Pairwise Alignment
Sequences are related

- Darwin: all organisms are related through descent with modification
- => Sequences are related through descent with modification
- => Similar molecules have similar functions in different organisms

Phylogenetic tree based on ribosomal RNA: three domains of life
Why compare sequences?

- Determination of evolutionary relationships
- Prediction of protein function and structure (database searches).

Protein 1: binds oxygen

\[
\text{Sequence similarity}
\]

Protein 2: binds oxygen?
Pairwise alignments

43.2% identity;  

Global alignment score: 374

alpha  
V-LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTT(KTYFPHF-DLS-----HGSA

beta   
VHLTPEEKSAVTALKV--NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNP

alpha  
QVKGKHKVADALTNAVHAVDDMPNALSALSDDAHKLVRDPVNFKLLSSHCLLVTLAHL

beta   
KVAHGKKVLGAFSDGLAHLDNKLGFTATLSHELCDKLHVDPENFRLLGNVLVCVLAHHF

alpha  
PASEFTPAVHASLDKFLASVSTVLT SKYR

beta   
GKEF TPPVQAAYQKVVAGVANALAHKYH
Pairwise alignments

43.2% identity; Global alignment score: 374

```
alpha  V-LSPADKTNVKAAWGKVGAHAGEYGAEEALERMFLSFPTTKTYFPHF-DLS-----HGSA
beta   VHLTPEEKSAVTALWGKV---NVDEVGGEALGRLLVYVPWTQRFFESFGDLSTPDAVMGNP
       10  20  30  40  50

alpha  QVKGHGKKVADALTNAVAHVDMPNALSLALSALHAKLKLVRDPVNFKLLSHCLLVTIAHLL
beta   KVKAHGGKVLGAFLSGLALNLKGFATALSHELCDKLHVDPENFRLLGNVLVCVLASSHF
       60  70  80  90  100  110

alpha  PAEFTPAVHASLDKFLASVSTVLTSKYR
beta   GKEFTPPVQAAYQKVAGVANALAHKYH
       120  130  140
```

100.000% identity in 3 aa overlap

SPA
:::
SPA
Alignment scores: match vs. mismatch

Simple scoring scheme (too simple in fact...):

Matching amino acids: 5
Mismatch: 0

Scoring example:

K A W S A D V
: : : : : :
K D W S A E V
5+0+5+5+5+0+5 = 25
B. Amino acids with polar but uncharged side chains

Serine (S) and Threonine (T) have similar physicochemical properties

Aspartic acid (D) and Glutamic acid (E) have similar properties

=> Substitution of S/T or E/D occurs relatively often during evolution

=> Substitution of S/T or E/D should result in scores that are only moderately lower than identities
Pairwise alignments: conservative substitutions

43.2% identity; Global alignment score: 374

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
<th>110</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>120</td>
<td>130</td>
<td>140</td>
<td>120</td>
<td>130</td>
<td>140</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Protein substitution matrices

|     | A   | R   | N   | D   | C   | Q   | E   | G   | H   | I   | L   | K   | M   | F   | P   | S   | T   | W   | Y   | V   |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| A   | 5   | -2  | -1  | -2  | -1  | -1  | -1  | 0   | -3  | -2  | -3  | -3  | 0   | -3  | -3  | -2  | 0   | -3  | -3  | 0   | -3  |
| R   | -2  | 7   | -1  | -1  | 7   | -2  | 2   | 8   | -1  | -4  | -2  | -4  | 13  | 0   | -3  | 0   | -1  | -3  | -2  | -3  | 8   |
| N   | -1  | -1  | 7   | -2  | 2   | 8   | 13  | 1   | 0   | -2  | 10  | 0   | -3  | 0   | -3  | 0   | -1  | -3  | 1   | 0   | -2  |
| D   | -2  | -1  | -1  | 7   | -1  | 0   | 0   | -3  | 2   | -3  | 2   | 6   | -1  | 0   | 0   | -3  | 0   | -1  | -3  | 1   | 0   | -2  |
| C   | -1  | -4  | -3  | -4  | -2  | -2  | -2  | -3  | -4  | -4  | -4  | 5   | -2  | -2  | -3  | -4  | -2  | -2  | -3  | -4  | -3  | 2   |
| Q   | -1  | 1   | 0   | 0   | -3  | 7   | 2   | 6   | 0   | -1  | -1  | -3  | 1   | 0   | -2  | 10  | 1   | 0   | -2  | -3  | -3  | 6   |
| E   | -1  | 1   | 0   | 0   | -3  | 7   | 2   | 6   | 0   | -1  | -1  | -3  | 1   | 0   | -2  | 10  | 1   | 0   | -2  | -3  | -3  | 6   |
| G   | 0   | -3  | 0   | -1  | -3  | -2  | -3  | 8   | -2  | 0   | -2  | 1   | 0   | -3  | -3  | 6   | 0   | -2  | -3  | -3  | 6   | 0   |
| H   | -2  | 0   | 1   | -1  | -3  | 1   | 0   | -2  | 10  | 1   | 0   | -3  | -3  | 6   | 0   | -2  | -3  | -3  | 6   | 0   | -2  |
| I   | -1  | -4  | -3  | -4  | -2  | -2  | -2  | -3  | -4  | -4  | -4  | 5   | -2  | -2  | -3  | -4  | -2  | -2  | -3  | -4  | -3  | 2   |
| L   | -2  | -3  | -4  | -4  | -2  | -2  | -3  | -4  | -3  | -3  | 2   | 5   | 1   | 0   | -3  | -3  | 6   | 0   | -2  | -3  | -3  | 6   | 0   |
| K   | -1  | 3   | 0   | -1  | -3  | 2   | 1   | 2   | 1   | -2  | 0   | -3  | -3  | 6   | 0   | -2  | -3  | -3  | 6   | 0   | -2  |
| M   | -1  | -2  | -2  | -4  | -2  | 0   | -2  | -3  | -1  | 2   | 3   | 2   | 7   | 2   | 3   | 2   | 7   | 2   | 3   | 2   | 7   | 2   |
| F   | -3  | -3  | -4  | -5  | -2  | -4  | -3  | -4  | -1  | 0   | 1   | -4  | 0   | 8   | 0   | 1   | -4  | 0   | 8   | 0   | 1   |
| P   | -1  | -3  | -2  | -1  | -4  | -1  | -1  | -2  | -2  | -3  | -4  | -1  | -3  | -4  | 10  | 0   | 1   | 0   | -4  | 0   | 1   |
| S   | 1   | -1  | 1   | 0   | -1  | 0   | -1  | -3  | -3  | 0   | -2  | -3  | -1  | 5   | 0   | 1   | -4  | 0   | 1   |
| T   | 0   | -1  | 0   | -1  | -1  | -1  | -1  | -2  | -2  | -1  | -1  | -1  | -2  | 1   | 2   | 5   | 0   | -4  | 0   | 1   |
| W   | -3  | -3  | -4  | -5  | -2  | -4  | -3  | -4  | -1  | 0   | 1   | -4  | -4  | -3  | 15  | 1   | -4  | -4  | -3  | 15  |
| Y   | -2  | -1  | -2  | -3  | -3  | -1  | -2  | -3  | 2   | -1  | -1  | -2  | 0   | 4   | -3  | -2  | 2   | 8   | 2   |
| V   | 0   | -3  | -3  | -4  | -1  | -3  | -3  | -4  | -4  | 4   | 1   | -3  | 1   | -1  | -3  | -2  | 0   | -3  | -1  | 5   |

**BLOSUM50 matrix:**

- Positive scores on diagonal (identities)
- Similar residues get higher (positive) scores
- Dissimilar residues get smaller (negative) scores
Pairwise alignments: insertions/deletions

43.2% identity;  

<table>
<thead>
<tr>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha:</td>
<td>V-LSPADKTNVKA</td>
<td>AWGKVG</td>
<td>A</td>
<td>G</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
<th>110</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>120</th>
<th>130</th>
<th>140</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha:</td>
<td>PAE</td>
<td>F</td>
</tr>
<tr>
<td>beta:</td>
<td>G</td>
<td>K</td>
</tr>
</tbody>
</table>
Alignment scores: insertions/deletions

Affine gap penalties:
Multiple insertions/deletions may be one evolutionary event =>
Separate penalties for gap opening and gap elongation
Compute 4 alignment scores: two different alignments using two different alignment matrices (and the same gap penalty system)

Score 1: Alignment 1 + BLOSUM-50 matrix + gaps
Score 2: Alignment 1 + BLOSUM-Trp matrix + gaps
Score 3: Alignment 2 + BLOSUM-50 matrix + gaps
Score 4: Alignment 2 + BLOSUM-Trp matrix + gaps

Note: fake matrix constructed for pedagogic purposes.
### Handout: summary of results

<table>
<thead>
<tr>
<th></th>
<th>Alignment 1</th>
<th>Alignment 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOSUM-50</td>
<td>38</td>
<td>51</td>
</tr>
<tr>
<td>BLOSUM-Trp</td>
<td>118</td>
<td>91</td>
</tr>
</tbody>
</table>
Protein substitution matrices: different types

- **Identity matrix**
  (match vs. mismatch)

- **Chemical properties matrix**
  (use knowledge of physicochemical properties to design matrix)

- **Empirical matrices**
  (based on observed pair-frequencies in hand-made alignments)
  - PAM series
  - BLOSUM series
  - Gonnet
Searching for the *optimal alignment*...
The problem:
How many possible alignments are there?

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ACG</td>
<td>AC–G</td>
<td>--ACG</td>
<td>–A–CG</td>
</tr>
<tr>
<td>ACG</td>
<td>ACG–</td>
<td>AC–G–</td>
<td>A–CG–</td>
</tr>
<tr>
<td>--ACG</td>
<td>AC–G</td>
<td>--ACG</td>
<td>...</td>
</tr>
<tr>
<td>ACG–</td>
<td>A–CG</td>
<td>A–CG–</td>
<td></td>
</tr>
<tr>
<td>--ACG</td>
<td>AC–G</td>
<td>--ACG</td>
<td></td>
</tr>
<tr>
<td>AC–G</td>
<td>–ACG</td>
<td>AC--G</td>
<td></td>
</tr>
<tr>
<td>--ACG</td>
<td>ACG–</td>
<td>--ACG</td>
<td></td>
</tr>
<tr>
<td>A–CG</td>
<td>AC–G</td>
<td>A–C–G</td>
<td></td>
</tr>
<tr>
<td>--ACG</td>
<td>ACG–</td>
<td>--ACG</td>
<td></td>
</tr>
<tr>
<td>ACG–</td>
<td>A–CG</td>
<td>A--CG</td>
<td></td>
</tr>
<tr>
<td>A–CG</td>
<td>ACG–</td>
<td>--A–CG</td>
<td></td>
</tr>
<tr>
<td>AC–G</td>
<td>–ACG</td>
<td>ACG--</td>
<td></td>
</tr>
<tr>
<td>A–CG</td>
<td>--ACG</td>
<td>–A–CG</td>
<td></td>
</tr>
<tr>
<td>--ACG</td>
<td>ACG--</td>
<td>AC–G–</td>
<td></td>
</tr>
</tbody>
</table>
The problem:
How many possible alignments are there?

ACG    AC–G     --ACG     --A–CG
ACG    ACG–     AC–G–     A–CG–

--ACG    AC–G     --ACG     ...

Two protein sequences of length 100 amino acids can be aligned in approximately $10^{60}$ different ways.

Time needed to test all possibilities is same order of magnitude as the entire lifetime of the universe.

A–CG    ACG–     --ACG
ACG–    A–CG     A–CG

A–CG    ACG–     --A–CG
AC–G    --ACG     ACG--

A–CG    --ACG     --A–CG
--ACG    ACG--     AC–G–
Solution:
Dynamic programming
TCGCA
TCCA
Pairwise alignment: the solution

"Dynamic programming"
(the Needleman-Wunsch algorithm)

Filling a scoring matrix
Alignment depicted as path in matrix

TCGCA
TC-CA

TCGCA
T-CCA
Dynamic programming: computation of scores

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>C</th>
<th>G</th>
<th>C</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Any given point in matrix can only be reached from three possible previous positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.
Dynamic programming: computation of scores

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>C</th>
<th>G</th>
<th>C</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Any given point in matrix can only be reached from three possible positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

\[
score(x, y) = \max \left\{ \text{score}(x, y-1) - \text{gap-penalty} \right\}
\]
Dynamic programming: computation of scores

Any given point in matrix can only be reached from three possible positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

\[
\text{score}(x,y) = \max \begin{cases} 
\text{score}(x,y-1) - \text{gap-penalty} \\
\text{score}(x-1,y-1) + \text{substitution-score}(x,y)
\end{cases}
\]
Dynamic programming: computation of scores

Any given point in matrix can only be reached from three possible positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>C</th>
<th>G</th>
<th>C</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
score(x,y) = \max \begin{cases} 
  \text{score}(x,y-1) - \text{gap-penalty} \\
  \text{score}(x-1,y-1) + \text{substitution-score}(x,y) \\
  \text{score}(x-1,y) - \text{gap-penalty} 
\end{cases}
\]
Dynamic programming: computation of scores

Any given point in matrix can only be reached from three possible positions (you cannot “align backwards”).

=> Best scoring alignment ending in any given point in the matrix can be found by choosing the highest scoring of the three possibilities.

Each new score is found by choosing the maximum of three possibilities. For each square in matrix: keep track of where best score came from.

Fill in scores one row at a time, starting in upper left corner of matrix, ending in lower right corner.

\[
\text{score}(x, y) = \max \begin{cases} 
\text{score}(x, y-1) - \text{gap-penalty} \\
\text{score}(x-1, y-1) + \text{substitution-score}(x, y) \\
\text{score}(x-1, y) - \text{gap-penalty} 
\end{cases}
\]
Dynamic programming: example

\[ a_{i,j} = \max \begin{cases} a_{i,j-1} - 2 \\ a_{i-1,j-1} + p(i,j) \\ a_{i-1,j} - 2 \end{cases} \]

Gaps: -2
Dynamic programming: example

![Diagram of dynamic programming example]

\[ a[i,j] = \max \begin{cases} 
  a[i,j-1] - 2 \\
  a[i-1,j-1] + p(i,j) \\
  a[i-1,j] - 2 
\end{cases} \]
Dynamic programming: example

\[ a[i,j] = \max\begin{cases} a[i,j-1] - 2 \\ a[i-1,j-1] + p(i,j) \\ a[i-1,j] - 2 \end{cases} \]
Dynamic programming: example
Dynamic programming: example
Dynamic programming: example

\[
a[i,j] = \max \begin{cases} 
a[i,j-1] - 2 \\
a[i-1,j-1] + p(i,j) \\
a[i-1,j] - 2 \end{cases}
\]
Dynamic programming: example

\[
\begin{array}{ccccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
0 & -2 & -4 & -6 & -8 & -10 \\
T & 1 & -1 & -3 & -5 & -7 \\
T & -2 & 1 & 2 & 0 & -2 \\
s[i] & -4 & -1 & 0 & 1 & -1 \\
C & -6 & -3 & 1 & -1 & -2 \\
C & -8 & -5 & -2 & 0 & 2 \\
A & -8 & -5 & -2 & -1 & 0 \\
A & -8 & -5 & -2 & 0 & 2 \\
\end{array}
\]
Dynamic programming: example

\[
\begin{array}{ccccccc}
& T & C & G & C & A \\
0 & 0 & -2 & -4 & -6 & -8 & -10 \\
T_1 & -2 & 1 & -1 & -3 & -5 & -7 \\
T_2 & -4 & -1 & 2 & 0 & -2 & -4 \\
T_3 & -6 & -3 & 0 & 1 & 1 & -1 \\
T_4 & -8 & -5 & -2 & -1 & 0 & 2 \\
\end{array}
\]

\[
\frac{1+1-2+1+1}{T C - C A} = 2
\]
Global versus local alignments

Global alignment: align full length of both sequences. (The “Needleman-Wunsch” algorithm).

Local alignment: find best partial alignment of two sequences (the “Smith-Waterman” algorithm).
Local alignment overview

- The recursive formula is changed by adding a fourth possibility: zero. This means local alignment scores are never negative.

\[
\text{score}(x,y) = \max \begin{cases} 
\text{score}(x,y-1) - \text{gap-penalty} \\
\text{score}(x-1,y-1) + \text{substitution-score}(x,y) \\
\text{score}(x-1,y) - \text{gap-penalty} \\
0
\end{cases}
\]

- Trace-back is started at the highest value rather than in lower right corner
- Trace-back is stopped as soon as a zero is encountered
### Local alignment: example

<table>
<thead>
<tr>
<th></th>
<th>H</th>
<th>E</th>
<th>A</th>
<th>G</th>
<th>A</th>
<th>W</th>
<th>G</th>
<th>H</th>
<th>E</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>20</td>
<td>12</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>10</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>2</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>28</td>
<td>20</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>21</td>
<td>13</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>13</td>
<td>18</td>
<td>12</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>16</td>
</tr>
</tbody>
</table>

**AWGHE**

**AW-HE**
Substitution matrices and sequence similarity

- Substitution matrices come as series of matrices calculated for different degrees of sequence similarity (different evolutionary distances).

- "Hard" matrices are designed for similar sequences
  - Hard matrices are designated by high numbers in the BLOSUM series (e.g., BLOSUM80)

- "Soft" matrices are designed for less similar sequences
  - Soft matrices have low BLOSUM values (45)
Substitution matrices and sequence similarity

- Substitution matrices come as series of matrices calculated for different degrees of sequence similarity (different evolutionary distances).

- "Hard" matrices are designed for similar sequences
  - Hard matrices a designated by high numbers in the BLOSUM series (e.g., BLOSUM80)
  - Hard matrices yield short, highly conserved alignments

- "Soft" matrices are designed for less similar sequences
  - Soft matrices have low BLOSUM values (45)
  - Soft matrices yield longer, less well conserved alignments
What did you learn?

- Purpose of sequence alignment:
  - Find evolutionary relationships
  - Predict protein function

- The optimal alignment can be found using dynamic programming:
  - Local alignment: Smith-Waterman
  - Global alignment: Needleman-Wunsch

- Optimal alignment means having the best possible score given:
  1. Substitution matrix
  2. Set of gap penalties

- The optimal alignment is NOT necessarily the most biologically meaningful
Exercise 1 – Pairwise Alignment
Database Searching
Database searching

Using pairwise alignments to search databases for similar sequences
Database searching

Most common use of pairwise sequence alignments is to search databases for related sequences. For instance: find probable function of newly isolated protein by identifying similar proteins with known function.

Most often, *local alignment* ("Smith-Waterman") is used for database searching: you are interested in finding out if ANY domain in your protein looks like something that is known.

Often, full Smith-Waterman is too time-consuming for searching large databases, so heuristic methods are used (fasta, BLAST).
Database searching: heuristic search algorithms

FASTA (Pearson 1995)

- Uses heuristics to avoid calculating the full dynamic programming matrix
- Speed up searches by an order of magnitude compared to full Smith-Waterman
- The statistical side of FASTA is still stronger than BLAST

BLAST (Altschul 1990, 1997)

- Uses rapid word lookup methods to completely skip most of the database entries
- Extremely fast
- One order of magnitude faster than FASTA
- Two orders of magnitude faster than Smith-Waterman
- Almost as sensitive as FASTA
<table>
<thead>
<tr>
<th>BLAST flavors</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BLASTN</strong></td>
</tr>
<tr>
<td>Nucleotide query sequence</td>
</tr>
<tr>
<td>Nucleotide database</td>
</tr>
<tr>
<td><strong>BLASTP</strong></td>
</tr>
<tr>
<td>Protein query sequence</td>
</tr>
<tr>
<td>Protein database</td>
</tr>
<tr>
<td><strong>BLASTX</strong></td>
</tr>
<tr>
<td>Nucleotide query sequence</td>
</tr>
<tr>
<td>Protein database</td>
</tr>
<tr>
<td><strong>TBLASTN</strong></td>
</tr>
<tr>
<td>Protein query sequence</td>
</tr>
<tr>
<td>Nucleotide database</td>
</tr>
<tr>
<td>&quot;On the fly&quot; six frame translation of database</td>
</tr>
<tr>
<td><strong>TBLASTX</strong></td>
</tr>
<tr>
<td>Nucleotide query sequence</td>
</tr>
<tr>
<td>Nucleotide database</td>
</tr>
<tr>
<td>Compares all reading frames of query with all reading frames of the database</td>
</tr>
</tbody>
</table>
Searching on the web: BLAST at NCBI

Very fast computers dedicated to running BLAST searches

Many databases that are always up to date (e.g. NR and Human Genome)

Nice simple web interface

But you still need knowledge about BLAST to use it properly
When is a database hit significant?

**Problem:**

- Even unrelated sequences can be aligned (yielding a low score)
- How do we know if a database hit is meaningful?
- When is an alignment score sufficiently high?

**Solution:**

- Determine the range of alignment scores you would expect to get for random reasons (i.e., when aligning unrelated sequences).
- Compare actual scores to the distribution of random scores.
- Is the real score much higher than you’d expect by chance?
Significance of alignment score expressed as E-value

Searching a database of unrelated sequences results in scores following an extreme value distribution.

The exact shape and location of the distribution depends on the exact nature of the database and the query sequence.

**E-value:** the number of random hits to expect for any given score

Want E-values below 1 (the lower the better)
What did you learn?

• Heuristic methods (e.g. BLAST) is faster to search databases than Smith-Waterman

• BLASTN, BLASTP etc.

• E-value: Measure for significance of database hit
Exercise 2 – Using BLAST