Bioinformatics Course for ITN Natural Killer Cells

Heiko A. Schmidt

Center for Integrative Bioinformatics Vienna (CIBIV)
Max F. Perutz Laboratories (MFPL)
Vienna, Austria
http://www.cibiv.at

Lecturers




This lectures will be held by the following CIBIV members:

- Heiko SCHMIDT
- Sebastian BURGSTALLER-MUEHLBACHER
- Florian PFLUG
- Martin FAHRENBERGER

The Center for Integrative Bioinformatics Vienna (CIBIV), headed by Arndt von Haeseler, is a joint working group of the University of Vienna and the Medical University of Vienna located at the Max Perutz Labs.

Tentative Schedule

- 9:00-13:00 - Morning class (mixed lectures and hands-on) - including coffee break
- 13:00-14:00 - Lunch
- 14:00-18:00 - Afternoon class (mixed lectures and hands-on) - including coffee break

Tentative Topics

- **Day 1:** Similarity of sequences, pairwise/multiple sequence alignment, sequence retrieval, homology searches, determination of modular architecture, Sequence logos
- **Day 2:** Gene ontologies, online resources, functional gene networks and pathways, molecular interactions, variations between individuals, variations between species
- **Day 3:** Unix basics, installing software, command line tools, file formats, basic workflow steps, genome viewing/genome browser, alternatives to the command line
- **Day 4:** Introduction to basic statistics, R and Bioconductor; NGS data in R; accessing databases from R
- **Day 5:** Typical NGS readout after basic data analysis: sequences, coverage data, count data; ENCODE data; GO analysis; Pathways, motif detection

What is Bioinformatics?

Bioinformatics – a definition

What is Bioinformatics?

In a broad sense Bioinformatics deals with the application of methods from computer science and computers to answer biological questions.

(There are a number of differing definitions, depending on where the researchers come from.)

Bioinformatics is an inter-disciplinary field

Bioinformatics
- biology
  - molecular biology
  - genetics
  - ...
- computer science
  - algorithms
  - data bases
  - ...
- mathematics
  - statistics
  - propability theory
  - biomathematics
- medical sciences/biomedicine
- biophysics
- ...

Pairwise Sequence Alignment

Alignment

What is an alignment?

- **Alignment** is the procedure of writing two (or more) protein or DNA sequences in a way that a maximum of identical or similar characters are placed in the same column by adding gap characters ('-').

unaligned sequences:

```
seq1: LGPSKQTGASKGSSRIWDN

seq2: LNTKSAGASKGAILMRLGDAS
```

aligned sequences:

```
seq1: LGPSKQTGASKGS--SRIWDN-
      |   |  |||||    |  |
seq2: LN-TKSAGASKGAILMRLGDAS
```

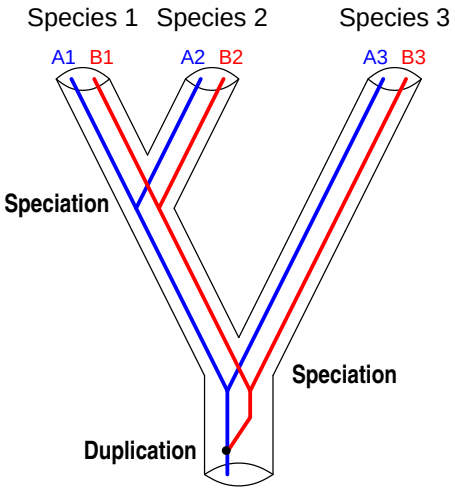

What are alignments used for?

Sequence alignment is one of the most fundamental procedures in Bioinformatics. It is used for

- Sequence comparison
- Sequencing: to combine sequenced fragments
- Search for genes
- Estimation of evolutionary distance
- Finding genes
- Finding relatives in databases
- Estimating function of genes and proteins
- Estimating structure of RNAs and proteins
- the basis to reconstruct evolutionary relationships and trees

Homology - related sequences

It usually only makes sense to construct alignments from **homologous** sequences, i.e., sequences that share a common ancestor.



Genes originating from

1. . . . *speciation* are **orthologous** (e.g., A1 and A3).
2. . . . *duplication* are **paralogous** (e.g., A2 and B2).

All genes A1, A2, A3, B1, B2, B3 descend from a *common ancestor* and are, hence, **homologous**.

Homology–Identity mistake

- Some people have published statements like:
  Sequences A and B are 30% homologous.
- **this is wrong!**
- What they mean is:
  Sequences A and B are 30% identical.
  That means they differ in 70% of they positions.

Note: sequences either are homologous or the are not!!!
There is no third alternative.

Point mutations

For alignments we consider the following point mutations in comparison of two sequences:

- **substitutions** (change of a character, in the alignment: mismatch)
- **insertion** of character(s) in one sequence
- **deletion** of character(s) from one sequence
- identical characters in both sequences are called a **match**

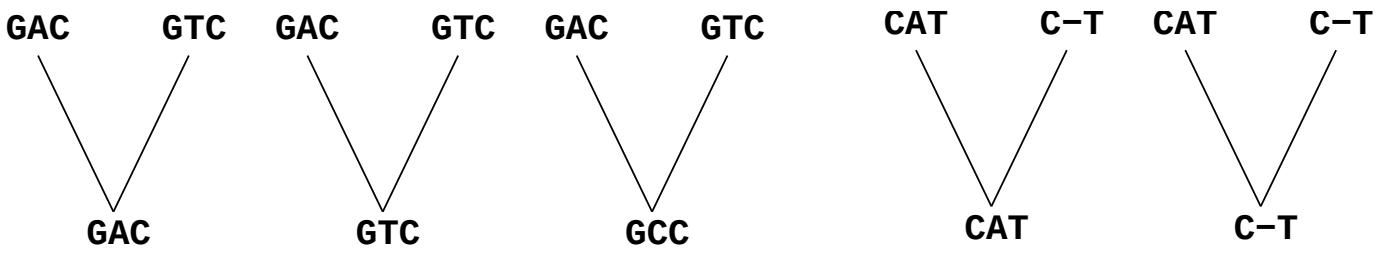
Direction of evolution

Since we don't know anything about the ancestral state and, hence, about the direction of the mutations

- we consider the substitution rate between two state **equal** in either direction.

Furthermore, we cannot distinguish insertions and deletions

- thus we will call them **indels**.



unknown ancestral states ***indels = insertions/deletions***

What is the optimal solution

Given two sequences A and B and a scoring function for two characters a and b

$$S(a,b) = \begin{cases} +5 & \text{if } a = b \text{ (match)} \\ -2 & \text{if } a \neq b \text{ (mismatch)} \\ -6 & \text{if } a \text{ or } b \text{ indel (gap)} \end{cases}$$

to score each alignment column.
Then we are looking for that alignment, that gives us the highest score $S(A,B)$ summing up the column scores $s(a,b)$ for all columns of the alignment.
For example:

T G A A C G T A
T A C A – – T A
+5–2–2+5–6–6+5+5 =4

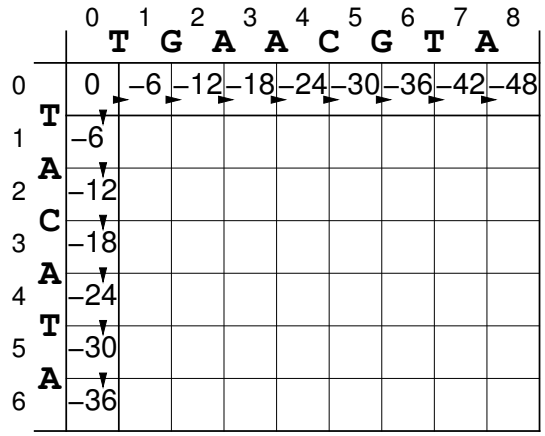
Why not just scoring all alignments?

- There are by far too many.
- There are about $\frac{2^{2N}}{\sqrt{2\pi N}}$ possible alignments,
- for two sequences of length $N = 300$ that is 10^{179} alignments.

Hence, we need a smart way to cut the computation short, like the **dynamic programming** approach for pairwise alignment by *Needleman and Wunsch* (1970).

Needleman-Wunsch algorithm

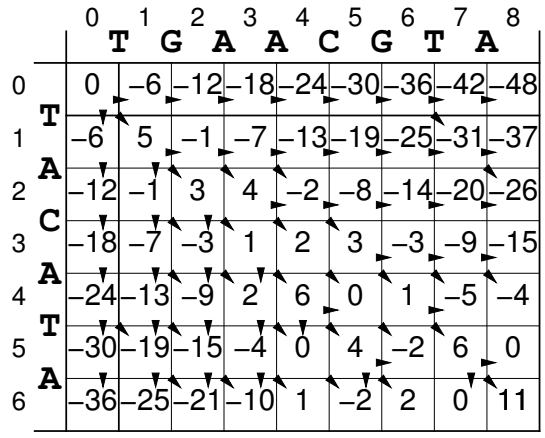
Given sequences A and B and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$



- Initialize an $N \times M$ matrix with the sequences A and B of length M and N.

Needleman-Wunsch algorithm

Given sequences A and B and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$



- Initialize an $N \times M$ matrix with the sequences A and B of length M and N.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) = \max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$
- $\sigma(i,j)$ always holds the optimal score for the alignment from the sequence start to (A_i, B_j).

Needleman-Wunsch algorithm

Given sequences A and B and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

		0 T	1 G	2 A	3 A	4 C	5 G	6 T	7 A	8
0		0	−6	−12	−18	−24	−30	−36	−42	−48
1	T	−6	5	−1	−7	−13	−19	−25	−31	−37
2	A	−12	−1	3	4	−2	−8	−14	−20	−26
3	C	−18	−7	−3	1	2	3	−3	−9	−15
4	A	−24	−13	−9	2	6	0	1	−5	−4
5	T	−30	−19	−15	−4	0	4	−2	6	0
6	A	−36	−25	−21	−10	1	−2	2	0	11

Resulting alignment and score:

T G A A C G T A
T − − A C A T A
+5−6−6+5+5−2+5+5 =11

- Initialize an $N \times M$ matrix with the sequences A and B of length M and N.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) = \max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$
- $\sigma(i,j)$ always holds the optimal score for the alignment from the sequence start to (A_i, B_j).
- The optimal score can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.

Needleman-Wunsch algorithm

Given sequences A and B and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

		0 T	1 G	2 A	3 A	4 C	5 G	6 T	7 A	8
0		0	−6	−12	−18	−24	−30	−36	−42	−48
1	T	−6	5	−1	−7	−13	−19	−25	−31	−37
2	A	−12	−1	3	4	−2	−8	−14	−20	−26
3	C	−18	−7	−3	1	2	3	−3	−9	−15
4	A	−24	−13	−9	2	6	0	1	−5	−4
5	T	−30	−19	−15	−4	0	4	−2	6	0
6	A	−36	−25	−21	−10	1	−2	2	0	11

Resulting alignment and score:

T G A A C G T A
T − − A C A T A
+5−6−6+5+5−2+5+5 =11
T G A A C G T A
T − A − C A T A
+5−6+5−6+5−2+5+5 =11

- Initialize an $N \times M$ matrix with the sequences A and B of length M and N.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) = \max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$
- $\sigma(i,j)$ always holds the optimal score for the alignment from the sequence start to (A_i, B_j).
- The optimal score can be found at $\sigma(N,M)$.
- The optimal alignment is retrieved by following the best values $\sigma(i,j)$.
- There can be more than one optimal alignment!

Local alignments

The Needleman-Wunsch algorithm always aligns the whole sequences producing a **global alignment**, while **end-free alignments** always involve a start and an end of any of the two sequences aligned.
However, if

- only a core of the sequence is conserved,
- the one sequence contains several conserved regions (e.g. genes) separated by regions with little conservation (intergenic regions)

local alignments (aligning subsequences with highest score) are much more reasonable.
This typically works also for sequences which only share a certain overlap (e.g. at the ends).

Smith-Waterman algorithm

Temple Smith and Mike Waterman (1981) found a way to adapt the Needleman-Wunsch algorithm to produce local alignments:

- the scoring function must contain negative values for mismatches,
- whenever $\sigma(i,j) < 0$ it is set to zero (here a possible backtrace will stop)
- the initial row and column with gap costs are set to zero
- the backtrace will start at the maximal $\sigma(i,j)$

Temple F. Smith and Michael S. Waterman (source: Wikipedia)

Smith-Waterman algorithm

Given sequences A and B and scoring function $s(a,b) = \begin{cases} +5 & a = b \\ -2 & a \neq b \\ -6 & a \text{ or } b \text{ indel} \end{cases}$

		0	1	2	3	4	5	6	7	8
			T	G	C	T	C	G	T	G
0		0	0	0	0	0	0	0	0	0
1	T	0	5	0	0	5	0	0	5	0
2	T	0	5	3	0	5	3	0	5	3
3	C	0	0	3	8	2	10	4	0	3
4	A	0	0	0	2	6	4	8	2	0
5	T	0	5	0	0	7	4	2	13	7
6	A	0	0	3	0	1	5	2	7	11

Resulting alignment and score:

T C G T
T C A T
+5+5−2+5 =13

- Initialize an $N \times M$ matrix with the sequences A and B of length M and N.
- Starting at the upper left corner set the intermediate scoring value $\sigma(i,j) =$
$$\max \begin{cases} \sigma(i-1,j-1) + s(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \\ 0 \end{cases}$$
- The optimal local alignment score is the maximal score among all $\sigma(i,j)$.
- The optimal local alignment is retrieved by backtracking until the $\sigma(i,j)$ of the current cell (i,j) gets zero.

Scoring Matrices

Using a scoring function like

$$S(a,b) = \begin{cases} +5 & \text{, if } a = b \text{ (match)} \\ -2 & \text{, if } a \neq b \text{ (mismatch)} \\ -6 & \text{, if } a \text{ or } b \text{ indel (gap)} \end{cases}$$

is a nice starting point, but has no biological motivation.
Instead more sophisticated scoring matrices and gap costs have been developed.

Gap costs

As you might have recognized, we use linear gap costs, i.e., every gap position is penalized the same no matter how long ℓ the whole gap might have grown.
More biologically meaningful are **affine gap costs** of the kind:

$$g(\ell) = g_o + \ell \cdot g_e$$

That means, that opening a gap is especially penalized with, e.g., gap opening cost $g_o = 11$ and gap extension cost $g_e = 2$.
The total gap cost has then to be subtracted from the current score.

More biological criteria for aa match/mismatch scoring

- differences in the **chemical properties** of amino acids (acidic/basic, polar/nonpolar, hydrophilic/-phobic, positively or negatively charged/neutral, size, ...)
- differences in the underlying **codons** (Glu↔Asp: 1, Glu↔Phe: 3)

These points should also influence the observed substitution rates among amino acids and are captured in PAM and BLOSUM matrices:

- observed aa exchanges in more **closely related** proteins (PAM)
- observed aa exchanges in **conserved blocks** of **distantly related** proteins (BLOSUM)

Incorporating more biological scoring schemes - PAM250

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	
C	12																				C
S	0	2																			S
T	−2	1	3																		T
P	−3	1	0	6																	P
A	−2	1	1	1	2																A
G	−3	1	0	−1	1	5															G
N	−4	1	0	−1	0	0	2														N
D	−5	0	**0**	−1	0	1	2	4													D
E	−5	**0**	**0**	−1	0	0	1	3	4												E
Q	−5	−1	−1	**0**	0	−1	1	2	2	4											Q
H	−3	−1	−1	0	−1	−2	2	1	1	3	6										H
R	−4	0	−1	0	−2	−3	0	−1	−1	1	2	6									R
K	−5	0	0	−1	−1	−2	1	0	0	1	0	3	5								K
M	−5	−2	−1	−2	−1	−3	−2	−3	−2	−1	−2	0	0	6							M
I	−2	−1	0	−2	−1	−3	−2	−2	−2	−2	−2	−2	−2	2	5						I
L	−6	−3	−2	−3	−2	−4	−3	−4	−3	−2	−2	−3	−3	4	2	6					L
V	−2	−1	0	−1	0	−1	−2	−2	−2	−2	−2	−2	−2	2	4	2	4				V
F	−4	−3	−3	−5	−4	−5	−4	−6	−5	−5	−2	−4	−5	0	1	2	−1	9			F
Y	0	−3	−3	−5	−3	−5	−2	−4	−4	−4	0	−4	−4	−2	−1	−1	−2	7	10		Y
W	−8	−2	−5	−6	−6	−7	−4	−7	−7	−5	−3	2	−3	−4	−5	−2	−6	0	0	17	W
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	

$$\sigma(i,j) = \max \begin{cases} \sigma(i-1,j-1) + pam250(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$$

Using more biological scoring schemes - BLOSUM62

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

$$\sigma(i,j) = \max \begin{cases} \sigma(i-1,j-1) + blosum62(A_i, B_j) & \text{match/mismatch} \\ \sigma(i-1,j) + s(A_i, -) & \text{gap in } B \\ \sigma(i,j-1) + s(B_i, -) & \text{gap in } A \end{cases}$$

Database Searching

Sequence Databases

How to determine structure, function, homologs ... for a yet unknown query sequence Q?

- There are large amounts of sequences and their annotations in the public databases.
- There are very diverse databases:
  - DNA sequences (INSDC - Intl. Nucl. Seq. Database Collaboration: ENA at EMBL, GenBank at NCBI, DDBJ at NIG),
  - protein sequences (SwissProt/UniProt),
  - genomic data (Ensembl), structures (PDB), etc.
- Sequences showing (high) similarity to Q, are likely to share the same structure, function, and history.
- Solution: Searching in the public databases for similar sequences can help to infer the whereabouts of new (unknown) sequences.
- Since it is unlikely to find completely identical sequences, approximate searches are necessary, to find similar sequences.
- Such search methods are usually based on pairwise sequence alignment comparing the query sequence Q to a large database T.

Optimal Alignment for Searching

Given a database D (length m) and a query sequence Q (length n).

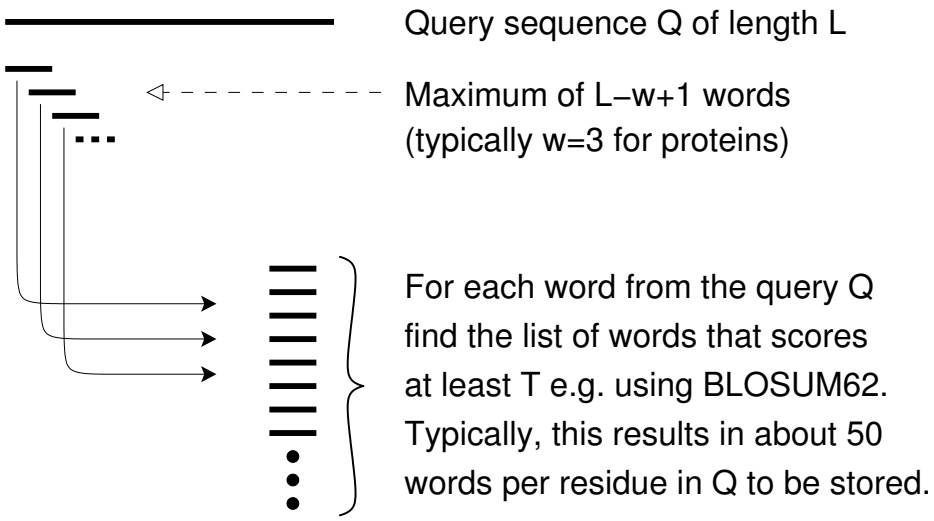
- The Needleman-Wunsch algorithm produces global alignments and is thus not suitable for database searching
- The Smith-Waterman algorithm guarantees to find the result with optimal score, but is very slow in large databases due to its $\Theta(n^2)$ runtime complexity ($m \times n$ cells).
- Thus, often faster, but heuristic approaches are used.

BLAST procedure: rough overview

(0) build an index of the databaset look-up
(1) Compile list of high-scoring strings (words)
(2) search for exact hits using index look-ups $\rightarrow$ seeds
(3) extend seeds $\rightarrow$ local alignments
(4) assess significance of the score

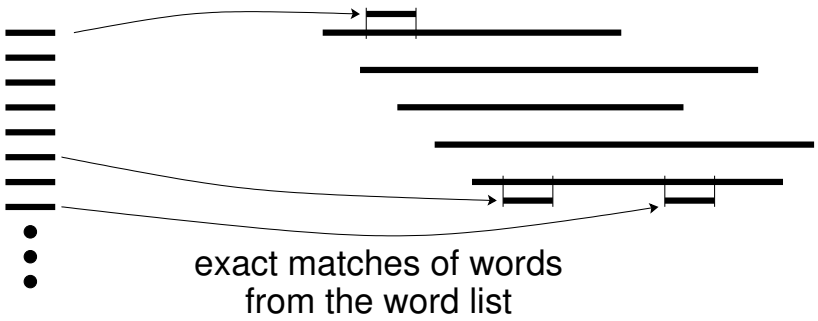
BLAST procedure: w-mer neighborhood of the query

(1) For the query Q construct the list of high-scoring words of length w (w-mer neighborhood).



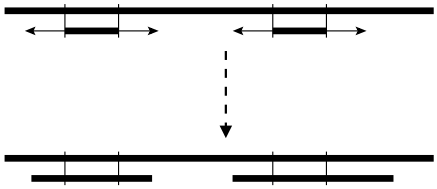
BLAST procedure: find w-mer hits in database

(2) Compare the word list to the database and identify exact matches.



BLAST procedure: extend *w*-mer hits

(3) For each word match, extend alignment (of hit and query) in both directions to find the maximal segment pairs (MSPs) that score greater than score threshold C



maximal segment pairs (MSPs)

Definitions: Given sequences S_1 and S_2

- Segment Pair = a pair of substrings from S_1 and S_2 of equal length aligned without gaps
- Maximal scoring Segment Pair (MSP) = segment pair which could not be shortened or extended without reducing the corresponding score
- High-scoring Segment Pair (HSP) = is an MSP scoring above a certain cutoff C.

BLAST procedure: extend *w*-mer hits (2)

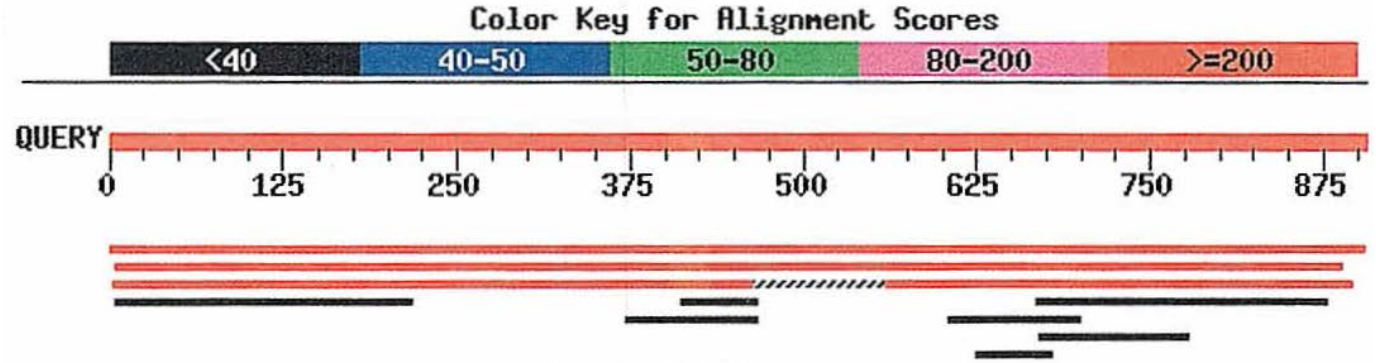
- Extending the word hit alignment in an optimal way to test whether it is part of a maximal scoring segment pair with score $> C \rightarrow$ HSP
- choice of C is guided by (a) scoring matrix and statistics of (b) Q and (c) DB, such that a score of C is unlikely to occur by chance in any of the DB sequences.
- Stop the extension if the score falls below certain thresholds (derived from previous hits) to improve speed
- Thus, (due to *w*-length words and cutoffs) BLAST does not guarantee to find all segments with score at least C, there is a very small chance of missing important pairs (computable with DP).
- In practice, BLAST is orders of magnitude faster than dynamic programming and its biological sensitivity/specificity is good for distant relationships, whereas for analyzing and displaying alignments Smith-Waterman optimization should be used.

BLAST output

```
sp|Q92889|XPF_HUMAN DNA-REPAIR PROTEIN COMPLEMENTING XP-F CELL (XERODERMA PIGMENTOSUM GROUP F COMPLEMENTING PROTI
(DNA EXCISION REPAIR PROTEIN ERCC-4) Length = 905  Score = 1659 bits (4249), Expect = 0.0  Identities = 838/905 (
Positives = 838/905 (92%)

Query: 1   MAPLLEYERQLVLELLDTDGLVVCARGLGADRLLYHFLQLHCHPACLVLVLNTQPAEEEY 60
           MAPLLEYERQLVLELLDTDGLVVCARGLGADRLLYHFLQLHCHPACLVLVLNTQPAEEEY
Sbjct: 1   MAPLLEYERQLVLELLDTDGLVVCARGLGADRLLYHFLQLHCHPACLVLVLNTQPAEEEY 60
.
.
Query: 301 SLRATEKAFGQNSGWLFLDSSTSMFINARARVYHLPDAXXXXXXXXXXXXXXXXXXXXXX 360
           SLRATEKAFGQNSGWLFLDSSTSMFINARARVYHLPDA
Sbjct: 301 SLRATEKAFGQNSGWLFLDSSTSMFINARARVYHLPDAKMSKKEKISEKMEIKEGEETKK 360
.
.

sp|P36617|RA16_SCHPO DNA REPAIR PROTEIN RAD16 Length = 892 Score =  485 bits (1236), Expect = e-136  Identities =
303/918 (33%), Positives = 497/918 (54%), Gaps = 76/918 (8%)

Query: 5   LEYERQLVLELLDTDGLVVCARGLGADRLLYHFLQLHCHPACLVLVLNTQPAEEEYFINQ 64
           L Y++Q+  EL++ DGL V A GL    ++  + L      P  L+L++     + E   ++
Sbjct: 9   LAYQQQVFNELIEEDGLCVIAPGLSLLQIAANVLSYFAVPGSLLLLVGANVDDIELIQHE 68
.
.

Query: 304 -----ATEKAFGQNSGWLFLDSSTSMFINARARVYHLPDAXXXXXXXXXXXXXXXXXXXX 358
                ++  +  Q S WL LD++  M    AR RVY   +
Sbjct: 309 LSVNVSSYPSNAQPSPWLMLDAANKMIRVARDRVYKESEGPNMDAIP------------- 355


sp|P06777|RAD1_YEAST DNA REPAIR PROTEIN RAD1 Length = 1100  Score =  231 bits (583), Expect = 4e-60  Identities =
136/369 (36%), Positives = 208/369 (55%), Gaps = 37/369 (10%)

Query: 559  LHEVEPRYVVLYDAELTFVRQLEIYRASRPGKPLRVYFLIYGGSTEEQRYLTALRKEKEA 618
            L E+ P Y+++++ +++F+RQ+E+Y+A       +VYF+ YG S EEQ +LTA+++EK+A
Sbjct: 704  LQEMMPSYIIMFEPDISFIRQIEVYKAIVKDLQPKVYFMYYGESIEEQSHLTAIKREKDA 763
.
```


BLAST output

```
BLASTP 2.0.5 [May-5-1998]
Query= human XP-F repair gene          (905 letters)

Database: Non-redundant SwissProt sequences 74,596 sequences; 26,848,718 total letters
```




```
Distribution of 11 BLAST Hits on the Query Sequence

                                                                    Score     E
Sequences producing significant alignments:                         (bits)  Value

sp|Q92889|XPF_HUMAN   DNA-REPAIR PROTEIN COMPLEMENTING XP-F CELL ... 1659  0.0
sp|P36617|RA16_SCHPO  DNA REPAIR PROTEIN RAD16                        485  e-136
sp|P06777|RAD1_YEAST  DNA REPAIR PROTEIN RAD1                         231  4e-60
sp|P40562|YIS2_YEAST  PUTATIVE ATP-DEPENDENT RNA HELICASE YIR002C      37  0.17
sp|Q10202|YAXB_SCHPO  PUTATIVE ATP-DEPENDENT RNA HELICASE C13F4.11C    36  0.38
```

BLAST flavors

The BLAST package contain programs for different purposes:

BLAST program	**query type**	**database type**
`blastn`	nucleotide	nucleotide
`blastp`	protein	protein
`blastx`	nucleotide (translated in 6 frames)	protein
`tblastn`	protein	nucleotide (translated in 6 frames)
`tblastx`	nucleotide (both translated	nucleotide in 6 frames)

BLAT - BLAST-Like Alignment Tool (Kent, 2002)

- To reduce search time BLAT requires only one single, but longer exact match (instead on several consecutive ones), to be candidate for the more rigorous search.
- This speeds up the candidate search, but makes the search less sensitive...
- that means it might miss (more) relevant hits.
- Nevertheless, BLAT works well if the database sequences and the query are closely related.
- However, this is the scenario BLAT was developed for (mapping reads against a closely related reference).

Specialized BLAST extension: Phi-BLAST

Pattern-hit initiated BLAST searches for 'regular expressions' of motifs in a protein database.

- Often we have certain patterns that have to occur in a sequence but we cannot write them down as one sequence,
- then Patterns can help.
- If the Patterns we are searching for must contain a Tryptophane and then after 9-11 residues a Phenylalanine, Valine, or Tyrosine followed by an Alanine
- we can code it as: `W-x(9-11)-[FVY]-A`
- and feed it to Phi-BLAST.

Multiple Sequence Alignment

Multiple Sequence Alignment (MSA)

What is a multiple sequence alignment?

- MSA is like a pairwise sequence alignment but instead of two sequences three or more are aligned, trying to maximize the number of identical or similar characters in a column by adding gaps ('-').

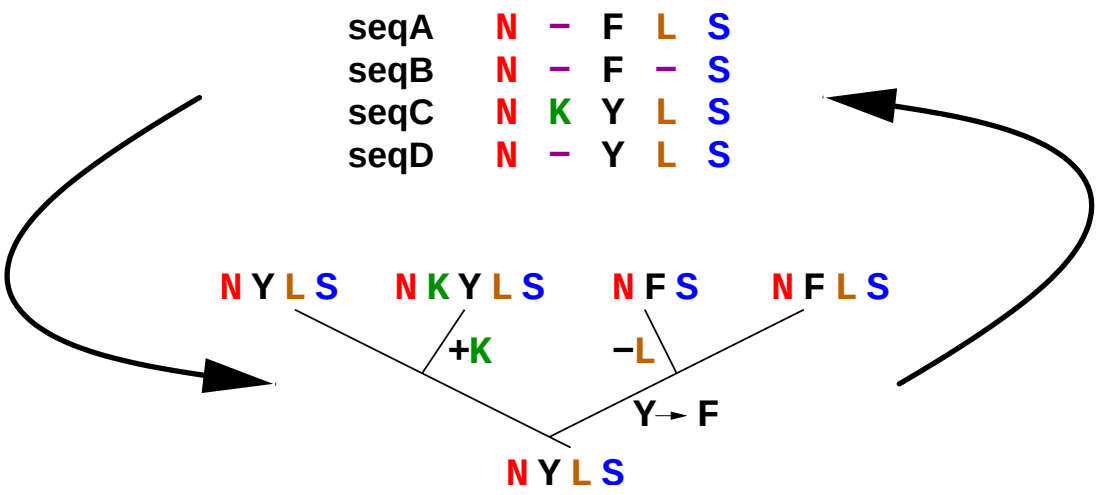
unaligned sequences:

```
seq1: LGPSKQTGASKGSSRIWDN
seq2: LNTKSAGASKGAILMRLGDAS
seq3: LGPTKSTGASKGAILSRLGDA
```

aligned sequences:

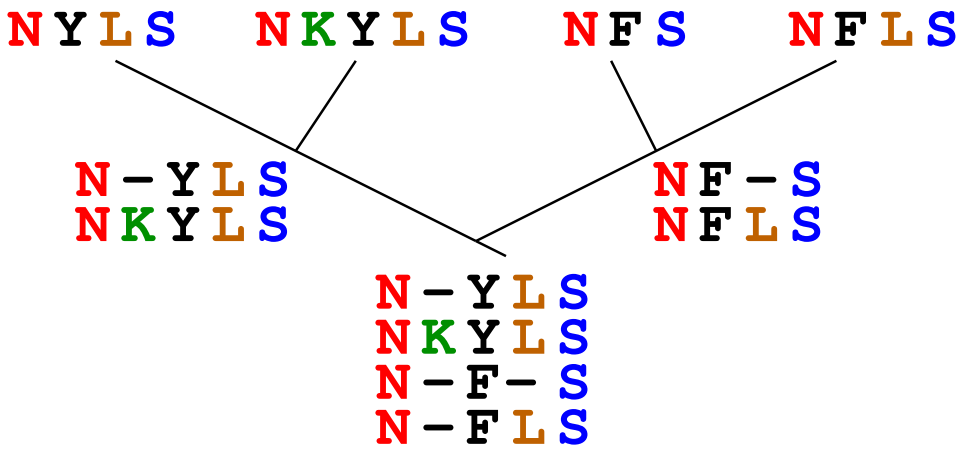
```
seq1: LGPSKQTGASKGS--SRIWDN-
seq2: LN-TKSAGASKGAILMRLWDAS
seq3: LGPTKSTGASKGAILSRLGDA-
```


MSA ↔ phylogeny relationship



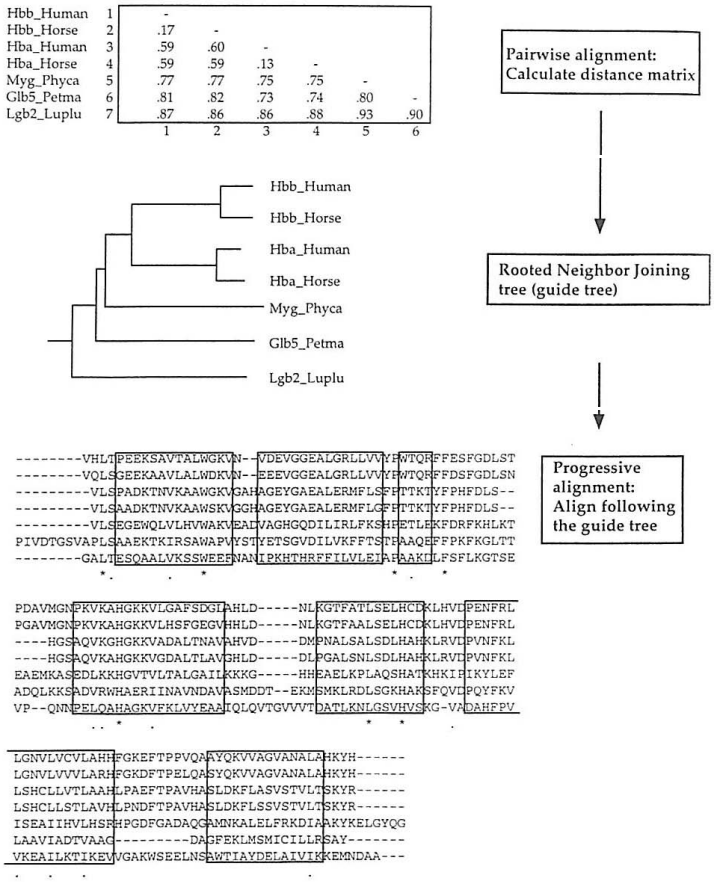
- Having the 'true tree' that reflects the history of our sequences facilitates significantly the alignment.
- Phylogenetic trees are usually based on alignments.
- **typical hen-and-egg problem!!!**

Progressive alignment



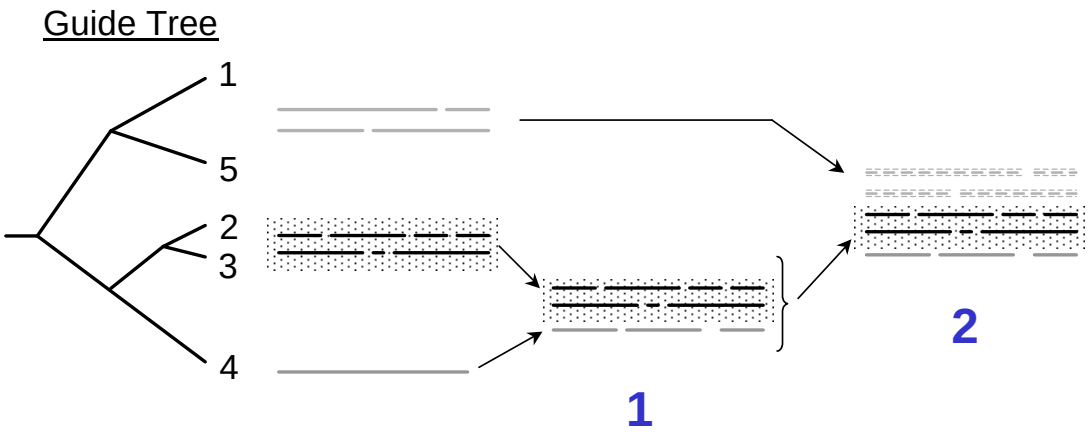
- Given a tree
- . . . the sequences are aligned progressing from the leaves to the root
- . . . aligning the subalignments where the branches meet

Progressive Alignment: ClustalW (Higgins *et al.*, 1994)



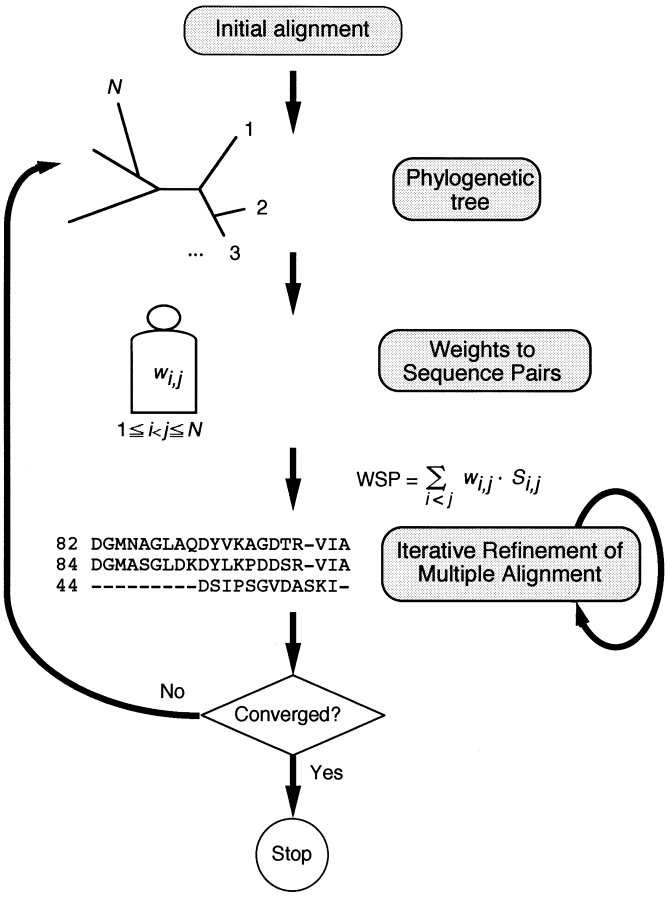
- ClustalW is one of the most frequently used MSA programs.
- however, not necessarily the best one.
- ClustalW Algorithm:
- construct all pairwise alignments and compute the pairwise distances
- construct Neighbor Joining tree as guide tree
- Along the guide tree (closely related sequences first) align sequences and sub-alignments.

Progressive Alignment: ClustalW, cont. (Higgins *et al.*, 1994)



- construct a tree by successively joining nearest neighbors
- the tree is **no** phylogeny, but a guide tree
- align the next 2 sequences (or profiles) with dynamic programming
- produce a new profile from the new alignment
- continue with next pair
- **Problem:** once a gap, always a gap, that means early mistakes are never corrected

Iterative alignment: PRRP (Gotoh, 1996)



Iterative scheme:

- Start with some initial alignment
- Construct a phylogenetic tree
- Use it to (re-)weight the sequence pair
- Iteratively refine the alignment to achieve higher score
- If the refinement brought benefit, start again.

MAFFT (Katoh et al. 2002+2005)

MAFFT is a fast and accurate approach for many sequences:

1. Progressive alignment FFT-NS-1:
   - compute rough distances and from those a guide tree
   - perform a progressive alignment using the guide tree
2. Progressive alignment FFT-NS-2:
   - use the FFT-NS-1 alignment to compute a better guide tree
   - again perform a progressive alignment
3. Iterative refinement FFT-NS-i:
   - start from alignment FFT-NS-2
   - iteratively improve the alignment
   - by optimizing the weighted sum-of-pairs score

- MAFFT uses a mathematical trick, the fast Fourier transform (FFT) to reduce the time complexity to find homologous regions between sequences.
- This approximation reduces the pairwise comparisons of two sequence of length n from $O(n^2)$ for dynamic programming to $O(n \log n)$ using FFT.

A comparison of MSA programs (Sievers et al., 2011)

Table 1 BAliBASE results

Aligner	Av score (218 families)	BB11 (38 families)	BB12 (44 families)	BB2 (41 families)	BB3 (30 families)	BB4 (49 families)	BB5 (16 families)	Tot time (s)	Consistency
MSAprobs	0.607	0.441	0.865	0.464	0.607	0.622	0.608	12 382.00	Yes
Probalign	0.589	0.453	0.862	0.439	0.566	0.603	0.549	10 095.20	Yes
MAFFT (auto)	0.588	0.439	0.831	0.450	0.581	0.605	0.591	1475.40	Mostly (203/218)
Probcons	0.558	0.417	0.855	0.406	0.544	0.532	0.573	13 086.30	Yes
Clustal Ω	0.554	0.358	0.789	0.450	0.575	0.579	0.533	539.91	No
T-Coffee	0.551	0.410	0.848	0.402	0.491	0.545	0.587	81 041.50	Yes
Kalign	0.501	0.365	0.790	0.360	0.476	0.504	0.435	21.88	No
MUSCLE	0.475	0.318	0.804	0.350	0.409	0.450	0.460	789.57	No
MAFFT (default)	0.458	0.258	0.749	0.316	0.425	0.480	0.496	68.24	No
FSA	0.419	0.270	0.818	0.187	0.259	0.474	0.398	53 648.10	No
Dialign	0.415	0.265	0.696	0.292	0.312	0.441	0.425	3977.44	No
PRANK	0.376	0.223	0.680	0.257	0.321	0.360	0.356	128 355.00	No
ClustalW	0.374	0.227	0.712	0.220	0.272	0.396	0.308	766.47	No

The figures are total column scores produced using bali score on core columns only. The average score over all families is given in the second column. The results for BAliBASE subgroupings are in columns 3–8. The total run time for all 218 families is given in the second last column. The last column indicates whether the method is consistency based.

- MAFFT (with auto-option) and Clustal Ω – good tradeoff between accuracy and runtime.
- Anyway, whatever method you use, never trust the alignment blindly. Always take a look at the alignments produced.
- Typically one can see easily, if something went totally wrong. Sequences that do not fit at all (e.g. non-homologous sequences or proteins translated from the wrong reading frame) should better be discarded before (re)aligning.

Reflecting the Multiple Sequence Alignments

(Sequence Consensus and Sequence Logos)

IUPAC/IUB Codes

IUPAC (Intl. Union of Pure and Applied Chemistry) and IUB (Intl. Union of Biochemistry) have published guidelines on how to code nucleotide and protein sequences (in 1 and 3-letter codes).

The DNA 1-letter codes are defined as (Comnish-Bowden/IUB, 1985):

- nucleotides:

A	Adenine
C	Cytosine
G	Guanine
T/U	Thymine (DNA), Uracil (RNA)

- triple-degenerate code:

B	not A	C, G, T/U
D	not C	A, G, T/U
H	not G	A, C, T/U
V	not T/U	A, C, G

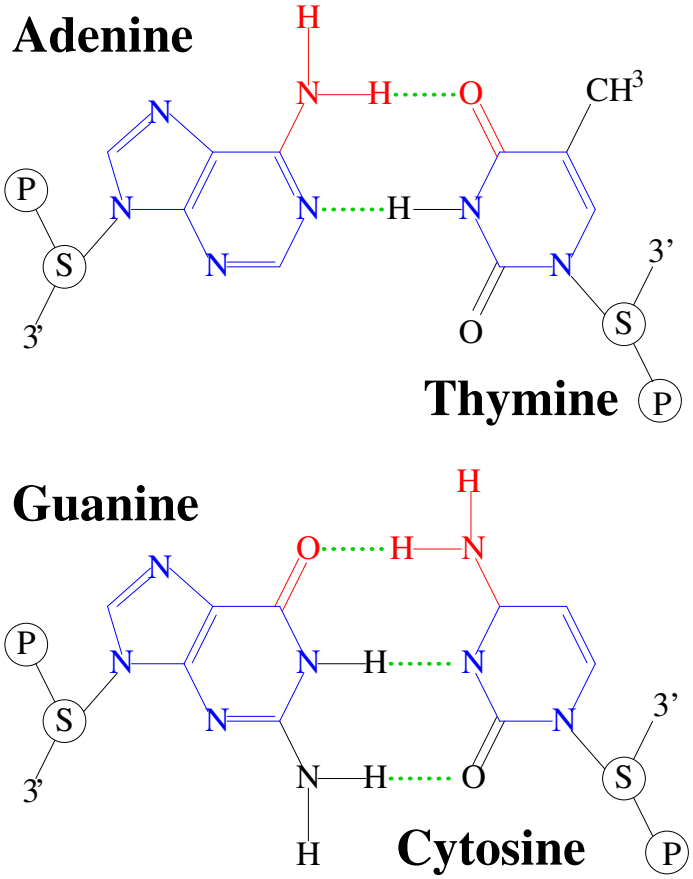
- double-degenerate codes:

W	Weak	A, T/U
S	Strong	C, G
R	puRine	A, G
Y	pYrimidine	C, T/U
M	aMino	A, C
K	Keto	G, T/U

- other:

N	aNy	A, C, G, T/U

IUPAC/IUB code: double-degenerate codes



W	Weak	A, T/U
S	Strong	C, G
R	puRine	A, G
Y	pYrimidine	C, T/U
M	aMino	A, C
K	Keto	G, T/U

One way to construct a consensus sequence

```
HEM13   CCCATTGTTCTC
HEM13   TTTCTGGTTCTC
HEM13   TCAATTGTTTAG
ANB1    CTCATTGTTGTC
ANB1    TCCATTGTTCTC
ANB1    CCTATTGTTCTC
ANB1    TCCATTGTTCGT
ROX1    CCAATTGTTTTG

        YCHATTGTTCTC
```

D'haeseleer (2006); Cavener (1987)

One way to construct a degenerate consensus sequence from a multiple DNA sequence alignment (using IUPAC/IUB code): For each column choose

- the nucleotide occurring in $> 50\%$ of sequences **and** $\geq$ twice as often as the 2nd most frequent
- a double degenerate symbol (R,Y,M,K,S,W) if according 2 bases occur in more than 75% of sequences
- a triple degenerate symbol (B,D,H,V) if one base does not occur at all
- an N, otherwise.

. . . but the consensus might be a highly unusual sequence.

Sequence Profiles

```
HEM13   CCCATTGTTCTC
HEM13   TTTCTGGTTCTC
HEM13   TCAATTGTTTAG
ANB1    CTCATTGTTGTC
ANB1    TCCATTGTTCTC
ANB1    CCTATTGTTCTC
ANB1    TCCATTGTTCGT
ROX1    CCAATTGTTTTG
```

nucleotide count matrix:

```
A 002700000010
C 464100000505
G 000001800112
T 422087088261
```

D'haeseleer (2006)

- Count matrices containing the number of occurrences of the nucleotides and
- Sequences profiles (or Position Frequency Matrix) also reflect the information in multiple sequence alignments.
- Profiles were used in progressive alignment strategies like ClustalW.

Sequence profile:

pos	1	2	3	4	5	6	7	8	9	10	11	12
A	.0	.0	.25	.875	.0	.0	.0	.0	.0	.0	.125	.0
C	.5	.75	.5	.125	.0	.0	.0	.0	.0	.625	.0	.625
G	.0	.0	.0	.0	.0	.125	1	.0	.0	.125	.125	.25
T	.5	.25	.25	.0	1	.875	.0	1	1	.25	.75	.125

Sequence Logos (frequency-based)

```
HEM13   CCCATTGTTCTC
HEM13   TTTCTGGTTCTC
HEM13   TCAATTGTTTAG
ANB1    CTCATTGTTGTC
ANB1    TCCATTGTTCTC
ANB1    CCTATTGTTCTC
ANB1    TCCATTGTTCGT
ROX1    CCAATTGTTTTG

        YCHATTGTTCTC

A 002700000010
C 464100000505
G 000001800112
T 422087088261
```

D'haeseleer (2006)

Sequence logos might be a better way of visualizing sequence motifs reflecting the number of occurrences of the nucleotides using a Position Frequency Matrix or Profile.

- a 'sequence logo' depicts the conservation pattern of a motif by stacks of letters which are scaled, e.g. by the frequency of occurrence:

Sequence Logos (based on information content)

```
HEM13   CCCATTGTTCTC
HEM13   TTTCTGGTTCTC
HEM13   TCAATTGTTTAG
ANB1    CTCATTGTTGTC
ANB1    TCCATTGTTCTC
ANB1    CCTATTGTTCTC
ANB1    TCCATTGTTCGT
ROX1    CCAATTGTTTTG

        YCHATTGTTCTC

    A 002700000010
    C 464100000505
    G 000001800112
    T 422087088261
```

D'haeseleer (2006)

- Originally sequence logos scale each stack i of letters by the column's information content of the bases b in bits:

$$I_i = 2 + \sum_b f_{b,i} \log_2 f_{b,i}$$

- In small samples like this, the information content are over-estimated and a correction has to be applied.

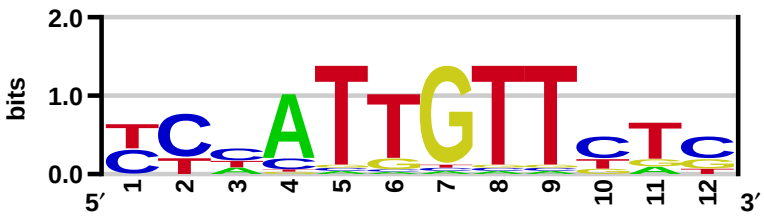
Sequence Logos (corrected for the small sample)

```
HEM13   CCCATTGTTCTC
HEM13   TTTCTGGTTCTC
HEM13   TCAATTGTTTAG
ANB1    CTCATTGTTGTC
ANB1    TCCATTGTTCTC
ANB1    CCTATTGTTCTC
ANB1    TCCATTGTTCGT
ROX1    CCAATTGTTTTG

        YCHATTGTTCTC

    A 002700000010
    C 464100000505
    G 000001800112
    T 422087088261
```

D'haeseleer (2006)

- The original sequence logos scale each stack i of letters by the column's information content of the bases b in bits:

$$I_i = 2 + \sum_b f_{b,i} \log_2 f_{b,i}$$



- This formula assumes that all nucleotides occur at equal frequency.

Sequence Logos (corrected for nucleotide content)

```
HEM13   CCCATTGTTCTC
HEM13   TTTCTGGTTCTC
HEM13   TCAATTGTTTAG
ANB1    CTCATTGTTGTC
ANB1    TCCATTGTTCTC
ANB1    CCTATTGTTCTC
ANB1    TCCATTGTTCGT
ROX1    CCAATTGTTTTG

        YCHATTGTTCTC

A 002700000010
C 464100000505
G 000001800112
T 422087088261
```

D'haeseleer (2006)

- To reflect possible base frequency biases, e.g. in GC content, it has been suggested to use relative entropy (=Kullback-Leibler distance):

$$I_{seq}(i) = -\sum_b f_{b,i} \log_2 \frac{f_{b,i}}{p_b}$$

where p_b is the overall frequency of b here with GC content in yeast (38%):

- %GC of *Caenorhabditis elegans* (36%), *Plasmodium falciparum* (19%), *Streptomyces coelicolor* (72%)

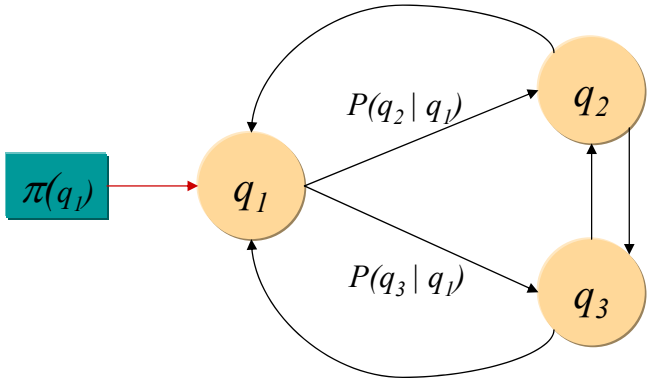
Hidden Markov Models

Markov model

The Markov chain is the tuple

$$M = (Q, P, \pi)$$

where:
Q is the set of states
P is the probability matrix of state transition
π is the vector of initial probabilities to start states

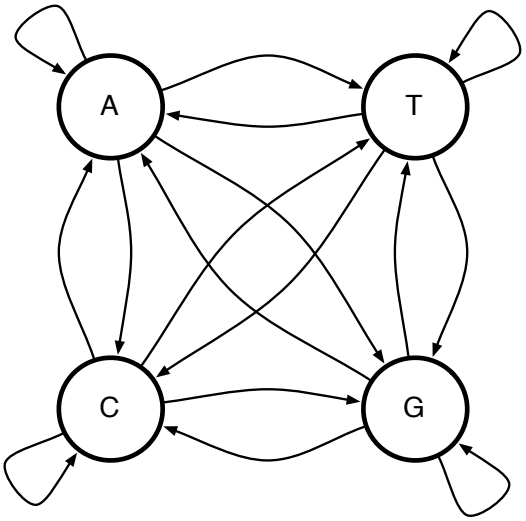


A Markov chain is traversed from state to state, producing a sequence of states.

Markov Chains: a simple Markov model for DNA

$$P(x = T|x = A) = a_{A,T}$$

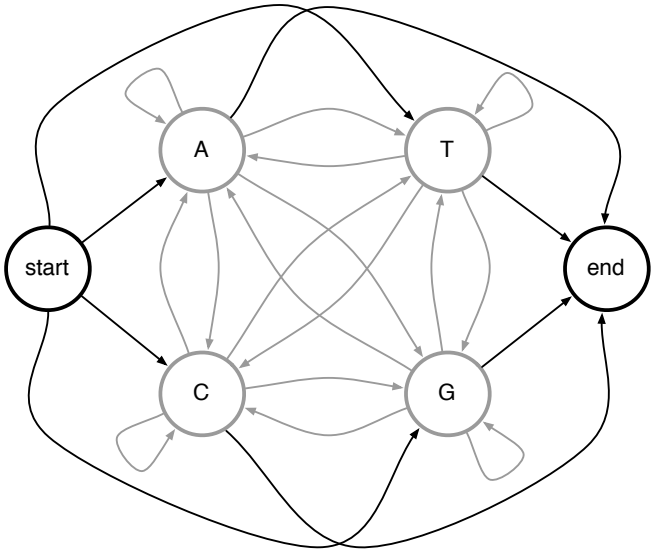
- 4 states A,C,G,T
- Transition between states occur with particular probabilities
- Each arrow has a probability parameter associated with it



$$a_{s,t} = P(x_i = t|x_{i-1} = s)$$

probability of making a transition from state s to state t

Include BEGIN and END state

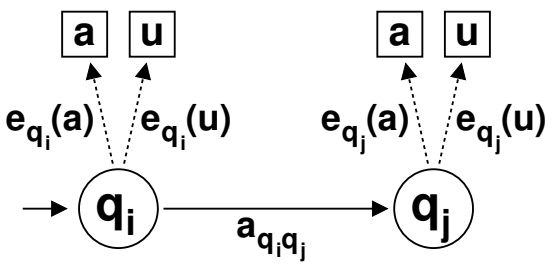


Two new states can be added to the Markov model.
These are treated as silent states, since they do not add to the sequence.

Hidden Markov Model (HMM)

Hidden Markov Models (HMMs) resemble Markov Models in having a finite number of **states** connected by **transitions**.

But the major difference between the two is that the states of the Hidden Markov Models are not associated with one symbol but with more than one symbol. Each state q can emit a symbol x with a probability given by the distribution of emission probabilities $e_q(x)$.



Hidden Markov Model (HMM)

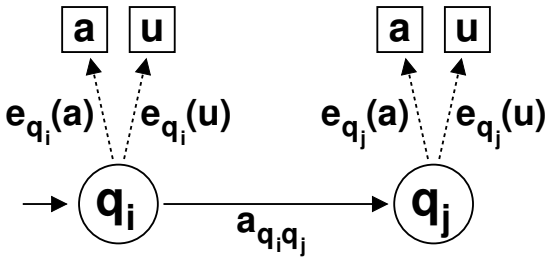
The HMM is a tuple

$$\mathcal{M} = (\Sigma, Q, A, E)$$

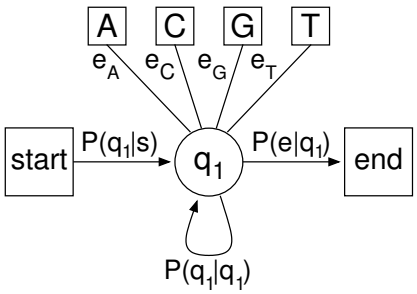
There are a finite number of states $\boldsymbol{Q}$ in the model

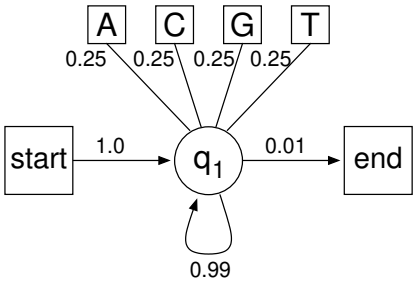
At a given time j, each new state q_j is entered from a previous state q_i, based upon a transition probability $a_{q_i,q_j} = P(q_j|q_i)$ from the probability distribution $\boldsymbol{A}$, which only depends on the previous state q_i (the Markovian property)

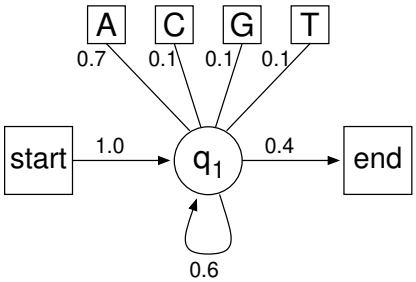
After each transition a symbol y_j from $\boldsymbol{\Sigma}$ is produced based on the current state q_j, with emission probability $e_{q_j}(y_j) = P(y_j|q_j)$



Example 1 (HMM)

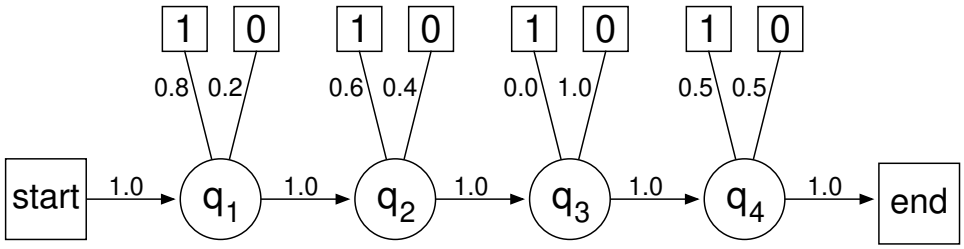






- AAAATGGTGAACCTGTCGTTCCG
- GAAA

Example 2 (HMM)



Can this HMM produce the following emitted sequences?

1 1 1 1 1	no
0 0 0 0	yes
1 0 0 1	yes
1 1 1 1	no
1 0	no

Position-specific Information in HMMs?

- Often we want to model classes of proteins or domains more specifically.
- What information do we have to model position-specific emission probabilities?
- **Profiles!** . . . extracted character frequencies from an alignment of relevant sequences.
- The resulting linear HMMs are called profile HMMs.

Profile HMM

The profile P of length n based on alphabet Y is the matrix

$$[e_i(y) : i = 1, ..., n \text{ and } y \in Y]$$

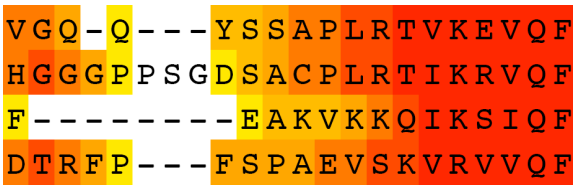
of probabilities.

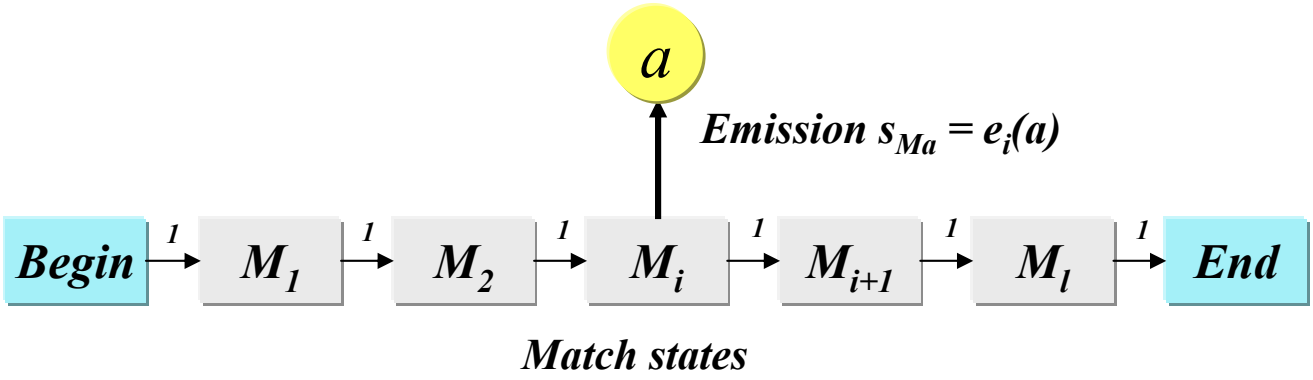
$e_i(y)$ **is the probability that** y **occurs on position** i **in the sequence.**

	1	2	3	4	5	6	...
S	$e_1(S)$	$e_2(S)$	$e_3(S)$	$e_4(S)$	$e_5(S)$	$e_6(S)$	...
A	$e_1(A)$	$e_2(A)$	$e_3(A)$	$e_4(A)$	$e_5(A)$	...	
K	$e_1(K)$	$e_2(K)$	$e_3(K)$	$e_4(K)$	...		
F	$e_1(F)$	$e_2(F)$	$e_3(F)$	...			
V	$e_1(V)$	$e_2(V)$	...				
...	$e_1(.)$	...					

The approach is to build a HMM with a repetitive structure of states but different emission probabilities in each position.

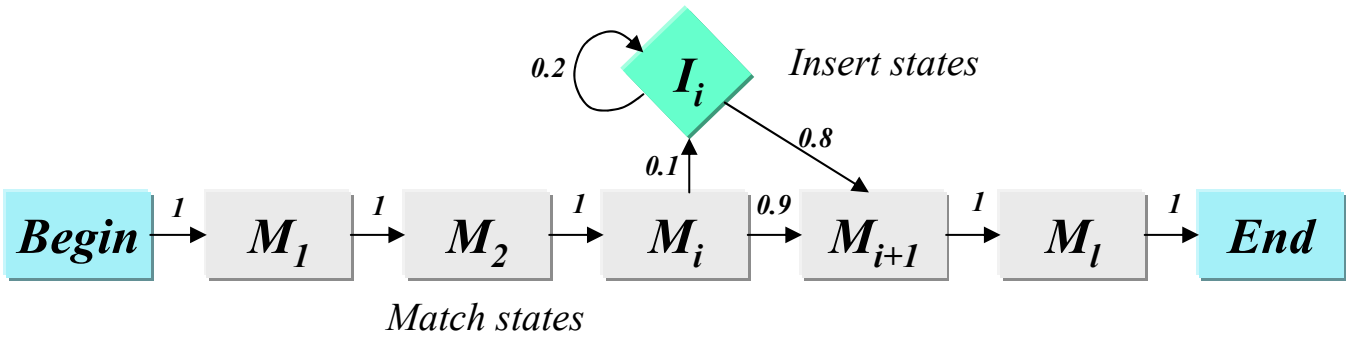
Profile HMM

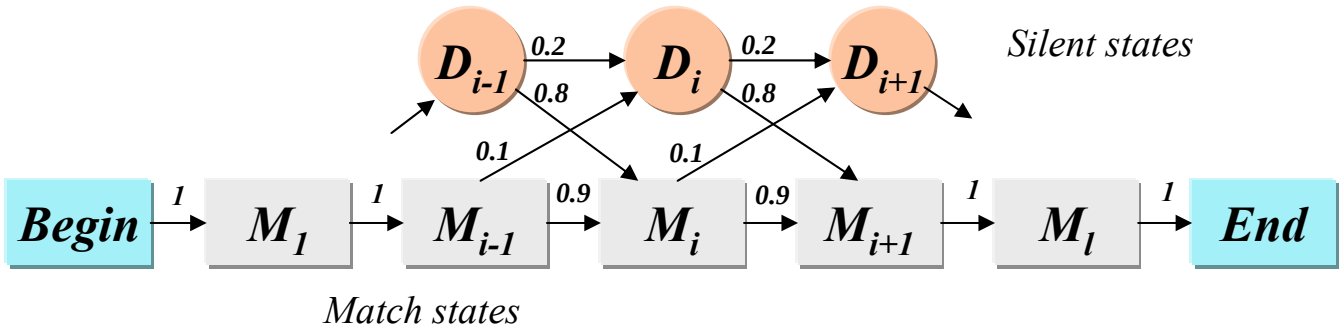




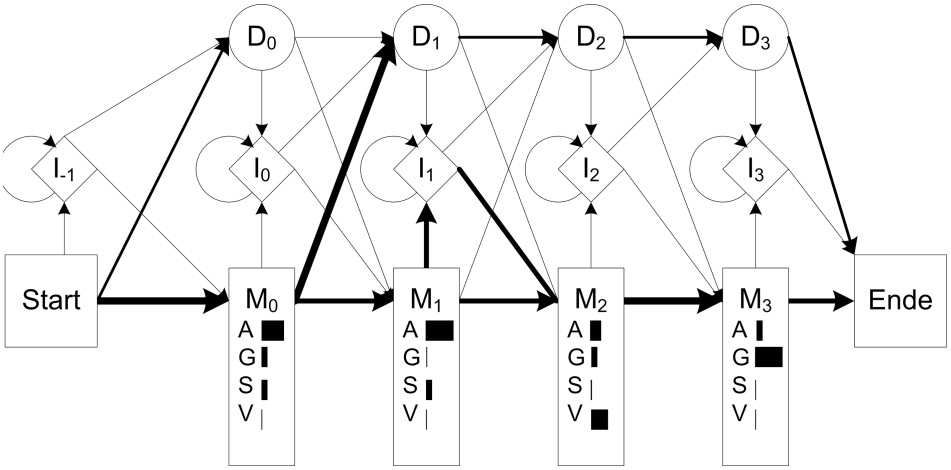
What about gaps? - Insertion/Deletion

Profile HMM



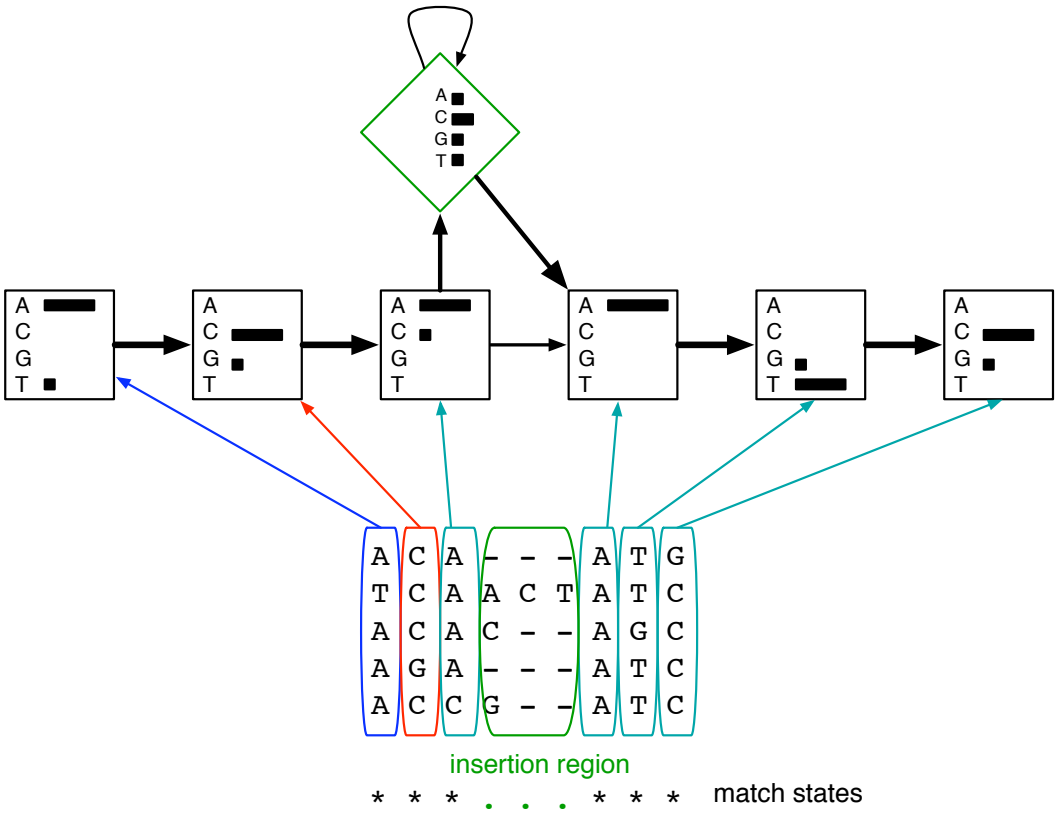


Profile HMM

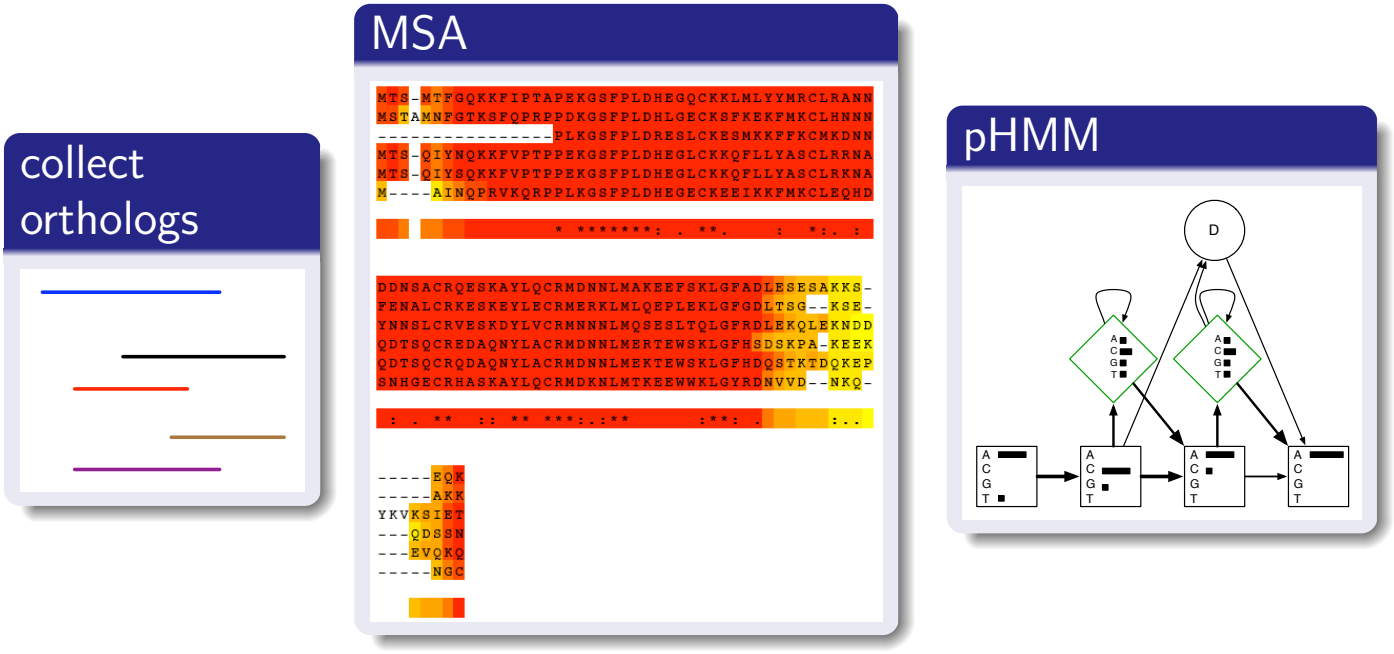


- How would you choose the number of states in the model?
- What states should be chosen?
- How are the model parameter chosen?

pHMM from aligned sequences

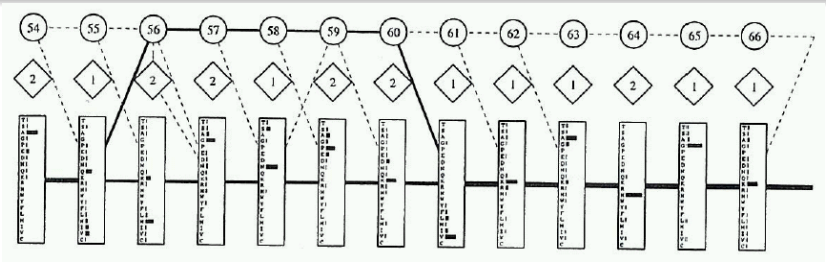


pHMM for related proteins



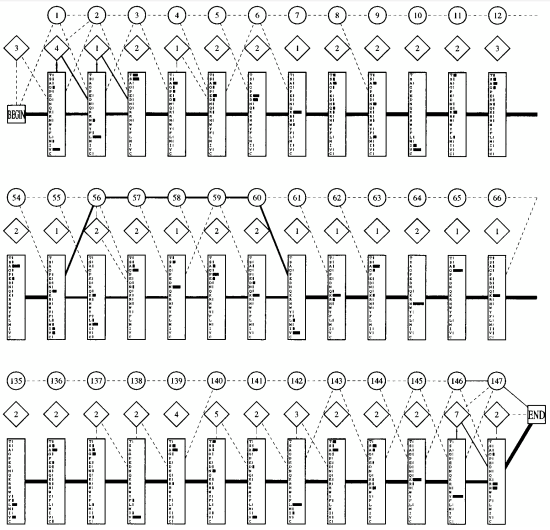
Is a new protein P also part of the group?

(p)HMMs - training



- Usually we do not know the best structure/probabilities for an HMM.
- We cannot evaluate all different possible paths through the HMM separately, neither for training nor for evaluation.
- Efficient algorithms exist to train pHMMs with sequence alignments (Baum-Welch algorithm).
- The training process changes the transition probabilities and, thus, leave a trace of the sequence family.
- Also the structure of the pHMM can be changed during training (States not used by at least half of the training set are merged with the insertion state; insertions present in more than half the training set are made a new match state.)
- Unfortunately, large training sets ($> 20 - 50$) are necessary to train HMMs

(p)HMMs - applications



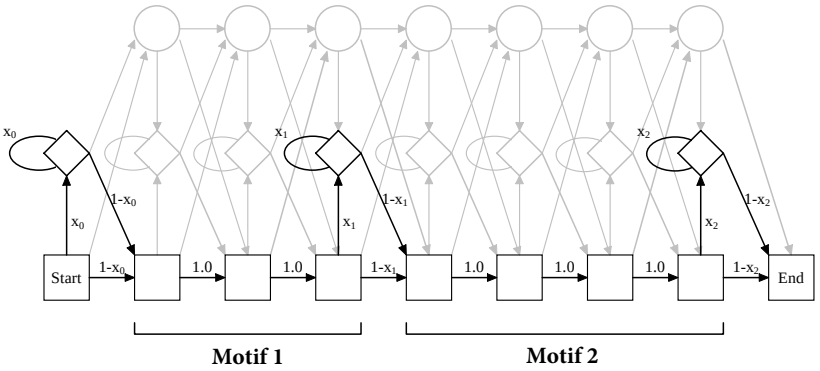
An HMM for globin sequences (Krogh et al. 1994)

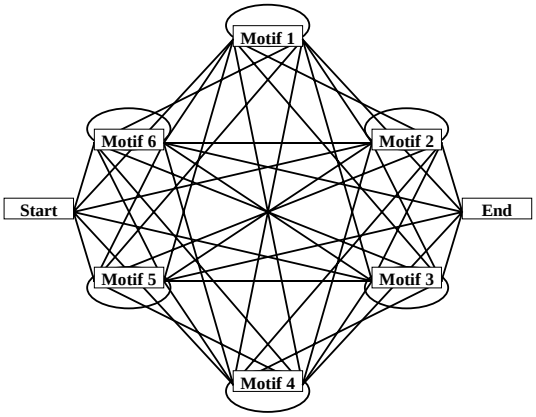
The main applications of pHMMs in Bioinformatics are certainly

- to search in databases for relatives of protein families with pHMMs generated from alignments of sequences from the respective protein-family
- to detect and annotate functional domains with pHMMs generated from alignments of their sequence motifs

(p)HMMs - application for multi-domain proteins

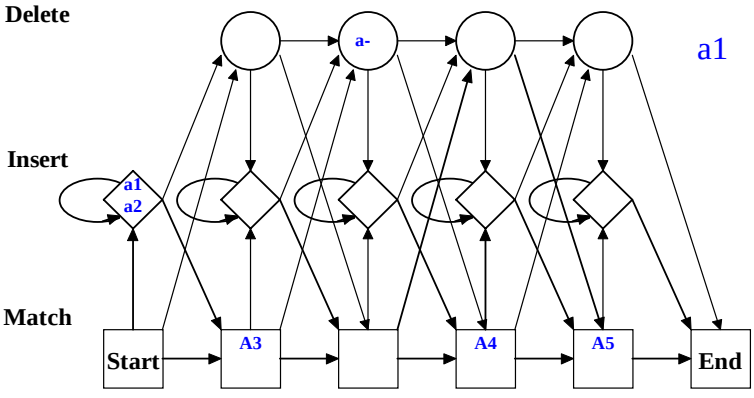
- One can search for proteins containing several domains
- by joining pHHMs to one linear HMM if the domains occur in a certain order
- but on can also join several domain allowing for unspecific orders





(p)HMMs - application sequence alignment

- Furthermore one can align sequences using HMMs
- by aligning the match states of the Viterbi path.
- E.g. aligning sequences $A_1A_2A_3A_4A_5$ and $B_1B_2B_3B_4B_5$.



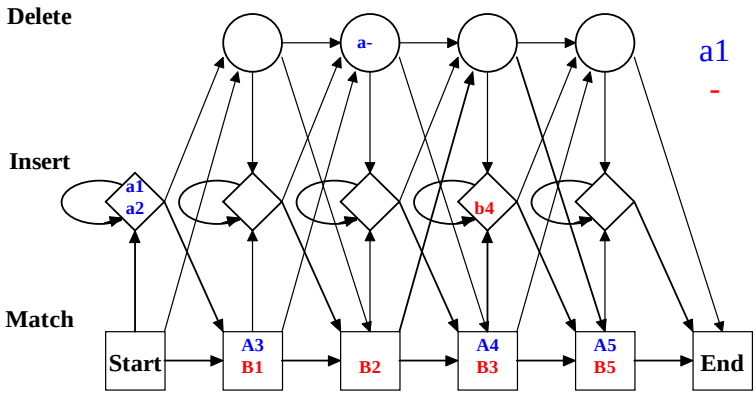
a1 a2 A3 - A4 A5

Please note,

- characters in insert/delete states have been marked by lower case letters in this example for distinction.
- a- in the deletion state is not really a character, but it is needed to avoid match state M2.
- characters mapped to the same insert states would be put in separate columns in the alignment.

(p)HMMs - application sequence alignment

- Furthermore one can align sequences using HMMs
- by aligning the match states of the Viterbi path.
- E.g. aligning sequences $A_1A_2A_3A_4A_5$ and $B_1B_2B_3B_4B_5$.



a1 a2 A3 - A4 - A5
- - B1 B2 B3 b4 B5

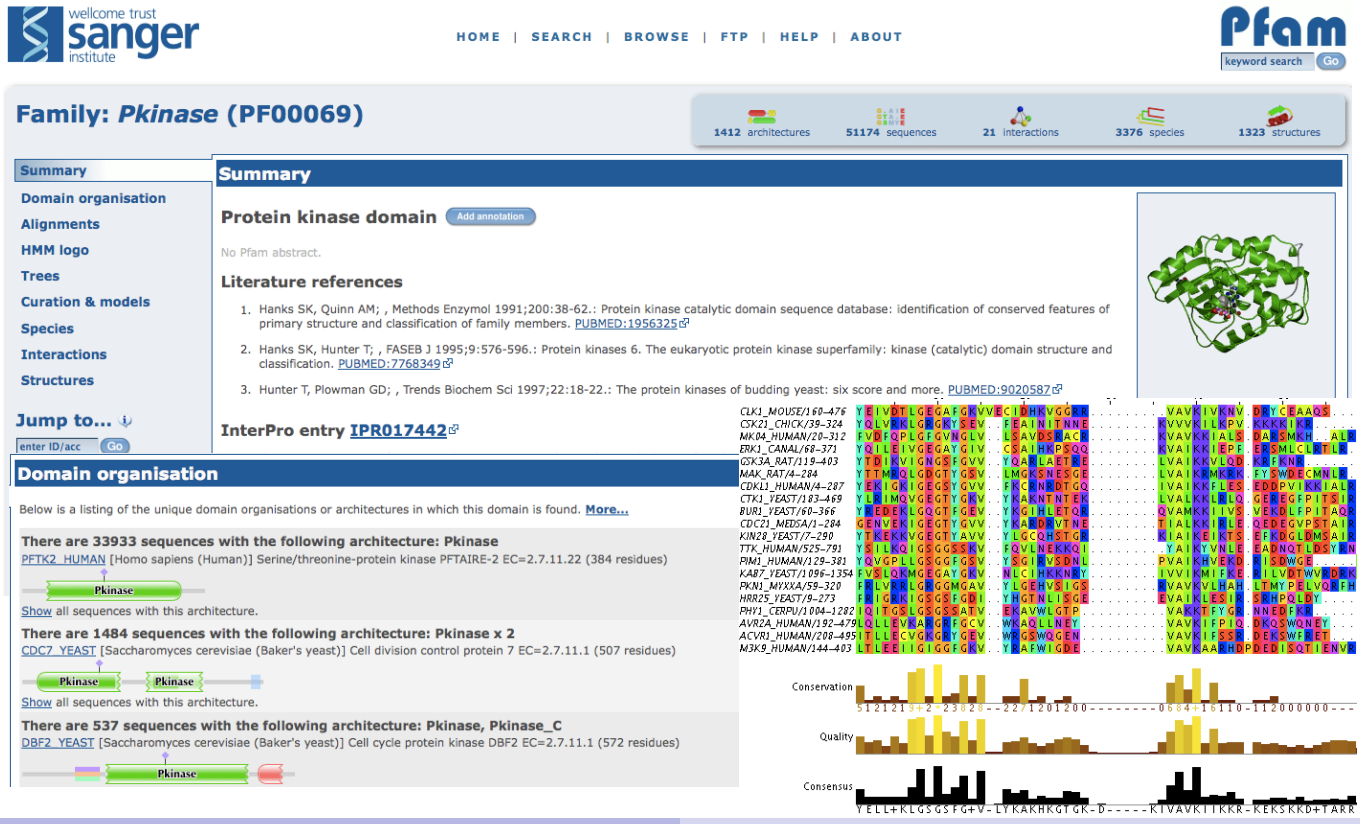
Please note,

- characters in insert/delete states have been marked by lower case letters in this example for distinction.
- a- in the deletion state is not really a character, but it is needed to avoid match state M2.
- characters mapped to the same insert states would be put in separate columns in the alignment.

Pfam

Protein families database of alignments and pHMMs.

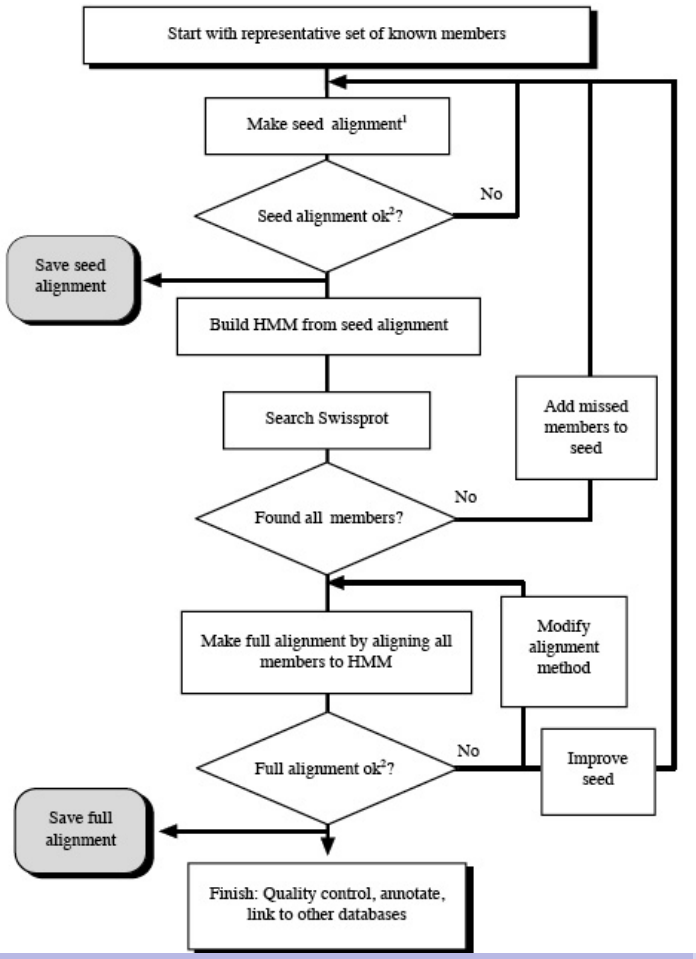
`http://pfam.sanger.ac.uk`



Pfam Database (Sonnhammer et al. 1997)

- The Pfam database contains Protein (Domain) Families based on the data in the Uniprot protein databases.
- There are two sections: Pfam (or Pfam-A) and Pfam-B
- **Pfam/Pfam-A**:
  - contains a set of hand curated seed alignments containing data from different sources (Uniprot, Prosite, Prodom, structural alignments, BLAST results, Repeats found with Dotter, published alignments)
  - from the seed alignments (profile) HMMs are created
  - the HMMs are used to collect additional data from Uniprot
  - create a full alignment (and HMM)
- **Pfam-B**: (abandoned 2013)
  - utilizes an automated clustering of all sequences from Uniprot in the ADDA database (without the sequences already used in Pfam-A).

Pfam-A Generation (Sonnhammer et al. 1997)



If the results during the curated generation of alignments/HMMs, one can optimize in different ways:

- seed alignment construction
- HMM construction
- generation of full alignment

Pfam Database Entries

An Entry in the Pfam database consists of:

- Annotation/summary about the protein (domain) family,
- the full alignment,
- the seed alignment (Pfam-A only),
- the (profile) HMM (Pfam-A only),
- background information about curation and HMM creation etc. (Pfam-A only)

Pfam currently (Rel. 31.0, 03/2017) contains

- **16712 Pfam-A** families based on *UniProtKB reference proteomes* (since Rel 29.0)
- last with both **Pfam-A (14831)** and **Pfam-B (544866)** was 27.0 (03/2013) based on SwissProt+SP-TrEMBL
- compared to **100 Pfam-A** and **11763 Pfam-B** families in Release 0.2, 01/1996 (based on SwissProt only).

Pfam Database Searching (Sonnhammer et al. 1997)

There are several ways to search in/with the Pfam HMM database:

- search with a query sequence against all HMMs in Pfam – e.g., to classify proteins or their domains
- one can download an HMM and search in a set of sequences to find (distant) homologs
- search with the whole set of HMMs against a set of (unknown) sequences, e.g., to annotate and/or find functional domains.