Multiple Sequence Alignment

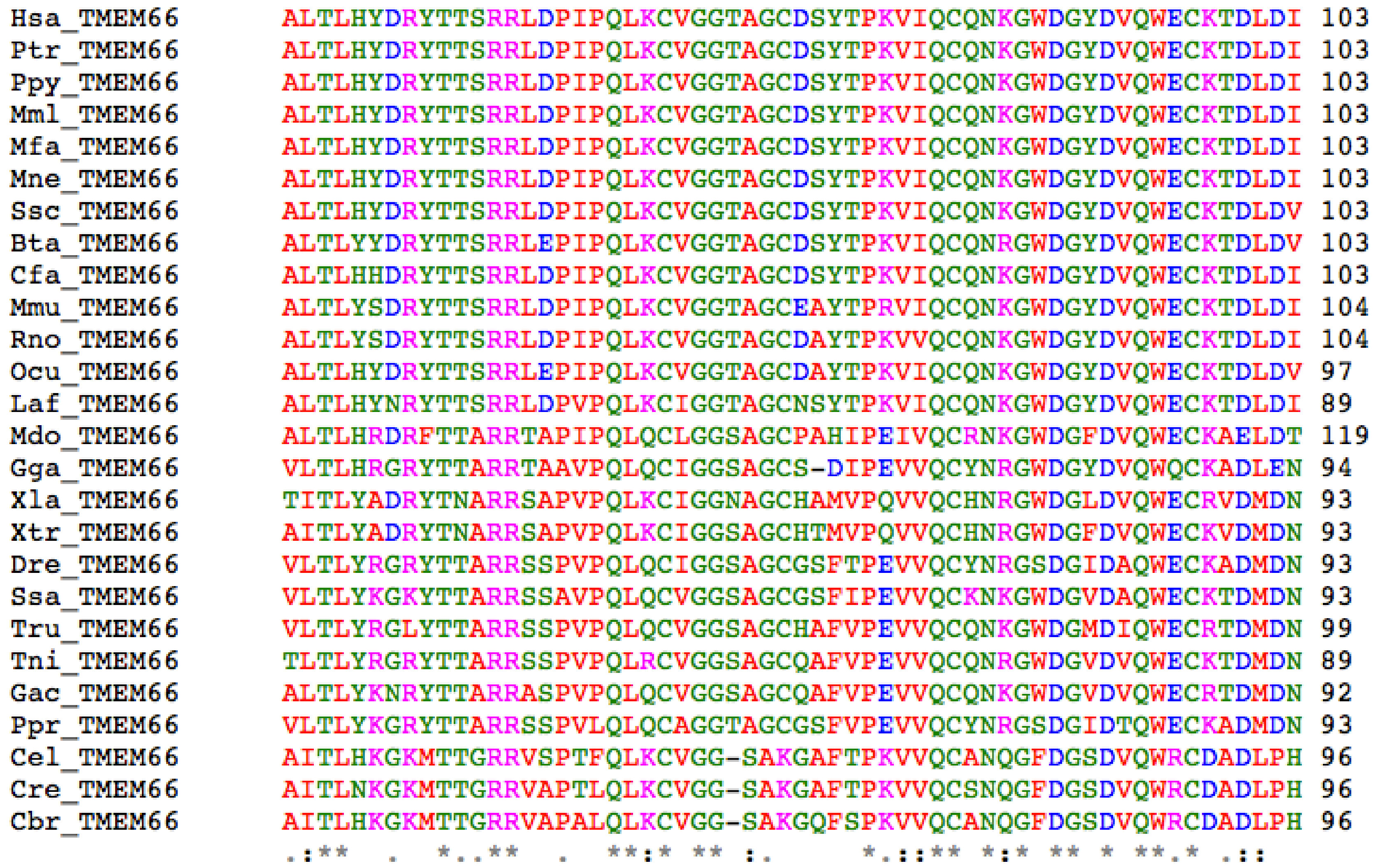
Multiple sequence alignment

- Multiple sequence alignment means positioning and adjusting more than two biological sequences (DNA, RNA, or protein) on top of each other.

- An MSA arranges protein sequences into a rectangular array with the goal that residues in a given column are **homologous** (derived from a common ancestor), **superposable** (in a 3D structural alignment) or play a **common functional role** (catalytic sites, nuclear localization signal, protein-protein interaction sites,...).

Optimal multiple sequence alignment

Idea: try to have a maximum of similar residues in a given column of the alignment

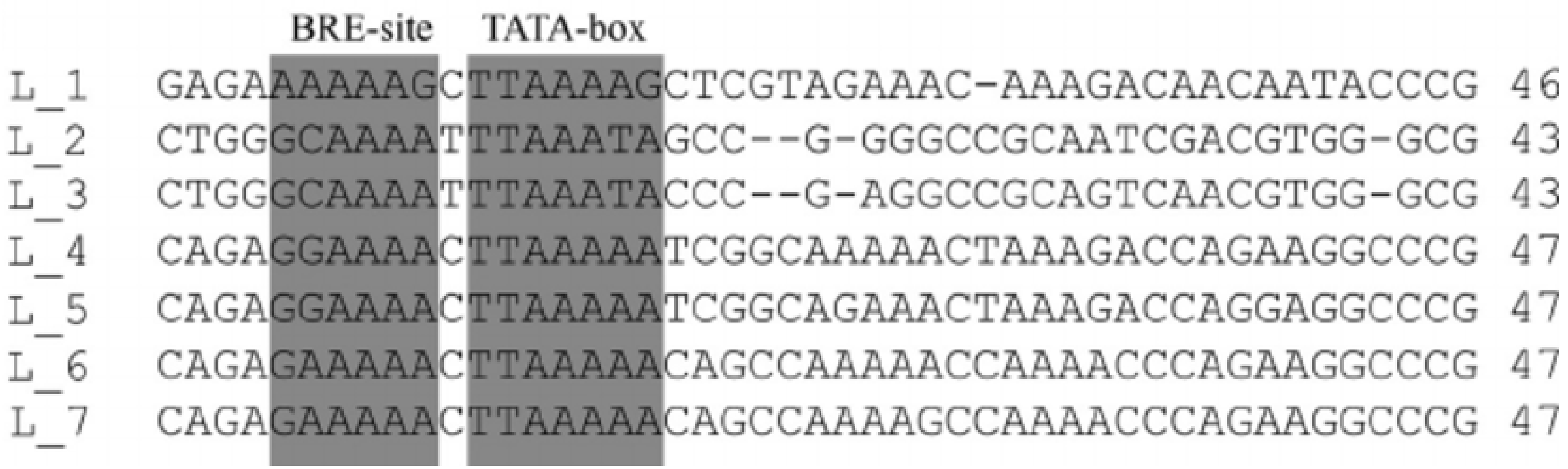


```
Hsa_TMEM66      ALTLHYDRYTTSRRLDPIPQLKCVGGTAGCDSYTPKVIQCQNKGWDGYDVQWECKTDLDI 103
Ptr_TMEM66      ALTLHYDRYTTSRRLDPIPQLKCVGGTAGCDSYTPKVIQCQNKGWDGYDVQWECKTDLDI 103
Ppy_TMEM66      ALTLHYDRYTTSRRLDPIPQLKCVGGTAGCDSYTPKVIQCQNKGWDGYDVQWECKTDLDI 103
Mml_TMEM66      ALTLHYDRYTTSRRLDPIPQLKCVGGTAGCDSYTPKVIQCQNKGWDGYDVQWECKTDLDI 103
Mfa_TMEM66      ALTLHYDRYTTSRRLDPIPQLKCVGGTAGCDSYTPKVIQCQNKGWDGYDVQWECKTDLDI 103
Mne_TMEM66      ALTLHYDRYTTSRRLDPIPQLKCVGGTAGCDSYTPKVIQCQNKGWDGYDVQWECKTDLDI 103
Ssc_TMEM66      ALTLHYDRYTTSRRLDPIPQLKCVGGTAGCDSYTPKVIQCQNKGWDGYDVQWECKTDLDV 103
Bta_TMEM66      ALTLYYDRYTTSRRLEPIPQLKCVGGTAGCDSYTPKVIQCQNRGWDGYDVQWECKTDLDV 103
Cfa_TMEM66      ALTLHHDRYTTSRRLDPIPQLKCVGGTAGCDSYTPKVIQCQNKGWDGYDVQWECKTDLDI 103
Mmu_TMEM66      ALTLYSDRYTTSRRLDPIPQLKCVGGTAGCEAYTPRVIQCQNKGWDGYDVQWECKTDLDI 104
Rno_TMEM66      ALTLYSDRYTTSRRLDPIPQLKCVGGTAGCDAYTPKVVQCQNKGWDGYDVQWECKTDLDI 104
Ocu_TMEM66      ALTLHYDRYTTSRRLEPIPQLKCVGGTAGCDAYTPKVIQCQNKGWDGYDVQWECKTDLDV 97
Laf_TMEM66      ALTLHYNRYTTSRRLDPVPQLKCIGGTAGCNSYTPKVIQCQNKGWDGYDVQWECKTDLDI 89
Mdo_TMEM66      ALTLHRDRFTTARRTAPIPQLQCLGGSAGCPAHIPEIVQCRNKGWDGFDVQWECKAELDT 119
Gga_TMEM66      VLTLHRGRYTTARRTAAVPQLQCIGGSAGCS-DIPEVVQCYNRGWDGYDVQWQCKADLEN 94
Xla_TMEM66      TITLYADRYTNARRSAPVPQLKCIGGNAGCHAMVPQVVQCHNRGWDGLDVQWECRVDMDN 93
Xtr_TMEM66      AITLYADRYTNARRSAPVPQLKCIGGSAGCHTMVPQVVQCHNRGWDGFDVQWECKVDMDN 93
Dre_TMEM66      VLTLYRGRYTTARRSSPVPQLQCIGGSAGCGSFTPEVVQCYNRGSDGIDAQWECKADMDN 93
Ssa_TMEM66      VLTLYKGKYTTARRSSAVPQLQCVGGSAGCGSFIPEVVQCKNKGWDGVDAQWECKTDMDN 93
Tru_TMEM66      VLTLYRGLYTTARRSSPVPQLQCVGGSAGCHAFVPEVVQCQNKGWDGMDIQWECRTDMDN 99
Tni_TMEM66      TLTLYRGRYTTARRSSPVPQLRCVGGSAGCQAFVPEVVQCQNRGWDGVDVQWECKTDMDN 89
Gac_TMEM66      ALTLYKNRYTTARRASPVPQLQCVGGSAGCQAFVPEVVQCQNKGWDGVDVQWECRTDMDN 92
Ppr_TMEM66      VLTLYKGRYTTARRSSPVLQLQCAGGTAGCGSFVPEVVQCYNRGSDGIDTQWECKADMDN 93
Cel_TMEM66      AITLHKGKMTTGRRVSPTFQLKCVGG-SAKGAFTPKVVQCANQGFDGSDVQWRCDADLPH 96
Cre_TMEM66      AITLNKGKMTTGRRVAPTLQLKCVGG-SAKGAFTPKVVQCSNQGFDGSDVQWRCDADLPH 96
Cbr_TMEM66      AITLHKGKMTTGRRVAPALQLKCVGG-SAKGQFSPKVVQCANQGFDGSDVQWRCDADLPH 96
                .:**  .  *..**  .  **:* ** :.     *.::** *:* ** * **.* .::
```

Why do we perform multiple sequence alignment?

- Detecting similarities between sequences (closely or distantly related) and conserved **regions/motifs/consensus** in sequences.

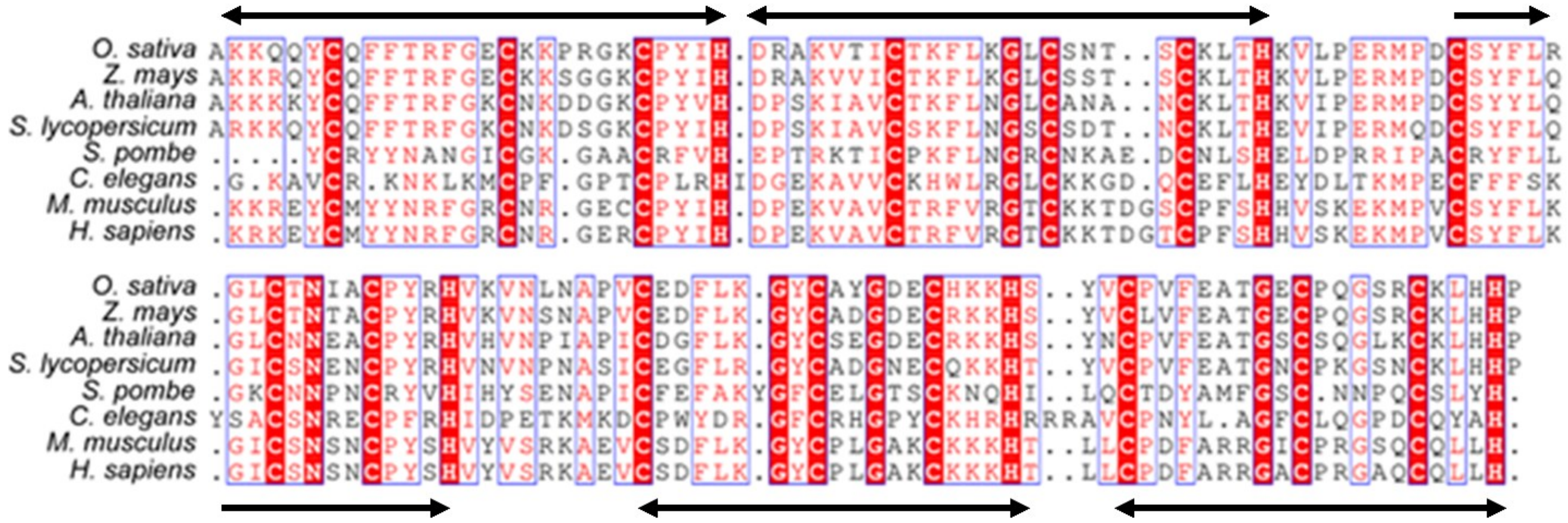
Sequence motif



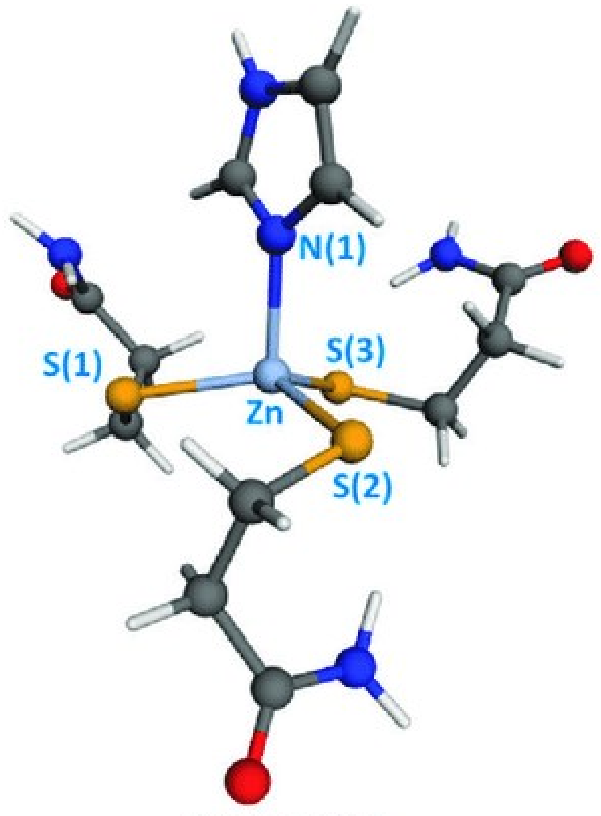
```
L_1   GAGAAAAAAGCTTAAAAGCTCGTAGAAAC-AAAGACAACAATACCCG 46
L_2   CTGGGCAAAATTTAAATAGCC--G-GGGCCGCAATCGACGTGG-GCG 43
L_3   CTGGGCAAAATTTAAATACCC--G-AGGCCGCAGTCAACGTGG-GCG 43
L_4   CAGAGGAAAACTTAAAAATCGGCAAAAACTAAAGACCAGAAGGCCCG 47
L_5   CAGAGGAAAACTTAAAAATCGGCAGAAACTAAAGACCAGGAGGCCCG 47
L_6   CAGAGAAAAACTTAAAAACAGCCAAAAACCAAAACCCAGAAGGCCCG 47
L_7   CAGAGAAAAACTTAAAAACAGCCAAAAGCCAAAACCCAGAAGGCCCG 47
```

Why do we perform multiple sequence alignment?

Structural motif



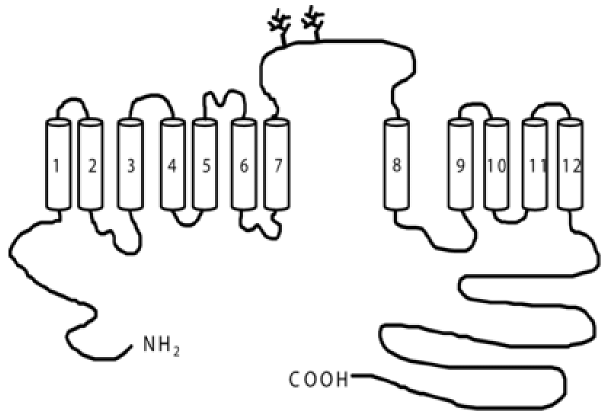
O. sativa AKKQQYCQFFTRFGECKKPRGKCPYIH.DRAKVTICTKFLKGLCSNT..SCKLTHKVLPERMPDCSYFLR
Z. mays AKKRQYCQFFTRFGECKKSGGKCPYIH.DRAKVVICTKFLKGLCSST..SCKLTHKVLPERMPDCSYFLQ
A. thaliana AKKKKYCQFFTRFGKCNKDDGKCPYVH.DPSKIAVCTKFLNGLCANA..NCKLTHKVIPERMPDCSYYLQ
S. lycopersicum ARKKQYCQFFTRFGKCNKDSGKCPYIH.DPSKIAVCSKFLNGSCSDT..NCKLTHEVIPERMQDCSYFLQ
S. pombe .....YCRYYNANGICGK.GAACRFVH.EPTRKTICPKFLNGRCNKAE.DCNLSHELDPRRIPACRYFLL
C. elegans .G.KAVCR.KNKLKMCPF.GPTCPLRHIDGEKAVVCKHWLRGLCKKGD.QCEFLHEYDLTKMPECFFSK
M. musculus .KKREYCMYYNRFGRCNR.GECCPYIH.DPEKVAVCTRFVRGTCKKTDGSCPFSHHVSKEKMPVCSYFLK
H. sapiens .KRKEYCMYYNRFGRCNR.GERCPYIH.DPEKVAVCTRFVRGTCKKTDGTCPFSHHVSKEKMPVCSYFLK
O. sativa .GLCTNIACPYRHVKVNLNAPVCEDFLK.GYCAYGDECHKKHS..YVCPVFEATGECPQGSRCKLHHP
Z. mays .GLCTNTACPYRHVKVNSNAPVCEDFLK.GYCADGDECRKKHS..YVCLVFEATGECPQGSRCKLHHP
A. thaliana .GLCNNEACPYRHVHVNPIAPICDGFLK.GYCSEGDECRKKHS..YNCPVFEATGSCSQGLKCKLHHP
S. lycopersicum .GICSNENCPYRHVNVNPNASICEGFLR.GYCADGNECQKKHT..YVCPVFEATGNCPKGSNCKLHHP
S. pombe .GKCNNPNCRYVHIHYSENAPICFEFAKYGFCELGTSCKNQHI..LQCTDYAMFGSC.NNPQCSLYH.
C. elegans YSACSNRECPFRHIDPETKMKDCPWYDR.GFCRHGPYCKHRHRRRAVCPNYL.AGFCLQGPDCQYAH.
M. musculus .GICSNSNCPYSHVYVSRKAEVCSDFLK.GYCPLGAKCKKKHT..LLCPDFARRGICPRGSQCQLLH.
H. sapiens .GICSNSNCPYSHVYVSRKAEVCSDFLK.GYCPLGAKCKKKHT..LLCPDFARRGACPRGAQCQLLH.



[$ZnCys_3His$]

Why do we perform multiple sequence alignment?

- Detection of **structural patterns** (hydrophobicity/hydrophilicity, gaps, etc.), thus assisting improved prediction of secondary and tertiary structures and loops and variable regions.



```
TM8
Rat      FAPLITAGIFGATLSSALAC
Mouse    FAPLITAGIFGATLSSALAC
Human    FAPLITAGIFGATLSSALAC
Rabbit   FAPLITAGIFGATLSSALAC
Flounder FAPLISAGIFGATLSSALAC
```

```
TM9
Rat      LAYAIAVAFIIIAELNTIAP
Mouse    LAYAIAVAFIIIAELNTIAP
Human    LAYAIAVAFIIIAELNTIAP
Rabbit   LAYAIAVAFIIIAELNTIAP
Flounder LTYVIAVCFVLIAELNTIAP
```

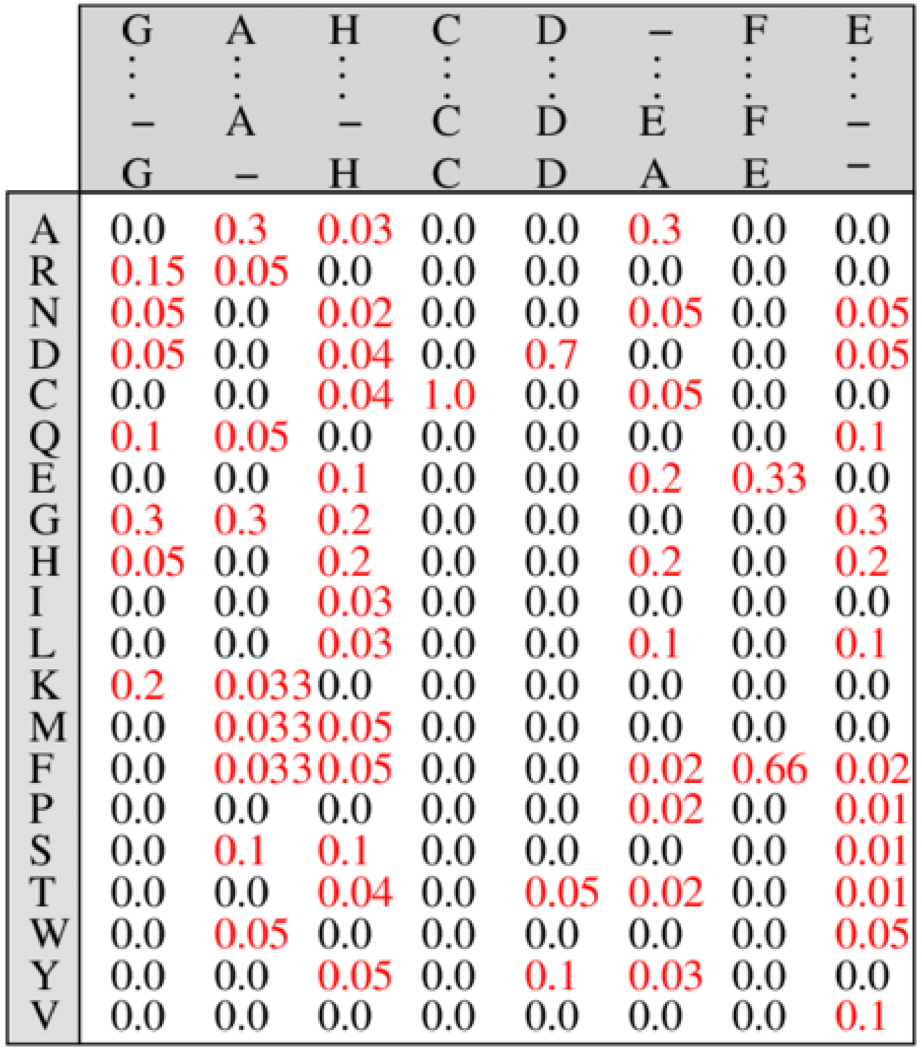
```
TM10
Rat      IISNFFLCSYALINFSCFHA
Mouse    IISNFFLCSYALINFSCFHA
Human    IISNFFLCSYALINFSCFHA
Rabbit   IISNFFLCSYALINFSCFHA
Flounder IISNFFLCSYALINFSCFHA
```

```
TM11
Rat      KWAALFGAVISVVIMFLLTW
Mouse    KWAALFGAVISVVIMFLLTW
Human    KWAALFGAIISVVIMFLLTW
Rabbit   KWSALFGAVVSVVIMFLLTW
Flounder KWLSLLGAVCCVVIMFLLTW
```

```
TM12
Rat      WAALIAIGVVLFLLLYVIYK
Mouse    WAALIAIGVVLFLLLYVIYK
Human    WAALIAIGVVLFLLLYVIYK
Rabbit   WAALIAIGVILFLLLYVIYK
Flounder WAALIAFGVVFFLLGYTLYK
```

Why do we perform multiple sequence alignment?

- Making patterns or **profiles** that can be further used to predict new sequences falling in a given family.



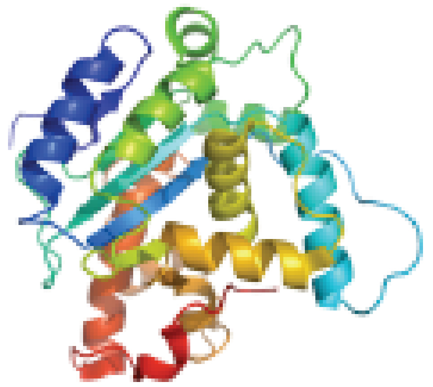
	G	A	H	C	D	–	F	E
	–	A	–	C	D	E	F	–
	G	–	H	C	D	A	E	–
A	0.0	0.3	0.03	0.0	0.0	0.3	0.0	0.0
R	0.15	0.05	0.0	0.0	0.0	0.0	0.0	0.0
N	0.05	0.0	0.02	0.0	0.0	0.05	0.0	0.05
D	0.05	0.0	0.04	0.0	0.7	0.0	0.0	0.05
C	0.0	0.0	0.04	1.0	0.0	0.05	0.0	0.0
Q	0.1	0.05	0.0	0.0	0.0	0.0	0.0	0.1
E	0.0	0.0	0.1	0.0	0.0	0.2	0.33	0.0
G	0.3	0.3	0.2	0.0	0.0	0.0	0.0	0.3
H	0.05	0.0	0.2	0.0	0.0	0.2	0.0	0.2
I	0.0	0.0	0.03	0.0	0.0	0.0	0.0	0.0
L	0.0	0.0	0.03	0.0	0.0	0.1	0.0	0.1
K	0.2	0.033	0.0	0.0	0.0	0.0	0.0	0.0
M	0.0	0.033	0.05	0.0	0.0	0.0	0.0	0.0
F	0.0	0.033	0.05	0.0	0.0	0.02	0.66	0.02
P	0.0	0.0	0.0	0.0	0.0	0.02	0.0	0.01
S	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.01
T	0.0	0.0	0.04	0.0	0.05	0.02	0.0	0.01
W	0.0	0.05	0.0	0.0	0.0	0.0	0.0	0.05
Y	0.0	0.0	0.05	0.0	0.1	0.03	0.0	0.0
V	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1

Why do we perform multiple sequence alignment?

Protein structure and function prediction

```
VTISCTGSSSNIGAG-NHVKWYQQLPG
VTISCTGTSSNIGS--ITVNWYQQLPG
LRLSCSVSGFIFSS--YAMYWVRQAPG
LSLTCTVSGTSFDD--YYSTWVRQPPG
PEVTCVVVDVSHEDPQVKFNWYVDG--
ATLVCTISDFYPGA--VTVAWKADS--
AALGCTVKDYFPEP--VTVSWNSG---
VSLTCTVKGFYPSD--IAVEWESNG--
```

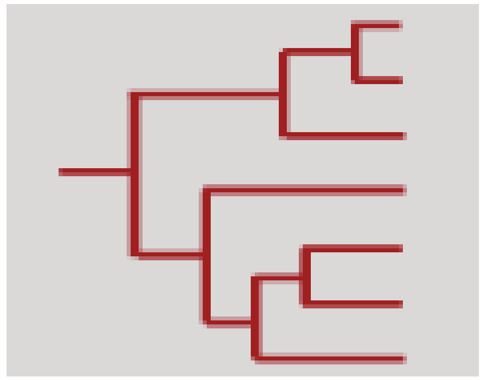
$\longrightarrow$



Phylogenetic inference

```
VTISCTGSSSNIGAG-NHVKWYQQLPG
VTISCTGTSSNIGS--ITVNWYQQLPG
LRLSCSVSGFIFSS--YAMYWVRQAPG
LSLTCTVSGTSFDD--YYSTWVRQPPG
PEVTCVVVDVSHEDPQVKFNWYVDG--
ATLVCTISDFYPGA--VTVAWKADS--
AALGCTVKDYFPEP--VTVSWNSG---
VSLTCTVKGFYPSD--IAVEWESNG--
```

$\longrightarrow$

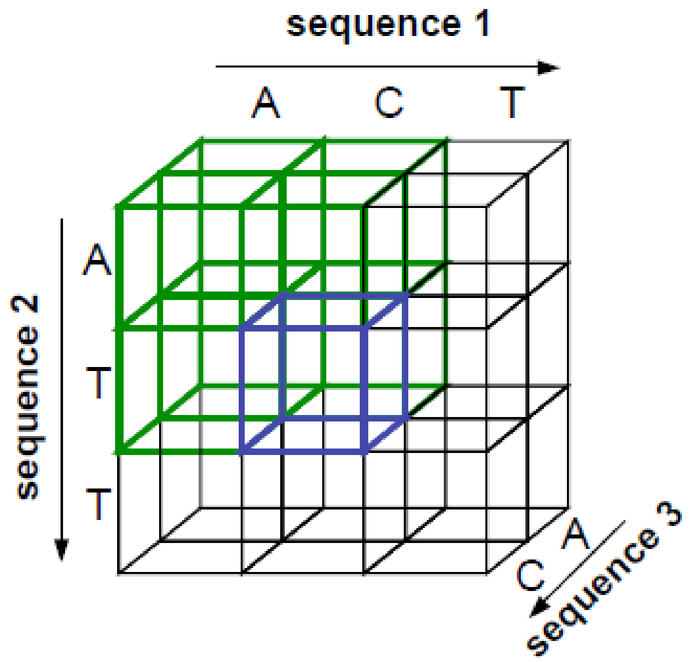


Challenges of multiple sequence alignment?

Obtaining a reliable multiple sequence alignment is not a trivial task! This problem can be formulated as a **combinatorial optimization problem.**

Two challenges:

- How to align the sequences in order to optimize the score?
- How to define a proper scoring system?



❖ As for the pair-wise alignment, the goal is to find a path (i.e. an alignment) in the (hyper) cube which leads to the optimal score (i.e. highest score).

Methods for Multiple Sequence Alignment

1. Optimal Dynamic Programming
2. Heuristic methods (Progressive alignments, Iterative alignments, Consensus alignments)
3. Hidden Markov models

Dynamic Programming

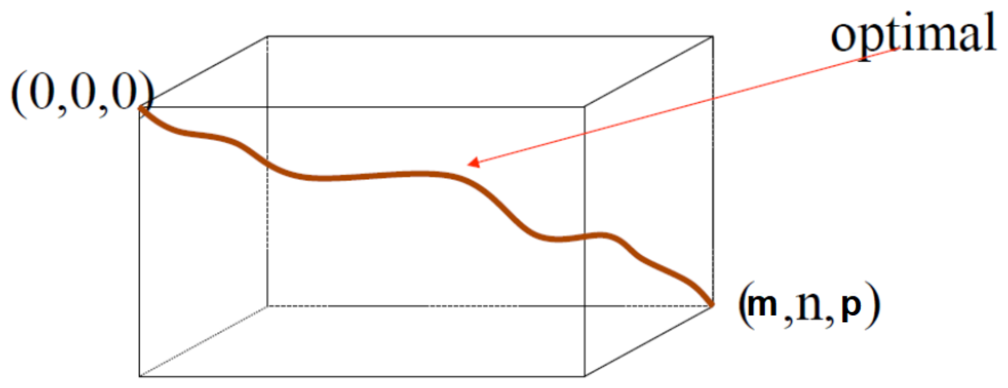
- For three sequences S1[m], S2[n] and S3[p]

$$\begin{matrix} S^1_1 \dots S^1_i \dots S^1_m \\ S^2_1 \dots S^2_j \dots S^2_n \\ S^3_1 \dots S^3_k \dots S^3_p \end{matrix}$$

$$\boxed{\begin{matrix} S^1_i \\ - \\ - \end{matrix}} \quad \boxed{\begin{matrix} - \\ S^2_j \\ - \end{matrix}} \quad \boxed{\begin{matrix} - \\ - \\ S^3_k \end{matrix}} \quad \boxed{\begin{matrix} S^1_i \\ S^2_j \\ - \end{matrix}} \quad \boxed{\begin{matrix} S^1_i \\ - \\ S^3_k \end{matrix}} \quad \boxed{\begin{matrix} - \\ S^2_j \\ S^3_k \end{matrix}} \quad \boxed{\begin{matrix} S^1_i \\ S^2_j \\ S^3_k \end{matrix}}$$

- Score (S^1_i,S^2_j,S^3_k) = score of optimum alignment among S1[m], S2[n], S3[p] where $1 \le i \le m$, $1 \le j \le n$, $1 \le k \le p$.

$$\text{Score}(S^1_i,S^2_j,S^3_k) = \max \begin{cases} \text{Score}(S^1_{i-1},S^2_j,S^3_k) + w(a_i, -, -) \\ \text{Score}(S^1_i,S^2_{j-1},S^3_k) + w(-, a_j, -) \\ \text{Score}(S^1_i,S^2_j,S^3_{k-1}) + w(-, -, a_k) \\ \text{Score}(S^1_{i-1},S^2_{j-1},S^3_k) + w(a_i, a_j, -) \\ \text{Score}(S^1_{i-1},S^2_j,S^3_{k-1}) + w(a_i, -, a_k) \\ \text{Score}(S^1_i,S^2_{j-1},S^3_{k-1}) + w(-, a_j, a_k) \\ \text{Score}(S^1_{i-1},S^2_{j-1},S^3_{k-1}) + w(a_i, a_j, a_k) \end{cases}$$



Dynamic Programming Programs

- MSA, Limited only up to 8-10 sequences (1989)
- DCA (Divide and Conquer; Stoye et al., 1997), 20-25 sequences
- OMA (Optimal Multiple Alignment; Reinert et al., 2000)
- COSA (Althaus et al., 2002)

Dynamic Programming MSA: Limitations

- In dynamic programming approach running time grows elementally with the number of sequences.

- For *k* sequences of length *n,* dynamic programming algorithm does $O(n^k \times (2^k-1))$ operations.

Exercise: Assume that we have a number of sequences that are 50 residues long and that a pairwise comparison of 2 such sequences takes one second of CPU time on a computer. An alignment of four sequences ($N=4$) *takes* $T = (2L)^{N-2} = 10^{2N-4} = 10^4$ *seconds = a few hours. If we had unlimited* memory and we were willing to wait for the answer until just before the Sun burns out in 5000000000 years, how many sequences could our computer align?

Heuristic approaches to multiple sequence alignment

Heuristic Methods

- Star alignment
- Progressive alignment methods
- Branch and bound
- Genetic algorithms
- Gibbs sampler
- Heuristic variants of Dynamic Programming Approach

Star alignment

Steps:

- Choose one sequence to be the center.
- Align all pair-wise sequences with the center.
- Merge the alignments: use the center as reference.
- Rule "once a gap always a gap".

Choosing the center

1. Try all possibilities and choose the resulting alignment that gives highest score Or
2. Take sequence S_c that maximizes

$$\Sigma_{\text{i different than c}} \text{pairwise-score}(S_c, S_i)$$

Star alignment

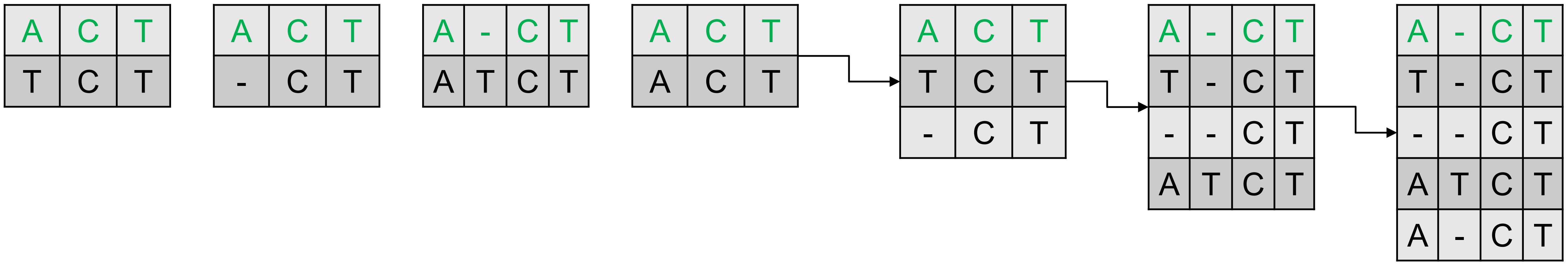
Merging the sequences in star alignment

- Use the center as the “guide” sequence.
- Add iteratively each pair-wise alignment to the multiple alignment.
- Gap insertion/deletions
    - If there is no gap (neither in the guide sequence in the multiple alignment nor in the merged alignment nor both have gaps), simply put the letter paired with the guide sequence into the appropriate column.
    - If pair-wise alignment produced a gap in the guide sequence, force the gap on the whole column of already aligned sequences.
    - If there is a gap in added sequence but not in the guide sequence, keep the gap in the added sequence.

Star alignment

Example 1

Let us take four sequences. S1: ACT, S2: TCT, S3: CT and S4: ATCT. Consider S1: ACT is at the center.



Star alignment

Example 2

Let us take other five sequences. S1: ATTGCCATT, S2: ATGGCCATT, S3: ATCCAATTTT, S4: ATCTTCTT and S5: ACTGACC. Consider S1: ATTGCCATT is at the center.

A	T	T	G	C	C	A	T	T
A	T	G	G	C	C	A	T	T

A	T	T	G	C	C	A	T	T	-	-
A	T	C	-	C	A	A	T	T	T	T

A	T	T	G	C	C	A	T	T
A	T	C	T	T	C	-	T	T

A	T	T	G	C	C	A	T	T
A	C	T	G	A	C	C	-	-

A	T	T	G	C	C	A	T	T	-	-
A	T	G	G	C	C	A	T	T	-	-
A	T	C	-	C	A	A	T	T	T	T

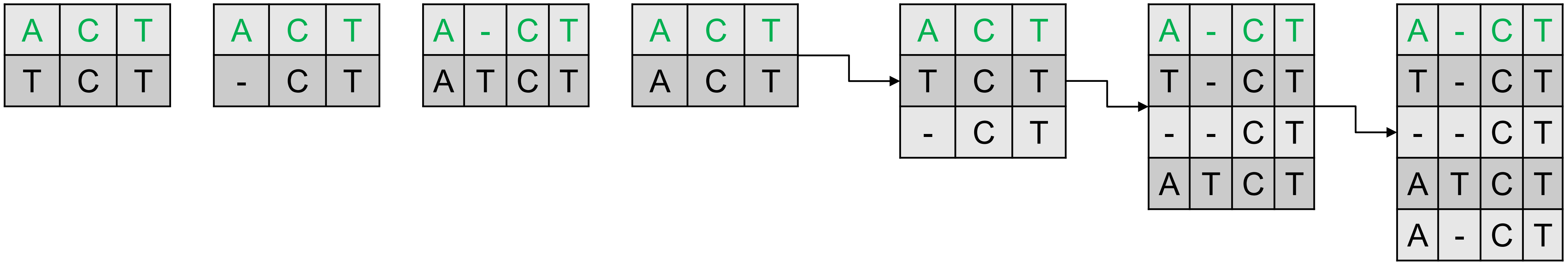
A	T	T	G	C	C	A	T	T	-	-
A	T	G	G	C	C	A	T	T	-	-
A	T	C	-	C	A	A	T	T	T	T
A	T	C	T	T	C	-	T	T	-	-

A	T	T	G	C	C	A	T	T	-	-
A	T	G	G	C	C	A	T	T	-	-
A	T	C	-	C	A	A	T	T	T	T
A	T	C	T	T	C	-	T	T	-	-
A	C	T	G	A	C	C	-	-	-	-

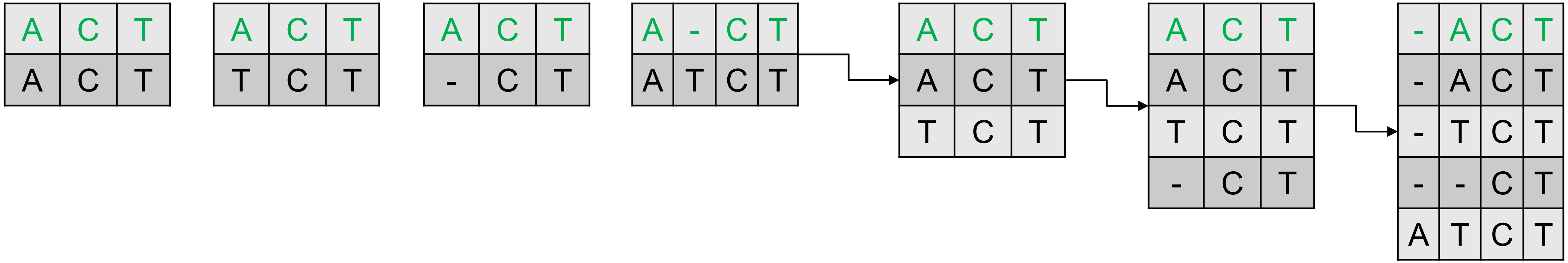
Star alignment

Limitations

Let us take four sequences. S1: ACT, S2: TCT, S3: CT and S4: ATCT. Consider S1: ACT is at the center. Order of pairing: (C, S2), (C, S3), (C, S4) and (C, S1).


A C T / T C T
A C T / - C T
A - C T / A T C T
A C T / A C T
A C T / T C T / - C T
A - C T / T - C T / - - C T / A T C T
A - C T / T - C T / - - C T / A T C T / A - C T

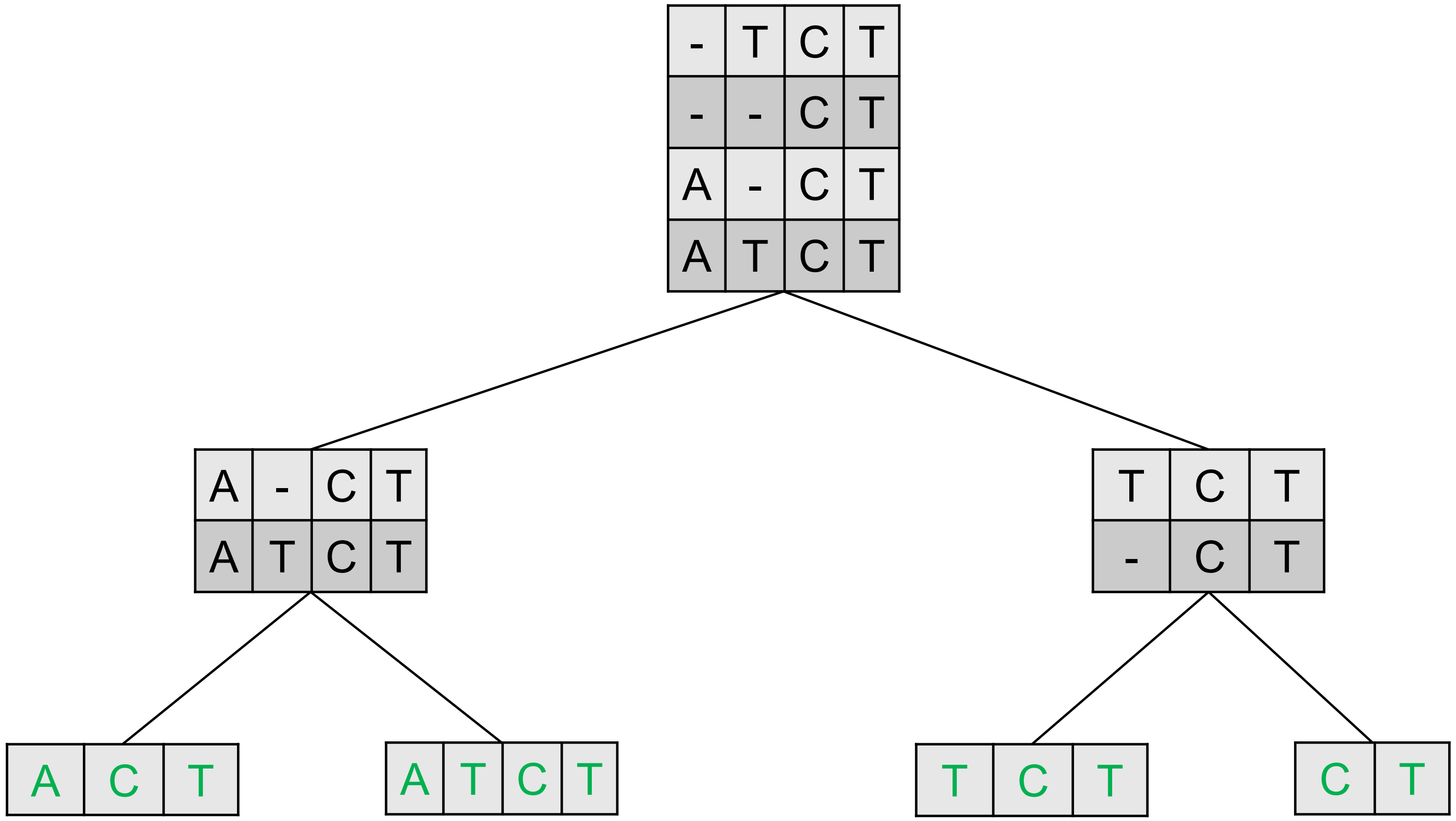
Order of pairing: (C, S1), (C, S2), (C, S3) and (C, S4).



Progressive alignment

- Idea is that (a) first align pair(s) of most closely related sequences and (b) then interactively align the alignments to obtain an alignment for larger number of sequences.

- Let us take four sequences. S1: ACT, S2: TCT, S3: CT and S4: ATCT.



Progressive Alignment

Progressive alignment: Steps

Step1: Calculate a distance matrix, representing the distance between each pair of sequences.

- Perform a pairwise alignment between each pair of sequences (dynamical programming or faster heuristic algorithm). From each pairwise alignment, calculate the distance between the two sequences.

$$d_{i,j} = \frac{s_{i,j}}{L_{i,j}}$$

$d_{i,j}$: distance between sequences *i and j*
$L_{i,j}$: length of the alignment
$s_{i,j}$: number of substitutions

Remarks
- Gaps are not taken into account in the distance metric.
- For n sequences, there are n(n–1)/2 pairwise alignments.

	seq 1	seq 2	...	seq n
seq 1	d1,1	d1,2	...	d1,n
seq 2	d2,1	d2,2	...	d2,n
...	...	...	...	...
seq n	dn,1	dn,2	...	dn,n

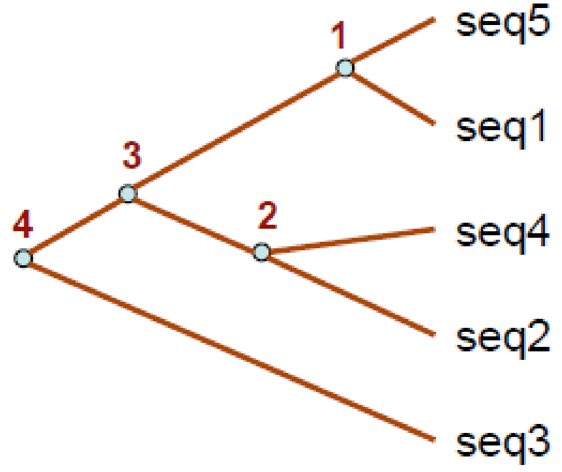
Remarks
- The matrix is symmetrical ($d_{i,j} = d_{j,i}$)
- Diagonal elements are null ($d_{i,i} = 0$)

Progressive alignment: Steps

Step 2: From this matrix, build a phylogenetic tree.

- A phylogenetic tree is calculated from the distance matrix:
  - First regroup the two closest sequences (e.g. **1**).
  - Recalculate the matrix (Note: there are multiple ways to do this).
  - Next the two closest sequences (e.g. **2**).
  - Repeat these previous steps until all the sequences are merged (e.g. **3** and **4**).

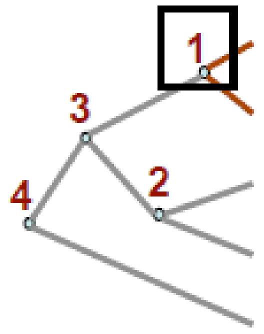
- This tree will then be used as guide to determine the order of incorporation of the sequences in the multiple alignment.



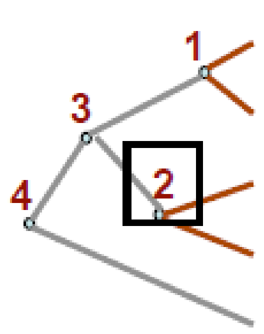
Progressive alignment: Steps

Step 3: Use this tree as guide to progressively align the sequences.

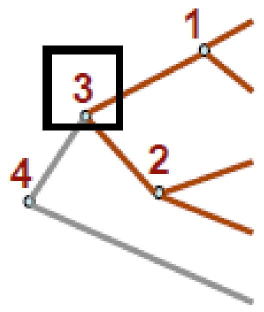
- Build a multiple alignment, by progressively incorporating the sequences according to the guide tree.



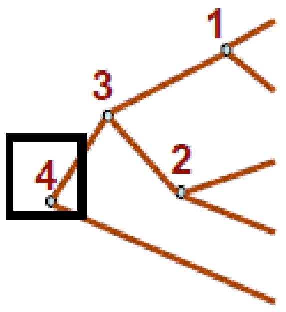
```
Seq5   GATTGTAGTA
Seq1   GATGGTAGTA
Seq4
Seq2
Seq3
```



```
Seq5   GATTGTAGTA
Seq1   GATGGTAGTA
Seq4   GATTGTTC--GTA
Seq2   GATTGTTCGGGTA
Seq3
```



```
Seq5   GATTGTA---GTA
Seq1   GATGGTA---GTA
Seq4   GATTGTTC--GTA
Seq2   GATTGTTCGGGTA
Seq3
```



```
Seq5   GATTGTA-----GTA
Seq1   GATGGTA-----GTA
Seq4   GATTGTTC----GTA
Seq2   GATTGTTCGG--GTA
Seq3   GATGGTAGGCGTGTA
```

Progressive alignment: merging a group of sequences

Pair group method using arithmetic mean (PGMA)

- Find two closest nodes (u, v).
- Create a parent (hypothetical) node w.
- Calculate the distance between w and a new node x as follows

$$\mathbf{D(w, x) = [D(u, x) + D(v, x)]/2}$$

Weighted pair group method using arithmetic mean (WPGMA)

- Find two closest nodes (u, v).
- Create a parent (hypothetical) node w.
- Calculate the distance between w and a new node x as follows

$$\mathbf{D(w, x) = w1 \times D(u, x) + w2 \times D(v, x)}$$

Progressive alignment: merging a group of sequences

Unweighted pair group method using arithmetic mean (UPGMA)

- Find two closest nodes (u, v).
- Create a parent (hypothetical) node w.
- Calculate the distance between w and a new node x as follows

D(w, x) = a(u) x D(u, x) + b(v) x D(v, x)

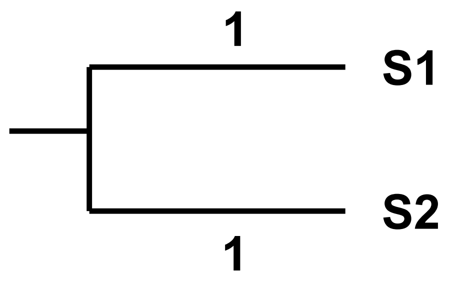
where a(u) = m(u)/[(m(u) + m(v)]
and m(u) = # of leaves under the node u.

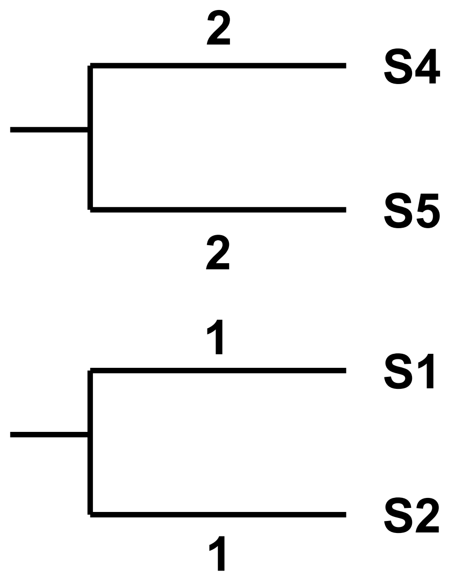
Progressive alignment: Example

Let us consider six sequences S1, S2, S3, S4, S5 and S6 for which the distance matrix is as follows:

	S1	S2	S3	S4	S5	S6
S1	-					
S2	2	-				
S3	4	4	-			
S4	6	6	6	-		
S5	6	6	6	4	-	
S6	8	8	8	8	8	-

	(S1, S2)	S3	S4	S5	S6
(S1, S2)	-				
S3	4	-			
S4	6	6	-		
S5	6	6	4	-	
S6	8	8	8	8	-

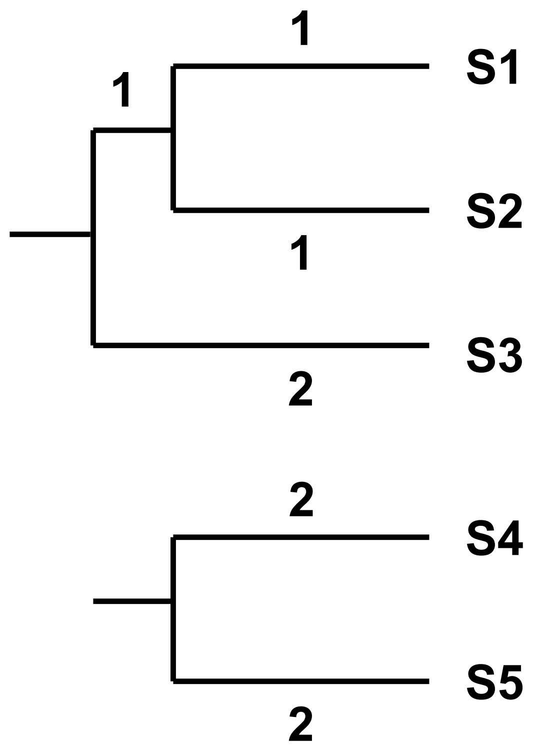


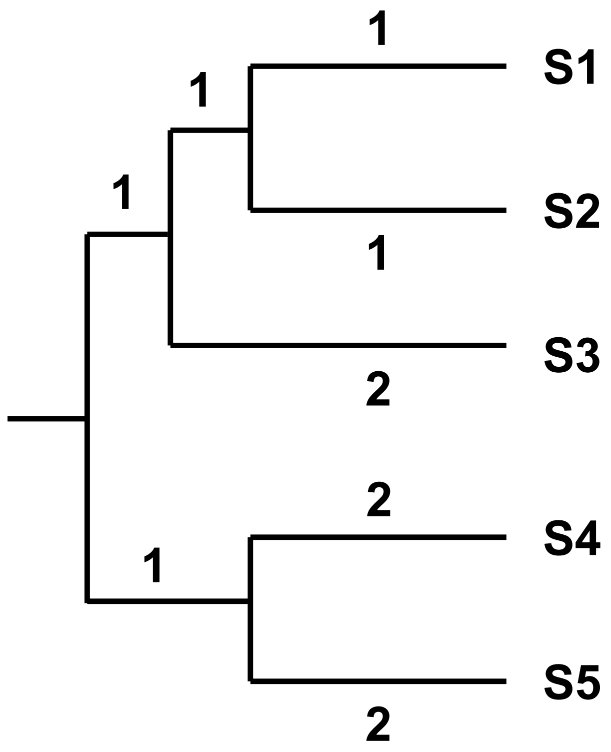


Progressive alignment: Example

	(S1, S2)	S3	(S4, S5)	S6
(S1, S2)	-			
S3	4	-		
(S4, S5)	6	6	-	
S6	8	8	8	-

	(S1, S2, S3)	(S4, S5)	S6
(S1, S2, S3)	-		
(S4, S5)	6	-	
S6	8	8	-

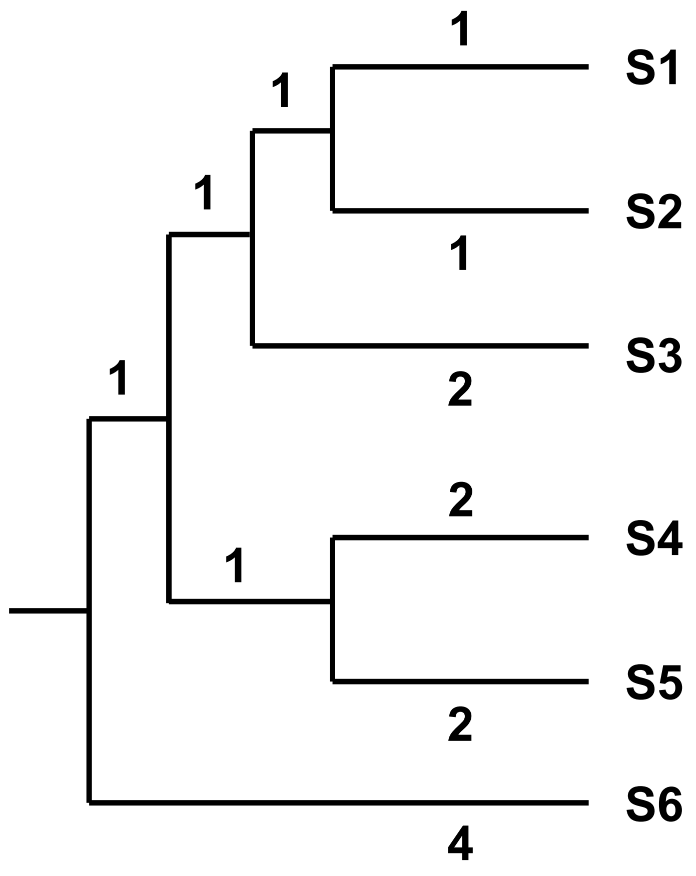




Progressive alignment: Example

	(S1, S2, S3, S4, S5)	S6
(S1, S2, S3, S4, S5)	-	
S6	8	

S1
S2
S3
S4
S5
S6



Progressive alignment: limitations

- This is a heuristic method. It is a practically tractable approach, but it cannot guarantee to return the optimal solution.
- The choice of the sequences (adding or removing one sequence may affect the overall alignment)
- The order of the sequences (i.e. the guide tree) => depends on the tree algorithm (UPGMA, NJ, etc.)
- The scoring parameters (substitution matrices gap penalties, etc.) also affect the alignments.

As a first step, a traditional progressive aligner calculates all N(N-1)/2 pairwise distances amongst all N input sequences. This requires memory and time proportional to N^2 for N sequences. Construction of the guide tree, usually has an additional time complexity of (N^2) to (N^3), depending on the algorithm used and its implementation. This may be computationally too demanding for much more than 10,000 sequences.

For example, with 100,000 sequences, we need to compute approximately 5 billion distances to construct a complete distance matrix as needed by standard implementations of Unweighted Pair Group Method of Arithmetic Mean (UPGMA) or Neighbor-Joining (NJ). Even if the sequences are short, and pair-wise distance calculations can be done relatively quickly, say at a rate of 5000 per second, this still requires of the order of 1 million seconds (11.57 days) of CPU time. Just to store the distance matrix is then difficult as it will take up of the order of 20 GB of disk space and/or memory.

Clustal Omega

In Clustal Omega, the main improvements over ClustalW are (1) use of the mBed algorithm for creating guide trees of any size and (2) a very accurate profile–profile aligner, based on the HHalign package.

The mBed algorithm reduces the time and memory complexity for guide tree calculation from $O(N^2)$ to $O(N(\log(N))^2)$.

The principle of mBed is to first estimate the distance between each sequence and a tiny subset of sequences selected on the basis of their length. For each sequence, the result is a distance vector that can be used to run a **hierarchical k-means clustering**, whose relatively low complexity (NlogN) allows large data sets of 10,000 sequences or more to be aligned.

This is achieved by calculating the pairwise distances of all N sequences with respect to $(\log(N))^2$ randomly chosen seed sequences only.

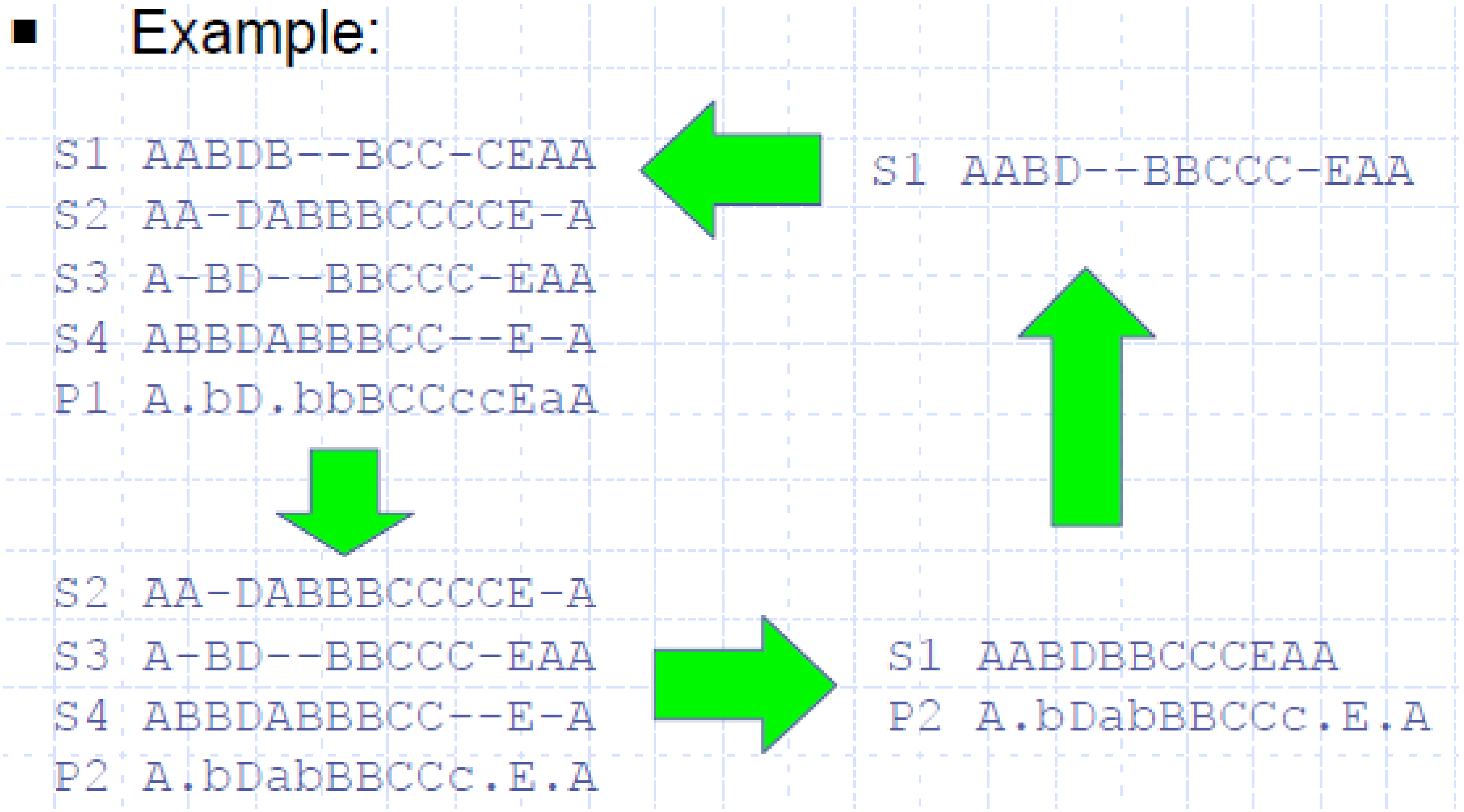
The mBed algorithm allows guide trees of hundreds of thousands of sequences to be made by restricting the calculation of sequence alignment scores to Nlog(N).

The pairwise distances are then clustered, using a bisecting k-means algorithm. Groups of sequences are bisected until a certain threshold for the cluster size is reached (e.g. 100).

The main algorithmic change over ClustalW is a new profile–profile engine, based on the HHalign software.

Iterative MSA

- Select the **highest scoring pair-wise alignment** to compute initial profile.
- Find a sequence that is most similar to the profile and align with profile. Repeat this until all sequences are included in MSA.
- Iterate the following process until convergence: select a sequence X_k and align it against the profile of the other sequences.



- Example:

```
S1 AABDB--BCC-CEAA
S2 AA-DABBBCCCCE-A
S3 A-BD--BBCCC-EAA
S4 ABBDABBBCC--E-A
P1 A.bD.bbBCCccEaA
```

```
S2 AA-DABBBCCCCE-A
S3 A-BD--BBCCC-EAA
S4 ABBDABBBCC--E-A
P2 A.bDabBBCCc.E.A
```

```
S1 AABDBBCCCEAA
P2 A.bDabBBCCc.E.A
```

```
S1 AABD--BBCCC-EAA
```

T-COFFEE: Tree-based Consistency Objective Function For alignmEnt Evaluation

The most common strategy to avoid local minima during a progressive alignment is the use of **consistency**. The rationale of consistency is relatively straightforward: given a set of sequences and their associated pairwise alignments, treated as constraints, scores for matching pairs of residues are re-estimated so as to deliver pairwise alignments more likely to be compatible with a globally optimal MSA.

The first strategy involving such a re-estimation of match costs was reported by Morgenstern as overlapping weights. This scheme later inspired the **T-Coffee** scoring scheme that has become the **archetypical progressive consistency-based aligner**.

Optimizing an alignment against a set of pre-defined constraint is known as the **Maximum Weight Trace problem**.

The **T-Coffee algorithm is a heuristic approach** that involves re-estimating the initial costs of every potential pairwise match by taking into account its compatibility with the rest of the pairwise alignment.

In a consistency-based algorithm, the most critical parameter is the **primary library**. Given a set of sequences, the primary library is a collection of all possible pairwise sequence comparisons. This library is used to define the consistency-based objective function. In the original T-Coffee, the library was a compilation of all pairs of residues found aligned in the entire pairwise local and global alignments. These residue pairs were weighted according to the estimated reliability of their source alignments.

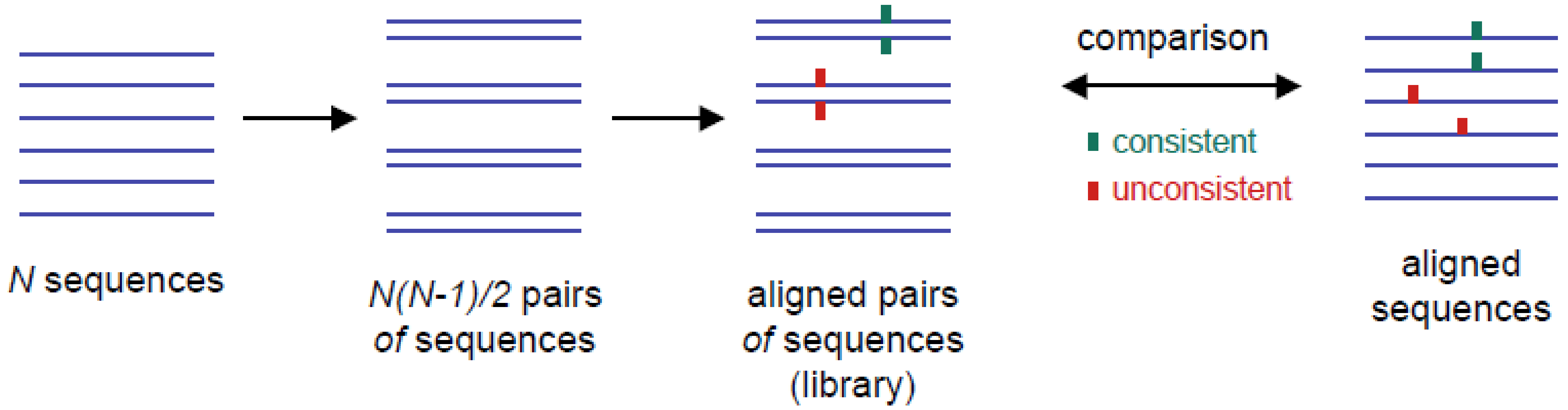
Chatzou et al., 2016, Briefings in Bioinformatics.

T-COFFEE: Tree-based Consistency Objective Function For alignmEnt Evaluation

Key Idea: improve scoring to reduce sensitivity, to optimize the consistency of the alignment by comparing the multiple alignment with all the pair-wise alignments (library).

How:

- Pre-compute library of pair-wise alignments and scores.
- Score is based on both global as well as local alignment.
- Incorporate structure data.


N sequences
N(N-1)/2 pairs *of* sequences
aligned pairs *of* sequences (library)
comparison
🟩 consistent
🟥 unconsistent
aligned sequences

$$\text{score} = \frac{\text{number of consistent residue pairs}}{\text{total number of residue pairs}}$$

MUSCLE: a multiple sequence alignment method with reduced time and space complexity

Algorithm overview
MUSCLE has three stages. At the completion of each stage, a multiple alignment is available and the algorithm can be terminated.

Stage 1: draft progressive
The first stage builds a progressive alignment.

Similarity measure: The similarity of each pair of sequences is computed, either using *k*-mer counting or by constructing a global alignment of the pair and determining the fractional identity.

Distance estimate: A triangular distance matrix is computed from the pairwise similarities.

Tree construction: A tree is constructed from the distance matrix using UPGMA or neighbor-joining, and a root is identified.

Progressive alignment: A progressive alignment is built by following the branching order of the tree, yielding a multiple alignment of all input sequences at the root.

MUSCLE: a multiple sequence alignment method with reduced time and space complexity

Stage 2: improved progressive
The second stage attempts to improve the tree and builds a new progressive alignment according to this tree. This stage may be iterated.

Similarity measure: The similarity of each pair of sequences is computed using **fractional identity computed from their mutual alignment in the current multiple alignment**.

Tree construction: A tree is constructed by computing a **Kimura distance matrix** and applying a **clustering method** to this matrix.

Tree comparison: The previous and new trees are compared, identifying the set of internal nodes for which the branching order has changed. If Stage 2 has executed more than once and the number of changed nodes has not decreased, the process of improving the tree is considered to have converged and iteration terminates.

Progressive alignment: A new progressive alignment is built. The existing alignment is retained of each subtree for which the branching order is unchanged; new alignments are created for the (possibly empty) set of changed nodes. When the alignment at the root is completed, the algorithm may terminate.

MUSCLE: a multiple sequence alignment method with reduced time and space complexity

Stage 3: refinement
The third stage performs iterative refinement using a variant of tree-dependent restricted partitioning.

Choice of bipartition: An edge is deleted from the tree, dividing the sequences into two disjoint subsets (a bipartition). Edges are visiting in order of decreasing distance from the root.

Profile extraction: The profile (multiple alignment) of each subset is extracted from the current multiple alignment. Columns containing no residues (i.e., indels only) are discarded.

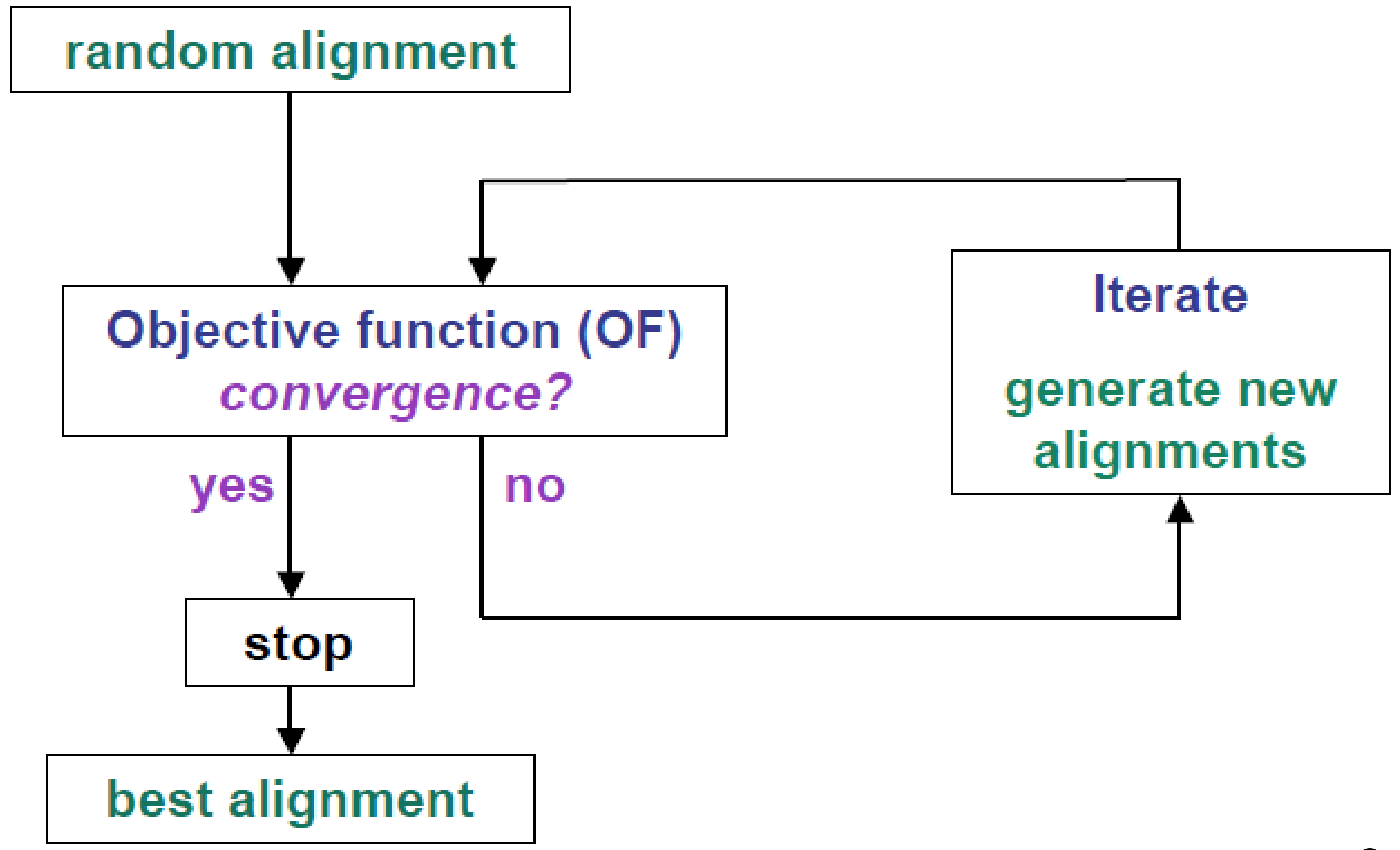
Re-alignment: The two profiles obtained are re-aligned to each other using profile-profile alignment.

Accept/reject: The sum-of-pair score of the multiple alignment implied by the new profile-profile alignment is computed. If the score increases, the new alignment is retained, otherwise it is discarded. If all edges have been visited without a change being retained, or if a user-defined maximum number of iterations has been reached, the algorithm is terminated. Visiting edges in order of decreasing distance from the root has the effect of first realigning individual sequences, then closely related groups.

Edgar, 2004, BMC Bioinformatics.

MSA by genetic algorithm

Principle:



How to define the objective function?

Weighted sum-of-pairs

$$\text{ALIGNMENT COST}(A) = \sum_{i=2}^{N} \sum_{j=1}^{i-1} W_{i,j}\, \text{COST}(A_i, A_j)$$

How to iterate (i.e. how to modify the alignment)?

Our goal is to generate an alignment with the lower cost as possible (= alignment with the highest score).

Scoring Multiple Sequence Alignments

How to score multiple sequence alignment?

- Number of matches (multiple longest common subsequence score)
- Sum of pairs (SP-Score)
- Entropy score

Multiple LCS Score

- A column is a “match” if all the letters in the column are the same

```
AAA
AAA
AAT
ATC
```

- Only good for very similar sequences

How to score multiple sequence alignment?

Sum of pairs

1. Scores each column according to a "sum-of-pairs" (SP) function using a substitution/scoring matrix.
2. Assumes independence between the columns.

$$S(m_i) = \sum_{k<l} s\left(m_i^k, m_i^l\right)$$

m_i^k = residue in sequence k in column i
S(a,b) = score from a substitution matrix
(PAM or BLOSUM for example)

$$S(m) = \sum_i S(m_i)$$

S(m) = score of the whole alignment m
$S(m_i)$ = score of column i in this alignment

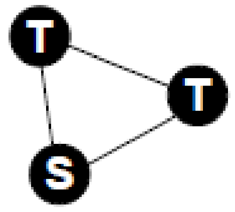
How to score multiple sequence alignment?

Sum of pairs: Example

```
123456789...              L
VTISCTGSSSNIGAG-NHVKWYQQLPG
VTISCTGTSSNIGS--ITVNWYQQLPG
LRLSCSVSGFIFSS--YAMYWVRQAPG
```

$S_6 = s(T,T) + s(T,S) + s(T,S)$

$S_{15} = s(G,-) + s(G,-) + s(-,-)$



$S_{alignment} = S_1 + S_2 + S_3 + \ldots + S_L$

A score is calculated for each column, using scoring matrices and gap penalties. Note that here a gap-gap penalty should also be specified.

The alignment score is the sum of the column scores.

How to score multiple sequence alignment?

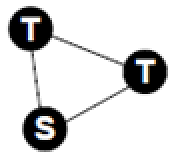
Normalized sum of pairs

$$S(m_i) = \frac{1}{N}\sum_{k<l} s\left(m_i^k, m_i^l\right)$$

```
123456789...                L
VTISCTGSSSNIGAG-NHVKWYQQLPG
VTISCTGTSSNIGS--ITVNWYQQLPG
LRLSCSVSGFIFSS--YAMYWVRQAPG
```

S_6=s(T,T)+s(T,S)+s(T,S)

S_{15}=s(G,-)+s(G,-)+s(-,-)



A score is calculated for each column, using scoring matrices and gap penalties. Note that here a gap-gap penalty should also be specified.

$S_{alignment}$ = 1/N (S_1 + S_2 + … + S_L)

The alignment score is the sum of the column scores.

How to score multiple sequence alignment?

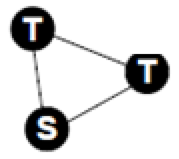
Weighted normalized sum of pairs

$$S(m_i) = \frac{1}{N} \sum_{k<l} w_{k,l}\, s\left(m_i^k, m_i^l\right)$$

```
123456789...                L
VTISCTGSSSNIGAG-NHVKWYQQLPG   W1
VTISCTGTSSNIGS--ITVNWYQQLPG   W2
LRLSCSVSGFIFSS--YAMYWVRQAPG   W3
```

S_6 = W1 x W2 x s(T, T) +
W1 x W3 x s(T, S) +
W2 x W3 x s(T, S)

S_{15} = W1 x W2 x s(G, -) +
W1 x W3 x s(G, -) +
W2 x W3 x s(-, -)



$S_{alignment}$ = 1/N (S_1 + S_2 + … + S_L)

A score is calculated for each column, using scoring matrices and gap penalties. Note that here a gap-gap penalty should also be specified.

The alignment score is the sum of the column scores.

How to score multiple sequence alignment?

Star consensus

$$S(m_i) = \sum_k s(M, m_i^k)$$

	1	2	3	4	5	6	7	8	9	10
S1	V	T	I	S	C	T	G	S	S	S
S2	V	T	I	S	C	T	G	T	S	S
S3	L	R	L	S	C	S	V	S	G	F
Consensus	V	T	I	S	C	T	G	S	S	S

$S_6 = s(T, T) + s(T, T) + s(T, S)$

How to score multiple sequence alignment?

Entropy method

Entropy for a multiple sequence alignment is the sum of entropies of its columns:

$$Entropy(MSA) = -\sum_{Over\,all\,columns} \; \sum_{X=A,T,G,C} p_X \log_2 (p_X)$$

Best case

$$E\begin{pmatrix} A \\ A \\ A \\ A \end{pmatrix} = 0$$

Worst case

$$E\begin{pmatrix} A \\ T \\ G \\ C \end{pmatrix} = -\sum_{X=A,T,G,C} \frac{1}{4} \log_2 \frac{1}{4} = -4(\frac{1}{4} * -2) = 2$$

How to score multiple sequence alignment?

Entropy method

$$-\sum_{X=A,T,G,C} p_X \log_2 p_X$$

Let us consider the following multiple DNA sequence alignment

A	A	A	A
A	A	A	C
A	A	C	G
A	T	T	T

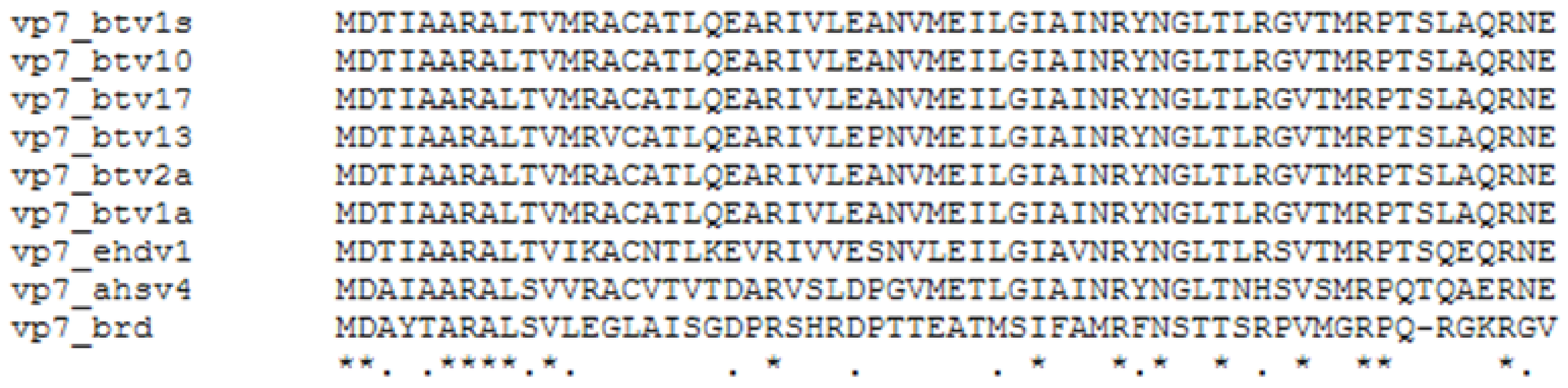
	1	**2**	**3**	**4**
A	1	0.75	0.50	0.25
C	0	0	0.25	0.25
G	0	0	0	0.25
T	0	0.25	0.25	0.25

	1	**2**	**3**	**4**
Entropy	-[1 x $\log_2 1$ + 0 + 0 + 0] = 0	-[0.75 x $\log_2$ 0.75 + 0 + 0 + 0.25 x $\log_2$ 0.25] = 0.81125	-[0.50 x $\log_2$ 0.50 + 0.25 x $\log_2$ 0.25 + 0 + 0.25 x $\log_2$ 0.25] = 1.50	-[0.50 x $\log_2$ 0.50 + 0.25 x $\log_2$ 0.25 + 0.25 x $\log_2$ 0.25 + 0.25 x $\log_2$ 0.25] = 2.00

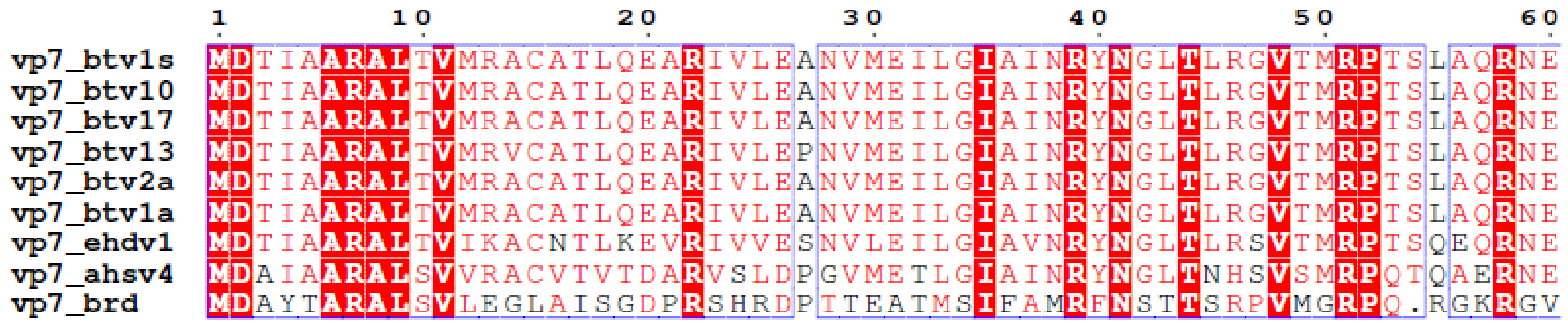
Editing and displaying multiple sequence alignments

Editing and displaying multiple sequence alignments

- Sequence editors are used for: manual alignment/editing of sequences, visualization of data, graphical enhancement of data for presentations.
- **CINEMA** (Colour INteractive Editor for Multiple Alignments)
- **ESPript** (Easy Sequencing in Postscript)

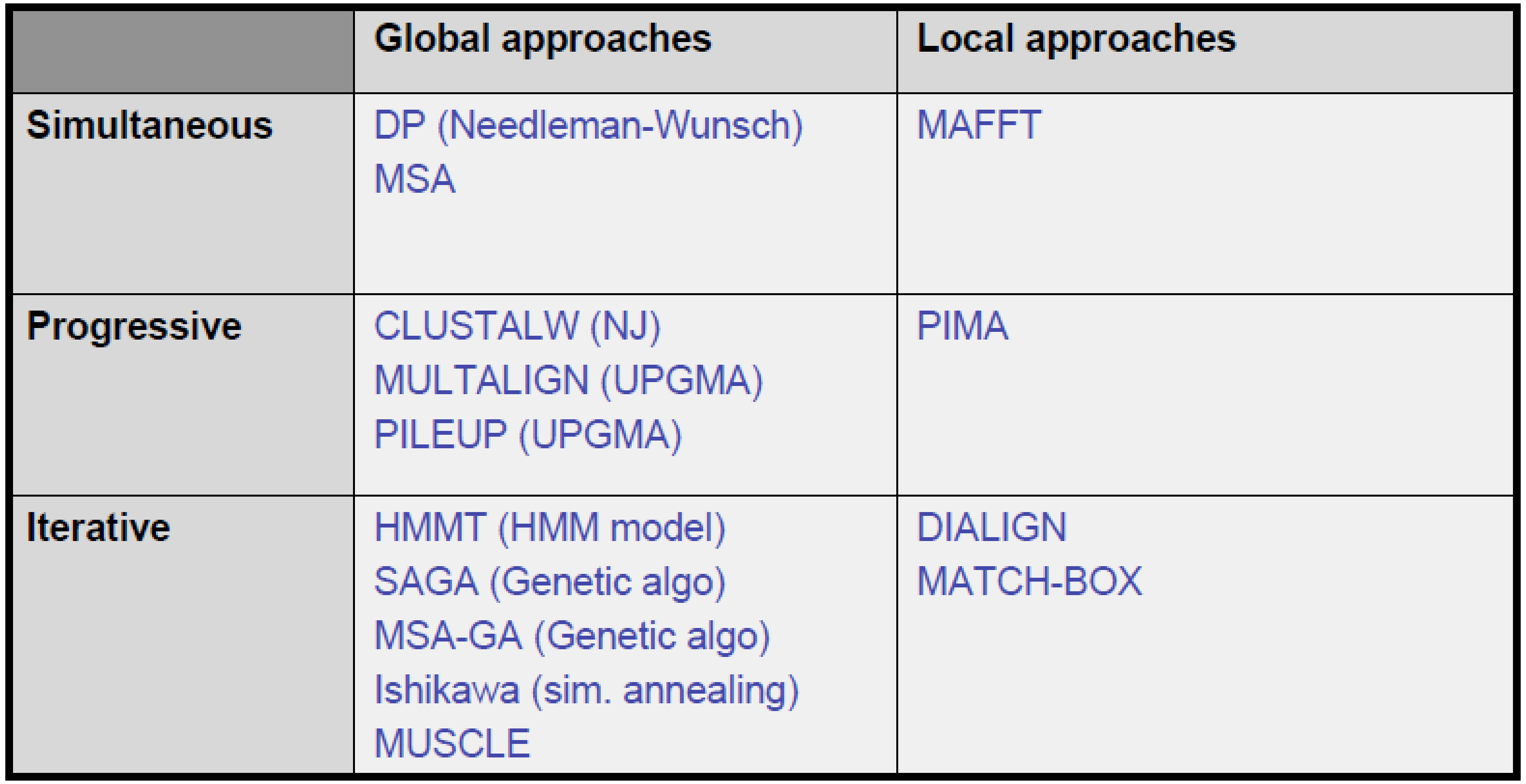


```
vp7_btv1s       MDTIAARALTVMRACATLQEARIVLEANVMEILGIAINRYNGLTLRGVTMRPTSLAQRNE
vp7_btv10       MDTIAARALTVMRACATLQEARIVLEANVMEILGIAINRYNGLTLRGVTMRPTSLAQRNE
vp7_btv17       MDTIAARALTVMRACATLQEARIVLEANVMEILGIAINRYNGLTLRGVTMRPTSLAQRNE
vp7_btv13       MDTIAARALTVMRVCATLQEARIVLEPNVMEILGIAINRYNGLTLRGVTMRPTSLAQRNE
vp7_btv2a       MDTIAARALTVMRACATLQEARIVLEANVMEILGIAINRYNGLTLRGVTMRPTSLAQRNE
vp7_btv1a       MDTIAARALTVMRACATLQEARIVLEANVMEILGIAINRYNGLTLRGVTMRPTSLAQRNE
vp7_ehdv1       MDTIAARALTVIKACNTLKEVRIVVESNVLEILGIAVNRYNGLTLRSVTMRPTSQEQRNE
vp7_ahsv4       MDAIAARALSVVRACVTVTDARVSLDPGVMETLGIAINRYNGLTNHSVSMRPQTQAERNE
vp7_brd         MDAYTARALSVLEGLAISGDPRSHRDPTTEATMSIFAMRFNSTTSRPVMGRPQ-RGKRGV
                **. .****.*.       . *   .      . *   *.*  * . *  **     *.
```



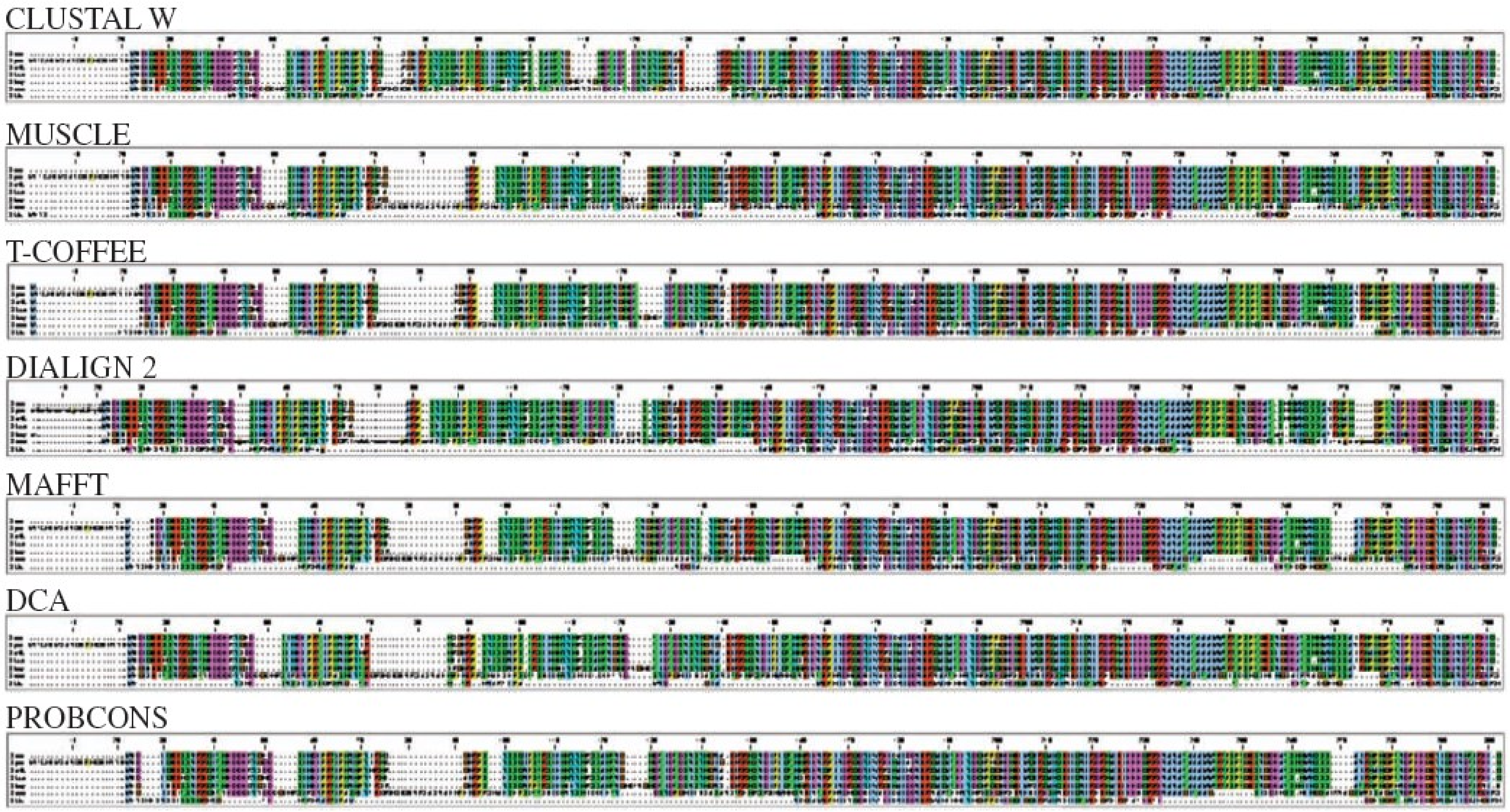
```
                1        10        20        30        40        50        60
vp7_btv1s       MDTIAARALTVMRACATLQEARIVLEANVMEILGIAINRYNGLTLRGVTMRPTSLAQRNE
vp7_btv10       MDTIAARALTVMRACATLQEARIVLEANVMEILGIAINRYNGLTLRGVTMRPTSLAQRNE
vp7_btv17       MDTIAARALTVMRACATLQEARIVLEANVMEILGIAINRYNGLTLRGVTMRPTSLAQRNE
vp7_btv13       MDTIAARALTVMRVCATLQEARIVLEPNVMEILGIAINRYNGLTLRGVTMRPTSLAQRNE
vp7_btv2a       MDTIAARALTVMRACATLQEARIVLEANVMEILGIAINRYNGLTLRGVTMRPTSLAQRNE
vp7_btv1a       MDTIAARALTVMRACATLQEARIVLEANVMEILGIAINRYNGLTLRGVTMRPTSLAQRNE
vp7_ehdv1       MDTIAARALTVIKACNTLKEVRIVVESNVLEILGIAVNRYNGLTLRSVTMRPTSQEQRNE
vp7_ahsv4       MDAIAARALSVVRACVTVTDARVSLDPGVMETLGIAINRYNGLTNHSVSMRPQTQAERNE
vp7_brd         MDAYTARALSVLEGLAISGDPRSHRDPTTEATMSIFAMRFNSTTSRPVMGRPQ.RGKRGV
```

Comparison of multiple sequence alignment algorithms



	Global approaches	Local approaches
Simultaneous	DP (Needleman-Wunsch) MSA	MAFFT
Progressive	CLUSTALW (NJ) MULTALIGN (UPGMA) PILEUP (UPGMA)	PIMA
Iterative	HMMT (HMM model) SAGA (Genetic algo) MSA-GA (Genetic algo) Ishikawa (sim. annealing) MUSCLE	DIALIGN MATCH-BOX

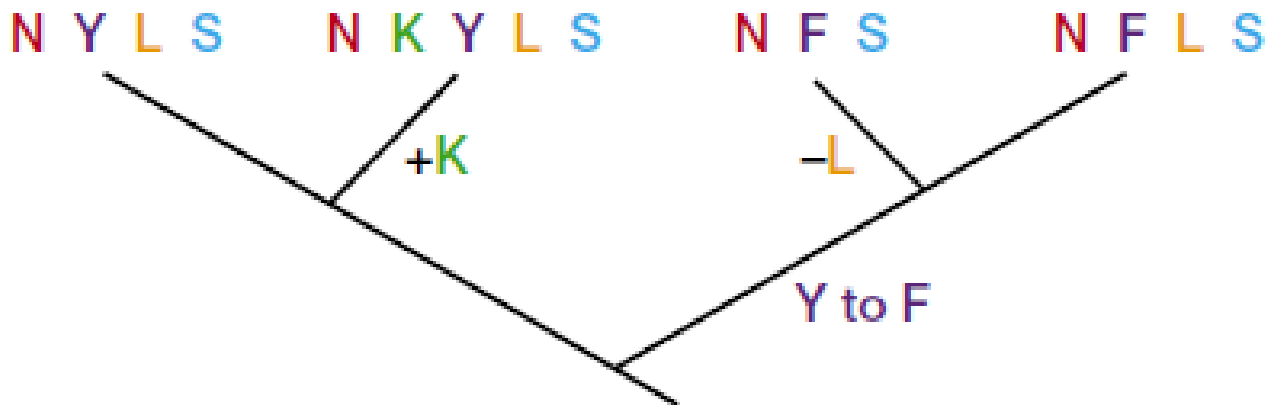
Different alignments are obtained with different programs



Wong et al., 2008, *Science*, 319, 473-476

Relationship of multiple sequence alignment to phylogenetic analysis

seqA	N	•	F	L	S
seqB	N	•	F	–	S
seqC	N	K	Y	L	S
seqD	N	•	Y	L	S



Thank You