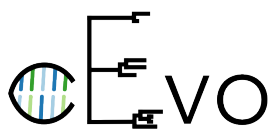
Computational Biology

Lecturers:
Tanja Stadler, Carsten Magnus & Tim Vaughan

Teaching Assistants:
Jūlija Pečerska, Jérémie Sciré,
Sarah Nadeau & Marc Manceau

Computational Evolution
Department of Biosystems Science and Engineering

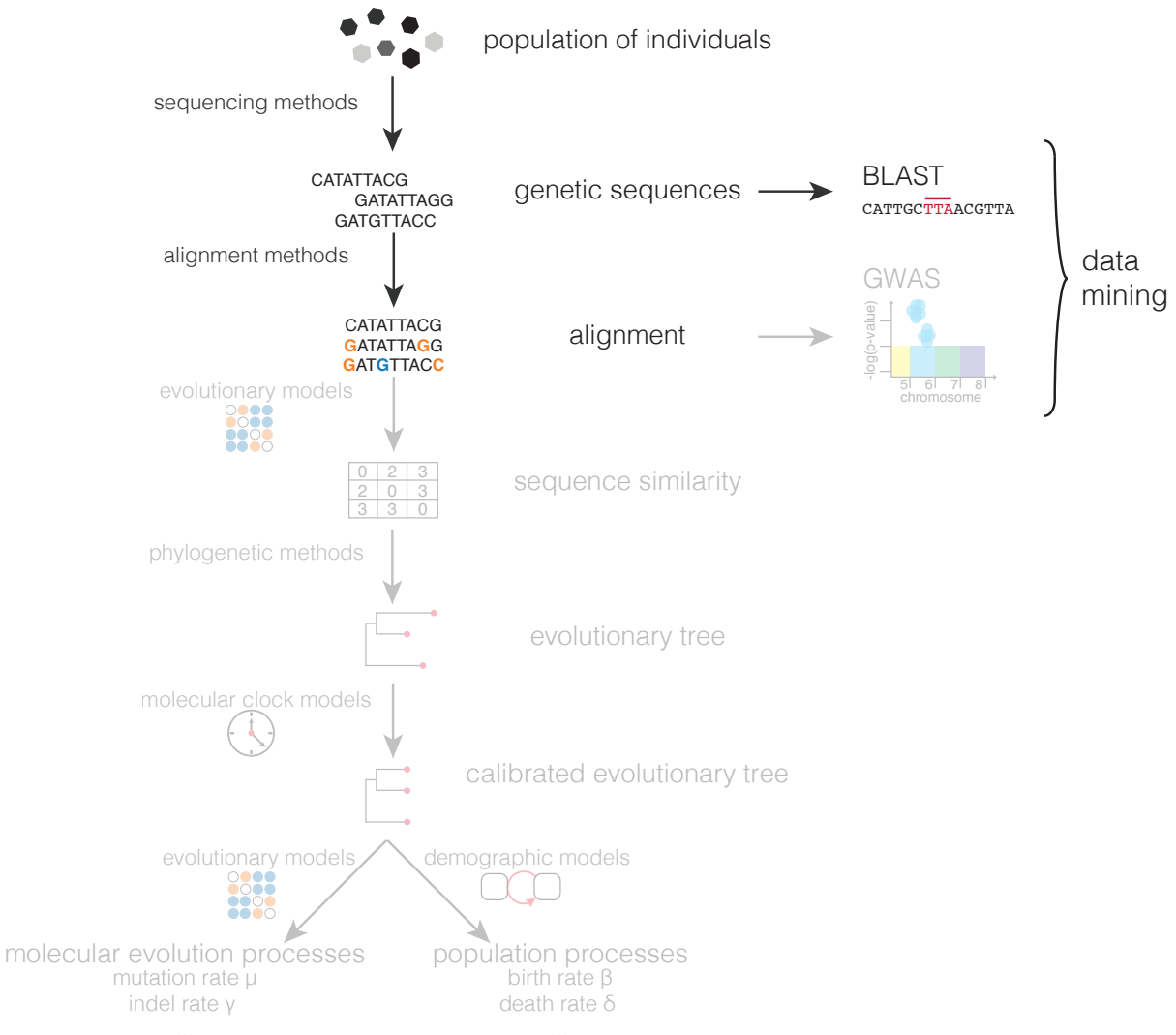
HS 2019



Sequencing Questions

- Name one weakness and one strength for each of the different sequencing methods.
- With which of the methods presented could you conceivably sequence the genome of a particular cell?
- Imagine you want to perform a paternity test. How would you go about testing whether the potential father is really the father? Could you think of reasons to find a false-negative relatedness?

Overview and outline of lecture 02



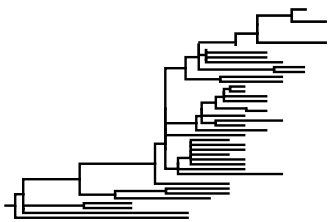
Key concepts.

Levels of evolution

genotype

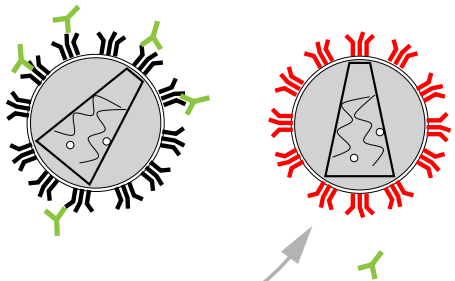
sequence level

```
ACUGAACGUGACUACUG
ACUGAACGUAACUACUG
```



phenotype

e.g. antigenic level: Antibody binding to HIV



codon: three nucleotides encode for one amino acid

one nucleotide change can already change the phenotype

alphabet:

4 nucleotides: DNA: TCAG
RNA: UCAG

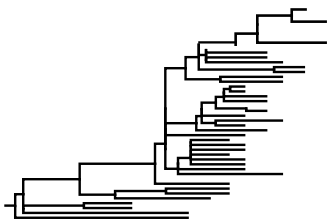
20 amino acids

Levels of evolution

genotype

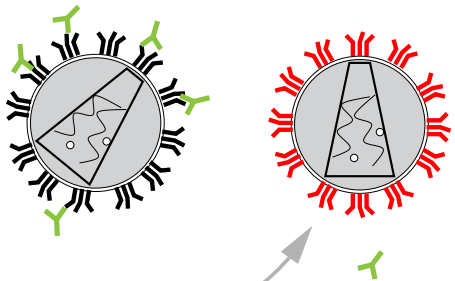
sequence level

```
ACUGAACGUGACUACUG
ACUGAACGUAACUACUG
```



phenotype

e.g. antigenic level: Antibody binding to HIV



codon: three nucleotides encode for one amino acid

one nucleotide change can already change the phenotype

alphabet:

4 nucleotides: DNA: TCAG
RNA: UCAG

20 amino acids

When comparing two nucleotide sequences we have to keep in mind that they are the result of mutation during replication (genotypic level) and selection (phenotypic level).

What can happen when DNA/RNA is copied

- mutation
- insertion
- deletion
- repeats
- inversions
- inverted repeats

What can happen when DNA/RNA is copied

- mutation
- insertion
- deletion
- repeats
- inversions
- inverted repeats

When we compare two sequences we need to know which positions in the sequences correspond to each other.

Pairwise sequence alignments.

Example: Comparison of triosephosphate isomerase

Triosephosphate isomerase:

- enzyme catalyzing the reversible interconversion of the triose phosphate isomers dihydroxyacetone phosphate and D-glyceraldehyde 3-phosphate
- essential for efficient energy production
- has been found in nearly every organism (including mammals, insects, fungi, plants, and bacteria)

Example: Comparison of triosephosphate isomerase

Triosephosphate isomerase:

- enzyme catalyzing the reversible interconversion of the triose phosphate isomers dihydroxyacetone phosphate and D-glyceraldehyde 3-phosphate
- essential for efficient energy production
- has been found in nearly every organism (including mammals, insects, fungi, plants, and bacteria)

NGTTDQVDKIVKILNEGQIASTDVVEVVVSPPYVFLPVVKSQLRPEIQVAAQNCW

NGDKASIADLCKVLTTGPLNAD__TEVVVGCPAPYLTLARSQLPDSVCVAAQNCY

Example: Comparison of triosephosphate isomerase

Triosephosphate isomerase:

- enzyme catalyzing the reversible interconversion of the triose phosphate isomers dihydroxyacetone phosphate and D-glyceraldehyde 3-phosphate
- essential for efficient energy production
- has been found in nearly every organism (including mammals, insects, fungi, plants, and bacteria)

```
NGTTDQVDKIVKILNEGQIASTDVVEVVVSPPYVFLPVVKSQLRPEIQVAAQNCW
||....!..!.|!|..|.!.:.  .||||. | .!|.:.!|||...! ||||||!
NGDKASIADLCKVLTTGPLNAD__TEVVVGCPAPYLTLARSQLPDSVCVAAQNCY
```

Visual comparison shows 36.4% identity

Problem: Which position is a mutation, which position is a deletion?

```
NGTTDQVDKIVKILNEGQIASTDVVEVVVSPPYVFLPVVKSQLRPEIQVAAQNCW
||....!..!.|!|..|.!.:.  .||||. | .!|.:.!|||...! ||||||!
NGDKASIADLCKVLTTGPLNAD__TEVVVGCPAPYLTLARSQLPDSVCVAAQNCY
```

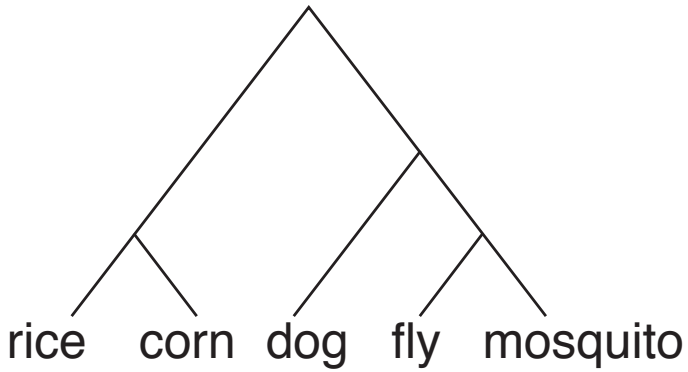
- How do we know at which position we have a mutation, an insertion or a deletion?

Problem: Which position is a mutation, which position is a deletion?

```
NGTTDQVDKIVKILNEGQIASTDVVEVVVSPPYVFLPVVKSQLRPEIQVAAQNCW
||....!..!.|!|..|.!.:.  .||||. | .!|.:.!|||...! ||||||!
NGDKASIADLCKVLTTGPLNAD__TEVVVGCPAPYLTLARSQLPDSVCVAAQNCY
```

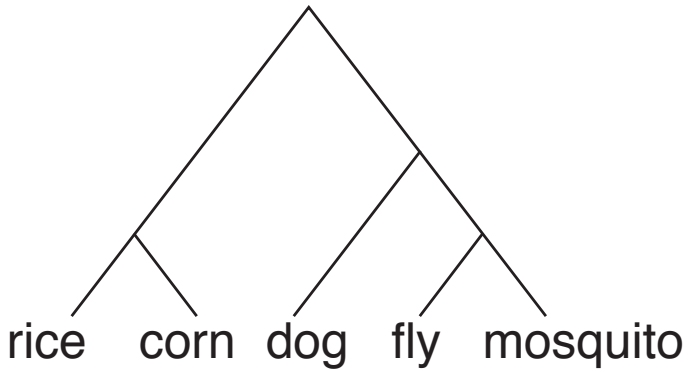
- How do we know at which position we have a mutation, an insertion or a deletion?
- We need alignments: a ‘correct’ alignment represents actual events such as substitutions and indels (= **in**sertion and **del**etions)

Alignment and phylogenies



Alignments are based on the idea that there is a common ancestor from which the genes have evolved. Thus, the ancestor had a certain nucleotide (or amino acid in the protein) at a certain position. The nucleotide (or protein) could have changed during the evolutionary history towards the descendants. Sequences with shared ancestry are referred to as **homologous.**

Alignment and phylogenies



Alignments are based on the idea that there is a common ancestor from which the genes have evolved. Thus, the ancestor had a certain nucleotide (or amino acid in the protein) at a certain position. The nucleotide (or protein) could have changed during the evolutionary history towards the descendants. Sequences with shared ancestry are referred to as **homologous.**

- obviously impossible to know for sure $\rightarrow$ take the alignment which has the highest probability/score (out of all alignments) under a certain model

Types of alignment

Pairwise alignments:

- protein-protein (as in the example)
- DNA/DNA alignments
- RNA/RNA alignments
- DNA or RNA with protein: very complicated as 3 nucleotides (codon) encode for one amino acid, here one must look for gaps and frameshifts within codons

Types of alignment

Pairwise alignments:

- protein-protein (as in the example)
- DNA/DNA alignments
- RNA/RNA alignments
- DNA or RNA with protein: very complicated as 3 nucleotides (codon) encode for one amino acid, here one must look for gaps and frameshifts within codons

In addition, to pairwise sequence alignment, we can also have a *multiple sequence alignment (MSA)*.

Pairwise sequence alignment

Example: align `AACTGCAAA` and `ACTACCA`

Local versus global alignments

Global alignment:

aligns one sequence to the other from start to the end

Local alignment:

finds the longest subsequences with highest similarity

Strategies for alignments

Several strategies to find a pairwise sequence alignment:

- Qualitative method: Dot-matrix method
- Exact method via dynamic programming: Needleman-Wunsch [Needleman and Wunsch, 1970] algorithm for global alignments and Smith-Waterman [Smith and Waterman, 1981] algorithm for local alignments
- Heuristic and fast methods: Word methods: e.g. BLAST

The dot-matrix method

Visually easy method to identify certain features of the two sequences.

To exemplify this method, we align the following sequences:

`CTG` and `CTAAG` `CTAAGAAG` and `CTAAG` `ATC` and `CTAAG`

The dot-matrix method

Visually easy method to identify certain features of the two sequences.
To exemplify this method, we align the following sequences:

`CTG` and `CTAAG` `CTAAGAAG` and `CTAAG` `ATC` and `CTAAG`

To do so, we arrange them in a matrix and highlight matching nucleotides:

	C	T	A	A	G
C					
T					
G					

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

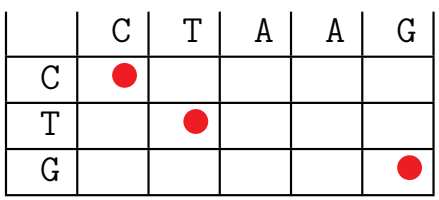
	C	T	A	A	G
A					
T					
C					

The dot-matrix method

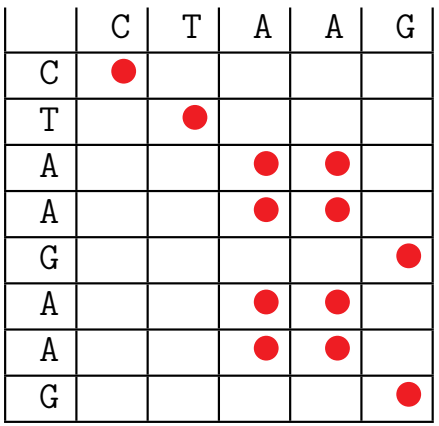
Visually easy method to identify certain features of the two sequences.
To exemplify this method, we align the following sequences:

`CTG` and `CTAAG` `CTAAGAAG` and `CTAAG` `ATC` and `CTAAG`

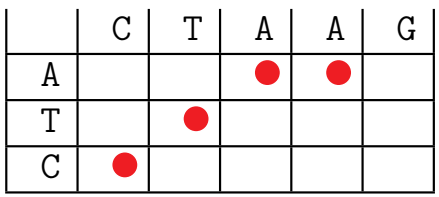
To do so, we arrange them in a matrix and highlight matching nucleotides:



gap in matrix
=
gap in sequence



repeated patterns/blocks = repeats



'reflected' diagonals
=
inversions

Dot-matrices: Pros and Cons

Pro:

- visually easy method to identify sequence features such as indels, repeats, inversions and inverted repeats

Cons:

- time-consuming
- does not give one optimal alignment

Quantitative alignment methods

- We now focus on a strategy for local alignments.
- Big question: Do we accept that a position in our alignment has a mutation or do we introduce a gap?

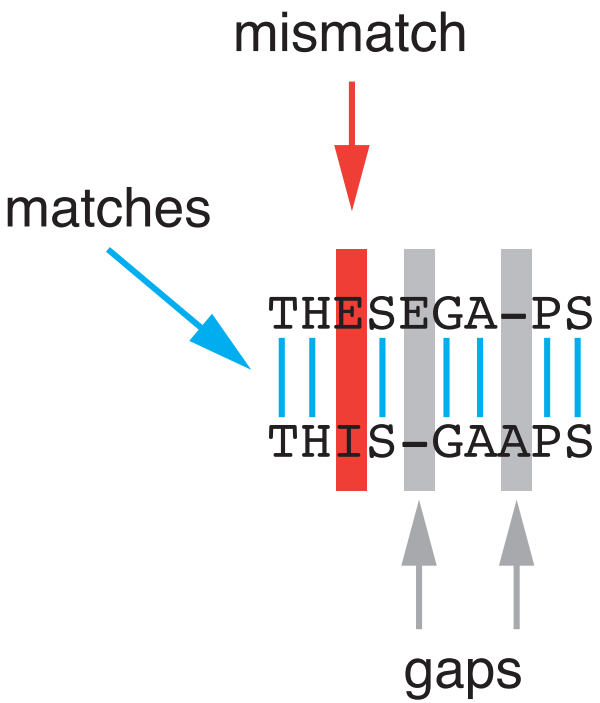
Quantitative alignment methods

- We now focus on a strategy for local alignments.
- Big question: Do we accept that a position in our alignment has a mutation or do we introduce a gap?

- **Strategy:** assign costs for different actions in the alignment process

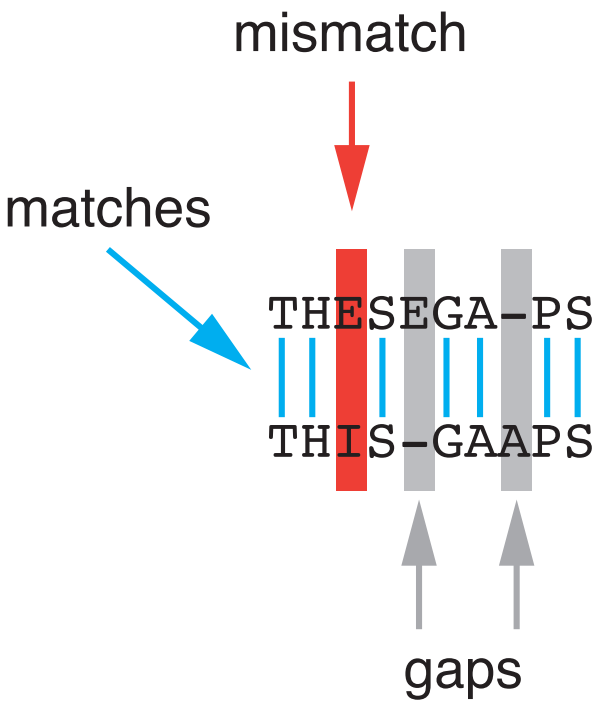
Basic idea

When comparing characters at one position, there are three possibilities:



Basic idea

When comparing characters at one position, there are three possibilities:



Compare each position of one sequence and ask if it is possible that this position aligns with any position of the other sequence.

Assign a score to each pairing, thereby:

- reward a match, e.g. score=3
- punish a mismatch, e.g. score=-1
- punish a gap, e.g. score=-2

Brute force approach

Task: find the optimal alignment of `AGAC` and `AATC`.

First strategy: find all alignments and score them.

Brute force approach

Task: find the optimal alignment of `AGAC` and `AATC`.

First strategy: find all alignments and score them.

- match: score=3
- mismatch: score=-1
- gap: score=-2

Brute force approach

Task: find the optimal alignment of `AGAC` and `AATC`.

First strategy: find all alignments and score them.

- match: score=3
- mismatch: score=-1
- gap: score=-2

...there are so many possible alignments

Number of possible alignments

How many possible alignments are there?

Let $a = a_1 a_2 \dots a_{m-1} a_m$ be a sequence of length m and $b = b_1 b_2 \dots b_{n-1} b_n$ a sequence of length n, *wlog* $n \leqslant m$. Let k be the number of gaps to be introduced into sequence a.

Number of possible alignments

How many possible alignments are there?

Let $\mathbf{a} = \mathbf{a}_1\mathbf{a}_2 \ldots \mathbf{a}_{\mathbf{m}-1}\mathbf{a}_{\mathbf{m}}$ be a sequence of length $\mathbf{m}$ and $\mathbf{b} = \mathbf{b}_1\mathbf{b}_2 \ldots \mathbf{b}_{\mathbf{n}-1}\mathbf{b}_{\mathbf{n}}$ a sequence of length $\mathbf{n}$, *wlog* $\mathbf{n} \leqslant \mathbf{m}$. Let $\mathbf{k}$ be the number of gaps to be introduced into sequence $\mathbf{a}$.

There are then
$\binom{\mathbf{m}+\mathbf{k}}{\mathbf{k}}$
possibilities to place these gaps
between the $\mathbf{m}$ letters of $\mathbf{a}$

Number of possible alignments

How many possible alignments are there?

Let $a = a_1 a_2 \dots a_{m-1} a_m$ be a sequence of length m and $b = b_1 b_2 \dots b_{n-1} b_n$ a sequence of length n, *wlog* $n \leqslant m$. Let k be the number of gaps to be introduced into sequence a.

There are then
$\binom{m+k}{k}$
possibilities to place these gaps between the m letters of a

Example:
$m = 5$, $n = 3$, $k = 1$

pos.	1	2	3	4	5	6
a	a_1	–	a_2	a_3	a_4	a_5

Number of possible alignments

How many possible alignments are there?

Let $\mathtt{a} = \mathtt{a}_1\mathtt{a}_2 \ldots \mathtt{a}_{\mathtt{m}-1}\mathtt{a}_{\mathtt{m}}$ be a sequence of length $\mathtt{m}$ and $\mathtt{b} = \mathtt{b}_1\mathtt{b}_2 \ldots \mathtt{b}_{\mathtt{n}-1}\mathtt{b}_{\mathtt{n}}$ a sequence of length $\mathtt{n}$, *wlog* $\mathtt{n} \leqslant \mathtt{m}$. Let $\mathtt{k}$ be the number of gaps to be introduced into sequence $\mathtt{a}$.

There are then
$\binom{\mathtt{m}+\mathtt{k}}{\mathtt{k}}$
possibilities to place these gaps between the $\mathtt{m}$ letters of $\mathtt{a}$

and
$\binom{\mathtt{m}}{\mathtt{m}-\mathtt{n}+\mathtt{k}}$
possibilities to place the number of gaps in $\mathtt{b}$ at the positions of $\mathtt{a}$

Example:
$\mathtt{m} = 5$, $\mathtt{n} = 3$, $\mathtt{k} = 1$

pos.	1	2	3	4	5	6
$\mathtt{a}$	$\mathtt{a}_1$	–	$\mathtt{a}_2$	$\mathtt{a}_3$	$\mathtt{a}_4$	$\mathtt{a}_5$

Number of possible alignments

How many possible alignments are there?

Let $a = a_1 a_2 \dots a_{m-1} a_m$ be a sequence of length m and $b = b_1 b_2 \dots b_{n-1} b_n$ a sequence of length n, *wlog* $n \leqslant m$. Let k be the number of gaps to be introduced into sequence a.

There are then
$\binom{m+k}{k}$
possibilities to place these gaps between the m letters of a

and
$\binom{m}{m-n+k}$
possibilities to place the number of gaps in b at the positions of a

Example:
$m = 5$, $n = 3$, $k = 1$

pos.	1	2	3	4	5	6
a		a_1	$-$	a_2	a_3	a_4	a_5
b	$-$	b_1	$-$	b_2	b_3	$-$

Number of possible alignments

Thus, the number of different alignments with k gaps in sequence a is:

$$\binom{m+k}{k}\binom{m}{m-n+k}$$

Number of possible alignments

Thus, the number of different alignments with k gaps in sequence a is:

$$\binom{m+k}{k}\binom{m}{m-n+k}$$

As we do not want to align gaps with gaps, k can only range from 0 to n. This means:

The total number of alignments between a sequence of length m and a sequence of length $n \leqslant m$ is:

$$\sum_{k=0}^{n} \binom{m+k}{k}\binom{m}{m-n+k}$$

Number of possible alignments

The number of possible alignments between a sequence of length $\mathfrak{m}$ and a sequence of length $\mathfrak{n} \leqslant \mathfrak{m}$ are:

$\mathfrak{m}\backslash\mathfrak{n}$	1	2	3	4	5
1	3				
2	5	13			
3	7	25	63		
4	9	41	129	321	
5	11	61	231	681	1683

For $\mathfrak{m}, \mathfrak{n} = 100$, which are short sequences, this brute force approach would need to calculate the scores of 2.05×10^{75} alignments!!!

$\rightarrow$ We need to find another strategy to align sequences!

Dynamic programming for sequence alignments

In order to not calculate scores for all possible alignments, dynamic programming was used:

- for global alignments: [Needleman and Wunsch, 1970] cited 5852 times (01 Oct 2018)
- for local alignments: [Smith and Waterman, 1981] cited 4876 times (01 Oct 2018)

Dynamic programming for sequence alignments

In order to not calculate scores for all possible alignments, dynamic programming was used:

- for global alignments: [Needleman and Wunsch, 1970] cited 5852 times (01 Oct 2018)
- for local alignments: [Smith and Waterman, 1981] cited 4876 times (01 Oct 2018)

Wikipedia In mathematics, management science, economics, computer science, and bioinformatics, **dynamic programming** (also known as dynamic optimization) is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions – ideally, using a memory-based data structure. The next time the same subproblem occurs, instead of recomputing its solution, one simply looks up the previously computed solution, thereby saving computation time at the expense of a (hopefully) modest expenditure in storage space.

The Smith-Waterman algorithm

The aim: Find the best **local** alignment of two sequences of length $\mathbf{m}$ and $\mathbf{n}$

Notation: seqA=$\mathbf{a_1\, a_2\, \ldots\, a_m}$, seqB=$\mathbf{b_1\, b_2\, \ldots\, b_n}$

The idea:

- dynamic programming: calculate optimal alignment to one point only once
- build a matrix for the two sequences with seqA corresponding to the row entries, seqB to the column entries
- the field $(\mathbf{i}, \mathbf{j})$ corresponds to the score of the optimal alignment with the nucleotides $\mathbf{a_i}$ and $\mathbf{b_j}$ as end of the alignment
- find the best way through the complete matrix
- to calculate: $\mathbf{m} \times \mathbf{n}$ steps

Example

Task: Align seqA=AATC and seqB=AGAC.

The score matrix:

$a.$ i= \ $b.$ j=		1 A	2 G	3 A	4 C
1 A					
2 A					
3 T					
4 C					

How to calculate the entries

The rules:

1. initialization:
   - 0th row and 0th column: set to 0
   - remaining rows and columns correspond to nucleotides

How to calculate the entries

The rules:

1. initialization:
   - 0th row and 0th column: set to 0
   - remaining rows and columns correspond to nucleotides
2. iteratively calculate the score $H(i,j)$ of the optimal alignment with a_i and b_j at the end for the field (i,j), and denote the direction from where the optimal alignment comes, according to:

$$H(i,j) = \max \left\{ \begin{array}{lll} 0 & & \\ H(i-1,j-1) + s(i,j) & \text{(mis-)match} & \searrow \\ H(i-1,j) + w & \text{gap in seqB} & \downarrow \\ H(i,j-1) + w & \text{gap in seqA} & \rightarrow \end{array} \right\}$$

where:
$s(i,j) = 3$ if $a_i = b_j$ (match)
$s(i,j) = -1$ if $a_i \neq b_j$ (mismatch)
$w = -2$ (gap penalty)

How to calculate the entries

The rules:

1. initialization:
   - 0th row and 0th column: set to 0
   - remaining rows and columns correspond to nucleotides
2. iteratively calculate the score $H(i,j)$ of the optimal alignment with a_i and b_j at the end for the field (i,j), and denote the direction from where the optimal alignment comes, according to:

$$H(i,j) = \max \begin{cases} 0 & & \\ H(i-1,j-1) + s(i,j) & \text{(mis-)match} & \searrow \\ H(i-1,j) + w & \text{gap in seqB} & \downarrow \\ H(i,j-1) + w & \text{gap in seqA} & \rightarrow \end{cases}$$

   where:
   $s(i,j) = 3$ if $a_i = b_j$ (match)
   $s(i,j) = -1$ if $a_i \neq b_j$ (mismatch)
   $w = -2$ (gap penalty)
3. start from the highest number (= score of the best local alignment) and walk backwards until a 0 is reached.

The Smith-Waterman algorithm: Steps 1 and 2

Task: Align seqA=AATC and seqB=AGAC.

$$H(i,j) = \max \begin{cases} 0 & & \\ H(i-1,j-1) + s(i,j) & \text{(mis-)match} & \searrow \\ H(i-1,j) + w & \text{gap in seqB} & \downarrow \\ H(i,j-1) + w & \text{gap in seqA} & \rightarrow \end{cases}$$

where:

$s(i,j) = 3$ if $a_i = b_j$; $s(i,j) = -1$ if $a_i \neq b_j$; $w = -2$

$a.$ \ $b.$ (j= / i=)	0	1 A	2 G	3 A	4 C
0	0	0	0	0	0
1 A	0				
2 A	0				
3 T	0				
4 C	0				

The Smith-Waterman algorithm: Step 3

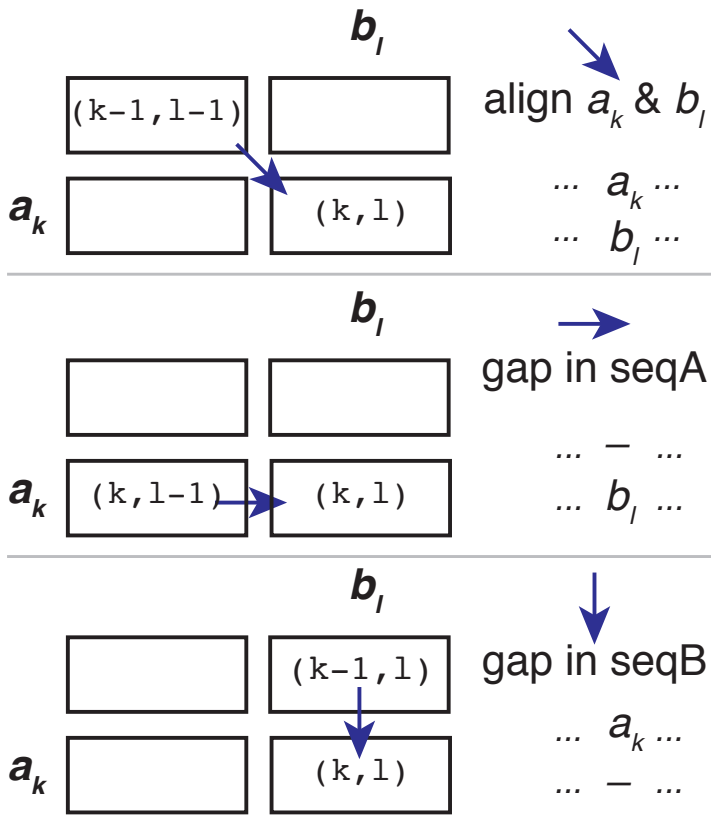
Task: Align seqA=AATC and seqB=AGAC.

a. \ b. (j= / i=)	0	1 A	2 G	3 A	4 C
0	0	0	0	0	0
1 A	0	3	1	3	1
2 A	0	3	2	4	2
3 T	0	1	2	2	3
4 C	0	0	0	1	5

The Smith-Waterman algorithm: Step 3

Task: Align seqA=AATC and seqB=AGAC.

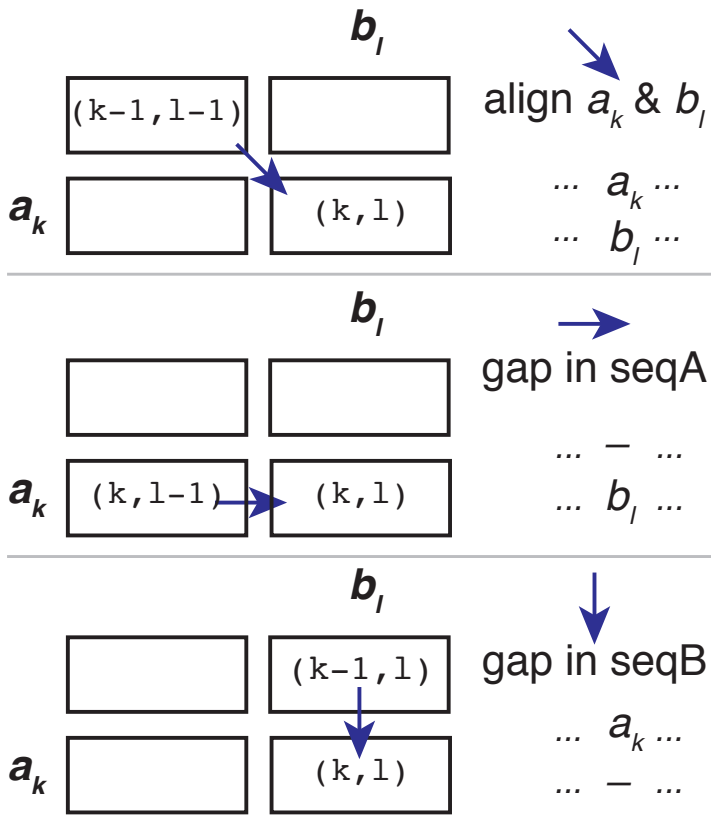
a. \ b. / i= \ j=	0	1 A	2 G	3 A	4 C
0	0	0	0	0	0
1 A	0	3	1	3	1
2 A	0	3	2	4	2
3 T	0	1	2	2	3
4 C	0	0	0	1	5



The Smith-Waterman algorithm: Step 3

Task: Align seqA=AATC and seqB=AGAC.

$a.$ \ $b.$ / i= \ j=	0	1 A	2 G	3 A	4 C
0	**0**	0	0	0	0
1 A	0	**3**	**1**	3	1
2 A	0	3	2	**4**	2
3 T	0	1	2	**2**	3
4 C	0	0	0	1	**5**



Optimal alignment:

A-ATC
AGA-C, Score = 5

The Smith-Waterman algorithm

Pros:

- fast in comparison to the brute force approach: $\mathbf{m} \times \mathbf{n}$ steps (in our example 16 steps instead of 321)
- finds the optimal local alignment (or one of the optimal alignments if there is more than one alignment with the same (highest) score)

Cons:

- Smith-Waterman only for local alignments
  Needleman-Wunsch algorithm also does global alignments
- only pairwise alignment possible
- still too slow for scanning against big libraries

The Needleman-Wunsch algorithm

The aim: Find the best **global** alignment of two sequences of length $\mathfrak{m}$ and $\mathfrak{n}$. The algorithm works analogously to Smith-Waterman with slightly changed **rules**:

1. Initialise the 0th row and 0th column according to: $\mathrm{H}(0, \mathfrak{j}) = \mathfrak{j} \times w$ and $\mathrm{H}(\mathfrak{i}, 0) = \mathfrak{i} \times w$

The Needleman-Wunsch algorithm

The aim: Find the best **global** alignment of two sequences of length m and n. The algorithm works analogously to Smith-Waterman with slightly changed **rules**:

1. Initialise the 0th row and 0th column according to: $H(0, j) = j \times w$ and $H(i, 0) = i \times w$
2. iteratively calculate the score $H(i, j)$ of the optimal alignment with a_i and b_j at the end for the field (i, j), and denote the direction from where the optimal alignment comes, according to:

$$H(i,j) = \max \left\{ \begin{array}{lll} H(i-1, j-1) + s(i,j) & \text{(mis-)match} & \searrow \\ H(i-1, j) + w & \text{gap in seqB} & \downarrow \\ H(i, j-1) + w & \text{gap in seqA} & \rightarrow \end{array} \right\}$$

where:
$s(i, j) = 3$ if $a_i = b_j$ (match)
$s(i, j) = -1$ if $a_i \neq b_j$ (mismatch)
$w = -2$ (gap penalty)

The Needleman-Wunsch algorithm

The aim: Find the best **global** alignment of two sequences of length m and n. The algorithm works analogously to Smith-Waterman with slightly changed **rules**:

1. Initialise the 0th row and 0th column according to: $H(0, j) = j \times w$ and $H(i, 0) = i \times w$
2. iteratively calculate the score $H(i, j)$ of the optimal alignment with a_i and b_j at the end for the field (i, j), and denote the direction from where the optimal alignment comes, according to:

$$H(i,j) = \max \left\{ \begin{array}{lll} H(i-1, j-1) + s(i,j) & \text{(mis-)match} & \searrow \\ H(i-1, j) + w & \text{gap in seqB} & \downarrow \\ H(i, j-1) + w & \text{gap in seqA} & \rightarrow \end{array} \right\}$$

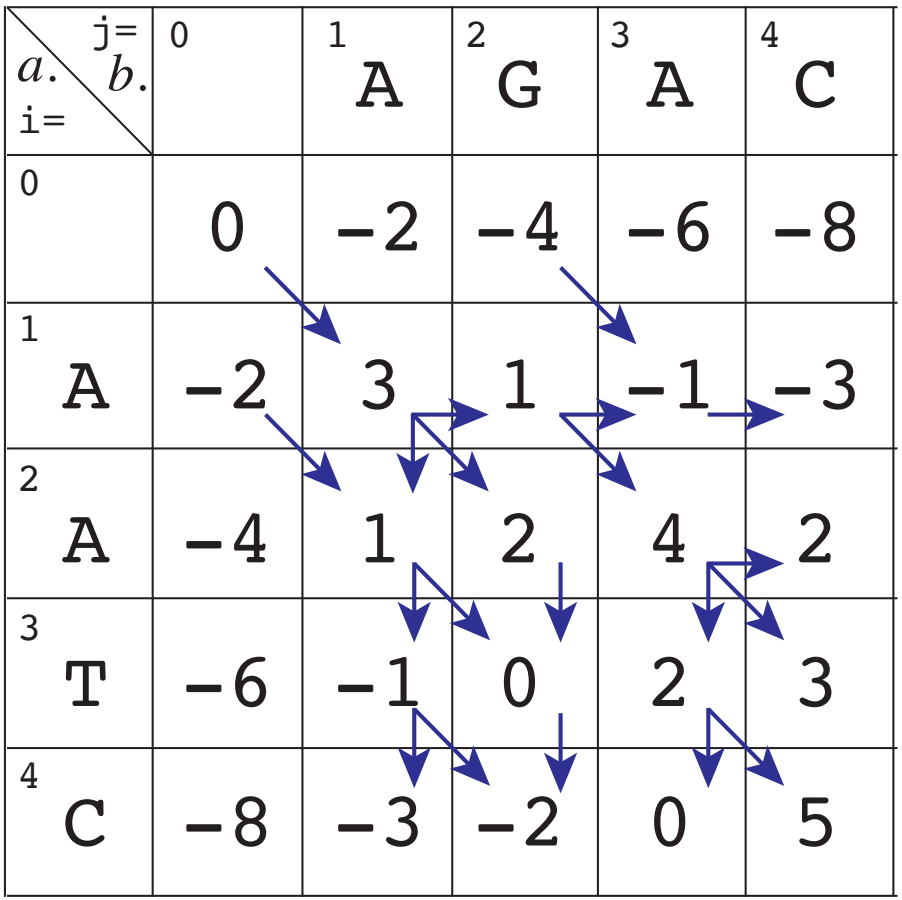
   where:
   $s(i, j) = 3$ if $a_i = b_j$ (match)
   $s(i, j) = -1$ if $a_i \neq b_j$ (mismatch)
   $w = -2$ (gap penalty)
3. start from the bottom right field (m, n) and follow the path up to the top left field $(0, 0)$.

Needleman-Wunsch: an example

Task: Find the best global alignment of `AATC` and `AGAC`

Needleman-Wunsch: an example

Task: Find the best global alignment of `AATC` and `AGAC`



Needleman-Wunsch: an example

Task: Find the best global alignment of `AATC` and `AGAC`

$a.$ i= \ $b.$ j=	0	1 A	2 G	3 A	4 C
0	**0**	−2	−4	−6	−8
1 A	−2	**3**	**1**	−1	−3
2 A	−4	1	2	**4**	2
3 T	−6	−1	0	**2**	3
4 C	−8	−3	−2	0	**5**

BLAST.

Heuristic approaches: Example 1

You want to know where the following sentences come from:

But soft! What light through yonder window breaks?
It is the east, and Juliet is the sun.

Heuristic approaches: Example 1

You want to know where the following sentences come from:

But soft! What light through yonder window breaks?
It is the east, and Juliet is the sun.

How would you search for it?

Heuristic approaches: Example 1

You want to know where the following sentences come from:

But soft! What light through yonder window breaks?
It is the east, and Juliet is the sun.

How would you search for it?

1. search for a substring, e.g. *Juliet* to reduce the number of books
2. look at the two neighbouring words, give a score to the matching sentences
3. continue only with those matches with high enough scores

Heuristic approaches: Example 1

Original quote:

> *But soft! What light through yonder window breaks?*
> *It is the east, and Juliet is the sun.*

What if a friend told you these sentences but you remember this:

> *But soft! What light through yonder window breaks?*
> *It is the* **west**, *and Juliet is the* **moon**.

How would you find the original sentence?

Heuristic approaches: Example 1

Original quote:

> *But soft! What light through yonder window breaks?*
> *It is the east, and Juliet is the sun.*

What if a friend told you these sentences but you remember this:

> *But soft! What light through yonder window breaks?*
> *It is the* **west**, *and Juliet is the* **moon**.

How would you find the original sentence?

1. search for a substring, e.g. *Juliet* to reduce the number of books
2. look at the two neighbouring words, give a score to the matching sentences
3. *define the score to give matching outputs accordingly*

Heuristic approaches: Example 2

Imagine you are a medical doctor. Someone comes to you with a disease. After many tests you still have not found the cause of the disease. You then take a blood sample and sequence all virus and bacteria in the blood. You will get many different sequences, some of them several times. One sequence is highly abundant:

ATGACAGTGACGGGGACATGGAGGAATTATCAACAATGGTGGATATGGGGAATCTTAGGCTTTTGG
ATGTTAATGATTTGTAATGGCTTGTGGGTCACAGTCTACTATGGGGTACCTGTGTGGAAAGAAGCAA
AAACTACTCTATTTTGTGCCTCAGATGCTAAATCATATGAGAAAGAGGTGCATAATGTCTGGGCTACA
CATGCCTGTGTACCCACAGACCCCAACCCACAAGAATTGGTTTTGGAAAATGTAACAGAAAATTTTA
ACATGTGGAAAAATGACATGGTAGATCAGATGCATGAAGATATAATCAGTTTATGGGATCAAAGCCTC
AAGCCATGTGTAAAGTTGACCCCGCTCTGTGTCACTCTAAACTGTAGCGATGCAAAGGTAAATGTAA
ATGATACCTATAATGGAACAAGGGAAGAAATAAAAAATTGCTCTTTCAATGCGACCACAGAATTAAGAG
ATAAGAAAAGGAGAGAATATGCACTCTTTTATAGACTTGATATAGTACCACTTAGTGGGGAGGGTAAT
AACAACAGTGAATATAGATTAATAAACTGTAATACCTCAGTCATAACACAAGCCTGTCCAAAGGTCAC
TTTTGACCCAATTCCTATACATTATTGTGCTCCAGCTGGTTATGCGATTCTAAAGTGTAATAATAAGAC
ATTCAATGGCACAGGACCATGCAATAATGTCAGCACAGTACAATGTACACATGGAATTAAGCCAGTA
GTTTCAACTCAACTATTGTTAAATGGTAGCCTAGCAGAAGAAGAGATAATAATTAGATCTGAAAACCT
GACAGACAATGTCAAAACAATAATAGTACATCTCAATGAACCTGTAGAGATTAATTGTACAAGACCCA

Heuristic approaches: BLAST

Strategy to find the source of the sequence:

- align the query sequence against all known sequences from genbank [genbank, 2016]
  - collection of DNA sequences
  - open source
  - up to 15 August 2018: 208 831 050 sequences
- report the *best* alignment

Heuristic approaches: BLAST

Strategy to find the source of the sequence:

- align the query sequence against all known sequences from genbank [genbank, 2016]
  - collection of DNA sequences
  - open source
  - up to 15 August 2018: 208 831 050 sequences
- report the *best* alignment
- **problem:** if using Smith-Waterman: $\sum_{i=0}^{208831050} m \times n_i$ operations must be performed

Heuristic approaches: BLAST

Strategy to find the source of the sequence:

- align the query sequence against all known sequences from genbank [genbank, 2016]
  - collection of DNA sequences
  - open source
  - up to 15 August 2018: 208 831 050 sequences
- report the *best* alignment
- **problem:** if using Smith-Waterman: $\sum_{i=0}^{208831050} m \times n_i$ operations must be performed
- **solution:** heuristic word methods: BLAST (**B**asic **L**ocal **A**lignment **S**earch **T**ool, [BLAST, 2016])
  - first paper [Altschul et al., 1990] cited 50 917 times, improvements for gapped sequences [Altschul et al., 1997] cited 46 150 times (01 Oct 2018)

The BLAST algorithm

1. split the query sequence into subsequences of length k

Example: `AATCAG`

```
AATCAG
AAT
 ATC
  TCA
   CAG
```

The BLAST algorithm

1. split the query sequence into subsequences of length k
2. search these k-letter words in the database sequences, similar words are allowed but scored, e.g. match +5, mismatch -3

Example: AATCAG

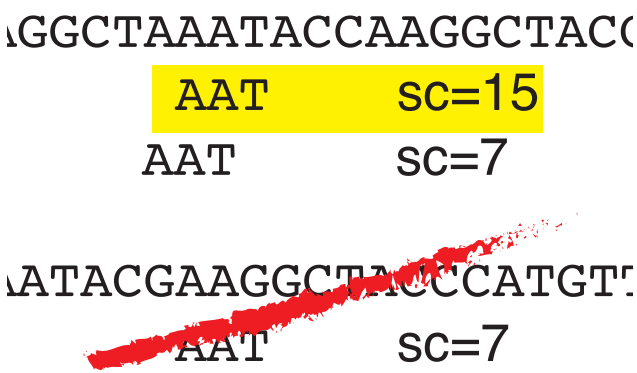
```
AGGCTAAATACCAAGGCTACC
   AAT      sc=15
  AAT       sc=7

AATACGAAGGCTACCCATGTT
   AAT      sc=7
```

The BLAST algorithm

1. split the query sequence into subsequences of length k
2. search these k-letter words in the database sequences, similar words are allowed but scored, e.g. match +5, mismatch -3
3. keep only the sequences with the highest scores

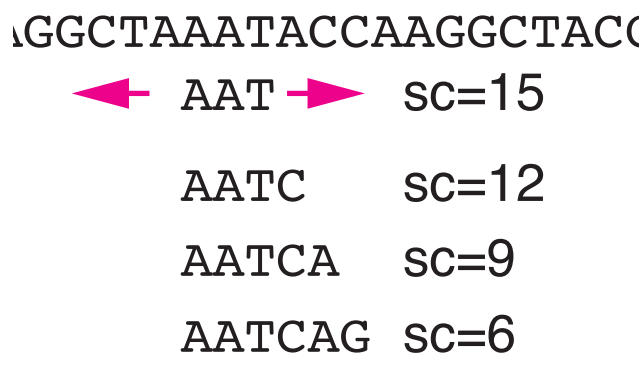
Example: AATCAG



The BLAST algorithm

1. split the query sequence into subsequences of length k
2. search these k-letter words in the database sequences, similar words are allowed but scored, e.g. match +5, mismatch -3
3. keep only the sequences with the highest scores
4. expand the k-letter word to the right and left, note the scores

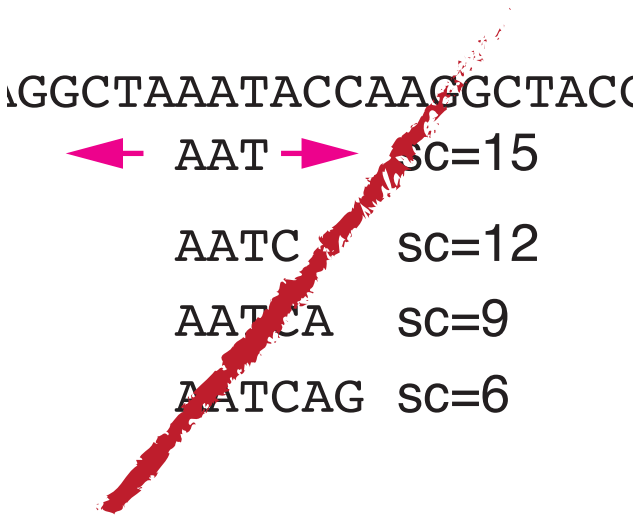
Example: `AATCAG`



The BLAST algorithm

1. split the query sequence into subsequences of length k
2. search these k-letter words in the database sequences, similar words are allowed but scored, e.g. match +5, mismatch -3
3. keep only the sequences with the highest scores
4. expand the k-letter word to the right and left, note the scores
5. stop if the score drops below a certain threshold

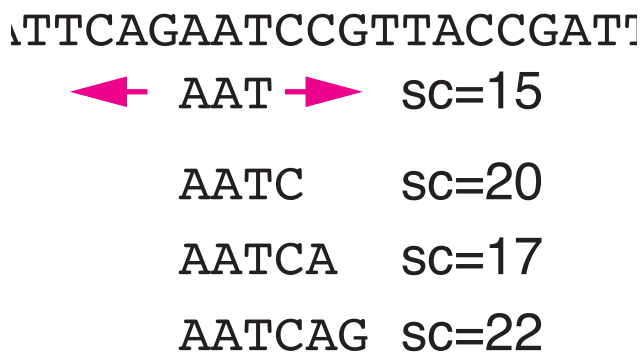
Example: AATCAG



The BLAST algorithm

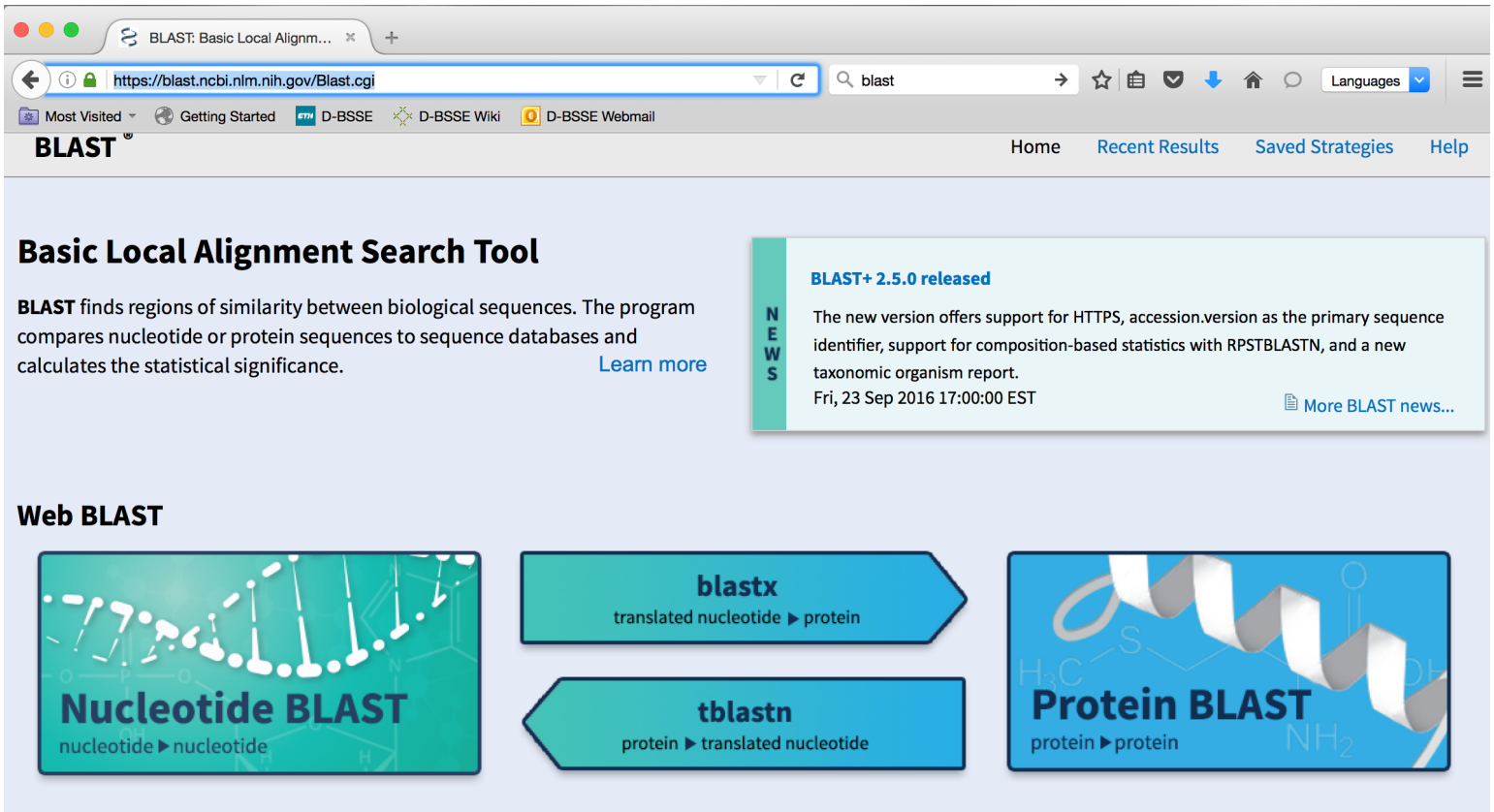
1. split the query sequence into subsequences of length k
2. search these k-letter words in the database sequences, similar words are allowed but scored, e.g. match +5, mismatch -3
3. keep only the sequences with the highest scores
4. expand the k-letter word to the right and left, note the scores
5. stop if the score drops below a certain threshold
6. keep only the pairwise alignments that are above the threshold
7. report these database sequences

Example: `AATCAG`



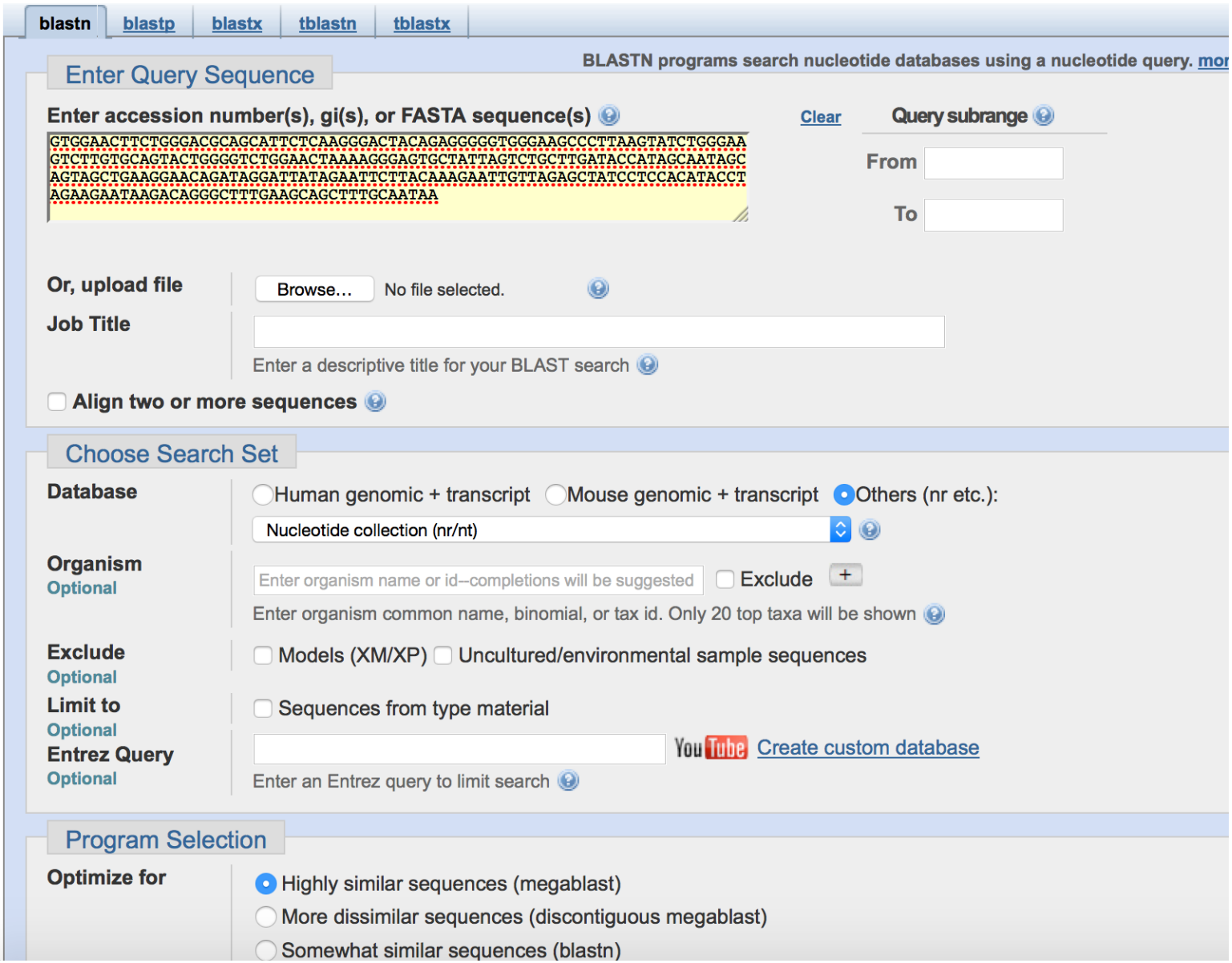
Example: Unknown sequence

Let's get back to our medical diagnostics problem. We are now performing a BLAST search with https://blast.ncbi.nlm.nih.gov/Blast.cgi

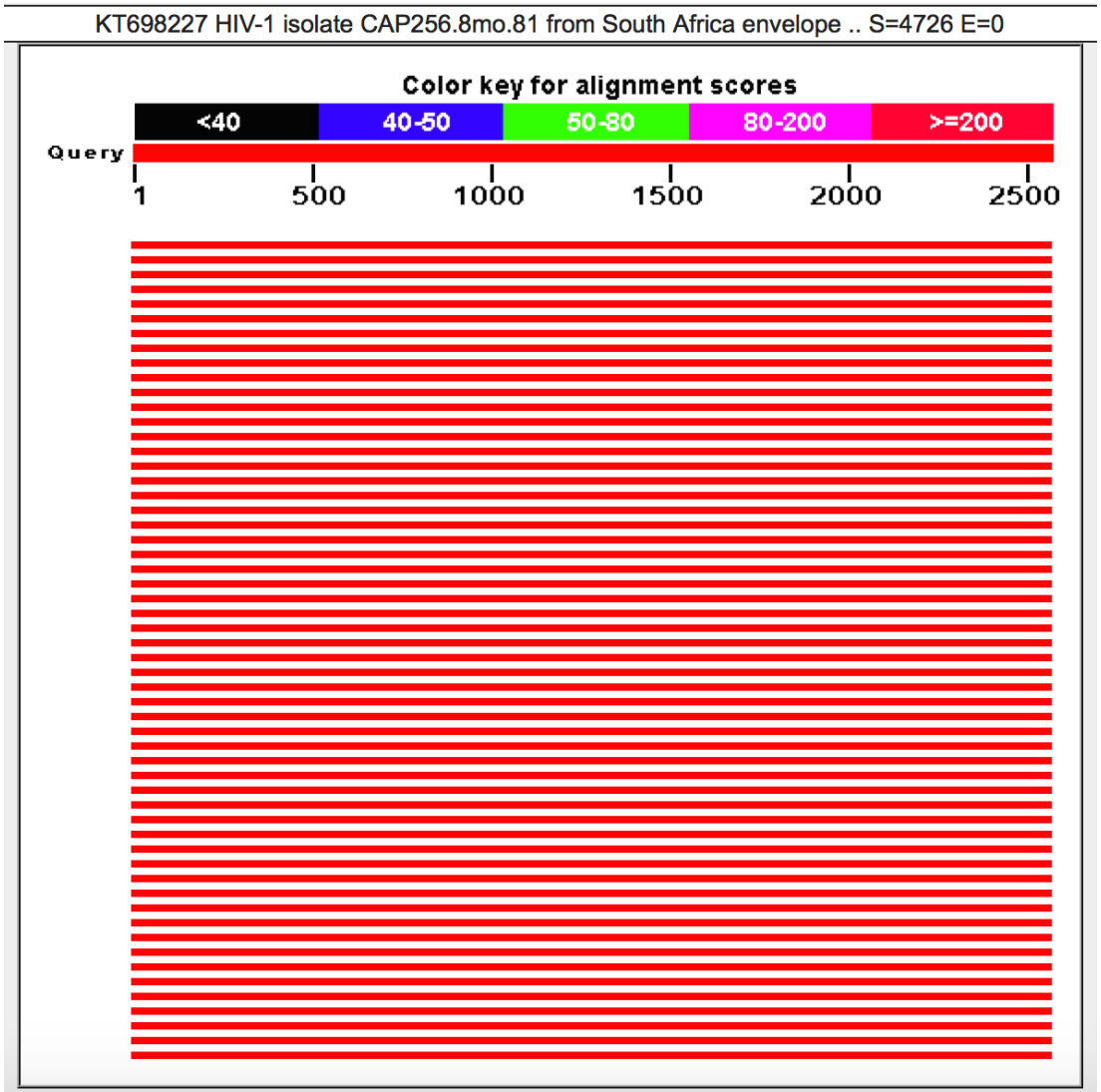


Example: Unknown sequence

Perform a nucleotide BLAST:



Example: Unknown sequence



- Sequence length: 2559
- run time: 19 seconds
- Sequence: Env from HIV-1*
- all closely matching strains shown

* [Doria-Rose et al., 2014]

Further applications

- Imagine you found a new protein in mice. Does this protein also exist in humans or other mammals?

Further applications

- Imagine you found a new protein in mice. Does this protein also exist in humans or other mammals?
- You found a certain protein in an organism. Are there other related proteins in that or other organisms?

Further applications

- Imagine you found a new protein in mice. Does this protein also exist in humans or other mammals?
- You found a certain protein in an organism. Are there other related proteins in that or other organisms?
- Detection of rare viral strains by metagenomics in patients with long history of sickness, e.g. [Lewandowska et al., 2015]

Pros and Cons

Pros:

- up to 50-100 times faster than direct alignment (e.g. with the Smith-Waterman algorithm)
- allows searches for exact matches but also for similarity up to a pre-defined degree

Cons:

- does not guarantee the optimal pairwise alignments of the query and database sequences
- **Limitation:** you can only find a match when the gene/sequence is available in the database

Multiple sequence alignment.

Multiple sequence alignment (MSA)

So far, we only looked at pairwise alignments. However, very often we want to compare different species, proteins from different organisms, etc. What we need is a way to align multiple sequences simultaneously in order to:

- reconstruct the evolutionary history of individuals in a phylogeny
- assess the sequence conservation of proteins

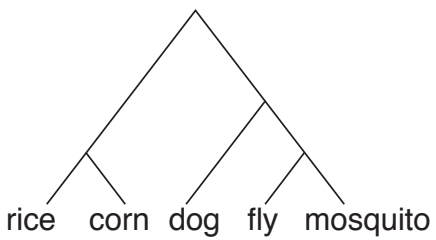
Computationally very demanding, mostly heuristic algorithms

Multiple sequence alignment (MSA)

So far, we only looked at pairwise alignments. However, very often we want to compare different species, proteins from different organisms, etc. What we need is a way to align multiple sequences simultaneously in order to:

- reconstruct the evolutionary history of individuals in a phylogeny
- assess the sequence conservation of proteins

Computationally very demanding, mostly heuristic algorithms



Triosephosphate isomerase

```
rice      NGTTDQVDKIVKILNEGQIASTDVVEVVVSPPYVFLPVVKSQLRPEIQVAAQNCW
corn      NGTADQVDKIVKILNEGQIASTDVTEVVVSPPYVFLPVVKSQLRPEIQVAAQNCW
dog       NGTKDQVDKIVKILNEGQIASTDVVEVVVSPPYVFLPVVKSQLRPEIQVAAQNCW
fly       NGTKASIDKIVKILNEGQIAST_VVEVVVSPPYVFLPVVRSQLRPEICVAAQNCW
mosquito NGDKASIADLCKVLTTGPLNAD__TEVVVGCPAPYLTLARSQLPDSVCVAAQNCY
```

In the MSA each column represents amino acids or nucleotides that have descended from the same position in the sequence as in the common ancestor. We seek the MSA that correctly represents the evolutionary history of a set of sequences.

Ad hoc approach: pairwise alignment against a reference strain

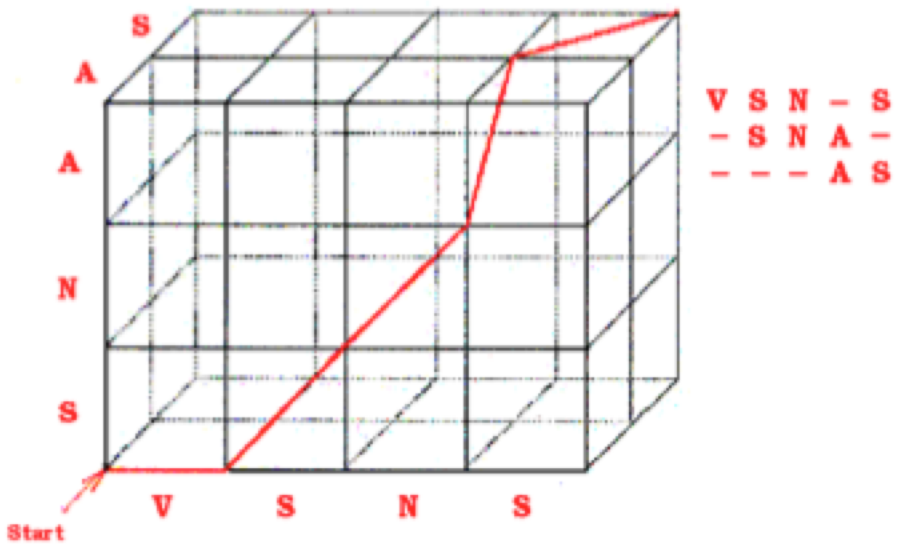
One way to perform multiple sequence alignment is to define a reference strain for the genome and pairwise align all sequences with the reference strain.

- Example: The reference strain for HIV-1 is HXB2 see [Korber et al., 2014]
- Advantage: position numbering is the same for each sequence
- Disadvantage: only possible when one knows which species the sequences come from

Strategies for multiple sequence alignment: arrays

Another approach extends the Smith-Waterman algorithm into more dimensions:

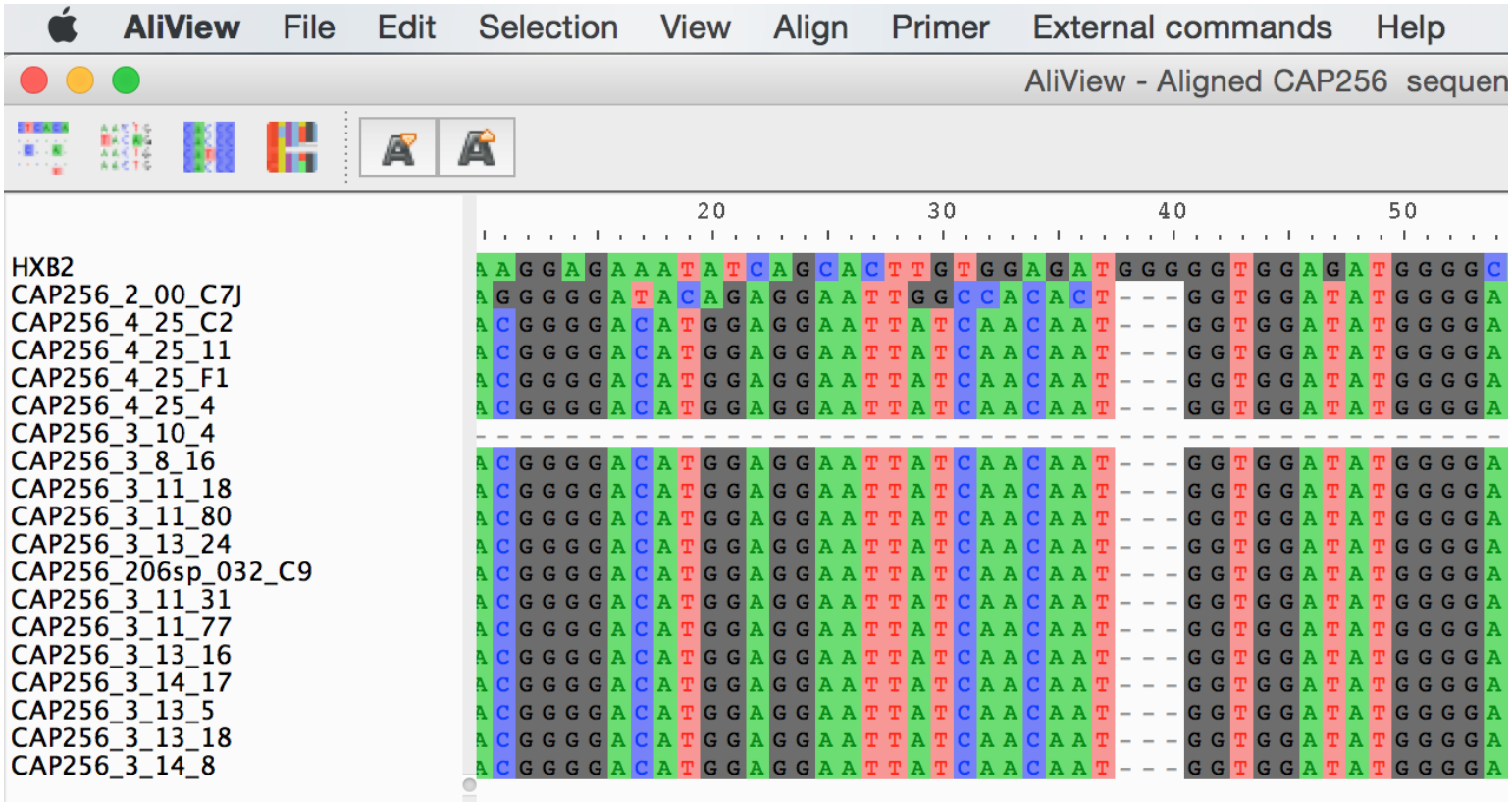
- extremely slow: k sequences of length m required m^k steps



[http://bioinfo3d.cs.tau.ac.il/Education/CS0405/ta2.ppt,]

Other alignment programs

There are different algorithms for MSA which are based on dynamic programming: ClustalW, Muscle, MAlign, etc. These are often implemented in alignment viewers, e.g. AliView: [Larsson, 2014]



data from [Doria-Rose et al., 2014]

What you learned in lecture 2

- Pairwise sequence alignments
  - brute force alignments
  - dot-matrix
  - dynamic programming: Smith-Waterman
- BLAST: heuristic approach
- Multiple sequence alignment

Further information

- Further information:
  - referenced papers
  - mathematical background: [Ewens and Grant, 2005]
  - Bioinformatics lecture by Professors Stelling and Beerenwinkel

Alignment, BLAST Questions

- What are the weaknesses and strengths of the different alignment methods (dot-matrix method, Smith-Waterman, Needleman-Wunsch, BLAST)?
- With which alignment methods do you get an optimal alignment?
- Do you obtain the same alignments when using different scoring schemes in the Smith-Waterman and Needleman-Wunsch algorithms?

References I

- Altschul, S. F., GISH, W., Miller, W., MYERS, E. W., and Lipman, D. J. (1990). Basic Local Alignment Search Tool. *Journal Of Molecular Biology*, 215(3):403–410.
- Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J. H., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402.
- BLAST (2016). https://blast.ncbi.nlm.nih.gov/Blast.cgi.
- Doria-Rose, N. A., Schramm, C. A., Gorman, J., Moore, P. L., Bhiman, J. N., DeKosky, B. J., Ernandes, M. J., Georgiev, I. S., Kim, H. J., Pancera, M., Staupe, R. P., Altae-Tran, H. R., Bailer, R. T., Crooks, E. T., Cupo, A., Druz, A., Garrett, N. J., Hoi, K. H., Kong, R., Louder, M. K., Longo, N. S., McKee, K., Nonyane, M., O'Dell, S., Roark, R. S., Rudicell, R. S., Schmidt, S. D., Sheward, D. J., Soto, C., Wibmer, C. K., Yang, Y., Zhang, Z., Program, N. C. S., Mullikin, J. C., Binley, J. M., Sanders, R. W., Wilson, I. A., Moore, J. P., Ward, A. B., Georgiou, G., Williamson, C., Karim, S. S. A., Morris, L., Kwong, P. D., Shapiro, L., and Mascola, J. R. (2014). Developmental pathway for potent V1V2- directed HIV-neutralizing antibodies. *Nature*, pages 1–8.
- Ewens, W. and Grant, G. (2005). *Statistical Methods in Bioinformatics – An Introduction*. Springer.
- genbank (2016). https://www.ncbi.nlm.nih.gov/genbank/.
- http://bioinfo3d.cs.tau.ac.il/Education/CS0405/ta2.ppt. http://bioinfo3d.cs.tau.ac.il/Education/CS0405/ta2.ppt.
- Korber, B. T., Foley, B. T., Kuiken, C. L., Pillai, S. K., and Joseph G. Sodroski, J. G. (2014). https://www.hiv.lanl.gov/content/sequence/HIV/REVIEWS/HXB2.html.
- Larsson, A. (2014). AliView: a fast and lightweight alignment viewer and editor for large datasets. *Bioinformatics*, 30(22):3276–3278.
- Lewandowska, D. W., Zagordi, O., Zbinden, A., Schuurmans, M. M., Schreiber, P., Geissberger, F.-D., Huder, J. B., Böni, J., Benden, C., Mueller, N. J., Trkola, A., and Huber, M. (2015). Unbiased metagenomic sequencing complements specific routine diagnostic methods and increases chances to detect rare viral strains. *Diagnostic Microbiology and Infectious Disease*, 83(2):133–138.
- Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal Of Molecular Biology*, 48(3):443–453.
- Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal Of Molecular Biology*, 147(1):195–197.