Refresher: BLOSUM – Brief Overview

- Based purely on counts and alignment
- BLOSUM65 < BLOSUM45 in evolutionary distance

a set of sequences $S = S_1 \ldots S_k$, where $S_i = s_{i1} \ldots s_{in}$,

s_{ij} is the j-th amino acid in sequence S_i.

The probability of observing each amino acid X =p(X)

$$p(X) = \sum_{i=1}^{k} c(X_j, S_i) / \sum_{j=1}^{20} \sum_{i=1}^{k} c(X_j, S_i)$$

where $c(X_j, S_i)$ is the count of amino acid X_j in sequence S_i

$$P(X_l, X_m \mid random) = 2p(X_l).p(X_m) \; or \; p(X_l)^2, if \;\; l = m$$

Refresher: In general ...log-odds for alignment

$P(X_l, X_m \mid data) = \#$ of X_l, X_m combinations/all possible pairwise combinations

finally take the log2-odds likelihood ratio and multiply by 2 for easy representation

Note: all possible pairwise combinations=k.$\{n(n-1)/2\}$

Blosum does use pseudo count to accomodate for combinations not observed:

add 1 in numerator, add 210 (20*19+20) in denominator

ie $P(\mathrm{X}_\mathrm{l}, \mathrm{X}_\mathrm{m} \mid data) = \dfrac{count(X_l, X_m)}{k.\{n(n-1)/2\} + 210}$

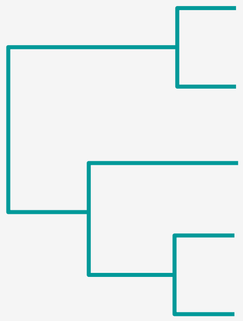
$$\lambda.\log \frac{P(\mathrm{X}_\mathrm{l}, \mathrm{X}_\mathrm{m} \mid data)}{P(\mathrm{X}_\mathrm{l}, \mathrm{X}_\mathrm{m} \mid random)} = subs.matrix$$

λ being a scaling factor for representation

What is phylogeny?

- The inference of evolutionary relationships
- The inference of putative common ancestors
- Trees (with branches and leaves!)

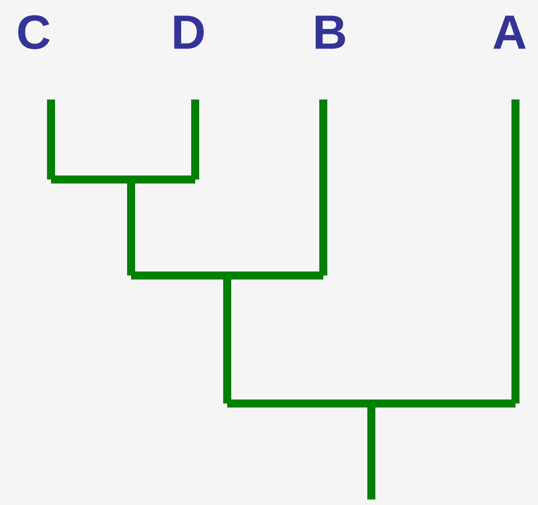
Example:



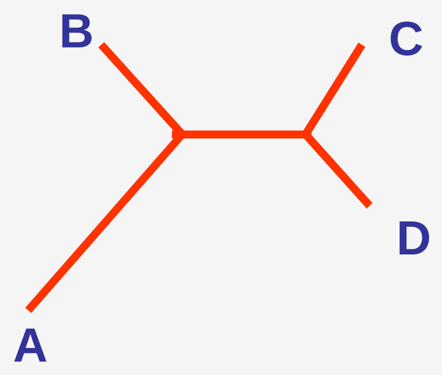
Orc:	ACAGTGACGCCCCAAACGT
Elf:	ACAGTGACGCTACAAACGT
Dwarf:	CCTGTGACGTAACAAACGA
Hobbit:	CCTGTGACGTAGCAAACGA
Human:	CCTGTGACGTAGCAAACGA

Rooted vs Unrooted tree

Rooted tree

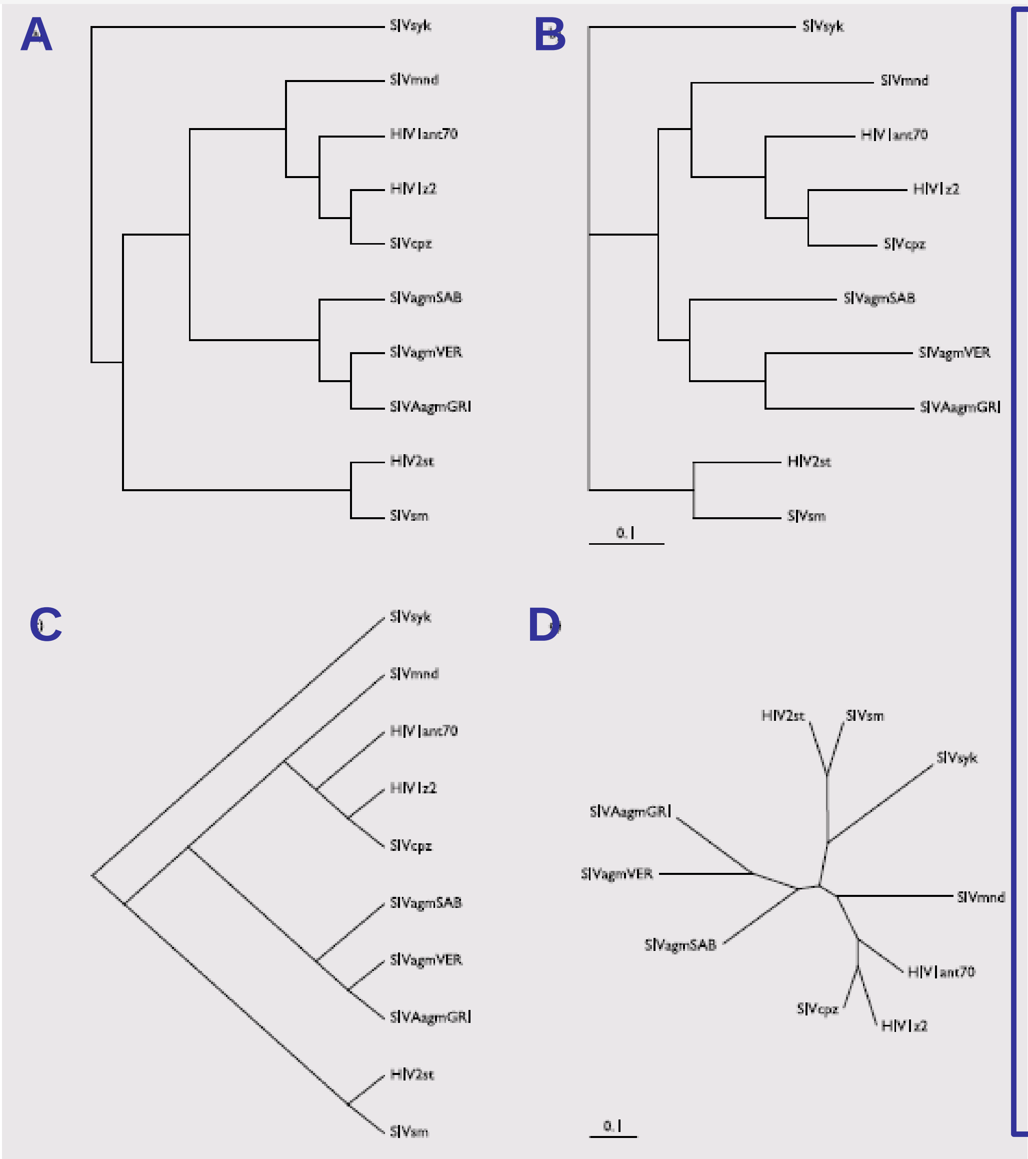


Unrooted tree



OTUs – Operational taxonomic units (leaves)
HTUs – Hypothetical taxonomic units (internal nodes)

Types of dendograms – diagramatic representation of phylogenetic trees



A: phenogram (Overall similarity based)

Important: grouping of OTUs
Meaningless: Vertical separation and branch lengths

B: a phylogram (similarity based)

important : Branch length and grouping

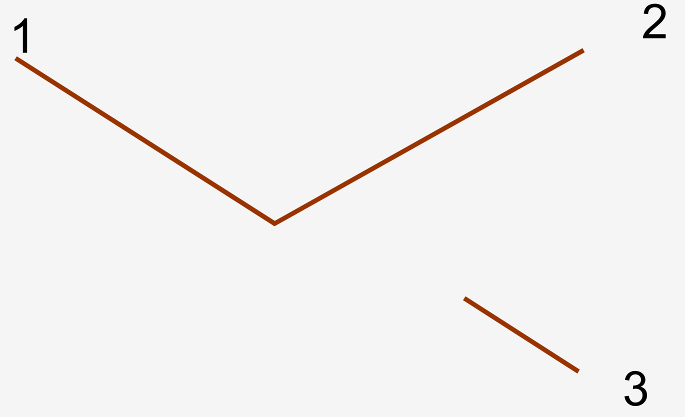
C: a cladogram (rooted)

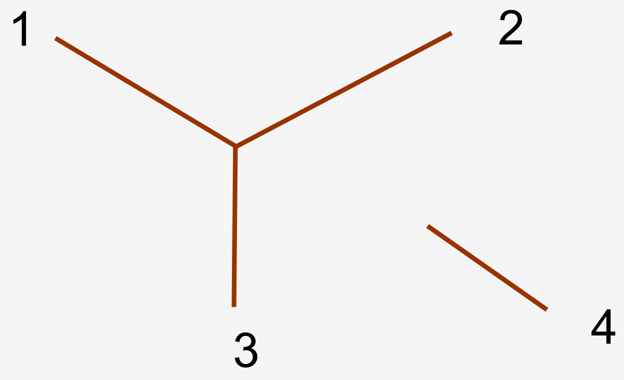
Important: grouping
Meaningless: Angles, lengths

D: a radial phylogram

Same as B

Calculating the number of unrooted trees





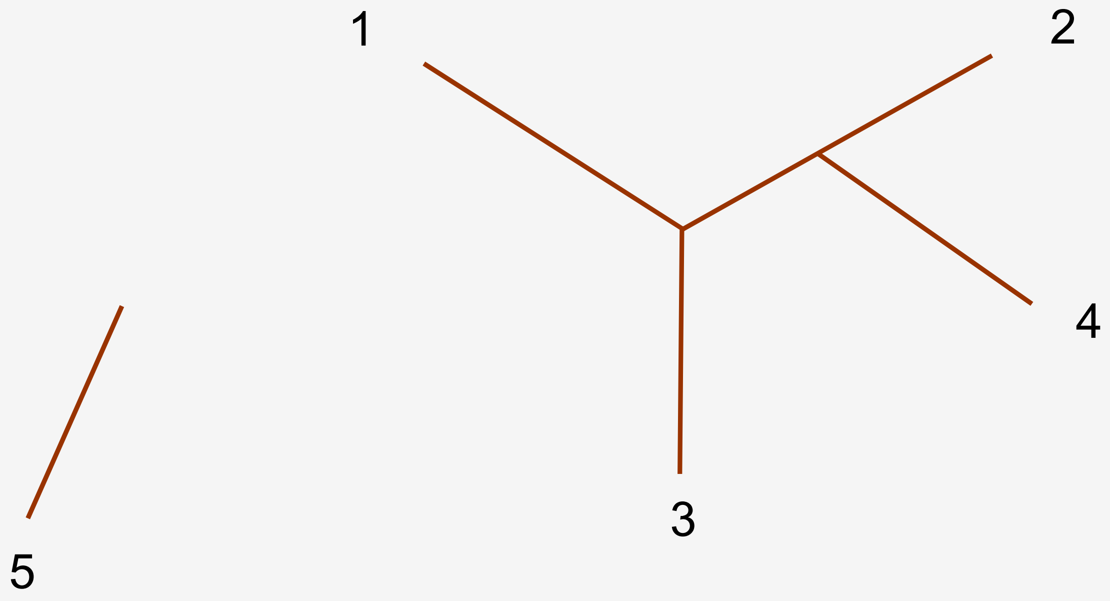
How many possibilities are there for adding leaf 3?

- 1

How many possibilities are there for adding leaf 4?

- 3

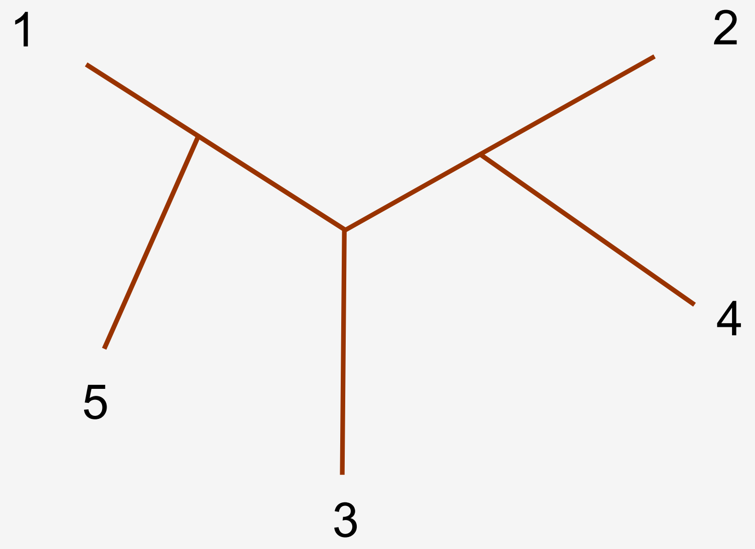
Calculating the number of unrooted trees



- How many possibilities are there for leaf 5? For the 5th leaf, there are 5 possibilities
- *ie* to add the nth leaf, we can add at the immediate leaf branches+ the number of internal branches

ie (n-1) “terminal-branches” + (n-4) “internal branches”=2n-5

Total number of trees?



N = 10

#unrooted: 2,027,025
#rooted: 34,459,425

N = 30
#unrooted: 8.7×10^{36}
#rooted: 4.95×10^{38}

- #unrooted trees for *n* taxa: (2n-5)*(2n-7)*...*3*1 = (2n-5)! / [2n-3*(n-3)!]
- #rooted trees for *n* taxa: (2n-3)*(2n-5)*(2n-7)*...*3 = (2n-3)! / [2n-2*(n-2)!]

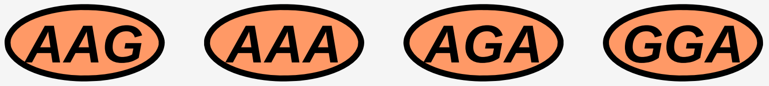
Commonly represented as 2n-5!! or 2n-3!!

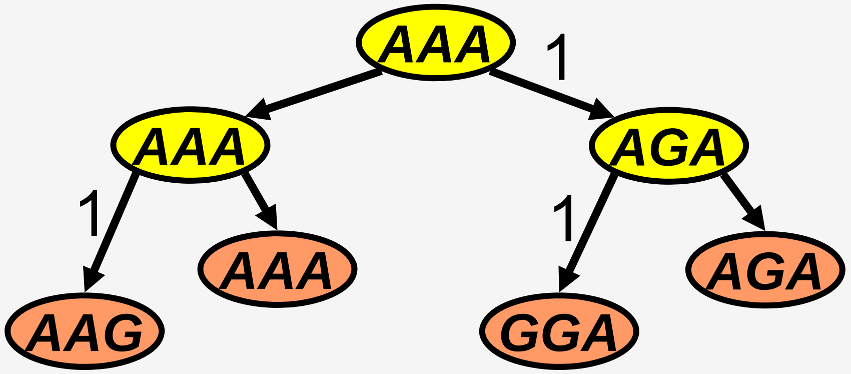
Parsimony Approach to Evolutionary Tree Reconstruction

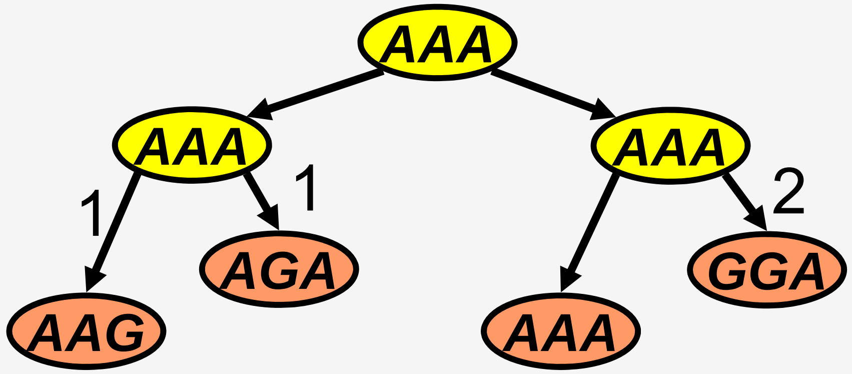
Parsimony ≡ frugality, “ less is better”

- Assumes observed character differences resulted from the fewest possible mutations
- Seeks the tree that yields lowest possible **parsimony score -** sum of cost of all mutations found in the tree

Example for calculating parsimony score







Total #substitutions = 3

Total #substitutions = 4

The left tree is preferred over the right tree.

The total number of changes is called the **parsimony score**.

Parsimony Based Tree Reconstruction

1. A procedure to find the minimum number of changes needed to explain the data for a given tree topology, where species are assigned to leaves.
2. A search through the space of trees.
3. Efficient algorithms for (1). (2) is hard, we can use heuristic approaches.

Sankoff algorithm to count evolutionary changes in a given tree

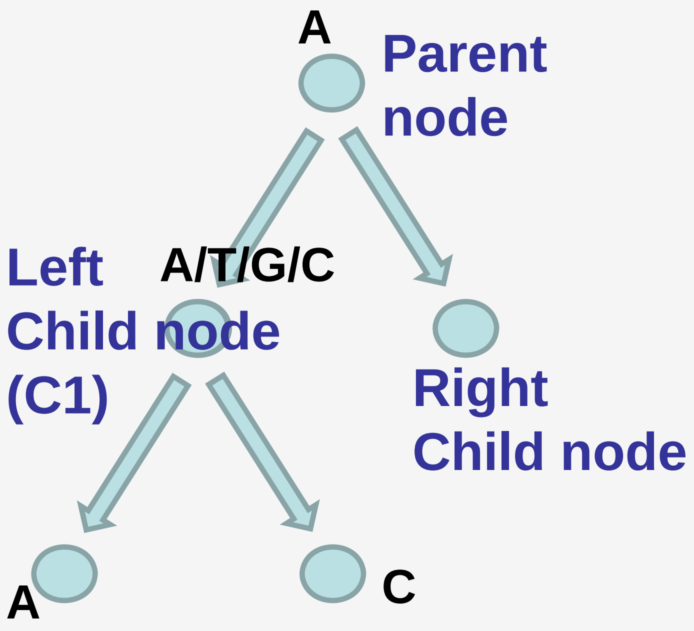
- Step-1: Define a cost matrix [c_{ij}], representing changes from character state *i* to state *j*

	A	**C**	**G**	**T**
A	0	2.5	1	2.5
C	2.5	0	2.5	1
G	1	2.5	0	2.5
T	2.5	1	2.5	0

- Step-2: Starting from the leaves, we work down at each node, k to calculate a score $S_k(i)$ for each state, *i (i= 1.. 4 for DNA)*
    - *S_k is the minimal cost of events for all subtrees above that node, k*

Sankoff algo – cont'd

- Remember for each internal node ("parent") we have two sub-nodes ("children")
- So the scores need to be added up on both branches
- First, consider say the left branch, with left child node at state *i* and we want to evaluate the score for parent node at state A.
    - If S(left child node) is known for each states, the S(parent node@j) will be

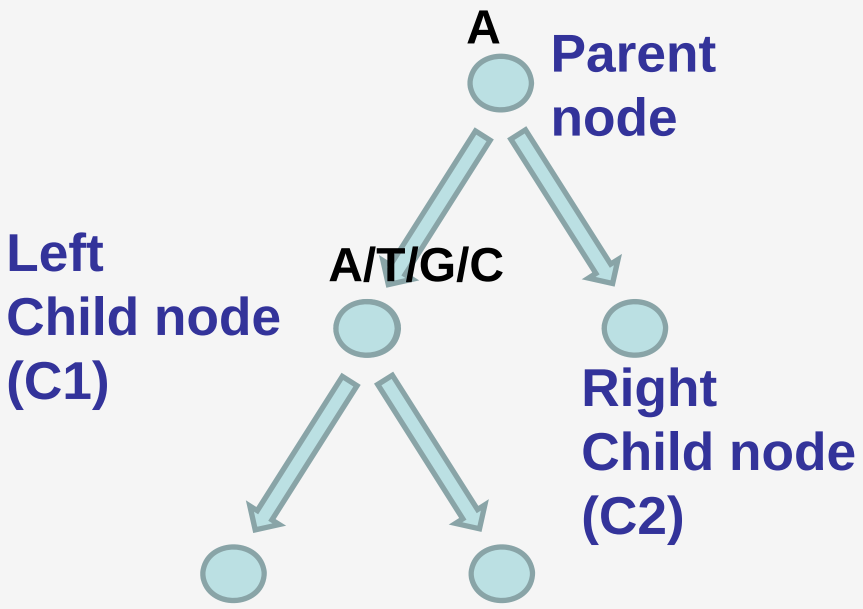


$$S_p(A)\,|_{LeftArm} = \min \begin{Bmatrix} c_{A\to A} + S_{C1}(A) \\ c_{A\to T} + S_{C1}(T) \\ c_{A\to G} + S_{C1}(G) \\ c_{A\to C} + S_{C1}(C) \end{Bmatrix}$$

Sankoff algo – cont'd

$$S_p(A)\,|_{LeftBranch} = \min\begin{Bmatrix} c_{A\to A} + S_{C1}(A) \\ c_{A\to T} + S_{C1}(T) \\ c_{A\to G} + S_{C1}(G) \\ c_{A\to C} + S_{C1}(C) \end{Bmatrix}$$

$$= \underbrace{\min_{j=A/T/G/C}\left\{c_{A\to j} + S_{C1}(j)\right\}}_{LeftBranch}$$

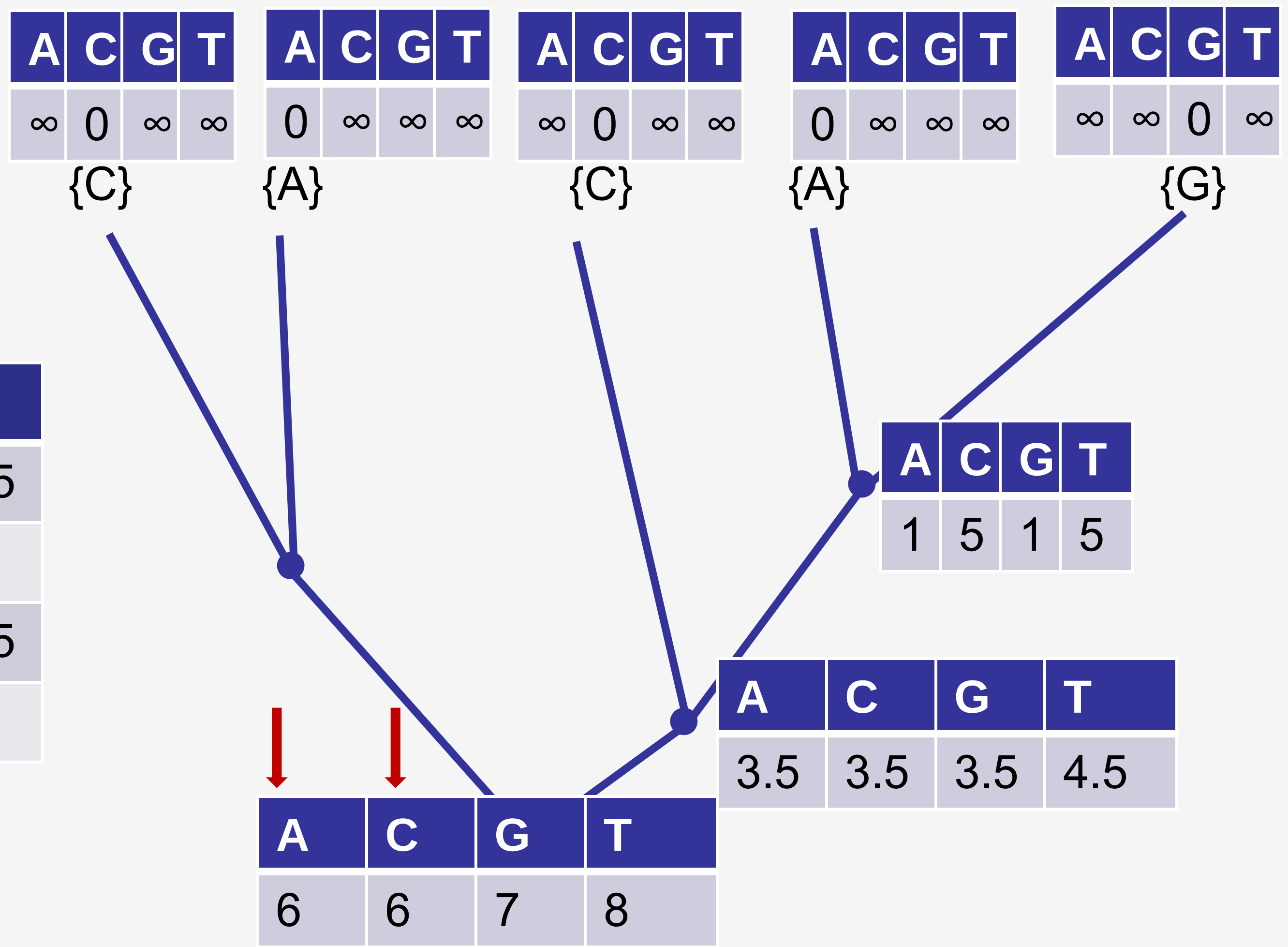


$$S_p(i) = S_p\,'(i)_{left} + S_p\,'(i)_{right}$$

$$= \underbrace{\min_{j=A/T/G/C}\left\{c_{i\to j} + S_{C1}(j)\right\}}_{LeftBranch} + \underbrace{\min_{k=A/T/G/C}\left\{c_{i\to k} + S_{C2}(k)\right\}}_{RightBranch}$$

Sankoff algorithm – Example

$$S_p(i) = \underbrace{\min_{j=A/T/G/C}\left\{c_{i\to j} + S_{C1}(j)\right\}}_{LeftArm} + \underbrace{\min_{k=A/T/G/C}\left\{c_{i\to k} + S_{C2}(k)\right\}}_{RightArm}$$



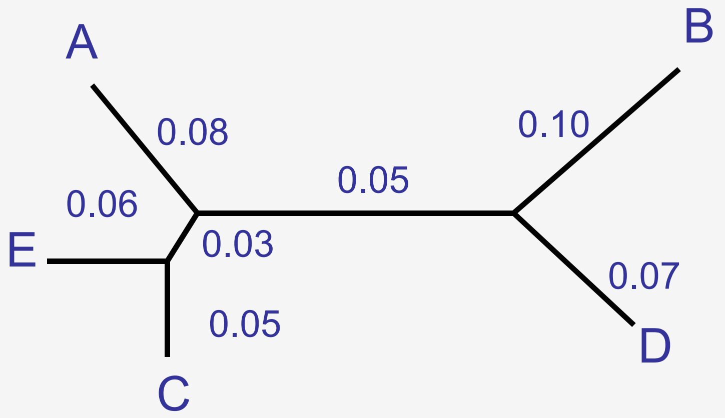
Cost matrix

	A	C	G	T
A	0	2.5	1	2.5
C	2.5	0	2.5	1
G	1	2.5	0	2.5
T	2.5	1	2.5	0

Assigning branch lengths using Least squares

We have an observed matrix of Distances between sequences from all possible pair-wise comparisons – Remember D and K in JC model?

	A	B	C	D	E
A	0	0.23	0.16	0.20	0.17
B	0.23	0	0.23	0.17	0.24
C	0.16	0.23	0	0.15	0.11
D	0.20	0.17	0.15	0	0.21
E	0.17	0.24	0.11	0.21	0



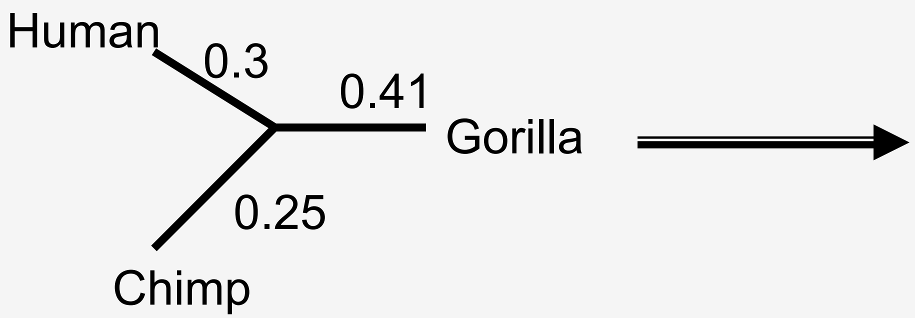
Observed distances denoted by D_{ij} and the “real branch” lengths to be predicted as d_{ij}

$$Q(T) = \sum_{i=1}^{n} \sum_{j=1;\, j \neq i}^{n} w_{ij}(D_{ij} - d_{ij})^2$$

$$w_{ij} = \text{Method Dependent Weight}$$

Motivation: From a distance table to a tree

Each tree has branch lengths from which "predicted" set of distances can be computed: d(i,j) (small d, denotes the distance of the branches, unlike the observed pairwise distances D).



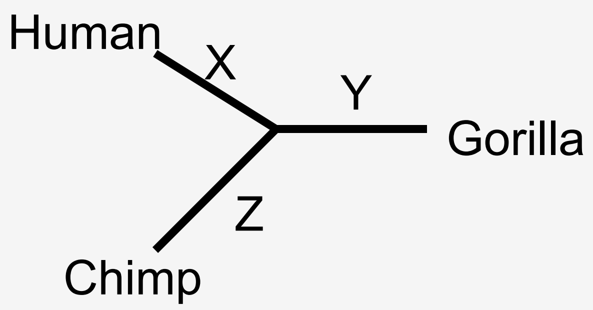
d(Human,Chimp) = 0.55

d(Human,Gorilla) = 0.71

d(Chimp, Gorilla) = 0.66

Motivation: From a distance table to a tree

The question is can we find branch lengths, so that the d's are equal to the D's?



D(Human,Chimp) = 0.3 =X + Z

D(Human,Gorilla) = 0.4 = X + Y

D(Chimp, Gorilla) = 0.5 = Y + Z

X+Z = 0.3

X+Y = 0.4

Y+Z = 0.5

Y-Z = 0.1

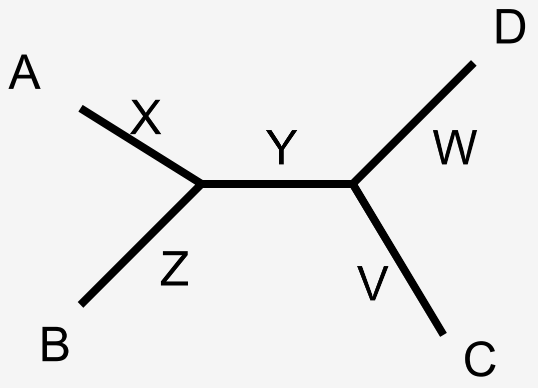
Y+Z = 0.5

Y = 0.3

Z = 0.2

X = 0.1

Is there always a solution??



$\Longrightarrow$

5 Variables,

6 Equations,

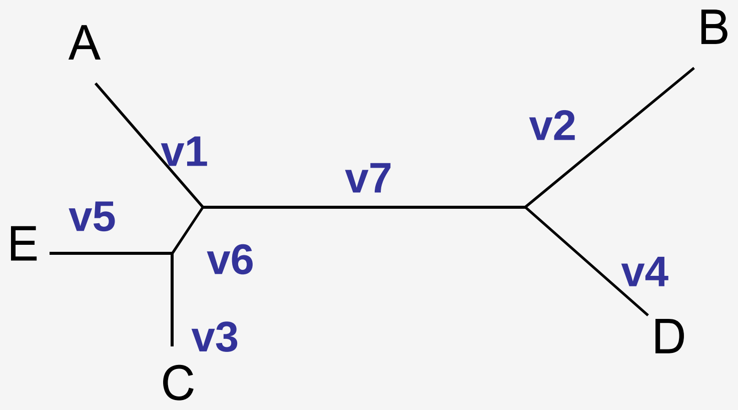
It might be that there's no solution

$$NC_2 > 2N - 3 \text{ for N} > 3$$

Least squares – Cont'd

$$d_{ij} = \sum_k x_{ij,k} v_k$$

introduce an indicator variable $x_{ij,k}$, which is 1 if branch v_k lies in the path from species i to species j and 0 otherwise



$$d_{12} = 1v_1 + 1v_2 + 0v_3 + 0v_4 + 0v_5 + 0v_6 + 1v_7$$

$$d_{13} = 1v_1 + 0v_2 + 1v_3 + 0v_4 + 0v_5 + 1v_6 + 0v_7$$

$\ldots$

$$d_{45} = 0v_1 + 0v_2 + 0v_3 + 1v_4 + 1v_5 + 1v_6 + 1v_7$$

LS- Cont'd

when the weights are 1.0 $Q(T) = \sum_{i=1}^{n} \sum_{j=1;\, j \neq i}^{n} (D_{ij} - d_{ij})^2$

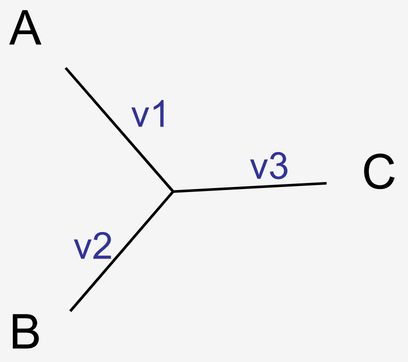
$$= \sum_{i=1}^{n} \sum_{j=1;\, j \neq i}^{n} (D_{ij} - \sum_{k} x_{ijk} v_k)^2 \Rightarrow \frac{dQ}{dv_k} = -2 \sum_{i=1}^{n} \sum_{j=1;;\, j \neq i}^{n} x_{ij,k} (D_{ij} - \sum_{k} x_{ij,k} v_k) = 0$$

$$X^T D = (X^T X) v$$

$$v = (X^T X)^{-1} X^T D$$

X
No of rows=n(n-1)/2
No of columns=2n-3 (eq to k)

LS - example



	A	B	C
A	0	10	12
B	10	0	8
C	12	8	0

$$X = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \qquad X^T X = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \qquad (X^T X)^{-1} = \begin{pmatrix} \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} \end{pmatrix}$$

$$D = \begin{pmatrix} 10 \\ 12 \\ 8 \end{pmatrix}$$

$$v = (X^T X)^{-1} X^T D = \ldots$$

When we have weighted LS, then previous equations can be written:

$$X^T W D = (X^T W X) v$$

$$v = (X^T W X)^{-1} X^T W D$$

where W is a diagonal matrix with distance weights on main diagonal.

Evaluating a tree is good but how to build the tree? – Fast clustering-based algorithm for tree construction

UPGMA (unweighted pair group method using arithmetic averages)

Given two disjoint clusters C_i, C_j of sequences,

$$d_{ij} = \frac{1}{|C_i| \times |C_j|} \Sigma_{\{p \in Ci,\ q \in Cj\}} d_{pq}$$

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

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

$$D((ij),l) = (\frac{n(i)}{n(i)+n(j)})D(i,l) + (\frac{n(j)}{n(i)+n(j)})D(j,l)$$

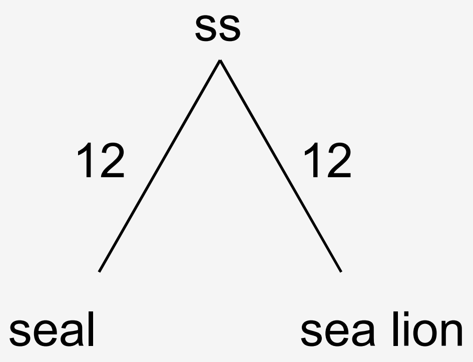
UPGMA algorithm

- Find *i* and *j* with smallest D_{ij}
- Create new group X by joining nodes *i* & *j*
- Compute distances between X (new member) and others (old)
- Place node X at $D_{ij}/2$
- Delete i and j; replace with X in D-matrix

The distance table

	dog	bear	raccon	weasel	seal	sea lion	cat	chimp
dog	0	32	48	51	50	48	98	148
bear		0	26	34	29	33	84	136
raccon			0	42	44	44	92	152
weasel				0	44	38	86	142
seal					0	24	89	142
sea lion						0	90	142
cat							0	148
chimp								0

Distance between these two taxa was 24, so each branch has a length of 12.



We call the parent node of seal and sea lion “ss”.

Removing the seal and sea-lion rows and columns, and adding the ss row and columns

	dog	**bear**	**raccon**	**weasel**	**ss**	**cat**	**chimp**
dog	0	32	48	51	?	98	148
bear		0	26	34	?	84	136
raccon			0	42	?	92	152
weasel				0	?	86	142
ss					0	89	142
cat						0	148
chimp							0

Computing dog-ss distance

	dog	bear	raccon	weasel	seal	sea lion	cat	chimp
dog	0	32	48	51	50	48	98	148

$$D((ij),k) = (\frac{n(i)}{n(i)+n(j)})D(i,k) + (\frac{n(j)}{n(i)+n(j)})D(j,k)$$

Here, i=seal, j=sea lion, k = dog.

n(i)=n(j)=1.

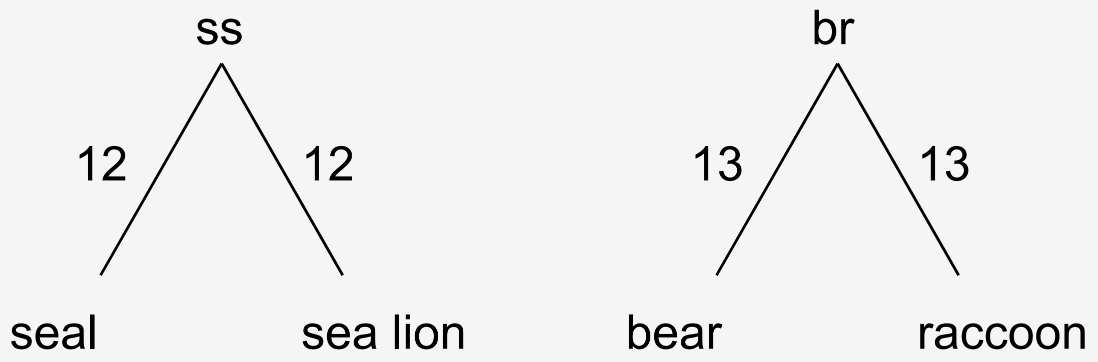
D(ss,dog) = 0.5D(sea lion,dog) + 0.5D(seal,dog) = 49.

The new table. Starting second iteration…

	dog	**bear**	**raccon**	**weasel**	**ss**	**cat**	**chimp**
dog	0	32	48	51	49	98	148
bear		0	26	34	31	84	136
raccon			0	42	44	92	152
weasel				0	41	86	142
ss					0	89	142
cat						0	148
chimp							0

Inferring tree

Distance between bear and raccoon was 26, so each branch has a length of 13.



We call the parent node of bear and raccoon “br”.

Computing br-ss distance

	dog	bear	raccon	weasel	ss	cat	chimp
ss	49	31	44	41	0	89.5	142

$$D((ij),k) = \left(\frac{n(i)}{n(i)+n(j)}\right)D(i,k) + \left(\frac{n(j)}{n(i)+n(j)}\right)D(j,k)$$

Here, i=raccoon, j=bear, k = ss.

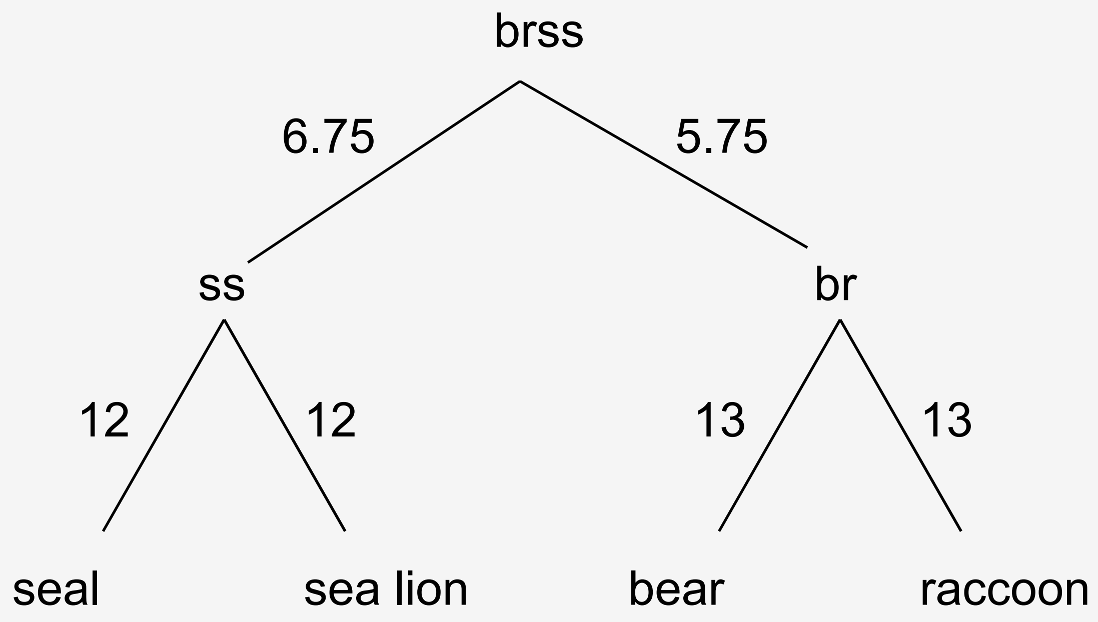
n(i)=n(j)=1. D(br,ss) = 0.5D(bear,ss)+0.5D(raccoon,ss)=37.5.

The new table. Starting next iteration…

	dog	br	weasel	ss	cat	chimp
dog	0	40	51	49	98	148
br		0	38	37.5	88	144
weasel			0	41	86	142
ss				0	89	142
cat					0	148
chimp						0

Inferring tree

Distance between br and ss was 37.5, so each branch has a length of 18.75. But this is the distance from brss to the leaves. The distance brss to ss is 18.75-12=6.75. The distance between brss to br is 18.75-13=5.75

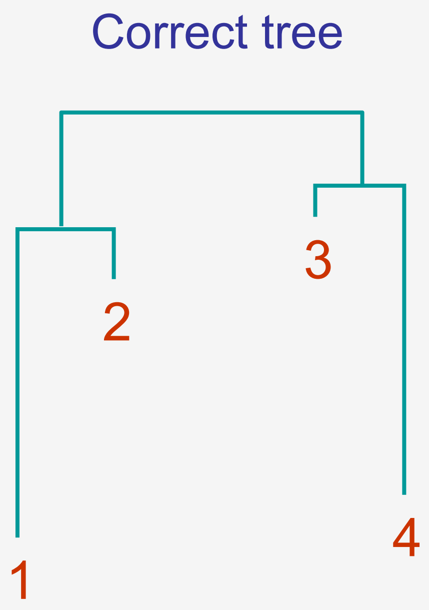


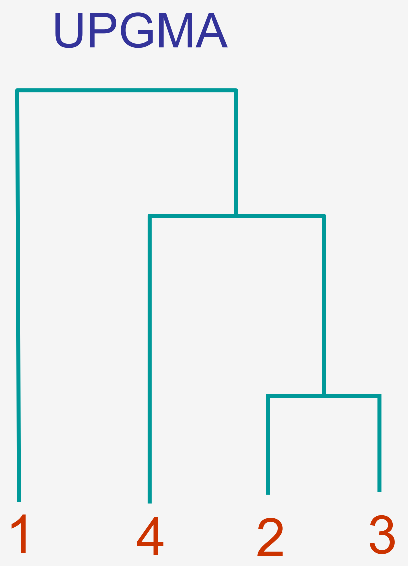
And so on…..

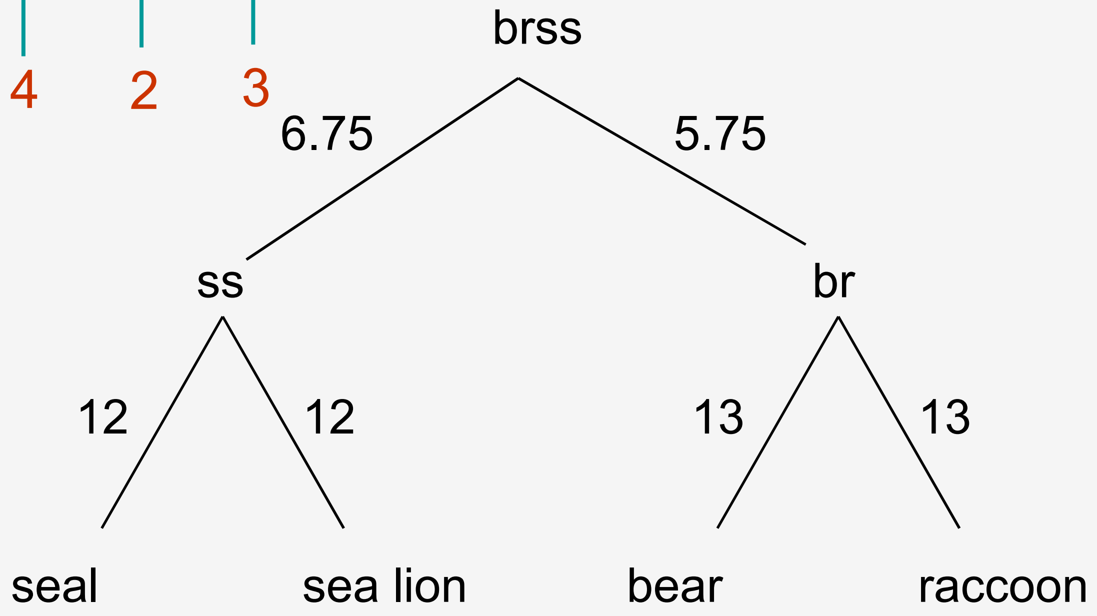
UPGMA’s Weakness

- The algorithm produces an **ultrametric** tree : the distance from the root to any leaf is the same
    - UPGMA assumes a constant molecular clock: all species represented by the leaves in the tree are assumed to accumulate mutations (and thus evolve) at the same rate. This is a major pitfall of UPGMA.

UPGMA’s Weakness: Example

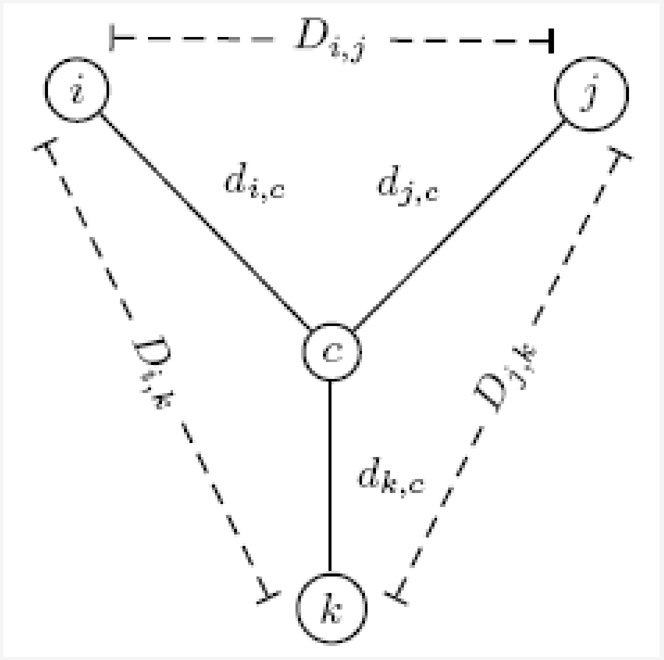






Clustering algorithm 2 – NJ (neighbor joining)

- Tree reconstruction for any 3x3 matrix is straightforward
- We have 3 leaves *i, j, k* and a center vertex *c*



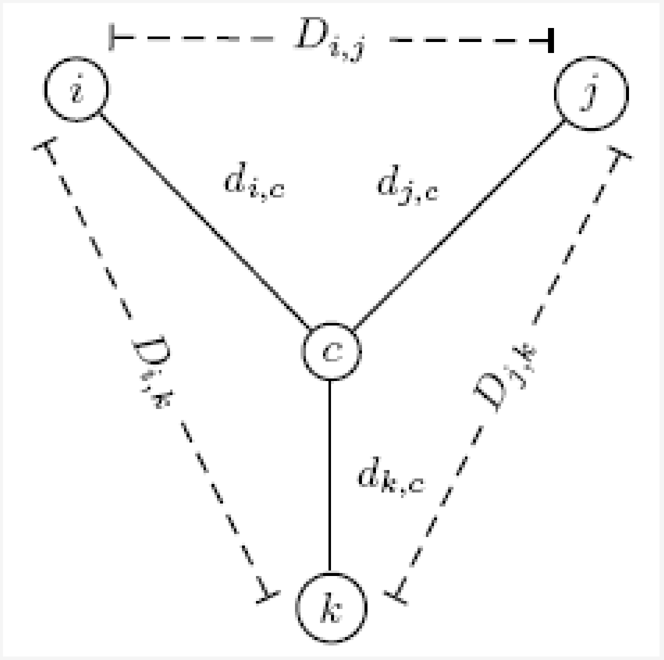
<u>Observe D’s, infer d’s</u>

$$d_{ic} + d_{jc} = D_{ij}$$

$$d_{ic} + d_{kc} = D_{ik}$$

$$d_{jc} + d_{kc} = D_{jk}$$

NJ –Cont'd



$$2d_{ic} + \quad D_{jk} \qquad = D_{ij} + D_{ik}$$

$$d_{ic} = (D_{ij} + D_{ik} – D_{jk})/2$$

Similarly,

$$d_{jc} = (D_{ij} + D_{jk} – D_{ik})/2$$
$$d_{kc} = (D_{ki} + D_{kj} – D_{ij})/2$$

Trees with > 3 Leaves – NJ Cont’d

- An unrooted tree with *n* leaves has *2n-3* branches

- This means fitting a given tree to a distance matrix *D* requires solving a system of “n choose 2” equations with *2n-3* variables

- This is not always easy to solve for *large n*

NJ algorithm

- For each tip compute $u_i = \sum_{j:j \neq i}^{n} D_{ij} / (n-2)$
- Choose i and j for which, $D_{ij} - u_i - u_j$ is smallest
- Join nodes i and j to X. Compute branch length fro i to X and j to X

$$v_{i \to X} = (D_{ij} + u_i - u_j)/2$$

$$v_{j \to X} = (D_{ij} + u_j - u_i)/2$$

- Compute the distance between X an remaining nodes

$$v_{X \to k} = (D_{ik} + D_{jk} - D_{ij})/2$$

NJ algorithm – Cont'd

- New node X is treated as a new tip and old nodes I, j are deleted
- If more than two nodes remain go back to step-1, else connect the two nodes (*l*,*m*) by $D_{l,m}$

Bootstrapping to get the best trees

Main outline of algorithm

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

Multiple sequence alignment using phylogenetic methods – Clustal-W

Pairwise alignment: calculation of distance matrix

↓

Rooted NJ tree (guide tree) and calculation of sequence weights

↓

Progressive alignment following the guide tree

```
human           ---MEEPQSDPSVEP-PLSQETFS 20
monkey          ---MEEPQSDPSIEP-PLSQETFS 20
mouse           MTAMEESQSDISLEL-PLSQETFS 23
rat             ---MEDSQSDMSIEL-PLSQETFS 20
xenopus         ---ME-PSSETGMDP-PLSQETES 19
chicken         ---MA-EEMEPLLEPTEVFMDLW- 19
                   *   . :  ::   :  :
```

Step 1-Calculation of Distance Matrix using pairwise alignment

Use the Distance Matrix to create a Guide Tree to determine the “order” of the sequences.

Hbb-Hu	**1**	**-**						
Hbb-Ho	**2**	**.17**	**-**					
Hba-Hu	**3**	**.59**	**.60**	**-**				
Hba-Ho	**4**	**.59**	**.59**	**.13**	**-**			
Myg-Ph	**5**	**.77**	**.77**	**.75**	**.75**	**-**		
Gib-Pe	**6**	**.81**	**.82**	**.73**	**.74**	**.80**	**-**	
Lgb-Lu	**7**	**.87**	**.86**	**.86**	**.88**	**.93**	**.90**	**-**
		1	**2**	**3**	**4**	**5**	**6**	**7**

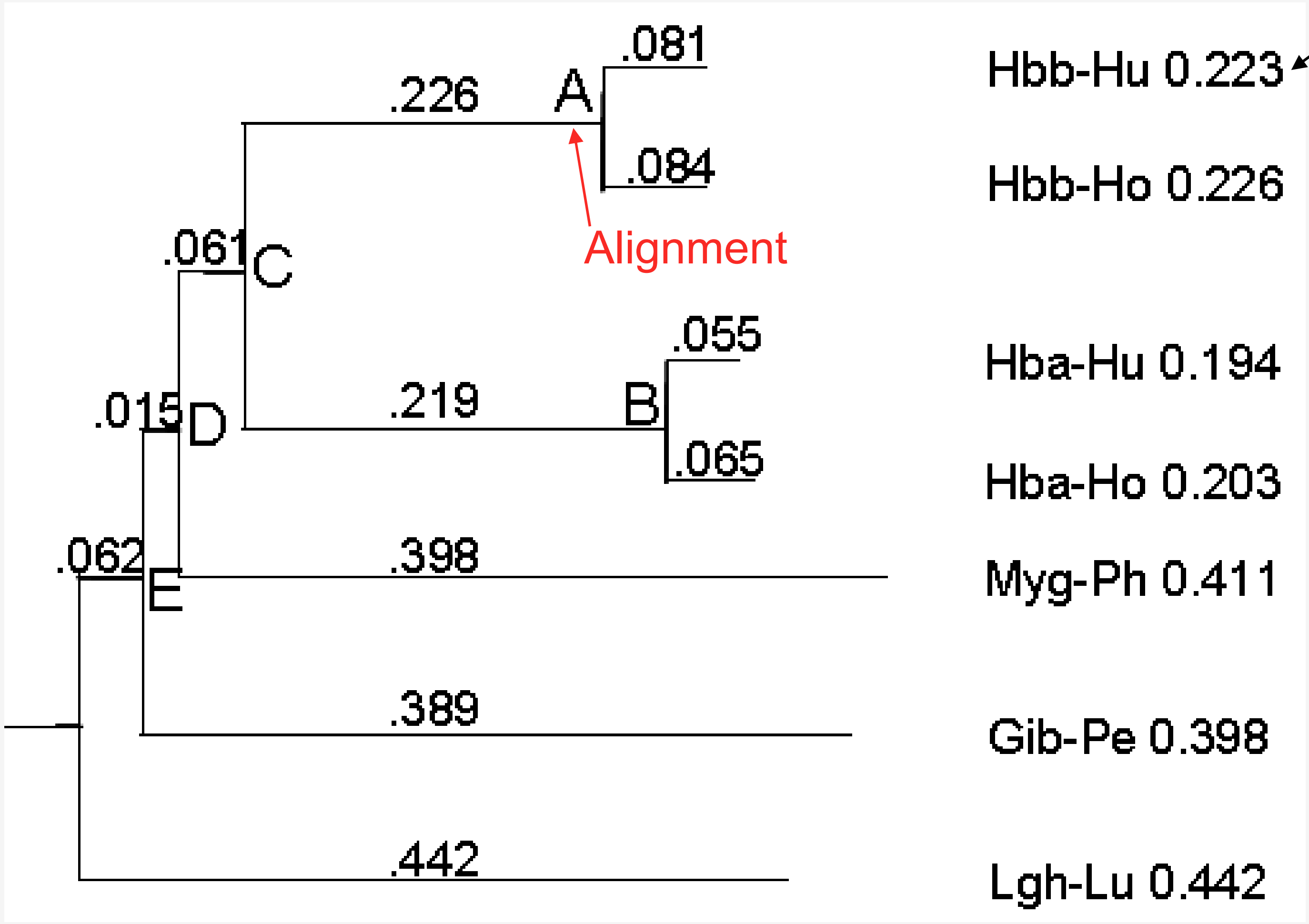
D = 1 – (I)

D = Difference score

$$I = \frac{\text{\# of identical aa's in pairwise global alignment}}{\text{total number of aa's in shortest sequence}}$$

Step 2-Create Rooted Tree and calculate weights

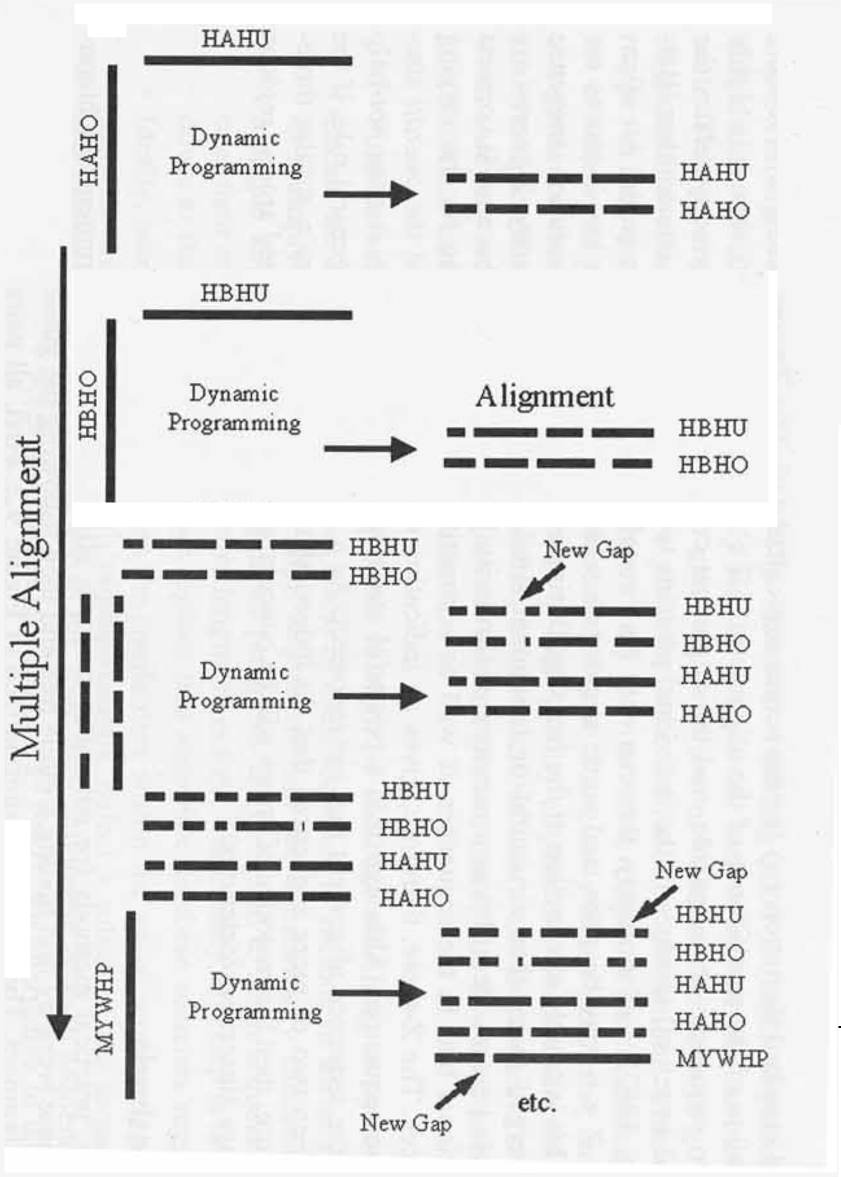
Neighbor joining algorithm – simple, to be discussed later

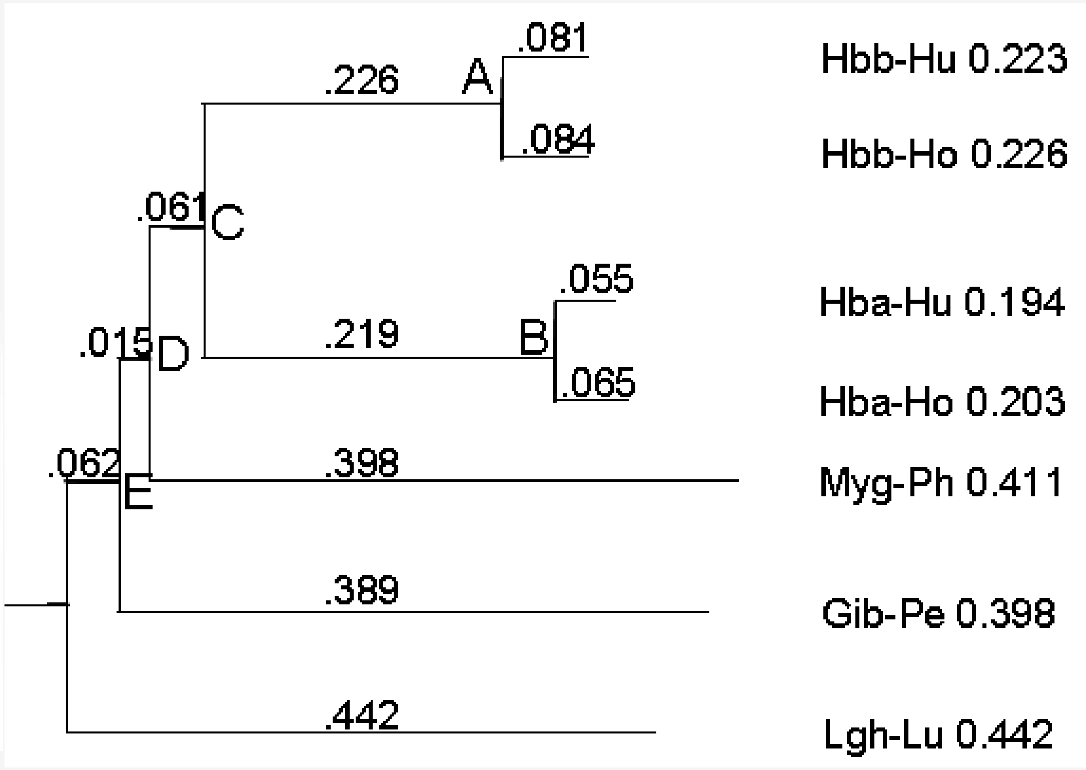


Weight

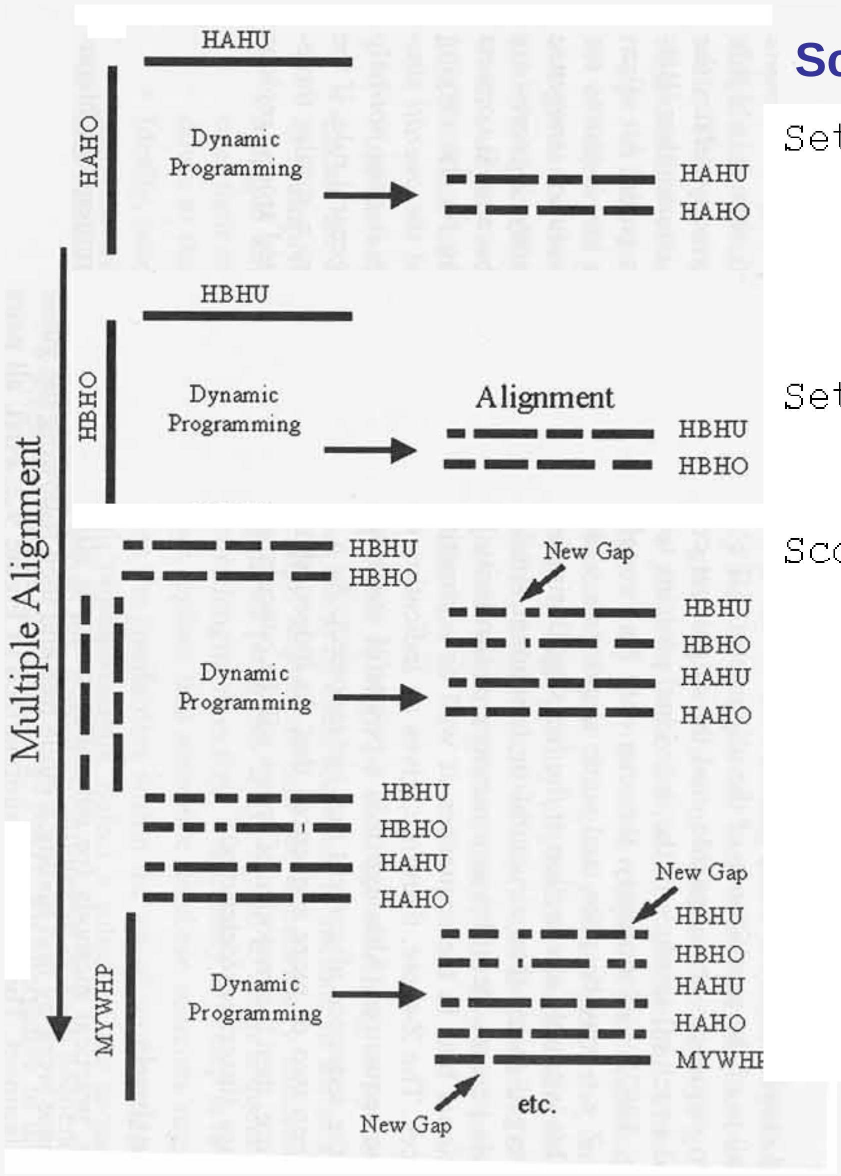
Order of alignment:
1 Hba-Hu vs Hba-Ho
2 Hbb-Hu vs Hbb-Ho
3 A vs B
4 Myg-Ph vs C
5 Gib-Pe vs D
6 Lgh-Lu vs E

Step 3-Progressive alignment





Step 3-Progressive alignment



Scoring during progressive alignment

```
Set of 4:      1 eeksav[t]al
               2 eekaav[l]al
               3 adktnv[k]aa
               4 adktnv[k]aa

Set of 2:      5 gewql[v]lhv
               6 aektk[i]rsa
```

$$\text{Score} = \left.\begin{array}{rl} & M(t,v)*W_1*W_5 \\ + & M(t,i)*W_1*W_6 \\ + & M(l,v)*W_2*W_5 \\ + & M(l,i)*W_2*W_6 \\ + & M(k,v)*W_3*W_5 \\ + & M(k,i)*W_3*W_6 \\ + & M(k,v)*W_4*W_5 \\ + & M(k,i)*W_4*W_6 \end{array}\right\} \text{divided by } 8$$

Recommended MSA Programs

- MUSCLE (fast and accurate)
- MAVID (genome-scale alignment)
- SAM (hidden markov, powerful and wide range of options)