Elements of Bioinformatics Autumn 2010

VELI MÄKINEN

HTTP://WWW.CS.HELSINKI.FI/EN/COURSES/582606/2010/S/K/1

Lecture Thu 4.11.

PROTEIN SEQUENCE ALIGNMENT AND MULTIPLE ALIGNMENT

Protein sequence alignment

- We have discussed alignment of DNA sequences
- Amino acid sequences can be aligned as well
- However, the design of the substitution matrix is more involved because of the larger alphabet
- Homologs can be easier identified with alignment of protein sequences:
  - *Synonymous (silent) mutations* that do not change the amino acid coding are frequent
    - Every third nucleotide can be mismatch in an alignment where amino acids match perfectly
  - Frameshifts, introns, etc. should be taken into account when aligning protein coding DNA sequences

Example

- Consider RNA sequence alignment:

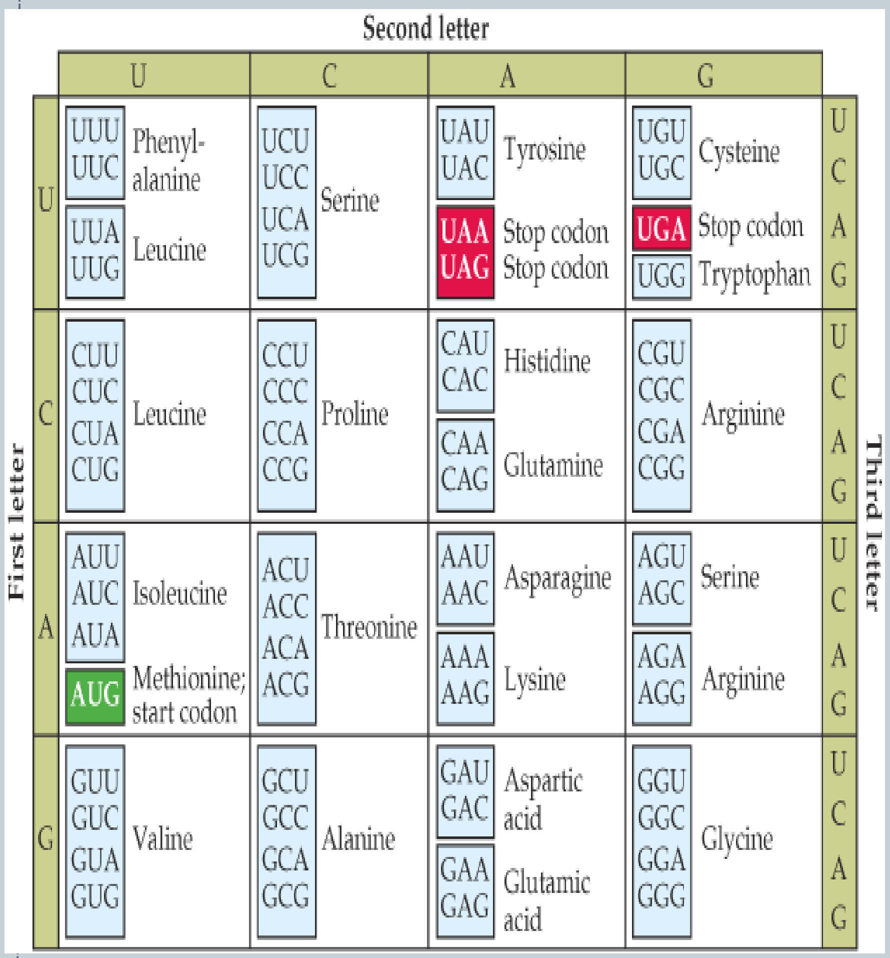
  AUGAUUACUCAUAGA...

  AUGAUCACCCACAGG...

- Versus protein sequence alignment:

  MITHR...

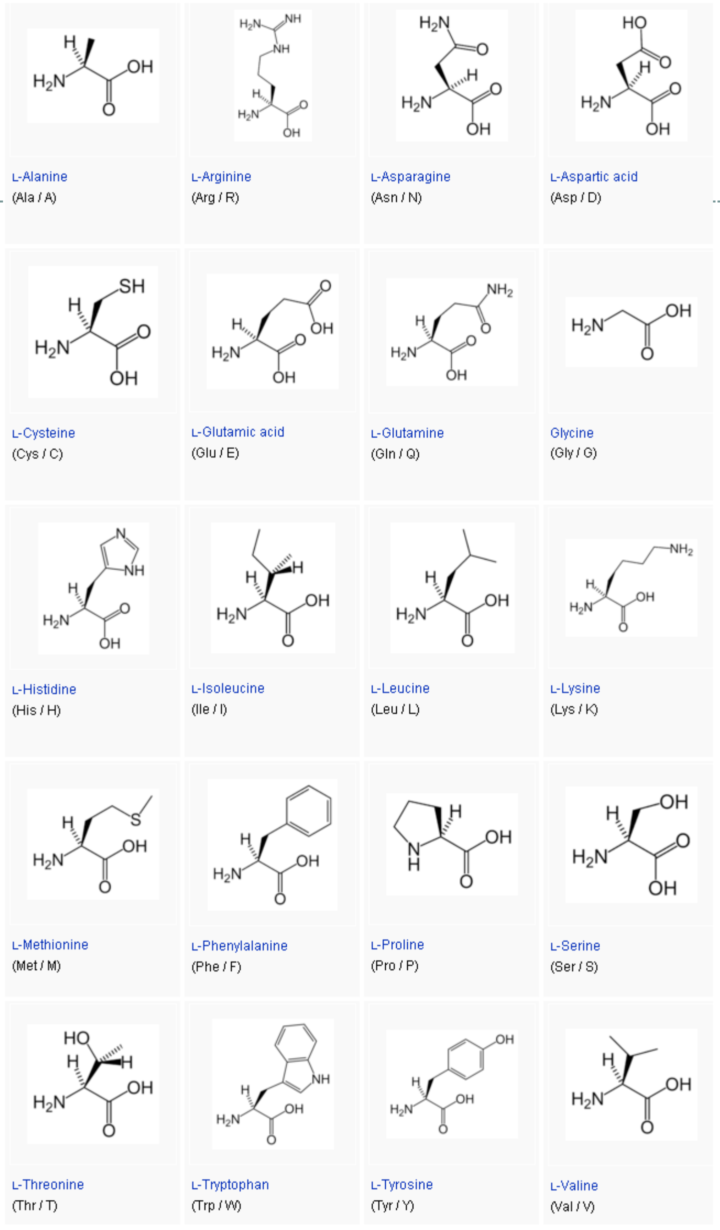
  MITHR...



	Second letter				
First letter	U	C	A	G	Third letter
U	UUU UUC Phenyl-alanine; UUA UUG Leucine	UCU UCC UCA UCG Serine	UAU UAC Tyrosine; UAA Stop codon; UAG Stop codon	UGU UGC Cysteine; UGA Stop codon; UGG Tryptophan	U C A G
C	CUU CUC CUA CUG Leucine	CCU CCC CCA CCG Proline	CAU CAC Histidine; CAA CAG Glutamine	CGU CGC CGA CGG Arginine	U C A G
A	AUU AUC AUA Isoleucine; AUG Methionine; start codon	ACU ACC ACA ACG Threonine	AAU AAC Asparagine; AAA AAG Lysine	AGU AGC Serine; AGA AGG Arginine	U C A G
G	GUU GUC GUA GUG Valine	GCU GCC GCA GCG Alanine	GAU GAC Aspartic acid; GAA GAG Glutamic acid	GGU GGC GGA GGG Glycine	U C A G

Scoring amino acid alignments

- Substitutions between chemically similar amino acids are more frequent than between dissimilar amino acids
- We can check our scoring model against this



http://en.wikipedia.org/wiki/List_of_standard_amino_acids

Score matrices

- Let $A = a_1a_2\ldots a_n$ and $B = b_1b_2\ldots b_n$ be sequences of equal length (no gaps allowed to simplify things)
- To obtain a score for alignment of A and B, where a_i is aligned against b_i, we take the ratio of two probabilities
  - The probability of having A and B where the characters match (*match model M*)
  - The probability that A and B were chosen randomly (*random model R*)

Score matrices: random model

- Under the random model, the probability of having A and B is

$$P(A, B|R) = \prod_i q_{ai} \prod_i q_{bi}$$

where q_{xi} is the probability of occurrence of amino acid type x_i

- Position where an amino acid occurs does not affect its type

Score matrices: match model

- Let p_{ab} be the probability of having amino acids of type a and b aligned against each other given they have evolved from the same ancestor c
- The probability is

$$P(A, B|M) = \prod_i p_{a_i b_i}$$

Score matrices: log-odds ratio score

- We obtain the score S by taking the ratio of these two probabilities

$$\frac{P(A,B|M)}{P(A,B|R)} = \frac{\prod_i p_{a_i b_i}}{\prod_i q_{a_i} \prod_i q_{b_i}} = \prod_i \frac{p_{a_i b_i}}{q_{a_i} q_{b_i}}$$

and taking a logarithm of the ratio

$$S = \log_2 \frac{P(A,B|M)}{P(A,B|R)} = \sum_{i=1}^{n} \log_2 \frac{p_{a_i b_i}}{q_{a_i} q_{b_i}} = \sum_{i=1}^{n} s(a_i, b_i)$$

Score matrices: log-odds ratio score

$$S = \log_2 \frac{P(A,B|M)}{P(A,B|R)} = \sum_{i=1}^{n} \log_2 \frac{p_{a_i b_i}}{q_{a_i} q_{b_i}} = \sum_{i=1}^{n} s(a_i, b_i)$$

- The score S is obtained by summing over character pair-specific scores:

$$s(a,b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

- The probabilities q_a and p_{ab} are extracted from data

Calculating score matrices for amino acids

- Probabilities q_a are in principle easy to obtain:
  - Count relative frequencies of every amino acid in a sequence database

$$s(a,b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

Calculating score matrices for amino acids

- To calculate p_{ab} we can use a known pool of aligned sequences
- BLOCKS is a database of highly conserved regions for proteins
- It lists *multiple aligned*, ungapped and conserved protein segments
- Example from BLOCKS shows genes related to human gene associated with DNA-repair defect xeroderma pigmentosum

$$s(a,b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

Block PR00851A
ID XRODRMPGMNTB; BLOCK
AC PR00851A; distance from previous block=(52,131)
DE Xeroderma pigmentosum group B protein signature
BL adapted; width=21; seqs=8; 99.5%=985; strength=1287

```
XPB_HUMAN|P19447 ( 74)   RPLWVAPDGHIFLEAFSPVYK 54
XPB_MOUSE|P49135 ( 74)   RPLWVAPDGHIFLEAFSPVYK 54
P91579 ( 80)             RPLYLAPDGHIFLESFSPVYK 67
XPB_DROME|Q02870 ( 84)   RPLWVAPNGHVFLESFSPVYK 79
RA25_YEAST|Q00578 ( 131) PLWISPSDGRIILESFSPLAE 100
Q38861 ( 52)             RPLWACADGRIFLETFSPLYK 71
O13768 ( 90)             PLWINPIDGRIILEAFSPLAE 100
O00835 ( 79)             RPIWVCPDGHIFLETFSAIYK 86
```

http://blocks.fhcrc.org

BLOSUM matrix

- BLOSUM is a score matrix for amino acid sequences derived from BLOCKS data
- First, count pairwise matches $f_{x,y}$ for every *amino acid type pair (x, y)*
- For example, for column 3 and amino acids L and W, we find 8 pairwise matches: $f_{L,W} = f_{W,L} = 8$

```
RPLWVAPD
RPLWVAPR
RPLWVAPN
PLWISPSD
RPLWACAD
PLWINPID
RPIWVCPD
```

Creating a BLOSUM matrix

- Probability p_{ab} is obtained by dividing f_{ab} with the total number of pairs (note difference with course book):

$$p_{ab} = f_{ab} / \sum_{x=1}^{20} \sum_{y=1}^{x} f_{xy}$$

- We get probabilities q_a by

$$q_a = \sum_{b=1}^{20} p_{ab}$$

```
RPLWVAPD
RPLWVAPR
RPLWVAPN
PLWISPSD
RPLWACAD
PLWINPID
RPIWVCPD
```

Creating a BLOSUM matrix

- The probabilities p_{ab} and q_a can now be plugged into

$$s(a,b) = \log_2 \frac{p_{ab}}{q_a q_b}$$

to get a 20 x 20 matrix of scores s(a, b).

- Next slide presents the BLOSUM62 matrix
  - Values scaled by factor of 2 and rounded to integers
  - Additional step required to take into account expected evolutionary distance
  - Described in Deonier's book in more detail

BLOSUM62

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	0	-4
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3	-1	0	-1	-4
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3	3	0	-1	-4
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3	4	1	-1	-4
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1	-3	-3	-2	-4
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2	0	3	-1	-4
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3	-1	-2	-1	-4
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3	0	0	-1	-4
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3	-3	-3	-1	-4
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1	-4	-3	-1	-4
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1	-4
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1	-3	-1	-1	-4
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1	-3	-3	-1	-4
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2	-2	-1	-2	-4
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0	0	-4
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0	-1	-1	0	-4
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3	-4	-3	-2	-4
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-3	-2	-1	-4
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4	-3	-2	-1	-4
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4	1	-1	-4
Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	0	0	-2	-1	-1	-1	-1	-1	-4
*	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1

Using BLOSUM62 matrix

```
MQLEANADTSV
 |  | |
LQEQAEAQGEM
```

$$s = \sum_{i=1}^{11} s(a_i, b_i)$$

$= 2 + 5 - 3 - 4 + 4 + 0 + 4 + 0 - 2 + 0 + 1$

$= 7$

Why positive score alignment is meaningfull?

- We have designed scoring matrix so that expected score of random match at any position is negative:

$$\sum_{a,b} q_a q_b s(a,b) < 0.$$

- This can be seen by noticing that

$$\sum_{a,b} q_a q_b s(a,b) = -\sum_{a,b} q_a q_b \log\left(\frac{q_a q_b}{p_{ab}}\right) = -H(q^2 \parallel p),$$

where H(q² || p) is the *relative entropy* (or *Kullback-Leibler divergence*) of distribution q² = q x q with respect to distribution p. Value of H(q² || p) is always positive unless q²=p. (Exercise: show why.)

What about gap penalties?

- Similar log-odds reasoning gives that the gap penalty should be $-\log f(k)$, where k is the gap length, and $f()$ is the function modeling the replication process (See Durbin et al., page 17).
  - $-\log \delta k$ for the linear model
  - $-\log (\alpha + \beta(k - 1))$ for the affine gap model
- However, logarithmic gap penalties are difficult (yet possible) to take into account in dynamic programming:
  - Eppstein et al. Sparse dynamic programming II: convex and concave cost functions. *Journal of the ACM*, 39(3):546-567, 1992.

What about gap penalties? (2)

- Typically some *ad hoc* values are used, like $\delta=8$ in the linear model and $\alpha=12$, $\beta=2$ in the affine gap model.
- It can be argued that penalty of insertion + deletion should be always greater than penalty for one mismatch.
  - Otherwise expected score of random match may get positive.

Multiple alignment

- Consider a set of d sequences on the right
  - Orthologous sequences from different organisms
  - Paralogs from multiple duplications
- How can we study relationships between these sequences?
- Aligning simultaneously many sequences gives better estimates for the homology, as many sequences vote for the same ”column”.

```
AGCAGTGATGCTAGTCG
ACAGCAGTGGATGCTAGTCG
ACAGAGTGATGCTATCG
CAGCAGTGCTGTAGTCG
ACAAGTGATGCTAGTCG
ACAGCAGTGATGCTAGCG
AGCAGTGGATGCTAGTCG
AAGTGATGCTAGTCG
ACAGCGATGCTAGGGTCG
```

Multiple alignment notation

- Let M denote the multiple alignment, i.e., a matrix with d sequences being the rows with gap symbols ”-” inserted so that all rows are the same length.
- Let M_{i*} and M_{*j} denote the i-th row (j-th column) in the alignment, respectively, and M_{ij} the symbol at i-th row and j-th column.

```
              j
  A--GC-AGTG--ATGCTAGTCG
  ACAGC-AGTG-GATGCTAGTCG
  ACAG--AGT--GATGCTA-TCG
  -CAGC-AGTG--CTG-TAGTCG
  ACA---AGTG--ATGCTAGTCG
i ACAGC-AGTG--ATGCTAG-CG
  A--GC-AGTG-GATGCTAGTCG
  A-AG----TG--ATGCTAGTCG
  ACAGCGA-TGCTAGGGT---CG
```

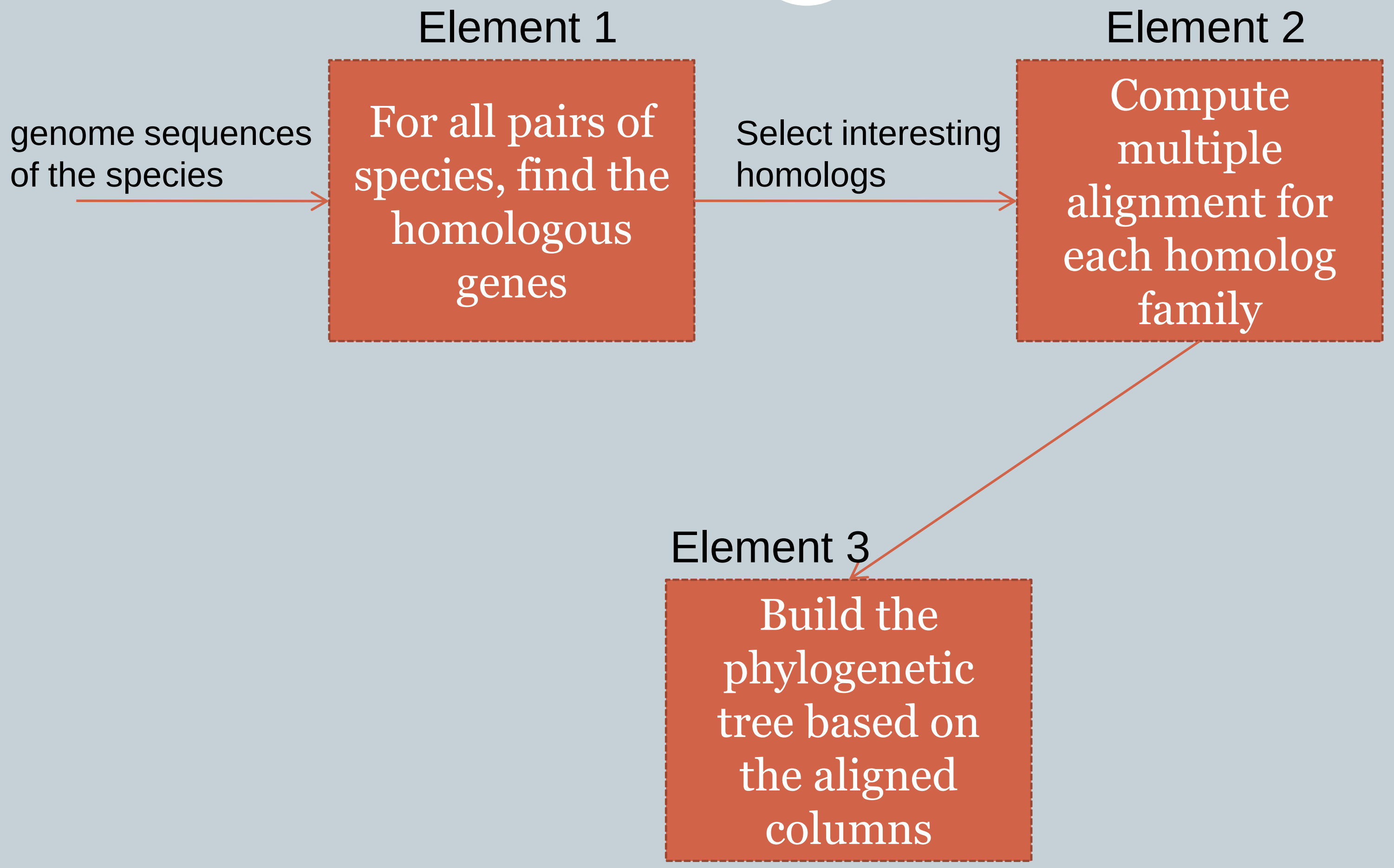
$M_{i,j}$=A

Applications of multiple alignment

- Amino acid scoring matrix estimation (chicken or the egg problem)
- Phylogeny by parsimony (chicken or the egg problem again)

```
A--GC-AGTG--ATGCTAGTCG
ACAGC-AGTG-GATGCTAGTCG
ACAG--AGT--GATGCTA-TCG
-CAGC-AGTG--CTG-TAGTCG
ACA---AGTG--ATGCTAGTCG
ACAGC-AGTG--ATGCTAG-CG
A--GC-AGTG-GATGCTAGTCG
A-AG----TG--ATGCTAGTCG
ACAGCGA-TGCTAGGGT---CG
```

Phylogeny by parsimony pipeline


Element 1
genome sequences of the species
For all pairs of species, find the homologous genes
Select interesting homologs
Element 2
Compute multiple alignment for each homolog family
Element 3
Build the phylogenetic tree based on the aligned columns

Optimal alignment of three sequences

- Alignment of $A = a_1a_2...a_i$ and $B = b_1b_2...b_j$ can end either in $(-, b_j)$, (a_i, b_j) or $(a_i, -)$
- $2^2 - 1 = 3$ alternatives
- Alignment of A, B and $C = c_1c_2...c_k$ can end in $2^3 - 1$ ways: $(a_i, -, -)$, $(-, b_j, -)$, $(-, -, c_k)$, $(-, b_j, c_k)$, $(a_i, -, c_k)$, $(a_i, b_j, -)$ or (a_i, b_j, c_k)
- Solve the recursion using three-dimensional dynamic programming matrix: $O(n^3)$ time and space
- Generalizes to d sequences but impractical with even a moderate number of sequences

Scoring multiple alignments

- Sum-of-pairs (SP) score:
  - $S(M) = \sum_j \sum_{i'<i} s(M_{i'j}, M_{ij})$, where s(a,b) is the given substitution score function.
  - Assumes all columns are independent.
  - Scores s(a,'-')=s('-',b) are the gap costs in the linear model.
  - For affine gap cost model, gaps are ignored from above and computed separately.
  - Widely used model in practice, but has the problem of counting the same substitutions several times:
    - See Durbin et al., page 140 for arguments against using this model.

Scoring multiple alignments (2)

- Minimum entropy score:
  - Let $c_j(a)$ be the number of times symbol a occurs at column M_{*j}.
  - $S(M) = -\sum_j \sum_a \log \frac{c_j(a)}{\sum_{a'} c_j(a')}$
  - Assumes all columns and all rows are independent.
    - No benefit from having close amino acids at the same column.
  - Gaps can either be counted as normal symbols, or separately in the case of affine gap costs.

Multiple alignment in practice

- In practice, real-world multiple alignment problems are usually solved with heuristics
- Progressive multiple alignment outline
  - Choose two sequences and align them
  - Choose third sequence w.r.t. two previous sequences and align the third against them
    - ”Once a gap, always a gap” principle
  - Repeat until all sequences have been aligned
  - Different options how to choose sequences and score alignments

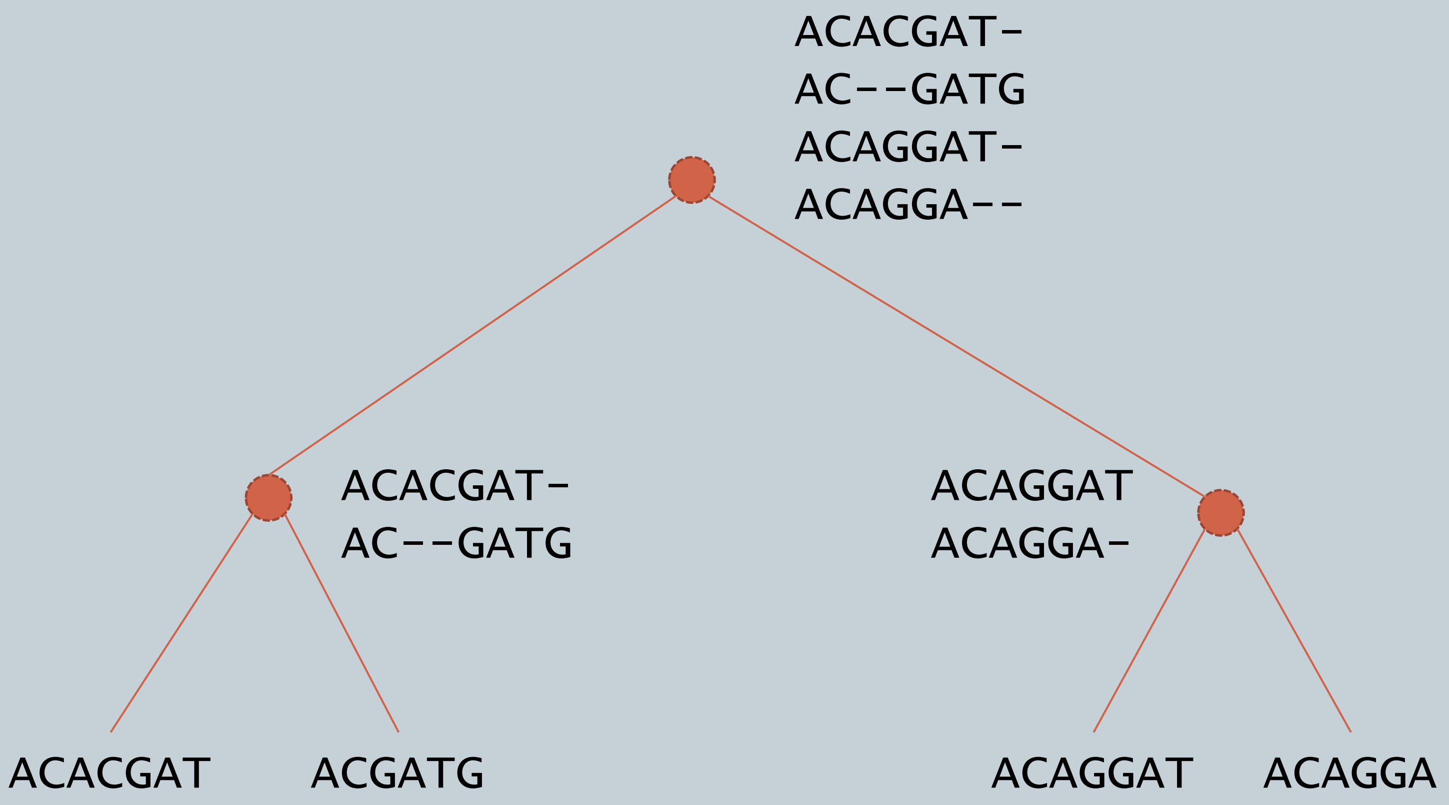
Multiple alignment in practice

- Profile-based progressive multiple alignment: CLUSTALW
  - Construct a distance matrix of all pairs of sequences using dynamic programming
  - Progressively align pairs in order of decreasing similarity
  - CLUSTALW uses various heuristics to contribute to accuracy

Generic framework for progressive multiple alignment

- Compute all pair-wise alignments for the d input sequences, converting the score into a distance D(A,B) between each sequence pair A,B.
- Use any *hierarchical clustering algorithm* on the distances D(,) to create a *guide tree* defining the order in which sequences are aligned:
  - Leaves represent the d sequences, and internal nodes the multiple alignment of the sequences in the leaves.
  - Multiple alignment to the root is created bottom-up aligning at each node sequence against sequence, sequence against multiple alignment, or multiple alignment against multiple alignment.

Progressive multiple alignment example



Practical exact algorithm for multiple alignment

- Small multiple alignments using SP score can be constructed without heuristics using a search space pruning technique by Carrillo & Lipman 1988:
  - Idea is to use sum of optimal pair-wise alignments as upper-bound for multiple alignment score, and a heuristically obtained multiple aligment as a lower-bound.
  - This gives lower-bound for each pair-wise alignment inside the optimal multiple alignment, and limits the cells in the high-dimensional dynamic programming matrix that need to be taken into account in the computation.

Practical exact algorithm for multiple alignment (2)

- Let $S(A_{i'},A_i)$ be the optimal global alignment score of sequences $A_{i'}$ and A_i whose alignments inside the multiple alignment have score
$$S_M(A_{i'},A_i) = \sum_j s(M_{i'j}, M_{ij}) \leq S(A_{i'},A_i).$$
- Obviously $S(M) \leq S_M(A_{i'},A_i) - S(A_{i'},A_i) + \sum_{k<l} S(A_k,A_l).$
- This gives the lower-bound
$$S_M(A_{i'},A_i) \geq S(M) + S(A_{i'},A_i) - \sum_{k<l} S(A_k,A_l)$$
$$\geq S(M') + S(A_{i'},A_i) - \sum_{k<l} S(A_k,A_l) = LB_{i'i},$$

where M' is a sub-optimal alignment computed using e.g. heuristic progressive alignment.

Practical exact algorithm for multiple alignment (3)

- Now find a set $B_{i'i}$ of coordinate pairs $(k_{i'},k_i)$ such that the best alignment of $A_{i'}$ and A_i through $(k_{i'},k_i)$ scores at least $LB_{i'i}$.
  - Compute $S(A_{i'}[1,k_{i'}],A_i[1,k_i])$ and $S(A_{i'}^{-1}[1,|A_{i'}|-k_{i'}],A_i^{-1}[1,|A_i|-k_i])$, where $^{-1}$ denotes the reverse of the sequence.
  - Set $B_{i'i}$ consists of all coordinate pairs $(k_{i'},k_i)$ where the sum of the two scores above is at least $LB_{i'i}$.
- Only coordinates $(k_1,k_2,\ldots,k_d)$ such that $(k_{i'},k_i)$ is in $B_{i'i}$ for all i',i need to be considered in filling the d-dimensional dynamic programming matrix to compute the optimal multiple alignment.