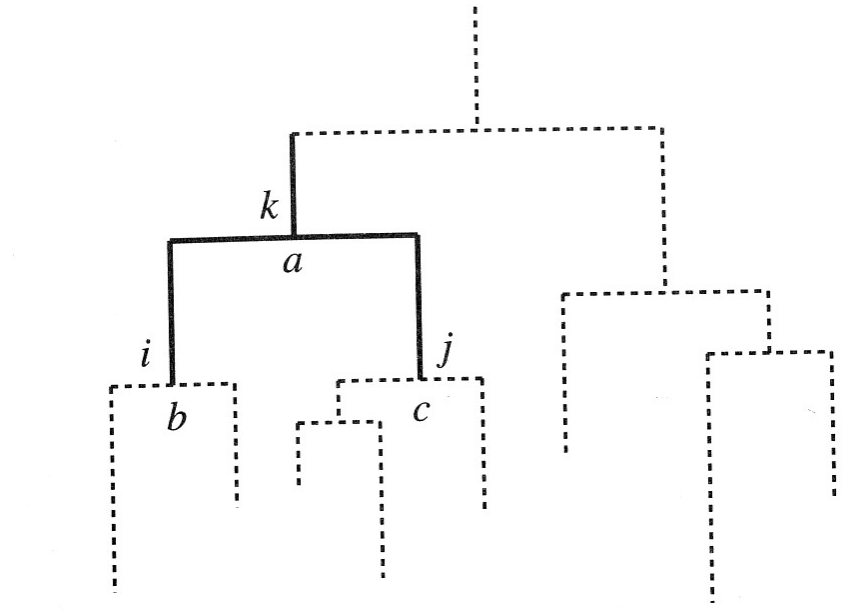


Figure: Recursion to calculate $P(L_k|a)$

Maximum Likelihood

Calculate $P(L_k|a)$ recursively for *internal* nodes k:

For daughter nodes i, j with nucleotides b, c and branch lenghts t_i, t_j:

$$P(L_k|a) = \sum_{b,c} P(b|a,t_i)P(L_i|b)P(c|a,t_j)P(L_j|c)$$

For leave k:

$$P(L_k|a) = \begin{cases} 1 \text{ falls } a = x_u^k \\ 0 \text{ sonst} \end{cases}$$

Likelihood for alignment column u:

$$\sum_a P(L_{2n-1}|a)q_a$$

Maximum Likelihood

Similar as in parsimony: position of root not relevant, if certain conditions are given.

For maximum likelihood: model for mutations *reversibel*, *i.e.*

$$P(b|a,t)q_a = P(a|b,t)q_b$$

Maximum Likelihood

Difference to weighted parsimony: calculate *sum* of probabilities for possible nucleotides at internal nodes instead of *minimum* cost.

Find best tree, similar as with *maximum parsimony*

- 'Hill-Climbing' in the 'landscape' of all possible trees.
- For given topology: find optimal branch lengths by standard optimization methods.

Filtered Spaced-Word Matches

Ongoing projects in Göttingen

Search for spaced-word matches *w.r.t.* given binary pattern P

Example (*FSWM*, $P = 11010001$)

S_1	A	T	C	A	G	G	A	C	A	T	A	C	G	C	C	A	T
S_2	C	G	G	A	C	A	T	G	C	T	C	C	A	G	C		

Filtered Spaced-Word Matches

Ongoing projects in Göttingen

Search for spaced-word matches *w.r.t.* given binary pattern P

Example (*FSWM*, $P = 11010001$)

S_1	A	T	C	A	G	G	A	C	A	T	A	C	G	C	C	A	T
S_2	C	G	G	A	C	A	T	G	C	T	C	C	A	G	C		

Filtered Spaced-Word Matches

Ongoing projects in Göttingen

Search for spaced-word matches *w.r.t.* given binary pattern P

Example (*FSWM*, $P = 11010001$)

S_1	A	T	C	A	G	G	A	C	A	T	A	C	G	C	C	A	T
S_2	C	G	G	A	C	A	T	G	C	T	C	C	A	G	C		

Filtered Spaced-Word Matches

Ongoing projects in Göttingen

Search for spaced-word matches *w.r.t.* given binary pattern P

Example (*FSWM*, $P = 11010001$)

S_1	A	T	C	A	G	G	A	C	A	T	A	C	G	C	C	A	T
S_2	C	G	G	A	C	A	T	G	C	T	C	C	A	G	C		

Filtered Spaced-Word Matches

Ongoing projects in Göttingen

Search for spaced-word matches *w.r.t.* given binary pattern P

Example (*FSWM*, $P = 11010001$)

S_1	A	T	C	A	G	G	A	C	A	T	A	C	G	C	C	A	T
S_2	C	G	G	A	C	A	T	G	C	T	C	C	A	G	C		

Filtered Spaced-Word Matches

Ongoing projects in Göttingen

Search for spaced-word matches *w.r.t.* given binary pattern P

Example (*FSWM*, $P = 11010001$)

S_1	...	*A*	*C*	*A*	*T*	*A*	*C*	*G*	*C*	...
S_2	...	*A*	*C*	*A*	*T*	*G*	*C*	*T*	*C*	...
		1	1	0	1	0	0	0	1	

Filtered Spaced-Word Matches

Ongoing projects in Göttingen

Search for spaced-word matches *w.r.t.* given binary pattern P

Example ($FSWM$, $P = 11010001$)

S_1	...	A	C	A	T	A	C	G	C	...
S_2	...	A	C	A	T	G	C	T	C	...
		1	1	0	1	0	0	0	1	

Consider nucleotides at *don't-care* positions to estimate distances

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C
S_2	C	A	G	A	C	A	G	A

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

$$
\begin{array}{lcccccccc}
S_1 & C & A & C & A & G & A & C & \\
S_2 & C & A & G & A & C & A & G & A \\
 & & & & & & & & \\
 & C & A & * & A & & & (S_1) &
\end{array}
$$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

C	A	$*$	A		(S_1)
A	C	$*$	G		(S_1)

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

C	A	$*$	A	(S_1)
A	C	$*$	G	(S_1)
C	A	$*$	A	(S_1)

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

C	A	$*$	A	(S_1)
A	C	$*$	G	(S_1)
C	A	$*$	A	(S_1)
A	G	$*$	C	(S_1)

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

C	A	$*$	A	(S_1)
A	C	$*$	G	(S_1)
C	A	$*$	A	(S_1)
A	G	$*$	C	(S_1)
C	A	$*$	A	(S_2)

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

$$
\begin{array}{lcccccccc}
S_1 & C & A & C & A & G & A & C & \\
S_2 & C & A & G & A & C & A & G & A
\end{array}
$$

$$
\begin{array}{cccc}
C & A & * & A \quad (S_1) \\
A & C & * & G \quad (S_1) \\
C & A & * & A \quad (S_1) \\
A & G & * & C \quad (S_1) \\
C & A & * & A \quad (S_2) \\
A & G & * & C \quad (S_2)
\end{array}
$$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

C	A	$*$	A	(S_1)
A	C	$*$	G	(S_1)
C	A	$*$	A	(S_1)
A	G	$*$	C	(S_1)
C	A	$*$	A	(S_2)
A	G	$*$	C	(S_2)
G	A	$*$	A	(S_2)

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

$$
\begin{array}{lcccccccc}
S_1 & C & A & C & A & G & A & C & \\
S_2 & C & A & G & A & C & A & G & A
\end{array}
$$

$$
\begin{array}{cccc}
C & A & * & A \quad (S_1) \\
A & C & * & G \quad (S_1) \\
C & A & * & A \quad (S_1) \\
A & G & * & C \quad (S_1) \\
C & A & * & A \quad (S_2) \\
A & G & * & C \quad (S_2) \\
G & A & * & A \quad (S_2) \\
A & C & * & G \quad (S_2)
\end{array}
$$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

C	A	$*$	A	(S_1)
A	C	$*$	G	(S_1)
C	A	$*$	A	(S_1)
A	G	$*$	C	(S_1)
C	A	$*$	A	(S_2)
A	G	$*$	C	(S_2)
G	A	$*$	A	(S_2)
A	C	$*$	G	(S_2)
C	A	$*$	A	(S_2)

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

$$
\begin{array}{llllllll}
S_1 & C & A & C & A & G & A & C \\
S_2 & C & A & G & A & C & A & G & A
\end{array}
$$

$$
\begin{array}{llll}
C & A & * & A \quad (S_1) \\
A & C & * & G \quad (S_1) \\
C & A & * & A \quad (S_1) \\
A & G & * & C \quad (S_1) \\
C & A & * & A \quad (S_2) \\
A & G & * & C \quad (S_2) \\
G & A & * & A \quad (S_2) \\
A & C & * & G \quad (S_2) \\
C & A & * & A \quad (S_2)
\end{array}
$$

List $\mathcal{L}$ of all spaced words in S_1 and S_2

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

$$
\begin{array}{lcccccccc}
S_1 & C & A & C & A & G & A & C & \\
S_2 & C & A & G & A & C & A & G & A
\end{array}
$$

$$
\begin{array}{cccc}
A & C & * & G \\
A & C & * & G \\
A & G & * & C \\
A & G & * & C \\
C & A & * & A \\
C & A & * & A \\
C & A & * & A \\
C & A & * & A \\
G & A & * & A
\end{array}
\qquad
\begin{array}{l}
(S_1) \\
(S_2) \\
(S_1) \\
(S_2) \\
(S_1) \\
(S_1) \\
(S_2) \\
(S_2) \\
(S_2)
\end{array}
$$

Sort $\mathcal{L}$ in lexicographic order

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

A	C	$*$	G	(S_1)
A	C	$*$	G	(S_2)
A	G	$*$	C	(S_1)
A	G	$*$	C	(S_2)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_2)
C	A	$*$	A	(S_2)
G	A	$*$	A	(S_2)

Identical spaced-words in *buckets* of $\mathcal{L}$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

A	C	$*$	G	(S_1)
A	C	$*$	G	(S_2)
A	G	$*$	C	(S_1)
A	G	$*$	C	(S_2)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_2)
C	A	$*$	A	(S_2)
G	A	$*$	A	(S_2)

Identical spaced-words in *buckets* of $\mathcal{L}$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

A	C	$*$	G	(S_1)
A	C	$*$	G	(S_2)
A	G	$*$	C	(S_1)
A	G	$*$	C	(S_2)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_2)
C	A	$*$	A	(S_2)
G	A	$*$	A	(S_2)

Identical spaced-words in *buckets* of $\mathcal{L}$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

A	C	$*$	G	(S_1)
A	C	$*$	G	(S_2)
A	G	$*$	C	(S_1)
A	G	$*$	C	(S_2)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_2)
C	A	$*$	A	(S_2)
G	A	$*$	A	(S_2)

Identical spaced-words in *buckets* of $\mathcal{L}$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

A	C	$*$	G	(S_1)
A	C	$*$	G	(S_2)
A	G	$*$	C	(S_1)
A	G	$*$	C	(S_2)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_2)
C	A	$*$	A	(S_2)
G	A	$*$	A	(S_2)

Identical spaced-words in *buckets* of $\mathcal{L}$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

$$
\begin{array}{lcccccccc}
S_1 & C & A & C & A & G & A & C & \\
S_2 & C & A & G & A & C & A & G & A
\end{array}
$$

$$
\begin{array}{cccc}
A & C & * & G \\
A & C & * & G \\
A & G & * & C \\
A & G & * & C \\
C & A & * & A \\
C & A & * & A \\
C & A & * & A \\
C & A & * & A \\
G & A & * & A
\end{array}
\qquad
\begin{array}{l}
(S_1) \\
(S_2) \\
(S_1) \\
(S_2) \\
(S_1) \\
(S_1) \\
(S_2) \\
(S_2) \\
(S_2)
\end{array}
$$

Identical spaced-words in *buckets* of $\mathcal{L}$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

A	C	$*$	G	(S_1)
A	C	$*$	G	(S_2)
A	G	$*$	C	(S_1)
A	G	$*$	C	(S_2)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_2)
C	A	$*$	A	(S_2)
G	A	$*$	A	(S_2)

Identical spaced-words in *buckets* of $\mathcal{L}$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

A	C	$*$	G	(S_1)
A	C	$*$	G	(S_2)
A	G	$*$	C	(S_1)
A	G	$*$	C	(S_2)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_2)
C	A	$*$	A	(S_2)
G	A	$*$	A	(S_2)

Identical spaced-words in *buckets* of $\mathcal{L}$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

A	C	$*$	G	(S_1)
A	C	$*$	G	(S_2)
A	G	$*$	C	(S_1)
A	G	$*$	C	(S_2)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_2)
C	A	$*$	A	(S_2)
G	A	$*$	A	(S_2)

Identical spaced-words in *buckets* of $\mathcal{L}$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

A	C	$*$	G	(S_1)
A	C	$*$	G	(S_2)
A	G	$*$	C	(S_1)
A	G	$*$	C	(S_2)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_2)
C	A	$*$	A	(S_2)
G	A	$*$	A	(S_2)

Identical spaced-words in *buckets* of $\mathcal{L}$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

S_1	C	A	C	A	G	A	C	
S_2	C	A	G	A	C	A	G	A

A	C	$*$	G	(S_1)
A	C	$*$	G	(S_2)
A	G	$*$	C	(S_1)
A	G	$*$	C	(S_2)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_1)
C	A	$*$	A	(S_2)
C	A	$*$	A	(S_2)
G	A	$*$	A	(S_2)

Identical spaced-words in *buckets* of $\mathcal{L}$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

$$
\begin{array}{llllllll}
S_1 & C & A & C & A & G & A & C \\
S_2 & C & A & G & A & C & A & G & A
\end{array}
$$

$$
\begin{array}{llll}
A & C & * & G \quad (S_1) \\
A & C & * & G \quad (S_2) \\
A & G & * & C \quad (S_1) \\
A & G & * & C \quad (S_2) \\
C & A & * & A \quad (S_1) \\
C & A & * & A \quad (S_1) \\
C & A & * & A \quad (S_2) \\
C & A & * & A \quad (S_2) \\
G & A & * & A \quad (S_2)
\end{array}
$$

Identical spaced-words in *buckets* of $\mathcal{L}$

Finding spaced-word matches

Example (Find spaced-word matches by *sorting*, $P = 1101$)

$$
\begin{array}{lcccccccc}
S_1 & C & A & C & A & G & A & C & \\
S_2 & C & A & G & A & C & A & G & A
\end{array}
$$

$$
\begin{array}{cccc}
A & C & * & G \\
A & C & * & G \\
A & G & * & C \\
A & G & * & C \\
C & A & * & A \\
C & A & * & A \\
C & A & * & A \\
C & A & * & A \\
G & A & * & A
\end{array}
\qquad
\begin{array}{c}
(S_1) \\
(S_2) \\
(S_1) \\
(S_2) \\
(S_1) \\
(S_1) \\
(S_2) \\
(S_2) \\
(S_2)
\end{array}
$$

Identical spaced-words in *buckets* of $\mathcal{L}$

Remove *low-scoring* spaced-word matches

To filter out random background spaced-word matches:

- Use nucleotide substitution matrix
  (Chiaromonte *et al.*, 2002)
- Calculate *score* for each spaced-word match:
  Sum of substitution scores at *don't-care* positions
- Discard spaced-word matches with score below threshold

Remove *low-scoring* spaced-word matches

	A	C	G	T
A	91	-114	-31	-123
C		100	-125	-31
G			100	-114
T				91

Example (Score of spaced-word match, $P = 1100101$)

S_1 :	G	C	T	G	T	A	T	A	C	G	T	C
S_2 :	G	T	A	C	A	C	T	T	A	T		

Remove *low-scoring* spaced-word matches

	A	C	G	T
A	91	-114	-31	-123
C		100	-125	-31
G			100	-114
T				91

Example (Score of spaced-word match, $P = 1100101$)

S_1 :	G	C	T	G	T	A	T	A	C	G	T	C
S_2 :	G	T	A	C	A	C	T	T	A	T		

Remove *low-scoring* spaced-word matches

	A	C	G	T
A	91	-114	-31	-123
C		100	-125	-31
G			100	-114
T				91

Example (Score of spaced-word match, $P = 1100101$)

S_1 :	G	C	T	G	T	A	T	A	C	G	T	C
S_2 :	G	T	A	C	A	C	T	T	A	T		

Remove *low-scoring* spaced-word matches

	A	C	G	T
A	91	-114	-31	-123
C		100	-125	-31
G			100	-114
T				91

Example (Score of spaced-word match, $P = 1100101$)

S_1 :	G	C	T	G	T	A	T	A	C	G	T	C	
S_2 :				G	T	A	C	A	C	T	T	A	T
P :					1	1	0	0	1	0	1		

Remove *low-scoring* spaced-word matches

	A	C	G	T
A	91	-114	-31	-123
C		100	-125	-31
G			100	-114
T				91

Example (Score of spaced-word match, $P = 1100101$)

S_1 :	G	C	T	G	T	A	T	A	C	G	T	C	
S_2 :				G	T	A	C	A	C	T	T	A	T
P :					1	1	0	0	1	0	1		

Nucleotides at *don't-care* positions

Remove *low-scoring* spaced-word matches

	A	C	G	T
A	91	-114	-31	-123
C		100	-125	-31
G			100	-114
T				91

Example (Score of spaced-word match, $P = 1100101$)

S_1 :	G	C	T	G	T	A	T	A	C	G	T	C	
S_2 :				G	T	A	C	A	C	T	T	A	T
P :					1	1	0	0	1	0	1		

Score = -31 + 91 -114 = -54

Remove *low-scoring* spaced-word matches

To remove background noise:

- Remove spaced words with score below T.
- Default value $T = 0$

To visualize distribution of spaced-word matches: plot number of spaced word matches against scores
('Spaced-word histogram')

Spaced-word histograms

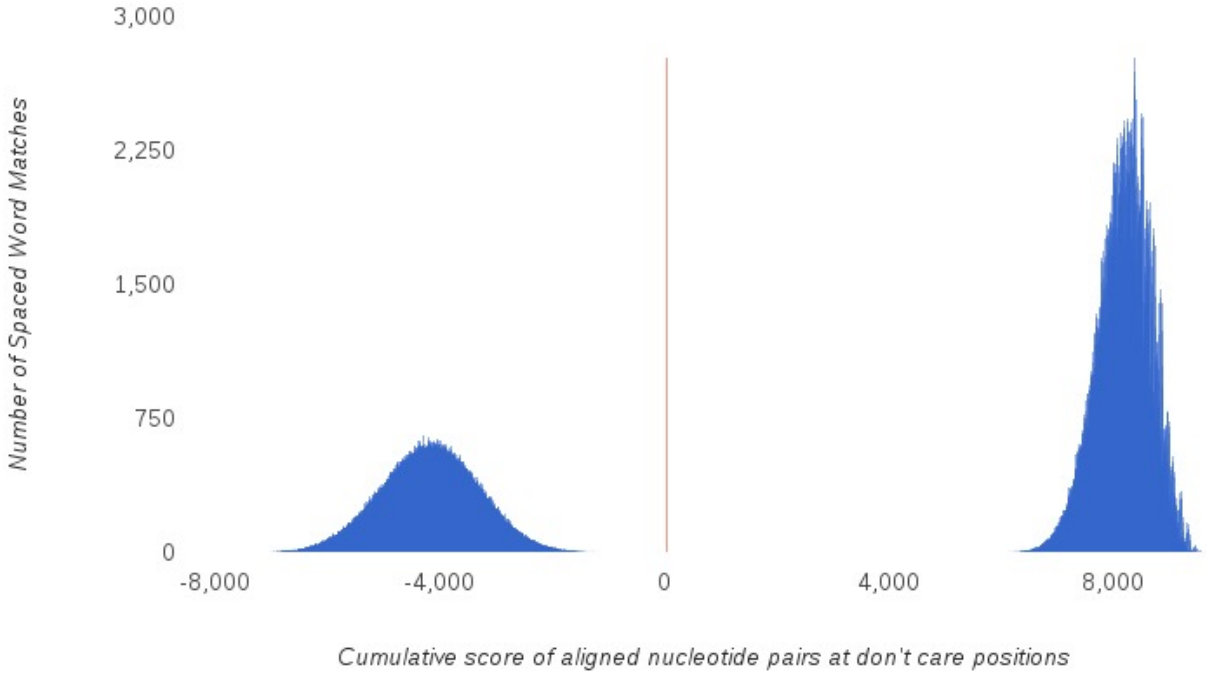


Figure: *i.i.d* sequences, 0.1 subst. per site, indel-free, 5 *Mb*

Spaced-word histograms

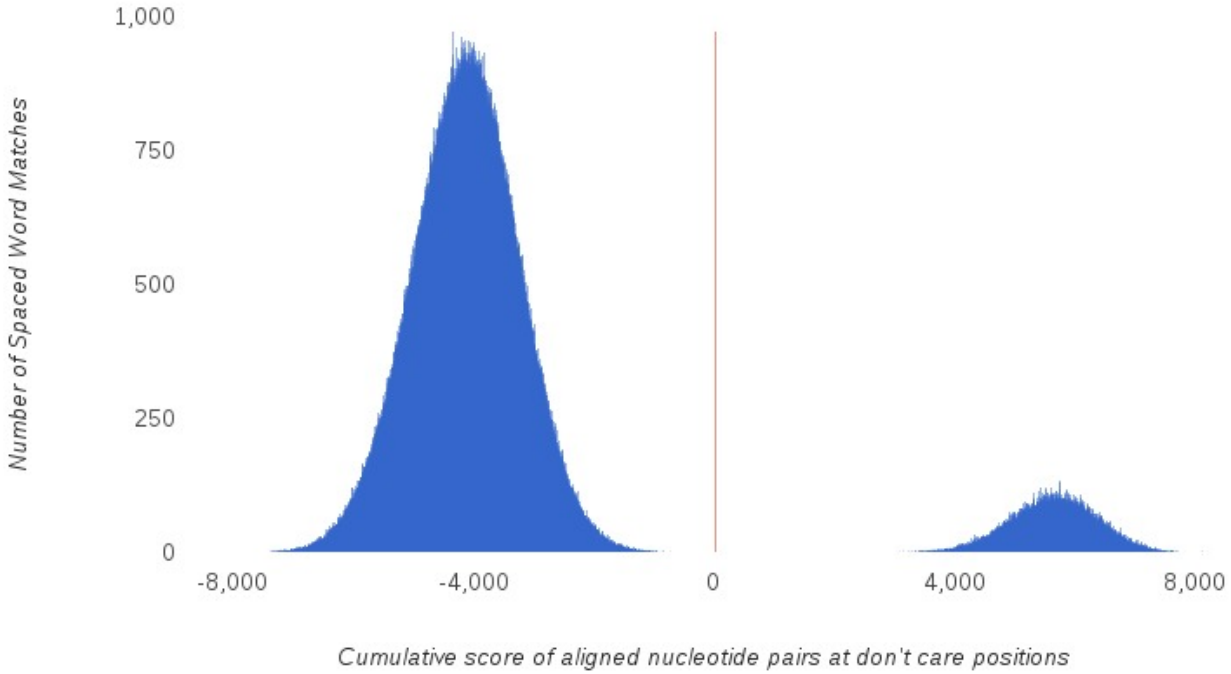


Figure: *i.i.d* sequences, 0.3 subst. per site, indel-free, 5 *Mb*

Spaced-word histograms

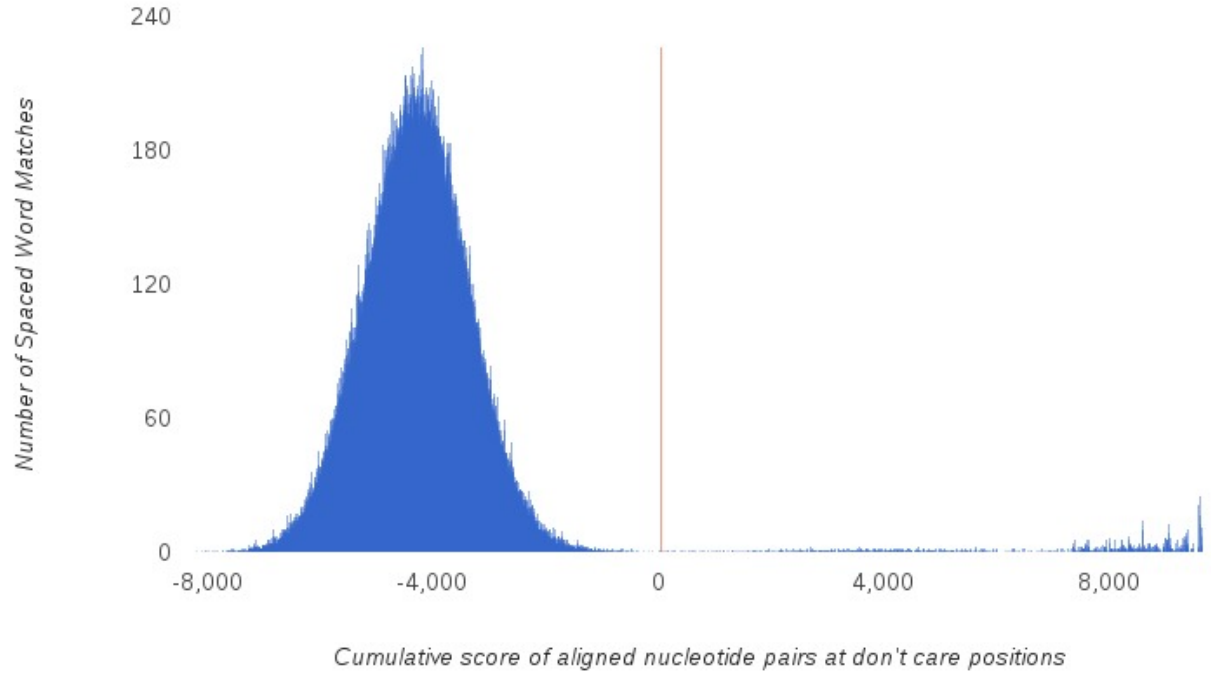


Figure: *Sagittula stellata* E37 vs *Rhodobacterales bacterium* HTCC2255.

Spaced-word histograms

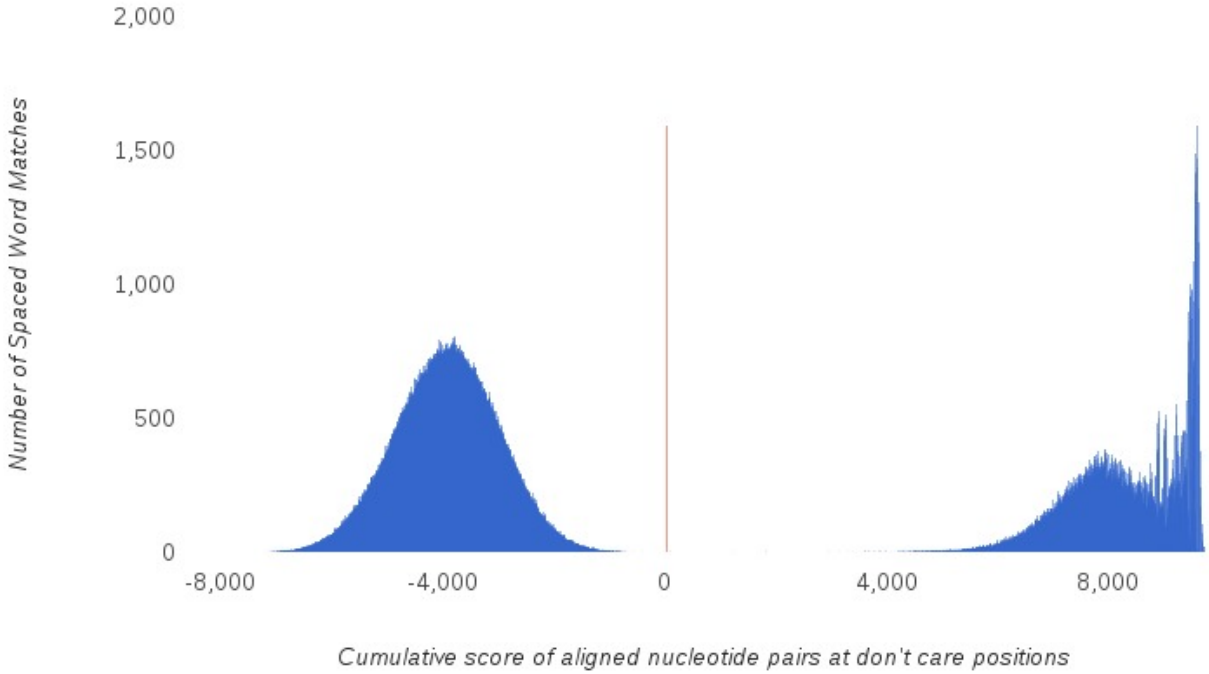


Figure: *Octadecabacter arcticus* 238 vs *Octadecabacter antarticus* 307.

Spaced-word histograms

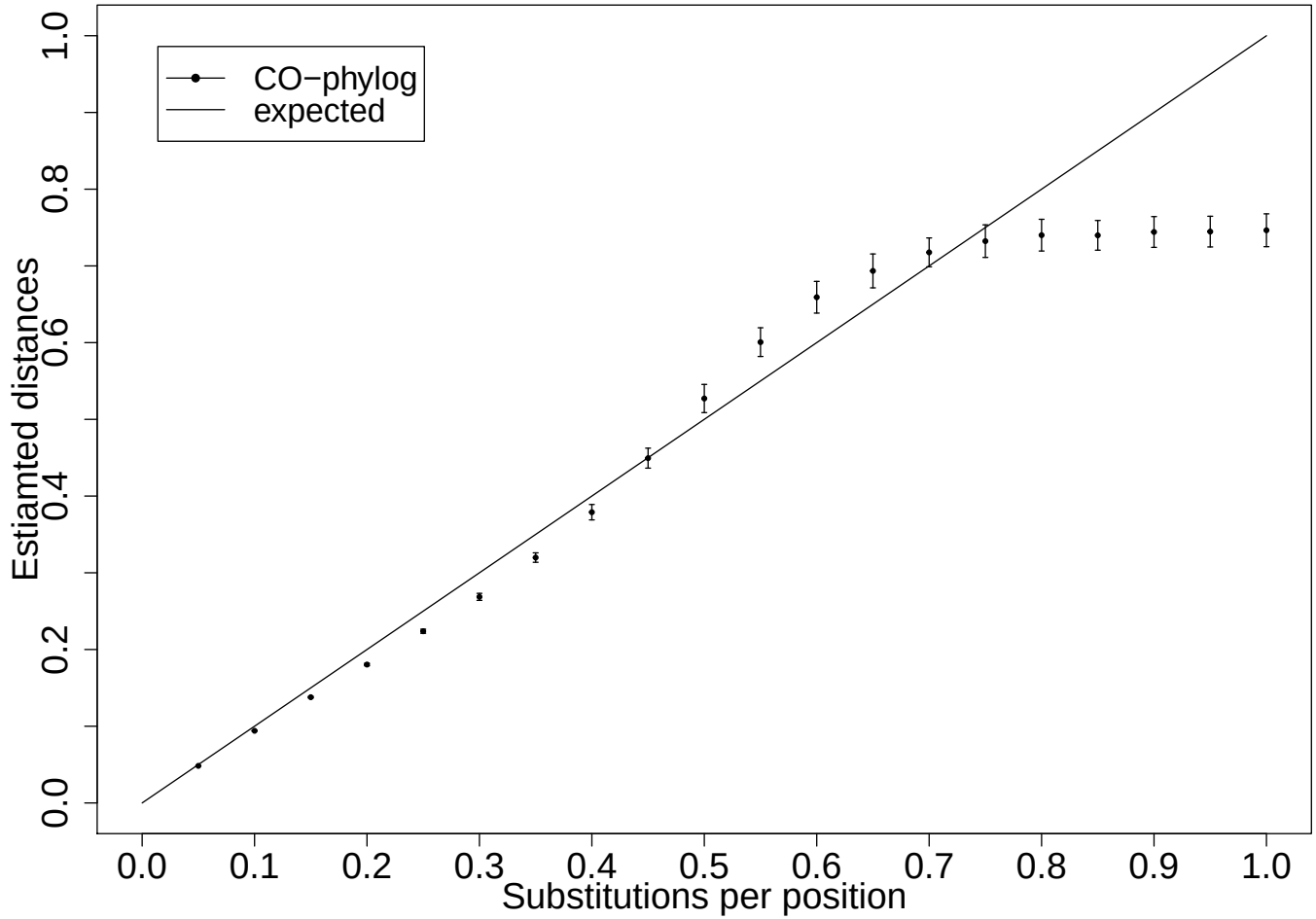
Accurate phylogeny reconstruction based on ‘filtered spaced word matches’ (*FSWM*)

Program evaluation (1): distances

Generate pairs of semi-artificial genome sequences:

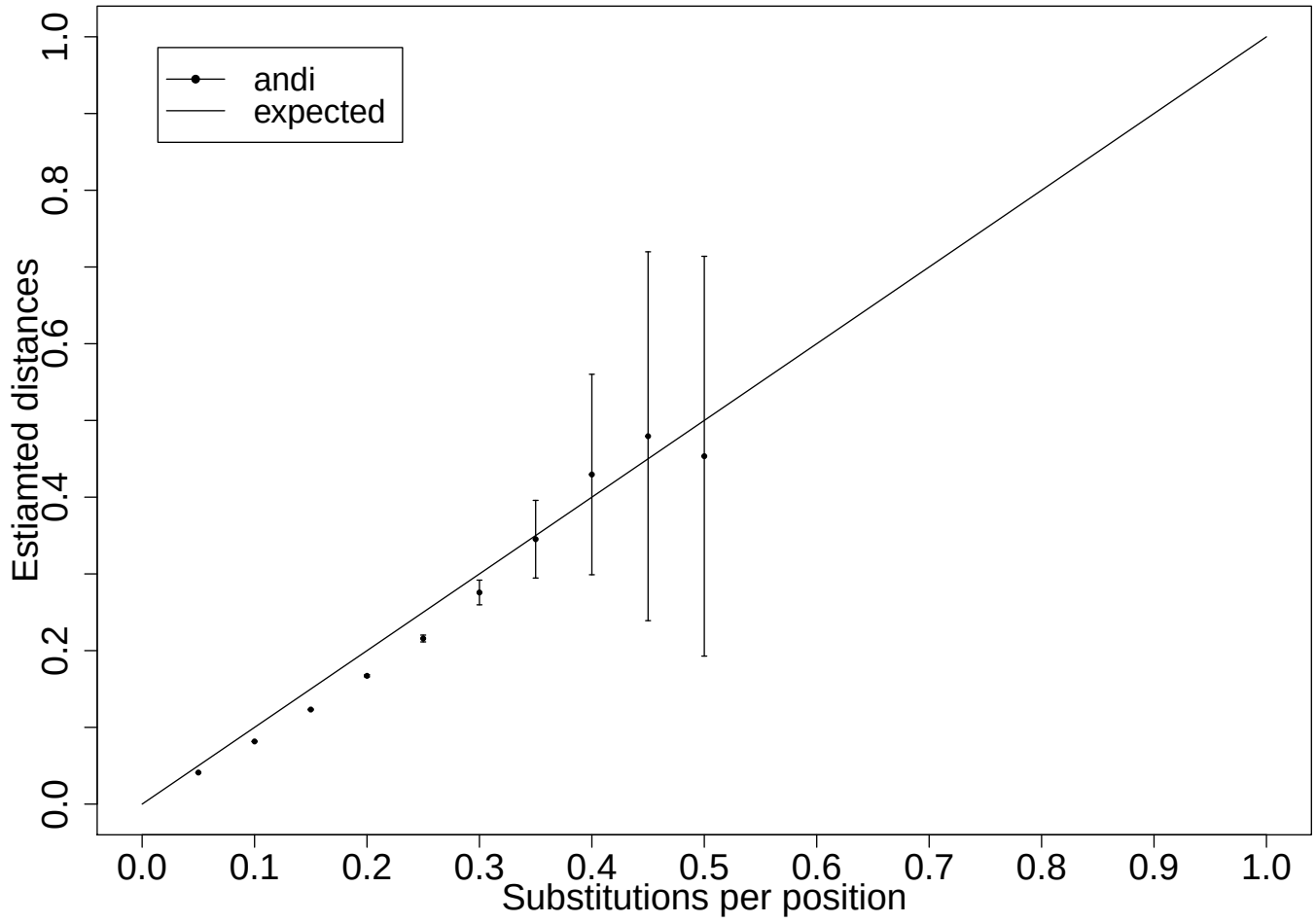
- *E. coli K12* as ‘ancestral’ genome
- Generate substitutions and indels for pairs of ‘descendent’ genomes – between 0 and 1 substitutions per position
- Compare estimated distances to ‘real’ distances

Program evaluation (1): distances



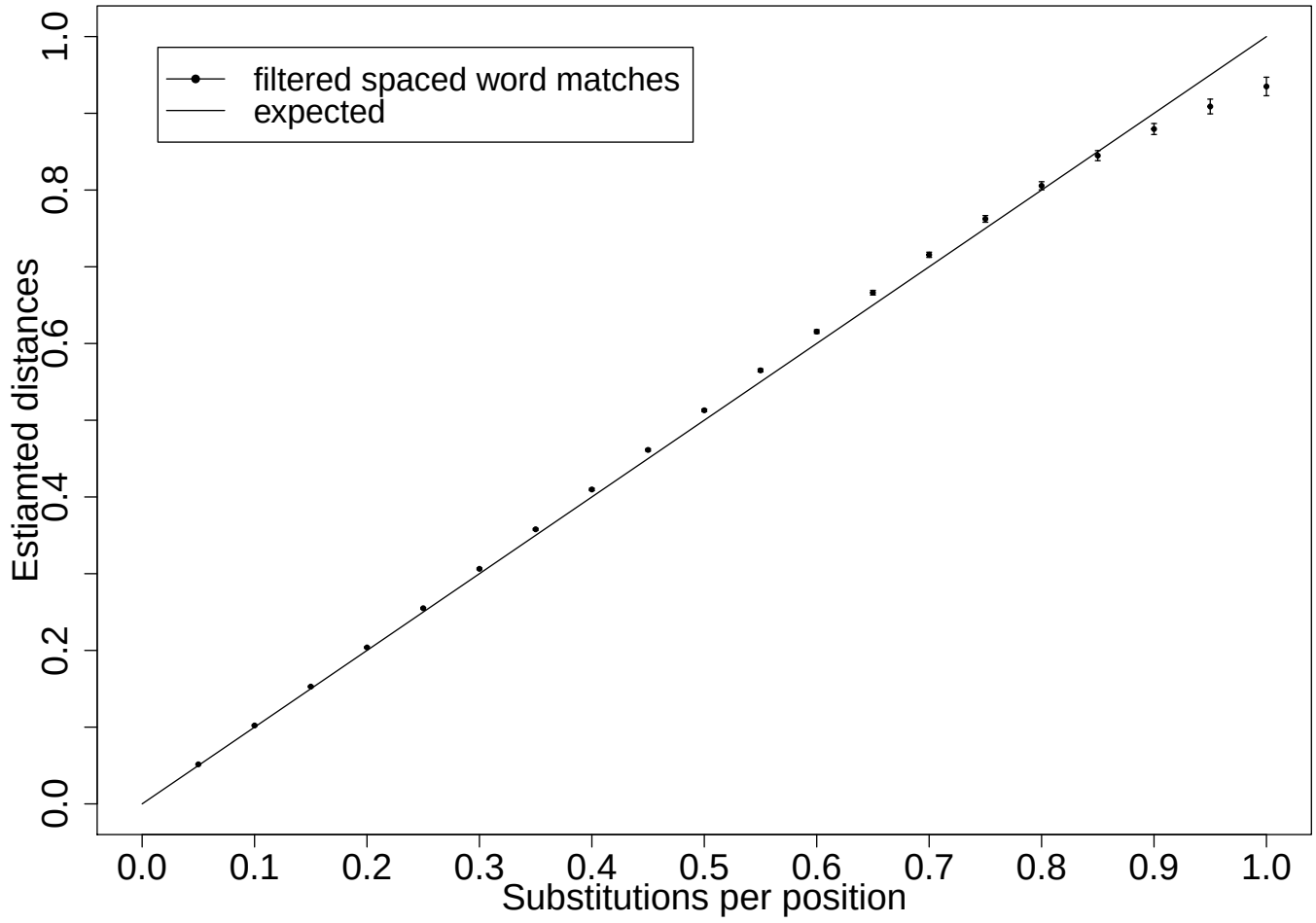
Co-phylog

Program evaluation (1): distances



andi

Program evaluation (1): distances



FSWM

Program evaluation (2): trees

Real-world benchmark data: 14 plant genomes (*Brassicales*)

Total size 4.8 Gb, up to 0.63 substitutions per site.

- No reasonable results with *andi*, distance too large
- *Co-phylog* did not finish

Program evaluation (2): trees

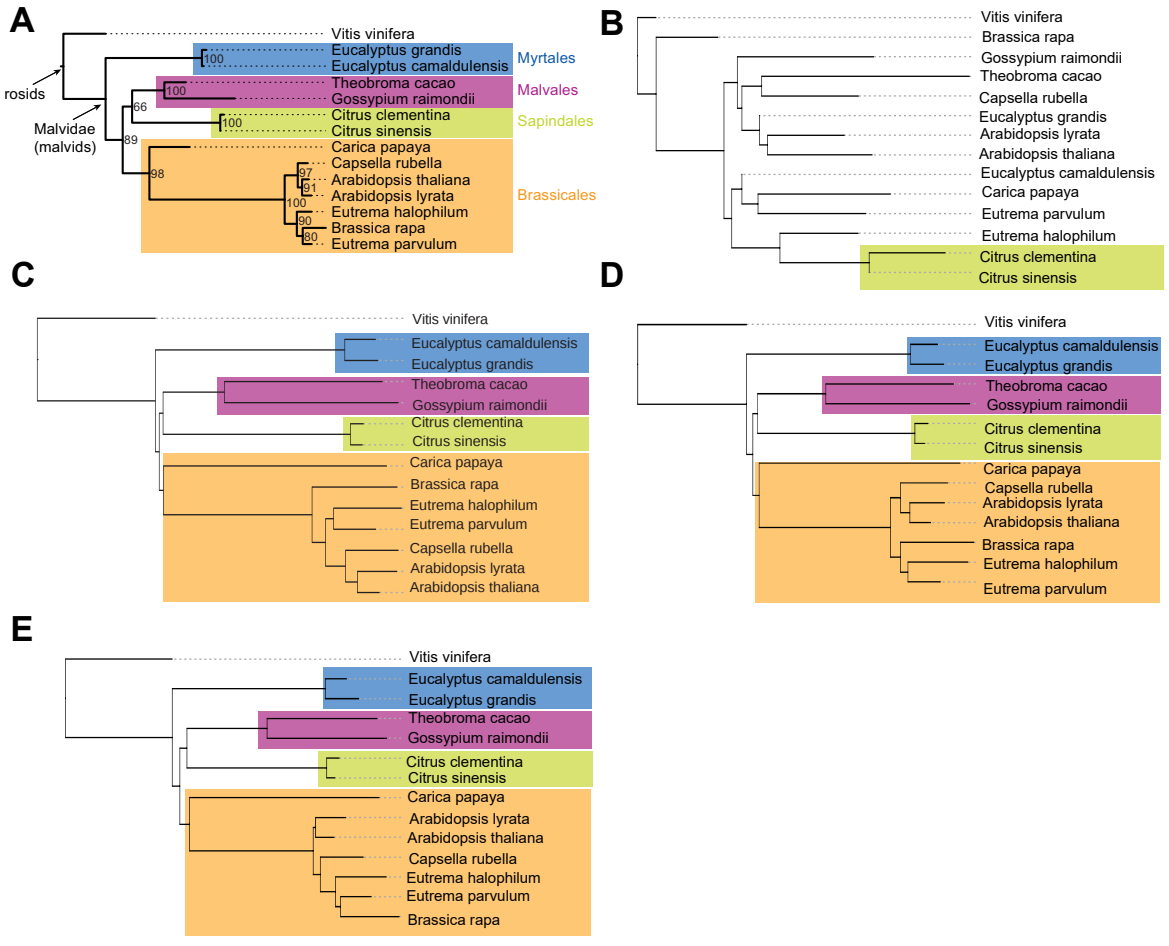


Figure: **A:** Reference tree (protein MSA, Likelihood), **B:** *andi*, **C-E:** FSWM with weight $w = 12, 13, 14$.

New projects

- Application to protein sequences
- Application to *metagenomics*
- Detecting *horizontal gene transfer*
- Application to database searching
- Heterogeneous substitution models
- Application to single reads