Sequence analysis and genomics

3. Multiple Sequence Alignments

Dr. Katja Nowick

Group leader TFome and Transcriptome Evolution
Bioinformatics group
Paul-Flechsig-Institute for Brain Research
www. nowicklab.info
nowick@bioinf.uni-leipzig.de

What is an alignment?

- Arranging sequences in a way to identify regions of similarity
- DNA, RNA, protein sequences
- Gaps are inserted, so that identical characters are in the same column
- The more common letters in the sequences the higher their similarity
- The number of non-matches is the *edit sequence*
- Goal: find the longest common subsequence or minimize number of necessary edits
- In example: 4 matches, edit distance = 3 (delete J, T→R, add N)

```
JAEHTE
AEHREN
```

```
JAEHTE-
 |||  |
-AEHREN
```

A simple example

```
ATCTCGAG
ATCTCCGAG
AGCTCGAG
ATCCGAG
```

```
ATCT-CGAG
ATCTCCGAG
AGCT-CGAG
ATC--CGAG
```

Editing: Introducing of substitutions, insertion, deletions

Global vs. Local Alignments

- treat the two sequences as potentially equivalent
- attempt to align every residue in every sequence
- most useful when the sequences in the query set are similar and of roughly equal size
- Can still have gaps
- e.g. for homologous genes

- Sequences may or not be related
- Attempts to find similar parts (substring) in a sequence
- more useful for dissimilar sequences that are suspected to contain regions of similarity or similar sequence motifs within their larger sequence context.
- e.g. looking for conserved domains or finding a sequence in the genome

```
FTFTALILLAVAV
F--TAL-LLA-AV
```

```
FTFTALILL-AVAV
--FTAL-LLAAV--
```

With sufficiently similar sequences, there is no difference between local and global alignments

Scoring alignments

- Substitution matrix to assign scores to matches or mismatches:
  for amino acids, different substitutions are usually rated differently
  for DNA and RNA, in practice often simply assign a positive match and a negative mismatch score

- Gap penalty for matching an amino acid/nucleotide in one sequence to a gap in the other

- The score of an alignment is the sum of the scores for each position plus gap costs

```
VAHV---D--DMPNALSALSDLHAHKL
AIQLQVTGVVVTDATLKNLGSVHVSKG
```

S(V,A) + s(A,I) + s(H,Q) + s (V,L) + 3 gaps +....

Scoring matrix

- Scoring matrix (costs for editing):
  insert value for match, mismatch, or gap
  e.g. match = 1
  mismatch = -1
  gap = -2

Certain matches and mismatches can also cost differently

	A	C	T	G	gap
A	1	-1	-1	-1	-2
C	-1	1	-1	-1	-2
T	-1	-1	1	-1	-2
G	-1	-1	-1	1	-2
gap	-2	-2	-2	-2	1

Needleman–Wunsch algorithm

Example:

String 1: TG
String 2: ATT

Scoring matrix:
match = 1
mismatch = -1
gap = -2

String 2

String 1	i =	0	1	2	3
		$	**A**	**T**	**T**
j = 0	**$**	0	-2	-4	-6
1	**T**	-2	-1	-1	-3
j = 2	**G**	-4	-3	-2	-2

Rules for filling in the matrix:

$$a_{0,0} = 0$$

$$a_{0,j} = a_{0,j-1} + gappenalty$$

$$a_{i,0} = a_{i-1,0} + gappenalty$$

$$a_{i,j} = \max(a_{i-1,j} + gappenalty,\ a_{i,j-1} + gappenalty,\ a_{i-1,j-1} + equal(i,j))$$

Needleman–Wunsch algorithm

Example:

String 1: TG
String 2: ATT

Scoring matrix:
match = 1
mismatch = -1
gap = -2

String 2

String 1	i =	0	1	2	3
		$	**A**	**T**	**T**
j = 0	**$**	**0**	-2	-4	-6
1	**T**	-2	**-1**	**-1**	-3
j = 2	**G**	-4	-3	-2	**-2**

The values in each cell represent the similarity of the alignment of the subsequences
The value in the bottom right corner represents the similarity of the two complete sequences: **-2**

Reconstructing the alignment = Backtracking: Where did I come from?

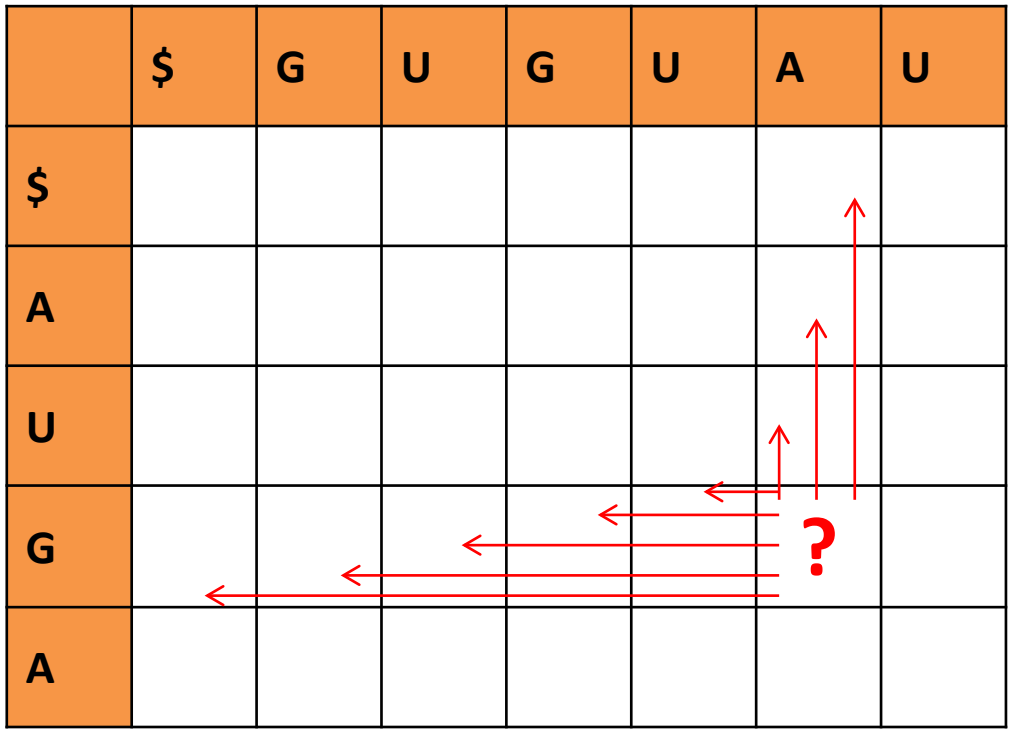
Go backwards always to the field with the highest value
Moving diagonally means match/mismatch
Moving left means insertion in string 1
Moving up means insertion in string 2

String 1: - TG
String 2: ATT
Score: -2 1 -1 = -2

Affine gap costs

String 2

String 1

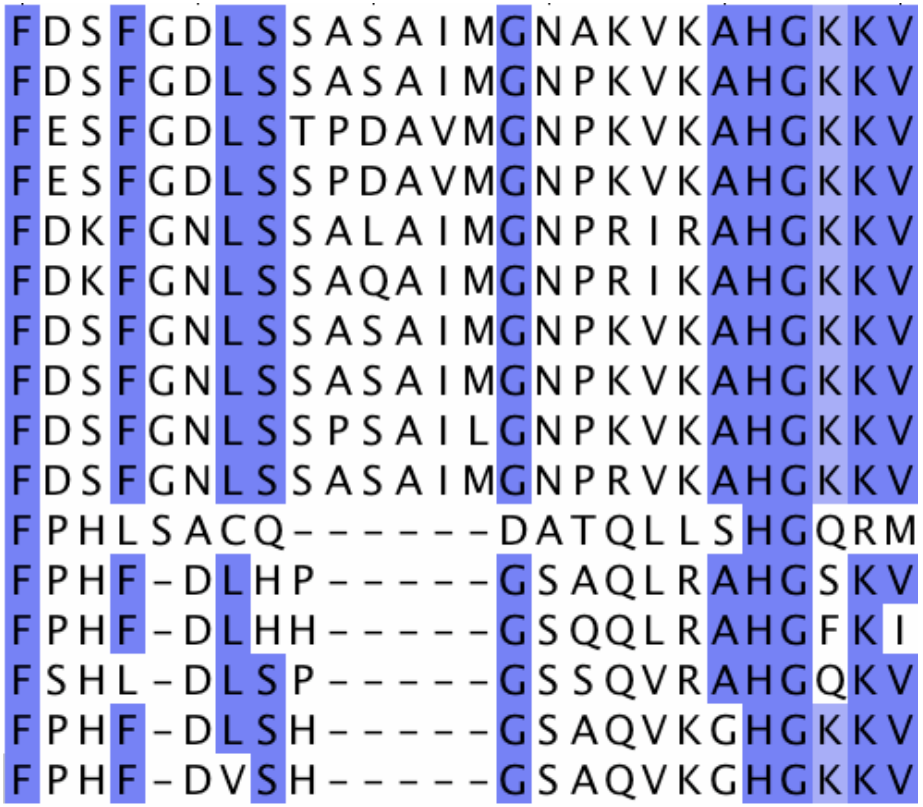


Need to check all these previous alignments and get the maximum score

Multiple Sequence Alignment (MSA)

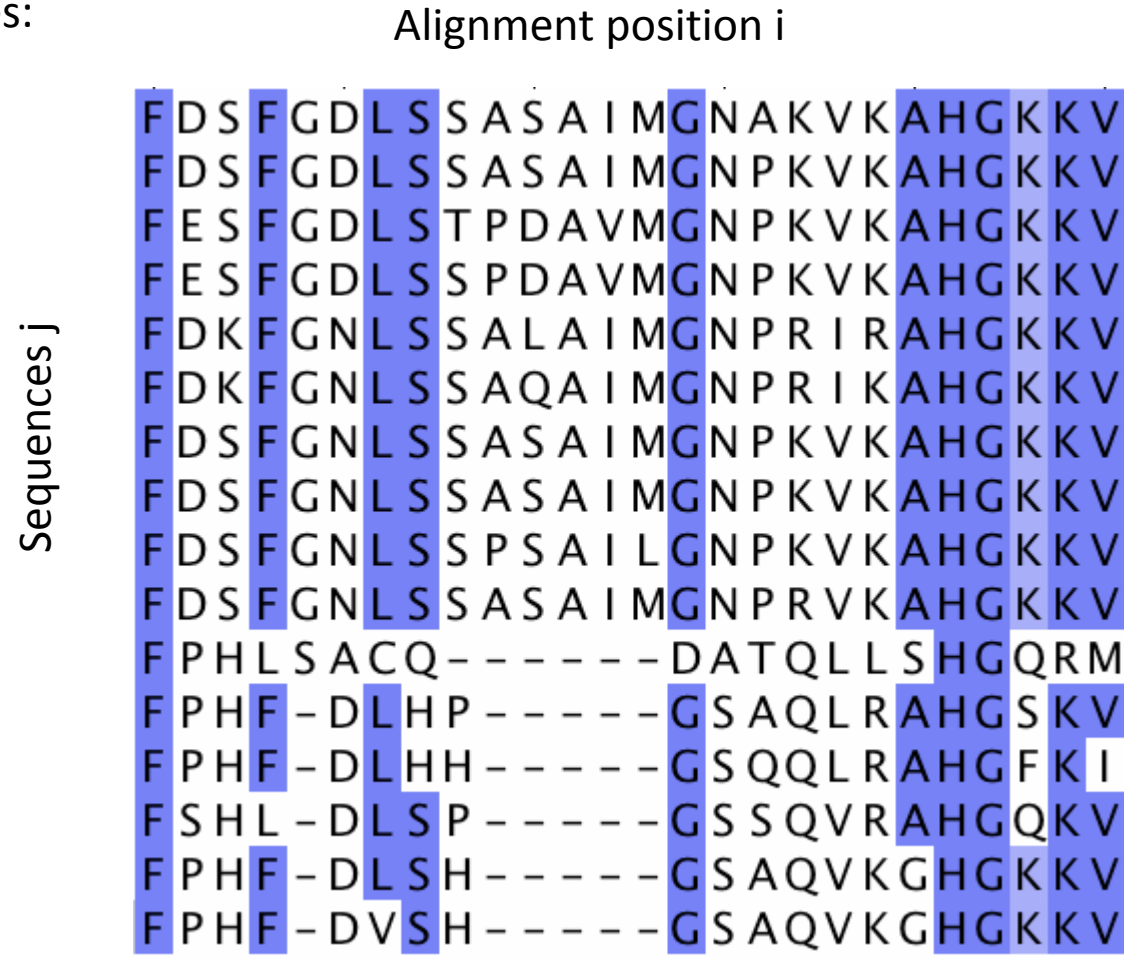
A multiple sequence alignment (MSA) arranges sequences into a rectangular array with the goal that residues in a given column are:

- Homologous, i.e. are derived from a single position in an ancestral sequence
- Superposable, i.e. correspond to a rigid local structural alignment
- Play a common functional role



Multiple Sequence Alignment (MSA)

Matrix for N sequences:



Consensus sequence: FDSFGDLSSASAIMGNPKVKAHGKKV

= the most frequent letter in each column

Applications of MSA

- Phylogenetic tree reconstruction
- Hidden Markov modeling (HMM)
- Secondary and tertiary structure prediction
- Function prediction
- PCR primer design

How to construct a MSA?

```
Sequence 1:     AGAC
Sequence 2:     AC
Sequence 3:     AG
```

Just align 2 sequences and then add the 3rd sequence?

Let's look at the pairwise alignments:

```
Sequence1:   AGAC
Sequence 2:  --AC

Sequence 1:   AGAC
Sequence 3:   AG--

Sequence 2:   AC
Seuqence 3:   AG
```

In which order should we align the sequences?

How to construct a MSA?

Pairwise alignments:

```
Sequence 1:   AGAC
Sequence 2:  --AC

Sequence 1:   AGAC
Sequence 3:   AG--

Sequence 2:   AC
Sequence 3:   AG
```

Possible MSAs:

```
Seq1:  AGAC
Seq2:  --AC
Seq3:  AG--

Seq1:  AGAC
Seq3:  AG--
Seq2:  --AC

Seq2:  AC--
Seq3:  AG--
Seq1:  AGAC

Seq2:  --AC
Seq3:  --AG
Seq1:  AGAC
```

→ In addition to scoring matrix (costs for mismatches, gap penalties)

Order of aligning affects the MSA

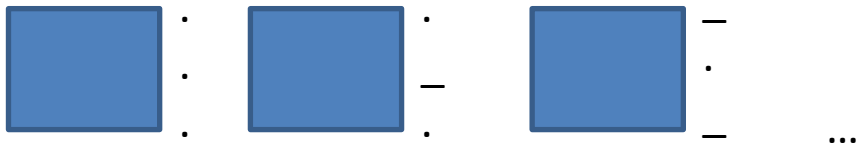
How to construct a MSA?

Extend the pairwise alignment method into a three sequence alignment:

Instead of a 2D matrix, set up a 3D matrix (a cube)
Fill the matrix basically using the same procedure as for 2 sequences
The path then goes diagonally through the cube from one corner to the opposite

Pairwise alignment:

Three sequence alignment:



… can be 0, 1, or 2 gaps per column
(general: # gaps = 0 … N-1)

$$a_{i,j} = \max \begin{cases} a_{i-1,j} + gappenalty(x_i, gap), \\ a_{i,j-1} + gappenalty(gap, y_j), \\ a_{i-1,j-1} + equal(i, j) \end{cases}$$

$$a_{i,j,k} = \max \begin{cases} a_{i-1,j,k} + gappenalty(x_i, gap, gap), \\ a_{i,j-1,k} + gappenalty(gap, y_j, gap), \\ a_{i,j,k-1} + gappenalty(\text{gap}, \text{gap}, z_k), \\ a_{i-1,j-1,k} + gappenalty(x_i, y_j, gap), \\ a_{i-1,j,k-1} + gappenalty(x_i, gap, z_k), \\ a_{i,j-1,k-1} + gappenalty(\text{gap}, y_j, z_k), \\ a_{i-1,j-1,k-1} + equal(i, j, k) \end{cases}$$

How to construct a MSA?

For N sequences: make a matrix with N dimensions
possibilities for # of gaps per column increases to N-1

Problem: computation time: **algorithmic complexity is something like $O(c^{2n})$**,
where c is a constant, and n is the number of sequences

If a pairwise alignment of two sequences takes 1 second,
then 4 sequences would take 10^4 seconds (2.8 hours),
5 sequences 10^6 seconds (11.6 days),
6 sequences 10^8 seconds (3.2 years),
7 sequences 10^{10} seconds (317 years)
…

→ We need a heuristics

Heuristics for MSA

With heuristics not sure to find the optimal MSA

Heavily investigated research field to improve methods in terms of:

1. Computational speed
2. Biological quality:

    How to judge what is the correct alignment?
    3D structure data only available for a handful of proteins
    Benchmarking (see later)

Most methods use the heuristic introduced by Fend & Doolittle 1987: **progressive alignment**

Progressive algorithm

Idea:

Start by aligning the two most similar sequences, then add next similar one and so on

1. Calculate all pairwise alignments and score them
2. Use scores to fill a distance matrix
3. Build a tree based on the distance matrix
4. Tree determines order of adding sequence to the alignment

Progressive algorithm

1. Calculate all pairwise alignments:

The score for each PW alignment is calculated as done before (using scoring matrix)

Example:

String 1: TG
String 2: ATT

Scoring matrix:
match = 1
mismatch = -1
gap = -2

String 2

String 1	i =	0	1	2	3
		$	**A**	**T**	**T**
0	**$**	**0**	-2	-4	-6
1	**T**	-2	**-1**	**-1**	-3
j = 2	**G**	-4	-3	-2	**-2**

Progressive algorithm

1. **Calculate all pairwise alignments:**

The score for each PW alignment is calculated as done before (using scoring matrix)

Example of 3 sequences:

Sequence 1: KEFHNGHT
Sequence 2: KYFHKAGNQHSPT
Sequence 3: KYFHKAGNGHT

KEFHN-G--H--T	**KEFH---NGHT**	**KYFHKAGNQHSPT**
KYFHKAGNQHSPT	**KYFHKAGNGHT**	**KYFHKAGNGH--T**
Score = 11	**Score = 27**	**Score 48**

Progressive algorithm

2. Generate distance matrix:

Calculate distance based on the PW alignment scores

KEFHN-G--H--T **KYFHKAGNQHSPT**	**KEFH---NGHT** **KYFHKAGNGHT**	**KYFHKAGNQHSPT** **KYFHKAGNGH--T**
Score = 11	**Score = 27**	**Score 48**

e.g. Feng and Doolittle method:

$$d = -\log \frac{S - S_{rand}}{S_{max} - S_{rand}}$$

S = alignment score of the two sequences S_1 and S_2

S_{rand} = alignment score of two random sequences of the same length and letter combination as S_1 and S_2 can be calculated using simulations or approximation

S_{max} = mean of scores for aligning S_1 and S_2 with itself

Progressive algorithm

2. Generate distance matrix:

Calculate distance based on the PW alignment scores
Put distances into distance matrix

$$d = -\log \frac{S - S_{rand}}{S_{max} - S_{rand}}$$

	S_1	S_2	S_3	S_4	S_5
S_1					
S_2					
S_3					
S_4					
S_5					

Progressive algorithm

2. Generate distance matrix:

Calculate distance based on the PW alignment scores
Put distances into distance matrix

$$d = -\log \frac{S - S_{rand}}{S_{max} - S_{rand}}$$

	S_1	S_2	S_3	S_4	S_5
S_1	0	a_{12}	a_{13}	a_{14}	a_{15}
S_2	a_{21}	0	a_{23}	a_{24}	a_{25}
S_3	a_{31}	a_{32}	0	a_{34}	a_{35}
S_4	a_{41}	a_{42}	a_{43}	0	a_{45}
S_5	a_{51}	a_{52}	a_{53}	a_{54}	0

Progressive algorithm

2. Generate distance matrix:

Calculate distance based on the PW alignment scores
Put distances into distance matrix

$$d = -\log \frac{S - S_{rand}}{S_{max} - S_{rand}}$$

KEFHN-G--H--T
KYFHKAGNQHSPT

Score = 11
d=1.30

KEFH---NGHT
KYFHKAGNGHT

Score = 27
d= 0.75

KYFHKAGNQHSPT
KYFHKAGNGH--T

Score 48
d=0.39

	S_1	S_2	S_3
S_1	0	1.30	0.75
S_2	1.30	0	0.39
S_3	0.75	0.39	0

Progressive algorithm

3. Cluster sequences based on distance matrix

e.g. by Neighbor Joining method (see some later lecture)
to generate a "guide" tree (rooted binary tree)

KEFHN-G--H--T
KYFHKAGNQHSPT

Score = 11
d=1.30

KEFH---NGHT
KYFHKAGNGHT

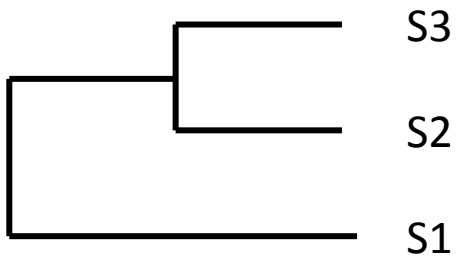
Score = 27
d= 0.75

KYFHKAGNQHSPT
KYFHKAGNGH--T

Score 48
d=0.39

	S_1	S_2	S_3
S_1	0	1.30	0.75
S_2	1.30	0	0.39
S_3	0.75	0.39	0

Guide tree:



Progressive algorithm

4. Add alignments progressively

Start with the two closest leaves, then add sequence by sequence moving towards the root

KEFHN-G--H--T
KYFHKAGNQHSPT

Score = 11
d=1.30

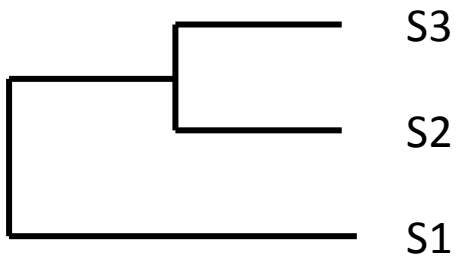
KEFH---NGHT
KYFHKAGNGHT

Score = 27
d= 0.75

KYFHKAGNQHSPT
KYFHKAGNGH--T

Score 48
d=0.39

Guide tree:



S2: **KYFHKAGNQHSPT**
S3: **KYFHKAGNGH--T**
S1: **KEFH---NGH--T**

→ **The MSA is different from PWs**

Adding a sequence to the alignment

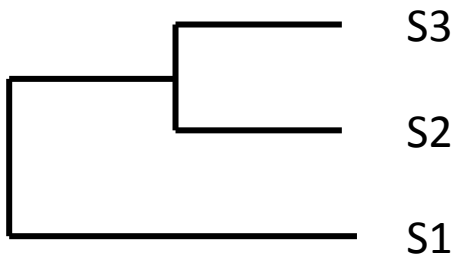
Given the alignment of Seq1 and Seq2 calculate the *Profile* of the alignment

S2: **AUGC-AUG**
S3: **ACGCAA-G**

Profile: **AU̲C̲GCA̲AU̲G**

$p_{\alpha i}$ = # of sequences that have letter α at position i

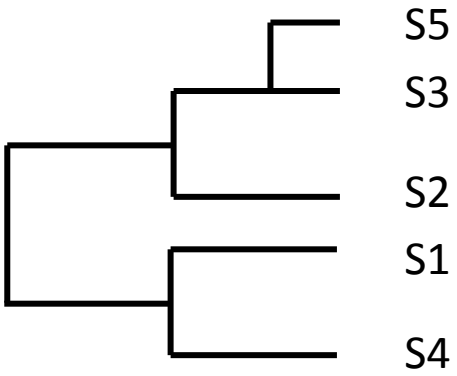
α \ i	1	2	3	4	5	6	7	8
A	2	0	0	0	1	2	0	0
U	0	1	0	0	0	0	1	0
G	0	0	2	0	0	0	0	2
C	0	1	0	2	0	0	0	0
-	0	0	0	0	1	0	1	0



Then align sequence S1 to the profile p

Alignment score is calculated as before for 2 sequences, but score for each position i can be weighted based on $p_{\alpha i}$

Combining profiles

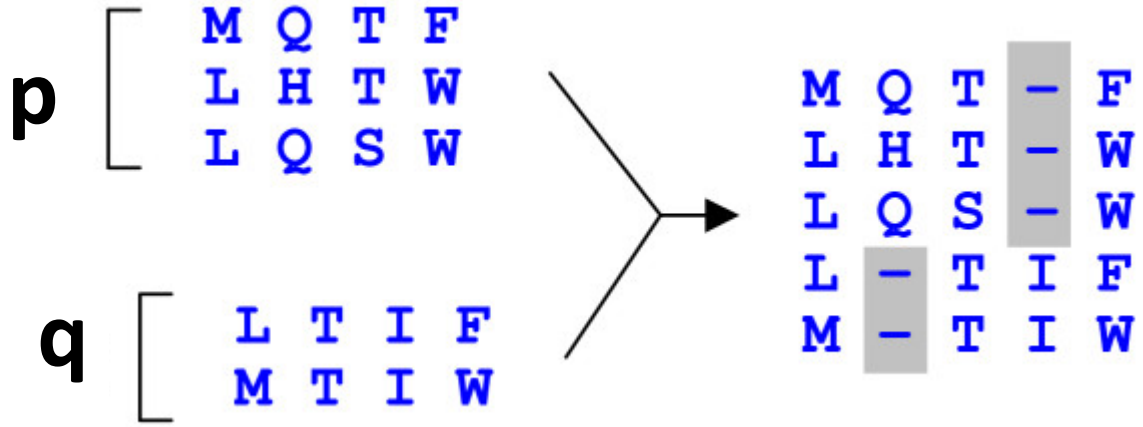


Make profile o from alignment of S3 and S5

Make profile p from alignment of o and S2

Make profile q from alignment of S1 and S4

Alignment profiles p and q



Two profiles are aligned to each other such that columns from p and q are preserved in the result; indels are inserted as needed.

How to treat existing gaps?

When adding sequences to a profile or combining profiles, already existing gaps are kept

Scoring of the MSA

Sum of pairs score **(SP score)**

= the sum of the scores of all pair wise alignments **within** the MSA

```
x:    KYFHKAGNQHSPT
y:    KYFHKAGNGH--T
z:    KEFH---NGH--T
```

$$\sigma(\tilde{x}, \tilde{y}, \tilde{z}) = \sigma(\tilde{x},\tilde{y}) + \sigma(\tilde{x},\tilde{z}) + \sigma(\tilde{y},\tilde{z})$$

Note:

$\leq \sigma_{opt}(x,y)$ $\leq \sigma_{opt}(x,z)$ $\leq \sigma_{opt}(y,z)$

Important notes

1. When aligning two sequences, the rest of the sequences are ignored

→ If the initial alignments are wrong,
the error is propagated into the rest of the MSA

2. Almost all MSA methods produce global MSAs

Only allowed operations: substitutions, insertions, deletions
Doesn't work for e.g. proteins with domain rearrangements

3. Manual alignments are superior over computational alignments

but are time consuming and not feasible for large sequences
in practice: use multiple methods/programs for your MSA and compare result
always check your alignment if it makes sense

Commonly used MSA programs

Differ in methods/algorithms/statistics for finding and scoring of MSA
accuracy/sensitivity
computational speed

Consistency based methods:
Try to maximize the agreement of all PW alignments
Often higher accuracy, but slower
e.g. T-COFFEE, PROBCONS

Iterative refinement:
The growing MSA is constantly checked and re-evaluated to optimize SP score
Early errors do thus not propagate that much
e.g. CLUSTALW, MUSCLE, MAFFT

"Local" MSA methods
Relaxed requirement for global alignability by allowing non-alignable regions
But still require co-linearity
MSA is build segment by segment instead of letter by letter
e.g. DIALIGN

Most of these programs can be accessed via the EBI website:
http://www.ebi.ac.uk/Tools/sequence.html
+ all have their own website

Commonly used MSA programs

Program	Advantages	Cautions
CLUSTALW	Uses less memory than other programs	Less accurate or scalable than modern programs
DIALIGN	Attempts to distinguish between alignable and non-alignable regions	Less accurate than CLUSTALW on global benchmarks
MAFFT, MUSCLE	Faster and more accurate than CLUSTALW; good trade-off of accuracy and computational cost. Options to run even faster, with lower average accuracy, for high-throughput applications.	For very large data sets (say, more than 1000 sequences) select time- and memory-saving options
PROBCONS	Highest accuracy score on several benchmarks	Computation time and memory usage is a limiting factor for large alignment problems (>100 sequences)
ProDA	Does not assume global alignability; allows repeated, shuffled and absent domains.	High computational cost and less accurate than CLUSTALW on global benchmarks
T-COFFEE	High accuracy and the ability to incorporate heterogeneous types of information	Computation time and memory usage is a limiting factor for large alignment problems (>100 sequences)

References: Edgar & Batzoglou (2006): Multiple sequence alignment; Wallace, Gordon, Higgins (2005): Multiple sequence alignments

Benchmarking

How to evaluate which MSA method/program performs best?

Need a set of “TRUE” MSAs - e.g. **BALIBASE**:

- contains automatically and manually produced alignments
- if possible, alignments were checked by structural alignments
- alignments cover different difficulties,
  - e.g. global vs. local, long gaps, different degrees of sequence similarity

Test program against BALIBASE to evaluate its sensitivity (number of correctly aligned positions)
Does not allow testing for specificity (e.g. no penalty for wrongly aligned positions)

How accurate are the most commonly used programs?
67-91% correctly aligned positions, depending on the program and alignment problem
No single program today performs best in all problems

There is no benchmarking database for DNA or RNA alignments

When to use which MSA program

CLUSTALW: at it's time (1994) dramatic progress in terms of sensitivity and speed
for a long time the favorite method for biologists, but there are better ones:

Input data	Recommendations
2–100 sequences of typical protein length (maximum around 10,000 residues) that are approximately globally alignable	Use PROBCONS, T-COFFEE, and MAFFT or MUSCLE, compare the results using ALTAVIST. Regions of agreement are more likely to be correct. For sequences with low percent identity, PROBCONS is generally the most accurate, but incorporating structure information (where available) via 3DCoffee (a variant of T-COFFEE) can be extremely helpful.
100–500 sequences that are approximately globally alignable	Use MUSCLE or one of the MAFFT scripts with default options. Comparison using ALTAVIST is possible, but the results are hard to interpret with larger numbers of sequences unless they are highly similar.
>500 sequences that are approximately globally alignable	Use MUSCLE with a faster option (we recommend maxiters-2) or one of the faster MAFFT scripts
Large numbers of alignments, high-throughput pipeline.	Use MUSCLE with faster options (e.g. maxiters-1 or maxiters-2) or one of the faster MAFFT scripts
2–100 sequences with conserved core regions surrounded by variable regions that are not alignable	Use DIALIGN
2–100 sequences with one or more common domains that may be shuffled, repeated or absent.	Use ProDA
A small number of unusually long sequences (say, >20,000 residues)	Use CLUSTALW. Other programs may run out of memory, causing an abort (e.g. a segmentation fault).

References: Edgar & Batzoglou (2006): Multiple sequence alignment; Wallace, Gordon, Higgins (2005): Multiple sequence alignments