Genomanalyse

Vorlesung Genomanalyse vom 18.10.2011
Übersicht

1 Alignments

   Pairwise Alignments
       Dot plot
       Pairwise alignment
       Scoring matrices
       Motivation of log-odds
       Affine gap scores

   Multiple Alignments
       Motivation and definitions
       Local multiple alignments
       Scoring multiple alignments
       Brute force multiple alignment
       Progressive alignment
References

- Skript von Clemens Gröpl (letztes Semester)
Evolution of sequences

The genomic sequences that we observe today are the result of a random evolutionary process in which sequences are

- copied with occasional errors,

- stretches of sequence (e.g. gene, chromosome, genome) are duplicated and subsequently modified

- and evolutionary pressure introduces a bias of changes towards a functioning organism.
Evolution of sequences

The genomic sequences that we observe today are the result of a random evolutionary process in which sequences are

- copied with occasional errors,
- stretches of sequence (e.g. gene, chromosome, genome) are duplicated and subsequently modified
- and evolutionary pressure introduces a bias of changes towards a functioning organism.
Evolution of sequences

The genomic sequences that we observe today are the result of a random evolutionary process in which sequences are

- copied with occasional errors,
- stretches of sequence (e.g. gene, chromosome, genome) are duplicated and subsequently modified
- and evolutionary pressure introduces a bias of changes towards a functioning organism.
A dot means that the species has at that position the same nucleotide as human.
Examples of similar sequences

Example (DNA segment of BRCA1 gene from vertebrates)

A dot means that the species has at that position the same nucleotide as human.

Example (A protein family with proteins from human and the plant *Arabidopsis thaliana* (ARATH))

proteins are from the SUMO family: Small Ubiquitin-like Modifier
Standard assumption in genome analysis

Conservation of function

Function and 3D structure of proteins with the same origin are often still conserved when sequences are only very weakly similar.

Assumption

A statistical highly significant similarity of sequences (DNA, RNA, protein) usually implies the same evolutionary origin and that there is also functional or structural similarity.

“Twilight zone” for finding sequence similarity

(chalk board)
Standard assumption in genome analysis

Conservation of function

Function and 3D structure of proteins with the same origin are often still conserved when sequences are only very weakly similar.

Assumption

A statistical highly significant similarity of sequences (DNA, RNA, protein) usually implies the same evolutionary origin and that there is also functional or structural similarity.

“Twilight zone” for finding sequence similarity

(chalk board)
Standard assumption in genome analysis

Conservation of function

Function and 3D structure of proteins with the same origin are often still conserved when sequences are only very weakly similar.

Assumption

A statistical highly significant similarity of sequences (DNA, RNA, protein) usually implies the same evolutionary origin and that there is also functional or structural similarity.

“Twilight zone” for finding sequence similarity

(chalk board)
Sequence homology

**Definition (homology)**

Two sequences (or genes) are **homologous** if they descend from a common ancestor sequence.

**Definition (orthology)**

Two homologous sequences (or genes) are **orthologous** if they were separated by a speciation event.

**Definition (paralogy)**

Two homologous sequences (or genes) are **paralogous** if they were separated by a duplication event.
**Sequence homology**

<table>
<thead>
<tr>
<th>Definition (homology)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two sequences (or genes) are <strong>homologous</strong> if they descend from a common ancestor sequence.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition (orthology)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two homologous sequences (or genes) are <strong>orthologous</strong> if they were separated by a speciation event.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition (paralogy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two homologous sequences (or genes) are <strong>paralogous</strong> if they were separated by a duplication event.</td>
</tr>
</tbody>
</table>
Sequence homology

**Definition (homology)**

Two sequences (or genes) are **homologous** if they descend from a common ancestor sequence.

**Definition (orthology)**

Two homologous sequences (or genes) are **orthologous** if they were separated by a speciation event.

**Definition (paralogy)**

Two homologous sequences (or genes) are **paralogous** if they were separated by a duplication event.
1 Alignments

Pairwise Alignments
- Dot plot
- Pairwise alignment
- Scoring matrices
- Motivation of log-odds
- Affine gap scores

Multiple Alignments
- Motivation and definitions
- Local multiple alignments
- Scoring multiple alignments
- Brute force multiple alignment
- Progressive alignment
### A pairwise protein alignment

<table>
<thead>
<tr>
<th></th>
<th>HUMAN</th>
<th>SHARK</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example (human and shark haemoglobin subunit alpha proteins)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HUMAN</td>
<td>------MVLSPADKTNVKAAWGKVGAGHAGEYGAEEALERMFLSFPTTKTY</td>
<td>STSTSTSDYSAADRAELAALSKVLDAQNAEAFGAEEALARMFTVYAATKSY</td>
</tr>
<tr>
<td>SHARK</td>
<td>FPHF-DLSHGSAQVKGHGKKVADALTNAVAHVDDMPNLALSDLHAK</td>
<td>FKDYKDFTAAAPSIAHGAKVVTALAKACDHLDDLKTHLHKLATFHGSE</td>
</tr>
<tr>
<td>HUMAN</td>
<td>LRVDPVNFKLLSHCLLVTLAAHLPAEFTPAPVHASLKDASKFASVSTVLTSKYR</td>
<td>LKVDPANFQYLSYCLEVALAVHL-TEFSPETHCALDKFTNVCHELSSRYR</td>
</tr>
</tbody>
</table>
### A pairwise protein alignment

**Example (human and shark haemoglobin subunit alpha proteins)**

<table>
<thead>
<tr>
<th></th>
<th>Human</th>
<th>Shark</th>
</tr>
</thead>
<tbody>
<tr>
<td>HUMAN</td>
<td>------MVLSPADKTNVKAAWGKVGAGHAGEYGAEEALERMFLSFPTTKTY</td>
<td></td>
</tr>
<tr>
<td>SHARK</td>
<td>STSTSTSDYSAADRAELAALSKVLAQNAEAFGAEALARMFTVYAATKSY</td>
<td></td>
</tr>
<tr>
<td>HUMAN</td>
<td>FPHF-DLSHGSAQVKGHGKKVADALTNAVAHVDDMPNALSDLHAHK</td>
<td></td>
</tr>
<tr>
<td>SHARK</td>
<td>FKDYKDFTAAPSIKAHGAKVVTALAKACDHLDLKLTHHLKLATFHGSE</td>
<td></td>
</tr>
<tr>
<td>HUMAN</td>
<td>LRVDPVNFKLLSHCLLVTAAHLPAEFPAPHAVASLDKFLASVSTVLTSKYR</td>
<td></td>
</tr>
<tr>
<td>SHARK</td>
<td>LKVDPANFQYLSYCLEVALAVHL-TEFSPEHTCALDKFLTNVCHELSSRYR</td>
<td></td>
</tr>
</tbody>
</table>

**Mutation:** A mutation in either lineage has lead to a difference.

**Mutation:** A mutation in either lineage has lead to a difference.

**Insertion:** An insertion in the protein from shark with respect to the human protein.

**Deletion:** A deletion in the protein from shark with respect to the human protein.

Note that the terms *insertion* and *deletion* are unclear here. It is not clear from this pairwise alignment whether in the third line there has been a deletion in the shark lineage or an insertion in the human lineage.
**A pairwise protein alignment**

**Example (human and shark haemoglobin subunit alpha proteins)**

<table>
<thead>
<tr>
<th>HUMAN</th>
<th>MVLSPADKTNVKAAWGKVGAGHAGEYGAEEALERMFLSFPTTKTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHARK</td>
<td>STSTSTSDYSAADRAELAALSKVLAQNAEAFGAEALARMFTVYAAATKSY</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HUMAN</th>
<th>FPHF-DLSHGSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHK</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHARK</td>
<td>FKDYKDFTAAPSIAKHGAKVVTALAKACDHLDDLKTHLHKLATFHGSE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HUMAN</th>
<th>LRVDPVNFKLLSHCLLLVTLAHLPAEFTPAVHASLDKFLASVSTVLTSKYR</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHARK</td>
<td>LKVDPANFQYLSCYCLEVALAVHL-TEFSPETHCALDKFLTNVCHELSSRYR</td>
</tr>
</tbody>
</table>

**Insertion:** An insertion in the protein from shark with respect to the human protein.
### Example (human and shark haemoglobin subunit alpha proteins)

<table>
<thead>
<tr>
<th>HUMAN</th>
<th>------MVLSPADKTNVKAAWGKVG AHAGEYGAELERMFLSFPTTKTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHARK</td>
<td>STSTSTSDYSAADRAELAALSKVLAQNAEAFGAEALARMFTVYAATKSY</td>
</tr>
<tr>
<td>HUMAN</td>
<td>FPHF-DLSHGSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHK</td>
</tr>
<tr>
<td>SHARK</td>
<td>FKDYKDFTAAPSIKAHGAKVVTALAKACDHLDDLKTHLHKLATFHGSE</td>
</tr>
<tr>
<td>HUMAN</td>
<td>LRVDPVFNKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR</td>
</tr>
<tr>
<td>SHARK</td>
<td>LKVDPANFQYLSYCLEVALAVHL-TEFSPE THCALDKFLTNVCHELSSRYR</td>
</tr>
</tbody>
</table>

**Deletion:** A deletion in the protein from shark with respect to the human protein.

**Mutation:** A mutation in either lineage has lead to a difference. **Insertion:** An insertion in the protein from shark with respect to the human protein. **Deletion:** A deletion in the protein from shark with respect to the human protein. Note that the terms *insertion* and *deletion* are unclear here. It is not clear from this pairwise alignment whether in the third line there has been a deletion in the shark lineage or an insertion in the human lineage.
### A pairwise protein alignment

**Example (human and shark haemoglobin subunit alpha proteins)**

<table>
<thead>
<tr>
<th>HUMAN</th>
<th>------MVLSPADKTNVKAAWGKVGAGHAGEYGAEEALERMFLSFPTTTKTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHARK</td>
<td>SSTDYSAADRAEAAALSKVLAQNAEAFGAEEALARMFTVYAATKSY</td>
</tr>
<tr>
<td>HUMAN</td>
<td>FPHF-DLSHGSAPQVKGHGKKVADALTNAVAVHVDDMPNALSALSDLHAHK</td>
</tr>
<tr>
<td>SHARK</td>
<td>FKDYKDFTAAAPSIAKAHGAKVVTALAKACDHLDLKLTHLHKLATFHGSE</td>
</tr>
<tr>
<td>HUMAN</td>
<td>LRVDPVFKNKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSSKYR</td>
</tr>
<tr>
<td>SHARK</td>
<td>LKVDPANQYLSYCEMAILAVHL-TEFSPEHTCALDKFLTNVCHELSSRYR</td>
</tr>
</tbody>
</table>

**Mutation**: A mutation in either lineage has lead to a difference.

**Insertion**: An insertion in the protein from shark with respect to the human protein.

**Deletion**: A deletion in the protein from shark with respect to the human protein.

Note that the terms *insertion* and *deletion* are unclear here. It is not clear from this pairwise alignment whether in the third line there has been a deletion in the shark lineage or an insertion in the human lineage.
1 Alignments

Pairwise Alignments
- Dot plot
- Pairwise alignment
- Scoring matrices
- Motivation of log-odds
- Affine gap scores

Multiple Alignments
- Motivation and definitions
- Local multiple alignments
- Scoring multiple alignments
- Brute force multiple alignment
- Progressive alignment
Dot Plot

- simple visualization tool for pairwise sequence comparison
- usually for DNA sequences (z.B. genomes, chromosomes)
- allows to detect repeats and inversions
Dot Plot

Example (dot plot)

compare two sequences $S$ and $T$:

$S = YQEWTYIVAREAQYE$

$T = CIVMREQY$
Dot Plot

Example (dot plot)

compare two sequences $S$ and $T$:

\[
S = \text{YQEWTYIVAREAQYE} \\
T = \text{CIVMREQY}
\]

Draw a “dot” at $(i, j)$ iff $S[i] = T[j]$. 
**Similarities in a Dot Plot**

Local similarities appear as diagonal line on the main diagonal

```
  Y   Q   E   W   T   Y   I   V   A   R   E   A   Q   Y   E
```

```
  C   I   V   M   R   E   Q   Y
```

...
Dot Plot

Similarities in a Dot Plot

... or as diagonal lines (minor diagonal), e.g. inversions.
Dot Plot

Segmental Duplications

Y Q E W T Y Q E A R E Y Q E I

C I V M R Y Q E
Comparing DNA one has to expect at least 25% of identity. Therefore naive dot plots show a lot of “noise”, which can be suppressed by drawing a dot at \((i, j)\) only if there are at least \(s\) matches in a diagonal of size \(w\) centered around \((i, j)\).
Dot Plot

Example (Two algae (Chlamy vs Volvox))

http://genome.jgi-psf.org/synteny
# Alignments

## Pairwise Alignments
- Dot plot
- Pairwise alignment

## Scoring matrices
- Motivation of log-odds
- Affine gap scores

## Multiple Alignments
- Motivation and definitions
- Local multiple alignments
- Scoring multiple alignments
- Brute force multiple alignment
- Progressive alignment
Definition: Pairwise Alignment

The purpose of alignments is

to softly match sequences, allowing for mutations, insertions and deletions.

Definition (Global Pairwise Alignment)

Let $S$ and $T$ be two strings over an alphabet $\Sigma$. Let $’-$’ denote a special gap symbol that is not in $\Sigma$. For any string $U$ over alphabet $\Sigma \cup \{-\}$, let $g(U)$ denote the string that is obtained when all gap symbols in $U$ are removed.

An global pairwise alignment of $S$ and $T$ is then a pair of strings $S'$ and $T'$ over $\Sigma \cup \{-\}$ such that

- $S = g(S')$,
- $T = g(T')$,
- $|S'| = |T'|$ and
- $\nexists i$ such that $S'[i] = T'[i] =’-’.$
Alignment Terms

- the positions of $S'$ and $T'$ are called **alignment columns**
- $S'$ and $T'$ are called **rows**
- a **match** is a column with identical (or similar) letters in both rows
- a **mismatch** is a column with different (or dissimilar) letters in both rows
- an **insertion** is a letter in the same column as a gap
- a **deletion** is a gap in the same column as a letter
- an **indel** is a column containing a gap
Purpose of Alignments

Pairwise alignments are relevant

- as a first step towards structural and functional analysis of newly determined sequences
- for genome comparisons
- for exploring genetic variations among individuals
- for cDNA-to-genome alignments \( \Rightarrow \) gene identification
- for database similarity searching
**Definition: Sequence Identity**

"Definition" of sequence identity

Let \((S', T')\) be an alignment of strings \(S\) and \(T\) and let

\[
I := \# \{ i \mid S'[i] = T'[i] \}
\]

be the number of aligned identical characters. Then the sequence identity \(ID\) is the percentage of aligned identical characters.
Definition: Sequence Identity

“Definition” of sequence identity

Let \((S', T')\) be an alignment of strings \(S\) and \(T\) and let

\[ I := \#\{ i \mid S'[i] = T'[i] \} \]

be the number of aligned identical characters. Then the sequence identity \(ID\) is the percentage of aligned identical characters.

Unfortunately, there are many variants for the denominator

1. \(ID := I / \min(|S|, |T|)\)

and there is no consensus on which to use. Even worse, often the denominator is not stated.
Definition: Sequence Identity

“Definition” of sequence identity

Let \((S', T')\) be an alignment of strings \(S\) and \(T\) and let

\[
I := \#\{i \mid S'[i] = T'[i]\}
\]

be the number of aligned identical characters. Then the sequence identity \(ID\) is the percentage of aligned identical characters.

Unfortunately, there are many variants for the denominator:

1. \(ID := I / \min(|S|, |T|)\)
2. \(ID := I / |S'|\)

and there is no consensus on which to use. Even worse, often the denominator is not stated.
Definition: Sequence Identity

“Definition” of sequence identity

Let \((S', T')\) be an alignment of strings \(S\) and \(T\) and let

\[
I := \# \{ i \mid S'[i] = T'[i] \}
\]

be the number of aligned identical characters. Then the sequence identity \(ID\) is the percentage of aligned identical characters.

Unfortunately, there are many variants for the denominator:

1. \(ID := I / \min(|S|, |T|)\)
2. \(ID := I / |S'|\)
3. \(ID := I / \# \{ i \mid S'[i] \neq -T'[i] \}\)

and there is no consensus on which to use. Even worse, often the denominator is not stated.
Definition: Sequence Identity

“Definition” of sequence identity

Let \((S', T')\) be an alignment of strings \(S\) and \(T\) and let

\[
I := \# \{ i \mid S'[i] = T'[i] \}
\]

be the number of aligned identical characters. Then the sequence identity \(ID\) is the percentage of aligned identical characters.

Unfortunately, there are many variants for the denominator

1. \(ID := I / \min(|S|, |T|)\)
2. \(ID := I / |S'|\)
3. \(ID := I / \# \{ i \mid S'[i] \neq T'[i] \}\)
4. \(ID := 2I / (|S| + |T|)\)

and there is no consensus on which to use. Even worse, often the denominator is not stated.
Number of Alignments

How many alignments are there?

Let $m = |S|$ and $n = |T|$ be the lengths of the two strings.
Let $0 \leq k \leq \min(m, n)$ be the number of matches (:= no gap) of an alignment $(S', T')$ of $S$ and $T$. Then $S'$ must contain $m - k$ gaps and $T'$ must contain $n - k$ gaps.

The number $A(m, n)$ of alignments of two strings of lengths $m$ and $n$ is therefore

$$A(m, n) = \sum_{k=0}^{\min(m, n)} \text{number of alignments with } k \text{ matches}$$

$$= \sum_{k=0}^{\min(m, n)} \text{number of possibilities to distribute } m + n - k \text{ columns to } k \text{ matches, } m - k \text{ gaps on } S' \text{ and } n - k \text{ gaps on } T'$$

$$= \sum_{k=0}^{\min(m, n)} \frac{(m + n - k)!}{k!(m - k)!(n - k)!}$$

$$= \Omega(2^{\min(m, n)})$$
Number of Alignments

For example,

\[ A(10, 10) = 8097453, \]

i.e. there are already more than 8 million alignments of two strings of length 10.
For example,

\[ A(10, 10) = 8097453, \]

i.e. there are already more than 8 million alignments of two strings of length 10.

Typical problem sizes \( \min(n, m) \) for alignments are in the order of hundreds and can go up to millions.
Scoring of an alignment

How should we pick of all possible alignments a good one that approximately reflects the true history of the sequences (mutations, insertions, deletions)?
Scoring of an alignment

How should we pick of all possible alignments a good one that approximately reflects the true history of the sequences (mutations, insertions, deletions)?

Scoring scheme

Let $A = (S', T')$ be an alignment of sequences $S$ and $T$. A simple but common way is to assign a score to each column of the alignment and then sum up these scores to score the alignment.

Definition (scoring matrix)

A scoring matrix is a function

$$s : (\Sigma \cup \{-\})^2 \rightarrow \mathbb{R}.$$
Intuition of scoring scheme

A scoring matrix \( s \) will typically give

- high values to similar or identical characters:
  \[ s(a, a) > 0 \]

Mismatches, insertions and deletions are punished. More on the topics of scoring matrices later.
Intuition of scoring scheme

A scoring matrix \( s \) will typically give

- high values to similar or identical characters:
  \[ s(a, a) > 0 \]
- low (or negative) values to dissimilar characters:
  \[ s(a, b) \text{ with } a \neq b \]
Scoring of an alignment

Intuition of scoring scheme

A scoring matrix $s$ will typically give

- high values to similar or identical characters: $s(a, a) > 0$

- low (or negative) values to dissimilar characters: $s(a, b)$ with $a \neq b$

- negative values to indels: $s(a, -), s(-, b) < 0$

Mismatches, insertions and deletions are punished.

More on the topics of scoring matrices later.
Intuition of scoring scheme

A scoring matrix $s$ will typically give

- high values to similar or identical characters: $s(a, a) > 0$
- low (or negative) values to dissimilar characters: $s(a, b)$ with $a \neq b$
- negative values to indels: $s(a, -), s(-, b) < 0$
Scoring of an alignment

Intuition of scoring scheme

A scoring matrix \( s \) will typically give

- high values to similar or identical characters:
  \( s(a, a) > 0 \)

- low (or negative) values to dissimilar characters:
  \( s(a, b) \) with \( a \neq b \)

- negative values to indels:
  \( s(a, -), s(-, b) < 0 \)

Mismatches, insertions and deletions are punished.
Scoring of an alignment

Intuition of scoring scheme

A scoring matrix $s$ will typically give

- high values to similar or identical characters:
  $s(a, a) > 0$

- low (or negative) values to dissimilar characters:
  $s(a, b)$ with $a \neq b$

- negative values to indels:
  $s(a, -), s(-, b) < 0$

Mismatches, insertions and deletions are punished.

More on the topics of scoring matrices later.
Scoring of an alignment

**Definition (score of an alignment)**

Given a scoring matrix $s$ the score of an alignment $A = (S', T')$ with $\ell$ columns is defined as

$$s(A) := \sum_{i=1}^{\ell} s(S'[i], T'[i]).$$

**Example (alignment and score)**

(Chalk board)
Finding a best-scoring alignment

**Needleman-Wunsch Problem**

Find a global alignment of two sequences $S$ and $T$ with maximal score.

- named after two authors who first discussed global alignments in 1970.
- can be solved using Dynamic Programming.
Finding a best-scoring alignment

**Definition (DP variables)**

Let

\[ M(i, j) := \text{maximal score of an alignment of } S[1..i] \text{ and } T[1..j] \]

for \( i = 1, \ldots, m := |S| \) and \( j = 1, \ldots, n := |T| \).

Define \( M(0, 0) := 0 \).
Finding a best-scoring alignment

### DP recursion

\[
M(i, j) = \max \begin{cases} 
M(i - 1, j - 1) + s(S[i], T[j]) \\
M(i - 1, j) + s(S[i], -) \\
M(i, j - 1) + s(-, T[j]) 
\end{cases}
\]  

for \( i, j > 0 \).

\[
M(i, 0) = M(i - 1, 0) + s(S[i], -)
\]  

\[
M(0, j) = M(0, j - 1) + s(-, T[j])
\]  

for \( i, j > 0 \).
Explanation of recursion

Let $i, j > 0$. There are three possibilities for the *last column* of an optimal alignment of $S[1..i]$ and $T[1..j]$:

- *no gap*: $S[i]$ and $T[j]$ are aligned
  
  $$M(i, j) = M(i-1, j-1) + s(S[i], T[j])$$

- *gap in $T$*: $S[i]$ is aligned to a gap
  
  $$M(i, j) = M(i-1, j) + s(S[i], -)$$

- *gap in $S$*: $T[j]$ is aligned to a gap
  
  $$M(i, j) = M(i, j-1) + s(-, T[j])$$

Since $M(i, j)$ is the score of the optimal alignment, it is the maximum of the three and therefore we have (4). (5) and (6) are special cases in which only the second or third possibility is possible.
**Explanation of recursion**

Let $i, j > 0$.

There are three possibilities for the *last column* of an optimal alignment of $S[1..i]$ and $T[1..j]$:

<table>
<thead>
<tr>
<th>No gap: $S[i]$ and $T[j]$ are aligned</th>
<th>Gap in $T$: $S[i]$ is aligned to a gap</th>
<th>Gap in $S$: $T[j]$ is aligned to a gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M(i, j) = M(i-1, j-1) + s(S[i], T[j])$</td>
<td>$M(i, j) = M(i-1, j) + s(S[i], T[j])$</td>
<td>$M(i, j) = M(i, j-1) + s(−, T[j])$</td>
</tr>
</tbody>
</table>

Since $M(i, j)$ is the score of the optimal alignment, it is the maximum of the three, and therefore we have (4).

(5) and (6) are special cases in which only the second or third possibility is possible.
**Explanation of recursion**

Let $i, j > 0$.

There are three possibilities for the *last column* of an optimal alignment of $S[1..i]$ and $T[1..j]$:

- **no gap:** $S[i]$ and $T[j]$ are aligned
  
  
  $M(i, j) = M(i-1, j-1) + s(S[i], T[j])$

- **gap in $T$:** $S[i]$ is aligned to a gap
  
  
  $M(i, j) = M(i-1, j) + s(S[i], -)$

- **gap in $S$:** $T[j]$ is aligned to a gap
  
  
  $M(i, j) = M(i, j-1) + s(-, T[j])$

Since $M(i, j)$ is the score of the optimal alignment it is the maximum of the three and therefore we have (4).
**DP recursion**

**Explanation of recursion**

Let $i, j > 0$.
There are three possibilities for the *last column* of an optimal alignment of $S[1..i]$ and $T[1..j]$: 

- **no gap:** $S[i]$ and $T[j]$ are aligned
  
  $M(i, j) = M(i-1, j-1) + s(S[i], T[j])$

- **gap in $T$:** $S[i]$ is aligned to a gap
  
  $M(i, j) = M(i-1, j) + s(S[i], -)$

- **gap in $S$:** $T[j]$ is aligned to a gap
  
  $M(i, j) = M(i, j-1) + s(-, T[j])$

Since $M(i, j)$ is the score of the optimal alignment it is the maximum of the three and therefore we have (4).

(5) and (6) are special cases in which only the second or third possibility is possible.
Needleman-Wunsch algorithm

1: \( M(0, 0) \leftarrow 0 \) // initialization of boundary case
2: for \( i = 1 \) to \( m \) do
3: \hspace{1em} for \( j = 1 \) to \( n \) do
4: \hspace{2em} compute \( M(i, j) \) according to DP recursion
5: \hspace{2em} set \( p(i, j) \) to either \((i - 1, j - 1)\), \((i - 1, j)\) or \((i, j - 1)\) corresponding to the entry of \( M \) that was used in line 4
6: \hspace{1em} end for
7: end for
8: output \( M(m, n) \) as optimal global alignment score
9: // backtracking starts
10: \( A \leftarrow \langle \text{empty alignment} \rangle \)
11: \( (i, j) \leftarrow (m, n) \)
12: repeat
13: \hspace{1em} depending on whether \( p(i, j) \) is \((i - 1, j - 1)\), \((i - 1, j)\) or \((i, j - 1)\) add alignment column \( \begin{pmatrix} S[i] \\ T[j] \end{pmatrix} \), \( \begin{pmatrix} S[i] \\ - \end{pmatrix} \) or \( \begin{pmatrix} - \\ T[j] \end{pmatrix} \), respectively, to left of \( A \)
14: \( (i, j) \leftarrow p(i, j) \)
15: until \( (i, j) = 0 \)
16: output \( A \) as optimal alignment
Needleman-Wunsch algorithm

<table>
<thead>
<tr>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>• running time: $O(mn)$</td>
</tr>
<tr>
<td>• memory: $O(mn)$</td>
</tr>
</tbody>
</table>
Needleman-Wunsch algorithm

Complexity

- **running time**: $O(mn)$
- **memory**: $O(mn)$

Variant of the algorithm

- do not store the matrix $p(\cdot, \cdot)$ of backtracking pointers
- instead, recompute the maximum each time during traceback
Needleman-Wunsch algorithm

### Complexity

- **running time**: $O(mn)$
- **memory**: $O(mn)$

### Variant of the algorithm

- do not store the matrix $p(\cdot, \cdot)$ of backtracking pointers
- instead, recompute the maximum each time during traceback

The running time of this variant is still $O(mn)$:
We add only $O(m + n)$ steps.
Needleman-Wunsch algorithm

### Complexity

- **running time**: $O(mn)$
- **memory**: $O(mn)$

### Variant of the algorithm

- do not store the matrix $p(\cdot, \cdot)$ of backtracking pointers
- instead, recompute the maximum each time during traceback

The running time of this variant is still $O(mn)$:
We add only $O(m + n)$ steps.
We save the space for $p(\cdot, \cdot)$. Depending on the representation of $p(\cdot, \cdot)$ and $M(\cdot, \cdot)$ this may be up to half the space.
Local alignment

**Definition (local alignment problem)**

Given two strings $S$ and $T$, find substrings $s$ of $S$ and $t$ of $T$, whose similarity (optimal global alignment score) is maximal over all pairs of substrings from $S$ and $T$. 
Local alignment

Definition (local alignment problem)

Given two strings $S$ and $T$, find substrings $s$ of $S$ and $t$ of $T$, whose similarity (optimal global alignment score) is maximal over all pairs of substrings from $S$ and $T$.

Motivation

When comparing long sequences often only a smaller part is conserved or alignable. E.g.

- protein domain
- cDNA to genome alignment
Local alignment

Example (DNA alignment)

global alignment:

```
--T--CC-C-AGT--TATGT-CAGGGGACACG-A-GCATGCAGA-GAC
 |
| |
AATTGCCGCC--GTCGT-T----TCAG----CA-GTTATG-T-CAGAT--C
```

local alignment:

```
tccCAGTTATGTCAGgggacacgagcatgcagagac
 |
| |
| |
| |
aattgccgcccgtcgttttcagCAGTTATGTCAGatc
```

toy example from Clemens Gröpl
Smith-Waterman algorithm

Finding a best-scoring \emph{local} alignment

- can be done using a DP algorithm
Smith-Waterman algorithm

Finding a best-scoring *local* alignment

- can be done using a DP algorithm
- which is a variant of the Needleman-Wunsch algorithm
Smith-Waterman algorithm

Finding a best-scoring *local* alignment

- can be done using a DP algorithm
- which is a variant of the Needleman-Wunsch algorithm
- a local alignment can now **start and end at any position** in the “alignment space” $[1..m] \times [1..n]$
Smith-Waterman algorithm

Finding a best-scoring \textit{local} alignment

- can be done using a DP algorithm
- which is a variant of the Needleman-Wunsch algorithm
- a local alignment can now start and end at any position in the "alignment space" $[1..m] \times [1..n]$
- an arbitrary start can be achieved by adding an option to 0 to the recursion
Smith-Waterman algorithm

Finding a best-scoring *local* alignment

- can be done using a DP algorithm
- which is a variant of the Needleman-Wunsch algorithm
- a local alignment can now *start and end at any position* in the “alignment space” $[1..m] \times [1..n]$
- an arbitrary start can be achieved by *adding an option to 0* to the recursion
- an arbitrary end can be achieved by maximizing over all entries of the DP-matrix
Local pairwise alignment

**Definition (DP variables)**

Let

$$N(i, j) := \text{maximal score of a global alignment of a suffix of } S[1..i] \text{ and a suffix of } T[1..j]$$

for $i = 1, \ldots, m := |S|$ and $j = 1, \ldots, n := |T|$.

Define $N(0, 0) := 0$.

String $a$ is a *suffix* of string $b$ of length $n$ if $a = b[i..n]$ for some $i$. 
Local pairwise alignment

\[ N(i, j) = \max \begin{cases} 
0 \\
N(i - 1, j - 1) + s(S[i], T[j]) \\
N(i - 1, j) + s(S[i], -) \\
N(i, j - 1) + s(-, T[j])
\end{cases} \]
Smith-Waterman algorithm

1: $M(0, 0) \leftarrow 0$ // initialization of boundary case
2: $p(0, 0) \leftarrow (-1, -1)$ // initialization of predecessor pointer to undefined value
3: for $i = 1$ to $m$ do
4:   for $j = 1$ to $n$ do
5:     compute $N(i, j)$ according to DP recursion (8)
6:     set $p(i, j)$ to either $(-1, -1), (i-1, j-1), (i-1, j)$ or $(i, j-1)$
7:     corresponding to whether the first, second, third or fourth argument of max
8:     in (8) is maximal
9:   end for
10: end for
11: $(i, j) \leftarrow \text{argmax}\{N(a, b) | 1 \leq a \leq m, 1 \leq b \leq n\}$
12: output $N(i, j)$ as optimal local alignment score
13: // backtracking starts
14: $A \leftarrow \langle$ empty alignment $\rangle$
15: while $p(i, j) \neq (-1, -1)$ do
16:   depending on whether $p(i, j)$ is $(i-1, j-1), (i-1, j)$ or $(i, j-1)$ add
17:     alignment column $\left( S[i] \atop T[j] \right)$, $\left( S[i] \atop - \right)$ or $\left( - \atop T[j] \right)$, respectively, to left of $A$
18:   $(i, j) \leftarrow p(i, j)$
19: end while
20: output $A$ as optimal alignment
Alignments

Pairwise Alignments
- Dot plot
- Pairwise alignment
- Scoring matrices
  - Motivation of log-odds
  - Affine gap scores

Multiple Alignments
- Motivation and definitions
- Local multiple alignments
- Scoring multiple alignments
  - Brute force multiple alignment
  - Progressive alignment
## Amino Acid Properties

<table>
<thead>
<tr>
<th>Amino Acids</th>
<th>3-letter</th>
<th>1-letter</th>
<th>Polarity</th>
<th>Charge</th>
<th>Hydrophathy Index</th>
<th>Average Res. Mass [Daltons]</th>
<th>Accessible Surface Area [Å²]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alanine</td>
<td>Ala</td>
<td>A</td>
<td>Nonpolar</td>
<td>Neutral</td>
<td>1.8</td>
<td>71.1</td>
<td>67</td>
</tr>
<tr>
<td>Arginine</td>
<td>Arg</td>
<td>R</td>
<td>Polar</td>
<td>Positive</td>
<td>-4.5</td>
<td>156.2</td>
<td>196</td>
</tr>
<tr>
<td>Asparagine</td>
<td>Asn</td>
<td>N</td>
<td>Polar</td>
<td>Neutral</td>
<td>-3.5</td>
<td>114.1</td>
<td>113</td>
</tr>
<tr>
<td>Aspartic acid</td>
<td>Asp</td>
<td>D</td>
<td>Polar</td>
<td>Negative</td>
<td>-3.5</td>
<td>115.1</td>
<td>106</td>
</tr>
<tr>
<td>Cysteine</td>
<td>Cys</td>
<td>C</td>
<td>Nonpolar</td>
<td>Neutral</td>
<td>2.5</td>
<td>103.1</td>
<td>104</td>
</tr>
<tr>
<td>Glutamic acid</td>
<td>Glu</td>
<td>E</td>
<td>Polar</td>
<td>Neutral</td>
<td>-3.5</td>
<td>129.1</td>
<td>138</td>
</tr>
<tr>
<td>Glutamine</td>
<td>Gln</td>
<td>Q</td>
<td>Polar</td>
<td>Neutral</td>
<td>-3.5</td>
<td>128.1</td>
<td>144</td>
</tr>
<tr>
<td>Glycine</td>
<td>Gly</td>
<td>G</td>
<td>Nonpolar</td>
<td>Neutral</td>
<td>0.0</td>
<td>57.1</td>
<td>60</td>
</tr>
<tr>
<td>Histidine</td>
<td>His</td>
<td>H</td>
<td>Polar</td>
<td>Positive</td>
<td>-3.2</td>
<td>137.1</td>
<td>151</td>
</tr>
<tr>
<td>Isoleucine</td>
<td>Ile</td>
<td>I</td>
<td>Nonpolar</td>
<td>Neutral</td>
<td>4.5</td>
<td>113.2</td>
<td>140</td>
</tr>
<tr>
<td>Leucine</td>
<td>Leu</td>
<td>L</td>
<td>Nonpolar</td>
<td>Neutral</td>
<td>3.8</td>
<td>113.2</td>
<td>137</td>
</tr>
<tr>
<td>Lysine</td>
<td>Lys</td>
<td>K</td>
<td>Polar</td>
<td>Positive</td>
<td>-3.9</td>
<td>128.2</td>
<td>167</td>
</tr>
<tr>
<td>Methionine</td>
<td>Met</td>
<td>M</td>
<td>Nonpolar</td>
<td>Neutral</td>
<td>1.9</td>
<td>131.2</td>
<td>160</td>
</tr>
<tr>
<td>Phenylalanine</td>
<td>Phe</td>
<td>F</td>
<td>Nonpolar</td>
<td>Neutral</td>
<td>2.8</td>
<td>147.2</td>
<td>175</td>
</tr>
<tr>
<td>Proline</td>
<td>Pro</td>
<td>P</td>
<td>Nonpolar</td>
<td>Neutral</td>
<td>-1.6</td>
<td>97.1</td>
<td>105</td>
</tr>
<tr>
<td>Serine</td>
<td>Ser</td>
<td>S</td>
<td>Polar</td>
<td>Neutral</td>
<td>-0.8</td>
<td>87.1</td>
<td>80</td>
</tr>
<tr>
<td>Threonine</td>
<td>Thr</td>
<td>T</td>
<td>Polar</td>
<td>Neutral</td>
<td>-0.7</td>
<td>101.1</td>
<td>102</td>
</tr>
<tr>
<td>Tryptophan</td>
<td>Trp</td>
<td>W</td>
<td>Nonpolar</td>
<td>Neutral</td>
<td>-0.9</td>
<td>186.2</td>
<td>217</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>Tyr</td>
<td>Y</td>
<td>Polar</td>
<td>Neutral</td>
<td>-1.3</td>
<td>163.2</td>
<td>187</td>
</tr>
<tr>
<td>Valine</td>
<td>Val</td>
<td>V</td>
<td>Nonpolar</td>
<td>Neutral</td>
<td>4.2</td>
<td>99.1</td>
<td>117</td>
</tr>
</tbody>
</table>
Amino Acid Properties

Amino Acid Venn Diagram
Amino Acid Properties

For the purpose of this class we need not know any particular amino acid properties.
Amino Acid Properties

For the purpose of this class we **need not know any particular amino acid properties.**

But we need to take into account that

1. at a particular position in a protein, an amino acid with particular properties may be required for function
Amino Acid Properties

For the purpose of this class we **need not know any particular amino acid properties.**

But we need to take into account that

1. at a particular position in a protein, an amino acid with particular properties may be required for function
2. some set of amino acids may be similarly fit
Amino Acid Properties

For the purpose of this class we **need not know any particular amino acid properties.**

But we need to take into account that

1. at a particular position in a protein, an amino acid with particular properties may be required for function
2. some set of amino acids may be similarly fit
3. simplifying, we can define an “overall similarity” for each pair of amino acids
Amino Acid Properties

For the purpose of this class we need not know any particular amino acid properties.

But we need to take into account that

1. at a particular position in a protein, an amino acid with particular properties may be required for function
2. some set of amino acids may be similarly fit
3. simplifying, we can define an “overall similarity” for each pair of amino acids
4. Remark: Later, we will also use a more detailed model that takes into account that a residue may have a certain property, that can not be defined through pairwise similarity
Score of a protein alignment

Reminder:
Neglect gaps for now.
The score of an alignment $A = (S', T')$ of two protein sequences $S$ and $T$ is the sum of the scores of the alignment columns:

$$s(A) = \sum_i s(S'[i], T'[i])$$
Score of a protein alignment

Reminder:
Neglect gaps for now.
The score of an alignment $A = (S', T')$ of two protein sequences $S$ and $T$ is the sum of the scores of the alignment columns:

$$s(A) = \sum_i s(S'[i], T'[i])$$

Scoring Matrix Principle
For two amino acids $a, b$, the score $s(a, b)$ is usually defined through

$$s(a, b) = \log \frac{p(a, b)}{q(a)q(b)},$$

(9)

where $p(a, b)$ is the probability of observing $a$ aligned to $b$ in correct alignments and $q(a), q(b)$ are the relative overall frequencies of amino acids $a$ and $b$, respectively.
Score of a protein alignment

Pairwise Scores

Some intuitive properties of (9):

- $s(\cdot, \cdot)$ is symmetric
**Score of a protein alignment**

### Pairwise Scores

Some intuitive properties of (9):

- $s(\cdot, \cdot)$ is symmetric
- probability of co-occurrence in the same column of correct alignments is a fuzzy concept, concretizing it leads to different scores (e.g. BLOSUM, PAM)

<table>
<thead>
<tr>
<th>Pairwise Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some intuitive properties of (9):</td>
</tr>
<tr>
<td>• $s(\cdot, \cdot)$ is symmetric</td>
</tr>
<tr>
<td>• probability of co-occurrence in the same column of correct alignments is a fuzzy concept, concretizing it leads to different scores (e.g. BLOSUM, PAM)</td>
</tr>
</tbody>
</table>
Score of a protein alignment

Pairwise Scores

Some intuitive properties of (9):

- $s(\cdot, \cdot)$ is symmetric
- probability of co-occurrence in the same column of correct alignments is a fuzzy concept, concretizing it leads to different scores (e.g. BLOSUM, PAM)
- $q(a)q(b)$ is the probability of pairings $(a, b)$ in random alignments of unrelated sequences
Score of a protein alignment

**Pairwise Scores**

Some intuitive properties of (9):

- $s(\cdot, \cdot)$ is symmetric
- probability of co-occurrence in the same column of correct alignments is a fuzzy concept, concretizing it leads to different scores (e.g. BLOSUM, PAM)
- $q(a)q(b)$ is the probability of pairings $(a, b)$ in random alignments of unrelated sequences
- the score is positive iff $p(a, b) > q(a)q(b)$, i.e. if $a, b$ co-occur more often than expected by chance
Score of a protein alignment

Pairwise Scores

Some intuitive properties of (9):

- $s(\cdot, \cdot)$ is symmetric
- probability of co-occurrence in the same column of correct alignments is a fuzzy concept, concretizing it leads to different scores (e.g. BLOSUM, PAM)
- $q(a)q(b)$ is the probability of pairings $(a, b)$ in random alignments of unrelated sequences
- the score is positive iff $p(a, b) > q(a)q(b)$, i.e. if $a, b$ co-occur more often than expected by chance
- normalizing with $q(a)q(b)$ has the effect that rare amino acids can have high scores
Score of a protein alignment

Pairwise Scores

Some intuitive properties of (9):

- \( s(\cdot, \cdot) \) is symmetric
- probability of co-occurrence in the same column of correct alignments is a fuzzy concept, concretizing it leads to different scores (e.g. BLOSUM, PAM)
- \( q(a)q(b) \) is the probability of pairings \((a, b)\) in random alignments of unrelated sequences
- the score is positive iff \( p(a, b) > q(a)q(b) \), i.e. if \( a, b \) co-occur more often than expected by chance
- normalizing with \( q(a)q(b) \) has the effect that rare amino acids can have high scores
- \( s(a, a) > 0 \) and, usually, \( a \) is the amino acid that is most similar to \( a \)
Score of a protein alignment

Pairwise Scores

Must satisfy

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

Interpretation: The expected score of two random unrelated amino acids must be negative. If this was not satisfied, then local alignments would not be meaningful: Long unrelated stretches were likely to be aligned.
### Example of a Scoring Matrix

#### BLOSUM62

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

- example $x = 62$ on previous slide
- made from alignment "BLOCKS", in which any two sequences are less than $x\%$ percent identical
- $p(\cdot, \cdot), q(\cdot)$ estimated using observed relative frequencies in alignment BLOCKS
- scaled and rounded to integers
1 Alignments

Pairwise Alignments
  - Dot plot
  - Pairwise alignment
  - Scoring matrices

Multiple Alignments
  - Motivation and definitions
  - Local multiple alignments
  - Scoring multiple alignments
  - Brute force multiple alignment
  - Progressive alignment
Neyman-Pearson-Lemma

Test of a simple hypothesis against a simple alternative

Suppose, we want to test a (simple) hypothesis \( H_0 \) against a (simple) alternative \( H_1 \). E.g.

\[
H_0 : \text{alignment column } x = \begin{pmatrix} a \\ b \end{pmatrix} \text{ is "not correct"}
\]

\[
H_1 : \text{alignment column } x \text{ is "correct"}
\]

In other words, \( H_0 \) is the \textit{null-hypothesis} that residues \( a \) and \( b \) are unrelated. \( H_1 \) is the alternative that residues \( a \) and \( b \) stem from a common ancestor residue.
Statistical Test

### Decision Function, Type 1 error, power

Let $\phi = \phi(x) \in \{0, 1\}$ be a decision function that assigns to an observation $x$ one of the the two hypotheses $(H_0, H_1)$. 

Let $P(\phi(x) = 1 | H_0)$ be the type 1 error. In our example, the type 1 error is that we conclude that $a$ and $b$ are related although they are not. We want to chose a $\phi$ that keeps the probability of this error low.

Let $P(\phi(x) = 1 | H_1)$ be the power. In our example, the power is the probability that we predict $a$ and $b$ to be related if they really are related. We want to chose a $\phi$ with high power.
Decision Function, Type 1 error, power

Let \( \phi = \phi(x) \in \{0, 1\} \) be a decision function that assigns to an observation \( x \) one of the the two hypotheses (\( H_0, H_1 \)).

Let

\[
P(\phi(x) = 1 \mid H_0) = \text{prob. that we decide for } H_1 \text{ although } H_0 \text{ is true}
\]

be the type 1 error. In our example, the type 1 error is that we conclude that \( a \) and \( b \) are related although they are not. We want to chose a \( \phi \) that keeps the probability of this error low.
### Statistical Test

#### Decision Function, Type 1 error, power

Let \( \phi = \phi(x) \in \{0, 1\} \) be a decision function that assigns to an observation \( x \) one of the two hypotheses \( (H_0, H_1) \).

Let

\[
P(\phi(x) = 1 \mid H_0) = \text{prob. that we decide for } H_1 \text{ although } H_0 \text{ is true}
\]

be the **type 1 error**. In our example, the type 1 error is that we conclude that \( a \) and \( b \) are related although they are not. We want to chose a \( \phi \) that keeps the probability of this error low.

Let

\[
P(\phi(x) = 1 \mid H_1) = \text{prob. that we decide for } H_1 \text{ if } H_1 \text{ is true}
\]

be the **power**. In our example, the power is the probability that we predict \( a \) and \( b \) to be related if they really are related. We want to chose a \( \phi \) with high power.
Neyman-Pearson-Lemma

Among all decision functions that satisfy an upper bound for the type 1 error, a decision function of the following form has maximal power:

\[ \phi(x) = 1 : \Leftrightarrow \frac{P(x|H_1)}{P(x|H_0)} > t \]  \hspace{1cm} (10)

for some constant threshold \( t \).
Neyman-Pearson-Lemma

Among all decision functions that satisfy an upper bound for the type 1 error, a decision function of the following form has maximal power:

\[
\phi(x) = 1 \iff \frac{P(x|H_1)}{P(x|H_0)} > t
\]

for some constant threshold \( t \).

Intuitively, this means that we should decide for the alternative, if the observation \( x \) is such that it is relatively likely in case \( H_1 \) is true (the denominator), compared to when \( H_0 \) is true (the nominator).
Connection between the Neyman-Pearson-Lemma and Scoring Matrices

In the alignment problem we are considering $H_0$ : the two sequences are unrelated and $H_1$ : the two sequences are evolutionary related and want to effectively decide among the two.
In the alignment problem we are considering $H_0$: the two sequences are unrelated and $H_1$: the two sequences are evolutionary related and want to effectively decide among the two. Using a scoring matrix $s(\cdot, \cdot)$ as and a DP-algorithm as above we are finding an alignment which maximizes

$$\sum_i s(S'[i], T'[i]) = \sum_i \log \frac{p(S'[i], T'[i])}{q(S'[i])q(T'[i])}$$

$$= \log \prod_i \frac{p(S'[i], T'[i])}{q(S'[i])q(T'[i])}$$
Connection between the Neyman-Pearson-Lemma and Scoring Matrices

In the alignment problem we are considering $H_0$ : the two sequences are unrelated and $H_1$ : the two sequences are evolutionary related and want to effectively decide among the two. Using a scoring matrix $s(\cdot, \cdot)$ as and a DP-algorithm as above we are finding an alignment which maximizes

$$
\sum_i s(S'[i], T'[i]) = \sum_i \log \frac{p(S'[i], T'[i])}{q(S'[i])q(T'[i])}
$$

$$
= \log \prod_i \frac{p(S'[i], T'[i])}{q(S'[i])q(T'[i])}
$$

As the logarithm is a monotonous function, this is equivalent to

$$
\prod_i \frac{p(S'[i], T'[i])}{q(S'[i])q(T'[i])} = \frac{(\prod_i p(S'[i], T'[i]))}{\prod_i q(S'[i])\prod_i q(T'[i])} \rightarrow \max
$$ (11)
Connection between the Neyman-Pearson-Lemma and Scoring Matrices

The expression in (11) can be interpreted under an independence assumption as

$$\frac{P(S,T \mid H_1)}{P(S,T \mid H_0)}.$$  \hspace{1cm} (12)

By the Neyman-Pearson-Lemma, we find the most truly related sequences (decide for $H_1$ if $H_1$ is true) under the constraint that the type 1 error is bounded (not too many false positives = unrelated sequences that we think are related), when using (12) as criterion. In that sense, the scoring matrix approach is optimal.

We just have to balance out a high power against a low probability of a type 1 error by considering a low alignment score threshold or a higher one.
Alignments

Pairwise Alignments
- Dot plot
- Pairwise alignment
- Scoring matrices
- Motivation of log-odds

Multiple Alignments
- Motivation and definitions
- Local multiple alignments
- Scoring multiple alignments
- Brute force multiple alignment
- Progressive alignment
Example (beetle - bee)

section from an alignment between a protein from the red flour beetle and from honey bee:

```
VNIDGTNCPIKKTFVVPKSPNDLDVYN----WRLSFQEQEGERELLHNYKNMKSSIRLIKK
+++D C KT+F +P D D+ N WRLSF QE+E+L Y +K IR +KK
IHVDLAPCR-NKTWFAIPLPTDDDDINNKNFYWRLSFSCQEKEILTQGRIKPVIRQMKK
```
Gap Scores Revisited

Example (beetle - bee)
section from an alignment between a protein from the red flour beetle and from honey bee:

VNIDGTNCPIKKTFVVPKSPNDLDVYN----WRLSFQEEREHHNYKNMKSSIRLIKK
+++D C KT+F +P D D+ N WRLSF QE+E+L Y +K IR +KK
IHVDLAPCR--NKTWFAPLPLPTDDINNKNFYWRLSFSCQEKEILTKYGRIRKPVIRQMKK

Example (chicken - opossum)
another alignment between chicken and opossum proteins:

FSPTKVADWFYDSISIVLSEIQKKPQRGMPKVEKVEKNGTIISIIL---GVGSSRMRLYDI
LCPTQVASWFSSLSSAVAVSPTTP---GAPRLVHAARHAGVTTILLSTPGPPPRLLFLDL

VPVVSFKGWPAVAQSWMENHFWDGKITEEEVISGFYLVPACSYKGKGDNEWRLSFARSE
VPGISVTGWPERTRT-----HAWMGPLAAGP---TSYYLVPFGAKEYDGLCWRICFARQE
**Gap lengths**

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
</table>
| **Compare** | AATCGGACTGGTAA  
|       | AA--------TGGCAA  
| **with** | AATCGGACTGGTAA  
|       | AAT-GG-C-----AA |

Under a typical linear scoring scheme, the right alignment scores better. But it’s less plausible: 3 indel events instead of 1 longer indel and 1 mutation.

**Observations/Conclusions**

- indels are not distributed homogenously over alignment
Gap lengths

Example

<table>
<thead>
<tr>
<th>Compare</th>
<th>AATCGGACTGGTAA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AA-------TGGCAA</td>
</tr>
</tbody>
</table>

with

<table>
<thead>
<tr>
<th>AATCGGACTGGTAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAT-GG-C------AA</td>
</tr>
</tbody>
</table>

Under a typical linear scoring scheme, the right alignment scores better. But it’s less plausible: 3 indel events instead of 1 longer indel and 1 mutation.

Observations/Conclusions

- indels are not distributed homogenously over alignment
- indel columns tend to cluster
Gap lengths

Example

<table>
<thead>
<tr>
<th>Compare</th>
<th>AATCGGACTGGTAA</th>
<th>with</th>
<th>AATCGGACTGGTAA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AA----------TGGCAA</td>
<td></td>
<td>AAT-GG-C------AA</td>
</tr>
</tbody>
</table>

Under a typical linear scoring scheme, the right alignment scores better. But it’s less plausible: 3 indel events instead of 1 longer indel and 1 mutation.

Observations/Conclusions

- indels are not distributed homogenously over alignment
- indel columns tend to cluster
- whether there is an insertion (deletion) at alignment position $i$ is not independent from the event whether we observe one at, say, position $i + 1$
### Gap lengths

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compare</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>with</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Under a typical linear scoring scheme, the right alignment scores better. But it's less plausible: 3 indel events instead of 1 longer indel and 1 mutation.

### Observations/Conclusions

- indels are not distributed homogenously over alignment
- indel columns tend to cluster
- whether there is an insertion (deletion) at alignment position $i$ is not independent from the event whether we observe one at, say, position $i + 1$
- neighboring indels could reflect an insertion or deletion event during evolution of more than 1 amino acid *in one step*
Scoring a gap

Gap penalty function

- let $p(n) < 0$ be the score associated with a gap of length $n$
Scoring a gap

**Gap penalty function**

- let $p(n) < 0$ be the score associated with a gap of length $n$
- $-p(n)$ is also called a **penalty**
Scoring a gap

Gap penalty function

- let $p(n) < 0$ be the score associated with a gap of length $n$
- $-p(n)$ is also called a penalty
- a simple scoring scheme as introduced above scores an indel position independently on the length of the complete gap.

It has linear gap penalties if, as usual, $s(a, -) = s(-, a)$ is independent of $a$:

$$p(n) = c \cdot n$$
Scoring a gap

<table>
<thead>
<tr>
<th>Gap penalty function</th>
</tr>
</thead>
<tbody>
<tr>
<td>let $p(n) &lt; 0$ be the score associated with a gap of length $n$</td>
</tr>
<tr>
<td>$-p(n)$ is also called a penalty</td>
</tr>
<tr>
<td>a simple scoring scheme as introduced above scores an indel position independently on the length of the complete gap. It has <strong>linear gap penalties</strong> if, as usual, $s(a, -) = s(-, a)$ is independent of $a$: $p(n) = c \cdot n$</td>
</tr>
</tbody>
</table>

linear gap penalties are not realistic and may lead to too many (short) gaps
Scoring a gap

Gap penalty function

- let $p(n) < 0$ be the score associated with a gap of length $n$
- $-p(n)$ is also called a penalty
- a simple scoring scheme as introduced above scores an indel position independently on the length of the complete gap.
- It has linear gap penalties if, as usual, $s(a, -) = s(-, a)$ is independent of $a$:

$$p(n) = c \cdot n$$

- linear gap penalties are not realistic and may lead to too many (short) gaps
- good/true alignments should have rather fewer longer gaps than many short gaps
Affine gap penalties

A scoring scheme in which the costs of a gap of length $n$ is of the form

$$p(n) = -g - e \cdot n$$

with constants $g, e \geq 0$ is said to have **affine gap penalties**.

- $g$ can be interpreted as the penalty for creating a gap at all and is called **gap opening penalty**
- $e$ can be interpreted as the penalty for any gap position and is called **gap extension penalty**
**Affine gap penalties**

- **affine gap penalties** usually model the evolution of gaps more **realistically** than linear gap penalties.
Affine gap penalties

- **affine gap penalties** usually model the evolution of gaps more realistically than linear gap penalties
- algorithms that find a best alignment in a scoring scheme with affine gap penalty **require more time** than algorithms under a linear gap model
Affine gap penalties

- **affine gap penalties** usually model the evolution of gaps more realistically than linear gap penalties
- algorithms that find a best alignment in a scoring scheme with affine gap penalty require more time than algorithms under a linear gap model
- with a suitable DP algorithm, a best-scoring alignment under an affine gap penalties model can be found in time $O(nm)$ (same complexity as linear gap penalties)
Affine gap penalties

- **affine gap penalties** usually model the evolution of gaps more realistically than linear gap penalties.
- algorithms that find a best alignment in a scoring scheme with affine gap penalty **require more time** than algorithms under a linear gap model.
- with a suitable DP algorithm, a best-scoring alignment under an affine gap penalties model can be found in time $O(nm)$ (same complexity as linear gap penalties).
- **affine gap penalties** constitute a good **compromise** between speed and accuracy and are most often used in practice.
Consider gain the **global pairwise alignment problem** for two strings \( S \) and \( T \) over alphabet \( \Sigma \) and a given scoring matrix 
\[
s : \Sigma \times \Sigma \to \mathbb{R}
\]. Each gap of length \( d \) should be scored using gap opening penalty \( g \) and gap extension penalty \( e \).
Consider gain the global pairwise alignment problem for two strings $S$ and $T$ over alphabet $\Sigma$ and a given scoring matrix $s : \Sigma \times \Sigma \rightarrow \mathbb{R}$. Each gap of length $d$ should be scored using gap opening penalty $g$ and gap extension penalty $e$.

**Idea**

- Here: have different DP variables for solutions to subproblems depending on whether a gap in $S$ or $T$ has already been opened (and therefore only needs to be extended)
- In general: “Remember” properties of solution to subproblem by defining variants of DP variables.
**Definition (DP variables)**

Let

\[ G(i, j) := \text{maximum score of any alignment of } S[1..i] \text{ and } T[1..j] \text{ that aligns } S[i] \text{ with } T[j] \]

\[ E(i, j) := \text{maximum score of any alignment of } S[1..i] \text{ and } T[1..j] \text{ that ends with a gap in } S \]

\[ F(i, j) := \text{maximum score of any alignment of } S[1..i] \text{ and } T[1..j] \text{ that ends with a gap in } T \]

\[ V(i, j) := \text{maximum score of any alignment of } S[1..i] \text{ and } T[1..j] \]
DP recursion for affine gap penalties

DP recursion

\[
\begin{align*}
V(i, j) &= \max\{G(i, j), E(i, j), F(i, j)\} \\
G(i, j) &= V(i - 1, j - 1) + s(S[i], T[j]) \\
E(i, j) &= \max\{E(i, j - 1), V(i, j - 1) - g\} - e \\
F(i, j) &= \max\{F(i - 1, j), V(i - 1, j) - g\} - e \\
\end{align*}
\]

boundary cases:

\[
\begin{align*}
V(i, 0) &= E(i, 0) = -g - ei & \text{for } 1 \leq i \leq |S| \\
V(0, j) &= F(0, j) = -g - ej & \text{for } 1 \leq j \leq |T| \\
\end{align*}
\]
1 Alignments

Pairwise Alignments
- Dot plot
- Pairwise alignment
- Scoring matrices
- Motivation of log-odds
- Affine gap scores

Multiple Alignments
- Motivation and definitions
- Local multiple alignments
- Scoring multiple alignments
- Brute force multiple alignment
- Progressive alignment
Global multiple alignment

Definition (global multiple alignment)

A global (multiple) alignment of \( k \) strings \( S = \{S_1, S_2, \ldots, S_k\} \) is an array \( (S'_1, S'_2, \ldots, S'_k) \) of strings possibly including gap characters, such that

- all \( S'_i \) have the same length \( \ell \)
- removing all gap characters from \( S'_i \) yields \( S_i \) for all \( i \)
- \( (S_1[j], S_2[j], \ldots, S_k[j]) \neq (\_, \_, \ldots, \_) \) for all \( 1 \leq j \leq \ell \)

Example

\[
-\text{NCDKLGWY-INS-ZQLP} \\
\text{SHCDQ-GWY-LG--EQLZ} \\
-\text{YCDKIGWY-IA--EQLP} \\
\text{SHCDELGWYEIGLGKQLP}
\]
Multiple versus pairwise alignment in “twilight zone”

“...two homologous sequences wisper ... a full multiple alignment shouts out lout” (Arthur Lesk, 1996)
Multiple versus pairwise alignment in “twilight zone”

excerpt from a multiple alignment

A88755_GIALA/381-716
FRDMIPETDSGD..KREFKLINIAUSPIFSGPAYS.D..IKTGLLLLOCFGGVQ.K......D...YRGQIHVLLVGP.D.
Q6S5E5_ENCCU/304-622
PIKIDKDEL.RKRP..DVYEILANSVAPSVCC..MED..TKKALVLQLFQGVVR.K..ELGS..SR..LREGIDINILAGDPG.
MCM4_DROME/447-772
RVELLQLL.AKKP..DIYDRLARAIAPSIYE..NDD..IKKGIQLQLFQGGT.K..KHAT.LERQNNRSERIHLLCGDPG.
Q95XQ8_CAEEL/404-730
RIQQIELL.SKR.R..DIMIYLAQSIAPSIYE..HDD..VKRGLLQLLFQGTR.KDDETTN.HT..K..LREGIDINILCGDPG.
Q661F1_DICDI/469-792
KEREIIEL.SKKP..DIYDIVKHSIAPIWIE..LED..IKKGIQLQCGGSK.K..SYQDYGK..FRGDINILCGDPG.
Q75SH5_NEUCR/580-912
EEERIKET.AARP..DIYDILLRSLSAPSIYE..MDD..VKKGILLQLFQGGTN.K..TTFK.GGR..YRGGDINILCGDPG.
CDC54_YEAST/505-834
DLAKIRE.VAARE..DIYSLRSLARSIAPSIYE..LED..VKKGILLQLFQGGTN.K..TTFK.GGR..YRGGDINILCGDPG.
MCM4_SCHPO/480-811
EVEIKIOQ.VSKRD..DIYDILSLRSLAPSIYE..MDD..VKGGILLLQLFQGGTN.K..SFHK..GASP.YRGGDINILCGDPG.
MCM3_ENTHI/167-531
DKEINIKK.VMKEE..NPINIFSKISAPSIYE..HSD..VKKAILLMLQVGGTP.K..IRLR..SR..VRRGDINILCGDPG.
A8BJJ1_GIALA/302-626
LATDLSVL.REKA..IPLHVLAISAPRHIHG..RED..KLAICLSCLVGGNE.I..VVPD..MK..VRGNINTFVGGGP.
Q9U1E0_LEIMA/317-646
QVEDIRD.Y.PDRE..AVIIEKLTSIAPEIWG..MED..VKKALCQLVGGSS...IAN.GIR..IRSDINILFMGD.
Q7R7J3_PLAYO/402-752
IMEEVKQL.KSSP..NLVRLAYNIAGPIYE..HED..VKKALQLQLQGGCT.K..KKKD.GGL..IRGDTHILMGDPG.
PROL_ARATH/313-638
EEEOS1AR.AEKG..DIYNdLRSLSAPEIYG..HED..IKKALLLLVLQGAPH.R..QLKD.GM..IRGHDVHLMGDPG.
CDC47_YEAST/397-725
VEERWME.LITG..DVYNLKASIAPSIYE..NLD..VKKALLLLVLQGVD.K..RVGD.GM..IRGDINVCLMGDPG.
Q7OS6I_ANOGA/318-640
TQEEVLDEL.AKD..DFYTRIASLAPSIYE..HDC..VKKALLLLLVLQGVD..RSPD.GM..IRGIDINLMDGD.
O16297_ENTHI/328-652
QSAEELV.RRKG..DNVLAYNIAPEIHF.G..HVD..VKKALLLLLVLQGDN..N..S..N.GM..IRGNCVLMGDPG.
Q7YY63_CRYPV/268-626
SRFRKFLEI.SQHP..NGIINLAKYVAPHVYG..YSQ..LKKGILLQLVGGVE.K..RTKD.NIK..LRGIDINCVGGPG.
Q8IFD0_PLAF7/423-751
DLKWLDI.ANHP..NTIDLAEACIPKQW..NLE..IKKAGLMNTGQ.GV.K..ITSN..CK..IRGIDINCVGGPG.
A9BLX7_GIALA/424-793
YRRAVSNM.RNDP..NIVSNLVSASPABHIYG..HET..VLGILLOLLIGGK.K..ITRS.EHL.SIRSDINILLGDPG.
MCM6_CAEEL/333-660
DRAKLLKK.SDKK..KEIKNIVDSLPNIY.Y.GNE..VKLGVLVLLMVGQAVK..SRDE.GTS..LRGIDINCVGGPG.
MCM6_DROME/325-649
EWHKIDYEM.SKDR..NLYQNNLSSLFFPSIY.G..NDE..IKKGILLOQFGGVA.K..TTTE.KTS..LRGIDINCVGGPG.
MCM6_RAT/19-343
EWEKVFEM.SQDQ..NLYHNLCTSSLPFSIY.G..NDE..IKKGILLQLFQGGVP.K..TTGE.GTS..LRGIDINCVGGPG.
MCM6_DICDI/407-731
EKKDLKKM.IKSK..KIQYQNVLNSPSIFG..HEE..IKKGVLVLLMGGVEH.K..KTPE.KIR..LRGIDINCVGGPG.
MCM6_YEAST/512-836
EINELKEM.VKDE..HVDLYLQPSAPAVF.G..HEA..IKKGILLOQMGGGH..STVE.GIK..LRGIDINCVGGPG.
MCM6_SCHPO/413-737
EIDDLRAML.VHSD..HIYSLTNSLSAPSYY.G..HEI..IKKGILLOQMGGHH.K..LTE.GIN..LRGIDINCVGGPG.
Q8H1A3_PEA/330-654
ELDEVQRM.RNTP..DFDFTKLVESVAPTIFG..HOD..IKRAILMLMSGHH..STHE.GIS..LRGIDINCVGGPG.
## Pairwise versus multiple alignment

<table>
<thead>
<tr>
<th><strong>pairwise alignment</strong></th>
<th><strong>multiple alignment</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>two sequences are aligned</td>
<td>more than two sequences are aligned</td>
</tr>
<tr>
<td>degree of sequence similarity is measured</td>
<td>can also work with relatively low sequence similarity</td>
</tr>
<tr>
<td>requires higher similarity for statistical significance</td>
<td>often done with sequences that are known to be homologous</td>
</tr>
<tr>
<td>often done to decide whether they are homologous</td>
<td>to predict which residues share common ancestry</td>
</tr>
<tr>
<td>used as a tool for database search</td>
<td>to detect residues that are particularly conserved through evolution</td>
</tr>
<tr>
<td></td>
<td>to infer which residues are important for function</td>
</tr>
</tbody>
</table>
A *local* (multiple) alignment of *k* strings \( \{S_1, S_2, \ldots, S_k\} \) is a global alignment of substrings \( \{s_1, s_2, \ldots, s_k\} \) \( (s_i \text{ substring of } S_i) \).
A local (multiple) alignment of \( k \) strings \( \{S_1, S_2, \ldots, S_k\} \) is a global alignment of substrings \( \{s_1, s_2, \ldots, s_k\} \) (\( s_i \) substring of \( S_i \)).

**Example (protein domain Hexokinase_2 (PF03727))**

**Pfam-seed alignment for PF03727**

<table>
<thead>
<tr>
<th>Accession</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>P33284/224-472</td>
<td>QTKMGIIGTGVNGAYDDVSGIIEKLEGLLPP...EDIGPDSPAINECEYSF.DNEHLVLPRKDYVIDEEO.SPPPGQQAQEKMNTSGYGLGEIMRLVLD</td>
</tr>
<tr>
<td>P50521/213-449</td>
<td>DTFMIGITGVTNGAYVEAQINQIPKLACK....CTGDMHLEINMGAT.DFSCLHSTR.YDLLLLHD.TPNAQRQIFKTVGGMYLGELFRAFLFH</td>
</tr>
<tr>
<td>P55557/219-458</td>
<td>QCEGIVYGTVGCNAYMEEMQNLVEVQ......DEGRMCVNTGWAGFDGQSGDELDEFLEYDRVDES.SANPGQQLYKJGKYMELYELVRLLR</td>
</tr>
<tr>
<td>P52789/223-462</td>
<td>NCEIGLVGTGNSACNYEEMRHDVMEV......DEGRMCINMEGWAGFDDGSLNNDETEFDQIEMG.SLNPQKQLEFKEISGMYGLELRLLVK</td>
</tr>
<tr>
<td>P27595/223-462</td>
<td>HCEVGLIVGTGNSACNYEELRDIFDWG......DEGRMCINTEGWQGDGSLLEDIEKEFDFREFRG.SLNPQKQRFKEIMSGYGLYELVRLVL</td>
</tr>
<tr>
<td>P17709/243-500</td>
<td>EPVIGCIDTGTNGCBYMEINKITKLPQELRDKLIEKTKHNIINIEGQSF.DNEKHLHPETKYVDVIDQKLSTNPQHLFEKEKSGMVFLGELVRLNLVD</td>
</tr>
<tr>
<td>P50906/224-469</td>
<td>EAKMLFSGTGNCAYDYVDNIPKLEKKV......DDIKSSSPAINCEYQAF.DNEHHIVLPRTKYDQIDEE.SPPPGQQAQEKMISGYYEGLVELRLVLD</td>
</tr>
<tr>
<td>P54807/225-473</td>
<td>ETKMGVI-TGTNGAYVDDVCEDIEKLQGKL......DEIPSPAINCEYQAF.DNEHHIVLPRTKYDQIDEE.SPPPGQQAQEKMISGYYEGLVELRLVLD</td>
</tr>
<tr>
<td>P27595/671-910</td>
<td>TCEGILVGTGNSACNYEEMKHEMVEG......NQRMQMCMNMGAGFDNGCSDDIRDFDKYVDEY.SLNSGQRNFENISGMYGLGEVRNLID</td>
</tr>
<tr>
<td>Q09756/228-469</td>
<td>GTEGVI-TGTGCNAYIEKFSIEIKLH.KY......DPFEDNNIIINCEYWSF.DNEHHIVLPRTKYDQIDEE.SPPPGQQAQEKMISGYYEGLVELRLVLD</td>
</tr>
<tr>
<td>P54806/225-472</td>
<td>ETKMVIFGTVNGAYDFVSQDIEKEKL.VEK......DIDPSNPPAINCEYQAF.DNEHHIVLPRTKYDQIDEE.SPPPGQQAQEKMISGYYEGLVELRLLLE</td>
</tr>
<tr>
<td>P52789/236-475</td>
<td>PCEGVLIVGTGNSACNYEEMVRHVAVLDE......DGRRCVSVEWYFSDGQLPVLTTFDHTLHDE.SLNGAQDFKEIKGLYGLYELVRLVLD</td>
</tr>
<tr>
<td>P52789/677-916</td>
<td>RCEGILVGTGNCYMEELRNVAGV...DSSGDNQMEGAFDQGDSLOGAMSLRFDASVQA.SINPCKQRFKEIMSGYGLGEVRHILL</td>
</tr>
<tr>
<td>Q02155/259-485</td>
<td>PCYIGIELMTGSNCYCYEPEKYYKYAG......KIINIEECDK...DILPSPIDLVMNK.SANRSQOLFEKSMGAYGLGEVRNFMQN</td>
</tr>
<tr>
<td>P52789/671-910</td>
<td>HCEVGLIVGTGNSACNYEELRNVAVLVEG......EEFRGVMDEGAFDNRCDDFTFEDQFVAVDEL.SLNGAQDFKEIKGLYGLYELVRLVLD</td>
</tr>
</tbody>
</table>
Local multiple alignment

**Definition (local multiple alignment)**

A *local* (multiple) alignment of $k$ strings \( \{S_1, S_2, \ldots, S_k\} \) is a global alignment of substrings \( \{s_1, s_2, \ldots, s_k\} \) \( (s_i \text{ substring of } S_i) \).

**Example (protein domain Hexokinase_2 (PF03727))**

**Pfam-seed alignment for PF03727**

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
</tbody>
</table>

---

**Table Continued**

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
</tbody>
</table>

---

**Table Continued**

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
</tbody>
</table>

---

**Table Continued**

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
<tr>
<td>123456789</td>
<td>123456789</td>
</tr>
</tbody>
</table>
Example (protein domain Hexokinase_2 (PF03727))

Domain organisation

Below is a listing of the unique domain organisations or architectures in which this domain is found. More...

There are 576 sequences with the following architecture: Hexokinase_1, Hexokinase_2
HXK1_ARATH [Arabidopsis thaliana (Mouse-ear cress)] Hexokinase_1 EC=2.7.1.1 (496 residues)
Show all sequences with this architecture.

There are 55 sequences with the following architecture: Hexokinase_1, Hexokinase_2, Hexokinase_1, Hexokinase_2
HXK1_BOVIN [Bos taurus (Bovine)] Hexokinase_1 EC=2.7.1.1 (918 residues)
Show all sequences with this architecture.

There are 37 sequences with the following architecture: Hexokinase_2
Q0CPV0_ASPTN [Aspergillus terreus (strain NIH 2624 / FGSC A1156)] Putative uncharacterized protein (339 residues)
Show all sequences with this architecture.

There are 16 sequences with the following architecture: Hexokinase_1, Hexokinase_2 x 2
B2W6G6_PYRTR [Pyrenophora tritici-repentis (strain Pt-1C-BFP) (Wheat tan spot fungus) (Drechslera tritici-repentis)] Glucokinase GLK1 (632 residues)
Show all sequences with this architecture.

There are 5 sequences with the following architecture: Hexokinase_1 x 2, Hexokinase_2
Q74996_YARLI [Yarrowia lipolytica (Candida lipolytica)] Hexokinase (534 residues)
Show all sequences with this architecture.

There are 4 sequences with the following architecture: Hexokinase_2, Hexokinase_1, Hexokinase_2
Q53QX9_HUMAN [Homo sapiens (Human)] Putative uncharacterized protein HK2 (Fragment) (573 residues)
Show all sequences with this architecture.

There are 2 sequences with the following architecture: Hexokinase_1, Hexokinase_2, Hexokinase_1
Induced pairwise alignment

Definition (induced pairwise alignment)

Given a multiple alignment $\mathcal{M}$, the induced pairwise alignment of two strings $S_i$ and $S_j$ is obtained from $\mathcal{M}$ by removing all rows except the two rows for $S_i$ and $S_j$ and removing all columns that contain only gaps.
### Induced pairwise alignment

**Definition (induced pairwise alignment)**

Given a multiple alignment $M$, the induced pairwise alignment of two strings $S_i$ and $S_j$ is obtained from $M$ by removing all rows except the two rows for $S_i$ and $S_j$ and removing all columns that contain only gaps.

**Example**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>$-$NCDKLGWY-INS-ZQLP</td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td>SHCDQ-GWY-LG--EQLZ</td>
<td></td>
</tr>
<tr>
<td>$S_3$</td>
<td>$-$YCDKIGWY-IA--EQLP</td>
<td></td>
</tr>
<tr>
<td>$S_4$</td>
<td>SHCDELGWYEIGLGBKQLP</td>
<td></td>
</tr>
</tbody>
</table>

induces the pairwise alignment on $S_1$ and $S_3$. 
How to score a multiple alignment?

**Definition (sum-of-pairs score)**

The **sum-of-pairs** score of a multiple alignment $\mathcal{M}$ is the sum of the scores of all pairwise alignments induced by $\mathcal{M}$.

**Example**

*(chalk board)*
How to score a multiple alignment?

**Definition (sum-of-pairs score)**

The *sum-of-pairs* score of a multiple alignment $\mathcal{M}$ is the sum of the scores of all pairwise alignments induced by $\mathcal{M}$.

**Example**

*(chalk board)*

**Other scoring methods for multiple alignments**

- weighted sum-of-pairs score $\sum_{i<j} w(i, j)d(S_i, S_j)$ with $d(S_i, S_j)$ being the score of the induced pairwise alignment of $S_i$ and $S_j$ and $w(i, j)$ being a weight reflecting evolutionary distance of $S_i$ and $S_j$
- scoring along a phylogenetic tree
- (Steiner) consensus string
Brute Force DP-Algorithm for Multiple Alignment

Generalize from pairwise alignments

- Consider a global multiple alignment problem of $r=3$ sequences $S_1, S_2, \ldots, S_r$ using the sum-of-pairs score.
Brute Force DP-Algorithm for Multiple Alignment

Generalize from pairwise alignments

- Consider a global multiple alignment problem of \( r=3 \) sequences \( S_1, S_2, \ldots, S_r \) using the sum-of-pairs score.
- Let \( s(a, b, c) = s(a, b) + s(a, c) + s(b, c) \) be the score of an alignment column in which \( a, b, c \in \Sigma \cup \{-\} \) are aligned with each other.
Generalize from pairwise alignments

- Consider a global multiple alignment problem of $r=3$ sequences $S_1, S_2, \ldots, S_r$ using the sum-of-pairs score.
- Let $s(a, b, c) = s(a, b) + s(a, c) + s(b, c)$ be the score of an alignment column in which $a, b, c \in \Sigma \cup \{-\}$ are aligned with each other.
- The score of a multiple alignment $A = (S'_1, S'_2, S'_3)$ of length $\ell$ of three sequences $S_1, S_2$ and $S_3$ is then

$$s(A) = \sum_{i=1}^{\ell} s(S'_1[i], S'_2[i], S'_3[i]).$$
Generalize from pairwise alignments

- Consider a global multiple alignment problem of \( r=3 \) sequences \( S_1, S_2, \ldots, S_r \) using the sum-of-pairs score.
- Let \( s(a, b, c) = s(a, b) + s(a, c) + s(b, c) \) be the score of an alignment column in which \( a, b, c \in \Sigma \cup \{-\} \) are aligned with each other.
- The score of a multiple alignment \( A = (S'_1, S'_2, S'_3) \) of length \( \ell \) of three sequences \( S_1, S_2 \) and \( S_3 \) is then

\[
s(A) = \sum_{i=1}^{\ell} s(S'_1[i], S'_2[i], S'_3[i]).
\]

- We want to find a multiple alignment with maximal score.
Brute Force DP-Algorithm for Multiple Alignment

Generalize from pairwise alignments

Idea: Same, same, but different.
Brute Force DP-Algorithm for Multiple Alignment

**Generalize from pairwise alignments**

Idea: *Same, same, but different.*

Let

**Definition (DP variables)**

Let

\[
M(i, j, k) := \text{maximal score of an alignment of } S_1[1..i], S_2[1..j] \text{ and } S_3[1..k]
\]  

(13)

for \( i = 1, \ldots, |S_1|, j = 1, \ldots, |S_2|, k = 1, \ldots, |S_3| \).
Brute Force DP-Algorithm for Multiple Alignment

DP recursion for 3 sequences

\[
M(i, j, k) = \max \begin{cases} 
M(i - 1, j - 1, k - 1) + s(S_1[i], S_2[j], S_3[k]) \\
M(i - 1, j - 1, k) + s(S_1[i], S_2[j], -) \\
M(i - 1, j, k - 1) + s(S_1[i], -, S_3[k]) \\
M(i, j - 1, k - 1) + s(-, S_2[j], S_3[k]) \\
M(i - 1, j, k) + s(S_1[i], -, -) \\
M(i, j, k - 1) + s(-, -, S_3[k]) \\
M(i, j - 1, k) + s(-, -, S_2[j]) 
\end{cases}
\]

for \( i = 1, \ldots, |S_1|, j = 1, \ldots, |S_2|, k = 1, \ldots, |S_3| \).
**Brute Force DP-Algorithm for Multiple Alignment**

**DP recursion for 3 sequences**

\[
M(i, j, k) = \max \begin{cases} 
M(i - 1, j - 1, k - 1) + s(S_1[i], S_2[j], S_3[k]) \\
M(i - 1, j - 1, k) + s(S_1[i], S_2[j], -) \\
M(i - 1, j, k - 1) + s(S_1[i], -, S_3[k]) \\
M(i, j - 1, k - 1) + s(-, S_2[j], S_3[k]) \\
M(i - 1, j, k) + s(S_1[i], -, -) \\
M(i, j, k - 1) + s(-, -, S_3[k]) \\
M(i, j - 1, k) + s(-, S_2[j], -) 
\end{cases}
\]

for \(i = 1, \ldots, |S_1|, j = 1, \ldots, |S_2|, k = 1, \ldots, |S_3|\).

The 7 arguments of “max” correspond to the 7 possibilities for the gap characters in the last column of an optimal alignment of \(S_1[1..i], S_2[1..j]\) and \(S_3[1..k]\).
Brute Force DP-Algorithm for Multiple Alignment

DP recursion for 3 sequences

\[
M(i, j, k) = \max \begin{cases} 
M(i-1, j-1, k-1) + s(S_1[i], S_2[j], S_3[k]) \\
M(i-1, j-1, k) + s(S_1[i], S_2[j], -) \\
M(i-1, j, k-1) + s(S_1[i], -, S_3[k]) \\
M(i, j-1, k-1) + s(-, S_2[j], S_3[k]) \\
M(i-1, j, k) + s(S_1[i], -, -) \\
M(i, j, k-1) + s(-, -, S_3[k]) \\
M(i, j-1, k) + s(-, -, -) 
\end{cases}
\]

for \(i = 1, \ldots, |S_1|, j = 1, \ldots, |S_2|, k = 1, \ldots, |S_3|\).

The 7 arguments of “max” correspond to the 7 possibilities for the gap characters in the last column of an optimal alignment of \(S_1[1..i], S_2[1..j]\) and \(S_3[1..k]\).

Boundary cases for \(i = 0, j = 0\) or \(k = 0\) can also be found (exercise).
Brute Force DP-Algorithm for Multiple Alignment

Algorithm sketch

1. compute $M$ for boundary cases ($i = 0, j = 0$ or $k = 0$)  
2. compute $M(i, j, k)$ for increasing $i, j, k$ using recursion  
3. $M(|S_1|, |S_2|, |S_3|)$ is the score of the best alignment  
4. backtrack
Brute Force DP-Algorithm for Multiple Alignment

Algorithm sketch

1. compute $M$ for boundary cases ($i = 0, j = 0$ or $k = 0$)
2. compute $M(i, j, k)$ for increasing $i, j, k$ using recursion
3. $M(|S_1|, |S_2|, |S_3|)$ is the score of the best alignment
4. backtrack

Time and space requirements

- running time: $O(|S_1| \cdot |S_2| \cdot |S_3|)$, “cubic” running time
Brute Force DP-Algorithm for Multiple Alignment

Algorithm sketch

1. compute \( M \) for boundary cases \((i = 0, j = 0 \text{ or } k = 0)\)
2. compute \( M(i, j, k) \) for increasing \( i, j, k \) using recursion
3. \( M(|S_1|, |S_2|, |S_3|) \) is the score of the best alignment
4. backtrack

Time and space requirements

- running time: \( O(|S_1| \cdot |S_2| \cdot |S_3|) \), “cubic” running time
- memory: \( O(|S_1| \cdot |S_2| \cdot |S_3|) \)
Brute Force DP-Algorithm for Multiple Alignment

Generalization for $r > 3$ sequences

Let $n_1, n_2, \ldots, n_r$ be the length of $r$ sequences that are to be aligned.

Further generalization like above possible.
Brute Force DP-Algorithm for Multiple Alignment

**Generalization for $r > 3$ sequences**

Let $n_1, n_2, \ldots, n_r$ be the length of $r$ sequences that are to be aligned.

Further generalization like above possible.

- Required running time and space with generalization:
  \[ O(|n_1| \cdot |n_2| \cdots \cdot |n_r|) \]
Brute Force DP-Algorithm for Multiple Alignment

Generalization for \( r > 3 \) sequences

Let \( n_1, n_2, \ldots, n_r \) be the length of \( r \) sequences that are to be aligned.

Further generalization like above possible.

- Required running time and space with generalization:
  \( O(|n_1| \cdot |n_2| \cdot \cdots \cdot |n_r|) \)

- Not feasible with typical problem sizes:
  \( n_i, r \) a few hundreds each
Progressive Alignment Heuristic

Idea

- find a multiple alignment using a stepwise heuristic
Progressive Alignment Heuristic

Idea

- find a multiple alignment using a stepwise heuristic
- do at most $r$ steps, each step relatively simple (polynomial time)
Progressive Alignment Heuristic

Idea

- find a multiple alignment using a stepwise heuristic
- do at most $r$ steps, each step relatively simple (polynomial time)
- from small to big: start by aligning only some sequences to each other
Progressive Alignment Heuristic

Idea

- find a multiple alignment using a **stepwise** heuristic
- do at most \( r \) steps, each step relatively simple (polynomial time)
- from small to big:
  start by aligning only some sequences to each other
- progressively add more sequences to alignments
Progressive Alignment Heuristic

Idea

- find a multiple alignment using a stepwise heuristic
- do at most $r$ steps, each step relatively simple (polynomial time)
- from small to big:
  - start by aligning only some sequences to each other
  - progressively add more sequences to alignments
  - until all sequences are aligned
Sequence Profile

Definition (Profile)

Let $A = (S'_1, \ldots, S'_k)$ be a multiple alignment of strings $S_1, \ldots, S_k$ over an alphabet $\Sigma$ of length $\ell$.

A profile (also sequence profile, weight matrix) for $A$ specifies for each column $i$ of $A$ the frequency of the characters in that column (a probability distribution over $\Sigma \cup \{-\}$):

$$p(y, i) \quad (1 \leq i \leq \ell, y \in \Sigma \cup \{-\})$$
Sequence Profile

Definition (Profile)

Let $A = (S'_1, \ldots, S'_k)$ be a multiple alignment of strings $S_1, \ldots, S_k$ over an alphabet $\Sigma$ of length $\ell$.

A profile (also sequence profile, weight matrix) for $A$ specifies for each column $i$ of $A$ the frequency of the characters in that column (a probability distribution over $\Sigma \cup \{-\}$):

$$p(y, i) \quad (1 \leq i \leq \ell, y \in \Sigma \cup \{-\})$$

Example (profile)

A : 
\[
\begin{array}{cccccc}
T & A & C & - & T \\
T & - & G & C & T \\
T & G & C & C & C \\
C & T & - & - & T \\
T & A & C & C & T \\
\end{array}
\]

\[
\begin{array}{c|ccccc}
A & 0 & .4 & 0 & 0 & 0 \\
C & .2 & 0 & .6 & .6 & .2 \\
G & 0 & .2 & .2 & 0 & 0 \\
T & .8 & .2 & 0 & 0 & .8 \\
- & 0 & .2 & .2 & .4 & 0 \\
\end{array}
\]
Aligning a Sequence against a Profile

Definition
Score for aligning a character to a profile position Let \( s(x, y) \) denote the score for aligning two characters (or gaps) \( x \) and \( y \). The score for aligning a character \( x \) to position \( j \) of profile \( p \) is defined as

\[
S(x, j) := \sum_{y \in \Sigma \cup \{-\}} s(x, y)p(y, j).
\]
Aligning a Sequence against a Profile

**Definition**

Score for aligning a character to a profile position Let $s(x, y)$ denote the score for aligning two characters (or gaps) $x$ and $y$. The score for aligning a character $x$ to position $j$ of profile $p$ is defined as

$$S(x, j) := \sum_{y \in \Sigma \cup \{-\}} s(x, y)p(y, j).$$

- $S(x, j)$ can be interpreted as the average score of $x$ with a random character of the $j$th profile position.
- The score of a sequence-to-profile alignment is the sum of the scores of the columns.
Aligning a Sequence against a Profile

Example (Sequence-Profile Alignment)

Consider the scoring scheme

\[
s(x, y) = \begin{cases} 
2 & \text{, if } x = y \neq - \\
-2 & \text{, if } (x = - \text{ and } y \neq -) \text{ or } (y = - \text{ and } x \neq -) \\
0 & \text{, if } x = y = - \\
-1 & \text{, otherwise.}
\end{cases}
\]

We set \( s(-, -) = 0 \) in order to generalize for profiles.
Aligning a Sequence against a Profile

Example (Sequence-Profile Alignment)

Consider the scoring scheme

\[
s(x, y) = \begin{cases} 
2 & \text{if } x = y \neq - \\
-2 & \text{if } (x = - \text{ and } y \neq -) \text{ or } (y = - \text{ and } x \neq -) \\
0 & \text{if } x = y = - \\
-1 & \text{otherwise.}
\end{cases}
\]

We set \( s(-, -) = 0 \) in order to generalize for profiles.

An alignment between the sequence \( T = \text{ATACT} \) and profile \( p \):

\[
\begin{array}{ccccccc}
 p & - & 1 & 2 & 3 & 4 & 5 \\
 S & A & T & A & - & C & T \\
\hline 
\text{score} & -1 & 1.4 & 0 & -0.8 & 0.8 & 1.4 \\
\end{array}
\]

This alignment has total score 1.8.
Aligning a Sequence against a Profile

DP variables

Let

\[ V(i, j) := \text{value of an optimal alignment of substring } T[1..i] \text{ with the first } j \text{ columns of } p \]
Aligning a Sequence against a Profile

**DP variables**
Let
\[
V(i, j) := \text{value of an optimal alignment of substring } T[1..i] \text{ with the first } j \text{ columns of } p
\]

**DP recursion**
\[
V(i, j) = \max \begin{cases} 
V(i-1, j-1) + S(T[i], j) \\
V(i-1, j) + s(T[i], -) \\
V(i, j-1) + S(-, j)
\end{cases}
\]
Aligning a Sequence against a Profile

DP variables

Let

\[ V(i, j) := \text{value of an optimal alignment of substring } T[1..i] \text{ with the first } j \text{ columns of } p \]

DP recursion

\[ V(i, j) = \max \begin{cases} V(i - 1, j - 1) + S(T[i], j) \\ V(i - 1, j) + s(T[i], -) \\ V(i, j - 1) + S(-, j) \end{cases} \]

Sequence-profile alignment...

... can be done using DP in time and space \( O(mn) \) if \( m \) and \( n \) are the length of the sequence and profile, respectively (details very similar to Needleman-Wunsch)
Aligning a Profile against a Profile

Profile-Profile Alignment

- Using a straightforward generalization we can define alignments between two profiles and their score.
Aligning a Profile against a Profile

Profile-Profile Alignment

- Using a straightforward generalization we can define alignments between two profiles and their score.
- Gaps can be inserted in both profiles.
Aligning a Profile against a Profile

Profile-Profile Alignment

- Using a straightforward generalization we can define alignments between two profiles and their score.
- Gaps can be inserted in both profiles.
- Scores are again defined as averages

\[ S(i, j) = \sum_{x, y} s(x, y)p_1(x, i)p_2(y, j). \]
Aligning a Profile against a Profile

Profile-Profile Alignment

- Using a straightforward generalization we can define alignments between two profiles and their score.
- Gaps can be inserted in both profiles.
- Scores are again defined as averages

\[ S(i, j) = \sum_{x, y} s(x, y)p_1(x, i)p_2(y, j). \]

- This problem will be used as subtask for progressive multiple alignment.
Aligning a Profile against a Profile

Profile-Profile Alignment

- Using a straightforward generalization we can define alignments between two profiles and their score.
- Gaps can be inserted in both profiles.
- Scores are again defined as averages

\[ S(i, j) = \sum_{x, y} s(x, y)p_1(x, i)p_2(y, j). \]

- This problem will be used as subtask for progressive multiple alignment.
- An alignment between two profiles (or between a profile and a sequence) representing alignments of sequences in disjoint sets \( M \) and \( N \) naturally defines a multiple alignment of all sequences in \( M \cup N \).

( chalk board )
Progressive Alignment Heuristic

Guide tree

- a binary tree $G$ with the sequences $S_1, \ldots, S_k$ as leaves
- determines the order of pairwise alignment steps
- an internal node $u$ represents an alignment of all leaves in the subtree rooted at $u$
- if $v$ and $w$ are children of $u$ then to get the alignment associated with $u$ we align the profiles or sequences associated with $v$ and $w$
- makes the progressive alignment method generic: methods differ in how the guide tree is constructed
Progressive Alignment Heuristic

Example

```
WCEAQTKNGQGWVPSNYITPVN
WWRLNDKEGYVPRLLLGLYP
AVVIQDNSDIKVVPKAKIIRD
YAVESEAHPGSFQPVAALERIN
WLNYNETTGERGDFPGTYVEYIGRKKISP
```

(example from Burkhard Morgenstern)
Progressive Alignment Heuristic

Example

WCEAQTKNGQGWVPSNYITPVN
WW--RLNDKEGYVPRNLLGLYP--AVVIQDNSDIKVVP--KAKIIRD
YAVESEASFQPVAALERIN
WLNYNEERGDFPGTYVEYIGRKKISP
⇒ first align 2nd and 3rd sequence
Progressive Alignment Heuristic

Example

```
WCEAQTKNGQGWVPSNYITPVN

WW--RLNDKEGYVPRNLLGLYP--KAKIIRD

YAVESEAVQ--PVAALERIN-------
WLNYNEERGDFPGTYVEYIGRKKISP
```

⇒ then 4th and 5th
Progressive Alignment Heuristic

Example

\[
\begin{align*}
WCEAQTKNGQGWVPSNYITPVN- \\
WW--RLNDKEGYVPRNLGLYP- \\
AVVIQDNSDIKVVP--KAKIIRD
\end{align*}
\]

\[
\begin{align*}
YAVESEASVQ--PVAALERIN----- \\
WLN-YNEERGDPFGTYVEYIGRKKISP
\end{align*}
\]

⇒ then the 1st to the profile of the alignment of \{2, 3\}
Progressive Alignment Heuristic

Example

```
WCEATKNGQGWVPSNYITPVN---------
WW--RLNDKEGYVPRNLGLYP--------
AVVIQDNSDIKVVP--KAKIIRD------
YAVESEA---SVQ---PVAALERIN-----
WLN--YNE---ERGDFPGTYVEYIGRKKISP
```

⇒ lastely, the two profiles of alignments of \{1, 2, 3\} and \{4, 5\}
Progressive Alignment Heuristic

Algorithm complexity

Let $k$ be the number of sequences and let $n$ be the maximal length of any sequence.
Progressive Alignment Heuristic

Algorithm complexity

Let $k$ be the number of sequences and let $n$ be the maximal length of any sequence. In each pairwise alignment step we require time and space $O(n^2)$. 
## Algorithm complexity

Let $k$ be the number of sequences and let $n$ be the maximal length of any sequence. In each pairwise alignment step we require time and space $O(n^2)$. There are $k - 1$ inner vertices in a binary tree with $k$ vertices.
Algorithm complexity

Let $k$ be the number of sequences and let $n$ be the maximal length of any sequence.

In each pairwise alignment step we require time and space $O(n^2)$.

There are $k - 1$ inner vertices in a binary tree with $k$ vertices. Therefore, the overall requirements are:

\[
\text{time} = O(kn^2) \\
\text{space} = O(kn^2)
\]
Problem of Progressive Alignments

“Once a gap – always a gap”

• Problem of greedy heuristic: false decisions are not taken back later
• Gap characters that are introduced will never disappear no matter what the evidence is.
• \( (\text{chalk board}) \)